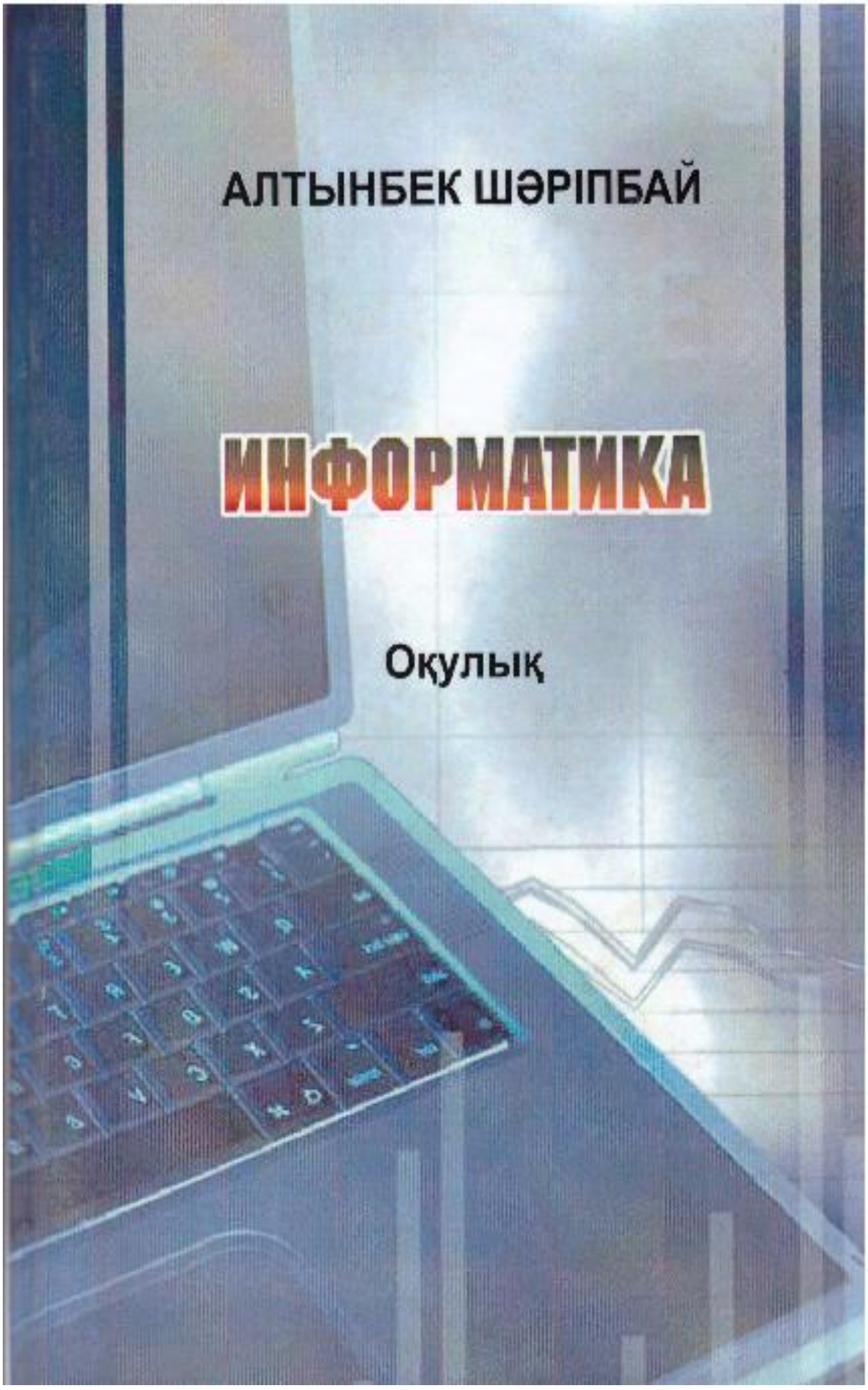

АЛТЫНБЕК ШӘРІПБАЙ
ИНФОРМАТИКА
Оқулық

АЛТЫНБЕК ШӘРІПБАЙ

ИНФОРМАТИКА

(Оқулық)

Алматы, 2015
Эверо

ӘОЖ 004.4(075.8)
КБЖ 32.973.26–018.1я73
Ш–25

Пікір берушілер:
Адамов А.А. – Л.Н.Гумилев атындағы ЕҰУ Математикалық және компьютерлік моделдеу кафедрасының меңгерушісі**,** т.ғ.д., профессор;
Данаев Н.Т. – Әл-Фараби атындағы ҚазҰУ Математика және механика Институтының директоры, ф.м.-ғ.д., ҚР ҰҒЫ корр-мүшесі, ҰИА академигі, ҚР Мемлекеттік сыйлығының лауреаты;
Өскенбаева Р.К. – Халықаралық ақпараттық технологиялар университетінің проректоры, т.ғ.д., профессор.

Автор:
Шәріпбай А.Ә. – Л.Н.Гумилев атындағы Еуразия ұлттық университетінің, «Теориялық информатика» кафедрасының профессоры, «Жасанды интеллект» ҒЗИ директоры, т.ғ.д., ҚР Мемлекеттік сыйлығының лауреаты.

Ш–25 Шәріпбай А.Ә. Информатика – Оқулық, - Астана, 2015, - 312 б.
ISBN 978-601-7454-76-0

Оқулық қазіргі кезде ең қажетті және өте жылдам дамып келе жатқан Информатика ғылымының негізгі бөліктерін оқып білуге арналған. Онда ақпарат, деректер құрылымы, алгоритм, алгоритмдік тіл, басқару құрылымдары, компьютерлік жүйе, программалау тілі, ақпаратты сұрыптау әдістері сияқты ұғымдар қарастыралады. Оқулықтың мазмұны 5В060200-Информатика мамандығы бойынша Мемлекеттік жалпыға міндетті білім беру стандартына сәйкес жазылған. Оқу материалы кредиттік технологияны қолдануға арналған, ол үш деңгейден тұрады: модул, блок, дәріс. Бұл білімді бақылаудың төрт деңгейде жүргізуге мүмкіндік береді: дәріс деңгейінде – ағымдағы бақылау, блок деңгейінде – аралық бақылау, модул деңгейінде – белестік бақылау, оқулық деңгейінде – қорытынды бақылау. Білімді бағалау осы деңгейлерге сәйкес құрастырылған тестілер арқылы жүзеге асырылады.

Оқулықпен Информатика, Есептеу техникасы және программалық қамтым, Ақпараттық жүйелер, Автоматтандыру және басқару, Математикалық және компьютерлік моделдеу, Ақпараттық қорғау жүйелері мамандықтарының студенттері, магистранттары, докоранттары, оқытушылары, ғалымдары және информатиканы өздігінен оқып білем дегендердің барлығы пайдалана алады.

1404000000
Ш–25
ISBN 978-601-7454-76-0

ӘОЖ 004.4(075.8)
КБЖ 32.973.26–018.1я73

МАЗМҰНЫ

АЛҒЫ СӨЗ

Қазіргі кезде Информатика (Computer Science) ең қажетті және ең маңызды ғылым болды. Себебі, информатиканың нәтижелері қолданбайтын адамның интеллектуалды тірлігінің салалары жоқ деп айтуға болады. Мысалы, компьютерлік программа арқылы әртүрлі есептер шығарылады, жобалар жасалынады, шешімдер қабылданады, үдерістер басқарылады, суреттер салынады, тұжырымдар дәлелденеді, музыка орындалады, ойындар ойнатылады және тағы басқа көптеген әрекеттер атқарылады.

Информатика адам қоғамының тек материалды өндірісіне ғана емес, интеллектуалды және рухани тірлігіне де әсер етіп оларды түбегейлі түрлендіретін зат пен энергияны меңгерген революциядан кейінгі ақпаратты жинақтау, өңдеу және жіберу есептерін компьютерлік программалар арқылы автоматты шешетін революцияны байланыстырады.

Информатика адамның интеллектуалды өміріндегі әртүрлі есептерді шығару және жеке үдерістерді автоматтандыру үшін вербалды тілдерде жазылған программалар арқылы компьютерде жүзеге асырылатын ақпараттық технологияларды жасау мен зерттеу проблемаларымен айналысады.

Информатика *Hardware – Қатты қаптым* (компьютерлер және олардың құрамдас бөліктері, компьютерлік кешендер мен торлар), *Software – Жұмсақ қаптым* (алгоритмдер, программалар, программалық кешендер мен жүйелер) және *Brainware – Ақылды қаптым* (математикалық моделдер және әдістер) деген үш бөлімнен тұрады.

Ұсынып отырған оқулық информатиканың осы бөліктерінің негіздерін оқып білуге арналған, онда оларға аттас үш модул бар. Әрбір модулда оның тақырыбына сәйкес түйін сөздер беріледі, негізгі ұғымның онтологиялық (құрылымдық және мағыналық) моделі көрсетіледі, оқу материалы жазылады және онымен қоса оқытудың келесі негізгі есептері шешіледі: 1) оқу материалымен танысу (презентация) үшін теориялық ақпарат беріледі; 2) оқу материалын ұғыну (семантизация) үшін мысалдар талданады; 3)

оқу материалын бекіту (валидация) үшін тапсырмалар ұсынылады; 4) танысқан, түсінген және бекітілген оқу материалын тексеру (верификация) үшін сұрақтар қойылады; 5) алған білімге баға (рейтинг) алу үшін тесттер қарастырылады. Оқулықтың соңында тапсырмалардың, сұрақтардың және тесттердің жауаптары беріледі.

Информатика объектілері атты модулда *Ақпарат*, *Шамалар*, *Деректер құрылымы* деген блоктар қаралып, олардың ішінде мынадай дәрістер берілген: *Ақпарат* - Ақпараттың анықтамасы, Хабар және мазмұн, Ақпараттың көлемдік бірлігі және мөлшері, Ақпараттың ықтималдық бірлігі және мөлшері; *Шамалар* – Шамалардың түрлері, Сандық шамалар түрлері, Сандық амалдар және олардың қасиеттері, Саңдардың санау жүйелері, Символдық шамалар түрлері, Символдық амалдар және олардың қасиеттері, Логикалық шамалар түрлері, Логикалық амалдар және олардың қасиеттері; *Деректер құрылымы* - Деректер құрылымын сыныптау, Жиындар, Жиындардағы амалдар, Тізімдер, Жолдар, кестелер және хеш кестелер, Графтар мен дарақтар, Стектер, кезектер және дектер.

Информатиканың қатты қамтымы атты модулда *Компьютерлер*, *Компьютерде ақпаратты бейнелеу* деген блоктар қаралып, олардың ішінде мынадай дәрістер берілген: компьютердің құрамы мен құрылымы, компьютердің жұмыс істеу прінсіпі мен ырғағы, компьютерлердің буындары, дербес компьютерлер; компьютерде таңбаларды бейнелеу, компьютерде сандарды бейнелеу.

Информатиканың жұмсақ қамтымы атты модулда *Алгоритм*, *Алгоритмдік тілдер*, *Басқару құрылымдары*, *Программалау тілдері* деген блоктар қаралып, олардың ішінде мынадай дәрістер берілген: *Алгоритм* - Өңдеу ұғымы, Алгоритм ұғымы; *Алгоритмдік тілдер* – Вербалды алгоритмдік тіл, Графикалық алгоритмдік тіл; *Басқару құрылымдары* - Басқару құрылымдары және олардың мәні, Тізбектеу, Қарапайым тармақтау, Баламалы тармақталу, Көп мәнді тармақтау, Алғышартты қайталау, Соңғышартты қайталау, Параметрлік қайталау; *Программалау тілдері* – Программа және программалау тілі ұғымы, Программалау тілдерінің сыныпталуы.

Информатиканың ақылды қамтымы атты модулда *Сұрыптау әдістері* деген блок қаралып, оның ішінде мынадай дәрістер берілген: Сұрыптау ұғымы, Ендірумен сұрыптау, Таңдаумен сұрыптау, Ауыстырумен сұрыптау.

Әдебиеттер тізімінде оқулықтың оқыту материалдарын құрған кезде пайдаланылған жалпы әдебиеттік көздер көрсетілген. Олардың ішінде кең таралған монографиялар, оқулықтар, оқу құралдары, мемлекеттік стандарттар және басқалар бар. Сондықтан осы оқулық мәтінінде оларға тікелей сілтеме жасалынбаған.

Оқулықта қолданылған барлық атау терминдер Қазақстан Республикасы Үкіметінің жанындағы Республикалық терминологиялық комиссиясы мақұлдаған «Информатика және есептеу техникасы» атты қазақша-орысша, орысша-қазақша сөздігінен алынған.

Автор бағалы ескерту мен ұсыныстар жасаған оқулыққа пікір берушілер Ә.Ә. Адамовқа, Р Н.Т. Данаевқа, ваға, және оқулықты техникалық редакциялаған З.Қадеркееваға, А.Сағадиеваға өз алғысын білдіреді

Автор:
ШӘРІПБАЙ Алтынбек Әміруұлы
техника ғылымдарының докторы, профессор,
Қазақстан Республикасының Мемлекеттік сыйлығының лауреаты

I. ИНФОРМАТИКА ОБЪЕКТІЛЕРІ

I.1. Ақпарат

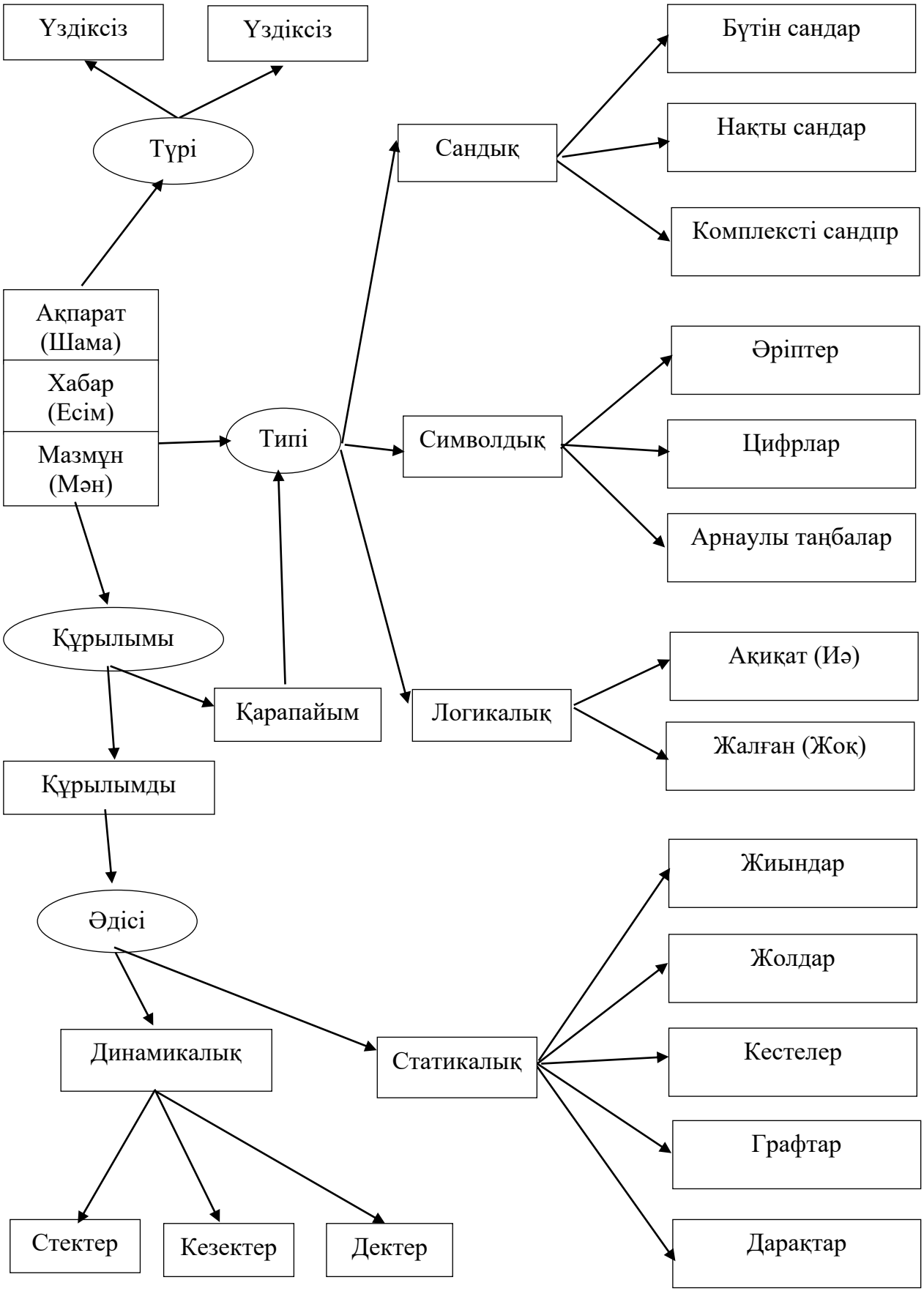
Бұл бөлімде информатика объектісі қарастырылады. Оған ақпарат (шама) жатады. Оның түрлері, типтері, құрылымдары және бейнелеу тәсілдері анықталады. Шамаларда анықталған амалдар және олардың қасиетттері беріледі. Деректер құрылымы және олардың түрлері қарастырылады. Деректер құрылымында анықталған амалдар және олардың қасиеттері беріледі. Деректер құрылымын бейнелеу тәсілдері қарастырылады.

Түйін сөздер: *ақпарат*, *хабар*, *мазмұн*, *жіберуші*, *қабылдаушы*, *байланыс арнасы*, *үздіксіз* (*аналогтық*) *ақпарат*, *дискретті* (*цифрлық*) *ақпарат*, *қатынас тілі*, *шама*, *есім*, *мән*, *символдық шама*, *әріптер*, *цифрлар*, *арнаулы таңбалар*, *сандық шама*, *бүтін сан*, *нақты сан*, *комплексті сан*, *логикалық шама*, *ақиқат*, *жалған*, *тұрақты шама*, *айнымалы шама*, *идентификатор*, *сандық амал*, *символдық амал*, *логикалық амал*, *коммутативтік*, *ассоциативтік*, *дистрибутивтік*, *ақпараттың өлшем бірлігі*, *ақпарат мөлшері*, *ақпараттың көлемдік мөлшері*, *ақпараттың ықтималдық мөлшері*, *жиын*, *жол*, *кесте*, *хеш-кете*, *тізім*, *граф*, *дарақ*, *кезек*, *стек.*

Ескерту: *ақпарат* ұғымын қазақ тіліндегі әдебиеттерде *ақпарат* деп жүр. Біз оны өзгертпей пайдаланамыз. Оның себебі осы ұғымның анықтамасы туралы оқығанда түсінікті болады.

Мақсат: *ақпарат ұғымын анықтау*; *оның түрін*, *құрылымын және типін қарау*, *шамалардың типтерімен танысу*, *амалдардың қасиеттерін білу*, *өрнектердің мәндерін анықтауды үйрену*, *ақпараттың өлшем бірлігін және мөлшерін анықтау.*

Құрылымы: Информатика объектілерінің құрамы мен құрылымы I.1-суретте көрсетілген.



I.1-сурет. Информатика объектілерінің құрамы мен құрылымы.

I.1.1. Ақпараттың анықтамасы

Ақпарат информатика ғылымының негізгі ұғымы, ол қандай-да бір нәрсе (тұлға, зат, ақиқат, оқиға, құбылыс, үдеріс) туралы оның бейнелену пішініне тәуелсіз мәлімет беру, жариялау дегенді білдіретін «*Information*» деген латын сөзінен шыққан.

Ақпарат ұғымының дәл (математикалық) анықтамасы жоқ. Егер біз оған дәл анықтама беруге тырыссақ, онда тағы да басқа анықталмаған ұғымға келеміз. Бірақ біз ақпаратты, оның дәл анықтамасын білмей–ақ, қабылдаймыз, түсінеміз, сақтаймыз, өңдейміз (түрлендіреміз) және керек болса басқа біреуге жібереміз (береміз) деп айта аламыз. Осыларды ескере отырып «ақпарат» ұғымына мынадай интуитивті (математикалық дәл емес) анықтама беруге болады.

Ақпарат – адамның санасына (қабылдауына) байланыссыз қоршаған дүниедегі материалдық немесе материалдық емес объектілер және субъктілер қасиеттерінің және қатынастарының бейнесі (белгісі).

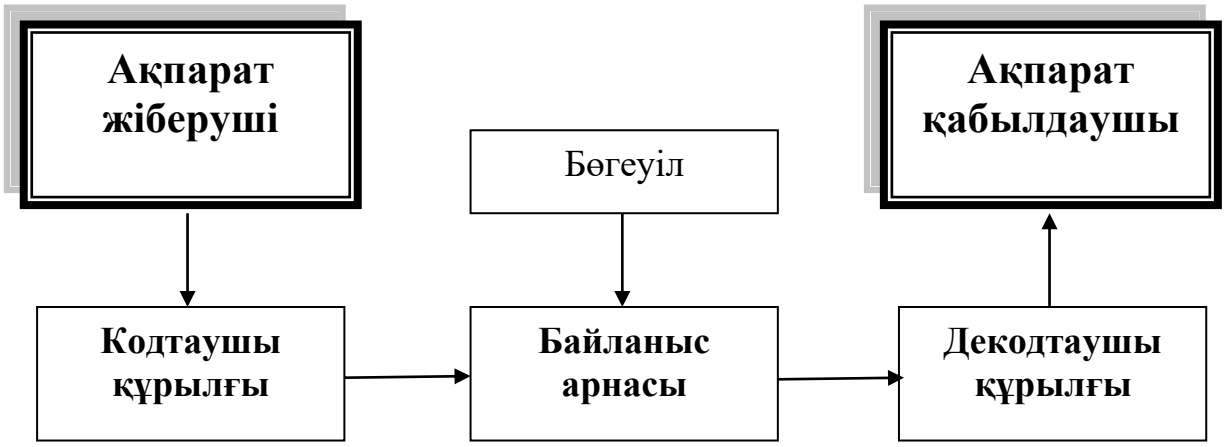
Ақпаратты бейнелеуге (белгілеуге) және түсінуге болады. Сондықтан, әрбір ақпаратта өзінің формасы (пішімі) және өзінің *мазмұны* болуы қажет.

Ақпараттың формасын *хабар* дейді. Хабар материалды энергетикалық (мысалы, жарық, дыбыс, қимыл, таңба және т.б. сияқты) түрде беріледі. Яғни, хабар белгілі бір қатынас тілінің өрнегі (сөйлемі) болады. Ондай қатынас тіліне:

- табиғи тілдер (қазақ тілі, орыс тілі, ағылшын тілі және т.б.);
- математика тілі (математикалық объектілердің қасиеттері мен қатынастарын көрсететін шартты белгілері бар өрнектер жиыны);
- әуез тілі (дыбыстық қатардағы дыбыстың қасиеттері мен қатынастарын көрсететін шартты белгілері бар өрнектер жиыны);
- мылқаулар мен кереңдер тілі (бет пен қолдың шартты қимылдары бар өрнектер жиыны) және т.б. жатады.

Жалпы, хабар туралы сөз қозғағанда оның *жіберушісі* және *қабылдаушысы* болатындығын естен шығармау керек. Олар тірі организмдер (адамдар, жануарлар, құстар, балықтар, насекомдар және т.б) және тірі емес (техникалық) құрылғылар болуы мүмкін. Мысалы, егер қабылдағыш адам болса, онда ол хабарды өзінің сезім органдары арқылы қабылдайды. Ал, техникалық құрылғыларда ол үшін әр түрлі құралдар қолданылады.

Хабар жіберушіден қабылдаушыға, *байланыс арнасы* деген, белгілі орта арқылы жіберіледі. Мысалы, дыбыстық хабардың осындай ортасы ретінде дыбыс толқындары тарай алатын ауаны алуға болады, ал жазбаша хабардың байланыс арнасы мәтін жазылған қағаз болады. Хабарды жіберу сұлбасы I.1.1- суретте көрсетілген.



I.1.1–сурет. Хабарды жіберу сұлбасы

Хабарды жіберу (қабылдау) кезінде жіберушінің (қабылдаушының) жағдайы уақытқа байланысты өзгереді. Сондықтан, хабарды жіберушінің (қабылдағыштың) материалдық–энергетикалық жағдайын сипаттайтын уақытқа байланысты $\boldsymbol{x}(\boldsymbol{t})$ функциясы (мұнда $\boldsymbol{t}$ – уақыт) деп қарастыруға болады. Бұл $\boldsymbol{x}(\boldsymbol{t})$ функциясы үздіксіз болуы да, дискретті (үздікті) болуы да мүмкін. Соған байланысты *үздіксіз* (*аналогтық*) *ақпарат* немесе *дискретті* (*цифрлы*) *ақпарат* болады. Мысалы, үздіксіз ақпаратқа уақытқа байланысты өзгеріп отыратын ортаның температурасын, ал

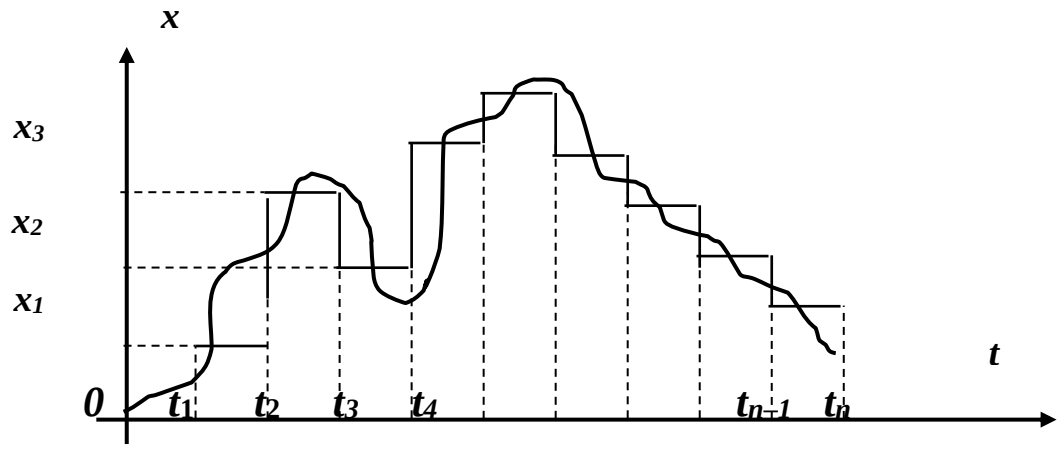
дискретті ақпаратқа хабардың белгілі бір қатынас тілінің таңбалары арқылы берілуін (жоғарыдағы мысалдар) жатқызуға болады.

Кез келген үздіксіз ақпаратты дискретті ақпаратқа айналдыруға болады, ол *дискреттеу* деп аталады. Функцияны дискреттеу үшін оның шексіз көп мәндерінің ішінен саны шектеулі және қалған мәндерді жуықтап сипаттай алатын мәндерді алады. Осындай бір әдістің мағынасы мынада:

1) функция графигінің абцисса өсі (функцияның анықталу облысы) саны шектеулі $t_1, t_2,..., t_n$ нүктелер арқылы ұзындықтары бірдей кесінділерге бөлінеді және әрбір кесіндіде функцияның мәні тұрақты, мысалы, осы кесіндідегі орта мәнге тең болады деп есептелінеді;

2) әрбір кесіндідегі функцияның мәнін функция графигінің ордината өсіне (функцияның өзгеру облысы) проекциялап $x_1, x_2,..., x_n$ нүктелерін табу керек.

Осылай табылған $x_1, x_2,..., x_n$ нүктелері үздіксіз $x(t)$ функциясының дискретті жуықтап бейнеленуі болып есептелінеді. Оның дәлдігін анықталу облысындағы кесінділерді кішірейте отырып қажетті талапты қанағаттандырғанша шексіз жақсарта беруге болады. Үздіксіз $x(t)$ функциясын дискреттеу I.1.2-суретте көрсетілген.



I.1.2-сурет. Үздіксіз $x(t)$ функциясын дискреттеу

Үздіксіз ақпаратты дискреттеу мүмкіншілігі оның хабарын қандайда бір тілдің әліпби таңбалары арқылы бейнелеуге мүмкіндік

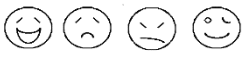
береді. Бұл ақпаратты компьютер арқылы *қабылдау*, *бейнелеу*, *сақтау*, *өңдеу* және *жіберу* үшін аса маңызды.

I.1.1. Мысалдар:

1. Суық қыс – жыл мезгілінің қасиетін сипаттайтын қазақ тілінің сөйлемі.
2. Земля круглая – жер планетасының қасиетін көрсететін орыс тілінің сөйлемі.
3. A+B = B+A – қосу амалының қасиетін көрсететін математикалық тілдің сөйлемі (өрнегі).
4. ☺ – мимика тіліндегі адамның қуанған жағдайы.
5. ➲ – жолда жүру ережесінің белгісі.
6. *Егер* B *онда* A1, *әйтпесе* A2 – белгілі B шартына байланысты берілген A1 әрекетін немесе A2 әрекетін орындау туралы жасанды тілдегі нұсқау, ал *Егер*, *онда*, *әйтпесе* – осы жасанды тілдің сөздері.
7. Телефондық байланыс: ақпарат жіберуші және қабылдаушы – абоненттер. Кодтаушы құрылғы – микрофон, дыбыстық тербелісті электр сигналына түрлендіреді. Байланыс арнасы – кәбіл. Декодтаушы құрылғы – құлаққап (динамик). Бөгеуіл байланыс арнасына әсер етеді де, байланыстың сапасын төмендетеді.

I.1.1. Тапсырмалар:

Мына сөздердің қандай қатынас тіліне жататындығын анықта

1. Астана.
2. $a + (b+c) = (a+ b) + c.$
3. 

Көмек:

1. Қала атауын білдіретін қазақ тілінің сөйлемі.
2. Қосу амалының қасиетін көрсететін математикалық өрнек.
3. Адамның көңіл күйін көрсететін мимикалық белгі.

I.1.1. Сұрақтар:

1. Ақпарат деген не?
2. Ақпарат ұғымының дәл (математикалық) анықтамасы бар ма?

3. Әрбір ақпараттың несі болуы керек?
4. Хабар деген не?
5. Хабар қандай түрде беріле алады?
6. Қатынас тілдерінің қандай түрлері бар?
7. Программалау тілдерін қатынас тілдеріне жатқызуға бола ма?
8. Хабарды жіберуші мен алушының арасында не болады?
9. Хабарды жіберу (қабылдау) кезінде жіберушінің (қабылдаушының) жағдайы неге байланысты өзгереді?
10. Аналогты ақпарат деген не?
11. Цифрлы ақпарат деген не?
12. Аналогты ақпаратты қалай цифрлайды?
13.Хабар мен мәннің арасында қандай қатынас бар?

I.1.1. Тесттер:

1. Information сөзі қай тілден шыққан?
A) Грек;
B) Латын;
C) Орыс;
D) Француз;
E) Араб.

2. $a+b+c$ қандай қатынас тіліне жатады?
A) Табиғи;
B) Логикалық;
C) Мимикалық;
D) Математикалық;
E) Химиялық.

3. Хабарды компьютер арқылы өңдеу үшін не істеу керек?
A) Дискреттеу;
B) Топтау;
C) Құрылымдау;
D) Жазу;
E) Оқу.

I.1.2. Хабар және мазмұн.

Жоғарыда келтірілген мысалдардан хабар ақпараттың мазмұнын тасушысы екендігі байқалды, яғни, *ақпараттың мазмұны оның хабарымен бірге анықталады.* Сондықтан хабарды жіберуші мен қабылдаушының арасында хабардың түрі (бейнесі) жайында және оның мазмұны туралы алдын ала анықталған келісім болады. Осындай келісімді интерпретациялау ережесі деп атайды.

I.1.2. Ескертпе:

1. Түрліше интерпретацияланған бір ғана хабар әр түрлі мазмұнды тасушы болады. Мысалы, баспасөздегі белгілі бір мақаланы оқырмандар өздерінің көзқарастарына байланысты әр түрлі қабылдауы мүмкін.

2. Бір ғана мазмұн әр түрлі хабар арқылы берілуі мүмкін. Мысалы, әр түрлі қатынас тіліндегі бір ғана баяндама.

Берілген хабардың интерпретациялану ережесі бір заңдылықпен құрылған бүкіл хабардың жиынына қолдануға болатын жалпы ережеден алынады. Мысалы, егер хабар белгілі бір табиғи тілдегі сөйлем ретінде берілсе, онда бұл сөйлемнің интерпретациялау (сөйлемнің мағынасын түсіндіру) ережесін осы тілдің барлық сөйлемдеріне қолданылатын жалпы интерпретациялау ережесінен алуға болады. Демек, жалпы интерпретациялау ережесіне сандардың мәндерін олардың жазылу сұлбасы бойынша анықтау ережесі де жатады. Сонымен қатар, мағыналары «ақиқат» немесе «жалған» болатын тұжырымдар мен жауаптары «*иә*» немесе «*жоқ*» болатын сұрақтар хабар ретінде жиі пайдаланылады. Мысалы, «*Қардың түсі ақ болады*», «*адам еш уақытта өлмейді*», «*Информатиканы оқу қызық па?*», «*Осы айтылған түсінікті ме?*». Мұндай хабарларды интерпретациялау жоғарыда айтылғандармен қатар қабылдаушының өмір тәжірибесіне немесе санасының деңгейіне байланысты.

Қорыта айтқанда, жіберуші мен қабылдаушының хабарды интерпретациялау ережесін меңгеру қабілеттіліктері олардың зерделеріне байланысты.

I.1.2-Мысалдар.

Хабар және мазмұн арасындағы байланыстарды байқау үшін бірнеше мысалдар қарастырайық. Олар I.1.2-кестеде берілген.

I.1.2-Кесте. Хабардың мысалдары.

№	1–ші хабар	2–ші хабар
1.	Мен жақсы оқимын.	Я учусь хорошо.
2.	Жалыннан мұз шығады.	1>3
3.	Сен мені түсіндің бе?	X=0?
4.	XXI	21
5.	Ертең қар жауады.	Шам сөніп қалды.
6.	☼♀♂♥	© ®
7.	I speek.	Мен сөйлеймін.

1,2,3,4–мысалдарда екі түрлі хабар арқылы бір ғана мазмұнның берілуі көрсетілген: 1–мысалдағы қазақ және орыс тілдеріндегі сөйлемдердің мағынасы біреу–ақ, ал 2–мысалда нәтижесі «жалған» болатын екі түрлі тұжырым. 3–мысалда нәтижесі «иә» немесе «жоқ» деген жауаптарды талап ететін сұрақтар. 4–мысалда бір ғана санның мәні екі түрлі жазылып тұр.

5–мысалда бір ғана хабар арқылы бірнеше мазмұнның берілуі көрсетілген. Сонда осы мысалдағы 1–хабар арқылы енді күн суытады немесе сырғанақ ойнауға болады т.б. түсінуге болады.

6–мысалда алдын ала берілген шартты белгілер арқылы қатынас жасау мүмкіндігі көрсетілген. Мұндай хабардың мазмұнын тек осы шартты белгілердің қолдануын білгеннен кейін ғана анықтауға болады.

7–мысалда екі табиғи тілде жазылған бір мағына берілген.

I.1.2. **Тапсырмалар:**

Хабар мен мазмұн арасындағы байланысты табыңыз:

1. 2+2=4;
2. Ертең күн ыстық болады;
3. Мен студентпін. Я студент.

Көмек:

Хабар табиғи тілдегі сөйлем ретінде берілсе, онда оны интерпретациялану ережесін осы тілдің барлық сөйлемдеріне қолданылатын жалпы интерпретациялау ережесінен алуға болады.

І.1.2. Сұрақтар:

1. Интерпретациялау ережесі деген не?
2. Хабарды қалай беруге болады?
3. Хабар мен мазмұнның арасында байланыс бірмәнді ме?

І.1.2. Тесттер:

1. Хабарды жіберуші мен қабылдаушының арасында хабардың түрі (бейнесі) жайында және оның мазмұны туралы алдын ала анықталған қандай келісім бар?

A) Интерполяциялау;
B) Интеграциялау;
C) Интерпретациялау;
D) Ақпараттандыру;
E) Интервенциялау;

2. XIII және 13 хабарлар мен мазмұн арасындағы қандай байланыс бар?

A) Бір ғана санның мәні екі түрлі жазылып тұр;
B) Көпмазмұнды хабар;
C) Бірмазмұнды хабар;
D) Бірмазмұн және бір хабар;
E) Ешқандай байланыс жоқ;

3. Ақпараттың мазмұнын не тасиды?

A) Интерпретация;
B) Арна;
C) Жіберуші;
D) Қабылдаушы;
E) Хабар.

I.1.3. Ақпараттың көлемдік бірлігі және мөлшері

«Ақпарат мөлшері» деген ұғымды анықтау өте қиын. Ол үшін алдымен ақпараттың өлшем бірлігін анықтап алу керек. Оны анықтаудың екі жолы бар: *көлемдік* және *ықтималдық*. Осы жолдардың екеуі де XX ғасырдың 40–шы жылдарында бірдей белгілі болды. Оларды ендіргендер АҚШ ғалымдары, информатика ғылымының негізін тұрғызушылардың бірі Джон фон Нейман және Клод Шеннон.

Джон фон Нейман ең алғаш компьютерлерді құруға болатындығын көрсетті, ол ақпарат мөлшерін көлемдік жолмен өлшеуге алып келді, ал Клод Шеннон ақпарат мөлшерін ықтималдық жолмен өлшеуді анықтады.

Ақпарат көлемінің ең кіші бөлінбейтін бірлігін ***bit – бит*** деп атайды. Ол ағылшын тілінің ***bi****nary digi****t*** екілік цифр деген екі сөзінен алынған. Оған себеп компьютер жасаушыларға ақпаратты сақтау және өңдеу үшін екілік санау жүйесімен жұмыс істеу қолайлы болғандығы: тек қана екі тұрақты жағдайы бар физикалық элемент көптеген қондырғылармен оңай жүзеге асырылады, екі тұрақты жағдайлар 0 және 1 цифрларымен белгіленеді. Мысалы, электр тоғының жоқтығын не барлығын білдіретін, кернеудің төмендігін не жоғарғылығын өлшейтін, магниттеудің қарама–қарсы болатындығын бақылайтын және т.с.с. қондырғыларды жасау оңай.

Екілік цифрлар 0 және 1 арқылы жазылған ақпараттың *көлемдік мөлшері* деп осы жазуға қолданылатын екілік цифрларының санын айтады, ол сан әрқашан бүтін болады.

Ақпараттың көлемдік мөлшерінің келесі өлшем бірлігін *byte – байт* деп атайды, ол сегіз биттен тұрады, яғни, 1 байт = 8 = 2^3 бит. Бір байт арқылы бір–бірінен бөлек көлемдік мөлшері 2^8 = 256 әртүрлі символдарды жазуға болады, яғни, бір байттың сыйымдылығы 256 бит болады.

Ақпараттың көлемдік мөлшері өте үлкен сан болады. Сондықтан, қолдануға ыңғайлы болу үшін көлемдік мөлшердің ірі өлшем бірліктері анықталып енгізілген. Ол өлшем бірліктері екіге

еселі, яғни, екінің дәрежесі болуы керек. Ақпараттың көлемдік өлшем бірліктерінің тізімі I.1.3- кестеде берілген.

I.1.3-кесте. Ақпараттың көлемдік өлшем бірліктерінің тізімі.

№	Атау	Таңба	Дәреже
	byte – байт	Б	10^0
	kilobyte – килобайт	Кб	10^3
	megabyte – мегабайт	Мб	10^6
	gigabyte – гигабайт	Гб	10^9
	terabyte – терабайт	Тб	10^{12}
	petabyte – петабайт	Пб	10^{15}
	exabyte – эксабайт	Эб	10^{18}
	zettabyte – зеттабайт	Зб	10^{21}
	yottabyte – йоттабайт	Йб	10^{24}

I.1.3. Мысалдар:

1. Егер кітапта 400 бет, әр бетте 50 қатар, ал әр қатарда 50 таңба болса, онда бұл кітаптың көлемі 400*50*50 = 1000000 Байт = 1 Мб болады, яғни, көлемі 1 Гб дискіде осындай 1000 кітапты сақтай аламыз.
2. Егер электрондық кітаптың көлемі 10 мегабайт және электрондық қойманың сыйымдылығы 100 гигабайт болса, онда оның ішінде 100*1024/10 = 10240 электрондық кітап сақтауға болады.
3. Егер электрондық кітаптың көлемі 10 мегабайт және электрондық қойманың сыйымдылығы 100 терабайт болса, онда оның ішінде 100*1024*1024/10 = 104857600 электрондық кітап сақтауға болады.

I.1.3. Тапсырмалар:

1. Ақпараттың мөлшерін өлшеу әдістерін атаңыз.
2. «РИМ» сөзінің көлемдік мөлшері табыңыз.
3. Бір байттың көлемін көрсетіңіз.

Көмек:

1. Джон фон Нейман және Клод Шеннон ашқан әдістер.

2. Компьютерде сақталатын барлық ақпарат (сөз, сан, сурет, дыбыс, компьютерлік программа) екілік цифрлар тізбегі түрінде жазылады.

3. Бір байттағы биттің санын көрсетіңіз.

I.1.3. Сұрақтар:

1. Ақпараттың ең кіші өлшем бірлігі қалай аталады?
2. Ақпарат мөлшерін анықтаған кім?
3. Ақпараттың үлкен өлшем бірлігі нешеге еселі?

I.1.3. Тесттер:

1. Бір байтта қанша бит бар?

A) 2;
B) 10;
C) 8;
D) 0;
E) 1.

2. Бір ГБ қанша мегабайтқа тең?

A) 1040;
B) 1024;
C) 10024;
D) 124;
E) 102400;

3. Бір ТБ қанша гигабайтқа тең?

A) 10024;
B) 1040;
C) 1024;
D) 124;
E) 102400.

I.1.4. Ақпараттың ықтималдық бірлігі және мөлшері

«Ақпараттың ықтималдық мөлшері» деген ұғымды енгізіп, талқыламас бұрын, ықтималдық теориясына қатысты бір тәжірибені қарастырайық. Мысал ретінде, *N* жағы бар (ең кең тараған *N* = алты жақты) ойын тасын лақтыруды алуға болады. Бұл тәжірибенің нәтижесі болып 1, 2,..., *N* цифрларының біреуі жазылған жағы жоғары қарап түскені есептелінеді.

Анықталмағандықты өлшейтін сандық өлшемді ендірейік, оны ***энтропия*** деп атап, *H* арқылы белгілейік. *N* және *H* шамалары бір–бірімен мынадай функционалдық қатынаста болады:

$$H = f(N), \qquad (1.1)$$

мұнда f функциясы өспелі, теріс емес және біз қарастырған $N = 1, 2, \ldots, 6$ үшін анықталған.

Енді ойын тасын лақтыруды жете қарастырайық:

1) ойын тасын лақтыруға дайындаламыз: тәжірибе нәтижесі белгісіз, яғни қандай–да бір анықталмағандық бар, оны *H1* арқылы белгілейік;

2) ойын тасы лақтырылды: тәжірибе нәтижесі туралы ақпарат алынды, осы ақпараттың мөлшерін *I* арқылы белгілейік;

3) бұл тәжірибе жүзеге асқаннан кейін оның анықталмағандығын *H2* арқылы белгілейік;

Тәжірибені жүзеге асыру барысында алынған ақпараттың мөлшері ретінде тәжірибе алдындағы және тәжірибе соңындағы анықталмағандықтардың айырмасын алайық:

$$I = H1 - H2. \qquad (1.2)$$

Әрине, нақты нәтиже алынған жағдайда, бұрын болған анықталмағандық жойылады, яғни, $H2 = 0$, сондықтан тәжірибенің соңында алынған ақпарат мөлшері алғашқы энтропияға дәл келетін болады, $I = H1$. Басқаша айтқанда, тәжірибедегі анықталмағандық тәжірибенің нәтижесі туралы ақпаратқа дәл келеді. Мұнда *H2*-нің

мәні нөлге тең болмауы мүмкін, мысалы, тәжірибе барысында келесі түскен жақтың жазуы 3-тен үлкен.

Келесі маңызды жағдай, ол (1.1) формуласындағы f функциясының түрін анықтау. Егер ойын тасының жақтар санын N арқылы, ал лақтыру санын M арқылы белгілесек, онда ұзындығы M-ге тең және N таңбадан тұратын вектормен анықталатын нәтиженің жалпы саны N –нің M дәрежесіне тең:

$$X=N^M. \qquad (1.3)$$

Мысалы, алты жақты ойын тасын екі рет лақтырғанда мына нәтиже $X = 6^2 = 36$ аламыз және әрбір нәтиже X қандайда бір қосақ ($X1$, $X2$) болады, мұнда $X1$ және $X2$ – бірінші және екінші лақтырудың нәтижелері, ал X – осындай қосақтардың жалпы саны.

Ойын тасын M рет лақтыруды, бір–біріне байланыссыз «бір ғана лақтырыс болатын» қосалқы жүйелерден тұратын күрделі жүйе ретінде қарастыруға болады. Мұндай жүйенің энтропиясы бір жүйенің энтропиясына қарағанда M рет көп (энтропияның аддитивті прінсіпі):

$$f(6^M) = M \cdot f(6).$$

Бұл формуланы кез келген N үшін де жазуға болады:

$$F(N^M) = M \cdot f(N). \qquad (1.4)$$

Енді формула (1.3)-тің сол және оң жақтарын логарифмдесек:

$$\ln X = M \cdot \ln N,$$

ары қарай M былай табылады:

$$M = \frac{\ln X}{\ln N}.$$

Осы M үшін табылған мәнді (1.4) формуласына қойсақ:

$$f(X) = \frac{\ln X}{\ln N} \cdot f(N).$$

Мұндағы $\frac{f(N)}{\ln N}$ оң тұрақтысын K арқылы белгілесек

$$f(X) = K \cdot \ln X,$$

немесе (1.1) формуласын ескеріп,

$$H = K \cdot \ln N.$$

Әдетте $K = \frac{1}{\ln 2}$. Осыдан Хартлидің формуласы шығады:

$$H = \log_2 N. \qquad (1.5)$$

Қандай да бір шаманы ендіргенде нені өлшем бірлігі ретінде аламыз деген сұрақ өте маңызды. Әрине, $N = 2$ болғанда $H=1$ болатыны айқын. Басқаша айтқанда, өлшем бірлігі ретінде теңықтималды екі нәтиженің біреуі алынатын тәжірибе өткізуге байланысты ақпарат мөлшерін алуға болады (мысал ретінде екі–ақ жағы бар тиынды лақтыру тәжірибесін алуға болады). Осындай ақпарат мөлшерінің өлшем бірлігін *бит* деп атайды.

Жоғарыда қарастырған тәжірибенің 6 бит барлық N нәтижесі теңықтималды болады. Сондықтан, әрбір нәтижеге жалпы тәжірибе анықталмағандығының бірден N бөлігі келеді:

$$\frac{\log_2 N}{N}.$$

Мұнда i-ші нәтиженің ықтималдығы P_i бір бөлінген N-*ге* тең екендігі айқын. Осыдан Шеннон формуласы бойынша энтропия былай табылады:

$$H = \sum_{i=1}^{N} P_i \cdot \log_2 (\frac{1}{P_i}). \qquad (1.6)$$

Формула (1.6) тәжірибе нәтижесі тең емес ықтималды (яғни, P_i әртүрлі болуы мүмкін).

I.1.4. Ескертпе:

Көлемдік және ықтималдық ақпарат мөлшерлерінің арасындағы қатынас бір мәнді емес. Екілік символдармен жазылған кез келген мәтін ақпараттың көлемді мөлшер мағынасында өлшенгенмен, ықтималды мөлшер мағынасында өлшене бермейді. Егер қандай да бір хабар екі мағынада да өлшенетін болса, онда олар міндетті түрде бірдей болады және ақпараттың ықтималдық мөлшері оның көлемдік мөлшерінен көп бола алмайды.

I.1.4. Мысалдар:

1. Екі символды 0 мен 1-ден тұратын әліпбиді қарастырайық. Егер осы екілік әліпбиде жазылған ақпаратта 0 мен 1–дің қолдануы теңықтималды болса ($P(0) = P(1) = 0{,}5$), онда осы әліпбиде жазған кезде бір таңбаға келетін ақпарат мөлшері былай анықталады:

$$H = 1\log_2 2 = 1 \text{ бит.}$$

Сонымен, екілік сөздегі ақпарат мөлшері оны жазуға қолданылған екілік таңбалардың санына тең болады.

2. Әліпбиі 34 таңбадан тұратын тілдің мәтінінде әрбір таңбаның пайда болуына байланысты ақпарат мөлшері (1.5) формула бойынша: $H = 1\log_2 34 = 5$ бит.

Бірақ, басқа тілдердегі сияқты, осы тілдің әріптері мәтінде бірдей жиілікте кездеспейді. (1.6) формула бойынша: *H= 4,72 бит.*

I.1.4. Тапсырмалар

1. Жәшікте әртүрлі түсті 16 шар бар. Ақ шар алынғаны туралы ақпарат көлемі қанша?

2. Таңбалар саны 42 болатын әліпбиде жазылған қазақ тіліндегі хабарлардағы әрбір символдың пайда болуына байланысты ақпарат мөлшерін анықтаңыз.

3. Бір мегабайтта жазылған хабарлардан әрбір екілік цифрдың пайда болуына байланысты ақпарат мөлшерін анықтаңыз.

Көмек:

1. $H = \sum_{i=1}^{N} P_i \cdot \log_2 (\frac{1}{P_i})$.

2. Ақпарат мөлшерін табуға (1.5) формуланы пайдаланыңыз.

3. Бір мегабайтта 10^6 бит бар екендігін ескеру керек.

I.1.4. Сұрақтар:

1. Анықталмағандықты өлшейтін сандық өлшем қалай аталады?

2. Ақпарат өлшеудің көлемдік және ықтималдық бірліктер арасындағы қатынас бірмәнді ме?

3. Хартлидің формуласы не үшін керек?

I.1.4. Тесттер:

1. Ақпарат көлемінің ең кіші бөлінбейтін бірлігі не?

A) Терабайт (terabyte);

B) Байт (Byte);

C) Мегабайт (megabyte);

D) Гегабайт (gigabyte);

E) Бит (bit).

2. Анықталмағандықты өлшейтін сандық өлшем

A) Эктропия;

B) Энтропия;

C) Өлшем;

D) Ақпарат;

E) Байт.

3. Хартли формуласы?

A) $H = \log_2 N$;

B) $H = \lg N$;

C) $H = \ln N$;

D) $H = \ln (N–1)/2$;

E) $H = \ln (N–1)^2$.

I.2. Шамалар

I.2.1. Шамалардың түрлері

Информатикада ақпарат ұғымын шама деген ұғыммен жиі ауыстырады, мұнда ақпарат хабарын шаманың атауы ретінде, ақпарат мазмұны шаманың мәні ретінде қарастырылады.

Шамалар – ақпаратты өңдеу кезінде қолданылатын объектілер.

Шамалар, өздерінің есімдеріне мәндер беру тәсіліне байланысты *тұрақты шамалар* және *айнымалы шамалар* болып бөлінеді.

Егер шамалар мәндері қолданып отырған қатынас тілінің жалпы *интерпретациялау* (түсіндіру) *ережелерін* құрған кезде анықталса, онда мұндай шамаларды тұрақты дейміз. Демек, тұрақты шамаға атау берген кезде оның мәні бірге анықталады. Мәнімен бірге оның типі де белгілі болады. Мысалы, егер 1, 3 және 5 цифрларынан құрылған 135 тізбегін тұрақты шаманың атауы деп қарастырсақ, онда оның мәні «бір жүз отыз бес» деген бүтін сан болады. Ал егер осы цифрлардан басқа тізбек 315 құрастырсақ, онда ол мәні «үш жүз он бес» деген бүтін сан болатын басқа тұрақты шаманың атауы болады. Осылардан мынадай тұжырым шығады:

Тұрақты шаманың есімі, мәні және типі өзгермейді, олардың барлығы бір мезгілде анықталады.

Енді *айнымалы шамаларды* анықтайық. Егер шамалардың атауларына берілген мәндер өңдеу кезінде өзгеретін болса, онда мұндай шамаларды *айнымалы шамалар* дейміз.

Әдетте, айнымалы шама мәнінің типі өзгермеуі тиіс. Ал олай болмаған жағдайда, осы шаманы өңдеу кезінде шешуі қате шығатын көп қиындықтар пайда болуы мүмкін.

Информатикада айнымалы шамаға атау беру үшін «идентификатор» деген ұғымды пайдаланады.

Идентификатор – әріптен басталатын және әріптер мен цифрлардан құралған ұзындығы шектелген тізбекті айтады.

Шаманың мәні өзінің типімен сипатталады. Жалпы, шаманың мәндері: *сандық, символдық* және *логикалық* типтерге бөлінеді.

Берілген шамаларды өңдеу үшін осы шамаларға *амалдар* қолдану керек. Ал амалдарды қолдану үшін олардың анықтамаларын, белгілерін және қасиеттерін білу қажет.

Амалдарды шамалардың қай типтерінде анықталғандығына байланысты *сандық амалдар*, *символдық амалдар* және *логикалық амалдар* деп топтастырып атауға болады.

Айта кететін бір жағдай, ол осы амалдардың қайсысы болмасын математиканың түрлі саласында анықталған және зерттелген. Мысалы, символдық амалдардың барлығы «Математикалық лингвистика» деген тілдің құрамы мен қасиеттерін зерттейтін саласында, логикалық амалдардың барлығы «Математикалық логика» деген саласында, ал сандық амалдардың барлығы математиканың басқа салаларында (арифметика, алгебра және т.б.) анықталады.

Әрбір шама типтерінде орындалатын амалдардың қасиеттерін анықтауға болады. Ол қасиеттер топтастырылады да, осы шамаларға байланысты *аксиоматика* құрайды.

Төменде қарастыратын әртүлі типті шамаларға арналған аксиоматикалардың көп ұқсастықтары бар. Олар типтері сәйкес өрнектердің *эквиваленттігін* көрсетеді, солардың ішінде *коммутативтік*, *ассоциативтік* және *дистрибутивтік* заңдылықтары бар. Осындай заңдылықтар күрделі өрнектерді қарапайымдап, ондағы амалдар санын қысқартады және оның есептелуін жеңілдетеді. Олардың практикадағы маңызы өте зор.

I.2.1. Мысалдар:

1. A – идентификатор;
2. XI – идентификатор;
3. N12B – идентификатор;
4. 7X – идентификатор емес, себебі 7 цифрынан басталып тұр;
5. A+B – идентификатор емес, себебі ішінде «+» бар.

I.2.1. Тапсырмалар

Төмендегі шамалар идентификатор бола ала ма?

1) A–B;
2) XY;
3) C6R7;

Көмек:

Идентификатор деп әріптен басталатын және әріптер мен цифрлардан тұратын тізбекті айтады.

I.2.1. Сұрақтар:

1. Шамалардың қандай типтері бар?
2. Идентификатор не үшін керек?
3. Айнымалы шама мен тұрақты шаманың айырмасы неде?

I.2.1. Тесттер:

1. Ақпаратты қандай ұғыммен ауыстыруға болады?
   A) Хабар;
   B) Идентификатор;
   C) Бейнелеу;
   D) Шама;
   E) Анықтау.

2. Есімі, мәні және типі өзгермейтін қандай шама?
   A) Анықталған шама;
   B) Айнымалы шама;
   C) Тұрақсыз шама;
   D) Тұрақты шама;
   E) Анықталмаған шама.

3. Шамалар, өздерінің есімдеріне мәндер беру тәсіліне байланысты қандай түрге бөлінеді?
   A) Тұрақты және айнымалы шамалар;
   B) Тұрақты және тұрақсыз шамалар;
   C) Айнымалы және тұрақсыз шамалар;
   D) Тұрақсыз және айқын шамалар;
   E) Нақты және тұрақсыз шамалар.

I.2.2. Сандық шамалар түрлері

Сандық тип сандар жиынынан және оларда анықталған амалдар мен осы амалдардың қасиеттерінен құрылады. Олар *бүтін сандар*, *нақты сандар* және *комплекс сандар* болып үшке бөлінеді.

I.2.2. Ескертпе:

Информатикада сандық шамаларды өңдеу әр түрлі сандардың санау жүйесін қажет етуі мүмкін. Ондай жүйелердің негіздері 2,3,4,. болуы ықтимал. Олардың бір бірінен айырмашылығы тек сандардың мәндерін әртүрлі әдіспен белгілеу, ал сандарда анықталған амалдардың түрі және олардың қасиеттері бірдей болады. Сондықтан төменде алдымен бізге жақсы таныс сандардың ондық (негізі 10) санау жүйесінде жазылуы, оларда анықталған амалдар түрі және осы амалдардың қасиеттері беріледі. Содан кейін сандардың басқа (негіздері басқа) санау жүйелері қарастыралады, себебі ондық санау жүйесіндегі сандарға байланысты айтылғандардың барлығы басқа санау жүйесіндегі сандарға жарайды деуге болады.

Бүтін сандар араб цифрлары арқылы кескінделеді, олардың теріс мәндерінің алдына «–» таңбасы жазылады, ал оң мәндерінің алдына «+» таңбасы жазылуы мүмкін.

Нақты сандар бейнелеу тәсіліне байланысты екі топқа бөлінеді: *тұрақты нүктелі нақты сандар* және *жылжымалы нүктелі нақты сандар*.

Тұрақты нүктелі нақты сандардың бейнесі бүтін бөліктен және бөлшек бөліктен тұрады. Бүтін бөлік бөлшек бөліктің алдында (сол жағында) орналасады және олар өзара оңдық белгісі деп аталатын «.» таңбасы арқылы айырылады. Олардың мәндерінің оң немесе терістігін көрсету үшін кескіндердің алдына «+» немесе «–» таңбасы жазылады. Кескіндердің екі бөліктері де араб цифрларымен бейнеленеді.

Жылжымалы нүктелі нақты сандардың кескіні *мантисса*, *санау жүйесінің негізі* және *реті* деп аталатын бөліктерден тұрады. Мантиссаның да, реттің де мәндері оң немесе теріс болуы мүмкін. Оларды белгілеу үшін мәндерінің алдында «+» немесе «–»

жазылады. Рет бүтін сандар сияқты, ал мантисса тұрақты нүктелі нақты сияқты бейнеленеді.

Егер ***M*** мантиссаны, ***p*** ретті, ***q*** санау жүйенің негізін белгілесе, онда жылжымалы нүктелі нақты сандар мынадай болады:

$$M * q^{p}$$

Осы айтылғандарды түсіну үшін I.2.2-кестеде жылжымалы нүктелі нақты сандарға мысалдар қарастырылған.

I.2.2 - кесте. Жылжымалы нүктелі нақты сандарға мысалдар

№	Мысал	Мантисса	Реті	Мәні
1.	$-12.*10^{3}$	–12	+3	–12000
2.	$0.3*10^{+2}$	0.3	+2	30
3.	$254*10^{-2}$	254	–2	2.54
4.	$1.5*10^{1}$	1.5	+1	15
5.	$+2.17*10^{2}$	2.17	+2	217

Бір ғана нақты санның жылжымалы нүктелі түрде көптеген жазылуы болуы мүмкін. Мысалы, бір ғана 3.14 деген санның мынадай жазылулары болады:

$$314.*10^{-2} = 31.4*10^{-1} = 3.14*10^{0} = 0.314*10^{1} = 0.0314.*10^{2} = \text{K}$$

Жылжымалы нүктелі нақты санның жазылуын бір ғана түрде беру үшін оны *нормалдау* қажет. Ол үшін мына шарт орындалуы керек:

$$q^{-1} \le |M| < 1,$$

мұнда $|\text{M}|$ – абсолют шама.

Мысалы, жылжымалы нүктелі нақты сандары $13.64*10^{2}$ және $0.00617*10^{-5}$ нормалданған түрде мынадай болады:

$0.1364*10^{4}$ және $0.617*10^{-7}$

Комплексті сандардың кескіні алгебралық қосынды түрінде беріледі. Оның бірінші қосылғышы (сол жақтағы) нақты бөлік, ал екінші қосылғышы (оң жақтағы) жорамал бөлік деп аталады. Нақты бөлік те, *жорамал* бөлік те нақты сандар түрінде жазылады. Оларды бір–бірінен айыру үшін жорамал бөліктен кейін оның белгісі ретінде латын әріпі «*i*» жазылады.

I.2.2. Мысалдар:

1. 3.14 – оң нақты сан, бүтін бөлігі 3, ал бөлшек бөлігі 14 болатын оң, тұрақты нүктелі нақты.

2. +5 – оң бүтін сан.

3. 0.2 – оң нақты сан, бүтін бөлігі 0, бөлшек бөлігі 2, оң, тұрақты нүктелі, нақты. Кейде, мұндайларды бүтін бөлігін көрсетпей–ақ.2 түріндегі жазуға болады.

4. -1.001 – теріс нақты сан, бүтін бөлігі 1, ал бөлшек бөлігі 001 теріс, тұрақты нүктелі нақты.

5. 0.0 – нақты сан, бүтін және бөлшек бөлігі де 0 тұрақты нүктелі нақты.

6. $0 + 3i$ – оң комплекс сан, нақты бөлігі 0, жорамал бөлігі 3.

7. $-3.14 + 2i$ – теріс комплекс сан, нақты бөлігі -3.14, жорамал бөлігі 2.

8. $1.7*10^2 - 0.12i$ – оң комплекс сан, нақты бөлігі $1.7*10^2$, жорамал бөлігі -0.12.

I.2.2. Тапсырмалар

Тұрақты және жылжымалы нүктелі нақты сандарды, комплекс сандарды анықта?

1. 40.23;
2. $1.21*10^2+5i$;
3. $3.3*_{10^{-2}}$.

Көмек:

1) Тұрақты нүктелі нақты сандардың бейнесі бүтін және бөлшек бөліктерден тұрады.

2) Комплекс сандардың нақты бөлігі және кіші латын *i* әріпімен аяқатлатын жорамал бөлігі болады.

3) Жылжымалы нүктелі нақты сандардың кескіні мантисса, санау жүйесінің негізі және реті бөліктерінен тұрады.

I.2.2. Сұрақтар:

1. Сандық шамалардың қандай түрлері бар?
2. Нақты сандар қандай топқа бөлінеді?
3. Комплекс сандар қандай бөліктерден тұрады?

I.2.2. Тесттер:

1. Жылжымалы нүктелі нақты санды нормалдау үшін не орындалуы керек?

A) $q^{+1} \le |M| < 1$;

B) $q^{-1} = |M| < 1$;

C) $q^{-1} \le |M| < 1$;

D) $q^{-1} > |M| < 1$;

E) $q^{-1} \le |M| < \infty$.

2. 0 + 3*i* санының нақты бөлігі мен жорамал бөлігі қайсы?

A) нақты бөлігі *i*, жорамал бөлігі 3 ;

B) нақты бөлігі 0, жорамал бөлігі 3;

C) нақты бөлігі 3, жорамал бөлігі *i* ;

D) нақты бөлігі *i*, жорамал бөлігі 0 ;

E) нақты бөлігі 0, жорамал бөлігі *i*.

3. Кез келген бүтін санды қандай цифрлармен кескіндейді?

A) Латын цифрлары;

B) Грек цифрлары;

C) Қазақ цифрлары;

D) Орыс цифрлары;

E) Араб цифрлары.

I.2.3. Сандық амалдар және олардың қасиеттері

Сандық типтерде анықталған амалдар бізге бастауыш мектептен *арифметикалық амалдар* ретінде таныс: *қосу*, *алу*, *көбейту*, *бөлу*. Олар мына таңбалармен «+», «–», «*», «/» сәйкес белгіленеді. Осы амалдарды қолданып сандық өрнектерді құруға болады. Әдетте, сандық өрнектерді жазу үшін біз амалдар таңбалары операндтар (аргументтер) арасында жазылатын *инфиксті жазбаны* пайдаланамыз. Мысалы, кез–келген a және b сандары үшін осы сандарының қосындысы болатын үшінші сан сәйкес қойылса, онда оны біз $a+b$ түрінде жазамыз.

Сандық өрнектердің мәндерін есептеу кезінде осы амалдардың приоритеттерін (орындалу реттерін) ескеру қажет: алдымен жоғары приоритетті көбейту мен бөлу, содан кейін төмен приоритетті қосу мен алу орындалады. Кейде орындалу ретін басқаша анықтау үшін *жақшаларды* да пайдалануға болады. Бір өрнекте бірнеше жақша қолдануға және оларды бірінің ішіне бірін жазуға болады: ең ішкі жақшадағы және ең солжақ жақшадағы амал бірінші орындалады.

Сандық өрнектерді жазудың *префиксті жазбасы* және *постфиксті жазбасы* деп аталатын жақшасыз тәсілдері де бар. Префиксті жазбада амалдар таңбалары өздерінің операндтарының алдында, ал постфиксті жазбада амалдар таңбалары өздерінің операндтарынан кейін жазылады. Мысалы, инфиксті жазбадағы сандық өрнек (*x*+3)*(y-2) префиксті жазбада * + *x* 3 - *y* 2 деп, ал постфиксті жазбада *x* 3 + *y* 2 - * деп жазылады.

I.2.3. Ескертпе:

1. Белгілеулері бірдей болғанымен осы амалдардың әр түрлі сандық типте орындалу ережелері әр түрлі. Мысалы, бүтін сандарды қосу ережесі нақты сандарды қосуға жарамайды, сондай–ақ, нақты сандарды қосу ережесі комплекс сандарды қосуға жарамайды

2. Арифметикалық өрнектің префиксті және постфиксті жазбасы, олардың авторы поляк математигі Лукашевич құрметіне, *тура поляк жазбасы* және *кері поляк жазбасы* деп аталған.

Мейлі a, b және c – кез келген сандар және a^{-1} – a санына кері сан. Сонда сандық шамаларда анықталған «+» – *қосу* және «*» – *көбейту* амалдары I.2.3- кестедегі қасиеттерге ие.

I.2.3-кесте. Сандық амалдардың қасиеттері.

№	Аксиома	Сипаты
1	$a+b=b+a$	Коммутативтік заңы
2	$a*b=b*a$	
3	$a+(b+c)=(a+b)+c$	Ассоциативтік заңы
4	$a*(b*c)=(a*b)*c$	
5	$a*(b+c)=a*b+a*c$	Дистрибутивтік заңы
6	$(a+b)*c=a*c+b*c$	
7	$a+0=0+a=a$	Қосудың қасиеті
8	$a+(-a)=(-a)+a=0$	
9	$a*1=1*a=a$	Көбейтудің қасиеті
10	$a*a^{-1}=a^{-1}*a=1$	

8–аксиома әрбір оң a саны үшін оған қарама–қарсы теріс $-a$ саны болатындығы, ал 9 – аксиома әрбір нөлге тең емес a саны үшін оған кері сан $a^{-1}=1/a$ табылатындығын көрсетіп тұр.

I.2.3. Мысалдар:

1. $5*(b+c)=5*b+5*c$.
2. $3+(-3)=(-3)+3=0$.
3. $(4+8)/2+(5+3)*2=12/2+8*2=6+16=22$

I.2.3. Тапсырмалар::

1. 6 санына кері санды табыңыз.
2. $2a+3b-5a$ өрнегінің $a=5$ және $b=8$ болғандағы мәнін есептеу.
3. $(a+5)*(3-b)$ өрнегінің $a=10$ және $b=2$ болғандағы мәнін есептеу.

Көмек:

1. $a * a^{-1} = a^{-1} * a = 1.$
2. a мен b үшін $a+b$ белгілейтін үшінші сан сәйкес қойылады.
3. Амалдардың орындалу реттері және жақшаларды ескеру.

I.2.3. Сұрақтар:

1. Префиксті жазба деген не?
2. Инфиксті жазба деген не?
3. Постфиксті жазба деген не?
4. Белгілеулері бірдей амалдардың әр түрлі сандық типте орындалуы бірдей ме?

I.2.3. Тесттер:

1. Коммутативтік заңын табыңыз?

A) $a + b = b + a$;

B) $a + (b+c) = (a + b) + c$;

C) $a + 0 = 0 + a = a$;

D) $a*1 = 1* a = a$;

E) $a * (b+c) = a*b + a* c.$

2. Дистрибутивтік заңын табыңыз?

A) $a + (b+c) = (a + b) + c$;

B) $a * (b+c) = a*b + a* c$;

C) $a + 0 = 0 + a = a$;

D) $a*1 = 1* a = a$;

E) $a + b = b + a.$

3. Ассоциативтік заңын табыңыз?

A) $a * (b+c) = a*b + a* c$;

B) $a + (b+c) = (a + b) + c$;

C) $a + (-a) = (-a) + a = 0$;

D) $a*1 = 1* a = a$;

E) $a + b = b + a.$

I.2.4. Сандардың санау жүйелері

Жалпы, *санау жүйесі* деп сандардың цифрлар арқылы жазылу әдістері мен ережелер жиынын айтады. Сандардың цифрлар арқылы жазылуының бірнеше әдістері бар.

Барлық санау жүйесі мына шарттарды қанағаттандыру керек:

– берілген диапазондағы (аралықтағы) кез–келген сандардың мәндерін жазу мүмкіншілігінің болуы;

– әрбір цифрлар тізбегінің тек қана бір сандық мәнді анықтауы;

– амалдарды орындаудың қарапайымдылығы.

Барлық санау жүйелері *позициялық санау жүйелері* және *бейпозициялық санау жүйелері* болып бөлінеді:

1. *Позициялық санау жүйесінде* цифрлардың мәні олардың жазылуындағы орнына (позициясына) байланысты болады. Егер санның жазылуында бір ғана цифр бірнеше рет кездессе, онда ол әр түрлі мәнді анықтайды. Мысалы, сандардың араб цифрлары аркылы жазылуы: үш таңбалы (разрядты) 333 – үш жүз отыз үш санында сол жақтағы бірінші 3 цифры үш жүздікті, ортадағы 3 цифры үш ондықты, ал оң жақтағы 3 цифры үш бірлікті анықтайды.

2. *Бейпозициялық санау жүйесінде* цифрлардың мәні олардың берілген санның жазылуындағы орнына байланысты болмайды. Мысалы, сандардың рим цифрлары аркылы жазылуы: LXXXVIII – сексен сегіз санында L – елуді, X – онды, V – бесті, I – бірді бейнелейді.

Позициялық санау жүйесі өзінің негізімен сипатталады. Негіз осы жүйедегі қолданылатын цифрлар санын анықтайды. Мысалы, ондық жүйеде негіз – он, ал сегіздік жүйеде негіз – сегіз болады, екілік жүйеде негіз – екі және т.с.с.

Барлық позициялық санау жүйелерінің мүмкіншіліктері бірдей. Бірақ, олардың ішінде ондық санау жүйесі кең тараған. Оған себеп –адамның екі қолындағы саусақтардың саны он болғаны. Екі қолдың саусақтарымен бір ден онға дейін санау өте ыңғайлы және оңай. Онға дейін санап табиғи «есептеу құралдың» барлық мүмкіндігін тауысқаннан кейін, ондық 10 (он) санын жаңа, келесі позицияның (екі разрядты) бірлігі деп қарастырылуы санаға

қонымды. Ары қарай ондық он саны келесі позицияның (үш разрядты) бірлігі болады және тағы солай есептей бере ондық санау жүйесі туған. Бірақ, ондық санау жүйесі санауда бірден басым орын алған жоқ. Әртүрлі тарихи уақыттарда көп халық ондық емес санау жүйелерімен пайдаланған. Мысалы, көне түркілерде санау жүйесінің негізі жеті (1 апта = 7 күн), көне вавилондықтарда – алпыс (1 минут = 60 секунд, 1 сағат = 60 минут, 1 градус = 60 минут), ағылшындарда – он екі (1 жыл = 12 ай, 1 фут = 12 дюйм, 1 шиллинг = 12 пенс) болған.

I.2.4. Ескертпе:

Ең кіші санау жүйесінің негізі болатын сан – *екі*, ол *екілік санау жүйесі* деп аталады. Екілік санау жүйесіндегі сан тек 0 және 1 цифрларынан тұрады.

Кез келген ***q*** негізді позициялық санау жүйесіндегі берілген ***A*** санын төмендегідей өрнектеуге болады.

$$A_{(q)} = a_{n-1}q^{n-1} + \Lambda + a_1q^1 + a_0q^0 + a_{-1}q^{-1} + \Lambda + a_{-m}q^{-m}, \qquad (\boldsymbol{1})$$

мұнда $\boldsymbol{a_i}$ – санау жүйесінде қолданылатын цифрлар, n – бүтін бөліктегі разрядтар саны, m – бөлшек бөліктегі разрядтар саны, ($i=n-1,.,1,0,-1,.,-m$).

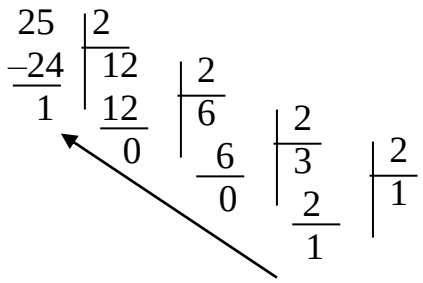
Осы санау жүйелердің арасында бізге керегі *ондық санау жүйесі* және *екілік санау жүйесі.* Олар I.2.4 - кестеде беріледі.

1.2.4 - кесте. Ондық цифрлар мен екілік цифрлар.

Ондық цифр	Екілік цифр
0	0
1	1
2	10
3	11
4	100
5	101
6	110
7	111
8	1000
9	1001

Бұл кестеден байқалып отырғаны, ол бір ғана сандық мәнді әр түрлі санау жүйесінде жазған кезде әр түрлі таңбалар (разрядтар) саны керектігі. Мысалы, ондық санау жүйесіндегі екі таңбалы 16 саны екілік санау жүйеде 10000 болып, бес таңбаны талап етеді.

Берілген ***p*** негізді бүтін санды жаңа ***q*** негізіне аудару үшін осы санды бөлінді ***q***-ден кіші болғанша бірнеше рет ***q***-ге бөлу керек. Шыққан бөліндіні жаңа ***q*** негіздегі санның ең үлкен разрядының мәні деп алып, ал қалған разрядтардың мәндері ретінде соңғы қалдықтан бастап бірінші қалдыққа дейінгі бағытта солдан оңға қарай тізбек құру керек. Мысалы, ондық жүйесіндегі 25 санын екілік жүйесіне аудару төмендегідей болады:

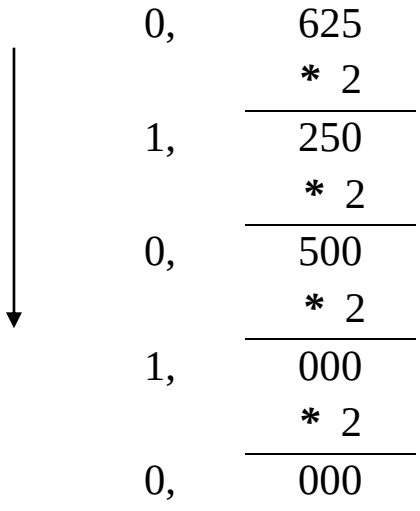


Яғни, $25_{(10)} = 11001_{(2)}$. Осының дұрыстығын тексеру үшін 11001 санын жоғарыда көрсетілген өрнек (1)-ге сәйкес жіктеледі:

$$11001_{(2)} = 1*2^4 + 1*2^3 + 0*2^2 + 0*2^1 + 1*2^0 = 16 + 8 + 0 + 0 + 1 = 25_{(10)}.$$

Екілік жүйесіндегі амалдардың орындалуы мен қасиеттері ондық жүйесіндегі осы амалдардың орындалуы мен қасиеттеріне ұқсас болады. Бұл амалдардың қасиеттері ондық сандарда анықталған сәйкес амалдардың қасиеттерімен бірдей.

Берілген ***p*** негізді дұрыс бөлшек санды жаңа ***q*** негізіне аудару үшін осы санды көбейтіндінің бөлшек бөлігінің разрядтарының мәндері нөлге тең болғанша немесе берілген дәлдік алынғанша бірнеше рет ***q***-ге көбейту керек. Жаңа ***q*** негізді дұрыс бөлшектің разрядтарының мәндері ретінде ең бірінші шыққан бүтін бөліктен бастап соңғы шыққан бүтін бөлікке дейінгі бағытта солдан оңға қарай тізбек құру керек. Мысалы, ондық жүйесіндегі 0.625 санын екілік жүйесіне ауыстыру былай болады:



Яғни, шыққан нәтиже мынадай: $0.625_{(10)} = 0.1010_{(2)}$.

I.2.4. Мысалдар:

1. Ондық санау жүйесіндегі төрт таңбалы бүтін сан ***1952*** үшін:

$$1952_{(10)} = 1*10^3 + 9*10^2 + 5*10^1 + 2*10^0$$

2. Ондық санау жүйесіндегі үш таңбалы бүтін бөлігі және үш таңбалы бөлшек бөлігі бар ***596.174***$_{(10)}$ саны былай өрнектеледі:

$$596.174_{(10)} = 5*10^2 + 9*10^1 + 6*10^0 + 1*10^{-1} + 7*10^{-2} + 4*10^{-3}$$

3. Екілік санау жүйесінде төрт таңбалы бүтін бөлігі және үш таңбалы бөлшек бөлігі бар ***1010.101***$_{(2)}$ саны былай жіктеледі:

$$1010.101_{(2)} = 1*2^3 + 0*2^2 + 1*2^1 + 0*2^0 + 1*2^{-1} + 0*2^{-2} + 1*2^{-3}$$

I.2.4. Тапсырмалар:

1. Екілік санау жүйесінде берілген 10000001 санын ондық санау жүйесіне аударыңыз.

2. Ондық санау жүйесінде берілген 129 санын сегіздік санау жүйесіне аударыңыз.

3. Ондық санау жүйесінде берілген 1952 санын оналтылық санау жүйесіне аударыңыз.

Көмек:

1. Кез келген ***q*** негізді позициялық санау жүйесінде берілген ***A*** санын төмендегідей өрнектеуге, болады.

$$A_{(q)} = a_{n-1}q^{n-1} + \Lambda + a_1q^1 + a_0q^0 + a_{-1}q^{-1} + \Lambda + a_{-m}q^{-m} \qquad (1)$$

2. Берілген ***p*** негізді бүтін санды жаңа ***q*** негізіне аудару оны бөлінді ***q***-ден кіші болғанша бірнеше рет ***q***-ге бөліп, шыққан бөліндіні жаңа ***q*** негіздегі санның ең үлкен разряд мәні деп алып, ал қалған разрядтардың мәндері ретінде соңғы қалдықтан бастап бірінші қалдыққа дейін бағытта солдан оңға қарай тізбек құрылады.

I.2.4. Сұрақтар:

1**.** Санау жүйесі деп нені айтады?

2. Барлық санау жүйелері қандай топтарға бөлінеді?

3. Бір санды әртүрлі санау жүйесінде бейнелеуге бола ма?

I.2.4. Тесттер:

1. Позициялық санау жүйесінде қандай байланыс бар?

A) Цифрлардың мәні олардың жазылуына байланысты болады;

B) Әріптердің мәні олардың жазылу орнына байланысты;

C) Санның мәні жазылуға байланысты болады;

D) Цифрлардың орны олардың жазылуындағы мәніне байланысты болады;

E) Цифрлардың мәні олардың жазылуындағы орнына байланысты болады.

2. Бейпозициялық санау жүйесінде.

A) цифрлар мәні олардың жазылу орнына байланысты емес;

B) әріптер мәні олардың жазылу орнына байланысты емес;

C) санның мәні оның жазылуына байланысты емес;

D) санның мәні оның жазылуындағы орнына байланысты болмайды;

E) цифрлар мәні олардың берілген санның жазылуындағы орнына байланысты болады,

3. Екілік санау жүйесінің негізі не?

A) 2;

B) 0 мен 1;

C) 1 мен 2;

D) 10;

E) 0.

I.2.5. Символдық шамалар түрлері

Символдық тип белгілі бір қатынас тілінің әліпби таңбаларынан тұратын тізбектердің жиынынан құралады. Мысалы, қазақ тілінде мұндай тізбектер ретінде осы тілдегі сөздер және сөйлемдер болуы мүмкін.

Информатикада қолданылатын қатынас тілдерінің әліпбилері негізінен:

- *әріптер*;
- *цифрлар;*
- *арнаулы таңбалардан* тұрады.

Әріптердің ішінде бас және кіші әріптер болуы мүмкін. Әдетте олар үшін латын әліпбиінің әріптері алынды. Кейде бір тілдің әліпбиі басқа тілдің әліпби әріптерімен толықтырылуы мүмкін. Мысалы, қазіргі қазақ тілінің әліпбиінде орыс тілінің *ё, э, ю, я, ц, ч, щ, ь, ъ* деген қосымша әріптері бар.

Цифрлар ретінде рим цифрлары емес, араб цифрлары *0,1,2,3,4,5,6,7,8,9* алынады.

Арнаулы таңбаларға *пайдаланылатын амалдардың таңбалары*, *тыныс белгілері*, *топтастыру таңбалары* (*жақшалар*), *кетік таңбасы* және т.б. жатады.

Символдық шамалардың мәндері символдық жақшаның ішіне жазылады. Символдық жақша ретінде, « ' » – дәйекше (апостроф) таңбасы қолданылады. Мысалы,

'ABC',
'2015',
'X+1 > 0',
'(3,14)'.

Сонымен, символдық шаманың мәні символдық жақшаға алынған әріптер, цифрлар немесе арнаулы таңбалардан құрылған тізбекпен кескінделеді екен. Әрбір тізбенің ұзындығын анықтауға болады, ол осы тізбектегі символдар санына тең болады. Ұзындықты екі « | » таңбасының ішінде белгілейді.

Мысалы, жоғарыдағы символдық шамалардың ұзындықтары мынадай:

|ABC| = 3,

|2015| = 4,

| X+1 > 0| = 5,

(3,14)| = 6.

Символдық шамалар жиынына абстракты шама «*бос тізбек*» енгізіледі. Бос тізбенің құрамында бірде–бір символ болмайды. Әдетте, бос тізбені «ε» таңбасымен белгілейді. Бос тізбенің ұзындығы нөлге тең, яғни, | ε | = 0.

I.2.5. Ескертпе:

Кетік бос тізбек емес, ол нақты бір таңба болып есептелінеді, бір кетік (бос орынның) ұзындығы бірге тең, яғни, | | = 1.

1.2.5. Мысалдар:

1) ‘ABCDE’ – латын әліпбиінің алғашқы бас әріптерінен құрылған тізбек.

2) ‘X+Y = 100’ – аралас таңбалардан құрылған тізбек.

3) ‘A1’– латын әріпімен цифрдан құрылған тізбек.

I.2.5. Тапсырмалар:

Мына тізбектердің типін анықтаңыз:

1.‘АӘБ’;

2.‘2013’;

3.‘X+Y > 1’;

Көмек:

Символдық тип белгілі бір қатынас тілінің әліпби таңбаларынан тұратын тізбектердің жиынынан құралады.

I.2.5. Сұрақтар:

1. Белгілі бір қатынас тілінің әліпби таңбаларынан тұратын тізбектердің жиыны нені құрайды?

2. Символдық шаманың мәні қалай беріледі?

3. Информатикада қолданылатын қатынас тілдерінің әліпбилері неден тұрады?

4. Инфориатикада цифрлар ретінде нелер қабылданады?

5. Инфориатикада арнаулы таңбалар ретінде нелер қабылданады?

6. Информатикада әріптер ретінде ретінде нелер қабылданады?

7. |ABCD| – символдық шаманың ұзындығы қа ншаға тең?

8. Бос тізбеден кетіктің айырмашылығы неде?

9. Бос тізбенің ұзындығы неге тең?

10. Кетіктің ұзындығы неге тең?

I.2.5. Тесттер:

1**.** Информатикада қолданылатын қатынас тілдерінің әліпбилері қандай топтан тұрады?

A) Әріптер, цифрлар және арнаулы таңбалар тобынан тұрады.

B) Әріптер, цифрлар және арнаулы таңбалар тобынан тұрмайды.

C) Цифрлар және арнаулы таңбалар тобынан.

D) Арнаулы таңбалар тобынан ғана.

E) Әріптер, цифрлар тобынан ғана.

2. Аралас таңбалардан құрылған тізбек қайсы?

A) ‘210’ ;

B) ‘ABC’ ;

C) ‘X+1 > 0’;

D) ‘( )’;

E) ‘96’.

3**.** |ABCDFHTRY| тізбегінің ұзындығы нешеге тең?

A) 0;

B) 1;

C) 10;

D) 9;

E) 11.

I.2.6. Символдық амалдар және олардың қасиеттері

Символдық шамаларда анықталған амалдарды негізінен екі топқа жатқызуға болады:

1) Берілген екі символдық тізбектен бір символдық тізбек құруға мүмкіндік беретін *құрастыру* амалдары;

2) Берілген символдық тізбектен белгілі бір шарт бойынша таңбаны немесе осы тізбектің бөлігін алып тастауға мүмкіндік беретін *бөлу* амалдары.

Құрастыру амалдарының ішінде ең қарапайымы – *конкатенация* (*тіркестіру*) амалы. Бұл амалды «·» таңбасымен белгілесек, онда оны берілген екі символдық тізбек X және У үшін былай анықтауға болады: X–тің мәнінің оң жағына У–тің мәнін тіркегеннен кейін Z тізбегі шығады және ол мына X•Y=Z түрінде жазылады. Мысалы, егер X–тің мәні ‘ҚАЗАҚ’, ал Y–тің мәні ‘СТАН’ болса, онда Z–тің мәні ‘ҚАЗАҚСТАН’ болады. Кейбір қатынас тілінде бұл амалдың таңбасы жазылмайды.

Құрастыру амалына *дизъюнкция* (*таңдау*) амалы да жатады. Бұл амал «|» таңбасымен белгіленеді. Мысалы, A=‘АЛМАТЫ’, B=‘АСТАНА’ болса, онда C = A|B = ‘АЛМАТЫ’ | ‘АСТАНА’.

Құрастыру амалынының тағы да бірі «*» – *итерация* болады. Ол туынды (күрделі) амал және ол *конкатенация* және *дизъюнкция* амалдары арқылы анықталады. Мысалы, кез келген символдық шама X үшін мынаны жазуға болады:

$$X^* = \varepsilon \mid X \mid X^2 \mid X^3 \mid \ldots \mid X^n \mid \ldots,$$

мұнда $X^n = \underbrace{X \cdot X \cdot \ldots \cdot X}_{n}$.

Құрастыру амалдары арқылы символдық өрнекті дұрыс құру үшін, осы амалдардың орындалу ретін білу керек. Олардың реті мынадай: бірінші – итерация амалы «*», екінші – конкатенация амалы «•», үшінші – дюзъюнкция амалы «|». Кейде амалдардың орындалу ретін көрсету үшін жақшалар пайдаланылады. Мысалы, мына $0|10^*$ және $(0|(1(0^*)))$ екі өрнектің мәндері бірдей болады.

Әдетте, *бөлу* амалдары функция түрінде жазылады, яғни, олардың жалпы түрі мынадай болады:

$$\boldsymbol{F}(X_1, X_2,..., X_n),$$

мұнда ***F*** – функцияның атауы, ал $X_1, X_2,..., X_n$ – оның аргументтері.

Бөлу амалдарының ішіндегі ең қарапайымы – берілген тізбектен оның белгілі орнынан (позициясынан) бастап саны белгілі таңбаларды алып тастау амалы. Енді осы бөлу амалының атауын ***БӨЛ*** деп алып, Х тізбегінен саны М болатын таңбаларды К позициясынан алып тастауды былай анықтауға болады.

Егер У-тің мәні Х-тің мәніндегі К позициясынан бастап саны М болатын таңбаларды алып тастағаннан кейін шықса, онда

БӨЛ (Х, К, М) = У.

Енді символдық амалдардың қасиеттерін қарастырайық.

Мейлі α, β және γ – кез келген бос тізбек болмайтын символдық шама және ε – бос тізбек болсын. Сонда символдық шамаларда анықталған құрастыру «•» – *конкатенация* (*тіркеу*), «|» – *дизъюнкция* (*таңдау*) және «*» – *итерация* амалдарының аксиомалары (қасиеттері) I.2.6 - кестеде көрсетіледі.

I.2.6-кесте. Символдық амалдардың қасиеттері.

№	Аксиома	Сипаты
1	$\alpha\cdot\beta \neq \beta\cdot\alpha$	Бос емес тізбелер үшін коммутативтік емес заңы
2	$\alpha\cdot\varepsilon = \varepsilon\cdot\alpha = \alpha$	Бос тізбек үшін коммутативтік заңы
3	$\alpha \mid \beta = \beta \mid \alpha$	Бос емес тізбектердің коммутативтігі
4	$\alpha\cdot(\beta\cdot\gamma) = (\alpha\cdot\beta)\cdot\gamma$	Асоциативтік заңы
5	$\alpha \mid (\beta \mid \gamma) = (\alpha \mid \beta) \mid \gamma$	Асоциативтік заңы
6	$\alpha\cdot(\beta \mid \gamma) = \alpha\cdot\beta \mid \alpha\cdot\gamma$	Дистрибутивтік заңы
7	$(\alpha \mid \beta)\cdot\gamma = \alpha\cdot\gamma \mid \beta\cdot\gamma$	Дистрибутивтік заңы
8	$\alpha \mid \alpha = \alpha$	Редукция заңы
9	$(\alpha^*)^* = \alpha^*$	Редукция заңы
10	$\alpha^* = \varepsilon \mid \alpha^1 \mid \alpha^2 \mid ... \mid \alpha^n \mid ...$	Итерацияның қасиеті
11	$\alpha^+ = \alpha \mid \alpha^*$	Итерацияның қасиеті

I.2.6. Анықтама.

1. Егер β = αξ орындалатындай ξ тізбесі бар болса, онда α тізбесі β тізбесінің *префиксі* (*алды*) деп аталады, ол α⊂β деп белгіленеді, яғни,

$$\alpha \subset \beta \rightleftharpoons \exists \xi(\beta = \alpha\xi).$$

2. Егер β = ζα орындалатындай ζ тізбесі бар болса, онда α тізбесі β тізбесінің *суфиксі* (*арты*) деп аталады, ол β⊃α деп белгіленеді, яғни

$$\alpha \supset \beta \rightleftharpoons \exists \zeta(\beta = \zeta\alpha).$$

3. Егер β = ζαξ болатындай қандай да бір ζ және ξ тізбелері бар болса, онда α тізбесі β тізбесінің *ішкі тізбесі* деп аталады, ол α⊆β деп белгіленеді, яғни $\alpha \subseteq \beta \rightleftharpoons \exists\zeta\exists\xi(\beta = \zeta\alpha\xi)$.

I.2.6. Мысалдар:

1. Егер Х–тің мәні ‘АЛТЫНБЕК’, К=1, а М=5 болса, онда ***БӨЛ*** (Х, К,М) = ***БӨЛ*** (‘АЛТЫНБЕК’, 1, 5) = ‘БЕК’
2. Егер Х–тің мәні ‘АЛА’, ал Ү–тің мәні ‘ТАУ’ болса, онда Х•Ү = ‘АЛАТАУ’ ≠ Ү• Х = ‘ТАУАЛА’.
3. Егер Х–тің мәні ‘ТАУГҮЛ’, К=4, ал М=3 болса, онда ***БӨЛ*** (Х, К,М) = ***БӨЛ*** (‘ТАУГҮЛ’, 4, 3) = ‘ТАУ’.
4. $\varepsilon \subset abcd, a \subset abcd, ab \subset abcd, abc \subset abcd, abcd \subset abcd.$
5. $abcd \supset \varepsilon, abcd \supset d, abcd \supset cd, abcd \supset bcd, abcd \supset abcd.$

I.2.6. Тапсырмалар:

1. Егер А=’АЛА’, В= ’ТАУ’ болса, онда А·В=?,
2. Егер Х= ‘ЖЕРОРТА’, К=4 және М= 4, онда БӨЛ (Х,К,М) = ?
3. Егер Х = ’БУРАБАЙ’, Ү =’БУРА’, онда Х ⊃Үнемесе Ү ⊂Х?

Көмек:

Символдық шамаларда екі топ амалдар анықталған:

1. Құрастыру амалдары;
2. Бөлу амалдары;
3. Тізбелер арасындағы қатынастарды білу керек.

I.2.6. Сұрақтар:

1**.** Символдық амалдардың неше тобы бар?

2**.** Символдық шамалардағы итерация амалы қандай амалдар арқылы анықталады?

3. Бос тізбе деген не?

4. Тізбедегі бөлу амалы деген не?

5. Бөлу амалдарының ішіндегі ең қарапайымы қалай аталады және қалай анықталады?

6. Тізбенің префиксі қалай белгіленеді және қалай анықталады?

7. Тізбенің суфиксі қалай белгіленеді және қалай анықталады?

I.2.6. Тесттер:

1**.** Егер A=’ШЫМ’, B= ’КЕНТ’ болса, онда A·B=?,

A) ШЫМ;

B) ШЫМКЕНТ;

C) КЕНТ;

D) КЕНТШЫМ;

E) ШЫН.

2. Егер X=‘ЖЕЗҚАЗҒАН’, K=4, M=6, онда БӨЛ (X,K,M) = ?

A) ҒАН;

B) ҚАЗ;

C) ЖЕЗҚАЗ;

D) ЖЕЗ;

E) ҚАЗҒАН.

3. $\alpha \mid \beta = \beta \mid \alpha$ аксиоманы сипаттаңыз.

A) Бос емес тізбелер үшін коммутативті емес;

B) Бос тізбек үшін коммутативті;

C) Бос емес тізбелер үшін дистрибутивтік заңы;

D) Бос емес тізбелер үшін ассоциатиивтік заңы;

E) Бос емес тізбелер үшін коммутативті.

I.2.7. Логикалық шамалар түрлері

Логикалық тип логикалық мәндерден құралады. Логикалық мәндерге «*ақиқат*» және «*жалған*» кіреді.

Логикалық мәндер келесі түрде берілуі мүмкін:

1)«*ақиқат*» немесе «*иә*» немесе « *1* » немесе «+»;

2)«*жалған*» немесе «*жоқ*» немесе « *0* » немесе «–»;

Бұл мәндер белгілі бір шарттардың нәтижесі болады. Ондай шарттарға мәндері «*ақиқат*» немесе «*жалған*» болатын тұжырымдар, әлде «*иә*» немесе «*жоқ*» деген жауаптарды талап ететін сұрақтар жатады. Логикалық мәндер ретінде I.1.2 - кестедегі 1, 2 және 3-ші жолдардағы хабарлардың мәндерін алуға болады.

Әртүрлі әдебиет көздерінен «***жалған***» және «***жоқ***» логикалық мәндері (логикалық константа) орнына «***false***», «***non***» немесе «**0**» пайдаланылады, ал логикалық мәндері (логикалық константа) «***ақиқат***» және «***иә***» орнына «***true***», «***yes***» или «**1**» пайдаланылады**.** Сондықтан ары қарай ыңғайлы болу үшін логикалық мәндер ретінде тек 0 мен 1 пайдалануға болады. Сонда кез келген логикалық айнымалылар {0, 1} жиынынан мәндер қабылдайды деп есептейміз.

Ең қарапайым шарт келесі қатынас (салыстыру) амалдары арқылы беріледі: «<» – «*кіші*»; «=» – «*тең*»; «>» – «*үлкен*»; «≤» – «*кіші немесе тең*»; «≥» – «*үлкен немесе тең*».

Күрделі шарттарды құрастыру үшін логикалық мәндерде анықталған логикалық амалдарды қолдануымыз керек. Осындай амалдардың түрі және олардың қасиеттері төмендегі «Логикалық амалдар және олардың қасиеттері» тақырыбында қарастырылады.

I.2.7. Мысалдар:

1. «2 *кіші* 5» деген қарапайым шарттың мәні «ақиқат» болады.

2. «1 тең 3» деген қарапайым шарттың мәні «жалған» болады.

3. «***A*** кіші немесе тең 7» деген қарапайым шарт «ақиқат» немесе «жалған» мәнін ***A*** айнымалы шаманың мәніне тәуелді.

I.2.7. Тапсырмалар:

Мына шарттардың «*ақиқат*» немесе «*жалған*» екенін анықтаңыз

1. 15=20;
2. 2>12;
3. 7<14.

Көмек:

Логикалық мән белгілі бір тұжырымдар, «*иә*» немесе «*жоқ*» деген жауаптарды талап ететін сұрақтар жатады.

I.2.7. Сұрақтар:

1. Логикалық типтерді анықтайтын мәндер?
2. Логикалық мәндер қалай беріледі?
3. «Компьютер адамнан ақылды деген дұрыс емес» деген тұжырым қандай мән қабылдайды?

I.2.7.Тесттер:

1. 45=60 логикалық мәнін анықтаңыз?
   A) Жалған;
   B) Ақиқат;
   C) Жуық;
   D) Тең;
   E) Жорамал.

2. 17<48 логикалық мәнін анықтаңыз?
   A) Ақиқат;
   B) Жалған;
   C) Жуық;
   D) Тең;
   E) Жорамал.
3. –12≥0 логикалық мәнін анықтаңыз?
   A) Ақиқат;
   B) Жалған;
   C) Жуық;
   D) Тең;
   E) Жорамал.

I.2.8. Логикалық амалдар және олардың қасиеттері

Логикалық амалдардың әр түрі бар. Біз солардың ішіндегі ең қарапайымдарын қарастырамыз. Олар: *емес*, *және*, *немесе* амалдары. Осы амалдарды пайдаланып кез келген күрделі логикалық амалдарды анықтап алуға болады, яғни, олар кез келген күрделі шарттарды құрастыруға мүмкіндік береді.

Қолданылатын қатынас тіліне байланысты *логикалық мәндер*, *салыстыру* (*қатынас*) *амалдары* және *логикалық амалдардың* белгілері әр түрлі болуы мүмкін. Мысалы,

1) *Салыстыру* (*қатынас*) *амалдары*:

« = » – «*тең*»;

« > » – «*үлкен*»;

« < » – «*кіші*»;

2) *Логикалық амалдар*:

«¬» – «*инверсия*» немесе «*емес*»

« & » немесе « ∧ » – «*коньюнкция*» немесе «*және*»

« | » немесе « ∨ » – «*дизьюнкция*» немесе «*немесе*»

Кез келген логикалық өрнекте салыстыру амалдары логикалық амалдардан бұрын орындалуы тиіс, ал логикалық амалдар ішінде ең алдымен «*инверсия*» амалы, содан кейін «*конъюнкция*» амалы, ал ең соңында «*дизъюнкция*» амалы орындалады.

Логикалық амалдардың мәндерін (моделдерін) кесте арқылы анықтауға болады. Мысалы, А және В логикалық айнымалылары берілсе, оларды төмендегідей анықтауға болады:

1. *Конъюнкция* (*және*):

A	B	A∧B
0	0	0
1	0	0
0	1	0
1	1	1

Нәтиже тек екі берілгендер «*ақиқат*» болғанда ғана «*ақиқат*» болады, ал басқа жағдайдың бәрінде нәтиже «*жалған*» болады.

2. *Дизъюнкция* (*немесе*):

A	B	A∨B
0	0	0
0	1	1
1	0	1
1	1	1

Нәтиже тек екі берілгендер «*жалған*» болғанда ғана «*жалған*» болады, ал басқа жағдайдың бәрінде нәтиже «*ақиқат*» болады.

1. *Инверсия* (*емес*):

	¬A
0	1
1	0

Егер А–ның мәні «*жалған*» болса, онда оны теріске шығару «*ақиқат*» болады және керісінше.

Логикалық амалдардың қасиеттерін талқылаудың алдында, кез– келген шамалардың типінде анықтауға болатын, *салыстыру амалдарын* қарастырайық. Салыстыру амалын шамалардың мөлшері немесе басқа өлшенетін және бір–бірінен ажыратылатын сипаттарының арасында да анықтауға болады.

Салыстыру амалдарының түрі өте көп. Соған қарамастан, барлық салыстыру амалдарының нәтижесін логикалық мәндер «ақиқат» және «жалған» арқылы белгілеуге болады. Сонымен бірге, барлық салыстыру амалдарының жалпы қасиеттері болады, мысалы, *рефлексивтік*, *симметриялық*, *транзитивтік*:

Мейлі *a*, *b* және *c* шамаларында анықталған ***R*** салыстыру амалы берілсін. Сонда ***R*** салыстыру амалы мынадай қасиеттерге ие:

1) Егер әрбір *a* үшін *a**R**a* орындалса, онда ***R*** *рефлексивтік* болады;

2) Егер әрбір *a* және *b* үшін *a**R**b* орындалғаннан *b**R**a* орындалса, онда ***R*** *симметриялық* болады;

3) Егер әрбір *a*, *b* және *c* үшін ***aRb*** және ***bRc*** орындалғаннан ***aRc*** орындалса, онда ***R*** *транзитивтік* болады.

Енді ***R***-дың орнына бізге бұрыннан белгілі «<» – кіші, «=» – тең және «>» – үлкен деген салыстыру таңбаларын пайдаланып мынадай қасиеттерді жазамыз:

1. Кез келген *a* және *b* шамалары $a < b$, $a = b$ немесе $a > b$ тек қана бір салыстыруында болады:

2**.** Егер $a>b$ және $b>c$ болса, онда $a>c$ болады.

3. Егер $a= b$ және $b= c$ болса, онда $a=c$ болады.

4**.** Егер $a< b$ және $b<c$ болса, онда $a<c$ болады.

Енді логикалық амалдардың қасиеттерін қарастырайық.

Егер *p*, *q* және *r* – кез келген логикалық шама болса және ***0*** – жалған, ***1*** – ақиқат болса, онда логикалық шамаларда анықталған «∧» – *конъюнкция*, «∨» – *дизъюнкция* және «¬» – *инверсия* амалдарының қасиеттері I.2.8 - кестедегідей болады.

I.2.8 -кесте. Логикалық амалдардың қасиеттері.

№	Аксиома	Сипаты
1	$p\wedge q \equiv q\wedge p$	Коммутативтік заңы
2	$p\vee q \equiv q\vee p$	Коммутативтік заңы
3	$p\wedge(q\wedge r) \equiv (p\wedge q)\wedge r$	Ассоциативтік заңы
4	$p\vee(q\vee r) \equiv (p\vee q)\vee r$	Ассоциативтік заңы
5	$p\wedge(q\vee r) \equiv (p\wedge q)\vee(p\wedge r)$	Дистрибутивтік заңы
6	$p\vee(q\wedge r) \equiv (p\vee q)\wedge(p\vee r)$	Дистрибутивтік заңы
7	$\neg(p\vee q) \equiv \neg p\wedge\neg q$	Де Моргана заңы
8	$\neg(p\wedge q) \equiv \neg p\vee\neg q$	Де Моргана заңы
9	$\neg(\neg p) \equiv p$	Екі рет емес заңы
10	$p \equiv p$	Бірдейлік заңы
11	$p\vee\neg p \equiv 1$	Үшіншіні жою заңы
12	$p\wedge\neg p \equiv 0$	Қарама–қайшылық заңы
13	$p\wedge p \equiv p$	Конъюнкция қасиеті

14	$p \wedge 1 \equiv p$	
15	$p \wedge 0 \equiv 0$	
16	$p \wedge (p \vee q) \equiv p$	
17	$p \vee p \equiv p$	Дизъюнкция қасиеті
18	$p \vee 1 \equiv 1$	
19	$p \vee 0 \equiv p$	
20	$p \vee (p \wedge q) \equiv p$	

I.2.8. Мысалдар:

1. A∨¬B ∧ C өрнегінде алдымен ¬B, содан кейін ¬B ∧ C, ал ең соңынан A ∨¬B ∧ C орындалады.

2. A = 0∨ B>1 өрнегінде ең алдымен салыстыру амалдары A = 0 және B>1 орындалады, содан кейін логикалық амал орындалады.

3. ¬(A=0) ∧ B=1 өрнегінде алдымен салыстыру амалдары A=0 және B=1 орындалады, содан кейін логикалық амалдар ¬, ∧ орындалады.

I.2.8. Тапсырмалар:

Мына өрнектегі амалдардың орындап мәнін табыңыз.

1. (1 < 2) ∧ 1=3 ∨ 2 <5;
2. 1>3 ∨ 1<2;
3. 10>13 ∧ 21<22.

Көмек:

Осы амалдар ішінде ең алдымен «инверсия» амалы, содан кейін «конъюнкция» амалы, ал ең соңында «дизъюнкция» амалы орындалады.

I.2.8. Сұрақтар:

1. 2. «&» немесе «∧» – таңбасы қандай логикалық амалды білдіреді?

2. Мына кестедегі бос жерлердің мәндері қандай болады?

A	B	A∨B
0	0	
0	1	
1	0	
1	1	

I.2.8.Тесттер:

1. 2>5 ∨ 2<6 өрнегіндегі амалдардың орындалу реттері бойынша орындағанда, осы өрнектің мәні қандай болады?

A) 2;

B) 1;

C) 5;

D) 6;

E) 0.

2**.** D∨¬F ∧ G өрнегінің орындалу реті қалай болады?

A) алдымен ¬F, содан кейін ¬F ∧ G, ал соңынан D ∨¬F ∧ G орындалады.

B) алдымен ¬F ∧ G, ал ең соңынан D ∨¬F ∧ G орындалады.

C) алдымен ¬F, ал ең соңынан D ∨¬F ∧ G орындалады.

D) алдымен ¬F, содан кейін ¬F ∧ G орындалады.

E) алдымен ¬G, содан кейін ¬G ∧ D, ал ең соңынан D ∨¬F ∧ G орындалады.

3. Де Моргана заңы қайсы?

A) $\neg(\neg p) \equiv p$;

B) $p \equiv p$;

C) $\neg(p \vee q) \equiv \neg p \wedge \neg q$;

D) $p \wedge \neg p \equiv 0$;

E) $p \vee \neg p \equiv 1$.

I.3. Деректер құрылымы

I.3.1. Деректер құрылымын сыныптау

Жоғарыда I.2 блокта біз бөлінбейтін (атомдық) ақпарат бірлігімен сипатталатын бір типті (бүтінсанды, нақтысанды, логикалық, символдық) деректерден құралған қарапайым (базалық) шамаларды қарастырдық. Бірақ өмірде кездесетін көптеген есептер шамалардан-элементтерден құралған құрамды шамаларды талап етеді. Олардың элементтері ретінде қарапайым шамалармен қатар, құрамды шамалар да бола алады. Кейбір әдебиет көздерінде құрамды шамалар деген сөз тіркесінің орнына *құрылымды деректер* немесе *деректер құрылымы* дегенді пайдаланады. Олардың мағыналары бір деп есептеп, біз ары қарай соңғы сөзтіркесті, яғни деректер құрылымын қолданамыз.

Информатикада кездесетін құрылымды шамалар *жиын* негізінде құрылады деп айтуға болады. Себебі, кез–келген құрылымды шамалар белгілі бір тәртіппен ұйымдастырылған қарапайым шамалар *жиынынан* құрастырылады. Олай болса, біз алдымен осы «*жиын*» деген ұғымның анықтамасын, белгілеулерін, онда анықталған амалдардың және олардың қасиеттерін қарастырамыз.

Біз деректер құрылымы деп элементтер жиыны мен олардың арасындағы байланыс жиындарын түсінеміз. Деректер құрылымының көп түрлері бар. Олар былай бөлінеді:

1) элементтер арасында байланыстың болуы бойынша *байланыссыз деректер құрылымы* және *байланысты деректер құрылымы* деп бөлінеді. Байланыссыз деректер құрылымы элементтер арасында байланыстың жоқтығымен сипатталады. Оларға векторлар, жиымдар, жолдар, стектер және кезектер жатады. Байланысты деректер құрылымы элементтер арасында байланыстың барлығымен сипатталады. Оларға бірбайланысты тізімдер, екібайланысты тізімдер, көпбайланысты тізімдер жатады.

2) көлемнің (элементтер санының) өзгермелуі бойынша *динамикалық деректер құрылымы*, *статикалық деректер құрылымы* деп бөлінеді. Динамикалық деректер құрылымында элементтер саны (құрылым қуаты) азайып немесе көбейіп отырады және өңдеудің бір кезінде онда ешқандай элементтің болмауы да мүмкін. Статикалық құрылымның элементтер саны өзгермейді, ол өңдеу басында бір–ақ рет анықталады. Осы деректер құрылымына мыналар жатады: динамикалық – сызықтық байланысқан тізімдер және тармақты байланысқан тізімдер, графтар, дарақтар, стектер, кезектер, дектер; статитискалық – жиымдар, жиындар, жазбалар, кестелер.

3) элементтер арасында байланыстарды ұйымдастыру бойынша *сызықтық деректер құрылымы* және *сызықтық емес деректер құрылымы* деп бөлінеді. Өз кезегінде, сызықтық деректер құрылымы жадта элементтердің өзара орналасуына тәуелді *элементтердің жадта тізбектеліп орналасатын құрылымға* (векторлар, жолдар, жиымдар, стектер, кезектер, дектер, хеш-кестелер) және жадта элементтері тәуелсіз байланысқан таратылған құрылымға (бірбайланысты сызықтық тізім және екібайланысты сызықтық тізім) бөлінеді. Сондай-ақ, сызықтық емес деректер құрылымы элементтерінің өзара байланыс түрі бойынша *иерархиялық деректер құрылымына* (көпбайланысты тізім, екілік дарақтар, *n*-орынды дарақтар, иерархиялық тізім), *торлық деректер құрылымына* (көпбайланысты тізім, жай граф, бағытталған граф), кестелік деректер құрылымына (екі өлшемді жиым, көп өлшемді деректер базасының кестесі) бөлінеді. Иерархиялық деректер құрылымында элемент иерахияда жоғары тұрған элементке сілтеме немесе төмен тұрған элементтерге сілтемелер және өзінің деңгейіндегі сызықтық тізбектегі реттік нөмірмен сипатталады. Торлық деректер құрылымында элементтер өздерінің көршілес элементтеріне байланыстар жиынтығымен сипатталады және ешқандай элемент айқын ерекшеленбейді.

4) бейнелену тәсілі бойынша *физикалық деректер құрылымы мен логикалық деректер құрылымына* бөлінеді. Физикалық деректер құрылымы – деректердің компьютер жадына бейнеленуінің тәсілі. *Логикалық деректер құрылымы* – деректер құрылымын компьютер жадында бейнеленуін ескермей қарастыру. Жалпы жағдайда физикалық және оған сәйкес логикалық құрылымдары арасында айырмашылық бар, оның дәрежесі құрылымның өзіне және ол бейнеленуі тиіс ортаның ерекшелігіне тәуелді. Осыған байланысты логикалық құрылымды физикалық құрылымға бейнелейтін және керісінше, физикалық құрылымды логикалық құрылымға бейнелейтін процедура бар.

5) деректерді сақтау үшін пайдаланатын жад түрі бойынша: *оперативтік жад үшін деректер құрылымы* және *сыртқы жад үшін деректер құрылымына* бөлінеді. Оперативтік жад үшін деректер құрылымы – статикалық және динамикалық жадта орналасқан деректер. Барлық жоғарыда келтірілген дектер құрылымы – оперативтік жад үшін құрылымдар. Сыртқы жад үшін деректер құрылымы файлдық құрылым немесе файлдар деп аталады. Файлдық құрылым мысалы ол тізбекті файл, көпбөлімді файл, екілік дарақтар.

Көрсетілген сыныптау толық емес, сыныптауға кірмей қалған деректер құрылымы бар.

Деректер құрылымының элементтерін компьютер жадында физикалық бейнелеудің екі тәсілі бар: *сыбайлас бейнелеу* және *байланысқан бейнелеу*.

Сыбайлас бейнелеуде кез келген элементтің адресін есептеу үшін барлық жағдайда құрылым дескрипторында болатын элемент нөмірін (индексін) және ақпаратты білсе жеткілікті. Мұнда кез келген элементтің адресі бастапқы адрестен немесе алдыңғы элемент адресінен тура еспетелінеді. Бұл деректер құрылымының элементіне өте жылдам қолжетімді қамтамасыз етеді. Бірақ құрылым элементтерінің логикалық тізбегін өзгерткенде деректерді

жадта жылжытуға тура келеді. Басқаша айтқанда, сыбайлас бейнелеу өте тиімді, бірақ икемсіз тәсіл.

Сыбайлас бейнелеу барлық элементтері біртипті және реттелген тізбек құратын статикалық деректер құрылымына ғана қолданылады. Осындай құрылымдардың элементтерін бүтін сандармен нөмірлеуге болады. Кейде элементтердің нөмірлерін индекстер дейді. Мұндай деректер құрылымына *жолдар*, *векторлар*, *жиымдар* жатады.

Байланысқан бейнелеуде деректер құрылымының элементтері арасында байланыс орнату үшін нұсқағыштар пайдаланылады. Мұнда деректер құрылымының әрбір элементі екі түрлі өріспен қамтамасыз етіледі:

– *деректер өрісі*, онда құрылым жасау үшін қолданылған деректер қамтылады (жалпы жағдайда деректер өрісінің өзі интегралданған құрылым: жазба, вектор, жиым және т.б. болады);

– *байланыстар өрісі*, онда берілген элементті басқа элементтермен байланыстыратын бір немесе бірнеше нұсқағыштар қамтылады.

Байланысқан бейнелеудің келесі артықшылықтары бар:

– деректер құрылымын әжептеуір өзгертуді қамтамасыз ететін мүмкіншілігі бар;

– деректер құрылымының мөлшері тек қолжетімді компьютер жадымен ғана шектеледі;

– құрылым элементтерінің логикалық тізбегін өзгерткенде деректерді жадта жылжытудың қажеті жоқ, оған тек нұсқағыштарды түзеткен жарайды.

Байланысқан бейнелеу келесі кемшіліктерге ие:

– нұсқағыштармен жұмыс істеу индекстермен жұмыс істеуге қарағанда программалаушының жоғары біліктілігін талап етеді;

– байланыстар өрісіне комьютердің қосымша жады жұмсалады, оның көлемі нұсқағыштардың санына тәуелді, ол деректер өрісі үшін бөлінген жадтан көп болып кетуі мүмкін;

– элемент адресі алғашқы деректерден есептеліне алмайды, сондықтан құрылым дискрипторында қамтылған сыртқы нұсқағыштар арқылы құрылымға кіргеннен кейін құрылымның қажетті элементін іздеу нұсқағыштар тізбегімен элементтен элементке өтумен орындалады, ол әжептеуір уақытты талап етеді.

Демек, байланысқан бейнелеу тиімсіздеу, бірақ икемді тәсіл. Сондықтан, байланысқан бейнелеу элементтеріне нөмірлері арқылы қолжетімі бар логикалық құрылымдары жол, вектор, жиым түрінде болатын есептерде еш уақытта қолданылмайды, бірақ логикалық құрылымдары элементтерге қолжетудің басқа тәсілін талап ететін (кестелер, тізімдер, графтар және т.б.) есептерде жиі қолданылады.

I.3.1. Мысалдар:

1. Жолдар байланыссыз және сызықтық құрылымына кіреді.
2. Стектер сызықтық және динамикалық деректер құрылымына жатады.
3. Графтар динамикалық және сызықтық емес деректер құрылымы.

I.3.1. Тапсырмалар:

1. Статикалық деректер құрылымдарын көсетіңіз.
2. Динамикалық деректер құрылымдарын көрсетіңіз.
3. Байланыссыз деректер құрылымдарын көсетіңіз.

Көмек:

1. Динамикалық деректер құрылымы өзінің көлемін өзгертеді.
2. Статикалық деректер құрылымы ештеме өзгерпейді.
3. Байланыссыз деректер құрылымында байланыс жоқ.

І.3.1. Сұрақтар:

1. Деректер құрылымы ұйымдастыру бойынша қандай түрге бөлінеді?

2. Сызықтық емес деректер құрылымы элементтерінің өзара байланысы бойынша қандай түрге бөлінеді?

3. Бейнелену тәсілі бойынша деректер құрылымы қандай түрге бөлінеді?

І.3.1. Тесттер:

1. Жолдың қандай белгісі бар?

A) тізбектілік;

B) иерархиялық;

C) динамикалық;

D) байланыстық;

E) сызықтық емес.

2. Жиым деген не?

A) әртипті реттелген элементтер жиыны;

B) біртипті реттелмеген элементтер жиыны;

C) біртипті реттелген элементтер жиыны;

D) бірбайланысты элементтер жиыны;

E) байланыссыз элементтер жиыны.

3. Деректер құрылымы деген не?

A) ішкі және сыртқы жадтардың алмасу деректерінің жиыны;

B) элементтер деректерінің жиыны және олардағы амалдар жиыны;

C) элементтер деректерінің жиыны және оларға мәндерді меншіктеу тәсілдерінің жиыны;

D) элементтер деректерінің жиыны және оларға мәндерін өзгерту тәсілдерінің жиыны;

E) элементтер деректерінің жиыны және олардың арасындағы байланыстар жиыны.

I.3.2. Жиындар

Информатикада «жиын» ұғымы өте маңызды роль атқарады. Бірақ ол, «ақпарат» ұғымы сияқты, дәл анықталмайды және алғашқы (аксиоматикалық) ұғымдардың біріне қабылданады, яғни, бұл ұғым басқа ұғымдарға айналмайды. Сондықтан «жиын» ұғымына келесі интуитивті анықтама беруге болады.

Жиын – біртұтас болып ойландыратын қайсыбір белгімен (қасиетпен) ажыралатын объектілер жинағы.

Жиында қамтылатын объектілер *жиын элементтері* деп аталады.

Жиын үлкен әріптен, ал оның элементтері кіші әріптермен немесе араб цифрларымен белгіленеді. Мысалы, натурал сандарының жиыны ***N*** әріпімен, ал оның элементтері 1, 2, 3,… араб цифрларымен белгіленеді.

Жиын анықтамасын мына түрде жазуға болады:

<*жиынның атауы*> = {<*жиынның анықтамасы*>},

мұнда жиын элементтері ирек { және } жақшаның ішінде көрсетіледі.

Жиын анықтамасы екі түрде беріледі:

1) жиынның барлық элементтерін көрсету арқылы анықтау;

2) жиының барлық элементтерін сипаттау арқылы анықтау.

Барлық жиындар ішінде екі жиын айырықша ерекшеленеді:

1) ***Ø*** *– бос жиын* бір де бір элементті қамтымайды;

2) ***U*** *– универсал жиын* (*универсум*) қарастыралатын типтегі (пәндік облыстағы) барлық элементтерді қамтиды.

Мысалы, универсум болып:

1) сандар теориясында – барлық бүтін сандар жиыны;

2) тілдер теориясында – әліпбидегі барлық сөздер жиыны;

3) геометрияда – берілген геометриялық кеңістікте барлық нүктелер жиыны.

Жиын мен элемент арасында қатынас амалдарын анықтауға болады. Солардың ішіндегі негізгісі: белгілі бір элемент белгілі бір жиынға кіретіндігін білдіретін қатынас, оны «$\in$» таңбасымен, ал

осыған кері қатынасты «$\notin$» таңбасымен белгілейді. Мысалы, $a \in A$ жазуы белгілі a элементі белгілі бір A жиынына кіреді, немесе A жиынының элементі a дегенді немесе A жиынының элементі a болады дегенді, ал жазу $x \notin A$ керісінше A жиынына x кірмейді дегенді білдіреді.

Сондай-ақ, осындай қатынасты жиындар арасында анықтауға болады: A жиыны B жиынының ішжиыны болады, егер A жиынына кіретін кез-келген элемент B жиынына кірсе:

$$(A \subset B) \Leftrightarrow (x \in A \Rightarrow x \in B).$$

A жиынының *қуаты* деп оның элементтерінің санын айтады, ол $|A|$ арқылы белгіленеді, $|\varnothing| = 0$.

I.3.2. Ескерту.

1. Бос жиын кез-келген бос емес A жиынының ішжиыны болады, яғни. $\varnothing \subset A$, $|\varnothing| < |\mathbf{A}|$.

2. Натурал сандар жиыны N-*ң* қуаты шексіз, яғни $|N| = \infty$.

Кейбір жайтты орнату үшін «*есепті жиын*» ұғымын аламыз. Жиын есепті болады, егер оның және натурал сандар жиынының арасында өзара бірмәнді сәйкестік бар болса, яғни есепті жиын элементтерін натурал сандармен нөмірлеп шығуға болады. Бұл есепті жиын мен натурал сандар жиыны теңқуаттылығын көрсетеді.

I.3.2. Мысалдар:

1. $X = \{x \mid x>0\}$, мұнда X жиынының элементтері тек оң айнымалы шамалар x болады.

2. Егер $A = \{a, b, c, d, e\}$, онда $|A| = 5$.

3. $N = \{n \mid n \in Z \ \& \ n>0\}$ – натурал сандар жиыны, мұнда Z бүтін сандар жиыны.

I.3.2. Тапсырмалар:

Мейлі A = $L \cup D \cup S$ – компьютерде қолданылатын барлық таңбалар жиыны болсын, мұнда L – латынның әріптерінің жиыны, D – араб цифрларының жиыны, S – барлық арнаулы таңбалар жиыны болса:

1. L-дің барлық элементтерін анықтаңыз;

2. D-нің барлық элементтерін анықтаңыз;

3. S-тің барлық элементтерін анықтаңыз.

Көмек:

1 және 2 тапсырмаларда жиынның барлық элементтерін көрсету арқылы, 3 тапсырмада жиынның барлық элементтерін сипаттау арқылы анықтауды қолдану керек.

I.3.2. Сұрақтар:

1. Жиынға кіретін элементтердің саны қалай аталады?
2. Жиынды қалай анықтауға болады?
3. Бос тізбелер жиынының ұзындығы неге тең?

I.3.2. Тесттер:

1. Егер L жиыны латын әліпбиінің кіші әріптерінен ғана тұрса, яғни, $L = \{a, b, c, d, e, f, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z\}$, онда $|L|$ неге тең?

A) 1;
B) 26;
C) 28;
D) 0;
E) 30.

2. Егер $D = \{d \mid d$ – бүтін сан және $0 \leq d \leq 9$ орындалса$\}$, яғни, $D=\{0,1,2,3,4,5,6,7,8,9\}$ болса, онда $|D|$ неге тең?

A) d;
B) 9;
C) 10;
D) 0;
E) 1.

3. Белгілі бір элемент белгілі бір жиынға кіретіндігін білдіретін қатынас қандай таңбамен белгіленеді?

A) $\cap$;
B) $\notin$;
C) $\in$;
D) $\varnothing$;
E) &.

I.3.3. Жиындардағы амалдар

Жиындарда бірнеше амалдар анықталған. Мысалы, егер X және Y жиындары берілсе, онда олар үшін мынадай амалдарды анықтауға болады:

1. *Бірігу амалы*: X жиыны мен Y жиынының бірігуі деп, элементері осы екі жиынның біреуінің элементтерінен тұратын үшінші Z жиынының пайда болуын айтады және оны былай белгілейді:

$$X \cup Y = Z = \{z \mid z \in X \vee z \in Y\}.$$

2. *Қиылысу амалы*: X жиыны және Y жиынының қиылысуы деп, элементері осы екі жиынның ортақ элементтерінен тұратын үшінші Z жиынының пайда болуын айтады және оны былай белгілейді:

$$X \cap Y = Z = \{z \mid z \in X \wedge z \in Y\}.$$

3. *Айырым амалы*: X жиыны мен Y жиынының айырымы деп, элементері X жиынының элементтерінен тұратын үшінші Z жиынының пайда болуын айтады және оны былай белгілейді:

$$X \setminus Y = Z = \{z \mid z \in X \wedge z \notin Y\}.$$

4. *Симметриялық айырмасы*: X және Y жиындарының симметриялық айырмасы тек қана X жиынының элементтерінен немесе Y жиынының элементтерінен тұрады, ол былай белгіленеді:

$$X \triangle Y = \{x \mid (x \in X \wedge x \notin Y) \vee (x \in Y \wedge x \notin X)\}.$$

5. *Тік көбейту амалы*: X жиыны және Y жиынының тік көбейтуі деп, элементері осы екі жиынның элементтерінен құралған *қосақ* болып келетін үшінші Z жиынының пайда болуын айтады және оны былай белгілейді:

$$X \times Y = Z = \{z \mid z = (x, y) \wedge x \in X \wedge y \in Y\}.$$

Жиындарда анықталған осы амалдардың қасиеттері және олардың сипаттары I.3.2-кестеде көрсетілген.

I.3.2-кесте. Жиындық амалдардың қасиеттері.

№	Аксиома	Сипаты
1	$X \cup Y = Y \cup X$	Бірігу қасиеттері
2	$X \cup X = X$	
3	$X \cup \varnothing = \varnothing \cup X = X$	
4	$X \cup (Y \cup Z) = (X \cup Y) \cup Z = X \cup Y \cup Z$	
5	$X \cap Y = Y \cap X$	Қиылысу қасиеттері
6	$X \cap X = X$	
7	$X \cap \varnothing = \varnothing \cap X = \varnothing$	
8	$X \cup (Y \cup Z) = (X \cup Y) \cup Z = X \cup Y \cup Z$	Ассоциативтік заңы
9	$X \cap (Y \cap Z) = (X \cap Y) \cap Z = X \cap Y \cap Z$	
10	$X \cup (Y \cap Z) = (X \cup Y) \cap (X \cup Z)$	Дистрибутивтік заңы
11	$X \cap (Y \cup Z) = (X \cap Y) \cup (X \cap Z)$	
12	$X \setminus Y \neq Y \setminus X$	Айырым қасиеттері
13	$X \setminus X = \varnothing$	
14	$X \setminus \varnothing = X$	
15	$X \setminus (Y \setminus Z) = (X \setminus Y) \setminus Z = X \setminus Y \setminus Z$	
16	$X \triangle \varnothing = X$	Нөл қасиеті
17	$X \triangle X = \varnothing$	Идемпотенттік заңы
18	$X \triangle Y = (X \cup Y) \setminus (X \cap Y$	Симметриялық айырма
19	$X \triangle Y = Y \triangle X$	Коммутативтік заңы
20	$(X \triangle Y) \triangle Z = X \triangle (Y \triangle Z) = X \triangle Y \triangle Z$	Ассоциативтік заңы
21	$X \times Y \neq Y \times X$	Тік көбейту қасиеттері
22	$X \times X = X \times X$	
23	$X \times \varnothing = \varnothing \times X = \varnothing$	
24	$X \times (Y \times Z) = (X \times Y) \times Z = X \times Y \times Z$	

A жиыны және B жиынының арасында анықталған кез келген *қатынас* осы екі жиынның тік көбейтіндісінің ішжиыны болады, мысалы, егер осындай қатынасты «~» таңбасымен, ал ішжиын дегенді «$\subseteq$» таңбасымен белгілесек, онда былай жазуға болады:

$$A \sim B \subseteq A \times B .$$

Мейлі A және B қандай-да бір жиындар болсын. Сонда осы A

және B арасында келесі қатынастарды анықтауға болады:

1. A ***қамтылады*** B, егер A жиынының әрбір элементі B жиынына да кірсе:

$$A \subseteq B \Leftrightarrow \forall a \in A: a \in B.$$

2. A ***қамтиды*** B, егер B қамтылады A:

$$A \supseteq B \Leftrightarrow B \subseteq A.$$

3. A ***тең*** B, егер A және B бір-біріне қамтылса:

$$A = B \Leftrightarrow (A \subseteq B) \wedge (B \subseteq A).$$

4. A ***қатал қамтылады*** B, егер A қамтылса B, бірақ тең емес:

$$A \subset B \Leftrightarrow (A \subseteq B) \wedge (A \neq B).$$

5. A ***қатал қамтиды*** B, егер B қатал қамтылады A:

$$A \supset B \Leftrightarrow B \subset A.$$

6. A және B ***қиылыспайды***, егер олардың ортақ элементтері болмаса: A және B қиылыспайды $\Leftrightarrow \forall a \in A: a \notin B$.

7. A және B ***жалпы жағдайда болады***, егер тек қана A жиынына кіретін элемент, тек қана B жиынына кіретін элемент және екі жиынға да кіретін элемент бар болса:

$$A \text{ және } B \text{ жалпы жағдайда болады} \Leftrightarrow$$
$$\exists a, b, c: (a \in A) \wedge (a \notin B) \wedge (b \in B) \wedge (b \notin A) \wedge (c \in A) \wedge (c \in B).$$

Егер әрбір $y \in Y$ элементі тек бір ғана $x \in X$ элементіне сәйкес болуы керек деген талап қойылатын болса, онда X пен Y жиындарының арасындағы қатынасты *функция* деп атайды.

Егер функцияны f арқылы белгілесек, яғни $f \subseteq X \times Y$ болса, онда $f: X \to Y$ деп жазамыз немесе $y \in Y$ элементі $x \in X$ элементіне сәйкес қойылады дегенді былай жауға болады $y = f(x)$.

Жиын элементтеріне *индекстер* беруге, бүтін сандарды сәйкестендіріп қосақтауға болады. Жиын элементтеріне индекстер беруді *индекстеу* деп, ал осындай жиынды *индекстелінген жиын* деп атайды. Ол функция арқылы беріледі. Мысалы, индекстеуді I арқылы белгілесек, онда X жиынының элементтерін индекстеу үшін мынадай $I: X \to \{1,2,3,...,|X|\}$ функция алуға болады. Сонда X жиынын былай $X = \{x_1, x_2, x_3, ..., x_{|X|}\}$ жазуға болады.

Индекстеу жиын элементтерінің әрқайсысына индекс мәнін көрсету арқылы сілтеме жасауға мүмкіндік береді.

Жиын элементтері сұрыпталған болуы мүмкін. Ол үшін, мысалы, «≤» таңбасымен белгіленген қатынас амалы анықталуы керек. Сонда *X* жиынын *толық сұрыпталған* дейді, егер де мына шарттар орындалса:

1) $x_i \le x_j$ немесе $(x_i, x_j) \in \le$;

2) кез–келген x_i және x_j үшін $x_i \le x_j$ немесе $x_j \le x_i$, $(i \neq j)$;

3) $x_i \le x_j$ және $x_j \le x_k$ болса, онда $x_i \le x_k$, $(i \neq j \neq k)$.

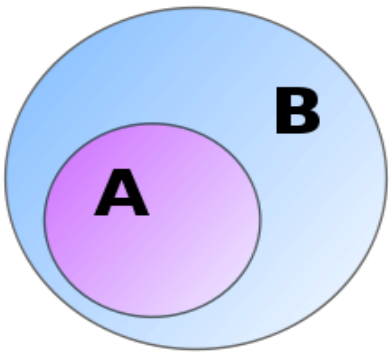
X жиыны *жартылай сұрыпталған* болады, егер де осы шарттардың ішінде тек 2)-ші шарт орындалмаса.

I.3.3. Мысалдар:

1. 7-ден кіші натурал сандар жиыны *A* былайша жазылады:
   A={*x*/*x* – натурал сан, *x*<7} немесе A={1, 2, 3,4, 5, 6};
2. 16061952 санының әр түрлі цифрларынан тұратын жиын былай жазылады {0, 1, 2, 5, 6, 9};
3. «информатика» деген сөздегі әр түрлі әріптер жиыны былай жазылады {а, и, к, м, н, о, р, т, ф}.

I.3.3. Тапсырмалар:

1. 3х–7=3(х+5) теңдеуінің түбірлерінің жиынын іздестіру керек.
2. *A* – сіздің мектептегі барлық оқушылар жиыны, ал *B* – сіздің сыныптағы оқушылар жиыны болсын. *B* жиынының атауын табыңыз.
3. Суреттегі A және B жиындарының арасындағы қатынасты атаңыз.

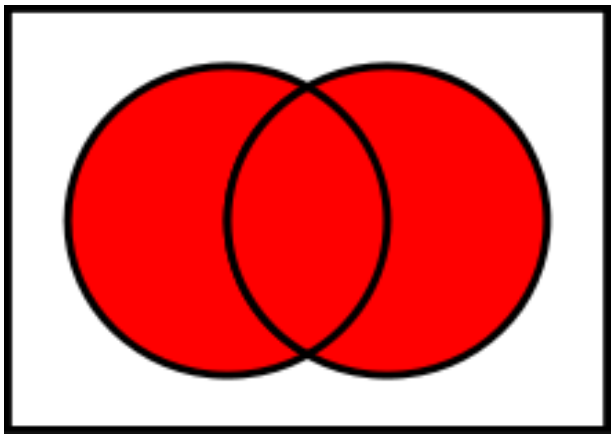


Көмек:

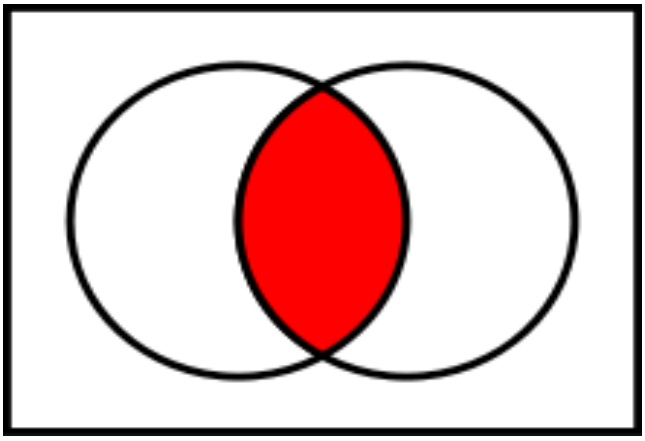
1. Берілген теңдеу қажетті түрлендірулер арқылы шешіледі.
2. *B* жиыны *A* жиынының бөлігі болады.
3. *A* жиыны *B* жиынының бөлігі болады.

I.3.3. Сұрақтар:

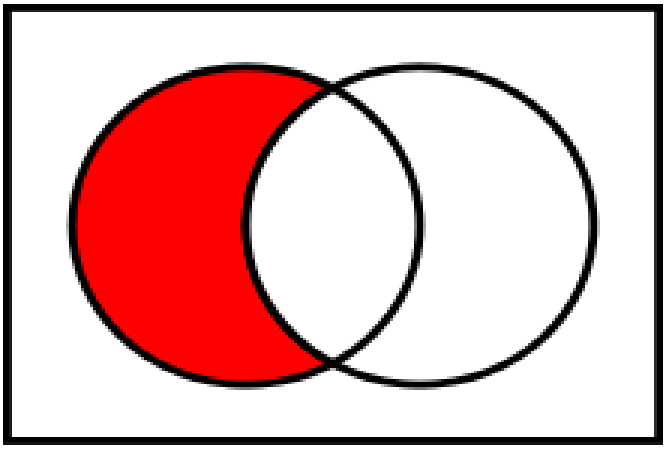
1. 1. Суретте қандай амал көрсетілген?



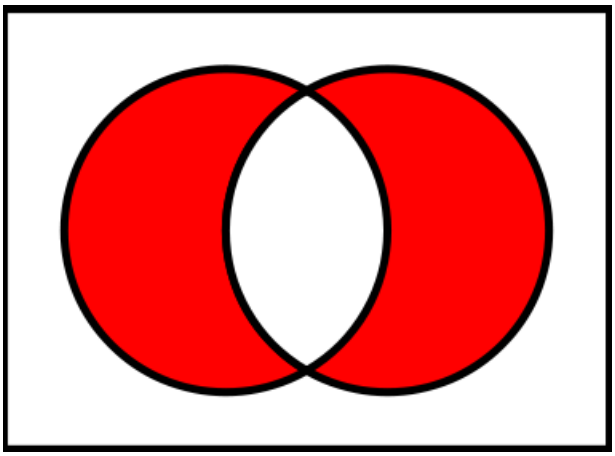
2. Суретте қандай амал көрсетілген?



3. Суретте қандай амал көрсетілген?



4. Суретте қандай амал көрсетілген?



I.3.3. Тесттер:

1. Берілген $A=\{0, 2, 4, 6\}$ және $B=\{-2, -1, 0, 1, 2\}$ жиындары үшін $A\cap B$ амалының нәтижесі қандай болады?

A) $\{0, 2\}$;
B) $\{0, 2, 4, 6\}$;
C) $\{-2, -1, 0, 1, 2\}$;
D) $\{0, 2, 4, 6, -2, -1, 0, 1, 2\}$;
E) $\varnothing$.

2. Берілген $A=\{1, 3, 5\}$ және $B=\{2, 4, 6, 8\}$ жиындары үшін $A\cup B$ амалының нәтижесі қандай болады?

A) $\{1, 2, 3, 4, 5, 6, 8\}$;
B) $\{1, 3, 5\}$;
C) $\{2, 4, 6, 8\}$;
D) $\varnothing$;
E) $\{1, 3, 8\}$.

3. Берілген $A=\{а, б, в, г, д, е, ж\}$ және $B=\{г, е, ж\}$ жиындары үшін $A\setminus B$ амалының нәтижесі қандай болады?

A) {а, б, в, г, д, е, ж};
B) {а, б, в, д};
C) {г, е, ж, з};
D) { г, е};
E) $\varnothing$.

I.3.4. Жолдар, жиымдар және кестелер

Жолдар – ақырлы жиын (әліпби) элементтерінен (символдарынан) тұратын статикалық байланыссыз сызықтық деректер құрылымы. Мысалы, егер әліпби ретінде осы кітапта қолданылған барлық әріптерден, цифрлардан, тыныс белгілерінен, амал таңбаларынан, әртүрлі жақшалардан және басқа символдардан тұрады десек, онда оны жол деп қарастыруға болады.

Бір жолды екінші жолдан ажырату үшін арнаулы символ-бөлгіш қолданылады немесе әліпби ауыстырылады.

Жолдардың мынадай қасиеттері бар:

1. Жол индекстелінген емес және сұрыпталған болуы қажет, себебі керекті таңбаға қолжетім тізбенің белгілі бір жағынан басталады.

2. Егер жолды белгілі бір қатынас тілінің сөйлемі ретінде қарасақ, онда осындай жолды құруға арналған белгілі грамматикалық ережелер болуы қажет, себебі бұл жағдайда жолдың ішінде, тілдің грамматикалық ережесіне сәйкес, қосымша құрылым бар болады.

Жолды өңдеген кезде бізге белгілі бір таңбаны табу немесе осы таңбаның жол ішіндегі кездесу жиілігін анықтау немесе осы таңбаны басқа таңбаға ауыстыру сияқты әрекеттерді жасау керек.

Жолда анықталған кез–келген күрделі амал *таңбаның орнын табу*, *таңбаны қою*, *таңбаны жою* деген (сандарда анықталған *қосу* мен *алу* амалдары сияқты) ең қарапайым амалдар комозициясынан тұрады. Мысалы, екі таңбаның орнын ауыстыру үшін екі рет таңбаның орнын табу, екі рет таңбаны жою және екі рет таңбаны қою амалдары орындалуы керек.

Кестелер – индекстелінген бір жиынның немесе бірнеше жиындардың тік көбейтіндісінің элементерінен құрылған статикалық байланыссыз сызықтық деректер құрылымы.

Егер кесте бір ғана жиынның элементтерінен құрылса, онда оны *бір өлшемді кесте*, ал бірнеше (екі немесе одан да үлкен) жиындардың тік көбейтіндісінің элементтерінен құрылса, онда оны

көп өлшемді кесте дейміз. Өлшем саны элементтер индексінің санын көрсетеді. Сондықтан, элементтер бір индексті, екі индексті, үш индексті және т.с.с. болады. Мысалы, бір өлшемді *A*, екі өлшемді *B*, екі өлшемді *C* кестелерінің индекстері былай болады:

$$A=(a_1,a_2,a_3), B=(b_{11},b_{12},b_{21},b_{22},b_{31},b_{33}), C=(c_{111},c_{112},c_{121,},c_{122},c_{211},c_{212},c_{221},c_{222}).$$

Көп өлшемді кестенің әрбір элементіне оның индекстерінің мәндерін көрсетіп қолжетім жасауға болады. Мысалы, екі өлшемді алты элементі бар *B* кестесінің элементтерін былай белгілеуге болады: $b_{11},b_{12},b_{21},b_{22},b_{31},b_{33}$.

Егер кестенің барлық элементтері бір типті болса, онда мұндай кестені *жиым* деп атайды. Жиым бір өлшемді болғанда оны кейде *вектор* деп атайды.

Жиымда элементтің компьютер жадында бірмәнді орналасу орны элементтердің бір типтілігімен қаматамасыз етіледі және ол оның индексі(тері) мен жад ұяшығының мөлшерінің көбейтіндісімен анықталады. Бір өлшемді жағдайда элементтің жадтағы орналасу орны мына формуламен есптеледі:

Адрес(*элемент*(*index*)) = *index* ұяшық_мөлшері*

Кестелі деректерді сақтау кезінде ажыратқыштар пайдаланылады, екі өлшемді кестеде екі түрлі ажырату болады: *тік ажыратқыш* және *көлденең ажыратқыш*. Егер екі өлшемді кестені символдық жол түрінде сақтау қажет болса, онда бір символ-ажыратқышты бір жолға жататын элементтер арасында, ал екінші ажыратқышты жолдарды ажырату үшін қолданады.

Кестені элементерінің типтеріне сәйкес анықталған сандық, символдық және логикалық амалдарды орындап өңдеуге болады. Сонымен қатар, кестені сұрыптауға және оның белгілі бір шартты қанағаттандыратын элемент(тер)ін табуға немесе олардың мәндерін өзгертуге болады.

I.3.4. Ескерту.

Кесте элементтері атомдық емес құрылымды болуы мүмкін, яғни бір деректер құрылымын екінші деректер құрылымына салуға болады. Егер кесте элементі қайтадан кесте болса, онда кестелердің салымдығы туралы айтуға болады. Осындай салымды деректер

құрылымы арқылы иерархиялық және торлық деректер құрылымын жүзеге асыруға болады.

I.3.4. Мысалдар:

1. Мейлі $A = \{0, 1\}$ жиыны берілсін. Сонда мына тізбе 111101010101 A жиынында анықталған жол болады және ол 0 мен 1-де орындалған конкатенеция (тіркем) амалымен құрылады.

2. Мына вектор $d(1)$ $d(2)$ $d(3)$ $d(4)$ $d(5)$ түрінде бес біртипті заттар (элементтер) беріледі, мұнда $d(i)$, $i = 1,2,3,4,5$ осы қатардағы нақты элементті көрсетеді.

3. MS Word, MS Excel, MS Access дестелеріне кестелер екі өлшемді кестелер болады, оның элементі жол индексі мен баған индексінің қиылысында болады.

4. Басу (экранға шығару) кезінде екі өлшемді кестелерде тік ажыратқыш және көлденең ажыратқаш ретінде сызық қолданылады. Төменде күн жүйесінің планеталары көрсетілген:

Планета	**Күнге дейінгі арақашықтық**	**Салыстырмалы масса**	**Серіктер саны**
Меркурий	0,39	0,056	0
Венера	0,67	0,88	0
Земля	1,0	1,0	1
Марс	1,51	0,1	2
Юпитер	5,2	318	16

Күн жүйесінің планеталары кестесінің символдық жол түріндегі жазылуы:

Меркурий0,39*0,056*0#Венера*0,67*0,88*0#Земля*1,0*1,0*1#Марс*1,51*0,1*2#Юпитер 5,2*318*16#

Мұнда * – элементтер арасындағы символ-ажыратқыш, # – жолдар арасындағы символ-ажыратқыш.

I.3.4.Тапсырмалар:

1. Ел астанасының атын жол түрінде жазыңыз және ол құралған элементтер қай әліпбиге жататындығын айтыңыз.

2. Екі өлшемді жиымның нақты мысалын математикадағы құрылым арқылы көрсетіп, оған математикалық амал қолданыңыз.

3. Біртоптастарыңның кестесін тұрғызыңыз, онда тегі-аты, жасы, жынысы, ұлты көрсетілуі керек.

Көмек:

1. Әліпби қолданылатын тілге тәуелді.
2. Екі өлшемді жиымда жол және баған болады.
3. Бұл үшін MS Word-ғы кестені пайдаланса болады.

I.3.4. Сұрақтар:

1. Бір өлшемді кесте деген не?
2. Екі өлшемді кесте деген не?
3. Көп өлшемді жиымда өлшем мөлшері неге тең?

I.3.4. Тесттер:

1. Жолдағы күрделі амал қандай қарапайым амалдардан тұрады?

A) табу, қою, жою;
B) бірігу, қиылысу, толтыру;
C) қосу, алу, бөлу;
D) конкатенация, итерация, интеграция;
E) проекция, бөлу, мультипликация.

2. Жиымдағы өлшем саны неге тең?

A) индекстер санына;
B) элементтер санына;
C) индекстер мәндерінің көбейтіндісі;
D) индекстер мәндерінің кішісі;
E) индекстер мәндерінің үлкені.

3. Хеш-кестенің элементін табу үшін не көрсету керек?

A) кілттер, мәндер;
B) тік индекстер, көлденең индекстер;
C) кілттер, индекстер;
D) индекстер, мәндер;
E) элементтің тура адресі, кілттер.

I.3.5. Тізімдер

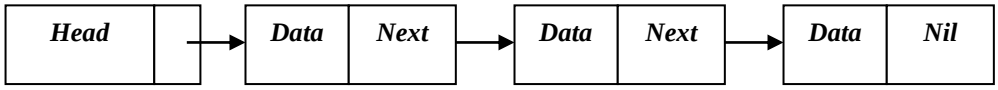
Тізім – айнымалы сан элементтері бар реттелген деректер құрылымы.

Тізімдер типтелген немесе типтелмеген болуы мүмкін. Егер тізім типтелген болса, онда оның элементтерінің типі берілген болады және оның барлық элементтері тізім элементтерінің берілген типімен үйлесімді типке ие болуы керек.

Тізім сұрыпталған және сұрыпталмаған болуы мүмкін. Егер тізім элементтер арасында көршілес қатынасын көрсетсе, онда ол *сызықтық тізім* деп аталады. Егер тізімде мөлшерге (ұзындыққа) шектеу қойылмаса, онда ол жадта байланысқан құрылым түрінде бейнеленеді. Жүзеге асыруға тәуелді тізім элементіне қандай да бір қолжетім болуы мүмкін.

Сызықтық бірбайланысты тізім – бір-бірімен нұсқағыштар арқылы тізбекті баланысқан бір типті элементтерден тұратын динамикалық сызықтық деректер құрылымы. Тізімнің әрбір элементінің келесі элементке нұсқағышы бар. Соңғы элемент нұсқағышы ***Nil*** болады. Әрбір сызықтық бірбайланысты тізімде әдетте пішімі жағынан тізімнің басқа элементінен бөлек тізім басы деп аталатын ерекше элемент болуы керек.

Графикалық бейнелеуде тізімдегі байланыс бағыттама арқылы кескінделеді. I.3.5.1-суретте бірбайланысқан тізім көрсетілген.



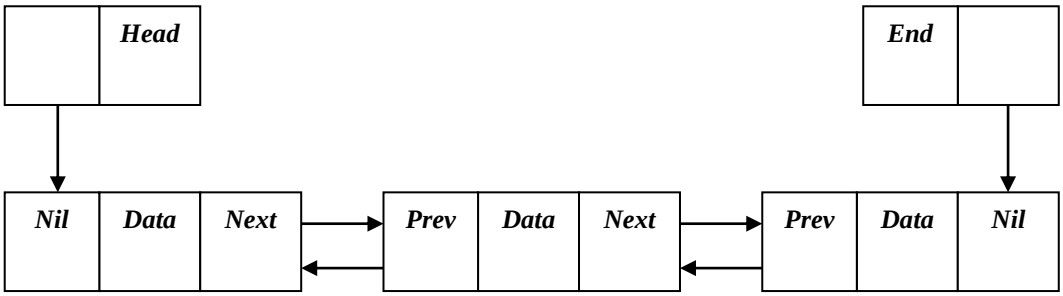
I.3.5.1-сурет. Бірбайланысқан тізім құрылымы.

Мұнда ***Head*** – тізім басы, ***Data*** –деректер, ***Next*** – тізімнің келесі элементіне нұсқау, ***Nil*** – тізім соңының белгісі.

Байланысқан тізім элементтерінің реті элементтердің компьютер жадында орналасу ретімен сәйкес болмауы мүмкін, ал тізімді орағыту реті әрдайым тізімнің ішкі байланыстарымен айқын беріледі. Сызықтық бірбайланысты тізімді өңдеу әрдайым ыңғайлы емес, себебі қарсы бағытта жылжу мүмкіндігі жоқ. Мұндай мүмкіндікті сызықтық екібайланысты тізім береді, онда екі жаққа

жылжу қарастырылған: бастан соңына дейін және керісінше.

Сызықтық екібайланысты тізім – әрбір элементі екі нұсқағышты қамтиды: келесіге және алдындағысына динамикалық сызықтық байланысқан деректер құрылымы. Тізімді ыңғайлы өңдеу үшін тағы да бір ерекше элемент қосылады – тізім соңының нұсқағышы. Әрбір элементте екі нұсқағыштың болуы тізімді қиындатады және жадтың қосымша шығынын әкеледі, бірақ тізімдегі кейбір амалдарды тиімді орындауды қамтамасыз етеді. Сызықтық екібайланысты тізім құрылымы I.3.5.2-суретте көрсетілген.



I.3.5.2. Сурет. Сызықтық екібайланысты тізім құрылымы.

Мұнда ***Head*** – тізім басы, ***Data*** – деректер, ***Prev*** – алдыңғы элементке нұсқағыш, ***Next*** – келесі элементке нұсқағыш, ***Nil*** – тізім соңының белгісі.

Сақиналық тізім – сызықтық бірбайланысты немесе екібайланысты тізімдер соңғы элементтің нұсқағышы бірінші элементке нұсқайтын түрі.

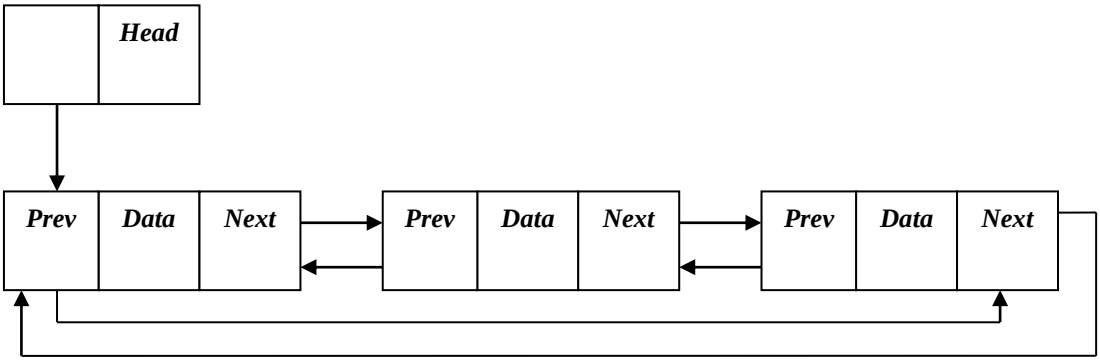
Егер сақиналық тізім сызықтық екібайланысты тізім негізінде ұйымдастырылса, онда бірінші және соңғы элементтерде нұсқағыштар қайталан анықталады. Мұндай тізімдермен жұмыс істеген кезде тізімде орындалатын кейбір процедуралар бірталай жеңілдетіледі. Бірақ мұндай тізімді қарау кезінде шексіз қайталауға түсіп қалмау үшін кейбір қауіпсіздік шараларды қабылдау керек.

Жадта тізім дескриптор және мөлшерлері мен пішімдері бірдей бір-бірімен нұсқағыштар арқылы сызықтық реттелген тізбекке байланысып, жадтың қандай да бір аумағында орналасқан жазбалардың жинағы.

Жазба деректер өрісін және тізімдегі көрші элементке

нұсқағышты қамтиды. Мұнда кейбір деректер өрістері тізім элементіне қатысты қосымша ақпараты бар жад блоктарының нұсқағыштары бола алады. Тізім дескрипторы ерекше жазба түрінде жүзеге асырылады және тізім туралы мынадай ақпаратты қамтиды: тізім басының адресі, құрылым коды, тізім аты, тізімнің ағымдағы саны, элемент сипаты және т.б. Дескриптор жадтың тізім элементтері орналасқан аумақта болады немесе оған жадта басқа орын бөлінеді.

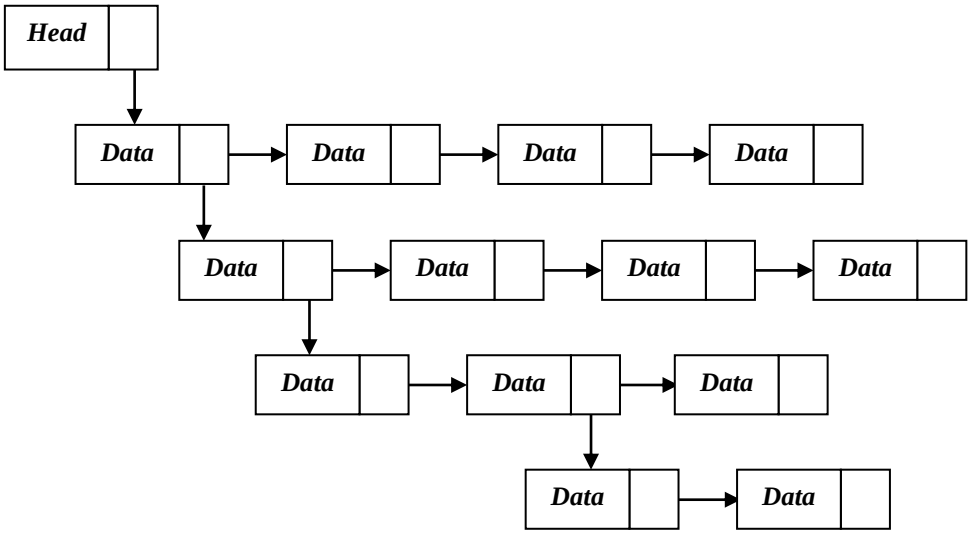
Сақиналық екібайланысты тізім құрылымы I.3.5.3-суретте көрсетілген.



I.3.5.3-сурет. Сақиналық екібайланысты тізім құрылымы.

Иерархиялық тізім – сызықтық тізіммен дарақтың қосындысы. Тізімнің әрбір элементі иерахияның келесі ішдеңгейіндегі тізім басы бола алады және ол көпбайланысты болуы керек.

Иерархиялық тізім құрылымы I.3.5.4-суретте көрсетілген.



I.3.5.4. Сурет. Иерархиялық тізім құрылымы.

Тізімдерде келесі амалдар анықталған:

– тізімнің бас элементін есептейтін амал;

– тізімге қолжетім амалы, ол алғашқы тізімнің, бас элементінен басқа, барлық элементтерінен тұрады;

– тізімге элементті қосу амалы(басына, соңына немесе кез келген элементтен кейін тізімнің ішіне);

– тізімнің ұзындығын (элементтер санын) есептейтін амал;

– тізімнің бостығын тексеру амалы.

Байланысқан тізімдердің артықшылықтары және кемшіліктері болады. Олардың артықшылықтарына мыналар жатады:

– элементтерді қосу мен жоюдың жеңілділігі;

– мөлшер тек компьютер жадының көлемімен және нұсқағыштардың разрядтылығымен ғана шектеледі;

– элементтерді динамикалық қосу және жою.

Байланысқан тізімдердің келесі кемшіліктері бар:

– тізімдегі сілтеме бойынша элемент адресін анықтау қиындығы (жиымда нөмір немесе индекс бойынша);

– нұсқағыш өрісіне (алдыңғы элемент және келесі элемент нұсқағышы) қосымша жад жұмсалады (жиымда нұсқағыштар қажет емес);

– тізіммен жұмыс істеу жиымға қарағанда бәсеңдеу, себебі тізімнің кез келген элементіне қолжетім оның барлық алдындағы элементтерден өткеннен кейін ғана жүзеге асады;

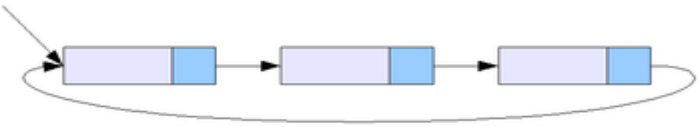
– тізім элементтері жадты шашыраңқы орналасады, ол процессоррдың кэштеуіне теріс әсер етеді;

– байланысқан тізіммен қосындыны есептеу сияқты параллель векторлық амалдар жүргізу қиын (бірақ мүмкін).

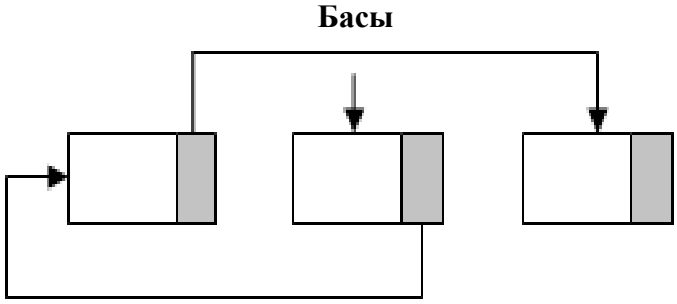
Егер тізімнің ұзындығы нөлге тең болса, онда ол бос тізім деп аталады, яғни бос тізімде ешқандай элемент болмайды..

I.3.5. Мысалдар:

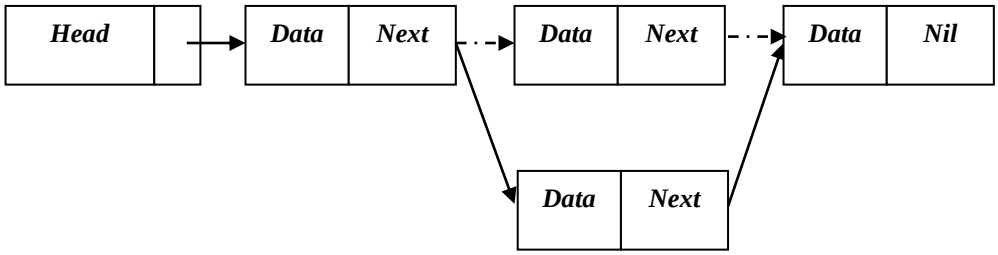
1. Бірбайланысты сақиналық тізім: соңғы элемент бірінші элементтің нұсқағышын қамтиды:



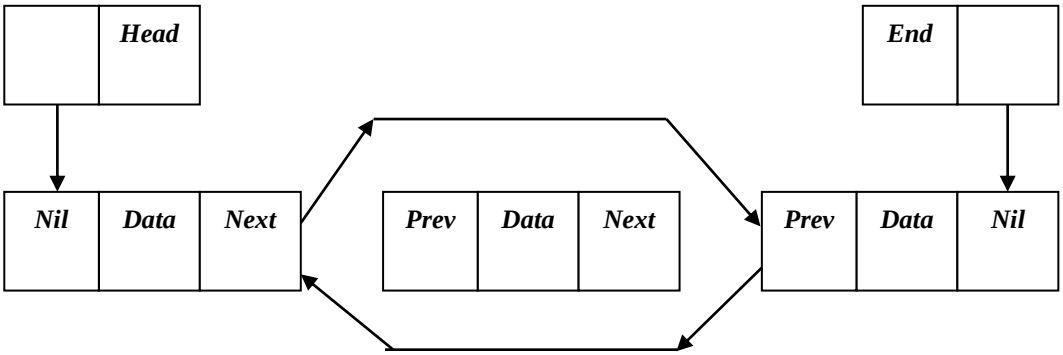
2. Элементтер реті бірінші элементке сілтеме және қалған элементтер сілтемелерінің тізбегімен анықталады:



3. Бірбайланысты тізімге жаңа элемент қосу былай жүзеге асады.



4. Екібайланысты тізімде лементті жою былай жүзеге асады.



I.3.5.Тапсырмалар:

1. Бүтін сандар 1, 9, 5, 2 үшін сызықтық тізім құрыңыз.
2. Vegatable –көкөніс және Fruit- жемістен тұратын Food –ас үшін иерархиялық тізім құрыңыз.
3. Бүтін сандар 3, 5, 1 үшін екібайланысты тізім құрыңыз.

Көмек:

1. Элементтері бір-бірімен нұсқағыш арқылы байланысқан.
2. Әрбір элемент екі нұсқағышты қамтиды.
3. Сызықтық тізім мен дарақтардан құралған.

І.3.5. Сұрақтар:

1. Сызықтық бірбайланысты тізім деген не?
2. Сызықтық екібайланысты тізім деген не?
3. Сақиналық тізім деген не?
4. Иерархиялық тізім деген не?
5. Бос тізімнің ұзындығы неге тең?

І.3.5. Тесттер:

1. Екібайланысқан тізім элементінің қандай нұсқағыштары бар?

A) келесі элементке және алдыңғы элементке;
B) бірінші элементке және соңғы элементке;
C) басына және соңына;
D) бірінші элементке және ортаңғы элементке;
E) ортаңғы элементке және бірінші элементке.

2. Тізім ұзындығы неге тең?

A) нұсқағыш санына;
B) элемент санына;
C) байланыс санына;
D) үзбелер санына;
E) сілтемелер санына.

3. Сақиналы тізімде қандай өрістер болады?

A) алдыңғы, деректер, келесі;
B) басы, деректер, соңы;
C) бірінші, орта, соңы;
D) алдыңғы, келесі, соңы;
E) бірінші, келесі, соңы.

I.3.6. Хеш-кестелер

Хеш-кестелер – элементтері *кілттер*, *мәндер* қосағы болатын динамикалық деректер құрылымы.

Хеш-кестелер әдеттегі жиымның жалпыланған түрі. Бірақ жиымның кілті тек бүтін сан болса, хеш-кестелер үшін кілт хеш-кодты есептеуге болатын кезкелген объект болуы мүмкін.

Хеш-кестелерде үш түрлі амалдар анықталған:

- *жаңа қосақты қосу*;
- *кілт бойынша қосақты іздеу*;
- *кілт бойынша қосақты жою*.

Хештеу идеясы әдеттегі ***H***[0..*m*-1] жиымындағы кілттерді таратуға негізделген. Тарату әрбір элементтің кілті үшін қайсыбір *h* хеш-функциясын есептеумен жүзеге асырылады. Бұл функция кілт негізінде ***H*** жиымы үшін индекс қызметін атқаратын *n* бүтін санын есептейді. Әрине, әртүрлі объектілер үшін әртүрлі хеш-код беретін хеш-функциясын ойлап табу керек.

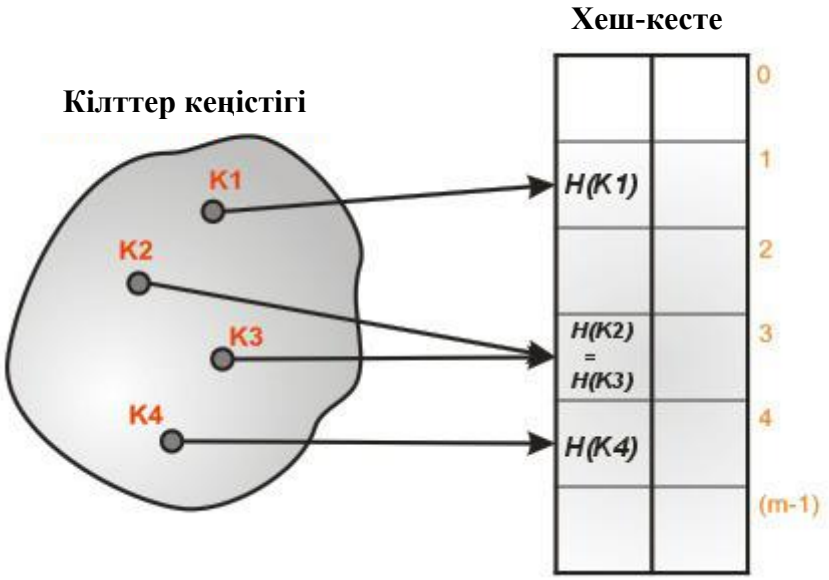
Хеш-кесте элементтің орналасу орнын (адресін) хеш-функцияның мәні ретінде есептеу есебінен элементтерді тез іздеуге оңтайландырылған. Хеш-функцияның аргументі элементпен қауымдастырылған кілт, мысалы, оның реттік нөмірі болады. Әрбір деректер типі үшін өзінің хеш-функциясын жасақтауға болады. Бірақ хеш-функцияларға қойылатын негізгі талаптар бар: ол кілттерді хеш-кестенің ұяшықтарына мейлінше бірыңғай таратуы керек және оңай есептелуі қажет.

Хеш-кестеде амалдың орындалуы кілтке тәуелді хеш-функциясын есептеуден басталады.

Хеш-кестедегі амалдың орындалуы кілтке тәуелді хеш-функцияны есептеуден басталады. Алынатын *i* = *hash* (*key*) хеш-мәні ***H*** жиымында индекс рөлін атқарады. Сонан кейін, орындалатын (қосу, іздеу және жою) амалы ***H***[*i*] жиымының сәйкес ұяшығында сақталатын объектіге бағытталып жіберіледі.

Төмендегі 1.3.6.1-суретте кілтке қолданылған *h* хеш-функциясының нәтижесі хеш-кестесінің индексі болатыны көрсетілген.

Сонымен қатар, бұл сурет негізгі проблеманың бірін көрсетеді. Кілттер саны *n*-ге салыстырғанда хеш-кесте мөлшерінің мәні *m* жеткілікті кіші немесе нашар хеш-функциясы болғанда, екі кілт ***H*** кестесінің бір ғана ұяшығына хеширленуі мүмкін. Бұл жағдай *қайшылық* деп аталады. Жақсы хеш-функциялар қайшылықтардың ықтималдығын азайтуға тырысады, бірақ барлық мүмкін болатын кілттердің кеңістігі ***H*** хеш-кестесінің мөлшерінен үлкен болу себепті, қайшықтан қашу болмайтын сияқты. Бұл жағдайларда қайшылықтарды шешу үшін бірнеше технологиялар даярланған. Олардың негізгілерін біз қарастырамыз.



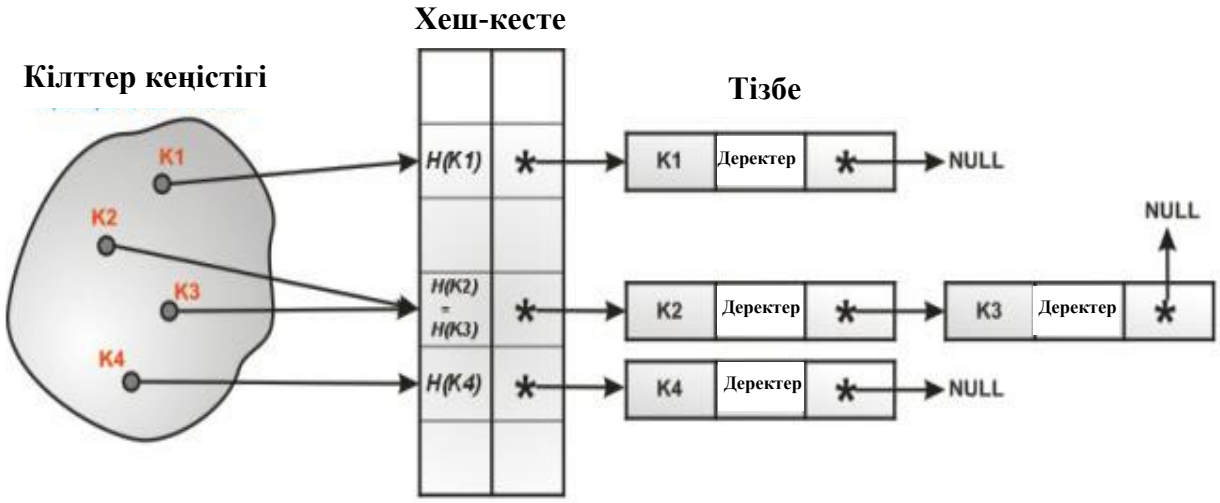
1.3.6.1-сурет. Хеш-кестесінде кілттерді индекстеу.

Хеш-кесте элементтері жұптар тізімі (хеш-кесте, тізбелер) немесе жұптар (хеш-кесте, ашық адрестеу) болатын қандай да бір ***H*** жиынын қамтиды.

Хеш-кестенің екі негізгі түрі бар: *тізбелі хеширлеу* (*ашық хеширлеу*) және *ашық адрестелген хеширлеу* (*жабық хеширлеу*).

Тізбелі хеширлеу. Тізбелі хеширлеу жағдайында, бір ғана ұяшыққа хеширленген элементтер байланысқан тізімге бірігеді.

1.3.6.2-суретінде тізбелі хеширлеу көрсетілген.



1.3.6.2-сурет. Тізбелі хеширлеу.

Мұнда $H(K2)$ және $H(K3)$ элементтері бір ұяшыққа хеширленген, сондықтан олар байланысқан тізімге біріктірілген.

Қашылықты табу идеясы мейлінше қарапайым. Егер элементті хеш-кестенің берілген ұяшығына қосарда біз байланысқан тізімнің элементіне сілтеме кездестірсек, онда қайшылық туады. Сөйтіп, біз элементімізді түйін ретінде тізімге орналастырамыз. Іздеу кезінде біз тізбемен, қашан керек жерге жеткенше кілттерді өзара эквивалентілікке салыстыра отырып, жүреміз. Жою кезінде де осындай жағдай болады.

Орналастыратын эемент кестеде жоқ екендігін ескергенде, орналастыру процедурасы $O(1)$ уақыт алады. Іздеу уақыты $O(n)$-нен үлкен бомайды. Егер барлық элементтер жалғыз ұяшыққа хеширленсе, онда іздеу уақыты $O(n)$-ге тең болады.

Ашық адресті хеширлеу. Ашық адрестеу әдісінде барлық элементтер хеш-кестінде, байланысқан тізімді пайдаланбай тікелей сақталады. Ашық адрестеу әдісін пайдаланған кезде, оның тізбелі хеширлеуден айырмашылығы, хеш-кесте толығымен толтырылған жағдай болуы мүмкін, онда кестеге жаңа элемент қосуға болмайды. Сондықтан пайда болған қайшылықтың шешімі хеш-кестенің мөлшерін, қайта құруды жасай отырып, динамикалық үлкейтумен табылады.

Қайшылықты шешу үшін бірнеше жолдар қолданылады.

Олардың ішіндегі ең қарапайымы – сызықтық зерттеу әдісі. Бұл жағдайда қайшылық пайда болса, ағымдағы ұяшықтан кейінгі ұяшықтар бірінен кейін бірі, біздің элемент орналасатын бос ұяшық табылғанша, тексеріледі. Кестенің соңғы индексіне жеткенде, біз кестені «циклды» жиым ретінде қарастырып, кестенің алдына секіреміз. 1.3.6.3-суретте сызықты зер **Хеш-кесте** өрсетілген.

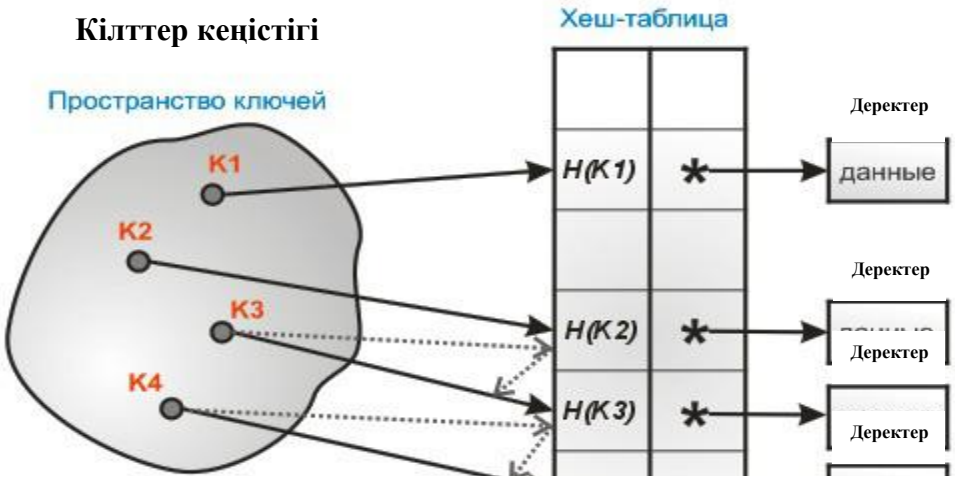


Рисунок 1.3.6.3. Сызықты зерттеу әдісімен хеширлеу.

Сызықты хеширлеу мейлінше оңай жүзеге асырылады, бірақ онымен қомақты – кластерлеу проблемасы байланысқан. Бұл іздеудің орта уақытын үлкейтетін бос емес ұяшықтардың ұзын тізбесін жасау құбылысы. Кластерлеу әсерін төмендету үшін басқа стратегия – екіленген хеширлеу пайдаланылады. Бұл стратегияның идеясы, ұяшықта қайшылық болғандағы бір ұяшыққа сызықтық ығыстырудың орнына, басқа хеш-функцияны қолдануда болады.

Ашық адрестеуді жүзеге асырудағы қиын сұрақтардың бірі – ол элементті жою амалы. Егер біз хеш-кестедегі элементті жай ғана жоятын болсақ, онда кілт іздеуді мүмкін болмайтындай етеміз, себебі толтыру кезінде ағымдағы ұяшық бос емес болуы мүмкін. Біз келешекте ескеру үшін тазаланған ұяшықтарды қандай да бір тамғамен белгілей аламыз.

Кейбір арнаулы жағдайларда қайшылықтардан мүлде құтылуға болады. Мысалы, егер элементтің барлық кілттері алдын ала белгілі болса (немесе өте сирек өзгерсе), онда оларды хеш-кестенің ұяшықтарына қайшылықсыз орналастыратын қандай да бір жетілдірілген хеш-функция табуға болады. Осындай хеш-

функцияны пайдаланатын хеш-кестелер қайшылықты шешетін механизмді қажет етпейді және олар *тура адрестелетін хеш-кестелер* деп аталады.

Егер хеш-функция *n* кілтті кестенің *m* ұяшықтарына бірыңғай таратса, онда әрбір тізімде *n/m* кілттер ораналасады.

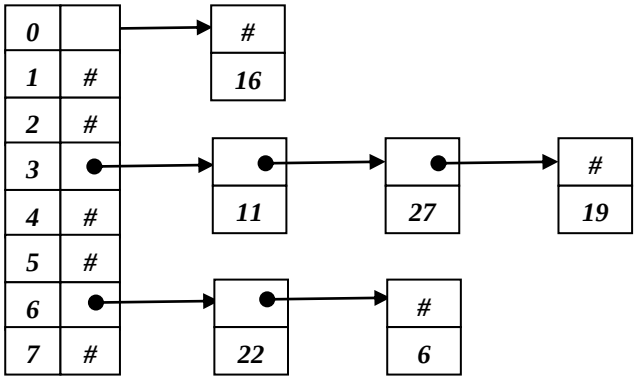
Сақталатын элементтер *n* санын ***H*** жиымының *m* мөлшеріне (хеш-функцияның мүмкін мәндерінің санына) бөлінген *n/m* мәнін хеш-кестенің толтыру коэфициенті – *load factor* деп аталады. Бұл мән амалдар орындалуының орта уақыты тәуелді болатын маңызды параметр болады.

Енді хеш-функцияға қойылатын негізгі талаптарды келтірейік:

- функция 32-битті бүтін мәнді қайтаруы керек;
- функция кірісінде ұзындығы айқын берілмеген бос емес тізбені қабылдауы керек;
- функция мейлінше көп бірыңғай таратылған хеш-мәнді беруі керек.

I.3.6. Мысалдар:

Төмендегі суретте көрсетілген хеш-кесте 8 элементтен тұратын жиым болады. Оның әрбір элементі санды сақтайтын сызықтық тізімге нұсқағыш. Мұнда хеш-функция кілтті 8-ге бөледі, ал қалдық кестенің индексі болады, оның мәні 0-ден 7-ге дейін. Ал хеш-кестеде адрестеу үшін 0-ден 7-ге дейінгі сандар қажет болғандықтан, алгоритм индекстердің қолжетім мәндерін береді.



I.3.6. Тапсырма:

1. Кілттері біздің елдегі облыс орталықтары, ал мәндері бүтін сан болатын хеш-кестесін тізбелі хеширлеу әдісімен тұғызыңыз.
2. Кілттері идентификаторлар, ал мәндері адрестер болатын

хеш-кестесін ашық адресті хеширлеу әдісімен тұғызыңыз.

3. Кілттері аты жөні, ал мәндері жеке айқындау коды (ЖАК) болатын хеш-кестесін сызықтық хеширлеу әдісімен тұғызыңыз.

Көмек:

Хеш-кестенің анықтамасы мен оларды құру әдістерін пайдаланыңыз.

I.3.6. Сұрақтар:

1. Қандай жағдайда қайшылық пайда болады?
2. Қандай жағдайда хеш-кестеде іздеу мүмкін емес?
3. Қайта хеширлеуде адресті өзгерту әдісі қалай таңдалады?

Тесты I.3.6.

1. Хеш-кестелерде қандай амалдар анықталған?

A) жаңа жұпты қосу, кілт бойынша іздеу, кілт бойынша жұпты жою;

B) жаңа жұпты қосу;

C) кілт бойынша іздеу;

D) кілт бойынша жұпты жою;

E) жаңа жұпты қосу, кілт бойынша іздеу.

2. Қайшылық деген не?

A) бір кілт бір ұяшыққа хеширленеді;

B) бір кілт екі ұяшыққа хеширленеді;

C) екі кілт бір ұяшыққа хеширленеді;

D) екі кілт екі ұяшыққа хеширленеді;

E) көп кілт көп ұяшыққа хеширленеді;;

3. Хеш-кестенің элементін табу үшін не көрсету керек?

A) кілттер, мәндер;

B) тік индекстер, жатық индекстер;

C) кілттер, индекстер;

D) индекстер, мәндер;

E) элементтердің тура адрестері, кілттер.

1.3.7. Графтар мен дарақтар

Граф – *төбелер* мен *қабырғалар* деген қосақтар жиынынан құрылған динамикалық торлық байланысты деректер құрылымы. Әрбір төбе бірнеше басқа төбелермен немесе өз-өзімен байланыса алады және төбелер ешқандай иерархия жасамайды. Формалды түрде граф деп мына $G = (V, E)$ қосақтар жиынын айтады, мұнда V – *төбелер жиыны*, ал E – *қабырғалар жиыны*, нақтысында ол V жиынында анықталған қатынас, яғни, $E \subseteq V \times V$. Граф төбелері мен қабырғалары оның элементтері деп те аталады. Графтағы төбелер саны $|V|$ графтың реті, ал қабырғалар саны $|E|$ графтың мөлшері болады.

Графтағы $e = (x_i, x_j) \in E$ қабырғасы $x_i \in V$ және $x_j \in V$ төбелерін қосады деп айтады, i=1, 2, … ; j=1, 2, … . Өз кезегінде, $x_i \in V$ және $x_j \in V$ төбелері $e = (x_i, x_j)$ қабырғасы шектейді, яғни олар қабырғаның шеткі төбелері (шеттері) болады.

Граф қабырғаларына қатысты мына анықтамалар беріледі:

1. Егер екі қабырға ортақ шеткі төбеге ие болса, онда осы қабырғалар *сыбайлас* болады;

2. Бірдей шеткі төбелері бар қабырғалар *еселенген* қабырғалар деп аталады.

3. Егер граф қабырғасының шеткі төбелері бірдей болса, онда ол қабырға *тұзақ* деп аталады.

1.3.7. Ескертпе: Тұзақ төбесінің дәрежесін еспетегенде, қабырға екі рет саналады.

Граф төбелеріне қатысты қосымша мыналар анықталады:

1. Егер екі төбе бір ғана қабырғаның шеткі төбелері болса, онда осы төбелер *сыбайлас* деп аталады;

2. Егер төбе қабырғаның шеткі төбесі болса, онда осы *төбе қабырғаға инцидентті* (*қабырға төбеге инцидентті*) болады.

3. *Төбе дәрежесі* деп осы төбеге инцидентті қабырғалар санын айтады, мұнда тұзақ екі рет есептелінеді.

4. Егер төбе бір ғана қабырғаның шеті болса, онда ол *ілінген* (*жапырақ*) деп аталады, оның дәрежесі 1-ге тең.

5. Егер қабырға ешбір төбеге инцидентті болмаса, онда ол *оқшауланған* деп аталады, оның дәрежесі 0-ге тең.

Графтардың бірнеше типтері бар. Графтардың типтеріне қатысты келесі анықтамаларды беруге болады:

1. Если V және E жиындары ақырлы болса, онда граф ақырлы деп аталады.

2. Төбелер саны n және қабырғалар саны m болатын ақырлы графты (n; m)-*граф* деп атайды.

3. Төбелері тек қана оқшауланған болатын графты *бос граф* немесе *нөл-граф* деп атайды.

4. Тұзақсыз және еселенген қабырғасыз графты *жай граф* деп атайды.

5. Кез келген екі төбесі қабырғамен қосылған жай графты *толық граф* деп атайды.

6. Егер жай графтың төбелер жиыны V екі қиылыспайтын V_1 және V_2 ішжиындарға бөлінетін болса және бір ғана ішжиын төбелерін қосатын қабырғалар болмаса, онда оны *екіөзекті граф* немесе *биграф* деп атайды.

7. Тұзақсыз, бірақ еселенген қабырғалары бар графты *мульти-граф* деп атайды.

8. Ең болмаса бір ғана тұзағы бар графты *псевдограф* деп атайды. Псевдографта еселенген қабырғалар болуы мүмкін.

9. Егер графтың әрбір төбесінен сандары бірдей қабырғалар шықса және әрбір төбесіне сандары бірдей қабырғалар кірсе, онда оны *регулярлы граф* деп атайды.

10. Егер графтың әрбір қабырғасы үшін бағыт анықталса, онда осындай графты *бағытталған граф* деп атайды.

11. Егер графтың әрбір қабырғасы салмаққа ие болса, онда осындай графты *салмақталған граф* деп атайды, яғни $w : E \to R$ функциясын анықтауға болады, мұнда R – нақты сандар жиыны, w – қабырға салмағы және $w \geq 0$.

Графтағы жол деп көрші екі қабырғаның ортақ төбесі болатындай және ешбір қабырға бір реттен артық кездеспейтіндей етіп бір төбеден басқа төбеге жүргізілген тізбекті айтады, яғни

формалды түрде графтағы жол деп $\{(x_1,x_2),(x_2,x_3),...,(x_{m-1},x_m)\}$ қосақтары қабырға болатындай $(x_1,x_2,x_3,...,x_{m-1},x_m)$ төбелердің тізбегін айтады. Екі төбе $x_i \in V$ және $x_j \in V$ *байланысқан* (*байланыспаған*) дейді, егер x_i-ден x_j-ге жол болса (болмаса). Бұл жол екі бағытта болуы мүмкін. Егер графтағы әрбір екі төбе байланысқан болса, онда мұндай графты *байланысқан граф* дейді. Егер графта бір ғана байланыспаған төбелер қосағы табылса, онда графты *байланыспаған граф* дейді. Егер граф төбелерінің барлық қосақтары екі бағытта байланысқан болса, онда оны *қатты байланысқан граф* дейді.

Егер бір төбеден шыққан жол осы төбеге қайтып кіретін болса, онда осы жолды *тұйық* (*цикл*) деп атайды, яғни, тұйықта бастапқы және соңғы төбе бірдей болады. Егер тұйық ешбір төбеден бір реттен көп өтпейтін болса, онда оны *қарапайым тұйық* дейді. Егер тұйық бір төбеден шығып тікелей осы төбеге қайтып кірсе, онда оны *тұзақ* дейді, яғни тұзақта бір ғана төбе болады.

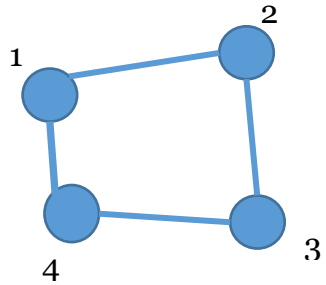
Жолдың ұзындығы деп осы жолдағы қабырғалар санын айтады. Егер граф қабырғаларының салмақтары олардың ұзындықтары болса, онда жолдың ұзындығы былай есептеледі:

$$w(x_1,x_2,x_3,...,x_{m-1},x_m)=\sum_{i=1}^{m-1} w(x_i,x_{i+1})$$

Графтағы белгілі есептер мыналар болады: *екі графты салыстыру*, *бір төбеден екінші төбеге баратын ең кіші жолды табу*, *графтағы тұйық жолдардың санын табу* және басқалар.

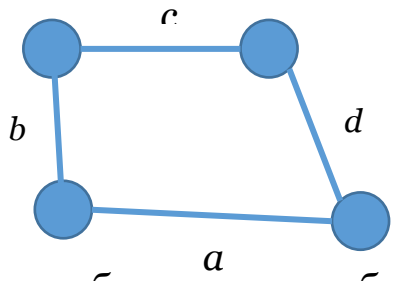
I.3.7.1. Мысалдар:

1. Мейлі берілген граф мынадай болсын:



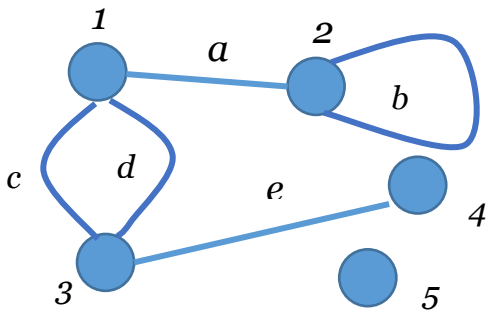
Бұл графта 1 және 2 төбелері сыбайлас, ал 1 және 3 төбелері сыбайлас емес.

2. Мейлі мынадай граф берілсін:



Бұл графта *a* және *b* қабырғалары сыбайлас, ал *b* және *d* қабырғалары сыбайлас емес.

3. Мейлі мынадай граф берілсін:



Бұл графта 1 және 2, 1 және 3, 3 және 4 төбелері сыбайлас, ал 5 төбесі оқшауланған. Сонымен қатар, 1, 2 және 3 төбелері үш дәрежелі, 4 төбесі ілінген, *c* және *d* төбелері еселенген, *b* қабырғасы тұзақ.

Дарақ – барлық төбелері байланысқан, ал жолдары тұйық емес граф, яғни тұйықсыз және тұзақсыз байланысқан граф.

Дарақта төбелер үш топқа бөлінеді:

1) *тамыр* – өзінен бір немесе бірнеше қабырға шығатын, бірақ өзіне ешқандай қабырға кірмейтін төбе, яғни бір де бір атасы болмайтын, бірақ көп ұрпақты бола алатын төбе;

3) *бүршік* – өзіне бір ғана қабырға кіретін, бірақ өзінен бір немесе бірнеше қабырға шығатын төбе, яғни бір ғана атасы бар, бірақ көп ұрпақты бола алатын төбе;

2) *жапырақ* – өзіне бір ғана қабырға кіретін, бірақ өзінен ешқандай қабырға шықпайтын төбе, яғни бір ғана атасы бар, бірақ бір де бір ұрпағы жоқ төбе.

Дарақта тамырдан бүршіктер арқылы жапырақтарға дейін жол бағыты болады. Бір дарақтың ішінде бірнеше дарақтар болуы мүмкін, оларды *ішдарақтар* дейді.

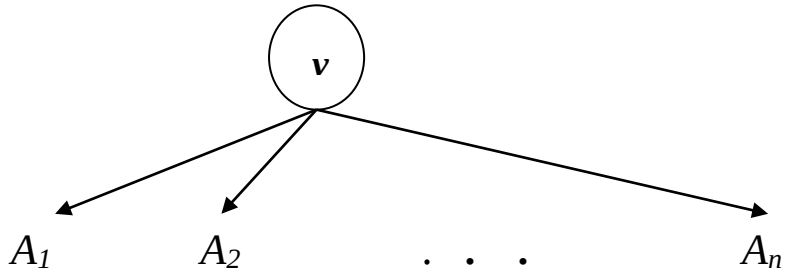
Енді жоғарыда айтылғандарды ескеріп, дарақтарға мынадай рекурсивті (өзіне-өзі сілтеме жасайтын) анықтама беруге болады:

1. *Рекурсия базисі*: жалғыз ғана *v* төбесінен тұратын $\{v\}$ жиыны дарақ, мұнда оның жалғыз төбесі оның бірмезгілде тамыры және жапырағы болады.

2. *Рекурсия қадамы*: егер *v* – төбе және $A_1, A_2, ..., A_n$ – дарақтар болса, онда тамыры *v* төбесі болатын және қабырғалары осы төбеден шығып $A_1, A_2, ..., A_n$ дарақшалардың тамырларына кіретін жаңа дарақ тұрғызуға болады.

3. *Рекурсивті қорытынды.* Дарақтар тек 1-ші және 2-ші ережелермен ғана алынады.

Осы анықтама I.3.5-суретте бейнеленген болады:



I.3.5-сурет. Дарақтардың анықтамасы.

Дарақтардың ішінде *екілік* (*бинарлық*) *дарақтар* ерекше орын алады. Оны былай анықтауға болады:

Екілік дарақ – әрбір төбесінің екіден аспайтын ұрпағы болатын дарақ. Осы төбе *аталық төбе* деп аталады, ал ұрпақтар *сол мұрагер* және *оң мұрагер* деп аталады.

Екілік дарақтың рекурсивті анықтамасын берейік. Бинарлық дарақ деп төбелердің мынадай жиынын айтады:

- әлде ешнәрсені қамтымайды (бос жиын);
- әлде солжақ ішдарақ және оңжақ ішдарақ деп аталатын екі бинарлық дарақпен байланысқан тамырдан тұрады.

Сонымен, бинарлық дарақ әлде бос болады, әлде деректер және әрқайсысы бос бола алатын екі ішдарақтан тұрады. Егер кейбір

төбеде екі ішдарақ та бос болса, онда ол жапырақтық болып табылады.

Формалды түрде екілік дарақты былай анықтауға болады:

<биндарақ > ::= nil | (<деректер> < биндарақ > < биндарақ >)

Дарақтарда мынадай есептер шығарылады: *дарақтарды аралау*, *дарақ бойынша іздеу*, *дараққа жаңа төбе қосу*, *дарақтағы төбені жою* және т.б.

Екілік дарақтар іздеу алгоритмдерінде пайдаланылады: екілік іздеу дарағының әрбір төбесі қандай да бір сұрыпталған жинақтың элементіне сәйкес келеді, оның барлық сол мұрагерлері – кіші элементтерге, ал барлық оң мұрагерлері – үлкен элементтерге. Дарақтағы әрбір төбе тамырдан осы төбеге дейін қайталанбайтын төбелер тізбегі – жолмен айқындалады. Жол ұзындығы иерархиядағы төбенің деңгейі болады.

Практикалық мақсатта әдетте екілік дарақтардың екі түрін пайдаланады: *екілік іздеу дарағы - binary search tree* (BST) және *екілік үйінді*.

Екілік іздеу дарағының келесі қасиеттері бар:

– сол ішдарақ және оң ішдарақ екілік іздеу дарағы болады;

– қандай да бір *v* төбесінің сол ішдарағының барлық төбелерінің деректер кілттерінің мәндері *v* төбесінің өзінің деректер кілтінің мәнінен кіші;

– қандай да бір *v* төбесінің оң ішдарағының барлық төбелерінің деректер кілттерінің мәндері *v* төбесінің өзінің деректер кілтінің мәнінен үлкен.

Әрбір төбедегі деректердің кілттері өздерінде салыстыру қатынасы анықталған болуы керектігі айқын.

Екілік үйінді немесе сұрыпталған дарақ келесі қасиеттерге ие:

– кез келген төбедегі мән оның ұрпақтарының мәндерінен кіші емес;

– жапырақтар тереңдігі (түбірге дейін арақашықтық) бір қабаттан аспайды;

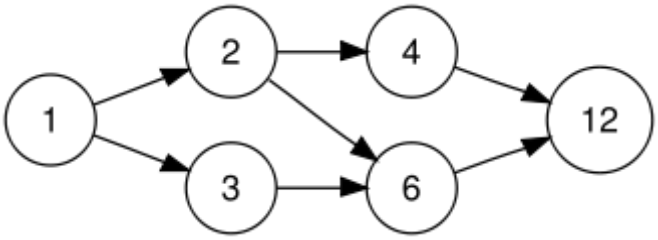
– соңғы қабат солдан оңға қарай толтырылады.

Жоғарыдағы үйінді *max-heap* деп аталады. Сондай-ақ, кез

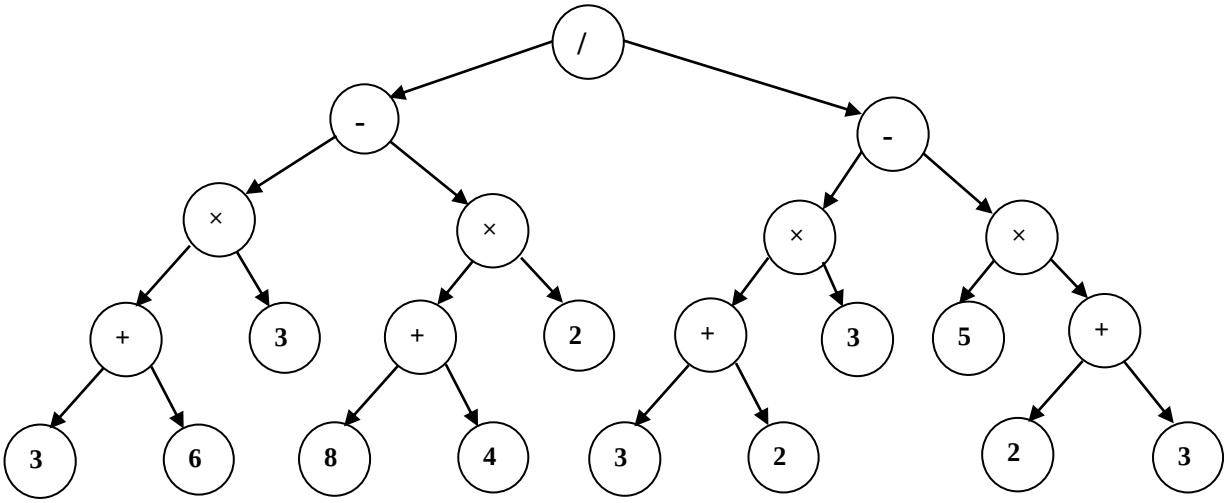
келген төбедегі мән, керісінше, оның ұрпақтарының мәндерінен үлкен емес үйінді бар. Осындай үйінді *min-heap* деп аталады.

I.3.7.2. Мысалдар:

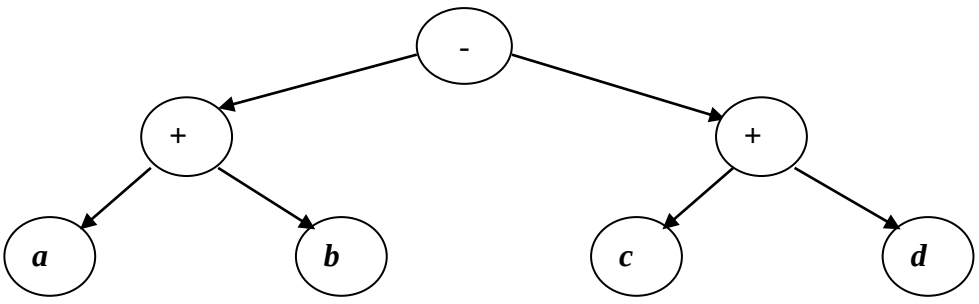
1. Ақырлы объектілердегі бинарлық қатынасты бағытталған граф түрінде бейнелеуге болады. Төменде 1-ден 12-ге дейінгі сандардағы бөлінгіштік қатынасы бейнеленген: 2 мен 3 саны 1-ге бөлінеді; 4 пен 6 саны 2-ге бөлінеді; 6 саны 2 мен 3-ге бөлінеді; 12 саны 4 пен 6-ға бөлінеді.



2. Екілік дарақпен кескінделген арифметикалық өрнек ((3+6)*3 - (8+4)*2) / ((3+2)*3 - 5*(2+3)) мәнін есептеу үшін осы дарақты жоғарыдан төмен және солдан оңға қарай орағыту керек:



3. Берілген $(a+b)$ - $(c+d)$ өрнегін екілік дарақ түрінде бейнелеу, мұнда -,+, a, b, c, d дарақтағы төбелер болуы керек.



Сол жақ ішдарақ Оң жақ ішдарақ

І.3.7. Тапсырмалар:

1. Идентификатор құрылымын сипаттау үшін бағытталған салмақталған граф тұрғызыңыз.

2. Келесі өрнекке ағаш тұрғызыңыз ((a / (b + c)) + (x * (y - z))).

Көмек:

1. Жалпылықты жоймай, қажетті графты салу үшін осы графтың салмақтары ретінде әріптер емес тек жалғыз әріп, цифрлар емес тек жалғыз цифр қарастырамыз

2. Сәйкес бинарлық ағашта жапырақтар операнд ретінде, ал қалған төбелер операциялар қызметін атқарады.

І.3.7. Сұрақтар:

1. Графта жол қалай құрылады?
2. Дарақ деген не?
3. Екілік дарақ деген не?

І.3.7. Тесттер:

1. Графтың қандай түрлері болады?
   A) бағытталған граф, бағытталмаған граф;
   B) бағытталған граф, анықталған граф;
   C) анықталған граф, бағытталмаған граф;
   D) анықталған граф, анықталмаған граф;
   E) анықталмаған граф, бағытталмаған граф.
2. Дарақ деген не?
   A) тұзақсыз және тұйықсыз граф;
   B) салмақтанбаған граф;
   C) торсыз және тұйықсыз граф;
   D) салмақтанған граф;
   E) бағытталған граф.
3. Екілік дарақ деген не?
   A) әрбір төбесінің екіден артық ұрпағы болмайтын дарақ;
   B) екі төбесі ғана болатын дарақ;
   C) тұйығы жоқ дарақ;
   D) тұзағы жоқ дарақ;
   E) бір төбесінің тікелей ұрпағы жоқ дарақ.

1.3.8. Стектер, кезектер және дектер

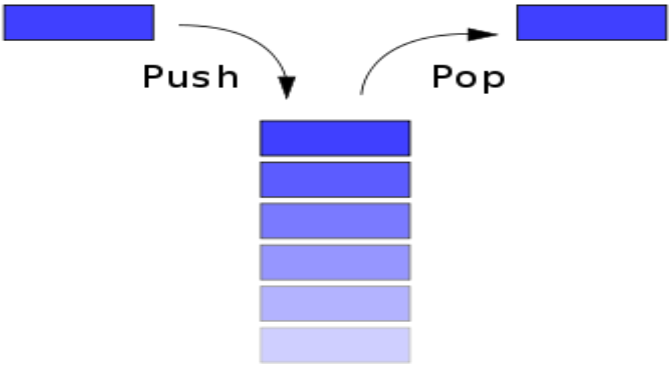
Стек (***Stack***) – динамикалық сызықтық деректер құрылымы, онда элементтер олардың түсу уақыты бойынша реттелген тізбек құрып, «*соңғы еніп бірінші шығады – Last In Firsrt Out* (*LIFO*)» деген прінсіпті жүзеге асырады, яғни ол тізбекке соңғы қосылған элементті тізбектен бірінші алып тастайды. Бұл тізбек стек төбесі деп аталатын шетінен ғана қолжетімді болады, яғни стектегі барлық амал тек стектің төбесінде ғана орындалады. Стекпен жұмыс істеу үшін стектің төбесіне нұсқағыш болуы және келесі амалдардың анықталуы керек:

- стекті құру (төбесіне нұсқағыш жасау);
- элементті стекке қосу (кіргізу - *Push*);
- элементті стектен жою (шығару - *Pop*);
- стек төбесіндегі элементті жоймай қарау;
- стектің күйін (бостығын) тексеру;
- стекті тазалау.

Стектер жүйелік программалау қамтымда, компиляторларда, әртүрлі рекурсивтік алгоритмдерде кеңінен қолданылады. Мысалы, программалық код орындалу барысында процедураны шақыру қажет болғанда, оның коды орындалып біткен соң шақыру нүктесінен кейінгі нұсқауға дұрыс қайту үшін, стектегі оны шақыру орнына сілтеме орналастырылады. Сондай-ақ, стектер рекурсивтік шақыруды ұйымдастыру үшін қолданылады: соңғы шақыру орындалып біткенше, барлық алдыңғы шақырулар стекте болады.

Стекті бірнеше біртипті элементтерді сақтауға және стектің жұмыс істеу прінсіпін жүзеге асыруға мүмкіндік беретін кез келген деректер құрылымы негізінде ұйымдастыруға болады. Стекті ұйымдастыру үшін ең қолайлысы бірбағытты тізім болады, оның басы стек төбесі ретінде таңдалады.

Стекті І.3.8.1-суреттегідей тік орнастырылған, жеке орындарға бөлінген түбінде серіппесі бар құбыр түрінде кескіндеуге болады.



I.3.8.1. Сурет. Стек кескіні.

Құбырдың жоғарғы шеті ашық, одан элементті стекке қосады (кіргізеді - *Push*) және одан жояды (шығарады - *Pop*).

Кезек – динамикалық сызықтық деректер құрылымы, онда элементтер олардың түсу уақыты бойынша реттелген тізбек құрып, «*бірінші еніп бірінші шығады – Firsrt In Firsrt Out* (*FIFO*)» деген прінсіпті жүзеге асырады, яғни ол тізбекке бірінші қосылған элементті тізбектен бірінші алып тастайды. Бұл тізбек екі жағынан: басынан және соңынан қолжетімді.

Кезекте келесі амалдар анықталған:

– кезекті құру (басына және соңына нұсқағыш жасау);

– элементті кезектің соңына қосу (қою - *enqueue*);

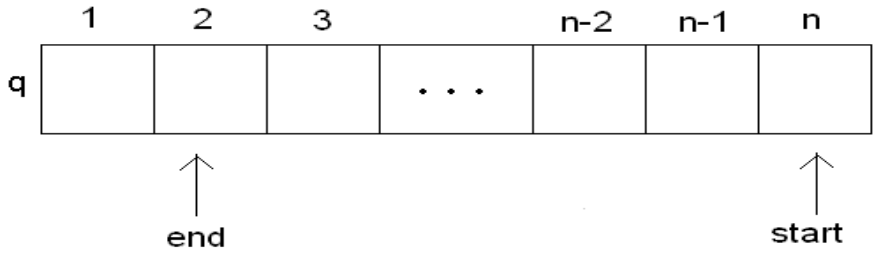
– элементті кезектің басынан жою (шығару - *dequeue*);

– стекті тазалау.

Практикада кезектер бір өлшемді жиым немесе байланысқан тізім арқылы жүзеге асырылады. Олар деректер құрылымының логикалық және физикалық деңгейлер арасындағы айырмашылықты жақсы көрсетеді (физикалық деңгейде – жиым немесе тізім, логикалық деңгейде - кезек). Бірақ кезекті екібағытты сақиналық тізімге бейнелеген жөн. Сонда ол *сақиналық кезек* деп аталады және оның бас буынында кезектің басына да соңына да нұсқағыш туралы ақпарат болады. Бұл жағдайда, біріншіден, кезек мөлшерінен шектеулік алынады, екіншіден, кезектің басына және соңына нұсқайтын айнымалылар басқаша пайдаланылады. Байланысқан тізім (сызықтық, сақиналық) түріне тәуелсіз сызықтық және сақиналық кезектер арсындағы айырмашылықтар

нақты жойылады.

Кезекті I.3.8.2-суреттегідей басына және соңына нұсқағышы бар ұяшықтарға бөлінген таспа түрінде кескіндеуге болады.



I.3.8.2. Сурет. Кезек кескіні.

Кезектердің қолдану аясын екі топқа бөлуге болады: *жүйелік қолдану* және *қолданбалы қолдану.*

Кезектердің жүйелік қолдануы:

- амалдар жүйесінің есептерін диспетчерлендіру;
- енгізу/шығаруды буферлеу.

Кезектердің қолданбалы қолдануы:

- үдерістерді моделдеу (мысалы, жаппай қамту жүйесін);
- қандай-да бір алгоритмдерде кезектерді көмекші деректер құрылымы ретінде пайдалану (мысалы, графта іздеу кезінде).

Дек – динамикалық сызықтық деректер құрылымы, онда элементтер тізбек құрады және бұл тізбектің кез келген екі жағынан қосылады және жойылады. Егер кезекте жылжу бір жаққа бірінші элементтен соңғыға жүрсе және стекте де жылжу бір жаққа, бірақ кері бағытта соңғы элементтен біріншіге қарай болса, онда декте жылжу екі бағытта да жүзеге асырылады.

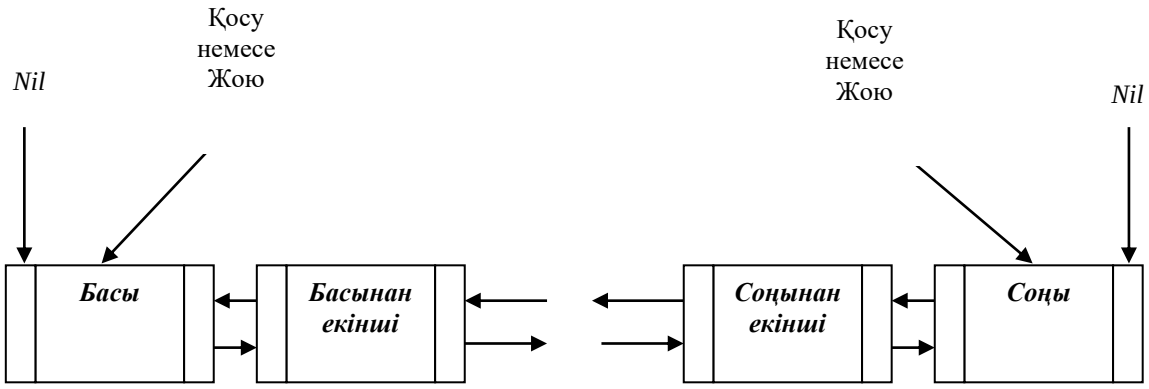
Дек сөзі Double Ended Queue (Deq) – *екі кірісті кезек* дегеннен шыққан. Декте негізгі амалдарды орындау үшін дектің басын және соңын білу қажет, олар сілтемелер арқылы анықталады.

Декте келесі амалдар анықталған:

- элементті дектің соңына қосу;
- элементті дектің басына қосу;
- элементті дектің соңынан жою;
- элементті дектің басынан жою;
- дектің мөлшерін анықтау;
- декті тазалау.

Декті екібайланысты тізіммен жүзеге асыруға болады. Сонда декті әрқайсысы үш өрістен тұратын (бірінші өріс – алдыңғы элементке сілтеме өрісі, екінші өріс – элемент өріс, үшінші өріс – келесі элементке сілтеме өрісі) үзбелер тізбегі түрінде кескіндеуге болады.

Декті I.3.8.3-суреттегідей басы мен соңына нұсқағышты көрсетіп екі байланысты тізім түрінде кескіндеуге болады.

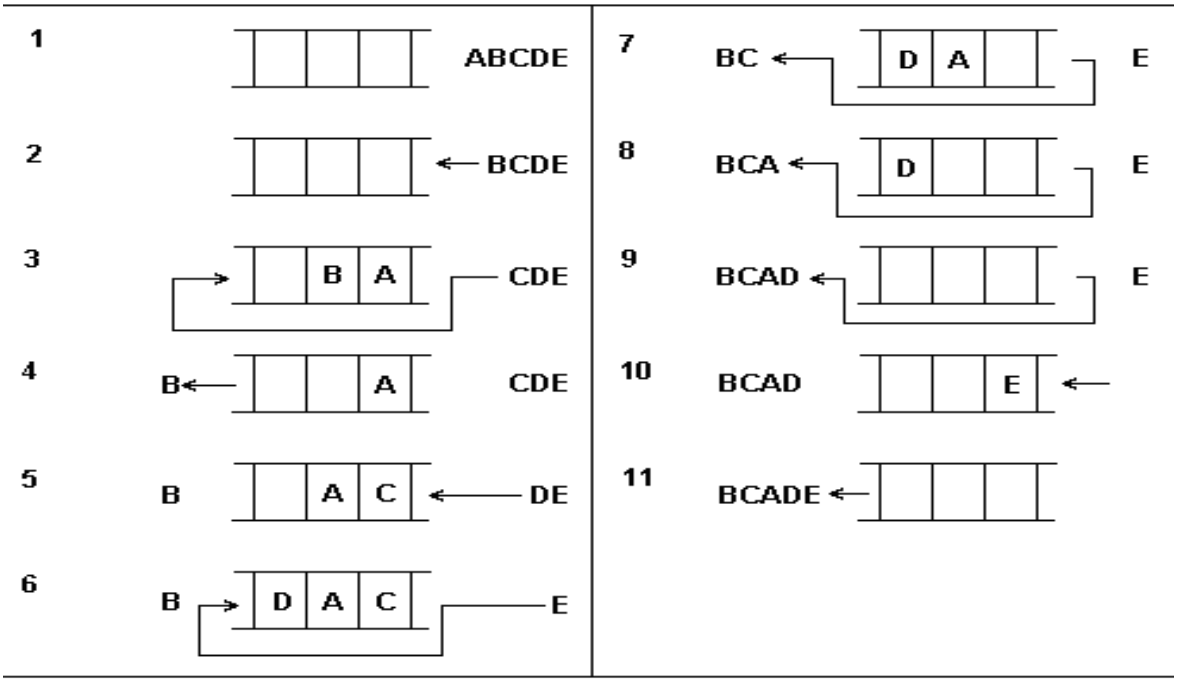


I.3.8.3. Сурет. Дек кескіні.

Бұл кескінде екі сілтеме *Nil-Пусто* бар, олардың әрқайсысы дектің екі жағында (басы мен соңында) жылжуға шектеу қояды.

I.3.8. Мысалдар:

1. Стек ретінде үстелдегі кітаптардың бумасын алуға болады. Кітаптар буманың үстіне қойлады және алынады.
2. Кезек ретінде сұранысқа байланысты қызмет көрсетуді алуға болады. Олардың ішіндегі ең бірінші қызмет ең бірінші келген сұраныс бойынша, ал ең соңғы қызмет ең соңғы сұраныс бойынша атқарылады.
3. Төменде I.3.8.4-суретте бес A, B, C, D, E элементтерін қосқандағы және жойғандағы дек күйлерінің тізбегі көрсетілген. Әрбір қадамда бағыттама дектің қай (сол немесе оң) жағынан қосу немесе жою амалдарының жүзеге асатынын көрсетеді



I.3.8.4. Сурет. Дектің өзгеру кезіндегі күйлері.

I.3.8. Тапсырмалар:

1. Өмірдегі қайсыбірдің стек принципі арқылы жұмыс істейтініне мысал келтіріңіз.
2. Өмірдегі қайсыбірдің кезек принципі арқылы жұмыс істейтініне мысал келтіріңіз.

Көмек:

1. Стектің прінсіпі «*бірінші еніп бірінші шығады*».
2. Кезектің прінсіпі «*бірінші еніп бірінші шығады*».
3. Декте жылжу екі жақты болады.

I.3.8. Сұрақтар:

1. Стек дегеніміз не?
2. Стекте қандай амалдар анықталған?
3. Кезек дегеніміз не?
4. Кезекте қандай амалдар анықталған?
5. Стек дегеніміз не?
6. Декте қандай амалдар анықталған?

I.3.8. Тесттер:

1. Қандай динамикалық сызықтық деректер құрылымын стек дейді?

A) барлық амалдар тек бір жағынан орындалатын динамикалық сызықтық деректер құрылымы;

B) бір шекті статикалық сызықтық деректер құрылымы;

C) екі шекті динамикалық сызықтық деректер құрылымы;

D) динамикалық иерархиялық деректер құрылымы;

E) статикалық иерархиялық деректер құрылымы;

2. Қандай динамикалық сызықтық деректер құрылымын кезек дейді?

A) элементтері бір жағынан қосылатын және екінші жағынан жойылатын динамикалық сызықтық деректер құрылымы;

B) элеменнттері бір жағынан жойылатын статикалық сызықтық деректер құрылымы;

C) элеменнттері бір жағынан қосылатын динамикалық сызықтық деректер құрылымы;

D) екі жақты динамикалық иерархиялық деректер құрылымы;

E) статикалық иерархиялық деректер құрылымы.

3. Қандай динамикалық сызықтық деректер құрылымын дек дейді?

A) элементтері екі жағынан қосылатын және жойылатын динамикалық сызықтық деректер құрылымы;

B) элеменнттері бір жағынан қосылатын динамикалық сызықтық деректер құрылымы;

C) элементтері бір жағынан қосылатын жойылатын динамикалық сызықтық деректер құрылымы;

D) элементтері бір жағынан қосылатын және жойылатын динамикалық сызықтық деректер құрылымы;

E) элементтері бір жағынан қосылатын және жойылатын статикалық сызықтық деректер құрылымы.

II. ИНФОРМАТИКАНЫҢ ҚАТТЫ ҚАМТЫМЫ

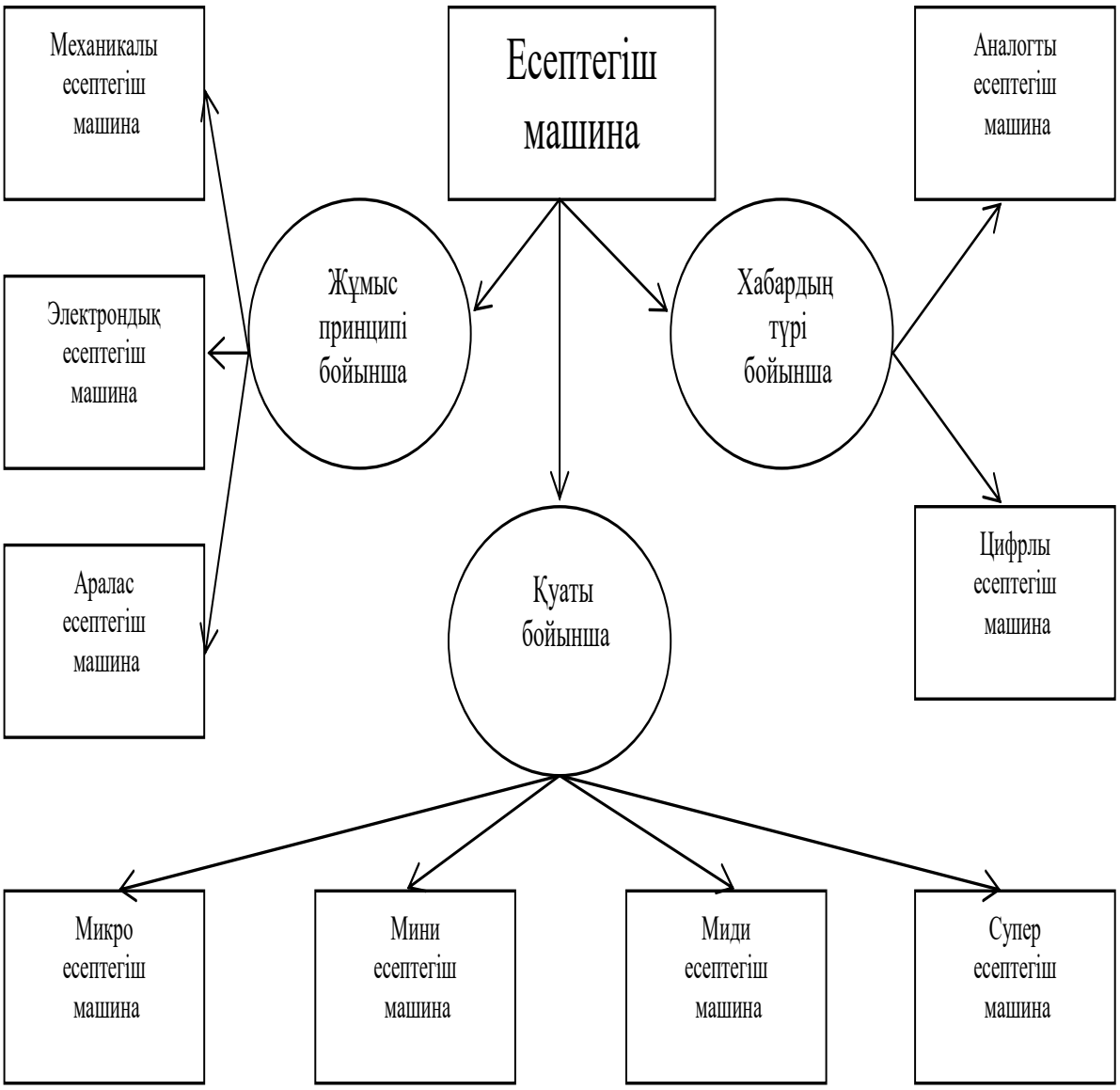
II.1. Компьютерлер

Бұл тарауда компьютерлердің логикалық құрылымы, түрлері мен буындары қарастырылады. Олардың аппараттық және программалық қамтымдары көрсетіледі.

Түйін сөздер: *автомат, автоматтандыру, компьютер, аналогты сигнал, цифрлы сигнал, аналогты компьютер, цифрлы компьютер, жад, бит, байт, ұяшық, адрес, адрестелетін сыйымдылық бірлігі, адрестік сөз, машиналық сөз, оперативтік жад, сыртқы жад, процессорр, регістр, дербес құрылғы, енгізу құрылғысы, шығару құрылғысы, магниттік табақ(ша), басқару тетігі, компьютердің жұмыс істеу прінсіпі, компьютердің жұмыс ырғағы, машиналық тіл, нұсқау, программа, компьютерлер буыны, электрондық лампа, транзистор, интегралы сұлба, программалау тілі, транслятор, мониторлық жүйе, операциялық жүйе, көп программалық режім, нақты уақыт режімі, компьютерлер кешені, байланыс арнасы, компьютерлік тор, жасанды зерде, код, кодтау, кодтау стандарты.*

Мақсат: *компьютердің шығу тарихы, құрамы, жұмыс істеу прінсіпі мен жұмыс ырғағы қарастырылған, деректерді компьтер жадында кескіндеу әдістері берілген.*

Құрылымы: Компьютердің түрлері II-суретте көрсетілген.



II–сурет. **Компьютерлердің түрлері**

II.1.1.Компьютердің құрамы мен құрылымы

«*Компьютер*» деген сөз XX–ғасырдың 40–шы жылдарында пайда болды. Ол деректерді сақтауды және өңдеуді *автоматтандыру* үшін жасалынған физикалық құрылғыны білдіреді. Компьютермен әртүрлі есептерді шығаруға болады. Ол үшін мынадай кезеңдерден өту керек:

– ақпаратты енгізу немесе алғашқы деректерді орнату;
– енгізілген ақпаратты өңдеу немесе түрлендіру;
– нәтижені анықтау және өңделген ақпаратты шығару.

Сонымен, есептегіш машина ақпаратты алады, сақтайды, берілген алгоритм бойынша өңдейді және одан шыққан нәтижені пайдаланушыға немесе басқа компьютерге береді.

Ақпарат өзінің *хабары* мен *мазмұны* арқылы сипаталатындығы және хабар *жіберушіден қабылдаушыға* материалды–энергетикалық (электр, жарық, дыбыс және т.с.с. сигналдар) түрінде берілетіндігі белгілі. Хабарды қабылдау қабылдаушының жағдайын сипаттайтын қандай–да бір шаманың уақытқа байланысты өзгерісіне тікелей тәуелді. Демек, ақпараттық хабарды ақпараттық үдеріс жүретін физикалық ортаның материалды–энергетикалық параметрлерін уақытқа байланысты өзгертетін $X(t)$ функциясымен бейнелеуге болады. Осы *функция үздіксіз* де (мысалы, уақытқа байланысты өзгеретін дененің жылдамдығы, ауаның температурасы және электр тогының күші сияқтылар), *үздікті* де (мысалы, уақытқа байланысты белгілі бір қатынас тіліндегі таңбалар тізбектері немесе дыбыстар тіркестері арқылы құрылатын сөздер мен сөйлемдер) болуы мүмкін. Үздіксіз функциялар арқылы бейнелетін хабар – *аналогты сигнал*, ал үздікті функциялар арқылы бейнелетін хабар – *дискретті сигнал* деп аталады. Адамның сезім мүшелерінің мүмкіндігі шектеулі болғандықтан үздіксіз ақпаратты дискретті түрде қабылдайды. Оған мысал ретінде термометр арқылы ауа температурасының цифрлық жуық мәнін анықтауды алуға болады. Бір қатар жағдайда ақпараттың үздіксіз түрінен дискретті түрге түрленуі өте тиімді

болады. Ол үшін *аналогты–цифрлық түрлендіргіш* деп аталатын арнаулы құрылғыларды пайдаланады.

Мысалы, ондай құрылғы ретінде дүкенде қолданылатын кәдімгі электрондық таразыны алуға болады: аналогтық сигнал – өнімнің салмағын грамм, килограмм сияқты өлшемдердің цифрлармен бейнеленген түріне түрлендіреді.

Сондықтан, компьютерлер ақпарат хабарының түріне байланысты *аналогты компьютерлер* (АК) немесе *дискретті компьютерлер* (ДК) болып бөлінеді: бұл автомобильдердің қолданатын жанармайдың түріне байланысты дизелді немесе бензинді деп бөлінгеніне ұқсас.

АК кейбір физикалық шамаларды бейнелейтін үздіксіз деректерді өңдейді. Мұнда физикалық шамалар ретінде электр тоғының күші, дене қозғалысының үдеуі сияқтылар алынады. Әдетте, АК арқылы тек арнаулы есептерді ғана шешуге болады, сондықтан бұлар көп тарамаған. Бұдан ары қарай АК қаралмайды.

Ең алғаш ДК тек сандық деректерді өңдеу, яғни ғылыми–техникалық, соның ішінде көбінесе математикалық есептерді шешу үшін ғана қолданылған. Сондықтан оның атындағы «дискреттік» деген сөздің орнына «цифрлық» деген анықтауышты пайдаланып *цифрлық компьютер* (ЦК) деп аталып кетті.

Компьютер құрамына мынадай құрылғылар кіреді:

– деректерді және оларды өңдеуге қажетті пәрмендерді жадтайтын құрылғы, ол *жад* деп аталады;

– пәрмендерде көрсетілген амалдарды орындайтын құрылғы, ол *процессорр* деп аталады;

– деректерді сақтау мен өңдеуге байланысты туатын барлық жұмыстарды басқаратын құрылғы, *басқару құрылғысы* деп аталады;

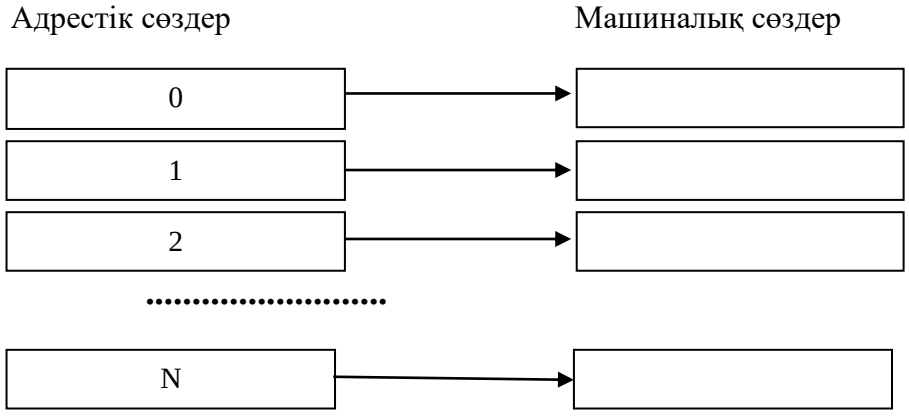
– компьютер жұмысын алғаш қосу және тоқтату үшін арналған құрылғы, ол *басқару тетігі* деп аталады.

Жадтың үш түрі бар: *аса оперативтік жад, оперативтік жад, сыртқы жад.* Аса оперативтік жад регістрлерден құрылады. Регістрлер өте жылдам жұмыс жасайды және олардың сыйымдылығы өте аз болады. Олар ақпаратты уақытша сақтауға

және түрлендіруге арналған. Оперативтік жад өзінің табиғаты бойынша тұрақты ақпаратты жадтауға арналған. Регістр мен оперативтік жад екі тұрақты жағдайы бар физикалық құрылғылардың тізбегі арқылы жүзеге асырылған. Бұл жағдайлар 0 және 1 цифрларымен бейнеленеді және **бит** – **bit** (ағылшынның **bi**nary digi**t** деген екі сөзінен құралған) деген сөзбен аталады. 0 мен 1-ден құралған ұзындығы сегіз болатын тізбек **байт** – **byte** деген сөзбен аталады. Сондықтан бір байтқа жазылатын әр түрлі ақпараттың саны $2^8 = 256$. Оперативтік жадта әрбір байт нөмірленеді. Байттың нөмірі оның *адресі* деп аталады. Адрестердің мәндері ретінде нөлден басталатын бүтін оң сандарды алады.

Программаны орындау үшін оның пәрмендері мен дектерін негізінен оперативтік жадта орналастыру керек. Осы орналастыруларды ұйымдастыру үшін бірнеше байттарды бір *сөзге* біріктіреді. Сөздің екі түрі бар: *адрестік сөз*, *машиналық сөз*. Адрестік сөзде 0 мен 1-ден тұратын адрестің кескіні орналасады. Адрестік сөздің сыйымдылығы өскен сайын компьютер жадының мүмкін сыйымдылығы да өсе береді. Машиналық сөзде бір пәрменді орындауға қажетті дербес деректі немесе амалды орындап біткеннен кейін шығатын әрбір нәтижені орналастыруға болады. Яғни, машиналық сөз компьютердің жұмыс бірлігінде қолданатын сыйымдылық бірлігі болады. Машиналық сөздің сыйымдылығы өскен сайын компьютердің жұмыс істеу қуаты да өсе береді

Оперативтік жадтағы адрестік сөзбен машиналық сөздің байланысы II.1.1.1- суретте көрсетілген.



II.1.1.1. Сурет. Оперативтік жад құрылымы

Сыртқы жад деректерді ұзақ уақыт сақтауға арналған. Сыртқы жадта деректер тіпті оларды тудырған программалардың орындалуы біткеннен кейінде де сақталуы мүмкін және кейіннен оларды осы программаны қайта жібергенде немесе басқа программалармен пайдалануға болады. Сондай-ақ, сыртқы жад программалар орындалмай тұрғанда олардың өздерін немесе орындалып жатқан программаның бір бөлігі мен деректерін сақтауға да пайдаланылады. Сыртқы жад пен оперативтік жадтың арасында тікелей алмасу бар, ал сыртқы жад пен регістрдің арасында тікелей алмасу жоқ болғандықтан, орындалып жатқан программаның белсенді бөлігі және осы кезеңде өңделетін деректер тек оперативтік жадта орналасуы қажет. Негізінен сыртқы жадтың қасиеттері оперативтік жадтың қасиеттеріндей: адрестелеу қасиеті, деректер құрылымы, сыйымдылықты өлшеу бірліктері және т.б.

Жад сыйымдылығының өлшем бірліктері арасындағы қатынастар II.1.1 -кестеде көрсетілген:

II.1.1-кесте. Жад сыйымдылығының өлшем бірліктері арасындағы қатынастар

1 килобайт	$= 1024^1$	$= 2^{10}$	= 1024 байт
1 мегабайт	$= 1024^2$	$= 2^{20}$	= 1 048 576 байт
1 гигабайт	$= 1024^3$	$= 2^{30}$	= 1 073 741 824 байт
1 терабайт	$= 1024^4$	$= 2^{40}$	= 1 099 511 627 776 байт
1 петабайт	$= 1024^5$	$= 2^{50}$	= 1 125 899 906 842 624 байт
1 эксабайт	$= 1024^6$	$= 2^{60}$	= 1 152 921 504 606 846 976 байт
1 зеттабайт	$= 1024^7$	$= 2^{70}$	= 1 180 591 620 717 411 303 424 байт
1 йоттабайт	$= 1024^8$	$= 2^{80}$	= 1 208 925 819 614 629 174 706 176 байт

Бірақ физикалық деңгейде сыртқы жадтқа өзге уақыттық сипаты бар басқа қолжетім әдістері қолданылады. Бұл оперативтік

жадқа тиімді болатын деректер құрылымы мен алгоритмдер сыртқы жадқа олай болмайтын екендігіне алып келеді.

Деректердің типтеріне сәйкес орындалатын амалдардың әр түрлі типтері бар. Сондықтан амалды орындау құрылғысының сәйкес әр түрлі бөліктері болады. Бұл құрылғыда өзінің бөліктеріне тән меншікті регістрлері бар. Оларда амалдардың деректері мен нәтижелері орналасады, сондай-ақ, программа пәрмендері мен басқарушы ақпарат уақытша сақталады. Амалдарды орындау құрылғысы мен басқару құрылғысын біріктіріп *орталық процессорр* деп атайды, оның құрылылымы II.1.1.2-суретте көрсетілген.

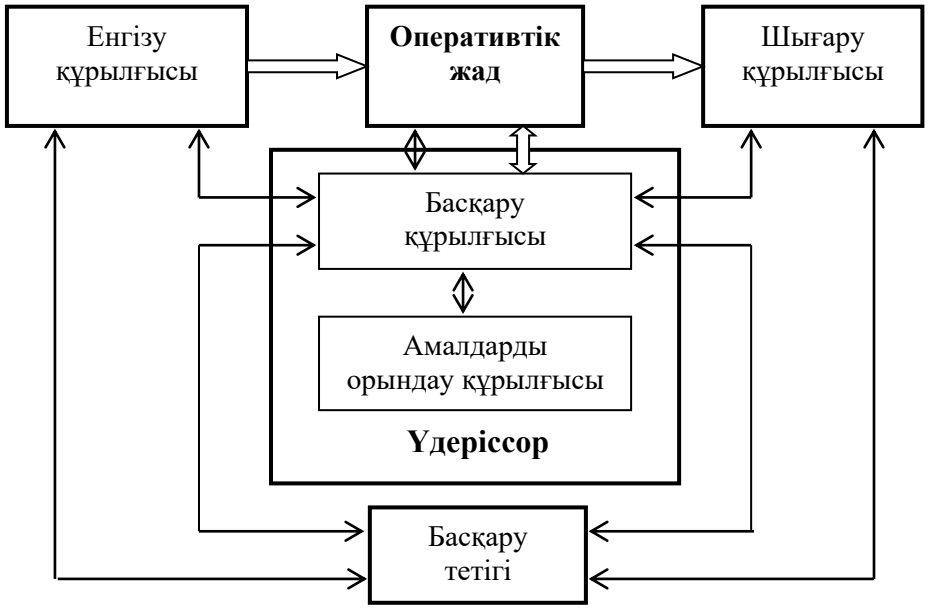
I.1.1.2. Сурет. Орталық үдеріссордың құрылымы

Компьютер сәулетіне сәйкес *процессорддың разрядтылығы* бар. Ол процессорддың бір амалды орындау үшін алынатын ақпараттың көлемдік мөлшері және оған сәйкес регістрдің сыйымдылығына тең болады. Процессорддың разрядтылығы

әрқашан 2-нің дәрежесіне тең болуы керек. Мысалы, 2^3 бит = 8 бит, 2^4 бит =16 бит, 2^5 бит = 32 бит, 2^6 бит = 64 бит және т.с.с.

Компьютердің *сыртқы жадтары* көлемі көп деректерді ұзақ уақытқа сақтау үшін арналған. Олардың қатарына *қатқыл диск*, *магниттік диск*, *лазерлік диск* сияқтылар жатады

Әмбебап ЦК жасаудың негізгі прінсіптерін ең алғаш АҚШ ғалымы Джон фон Нейман 1946 жылы жариялады. Олар бірігіп «фон Нейман сәулеті» деп аталды, оны кейде компьютердің классикалық сәулеті дейді, ол II.1.1.3-суретте көрсетілген.



II.1.1.3-сурет. Компьютердің фон-Нейман (классикалық) саулеті

Мұнда жіңішке бағыттама басқару сигналдарының, ал қосарланған бағыттама деректер мен нұқаулардың жолдары.

Енгізу құрылғысы деректер мен программаларды сыртқы жадтан оперативтік жадқа кіргізу, шығару құрылғысы программа орындалғанда шығатын нәтижелерді оперативтік жадтан сыртқы жадқа жіберу бойынша жұмыстарды атқарады.

Компьютердің классикалық сәулетінде көрсетілмеген көптеген сыртқы құрылғылар бар. Олар адамдардың компьютермен жұмыс істеуі ыңғайлы болуы үшін жасалынған құрылғылар. Мысалы, олардың қатарына *монитор* (*көру құрылғысы*), *принтер* (*баспа құрылғысы*), дискімен жұмыс істеу құрылғысы және т.б. жатады.

Кейде олардың барлығын компьютердің *сыртқы* (*қосалқы*) *құрылғылары* деп атайды.

II.1.1. Мысалдар:

1. Компьютермен жұмыс істеу ақпаратты енгізу, енгізілген ақпаратты өңдеу, нәтижені шығару деген кезеңдерден тұрады.

2. Оперативтік жад сыйымдылығының бірліктері сегізге еселі, яғни: 8 Мб, 16 Мб, 32 Мб, 64 Мб, 128 Мб, 256 Мб, 512 Мб, 1024 Мб (1 Гб), 2048 Мб (2 Гб) и 4096 Мб (4 Гб) және т.б.

3. Оперативтік жад пен процессорр арасындағы деректер алмасу екі тәсілмен орындалады: тікелей; регістр арқылы.

II.1.1. Тапсырмалар:

1. Компьютер жадының барлық мүмкін болатын көлемін есептеңіз, егер оның адрестік сөзі үш байттан тұратын болса.

2. Процессордың разрядтылығын анықтаңыз, егер машиналық сөз төрт байттан тұрса.

3. Жад сыйымдылығының өлшем бірліктері арасындағы мына қатынастарды түсіндіріңіз.

1 килобайт	$= 1024^1$	$= 2^{10}$	= 1024 байт
1 мегабайт	$= 1024^2$	$= 2^{20}$	= 1 048 576 байт
1 гигабайт	$= 1024^3$	$= 2^{30}$	= 1 073 741 824 байт
1 терабайт	$= 1024^4$	$= 2^{40}$	= 1 099 511 627 776 байт
1 петабайт	$= 1024^5$	$= 2^{50}$	= 1 125 899 906 842 624 байт
1 эксабайт	$= 1024^6$	$= 2^{60}$	= 1 152 921 504 606 846 976 байт
1 зеттабайт	$= 1024^7$	$= 2^{70}$	= 1 180 591 620 717 411 303 424 байт
1 йоттабайт	$= 1024^8$	$= 2^{80}$	= 1 208 925 819 614 629 174 706 176 байт

Көмек:

1. Бір байтта барлығы $2^8 = 256$ бір-бірінен бөлек ақпарат жазуға болады, ал бізде үш байт бар екендігін ескеру керек.

2. Процессорр разрядтылығы оның машиналық сөзіндегі биттер санына дәл болады.

3. Кестенің бірінші бағанында бірлік атаулары, екінші бағанында 1024-тің дәрежесі, үшінші бағанында 2-нің дәрежесі, ал төртінші бағанында үлкен бірліктердегі байт саны.

II.1.1. Сұрақтар:

1. Ақпараттық хабардың түріне байланысты компьютердің топтарын атаңыз?

2. Компьютердің классикалық сәулетіндегі қос бағыттама нені білдіреді?

3. Компьютердің адрестік сөзі мен машиналық сөзінің айырмашылығы неде?

II.1.1. Тесттер:

1. Ең алғаш компьютерді құру прінсіпін ұсынған кім?

A) Ч. Бэббидж;

B) Джон фон Нейман;

C) В. Лейбниц;

D) А. Тьюринг;

E) В. Ондер.

2. Компьютердің қандай негізгі құрылғылары бар?

A) оперативтік жад, процессорр, басқару құрылғысы;

B) оперативтік жад, процессорр, басқару пульті;

C) оперативтік жад, процессорр, принтер;

D) оперативтік жад, процессорр, монитор;

E) процессорр, принтер, пернетақта;

3. Жадтың ең кіші сыйымдылық бірілігі не?

A) килобит;

B) бит;

C) Мегабит;

D) Гигабит;

E) Терабайт.

II.1.2. Компьютердің жұмыс істеу прінсіпі мен ырғағы

Классикалық сәулетке сәйкес құрастырылған қазіргі компьютерлердің фон-Нейман ұсынған мынадай *жұмыс істеу прінсіптері бар*:

1) *жадтың адрестік прінсіпі* – компьютер жады байттар тіркесінен тұрады және әрбір байттар тіркесінің өзіндік адресі болады, ал деректер мен пәрмендердің жадқа жазылуы мен оқылуы осы байттар тіркесінің адресін көрсетумен жүзеге асады;

2) *деректер мен пәрмендердің бірдей болу прінсіпі* – пәрмендерді оперативтік жадқа жазғаннан кейін оларды деректер сияқты оларға амалдар қолданып басқа пәрмендер алуға болады;

3) *программалық басқарылу прінсіпі* - программа оперативтік жадқа жазылғаннан кейін компьютердің барлық жұмысы осы программаның орындалуымен басқарылады;

4) *компьютер жұмысының дискретті болу прінсіпі* – компьютер жұмысы жеке қадамдардың бос емес ақырлы тізбегі.

Енді компьютердің жұмыс ырғағын сипаттауға қатысы бар амалдарды орындайтын құрылғының *регістр* деп аталатын меншікті жадын сипаттайық. Регістр өте тез істейді және оның сыйымдылығы аз болады. Регістрдің саны бірнеше болуы мүмкін. Сондықтан оларды нөмірлейді немесе оларға өздерінің атқаратын қызметіне байланысты атаулар беріледі:

1) *пәрмен адресінің регістрі* (ПАР) орындалатын пәрменнің адресін сақтайды, оның сыйымдылығы адрестік сөзге тең;

2) *пәрмен регістрі* (ПР) орындалатын *пәрменнің* өзін сақтау үшін аргалған, оның сыйымдылығы машиналық сөзге тең;

3) *алғашқы және нәтиже деректер регістрі* (АНДР) орындалатын амалға қажетті алғашқы деректер мәнін немесе амал орындалып біткеннен кейін шығатын нәтиженің мәнін сақтау үшін арналған, оның сыйымдылығы машиналық сөзге тең;

4) *нәтиже белгісінің регістрі* (НБР) пәрмен бойынша амалды орындағаннан кейін шығатын нәтиженің қасиетіне байланысты 0 немесе 1 жазылады (мысалы, нәтиже теріс болса, онда 0 жазылады, әйтпесе 1 жазылады) сақтауға арналған, сыйымдылығы бір бит;

5) *ерекше жағдайдың регістрі* (ЕЖР) пәрменді орындау кезінде туатын «ерекше жағдайдың» белгіні сақтау үшін арналған, оның сыйымдылығы бір бит.

Компьютер әрбір қадамын өзінің жұмыс ырғағына сәйкес орындайды.

Компьютердің жұмыс ырғағы деп берілген программадағы бір пәрменді орындау кезінде туатын жұмыстарды айтады. Компьютер осы программаны түгел орындауы үшін өзінің жұмыс ырғағын бірнеше рет қайталауы керек.

Төменде классикалық құрылымындағы *компьютердің жұмыс ырғағы* талқыланады. Ол үшін мынадай жағдайлар қарастырылады:

1. Егер орындалатын программа және оған қажетті алғашқы деректер оперативтік жадта (ЖЖ) жазылған болса, онда жұмыс ырғағын бастау үшін басқару тетігі арқылы осы программаның ең бірінші бұйрығының адресін пәрмен адресінің регістріне (ПАР) жазып, басқару құрылғысын (БҚ) іске қосу қажет. БҚ осы адрес бойынша пәрменді оқу үшін ЖЖ–ға сигнал жібереді, содан кейін оқылған пәрменді пәрмен регістріне (ПР) жазады. Ары қарай БҚ осы пәрмен бойынша қандай амал немесе нұсқау орындалатындығын анықтайды.

2. Егер пәрмен бойынша амал орындалса, онда бұл амалға қажетті деректердің адрестерін анықтайды да осы адрестер бойынша деректерді ЖЖ–дан оқу және оларды алғашқы және нәтиже деректер регістріне (АНДР) жазуға басқару сигналдарын жібереді. Кейін БҚ анықталған амалды орындауға, одан шыққан нәтижені АНДР-ге жазуға амалдарды орындау құрылғысына (АОҚ) басқарушы сигнал жібереді де, келесі орындалатын пәрменнің адресін ПАР-ға жазады. Амал орындалып біткеннен кейін БҚ шыққан нәтижеге байланысты нәтиже белгісінің регістріне (НБР) және ерекше жағдайдың регістріне (ЕЖР) мәндер жазады да, нәтиженің мәнін АНДР-ден ЖЖ–ға жазады.

3. Егер пәрмен бойынша нұсқау орындалуы керек болса, онда бұл нұсқауды БҚ өзі орындайды.

Нұсқаулардың үш түрі болады: *тоқтату*, *шартсыз басқаруды беру* және *шартты басқаруды беру*.

Тоқтату нұсқауын орындаған кезде компьютердің барлық жұмысы тоқтайды.

Шартсыз басқаруды беру нұсқауында келесі орындалатын пәрменнің адресі көрсетіледі. Сондықтан шартсыз басқаруды беру нұсқауын орындаған кезде БҚ осы нұсқауда көрсетілген келесі орындалатын пәрменнің адресін ПАР–ға жазады.

Шартты басқаруды беру нұсқауында келесі орындалатын пәрменнің адресі белгілі бір шарттың орындалуына байланысты болады (мысалы, егер шарт орындалса, онда келесі орындалатын пәрмен осы нұсқауды көрсетілген адреспен табылады). Әдетте, шарт бұрын орындалған пәрмендердің нәтижесіне байланысты құрылады. Сондықтан, шартты басқаруды беру нұсқауын орындаған кезде БҚ белгілі мәнді АНДР–дегі немесе ЕЖР–дегі бар мәнмен салыстырады. Егер олар тең болса, онда ПАР–ға осы нұсқауда көрсетілген адрес жазылады, әйтпесе ПАР–ға ЖЖ–дағы осы орындалудағы пәрменнен кейінгі пәрменнің адресі жазылады.

Қорытындысында компьютердің жұмысына тікелей байланысты негізгі құрылғы процессоррдың қасиеттерін талдайық. Олар *процессоррдың жиілігі*, *процессоррдың жылдамдығы* және *процессоррдың өнімділігі*.

Процессоррдың жиілігі деп процессоррдың бір секундта орындайтын *тактылар* (бір пәрменді орындау кезінде туатын жұмыстар) санын айтады, оның өлшемі бірлігі *мегагерц* (*Мгц*), *гигагерц* (*Ггц*). Мысалы, XX ғасырдың 80-ші жылдардың ортасында шыққан ең үздік 32 разрядты Intel 80386 процессорларының жиілігі 25 Мгц–тен аспаған, ал кейін Intel Pentium IV процессорларының жиілігі 1,70 *Ггц*-ке жетті, яғни бұл көрсеткіш мың еседей өсті.

Процессоррдың жылдамдығы дегеніміз процессоррдың бір секундта орындай алатын амалдар саны, оның өлшем бірлігі *амал/секунд* (*а/с*). Мысалы, 20-ғасырдың 50-ші жылдарында шыққан компьютерлер бір секундта бірнеше мың ғана амалдар

орындады (Стрела – 3000 *а/с*), 70-ші жылдары кең тараған IBM 360 және ЕС ЭВМ тегіндегі компьютерлер секундына бірнеше жүз мың амалдар орындай алған, бірінші суперкомпьютер Cray–1-дің жылдамдығы 100 млн *а/с* болды, ал қазір кейбір компьютерлер бір секундта жүздеген миллиард амалдар орындай алады, яғни, компьютерлердің жылдамдығы жарты ғасыр ішінде бірнеше миллион есе өсіп отыр. Келешек нанокомпьютерлер мен нейрокомпьютерлердің жылдамдықтары одан ары өспекші.

Процессорлдың өнімділігі есептер тобына (ғылыми–техникалық, экономикалық және т.б.) тәуелді болады. Әр түрлі есептер үшін амалдардың салмағы әртүрлі болады. Мысалы, ғылыми–техникалық есептер үшін қабылданған амалдардың салмақтары II.1.1- кестеде келтірілген.

II.1.1- кесте. Ғылыми–техникалық есептердегі амалдар салмағы.

№	Амалдың аты	Салмағы
1	Тұрақты нүктелі нақты сандарды қосу және алу	33
2	Тұрақты нүктелі нақты сандар үшін көбейту	0,6
3	Тұрақты нүктелі нақты сандар үшін бөлу	0,2
4	Жылжымалы нүктелі нақты сандарды қосу, алу	7,3
5	Жылжымалы нүктелі нақты сандар үшін көбейту	4,0
6	Жылжымалы нүктелі нақты сандар үшін бөлу	1,6
7	Логикалық инверсия	1,7
8	Логикалық конъюнкция	1,7
9	Логикалық дизъюнкция	1,7
10	Шартсыз басқаруды беру	17,5
11	Салыстыру	4,0
12	Шартты басқаруды беру	6,5
13	Индекстеу	19,0
14	8 разрядқа жылжыту	4,6

Егер *i*-ші амалдың есептеу динамикасындағы кездесу жиілігі (салмағы) p_i, ал оның орындалу уақыты t_i болса, онда процессорлдың өнімділігін Гибсон әдісім ен былай есептелінеді:

$$P = \frac{\sum_{i=1}^{n} p_i}{\sum_{i=1}^{n} p_i t_i},$$

Компьютерлердің өнімділігін өлшеу үшін қажет бірлік атаулары келесі сөзден туындаған: флопс - FLOPS (Floating-point Operations Per Second) - берілген компьютер секундына қанша жылжымалы нүктелі амалын орындайтындығын көрсететін компьютер өнімділігін өлшеуге қолданылатын жүйеден тыс бірлік.

Қазіргі кезде компьютерлердің өнімділігін өлшеу үшін қолданып жүрген бірліктер тізімі II.1.2 - кестеде келтірілген.

II.1.2 - кесте. Компьютерлердің өнімділігін өлшеу бірліктері.

№	Өлшеу бірлігі	Пайдаланылған жыл	Дәреже 10	Сандық атауы
1.	Флопс	1941	10^0	Бір
2.	Килофлопс	1949	10^3	Мың
3.	Мегафлопс	1964	10^6	Миллион
4.	Гигафлопс	1987	10^9	Биллион (Миллиард)
5.	Терафлопс	1997	10^{12}	Триллион
6.	Петафлопс	2008	10^{15}	Квадриллион
7.	Эксафлопс	2019-ден кейін	10^{18}	Квинтиллион

II.1.2 Мысалдар:

1. XX-ғасырдың 60-жылдардың аяғында дүние жүзіндегі ең жылдам компьютер болып кеңестік электрондық есептеу машинасы БЭСМ-6 есептелді, оның жылдамдығы секундына 1 миллион амалдар, яғни, 1 Мегафлопс болды.

2. 2000-жылдары Intel Pentium процессоррлы дербес компьютерлердің өнімділігі секундына 50 миллион амал, яғни, 50 Мегафлопс болды.

3. 2000-жылдардың аяғында дүние жүзіндегі ең жылдам компьютер болып қытайдың Tianhe-2 суперкомпьютері есептелді, оның өнімділігі 30.7 петафлопс болды, ол американдық Cray суперкомпьютерінен екі еседей жылдам.

II.1.2 Тапсырмалар:

1. Компьютердің жұмыс прінсіпін атаңыз.
2. Регістрлердің түрлерін атаңыз.
3. Компьютердің жұмыс ырғағын сипаттаңыз.

Көмек:

1. Бұл прінсіптер негізінен компьютердің негізгі құрылғыларына, деректеріне, пәрмендеріне қатысты.
2. Регістрлер жылдамдығы жоғары аз көлемді құрылғы, ол компьютердің негізгі құрылғылары арасында байланыс жасайды.
3. Компьютердің жұмыс ырғағы тактылардан

II.1.2 Сұрақтар:

1. Процессорддың разрядтылығы дегеніміз не?
2. Процессорддың жиілігі дегеніміз не?
3. Процессорддың жылдамдығы дегеніміз не?

II.1.2 Тесттер:

1. Процессорр жиілігі немен өлшенеді?

A) герц;
B) байт;
C) флопс;
D) секунда;
E) пәрмен.

2. Процессорр жылдамдығы немен өлшенеді?

A) амал/секунд;
B) нұсқау/секунд;
C) пәрмен/секунд;
D) программа/секунд;
E) байт/секунд.

3. Компьютер өнімділігі немен өлшенеді?

A) флопс;
B) герц;
C) байт;
D) секунда;
E) пәрмен.

II.1.3. Компьютерлердің буындары

Компьютерлерді жіктеу тәсілдердің бірі – ол компьютерлерді буынға бөлу болып табылады. Жалпы, компьютерлерді буынға бөлу шартты негізінен олардың элементтер базасының өзгеруіне, өзінің құрамына кіретін құрылғылардың түрлері мен қасиеттерінің өзгеруіне және компьютерлер арқылы шығарылатын есептердің жаңа топтарының пайда болуына сәйкес болатын программалық қамтымдарының өзгеруіне тәуелді. Қазір компьютерлердің алты буыны белгілі деп айтуға.

Компьютердің бірінші буыны (1948 – 1958 жж.). Бұл буында компьютерлерінің басқару құрылғыларын құрастыру элементі электрондық лампалар болды, олардың жылдамдықтары ондаған мың *а/с*., разрядтылығы 31–43 бит, оперативтік жадтарының көлемі 1–4 Кб., амалдардың жұмыс ырғағы қатал тізбекті, яғни, келесі орындалатын амал ағымдағы амалдың орындалуы толық біткеннен соң ғана басталады, енгізу/шығару амалдары орындалып тұрғанда орталық процессорр тоқтап тұрады. Программа негізінен машиналық тілде қолмен жазылып орындалды. Жұмыс істеу режімі ашық болды, яғни, әрбір программалаушы басқару тетігінде өзі отырып программасын енгізіп жұмыс істетті. Негізінен сандық шамалармен байланысты болатын ғылыми–техникалық есептер шығарылды, символдық шаманы пайдалану жоқ болды. Стандартты программалар жасала басталды. Бұл компьютерлердің салмақтары 5–30 тонна болды және көлемдері өте үлкен болғандықтан, оларды орналастыру үшін жеке үлкен бөлмелер немесе ғимараттар қажет болды. Компьютердің классикалық сәулет прінсіптерін толық жүзеге асырып жасалынған ең бірінші компьютер «EDSAC» деген атпен 1949 жылы Англияда, Кембридж университетінде жасалынды. Бір жылдан кейін 1950 жылы «EDVAC» атты әмбебап ЦК АҚШ–та шықты. Бұрынғы Кеңес одағында ең бірінші «МЭСМ» деген компьютердің жасалуы 1947 жылы басталып 1951 жылы аяқталды, ал 1952–1953 жылдары келесі «БЭСМ–1» компьютері жасалынды. Ал еркін саудаға

шыққан ең бірінші компьютер UNIVAC (Universal Automatic Computer) 1951 жылы АҚШ-та жасалды.

Компьютердің екінші буыны (1955 – 1967 жж.). Бұл буынның компьютерлері транзисторлық болды, жылдамдықтары жүздеген мың а/с., разрядтылығы 31–48 бит, оперативтік жадтарының көлемі 8–128 Кб. Процессоррдың жұмысын *үзу* және оны өңдеу жүйесі пайда болды (ол негізінен, енгізу/шығару амалдарын орындау кезінде іске қосылады). Машиналық тілдің екілік кодтарын мнемоникалық кодтарға айналдырған ассемблер тілдері шықты. Алгоритмдік тілдерден машиналық тілге автоматты аударатын программалар – ассемблерлер, трансляторлар шықты, яғни, программа құру үшін деңгейлері жоғары келесі программалау тілдері пайдаланылды:

Fortran (1957) АҚШ – сандық есептерді шығару үшін;

Cobol (1958) АҚШ – экономикалық есептерді шығару үшін;

Lisp (1958) АҚШ – символдық деректерді өңдеу және шешім қабылдау үшін;

Algol (1958–1960) IFIP – ғылыми–техникалық есептерді шығару үшін;

PL/1 (1960) АҚШ – әмбебап программалау тілі.

Сонымен қатар, стандартты программалардың қоры үлкейді, жабық жұмыс істеу режімі қолданылды, яғни, программалаушы тікелей машинамен жұмыс істемейтін болды, ол өзінің жоғары деңгейдегі программалау тілінде жазылған программасын ары қарай машинадан өткізетін қызмет көрсететін топқа тапсырды. Программалардың жұмыс істеуін бақылау және басқару үшін алғашқы *мониторлық жүйелер* пайда болды, олардың өзінің тапсырмаларды басқару тілдері болды. Индексті арифметиканың шығуы, тікелей емес адрестеуді және динамикалық жадты қолдану, символдық шамалармен жұмыс істеу мүмкіншілігінің пайда болуы осы буынның құрылымдық ерекшелігін айқындады. Бұл компьютерлерге АҚШ-та UNIVAC, LARC, CDC 6600, IBM-1401, -7030, ал КСРО-да Урал-11, -14, -16 Минск-2, -12, -14, -22, М-20, -220, -222, БЭСМ-2, -4, -6, Мир-1, Наири жатады, олардың

оперативтік жадтың көлемі 32 Кбайттан 64 Кбайтқа шейін жетті, ал жылдамдықтары 20–30 мың *а/с* болды. Осы буынның ең жылдамы БЭСМ-6 болды, оның жылдамдығы 1 млн *а/с* жетті.

Компьютердің үшінші буыны (1968 – 1973 жж.). Бұл буынға интегралды микросұлбалар арқылы жасалынған компьютерлер мен компьютерлердің кешендері кіреді. Олардың жылдамдықтары миллиондаған а/с., разрядтылығы 32–64 бит, оперативтік жадтарының көлемі 64–1024 Кб жылдамдықтары 10 млн *а/с* жетті. Дамыған үзу жүйесі бар, енгізу/шығару амалдарының орындалуы орталық процессорлардың жұмысымен параллель жүргізетін қосымша процессорлар (арналар) қолданылды. Бұрын программалар атқаратын көп жұмыстар, соның ішінде үзуді ұйымдастыру мен өңделулер аппарат арқылы жүзеге асатын болды. Компьютердің сыртқы ортаны қабылдай және оған әсер ете алатын сенсорлық қондырымдар пайда бола бастады. Осылар компьютерді алдын ала енгізілген деректерді детерминді (бірмәнді) өңдейтін құрылғыдан сыртқы ортада туатын жағдайға қарай жұмыс істей алатын зерделі құрылғыға айналдырды. Оперативтік жадты қорғау және динамикалық бөлу іске асты. Көптеген есептерге, солардың ішінде символдық есептерге (Snobol, Lisp, Refal сияқтылар) және логикалық есептерге (Prolog, Miranda сияқтылар) жоғары деңгейлі проблемаға бағытталған программалау тілдері қолданылды, символдық есептер мен логикалық есептер үлесі көбейді. Программалардың жұмысын бастан аяқ басқаратын (сыртқы және ішкі ортадағы жағдайларға мақсатты жауап бере алатын) дамыған операциялық жүйелер жұмыс істеді.

Осы буынның негізгі ерекшелігі программалары төменнен жоғары қарай ұйқас болатындай біртекті және мүмкіншіліктері өспелі компьютерлердің бірнеше моделдерінен тұратын машиналар кешенінің пайда болуы. Мысалы, КСРО-да ЕС ЭВМ 1020–1050, ал АҚШ-та IBM/360–370 сияқты компьютерлердің бірыңғай жүйелері. Бұл компьютерлер арқылы оперативтік жадты немесе сыртқы құрылғылардың өрісін ортақ етуге болатын есептеу жүйелерін жасауға мүмкіншілік туды. Бір уақытта бірнеше программа жұмыс

істей алатындай етіп орталық процессоррдың уақытын бөлшектейтін мультипрограмдық режім іске асырылды. Сонымен қатар, нақты уақыт масштабында жұмыс істей алатын программалар пайда бола бастады. Олар технологиялық үдерістерді, ұшатын аппараттардың және басқа күрделі құрылғылардың жұмыстарын басқаруға мүмкіндік берді. 1969 жылы алғашқы глобалды компьютерлік тор пайда болды – ол қазір Интернет деп аталатын ғаламтордың ұрығы болды. Сол жылы бір мезгілде амалдық жүйе Unix және C («Си») программалау тілі шықты, олар программалық әлемнің дамуына керемет зор ықпал етті және осы күнге шейін өздерінің алдыңғы қатарлығын сақтауда.

Компьютердің төртінші буынын (1974 – 1982 жж.) 1971 жылы үлкен интегралды электрондық сұлба негізінде Intel фирмасы (АҚШ) құрастырған микропроцессоррдың пайда болуымен байланыстырады. Бұл буынның компьютерлері жоғары деңдейдегі программалық тілдерді тиімді пайдалану және проблемалық программистер үшін программалау үрдісін жеңілдету мақсатында жобаланды және олар негізінен ғылымдағы, өндірістегі, басқарудағы, денсаулық қорғаудағы, қызмет көрсетудегі және тұрмыстағы еңбек өнімділігін жылдам көтеруге арналды. Аппараттық жағынан олар үлкен немесе өте үлкен интегралданған микросұлбалар арқылы жасалынды. Интегралданудың жоғарғы дәрежесі электрондық аппаратуралардың орналасуын тығыздалуына әкелді, ал ол өз кезегінде компьютерлердің көлемін кішірейтуге, тезәрекеттігін ұлғайтуға және құнын арзандатуға әкеледі. Мұның бәрі компьютердің логикалық құрылымы мен программалық қамтымына едәуір әсер етті. Компьютердің құрылымы мен программалық қамтымы арасындағы байланыс тығыздала бастады, әсіресе амалдық жүйе программалары адамның қатысуынсыз компьютердің жұмысын үздіксіз жүргізуді ұйымдастырады. Мұндай компьютерлерге көппроцессоррлы *суперкомпьютерлер* мен *микрокомпьютерлер* (кейін оларды *дербес компьютерлер* деп атап кеткен) жатады. Суперкомпьютерлердің жылдамдықтары жүз миллионға шейін (мысалы, Cray-1

суперкомпьютерінің жылдамдығы 100 млн *а/с.*). Жалпы осы буындағы компьютерлер арқылы байланыс әдістері одан әрі дамып телефон, телеграф желілеріне қосылып *компьютерлік локалды* (*жергілікті*), *корпоративтік* және *глобалды торлар* (мысалы, Интернет) құрылды, өте үлкен *деректер архиві* жиналды, деректердің визуалды (бейнелік) түрдегі берілуі және өңделуі дамыды, нақты уақыт масштабында жұмыс істей алатын жүйелер кеңінен жүзеге асты. Бұл буынға КСРО-да мыналар кіреді: ЕС-1015, -1025, -1035, -1045, -1055, -1065, -1036, -1046, -1066, СМ-1420, -1600, -1700, барлық дербес компьютерлер (“Электроника МС 0501”, “Электроника-85”, “Искра-226”, ЕС-1840, -1841, -1842). Сонымен қатар, бұл буынға өте қуатты көппроцессоррлы есептеу кешендері "Эльбрус-1, -2, 3" және суперкомпьютерлер Amdahl 470V16 жатады, пәрмендерді конвейерлі өңдеу негізінде тиімді параллелдік прінсіпті пайдаланды. Кейін суперкомпьютер тобына тезәрекеттігі 20 мегафлопс (1 мегафлопс = 1 млн жылжымалы нүкте амалдары/секунд) моделдерді жатқызды. Осындай бірінші модел болып АҚШ–тың 1975 жылы жасалынған ILLIAC-IV еспетелінеді, оның тезәрекеттігі 50 мегафлопс болды. Суперкомпьютер тарихында 1976 жылы АҚШ-та жасалынған тезәрекеттігі 130 мегафлопс болатын Cray-1 моделі ерекше орын алды. Моделдің архитектурасы өте үлкен интегралдық жүйесімен (ӨҮИЖ) деректерді векторлық және скалярлық өңдеудің конвейерлік прициніне негізделген болды. Кейінгі моделдер Cray-2, Cray X-MP, Cray-3, Cray-4 тезәректтікті 10 мың мегафлопсқа жеткізді, ал Cray MP моделі жаңа кремнийлік микросұлбада 64 процессоррды қамтып, тезәректтікті 50 гигафлопсқа дейін көтерді.

Біздің елде кең қолданыстағы 4-ші буынға жататын дербес компьютерлер. Оларды біз басқа бөлімде жетік қарастырамыз.

Компьютердің бесінші буыны (1982 – 1992 жж.) аса үлкен, ультраүлкен, гигаүлкен интегралды сұлба негізінде жасалды. Бұл буын 1980 жылы Жапония жариялаған он жылдық жобадан басталды, онда компьютердің машиналық тілі ретінде логикалық программалау тілі Prolog–ты аппаратты түрде жүзеге асырып,

жасанды интеллект жүйесін құру көзделді. Осы есептерді шешу үшін мынадай бағыттар ұсынылды:

1. Пайдаланушыға өз есебін шешу үшін компьютермен диалог жүргізу мүмкіндігін беретін қарапайым интерфейс жасау. Мұндай интерфейс екі тәсілмен ұйымдастырулуы мүмкін: табиғи тілмен және графикалық тілмен. Бірақ қолжетімді интерфейс проблеманы тек жартылай шешті. Себебі, ол программалық қамтымның дайындалуына қатыспай, оның алдын ала дайындалғанына ғана қатынас жасауға мүмкіндік береді.

2. Ақырғы пайдаланушыны программалық өнімдерді жобалауға қатыстыру. Бұл тапсырушыны программа жасау үрдісіне тікелей қосуға мүмкіндік береді, нәтижесінде ол программалық өнімдерді жасаудың уақытын қысқартуға және олардың сапасын көтеруге мүмкіндік береді. Мұндай технология ақырлы пайдаланушының кәсіби білімін автоформалдауға байланысты және программалық өнімді жобалауға екі кезең ұсынылады: программист әмбебап «бос» программалық қабықша жасайды, ол нақты біліммен толтырылады және соны пайдалану арқылы практикалық есептер шығарылады.

Жасанды интеллект жасаудағы негізгі құраушының бірі ретінде ғылым мен техника әртүрлі бағыттары бойынша білімдер базасын алуға болады. Білімдер базасын жасау мен пайдалану үшін есептеу жүйесінің жоғарғы тезәрекеттігі және үлкен көлемді жад болуы қажет. Білім базасын толтыру, өңдеу және жаңартуды қамтамасыз ететін программаларын жасау үшін арнаулы объектіге бағытталған және логикалық программалау тілдері жасалынды. Бұл тілдердің құрылымдары компьютерлердің дәстүрлі фон-неймандық архитектурасынан жасанды интеллект жасайтын есептердің талаптарын ескеретін басқа архитектураға көшуді талап етеді.

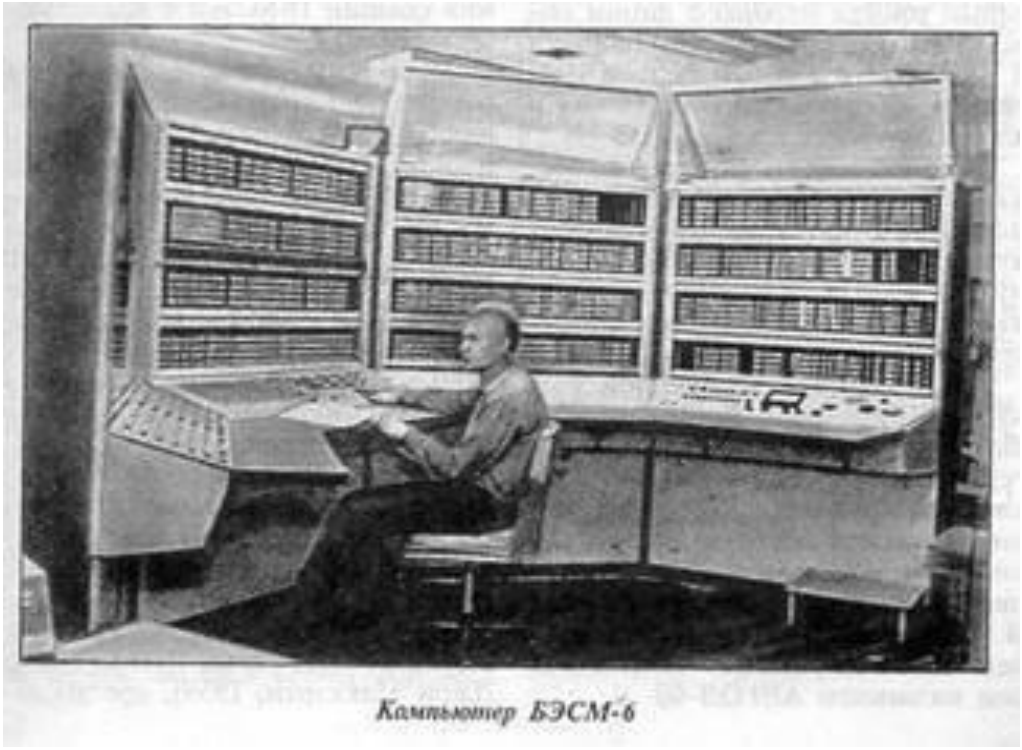
Дегенмен компьютерлердің бесінші буынына Pentium процессоррының негізінде құрастырылған виртуалды ортаны жасауға мүмкіндік беретін жылдамдығы 100 млн *а/с* мультимедиалық компьютер және жылдамдығы 120 млн *а/с* суперкомпьютерлер жатады. Оларға көпроцессоррлы Cray, Эльбрус сияқтылар жатады.

Компьютердің алтыншы буыны – өткен ғасырдың 90-шы жылдарының ортасынан бастап қолға алына бастады. Бұл компьютерлер жасанды *нейрон желісіне*, *көпмәнді логика* мен *бұлдыр логикаға* және *кванттық есептеу теориясы мен генетикалық алгоритмдерге* негізделіп жасалынады. Бұл компьютерлердің дамыған жасанды зердесі болады: олардың өзін–өзі оқытатын қабілеті және өздігінен кейбір мәселені түсініп (образды танып), жобалап, оны шешу немесе жүзеге асыру үшін керекті программаны немесе құрылғыны құрастыра алатын мүмкіншілігі болады.

II.1.3. Мысалдар:

1. Бірінші буынға жататын 1945 жылы АҚШ-та жасалған ең бірінші ЭНИАК (ENIAC) компьютерінің жады 20 сан–сөз, есептеу қуаты 1 секундта 357 көбейту амалы немесе 5000 қосу амалы, салмағы 27 тонна.
2. Екінші буынның ең бірінші сәтті коммерциялық миникомпьютері PDP-8 Англияда 1960 жылдары шыққан, 50 мыңнан астамы сатылды, ол кездегі ең үлкен компьютер саны.
3. IBM PC-сәйкес компьютерлер Қазақстанда ең көп тараған компьютерлер, олардың қуаты әрдайым өсуде, ал қолдану аясы кеңеюде, олар торға бірігіп пайдаланушыларға ақпаратпен алмасуына және бірмезгілде ортақ деректер базасына қолжетім жасауға мүмкіндік береді.

4. Екінші буын мен үшінші буын аралығындағы 1965 жылы КСРО-да жасалынған БЭСМ-6 компьютерінің жады 128 Кбайтқа, жылдамдығы 1 млн а/сек болды.



Компьютер БЭСМ-6

5. Үшінші буынға жататын 1964 жылы АҚШ-та шыққан 32-разрядты IBM 360 компьютерінің жады 16Mb, ал оның процессоррының тактылық жиілігі 1-ден 5MHz-ге дейін.

6. Төртінші буын компьютері Эльбрус-2 1985 жылы КСРО-да шыққан, өнімділігі 125 млн а/с (10 процессорр), жад 144 МБайт, сөзі 72 бит, енгізу–шығару арнасының өткізу мүмкіншілігі – 120 Мб/с.

7. Бесінші буын компьютері Жапонияда шыққан PIM/m-1.

II.1.3. Тапсырмалар:

1. Бірінші буын компьютерлерінің элементтік базасын

анықтаңыз.

2. Екінші буын компьютерлерінің программалық қамтымын анықтаңыз.

3. Компьютерлердің бесінші буынының ерекшеліктерін көрсетіңіз.

II.1.3. Сұрақтар:

1. Компьютердің элементтер базасы деген не?
2. Қомпьютерлердің қанша буыны бар?
3. Қомпьютерлердің 6-шы буынының қандай қасиеттері бар?

II.1.3. Тесттер:

1. Біздің елдегі кең қолданыстағы IBM PC қай буынға жатады?

A) Төртінші;
B) Бесінші;
C) Екінші;
D) Бірінші;
E) Үшінші.

2. Қазіргі таңда қолданыстағы компьютердің қанша буындары белгілі?

A) Алты;
B) Сегіз;
C) Төрт;
D) Тоғыз;
E) Он.

3. АҚШ-та шыққан iMac компьютерлердің қай буынына жатады?

A) Төртінші буын;
B) Бірінші буын;
C) Екінші буын;
D) Үшінші буын;
E) Бесінші буын.

II.1.4. Дербес компьютерлер

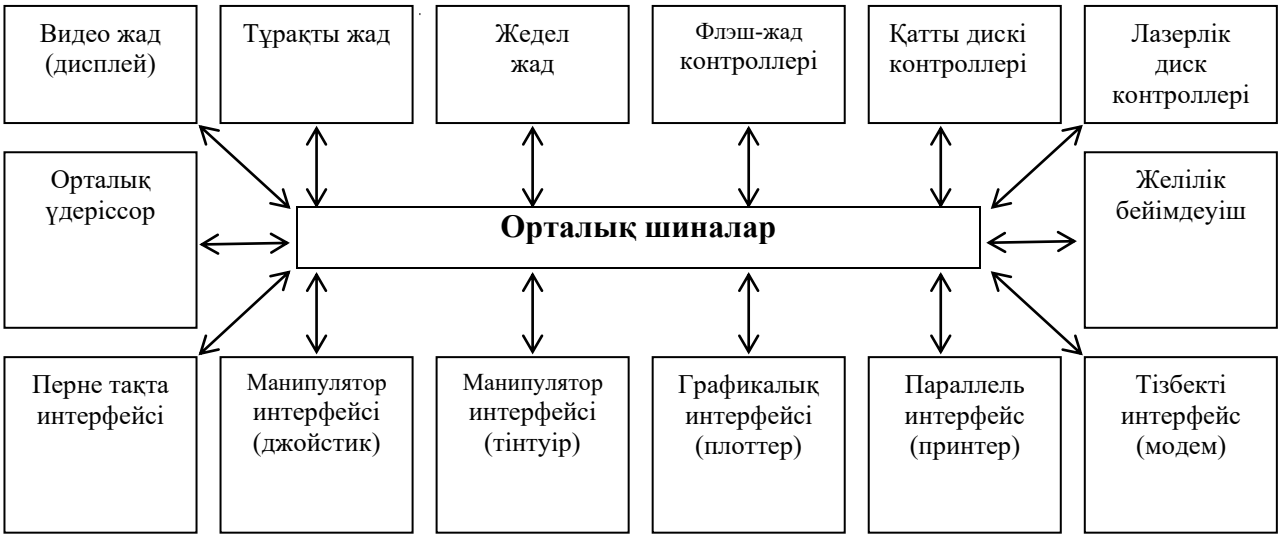
Жоғарыда компьютердің алғашқы үш буынына тән классикалық құрылымы сипатталды. Әрине, есептеу техникасын жасау технологиясы жылдам қарқынмен дамыған кезде бұл құрылым өзгермей қоймайды.

Жоғарыда айтылғандай компьютерлердің үшінші буыны транзистерден интегралды электрондық микросұлбаларға өтумен байланысты болды. Электрондық сұлбалардың көлемдерінің көп кішірейгені компьютерлердің негізгі құрылғыларының көлемдерін кішірейтуге алып келді де, өте жылдам жұмыс істейтін процессорлардың шығуына себеп болды. Өте жылдам істейтін негізгі құрылғылар мен ақырын жұмыс істейтін енгізу/шығару құрылғыларының арасында үлкен қарама–қайшылық туды. Сыртқы құрылғылардың жұмысын басқаратын процессорр көп уақытта сырттан келетін деректерді немесе нәтижені сыртқа шығаруды күтіп қарап тұрды. Бұл проблеманы шешу үшін процессоррды сыртқы дүниемен қатынас жасауды басқару міндетінен босатуды ұйғарылды, нәтижесінде ол жұмыс сыртқы құрығылардың процессорры деген арнаулы жасалынған құрылғыларға тапсырылды. Бұл құрылғылардың бірнеше атаулары бар: *байланыс арнасы*, *енгізу/шығару процессорры*, *контроллер.*

Контроллердің өзінің пәрмендер жүйесі болады және өзіне берілген сыртқы құрылғылардың жұмысын осы пәрмендер арқылы құрылған белгілі программа бойынша басқарады. Осындай программаның нәтижесі орталық процессорр оқи алатындай етіліп контроллердің ішкі регістріне жазылады. Берілген есепті шешетін программа орындалып жатқанда сыртқы дүниемен алмасу қажет болса, орталық процессорр ол туралы контроллерге тек тапсырма береді де, программаны орындауға байланысты өзінің басқа жұмыстарымен айналыса береді, ал сыртқы дүниемен алмасу жұмысын контроллердің өзі басқарады. Компьютердегі барлық байланыс орталық шина арқылы жүзеге асады. Шинаның үш бөлігі

болады: дере*ктер берілетін шина*, *адрестер берілетін шина*, *басқару берілетін шина*.

II.1.4-суретте дербес компьютердің құрылымы көрсетілген.



II.1.4. Сурет. Дербес компьютердің құрылымы

Жоғарыда көрсетілген архитектураны ашық деп атайды. Себебі, оған өзіңізге керек қана құрылғыларды алып ықшамдауға да болады немесе жаңа құрылғыларды қосып мүмкіншілігін күшейте беруге болады.

Деректерді шығаруға арнаулы құрылғы дисплей жасалынды, оның жұмыс істеу прінсіпі кәдімгі телевизор сияқты. Дисплей өте жылдам жұмыс істейтін болғандықтан оған арнаулы жад керек.

Қазіргі кезде дербес компьютерлерде деректер ағымының өсуіне байланысты шинаның бірнешеуі қажет бола бастады. Оның бір түрі өте жылдам істейтін құрылғылар үшін, ал екіншісі ақырын істейтін құрылғылар үшін ендірілді.

Сонымен дербес компьютерлердің ашық архитектурасы сыртқы құрылғылардың саны мен сапасының тұрақты дамып тұруына және мүмкіншіліктері жоғары жаңа орталық құрылғылардың шығуына себеп болды. Мұндай компьютерлерде орталық процессорлардан басқа бірнеше арнаулы (математикалық, видео) процессорлар жұмыс істей бастады және компьютер аралық байланыстар дамып неше түрлі (локалдық, корпорациялық, глобалды) компьютерлік желілер пайда болды.

Дербес компьютерлер әмбебап микропроцессорлар негізінде

құралады. 1974 жылы бірінші әмбебап 4500 логикалық элементті микропроцессорр Intel-8080 және микрокомпьютерлік технологияның стандарты жасалынды. Бұлар бірінші дербес компьютерлер жасаудың негізі болды. Intel-8080 негізінде Эдвард Робертс 1974 жылы ең бірінші дербес компьютер (ДК) жасады, оны Altair-808 деп атады. Осы компьютерге арнап Пол Аллен мен Бил Гейтс кең тараған қарапайым Basic тілінен транслятор жасады, бұл компьютердің интеллектуалдығын әжептеуір асырды. Осыдан кейін Altair-8800 ДК-ін 20-дан астам әртүрлі компаниялар мен фирмалар жасай бастады, ДК-индустрия пайда болды (ДК өндіру, оларды сату және жөндеу орталықтары, периодты баспалар, көрмелер, конференциялар және т.б.). 1977 жылы ДК-лерінің 3 моделін сериялық өндірісі жолға қойылды: Apple-2 (Apple Computers фирмасы), TRS-80 (Tandy Radio Shark фирмасы) және PET (Commodore фирмасы), олардың ішінде конкуренттің күресте ДК-лерді шығаруда Apple Computers фирмасы лидер болды, 1980 жылы оның жылдық табысы $117 млн болды.

1981 жылдан бастап IBM фирмасы өзінің кең тараған IBM PC/XT/AT и PS/2 деген ДК-лерін шығарып, ДК-лердің жаңа дәуірін ашты. Осыған байланысты пайдаланушыға маңызды стандарттау, бірізділеу, дамыған программалық қамтым және басқа сұрақтар оңтайлы шешіле бастады.

ДК-дің интерфейстері дамыған болғандықтан олармен жұмыс істеу ыңғайлы және тиімді. Ал ДК кең көлемде шығарыла бастағандықтан, олардың бағасы үнемі төмендей береді. Сондықтан оларды қолдану экономикалық жағынан да тиімді болды.

ДК-нің негізгі құрылғыларына жүйелік блок, монитор, пернетақта, тінтуір, ал қосымша құрылғыларына принтер, сканер, микрофон жатады.

Жүйелік блок жүйелік тақша (аналық плата), процессорр, оперативтік жад, қатқыл диск, қоректендіру блогы, бейнекарта, дискжетектер секілді көптеген маңызды құрылғылардан тұрады, оның артқы жағында монитор, пернетақта, тінтуір, принтер, модем, сканер, микрофонды қосатын порттар орналасқан.

Монитор компьютер өңдейтін ақпаратты шығаратын экраны бар құрылғы, ол сыртқы пішіні бойынша кәдімгі теледидарға ұқсайды.

Пернетақта компьютердің жұмысын басқара отырып, қажетті ақпаратты енгізу үшін қолданылатын құрылғы, яғни, ол әріптің, цифрдің және басқа таңбалардың пернелері көмегімен компьютерге кез келген ақпаратты енгізуге мүмкіндік береді.

Тінтуір экрандағы объектілердің белгілерін іздеу, таңдау, белгілеу және олар арқылы компьютер жұмысын басқаруға мүмкіндік беретін құрылғы.

Принтер ақпаратты қағазға басып шығаратын құрылғы, оның матрицалық, бүріккіш, лазерлік, сублимационалдық, жарық диодты түрлері бар.

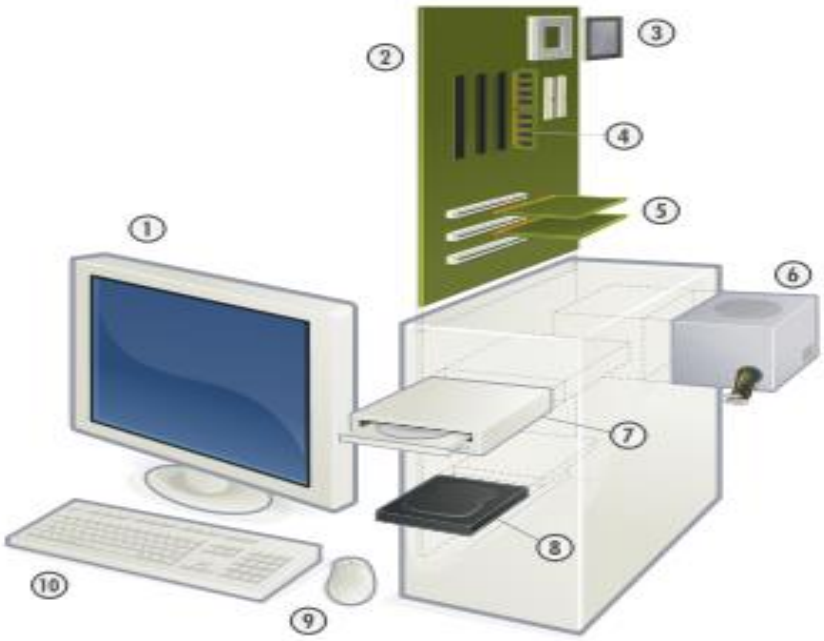
Сканер қағаздағы мәтін мен сурет кескінін компьютерге автоматты түрде енгізу үшін қолданылады, яғни, ол кескінді екілік кодқа ауыстырып, компьютердің жадына жазады.

II.1.4. Мысалдар:

1. Altair-8080 ДК-нің жүйелік блогы оперативтік жад 256 байт, бағасы $397, сыртқы құрылғыларды қосуға мүмкіншілігі бар

2. ДК IBM PC дербес компьютерінің түрі

3. Дербес компьютердің негізгі бөліктері.



Мұнда 1 – монитор, 2 – аналық тақта, 3 – орталық процессор, 4 – жедел жад, 5 – кеңейту картасы, 6 – көректену блогы, 7 – оптикалық қозғағыш, 8 – қатқыл диск, 9 – тінтуір, 10 –пернетақта.

4. Тасымалды ДК (Ноутбук – Notebook) түрі

5. Планшетті ДК түрі

II.1.4. Тапсырмалар:

Дербес компьютерлердің құрылымын көрсетіңіз.

Көмек:

Құрылымда ортақ шинаға барлық құрылғылар қосылады.

II.1.4. Сұрақтар:

1. Дербес компьютердің сыртқы құрылғыларына не жатады?
2. Барлық есептеулерді орындайтын электронды сұлба қалай

аталады?

3. Бірінші дербес компьютер қай жылы жасалынды?

4. Дербес компьютердің ортақ шинасы қандай қызмет атқарады?

5. Сыртқы құрылғылар жұмысын не басқарады?

II.1.4. Тесттер:

1. Дербес компьютерлердің шиналары қандай бөліктерге бөлінеді?

A) Деректер шинасы, адрестер шинасы, басқару шинасы.

B) Пәрмендер шинасы, адрестер шинасы, басқару шинасы, амалдар шинасы;

C) Деректер шинасы, алгоритмдер шинасы, басқару шинасы, сигналдар шинасы;

D) цифрлар шинасы, әріптер шинасы, адрестер шинасы, нұсқаулар шинасы;

E) Басқару шинасы, әдістер шинасы, пәрмендер шинасы, амалдар шинасы.

2. Дүние жүзі бойынша ең бірінші шығарылған дербес компьютер қайсы?

A) Apple;

B) Altair-808;

C) IBM PC XT;

D) Macintosh;

E) Yamaha.

3. Кез келген дербес компьютердің негізгі құрылғыларына не жатады?

A) жүйелік блок;

B) принтер;

C) сканер;

D) модем;

E) плоттер.

II.2. Компьютерде ақпаратты бейнелеу

II.2.1. Компьютерде таңбаларды бейнелеу

Ақпаратты компьютер арқылы өңдеу алдында оның жадында сақтау керек, онда ол тек 0 мен 1 таңбаларынан тұратын тізбек түрінде жазылады.

Ақпараттың 0 мен 1 таңбаларынан тұратын тізбек түріндегі кескінін *код* деп атайды. Ақпарат пен оның коды арасындағы тура сәйкестікті *кодтау* деп, ал оған кері сәйкестікті *декодтау* деп атайды (бұлар математикадағы тура функция және кері функция ұғымдарымен бірдей). Демек, компьютерге ендірілетін кез келген ақпарат *кодталуы* керек, ал компьютерден адам пайдалануы үшін алынатын ақпарат *декодталуы* (кері кодталуы) керек, яғни, символдық немесе графикалық түрге түрленуі керек. Осындай тура және кері сәйкестіктерді алу ережесін *кодтау ережесі* деп атайды.

Кез–келген құрама деректердің компьютерде бейнеленулері олардың құрамына кіретін қарапайым деректердің бейнеленулерінен құрастырылады. Қарапайым деректердің *логикалық, символдық* (таңбалық), және *сандық* болып бөлінетіні I.2-блогынан белгілі. Сондықтан осы қарапайым деректердің бейнелеу әдістерін қарастырсақ жеткілікті.

Логикалық деректер компьютердің жадында бір бит арқылы бейнеленеді. Оның мәні тек тек 0 немесе 1 болатыны II.1.1-дәрісінен белгілі, мұнда 0 логикалық жалғанға, ал 1 логикалық ақиқатқа сәйкес келеді.

Символдық деректер компьютердің жадында өзінің құрамына кіретін символдардың (таңбалардың) кодтарынан құралған 0 мен 1-ден тұратын тізбек арқылы бейнеленеді. Әрбір символдың коды кодтау кестесі деп аталатын екі өлшемді осы символ қамтылған жиымның индекстерінің мәндері арқылы беріледі. Кодтау кестесіндегі кодтаудың ретін *кодтау стандарты* деп атайды. Яғни, әрбір стандарт өзінің *кодтау кестесін* анықтайды. Ондай стандарттарға кең тараған ASCII, ANSI, Unicode және басқалары жатады. ASCII стандартында әрбір символ 8 бит, яғни 1 байтта

орналасады. ASCII батыс тілдеріне арналғандықтан, оны символдары ASCII қолдайтын 2^8 = 256-ға кірмейтін тілдері бар елдер мен аумақтарда пайдалану шектелген.

Осы шектеулерден айырылу өту үшін, Стандарттау жөніндегі халықаралық ұйымы (ISO - International Standards Organization) Latin-1 деп аталатын ASCII стандартына кірмейтін символдарды кодтаудың жаңа стандартын жасады. Microsoft фирмасы Latin-1 стандартын кеңейтіп оны ANSI деп атады. Бірақ ANSI бұрынғыша 8-битті кодтаумен қалды, ол тек 256 символды ғана бейнелей алады. Көптеген тілдерде мыңдаған символдар бар, айта кетсек, қытай, корей және жапон тілдерінде.

8-битті кодтау стандартының шектеулігінен шығу үшін Microsoft басқа Apple Computer, Inc, және IBM сияқты компаниялармен бірлесіп халықаралық символдар жиыны үшін символды кодтаудың жаңа стандартын анықтау мақсатында коммерциялық емес Unicode консорциумын құрды. Осында істелген жұмыс ISO-дағы жұмыспен біріктіріліп символдарды кодтаудың халықаралық стандарты Unicode шықты.

Unicode 16-биттік стандарт, ол барлығы 2^{16} = 65536 бір-бірінен бөлек әр түрлі таңбаларды кодтауға мүмкіндік береді. Сондықтан бұл стандарт арқылы кез келген ұлттық тілдің әліпби таңбаларын кодтауға болады. Ол тіпті көне түркі руникасын, санскрит және египет иероглифтері сияқты архаикалық тілдерді қолдайды және тыныс белгілерін, математикалық және графикалық символдарды қамтиды. Оның кодтау кестесінде қазіргі кезде ғылымның әртүрлі саласында қолданыста жүрген таңбалардың және әртүрлі декоративті таңбалардың кодтары бар.

Unicode стандартының бірінші үлгісі (1991 ж.) 16-битті бекітілген кеңдігі бар символды кодтауды қамтыды. Unicode-тың екінші үлгісінде (1996 ж.) кодтық кеңістікті кеңейту ұйғарылды: 16-биттік Unicode ендірілген жүйелермен үйлестікті қаматамасыз ету үшін Unicode символдарын кодтаудың 16 биттік сөздер түріндегі UTF (Unicode Transformation Format)-16 әдісін шығарды. Бұл кодтау Unicode символдарын жазуға мүмкіндік береді. Бір

символ UTF-16 кодтауда екі байт тізбегімен бейнеленеді. Екеудің алдыңғысы, үлкен немесе кіші, байт ретіне тәуелді. Ретті анықтау үшін байт ретінің тамғасы пайдаланылады.

Осы стандарттар негізінде әрбір ұлттық тілдің әліпби таңбаларына қатысты ұлттық стандарттар жасауға болады. Бұл стандарттар қазақ мәтіндерін баспа жүйесі мен графикалық пакеттерде, Web беттерді жасаған кезде және электрондық поштаны пайдаланғанда және т.б. жағдайларда Internet жүйесінде теру, жинақтау, өңдеу, басу және тасымалдауын қамтамасыз етуі қажет.

Қазақ әліпбиі әріптерінің 8-биттік кодтары MS DOS үшін II.2.1-кестеде көрсетілген.

II.2.1.1-кесте. MS DOS үшін қазақ әліпбиінің 8-битті кодтары

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
0		У		П		Р		А		В		Л		Я		Ю		Щ		И		Е			С	И	М	В	О	Л	Ы		1
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?	3
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_	5
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	⌂	7
8	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	9
A	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п	▒	▒	▓	│	┤	┤	╣	╗	╗	╣	╗	╝	╝	┘	┐	B
C	└	┴	┬	├	─	┼	╞	╟	╚	╔	╩	╦	╠	═	╬	╧	╨	╤	╥	╙	╘	╒	╓	╫	╪	┘	┌	▌	▄	I	i	h	D
E	р	с	т	у	ф	х	ц	ч	ш	щ	ъ	ы	ь	э	ю	я	Ә	ғ	Қ	Ң	Ө	Ұ	Ү	һ	ә	ғ	қ	ң	ө	ұ	ү		F

ANSI стандартында қазақ әліпбиі әріптерінің кодтары 8-биттік кодтау кестесінің тек екінші бөлігінде, 8-ші жолдан бастап 15-ші ($15_{10} = F_{16}$) жолға дейін орналасады, өйткені бұл кестенің бірінші бөлігінде 8-ші жолға дейін ағылшын әліпбиі әріптерінің, араб цифрларының, басқарушы және басқа да міндетті таңбалар кодтары орналасқан. Шрифтерді 8-биттік кодтау кестесінде жасайтын әрбір жасаушы, осы стандарттарда белгіленген кодтауды ұстануы қажет.

Сондай-ақ, осы стандарт кодтан өзге кодтарды пайдаланып жасалынған ақпараттық қорларды сәйкестендіру үшін және қазақ мәтіндерін 8-биттік стандарт кодтарынан 16-биттік стандарт кодына немесе кері аударатын программаларда қолданылуы

мүмкін.

Қазақ әліпбиі әріптерінің 8-биттік кодтары Windows үшін II.2.2- кестеде көрсетілген.II.2.1.2- кесте. Қазақ әліпбиінің Windows үшін 8-битті кодтары

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
8	128	129	‚ 130	131	„ 132	. 133	† 134	135	136	‰ 137	138	< 139	140	Қ 141	Һ 142	143
9	144	‘ 145	’ 146	“ 147	” 148	• 149	– 150	– 151	152	™ 153	154	> 155	156	қ 157	һ 158	159
A	160	Ұ 161	ұ 162	Ә 163	¤ 164	Ө 165	166	§ 167	168	© 169	Ғ 170	171	172	173	® 174	Ү 175
B	176	± 177	І 178	і 179	ө 180	µ 181	¶ 182	183	184	№ 185	ғ 186	187	ә 188	Ң 189	ң 190	ү 191
C	А 192	Б 193	В 194	Г 195	Д 196	Е 197	Ж 198	З 199	И 200	Й 201	К 202	Л 203	М 204	Н 205	О 206	П 207
D	Р 208	С 209	Т 210	У 211	Ф 212	Х 213	Ц 214	Ч 215	Ш 216	Щ 217	Ъ 218	Ы 219	Ь 220	Э 221	Ю 222	Я 223
E	А 224	Б 225	В 226	Г 227	д 228	е 229	ж 230	З 231	И 232	й 233	К 234	Л 235	м 236	Н 237	О 238	п 239
F	Р 240	С 241	Т 242	У 243	ф 244	х 245	ц 246	Ч 247	Ш 248	щ 249	Ъ 250	Ы 251	ь 252	Э 253	Ю 254	я 255

Қазақ әліпбиі әріптерінің Unicode стандартындағы 16-биттік кодтық беті II.2.3 - кестеде көрсетілген.

II.2.1.3 -кесте. Unicode стандартындағы қазақ әліпбиінің 16-битті кодтары

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	A	**B**	**C**	**D**	**E**	**F**
040	**Ё**						**І**									
041	**А**	**Б**	**В**	**Г**	**Д**	**Е**	**Ж**	**З**	**И**	**Й**	**К**	**Л**	**М**	**Н**	**О**	**П**
042	**Р**	**С**	**Т**	**У**	**Ф**	**Х**	**Ц**	**Ч**	**Ш**	**Щ**	**Ъ**	**Ы**	**Ь**	**Э**	**Ю**	**Я**
043	**а**	**б**	**в**	**г**	**д**	**е**	**ж**	**з**	**и**	**й**	**к**	**л**	**м**	**н**	**о**	**п**
044	**р**	**с**	**т**	**у**	**ф**	**х**	**ц**	**ч**	**ш**	**щ**	**ъ**	**ы**	**ь**	**э**	**ю**	**я**
045		**ё**					**і**									
046																
047																
048																
049			**Ғ**	**ғ**							**Қ**	**қ**				
04A			**Ң**	**ң**											**Ү**	**ү**
04B	**Ұ**	**ұ**									**Һ**	**һ**				
04C																
04D									**Ә**	**ә**						
04E									**Ө**	**ө**						
04F																

II.2.1 Мысалдар:

Төменгі кестеде кейбір таңбалардың 8 битті кодтары берілген.

Таңбалар	**Кодтар**
4	1 1 1 1 0 1 0 0
A	1 1 0 0 0 0 0 0
B	1 1 0 0 0 0 0 1
+	0 1 0 0 1 1 1 0
=	0 1 1 1 1 1 1 0

Сонда осы кодтар арқылы A + B = 4 деген таңбалар тізбегі компьютердің жадында болып кескінделеді:

II.2.1. Тапсырмалар:

1. 11001010 11001110 11001010 тізбегінің мағынасын ашыңыз.
2. 2015 тізбегін кодтаңыз.
3. DEF деген тізбекті қодтаңыз.

Көмек:

1. Цифрларды 8 ден бөліп қарастыру қажет.
2. Әрбір цифрға сәйкес код жазыңыз.
3. Әрбір әріпке сәйкес код жазыңыз.

II.2.1. Сұрақтар:

1. Логикалық деректерді компьютерде қалай бейнелейді?
2. Символдық деректерді компьютерде қалай бейнелейді?
3. ANSI пен Unicode стандарттарының айырмашылығы неде?

I.2.1. Тесттер:

1. ANSI стандартында деректер қандай жүйеде кодталады?

A) екілік жүйеде;
B) үштік жүйеде;
C) сегіздік жүйеде;
D) ондық жүйеде;
E) он алтылық жүйеде.

2. Unicode стандартында деректер қандай жүйеде кодталады?

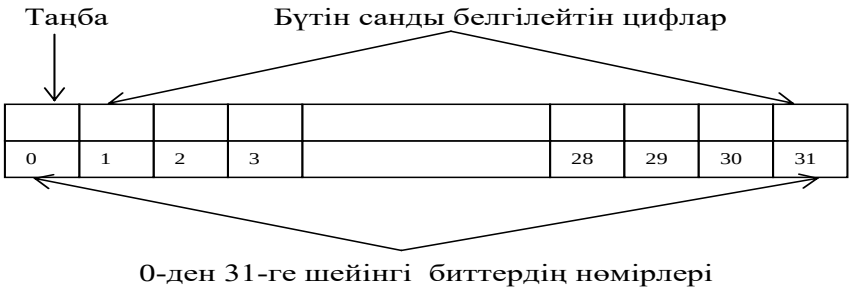
A) он алтылық жүйеде;
B) үштік жүйеде;
C) сегіздік жүйеде;
D) ондық жүйеде;
E) екілік жүйеде.

3. Unicode стандартында кодтау үшін қанша байт қолданады?

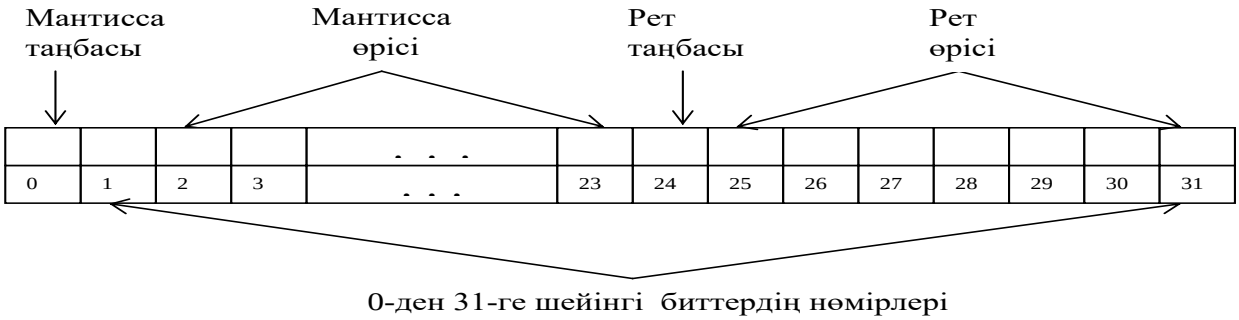
A) 2;
B) 1;
C) 3;
D) 5;
E) 4.

II.2.2. Компьютерде сандарды бейнелеу

Компьютерде тек бүтін сандар немесе нақты сандар өңделеді. Сондықтан оларды компьютер жадында бейнелеу әдісі болуы керек. Кез келген бүтін сан және нақты сандардың екілік санау жүйесінде бейнеленеді және оған бір *машиналық сөз* бөлінеді, соның ішінде санның таңбасы үшін тек бір бит арналады: теріс таңба 1, ал оң таңба 0 арқылы кескінделеді. Бір машиналық сөз бірнеше (1, 2, 4 немесе 8) байттан тұрады, олардың саны компьютердің буынына немесе моделіне байланысты. Компьютердің көптеген моделінде машиналық сөз 4 байттан, яғни 32 биттен тұрады. Бүтін сандардың компьютер жадында бейнелену үлгісі II.2.2.1-суретте көрсетілген.



II.2.2.1- сурет. Бүтін сандардың компьютер жадында бейнелену үлгісі



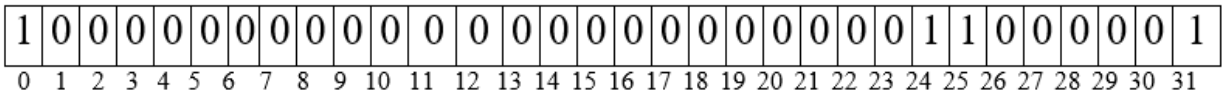
II.2.2.2- сурет. Нақты сандардың компьютер жадында бейнелену үлгісі

II.2.2. Мысалдар:

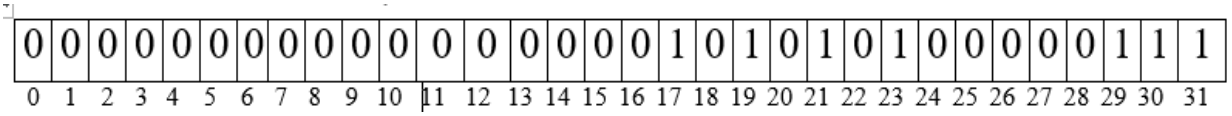
1. Оң +10101001 бүтін санының компьютер жадында бейнеленуі:

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	1	0	0	1
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

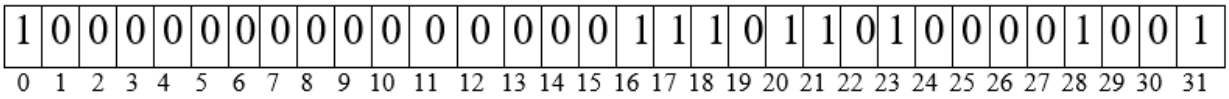
2. Теріс – 11000001 бүтін санының компьютер жадында бейнеленуі:



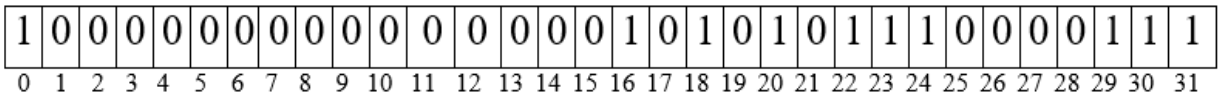
3. Оң жылжымалы нүктелі нақты $+0{,}1010101 * 2^3$ санының компьютер жадында бейнеленуі:



4. Теріс жылжымалы нүктелі нақты $-0{,}11101101 * 2^5$ санының компьютер жадында бейнеленуі:



5. 5. Теріс жылжымалы нүктелі нақты $-0{,}10101011 * 2^{-7}$ санының компьютер жадында бейнеленуі:



II.2.2. Тапсырмалар:

1. Мына оң бүтін 63 санын компьютер жадында бейнелеңіз.
2. Теріс жылжымалы нүктелі нақты $-0{,}11101101 * 2^5$ санын компьютер жадында бейнелеңіз.
3. Берілген ондық $X = -562_{10}$ санын екі байтты қосымша кодқа айналдырып, компьютер жадында бейнелеңіз.

Көмек:

1. Берілген 63 санын ондық санау жүйесінен екілік санау жүйесіне аудару керек те, бүтін сандардың компьютер жадында бейнелену үлгісін пайдалану қажет.
2. Жылжымалы нүктелі нақты сандардың компьютер жадында бейнелену үлгісін пайдалану керек те, оның мантиссамының және ретінің таңбаларын ескеру қажет.

3. Берілген *X* санының модулін екілік жүйеде бейнелеп, сол жағынан нөлмен 16 таңбаға шейін толтыру: 0000001000110010. Барлық таңбалардағы мәндерді түгел кері мәнге ауыстырып, шыққан кері кодқа 1 қосу керек

II.2.2. Сұрақтар:

1. Компьютерде бүтін сан қалай бейленеді?
2. Компьютерде жылжымалы нүктелі нақты сан қалай бейленеді?
3. Компьютерде бекітілген нүктелі нақты сан қалай бейленеді?

II.2.2. Тесттер:

1. Машиналық сөз қанша байттан тұрады?

A) 4 байттан;
B) 1 байттан;
C) 5 байттан;
D) 3 байттан;
E) 6 байттан.

2. Санның таңбасын кодтау үшін не бөлінеді?

A) 1 бит;
B) 2 бит;
C) 4 бит;
D) 6 бит;
E) 8 бит.

3. Ретті кодтау үшін қанша байт бөлінеді?

A) 1 байт;
B) 2 байт;
C) 3 байт;
D) 4 байт;
E) 5 байт.

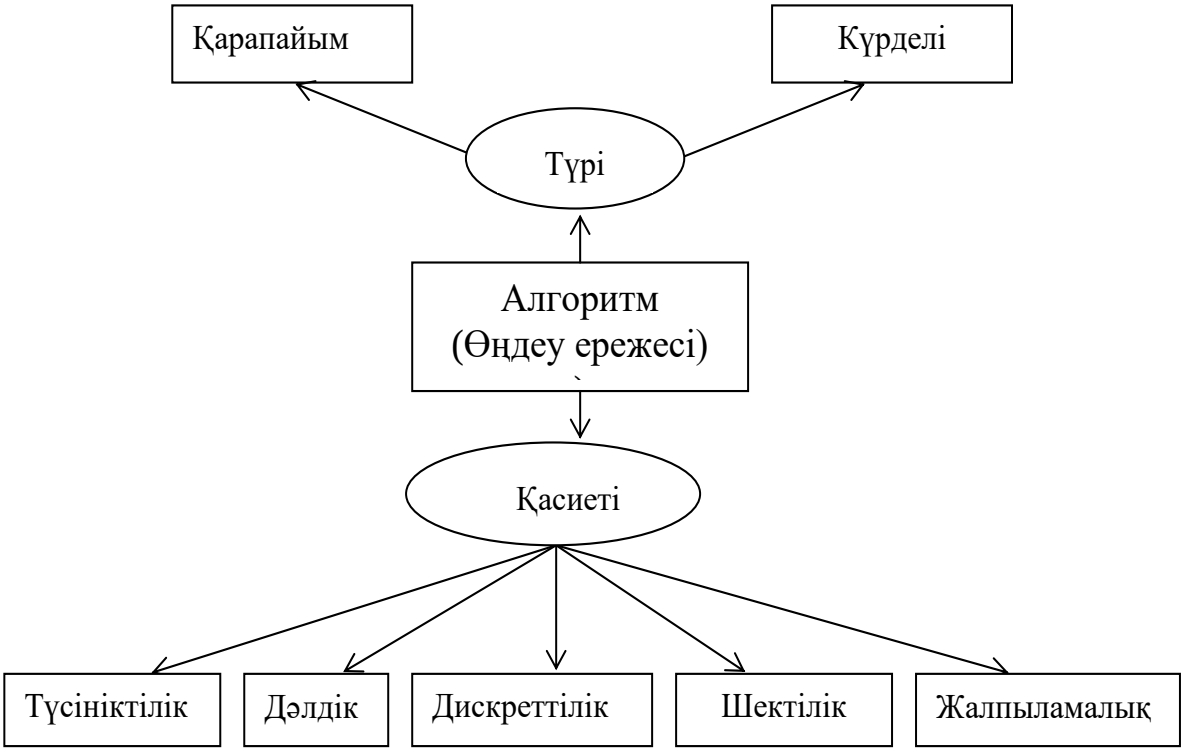
III. ИНФОРМАТИКАНЫҢ ЖҰМСАҚ ҚАМТЫМЫ

III.1. Алгоритмдер

Түйін сөздер: *алгоритм, өңдеу, нұсқау, амал, берілгендер, нәтиже, әрекет, күрделі әрекет, қарапайым әрекет, меншіктеу, құрастырушы, орындаушы, қатынас тілі, орындау хаттамасы, автоматтандыру, есептегіш техника, түсініктілік, дәлдік, дискреттілік, ақырлылық, жалпыламалық.*

Мақсат: *алгоритм ұғымының анықтамасын меңгеру, өңдеу ұғымын енгізу, алгоритмнің қасиеттерін қарастыру, нақты алгоритмдерді тұрғызу, орындау және тексеру.*

Құрылымы: Алгоритм ұғымының құрылымы III.1-суретте көрсетілген.



II.1- сурет. Өңдеу ұғымының құрылымы

III.1.1. Өңдеу ұғымы

Өңдеу информатика ғылымының негізгі ұғымдарының бірі болып есептелінеді.

«*Өңдеу*» деген сөздің бірнеше мағынасы бар. Бірақ бізге керегі оның «алдын ала анықталған мақсатқа жету үшін берілген шамаларды түрлендіру» деген мағынасы. Яғни, *өңдеу* деп алдын ала анықталып, бір ережеге жинастырылған нұсқаулар бойынша берілген шамаларға амалдар қолдануды айтады.

Ары қарай ыңғайлы болу үшін шамалар ұғымының орнына деректер ұғымын пайдаланамыз.

Деректерді өңдеу алдында *алғашқы деректер*, өңдеу кезінде *аралық деректер*, ал өңдеу біткеннен кейін *нәтиже деректер* деп атайды. Мысалы, егер берілген екі санды көбейтуді оларды қосқанның тізбегі арқылы берсек, онда берілген сандардың мәндері – *алғашқы деректер*, аралық қосындылар мәндері – *аралық деректер*, ал ақырлы қосындысы мәні нәтиже дерек болады.

Белгілі нұсқау бойынша берілген амалды орындауды *әрекет* деп атаймыз. Әрекеттер қарапайым және күрделі болуы мүмкін.

Информатикада ең қарапайым әрекеттердің бірі – *меншіктеу*. Оны жалпы түрде былай кескіндеуге болады:

<Атау> ← <Өрнек>,

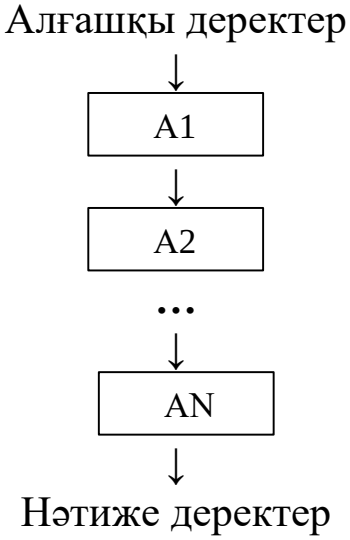
мұнда бұрыштық жақшалардың ішінде ұғымдар жазулы тұр, атау ретінде кез келген идентификатор алынады, « ← » – меншіктеу белгісі, өрнек ретінде сандық өрнек, символдық өрнек немесе логикалық өрнек алынады. Меншіктеу белгісі ретінде басқа таңбаны да алуға болады, мысалы « := ».

Меншіктеу әрекетінің семантикасы (*мағынасы*): алдымен өрнектің мәні есептелінеді, содан кейін ол мән атауға меншіктеледі, әрекет орындалып біткеннен кейін атаудың да, өрнектің де мәндері өзара тең болады.

Мысалы, *X←2+4*8* әрекетіндегі өрнектің мәні де атаудың мәні де осы әрекет орындалып біткеннен кейін *34*-ке тең болады.

Кез келген әрекетті белгілеп, әдетте үлкен латын әрпімен, оны графикалық түрде көрсетуге болады. Мысалы, егер ***A*** қарапайым әрекет болса, онда оны графикалық түрде III.1.1.1-суреттегідей етіп кескіндеуге болады.

Күрделі әрекет қарапайым әрекеттердің тізбегінен тұрады, яғни оның графикалық кескіні III.1.2-суреттегідей болады.



III.1.1.2 - сурет. Күрделі әрекеттің графикалық бейнеленуі

Мұнда A1, A2,., AN – күрделі әрекетті құрайтын қарапайым әрекеттер. Олардың орындалу реті жоғарыдан төмен қарай болады. Әрбір жоғарғы әрекеттің нәтижесі келесі төменгі әрекетке берілгендер қызметін атқарады.

Сонымен, өңдеуді анықтау үшін ереже құрастыру керек. Ол ережеде мынадай сұрақтардың жауаптары болады:

«Алғашқы деректердің типтері қандай?»

«Қандай әрекеттерді не үшін орындау керек?»

«Әрекеттердің орындалу реті қалай болады?»

«Нәтиже ретінде не есептелінеді?»

Жалпы, өңдеу ережесі туралы сөз қозғағанда, оған байланысты екі субьектіні естен шығармау керек. Олардың біреуі – ережені *құрастырушы*, ал екіншісі құрылған ережені *орындаушы* болады.

Орындаушының ережеде көрсетілген әрекеттерді орындай алатындай қабілеті болуы қажет. Ол үшін бір жағынан ережедегі нұсқауларды түсіне алуы, екінші жағынан берілген амалдардың орындалуын білуі керек. Сондықтан осындай ережелерді құрастырған кезде әрдайым орындаушының қабілетін ескерген жөн. Себебі, кейбір орындаушы берілген ережені орындай алады да, ал басқа орындаушыда ондай қабілет болмайды.

Ереже белгілі бір қатынас тілінде жазылады. Бұл қатынас тілінің барлық өрнектерінің мағыналарын құрастырушы мен орындаушы бірдей түсінуі керек. Яғни қатынас тілі бір мәнді болуы қажет: оның өрнектерін интерпретацияланған кезде бір ғана мағына шығуы керек.

III.1.1. Мысалдар:

1. «Бүтін санды бөлу» деген күрделі өңдеудің алғашқы деректері бүтін сандардың мәндері болады.

2. «Квадрат теңдеуді шешу» деген күрделі өңдеудің аралық нәтижесі, дерегі, дискриминант мәні болады.

3. «Орыс тілінен қазақ тіліне аудару» деген өңдеудің нәтижесі қазақ тіліндегі мәтін болады.

III.1.1. Тапсырма

1. «Квадрат теңдеудің түбірін табу» деген күрделі өңдеуді қарапайым өңдеулерге жіктеңіз.
2. «Шай демдеу» деген өңдеудің алғашқы деректері мен нәтижесін анықтаңыз.
3. «Жоғарғы білім алу» деген өңдеудің аралық нәтижелерін атаңыз.

Көмек:

1. Квадрат теңдеудің түбірін табу үшін оның коэфициенттерінің және дискриминантының мәндерін анықтау керек.
2. Шай демдеу үшін құрғақ шай, су, сүт, қант және т.б. керек.
3. Диплом алу үшін әрбір курсты сәтті аяқтау керек.

III.1.1. Сұрақтар

1. Информатикада «өңдеу» деген сөздің қандай мағынасы бар?
2. Өңдеудің алдына не беріледі?
3. Нәтиже деген не?

III.1.1. Тесттер:

1. Мына амалдардың қайсысы қарапайым?
   A) берілген екі санды өзара салыстыру;
   B) берілген сөздегі әріптер санын табу;
   C) берілген үш санның ішіндегі мәні үлкенін табу;
   D) берілген сөздегі әріптерді әліпби ретімен жазу;
   E) квадрат теңдеудің дискриминантын табу.

2. Ақпаратты өңдегенде ең соңында не шығады?
   A) алғашқы деректер;
   B) аралық деректер;
   C) қарапайым деректер;
   D) нәтиже деректер;
   E) құрылымды деректер.

1. Мыналардың қайсысы күрделі амал болады?
   A) A*B +C;
   B) X&Y;
   C) A-B;
   D) 3,14*R;
   E) 2,174/2.

III.1.2. Алгоритм ұғымы

Информатикада өңдеу ережесін *алгоритм* деп қарастырады. «*Алгоритм*» деген сөз IX ғасырда Орта Азияда туған ұлы ғалым *Әл–Хорезми* есімінің латынша «*Algorithm*» деп жазылуынан шыққан. Ол көп разрядты бүтін сандар үшін арифметикалық (*қосу, алу, көбейту, бөлу*) амалдарының орындалу ережелерін алғаш құрастырған. Мысалы, көп разрядты екі бүтін санды қосу үшін мынадай ережені орындау керек:

1) екі көп разрядты бүтін сандарының мәндерін анықтау;

2) осы екі санды бірінің астына бірін разрядтарын сәйкестендіріп жазу;

3) егер осы сандардың біреуінің үлкен разрядтары жетіспесе, оны нөлдермен толтыру;

4) ең кіші разрядта қосу амалын орындау және келесі разрядты қарастыруға көшу: егер келесі үлкен разрядқа өтетін бірлік пайда болса, онда оны жадтап қою керек;

5) барлық разрядтар біткенше оңнан солға жылжи қарастырып, бұрын есте сақталған бірлікті ескере және пайда болатын жаңа үлкен разрядқа өтетін бірлікті қайтадан жадтай отырып, егер олар бар болса, ағымдағы разрядта қосу амалын орындау;

6) нәтиже ретінде барлық разрядтарда қосу амалы орындалғаннан кейін, шыққан санды аламыз және есте сақталған бірлік болса, онда оны нәтиженің ең үлкен разрядының мәні ретінде есептейміз.

Бұл жазылған ережемен екі көп разрядты бүтін санды қосу үшін орындаушының қазақ тілін білуі (себебі, ереже қазақ тілінде жазылып тұр) және бір разрядты сандарды қосатын қабілеті болуы қажет.

Осы ережемен екі көп разрядты бүтін сандарды қосу хаттамасы III.1.2.1-кестеде көрсетілген.

III.1.2.1-кесте. Екі көп разрядты бүтін сандарды қосу хаттамасы.

Орындалатын нұсқаудың нөмірі	Орындалған нұсқаудың ізі	Түсініктеме
1	997 және 76	Екі санның мәндері анықталды
2	997 76	Бірінің астына бірі жазылды
3	997 076	Жетіспеген үлкен разряд нөлмен толықтырылды
4	997 + 076 13	Ең кіші разрядты қосу кезінде келесі үлкен разрядқа өтетін бірлік пайда болды
5	997 + 076 1 73	Келесі разрядты қосқан кезде тағы да үлкен разрядқа өтетін бірлік пайда болды
6	997 + 1 076 073	Соңғы үлкен разрядты қосқан кезде тағы да үлкен разрядқа өтетін бірлік пайда болды
7	107 3	Ең үлкен разрядқа өтетін бірлік есептеліп нәтиже алынды

Енді бізге керек *орындаушыны* ондық сан деген ұғымды білетін, екі көп разрядты бүтін санды көбейте алатын және қазақ тілін түсінетін қабілеті бар деп алып, кез келген екі ондық санды көбейтуге болатын ережені құрастырайық:

1) екі ондық санның мәндерін анықтау;

2) осы сандардағы ондық белгісін көрсететін үтірді ескермей, оларды бірінің астына бірін бүтін көбейткіштер ретінде жазу;

3) алдыңғы ережені қолданып көбейткіштерді көбейтіп, көбейтіндіні табу;

4) берілген екі ондық сандарындағы үтірдің оң жағындағы цифрлардың санын қосу;

5) нәтиже ретінде үтірден кейінгі цифрлар саны 4 қадамнан шыққан санға тең болатындай етіп жазылған 3 қадамнан шыққан көбейтіндіні аламыз. Мұнда көбейтіндідегі барлық цифрлардың саны 4 қадамнан шыққан саннан кем болса, онда үтірдің сол жағына бір нөл қойылады да, үтірдің оң жағына алдымен жетіспеген цифрлардың орнына нөлдер, соңынан көбейтіндінің мәні жазылады.

Кезінде осындай арифметикалық амалдарды орындау ережелерін алгоритм деп түсінсе, кейіннен «алгоритм» деген сөз түрлі математикалық есептерді шешу ережелерін белгілеуге қолданыла бастады. Мысалы, грек ғалымы Евклид құрастырған екі натурал санның ең үлкен ортақ бөлгішін табатын төмендегі ереже *Евклид алгоритмі* деп аталады:

1) екі натурал санын алып олардың мәндерін анықтау;

2) егер екі сан өзара тең болса, онда нәтиже ретінде осы сандардың кез келгенін алу керек, немесе осы сандардың үлкенін анықтау қажет;

3) үлкен санды үлкен сан мен кіші санның айырмасына алмастыру керек;

4-әрекетті 2 қадамнан бастап қайталау.

Бұл алгоритм, екі натурал сан M, N берілсе және $M > N$, онда M, N сандарының ең үлкен ортақ бөлгіші $M–N$, N сандарының ең үлкен ортақ бөлгішіне тең болады деген қасиетке негізделген.

Жоғарыда келтірілген мысалдардан алгоритмдегі көрсетілген нұсқауларды орындаушы өте қарапайым және бір-біріне ұқсас

болып келетін көптеген амалдарды орындайтынын байқауға болады. Осындай амалдарды орындауды алдын ала оқытылған басқа орындаушыға тапсыруға болатыны да айқын, демек *орындаушының әрекетін автоматтандыру* идеясы шығады.

Бұл идеяны жүзеге асыру үшін алдымен алгоритм ұғымына дәл анықтама беру қажет. Осы пікір математиканың «*Алгоритмдер теориясы*» деген жаңа саласын туғызды. Ол – алгоритм ұғымына дәл математикалық анықтама беруге және оған байланысты туатын теориялық проблемаларды зерттеуге бағытталған ғылым.

Алғашқы рет алгоритм ұғымының дәл математикалық анықтамасын 1937 жылы дискретті деректерді түрлендіре алатын абстрактылы есептеуіш машина құрастыру арқылы ағылшын математигі А.Тьюринг берді. Тура сол жылы Польшада туылған американдық логик Э.Л.Пост тағы бір абстрактылы машина ойлап тапты. Кейін олар «Тьюринг машинасы» және «Пост машинасы» деп сәйкес аталып кетті. Егер ол орындалатын амалдарды ескеріп алгоритмделсе, яғни абстрактылы машина тілінде жазылса, бұл абстракты машиналар кез келген проблеманы автоматтардың шығара алатындығын прінсіпиалды көрсетті.

«*Алгоритмдер теориясы*» саласындағы қол жеткен жетістіктер алгоритмдердің орындалуын автоматтандыруға мүмкіндік берген *компьютер* жасауға болатынын дәлелдеді.

Компьютердің пайда болуына негізінен алгоритм ұғымының кең тарауы шарт болды. Алгоритм деп тек математикалық есептерді ғана емес, басқа да есептерді шешетін ережелерді айта бастады. Олардың тобына *экономикалық есептер*, *басқару есептері*, *бір тілден екінші тілге аудару есептері* т.б.жатады.

Қазіргі кезде компьютер қолданылмайтын адамның зерделі қызмет саласын табу өте қиын. Бұл жағдай бүкіл адамзат алдына өте бір күрделі мәселе қойды: алгоритмдерді құрастыра білу және оларды компьютер (орындаушы) түсінетіндей тілде бейнелеу.

Дегенмен, алгоритмдерді құрастыруды үйрену үшін алгоритмдер теориясын білу шарт емес, оған алгоритмнің интуитивті анықтамасын, алгоритмдердің қасиеттерін және бейнелеу әдістерін білу жеткілікті.

Алгоритмнің интуитивті анықтамасын былай беруге болады:

Алгоритм деп ақырлы сан қадам жасау арқылы белгілі бір топтағы кез келген есепті шешуге арналған түсінікті, дәл және қай амалды қандай ретпен орындау керек туралы нұсқаулардың ақырлы тізбегін айтамыз.

Берілген анықтамадан *алгоритмнің қасиеттері* шығады:

1. *Жалпыламалық* алгоритмнің жеке есеп үшін ғана емес, осы сияқты есептердің бүкіл тобына қолданымдығын талап етеді.
2. *Түсініктілік* нұсқаулардың орындаушыға түсінікті тілде жазылуын талап етеді;
3. *Дәлдік* нұсқаулардың бір мәнділігін талап етеді;
4. *Дискреттілік* нұсқаулардың тізбек құратындығын және орындалуы жеке қадамнан тұратындығын талап етеді;
5. *Ақырлылық* нұсқаулардың орындалуы ақырлы сан қадам жасау арқылы жүзеге асатындығын және аяқталуынан нақты нәтиже шығатындығын талап етеді;

III.1.2. Мысалдар:

M және *N* натурал сандарының ең үлкен ортақ бөлгішін табу алгоритмін *ЕУОБ*(*M, N*) арқылы белгілеп, Евклид алгоритмін былай етіп жазуға болады:

1) *M, N* натурал сандарының мәндерін анықтау;
2) Егер $M=N$, онда *ЕУОБ*(*M, N*) $\leftarrow M$;
3) Егер $M>N$, онда $M \leftarrow M-N$ және 2 қадамнан қайталау;
4) $N \leftarrow N-M$ және 2-қадамнан бастап қайталау.

Осы жазылған алгоритмнің қасиеттерін көрейік.

1) Егер орындаушы қазақ тілін түсінсе және келтірілген амалдардың мағынасын білсе, онда бұл алгоритм оған *түсінікті*;

2) Алгоритмді түсінген әрбір орындаушы бірдей нәтиже алады, себебі оның барлық нұсқаулары бір мәнді;

3) Нұсқаулар бір-бірінен бөлініп тізбек құрып тұр және олардың орындалуы жеке қадам жасауды талап етеді;

4) Осы нұсқаулардың орындалуы шексіз болмайды. Себебі әр қайталау кезінде *M* санының мәні немесе *N* санының мәні кемиді. Соңында олар өзара тең болады да, нақты нәтиже шығады;

5) *M* және *N* сандарына әр түрлі мәндер бере отыра осы алгоритмнің кез келген натурал сандарға арналғандығы байқаймыз.

Мысал нөмірі	**Орындалатын нұсқаулар нөмірі**	**Нұсқаулардың орындалу барысы**
1	1	M←8, N←4
	3	8>4, M←8–4
	2	4=4, ЕУОБ (8,4)←4
2	1	M←7, N←3
	3	7>3, M←7–3
	3	4>3, M←4–3
	4	1<3, N←3–1
	4	1<2, N←2–1
	2	1=1, ЕУОБ (7,3) ←1

Жоғарғы кестедегіні алгоритмнің орындалу *хаттамасы* деп атайды, яғни *алгоритмді орындау кезіндегі* әрбір қадамның алғашқы деректері мен нәтижелерінің тізімі. Оны құру орындаушының ролін атқарғанмен бірдей.

III.1.2. Тапсырмалар:

1. A, B сандарының үлкенін табатын алгоритм жазыңыз.
2. A, B сандарының кішісін табатын алгоритм жазыңыз.
3. A, B сандарының арифметикалық ортасын табыңыз.

Көмек:

Есепті шығаруға керек амалдар тізбегі алгоритмді береді.

III.1.2. Сұрақтар:

1. Алгоритмнің интуитивті анықтамасы қандай?

2. Алгоритмнің қандай қасиеттері бар?

3. Алгоритмнің орындалу хаттамасы деген не?

III.1.2. Тесттер:

1. Алгоритмнің қандай қасиеттері бар?

A) Түсініктілік, дәлдік, дискреттілік, ақырлылық, жалпыламалық;

B) Түсініктілік, дәлдік, айқындылық, нұсқаулық, жалпыламалық;

C) Түсініктілік, дәлдік, дискреттілік, ақырлылық, жалпыламалық;

D) Түсініктілік, дәлдік, натурал, ақырлылық, жалпыламалық;

E) Шексіздік, дәлдік, дискреттілік, айқындылық, жалпыламалық;

2. Алгоритмнің жалпыламалық қасиеті нені анықтайды?

A) Алгоритмнің жеке бір есеп үшін емес, осы сияқты есептердің бүкіл класына арналғандығын көрсетеді;

B) Орындаушыға түсінікті тілде жазылуын талап етеді;

C) Нұсқаулардың бір мәнділігін талап етеді;

D) Нұсқаулардың тізбек құратындығын көрсетеді;

E) Нұсқаулардың орындалуынан кейін нәтиже шығатындығын көрсетеді.

3. Алгоритмнің ақырлылық қасиеті нені анықтайды?

A) Алгоритмнің жеке бір есеп үшін екендігін көрсетеді;

B) Нұсқаулардың түсініктілігін талап етеді;

C) Нұсқаулардың бір мәнділігін талап етеді;

D) Нұсқаулардың тізбек құратындығын көрсетеді;

E) Нұсқаулардың орындалуы ақырлы сан қадам жасаумен бітетіндігін және осыдан кейін нәтиже шығатындығын көрсетеді.

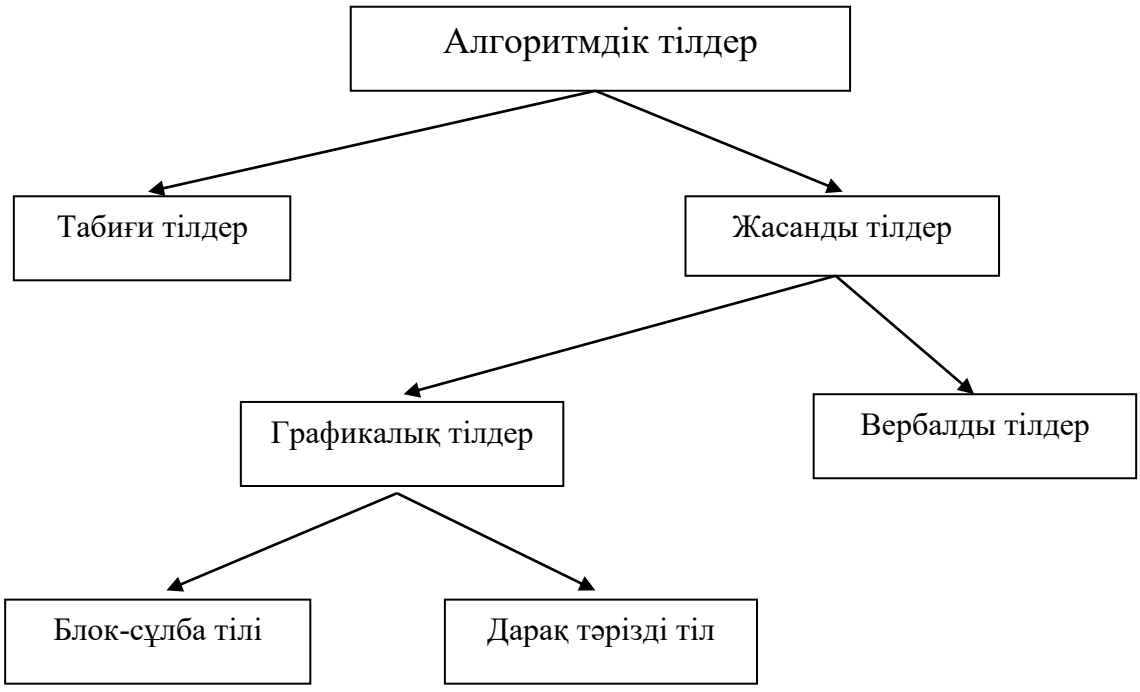
III.2. Алгоритмдік тілдер

Бұл тарауда алгоритмдік тілдердің түрлері және олардың мүмкіншіліктері қарастырылады.

Түйін сөздер: *алгоритмдік тіл*, *табиғи тіл*, *жасанды тіл*, *графикалық тіл*, *блок-сұлба*, *дарақ тәрізді графика*, *әліпби*, *басқару құрылымдары*, *тізбектеу*, *тармақталу*, *қайталау.*

Мақсат: *алгоритмнің бейнелеу әдістерімен танысу*, *графикалық тілдерді талдау*, *жасанды алгоритмдік тілді анықтау*, *жасанды тілде алгоритмдерді құрастыруды үйрену.*

Құрылымы: Алгоритмдік тілдердің құрылымы III.2-суретте көрсетілген.



III.2-сурет. Алгоритмдік тілдердің құрылымы

III.2.1. Вербалды алгоритмдік тіл

Өткен тарауда алгоритмді бейнелеудің бір әдісімен, яғни адамдар қатынас жасайтын, табиғи тілде бейнелеумен таныс болдық. Табиғи тілге, мысалы, қазақ тілі, орыс тілі, ағылшын тілі және т.б. тілдер жатады. Алгоритмдегі нұсқаулар мен амалдарды анықтай түсү үшін кейде осы тілдерге математикалық өрнектер қосылады.

Табиғи тілдерде бірмәнділік болмағандықтан, осы тілдерде жазылған алгоритмдегі нұсқаулардың бірнеше мағынасы болуы мүмкін. Әрине, бұл орындаушының осындай тілде жазылған алгоритмді орындауын қиындатады, тіпті ондай алгоритмді орындау мүмкін болмайды. Сондықтан табиғи тілдердің байлығына қарамастан, информатикада алгоритмдерді бейнелеу үшін бір мәнді жасанды тілдерді қолданады. Оларға *жасанды вербалды* (сөзді) *тілдер* және *графикалық тілдер* жатады.

Енді қазақ тілі мен математикалық өрнектерге негізделген жасанды вербалды алгоритмдік тілді анықтайық. Бұл тілде қазақ тілінің бірнеше «бекітілген» сөздері қолданылады. Олардың әрқайсысының алдын ала анықталған тек бір ғана мағынасы болады. Сондықтан оларды осы тілдің әліпбиіне кіргізуге болады. Сонымен қатар, бұл тілдің әліпбиіне латын және қазақ әліпбиінің таңбалары, араб және рим цифрлары, математикалық таңбалар, тыныс белгілері, жақшалар, меншіктеу амалының таңбасы, бос жерді белгілейтін таңба қосылады. Осы тілдің әліпбиі төмендегі III.2.1-кестеде беріледі.

III.2.1 -кесте. Жасанды вербалды алгоритмдік тілдің әліпбиі

Таңбалардың түрі	**Әліпбидегі бейнесі**
Әріптер	
Латын әріптері	A, B, C,., X, Y, Z, a, в, c,., x, y, z
Қазақ әріптері	А, Ә, Б,., І, Э, Я, а, ә, б,., і, э, я
Цифрлар	
Араб цифрлары	0, 1, 2, 3, 4, 5, 6, 7, 8, 9
Рим цифрлары	I, V, X

Арнаулы таңбалар	
Жақшалар	
Құрылымдық жақша	(,)
Символдық жақша	‘, ‘
Өлшемдік жақша	[,]
Амалдар таңбалары	
Сандық амалдар	+, –, *, /, ↑, %
Символдық амалдар	·, |
Қатынас амалдары	=, <, >, ≤, ≥
Логикалық амалдар	¬, ∧, ∨
Меншіктеу амалы	←
Тыныс белгілері	.,,, :, ;, !
Кетік таңбасы	␣
Бекітілген сөздер	БАСЫ, СОҢЫ, ЕНГІЗУ, ШЫҒАРУ, ЕГЕР, ОНДА, ӘЙТПЕСЕ, ТАҢДАУ, БОЛСА, ОРЫНДА, ҚАЙТАЛАУ, ӘЗІРГЕ, ШЕЙІН, ҮШІН, ҚАDAM, АЛГОРИТМ, БІТТІ

III.2.1. Мысалдар:

1. Қызметкерлердің мамандығы мен істеген жұмыс сағатына байланысты күндік жалақыны есептеу керек. Олардың үш түрлі мамандығы бар, оларды М1, М2, М3 деп, ал жұмыс уақытын Т1, Т2, Т3 деп белгілейміз. Осыларға сәйкес қызметкерлердің әр жұмыс сағаты 100, 200, 300 теңге тұрады. Егер қызметкер 8 сағаттан артық жұмыс істесе, онда оның жалақысына тағы да 10% үстеме қосылады. Осы есептің шешуі төмендегі алгоритмде жазылады.

```
АЛГОРИТМ   жалақы
  ЕНГІЗУ (М1, М2, М3, Т1, Т2, Т3)
     БАСЫ
          ЕГЕР Т1 > 8 ОНДА   М1 ←100 * Т1 + 100 * Т1 * 10 %
                ӘЙТПЕСЕ   М1 ← 100 * Т1
          ЕГЕР Т2 > 8 ОНДА   М2 ←200 * Т2 + 200 * Т2 * 10 %
```

```
                ӘЙТПЕСЕ   М2 ← 200 * Т2
          ЕГЕР Т3 ≤ 8  ОНДА  М3 ← 300 * Т3
                ӘЙТПЕСЕ  М3  ← 300 * Т + 300 * Т3 * 10 %
     СОҢЫ
   ШЫҒАРУ  (М1, М2, М3)
БІТТІ
```

2. Мына түрдегі A*X↑2 + B*X + C = 0 квадрат теңдеулерді шешетін алгоритмді құру керек. Ол үшін алдымен осы теңдеудің дискриминантын табу керек, содан кейін оның мәніне байланысты квадрат теңдеудің шешімі табылады. Сондықтан осындай квадрат теңдеулерді шешу үшін мына алгоритм ұсынылады:

```
АЛГОРИТМ   квадрат теңдеу
ЕНГІЗУ (A, B, C)
БАСЫ
  D = B↑B – 4A*C
  ЕГЕР D>0 ОНДА ШЫҒАРУ(
       ‘Теңдеудің 1-нші шешуі  X1= ’ (–B+Sqrt(D)) /2*A),
       ‘Теңдеудің 2-нші шешуі  X2= ’ (–B–Sqrt(D)) /2*A)
  ЕГЕР D=0  ОНДА ШЫҒАРУ ‘Теңдеудің шешуі X = ’ –B/2*A
  ЕГЕР D<0  ОНДА ШЫҒАРУ ‘Теңдеудің шешуі жоқ’
СОҢЫ
БІТТІ
```

III.2.1. Тапсырмалар:

1. Екі оң бүтін санның көбейтіндісін, оларды тізбектеп қосу арқылы табудың алгоритмін құрыңыз.
2. Бір оң бүтін санды екінші бүтін оң санға дәрежеге шығаруды, тізбектеп көбейту арқылы табудың алгоритмін құрыңыз.
3. Берілген санның квадратының жартысын табатын алгоритмді құрыңыз.

Көмек:

1.Күрделі амалдарды қарапайым амалдар арқылы бейнелеуді білу керек.

2. Күрделі амалдарды қарапайым амалдар арқылы бейнелеуді білу керек.

3.Квадратқа шығарып, 2-ге бөлу керек.

III.2.1. Сұрақтар:

1. Алгоритмдік тілдердің қандай түрлері бар?
2. Жасанды вербалды алгоритмдік тілдің әліпбиі деген не?
3. Жасанды вербалды алгоритмдік тілде қандай сөздер бар?

III.2.1. Тесттер:

1. Алгоритм термині қандай ұлы математиктің атымен аталған?

A) Джон фон Нейман;
B) Чарльз Беббидж;
C) Мухаммед аль-Хорезми;
D) Вильгельм Лейбниц;
E) Ада Лавлейс.

2. Алгоритмнің денесінде жазылған әрбір пәрмен не деп аталады?

A) Шарт;
B) Пәрмен жүйесі;
C) Алгоритмдік үдеріс;
D) Жай пәрмен;
E) Қадам.

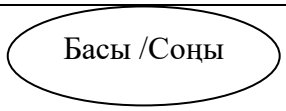
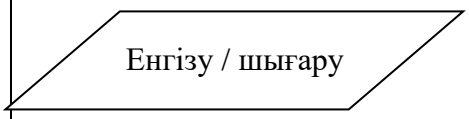
3. Алгоритмнің анықтамасына мына айтылымдардың қайсысы жақын?

A) Іздеп отырған нәтижені алу мақсатында атқарылатын әрекеттердің орындалу ретін анықтайтын нақты ережелер;
B) Ескертудегі әрекеттердің берілген орындалу тәртібі;
C) Қандайда бір қадам бойынша нәтиже алу;
D) Әртүрлі мәндегі біртектес есептерді таңдау;
E) Дұрыс жауап алу ережесі.

III.2.2. Графикалық алгоритмдік тіл

Алгоритмді бейнелеудің графикалық әдісін графикалық алгоритмдік тіл деп атайды, оның екі түрі бар: *блок-сұлба тілі*, *дарақ тәрізді тіл*. *Блок-сұлба тілі* алгоритмдегі әр түрлі әрекеттердің хронологиялық байланысын көрсете алады. Ол үшін бұл тілде түрлі әрекеттерді бейнелейтін геометриялық фигуралар және әрекеттердің орындалу ретін көрсететін бағыттама пайдаланылады. Бұлар блок-сұлба тілінің әліпбиін құрайды, ал олардың мағынасы алдын ала анықталған келісім бойынша беріледі. Блок-сұлба әліпбиі III.2.2-кестеде көрсетілген.

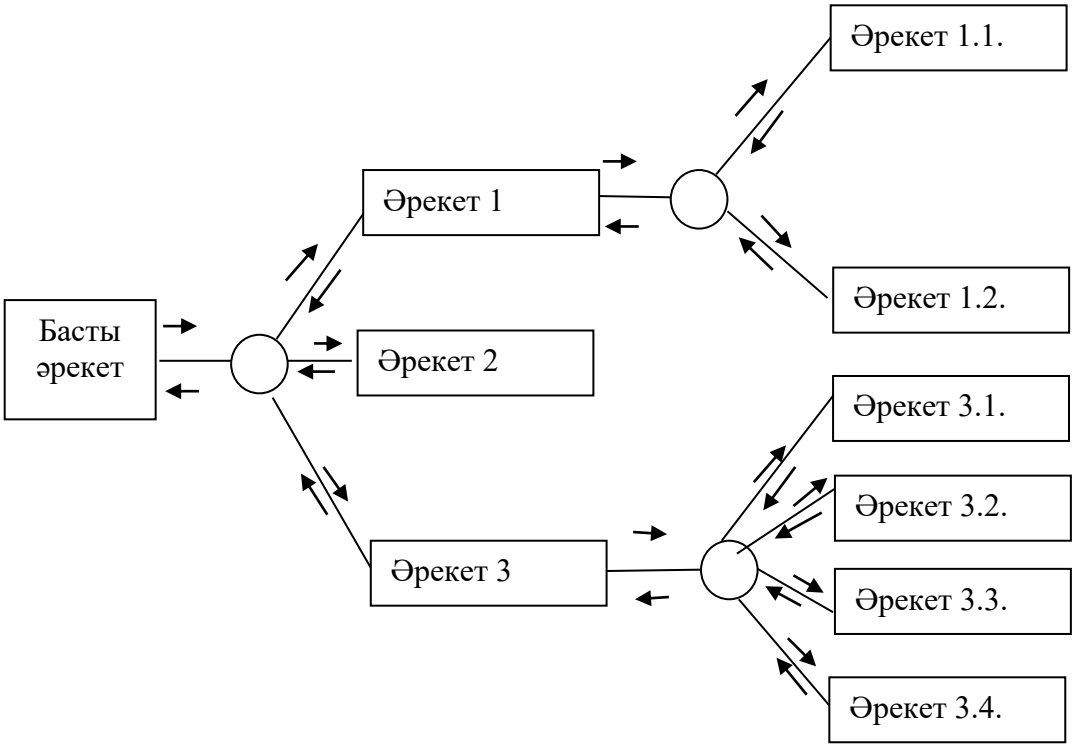
III.2.2-кесте. Блок-сұлба тілінің әліпбиі

Аталуы	**Бейнеленуі**	**Тағайындалуы**
Жұмыр		Алгоритмнің басын немесе соңын бейнелеу
Параллелограмм		Деректерді енгізу немесе шығару амалын бейнелеу
Тік төртбұрыш		Әрекетті бейнелеу және оның нөмірін көрсету
Ромб		Шартты тармақталуды бейнелеу
Шеңбер		Бірнеше жолдарды біріктіруді бейнелеу
Бағыттамалар		Жолдардың бағытын бейнелеу

Блок-сұлба тілінің элементін қарастырған кезде пайдаланылған «деректерді енгізу» және «нәтижені шығару» деген әрекеттердің мағыналарын біз есептегіш техникасының құрылымын өткен кезде анықтаймыз. Әзірше олардың орнына

«алғашқы деректердің мәндерін анықтау» және «шыққан нәтиженің мәнін дайындау» деп атап алып меншіктеу амалын пайдаланамыз.

Блок-сұлба тілі қарапайым алгоритмдерді бейнелеуге арналған графикалық әдіс. Ол жаңадан үйреніп жүрген құрастырушыға түсінікті және ыңғайлы болғанымен, күрделі алгоритмдерді бейнелеуге жарамсыз. Себебі күрделі алгоритм бөліктерінің блок-сұлбасы бөлек бірнеше бетке салынады да, олардың арасындағы байланысын және алгоритмнің жалпы логикасын байқау өте қиынға түседі. Сондықтан алгоритмдерді бейнелеудің тағы да бір графикалық әдісі – *дарақ тәрізді тілмен* танысамыз. Оның жалпы түрі III.2.2-суретте берілген.



III.2.2-сурет. Дарақ тәрізді тілдегі алгоритмдердің жалпы түрі

Алгоритмнің III.2.2-суретте бейнеленуі дарақка ұқсайды. Сондықтан оны дарақ тәрізді тіл деп атайды. Сол жақтағы ең шеткі әрекетті дарақтың тамыры, ал оң жақтағы ең шеткі әрекеттерді дарақтың жапырақтары дейді. Алдымен тамыр ретінде бас әрекет бейнеленеді, кейіннен ол өзін құрайтын бірнеше әрекеттерге жіктеледі, осы әрекеттер өздерін құрайтын кішірек әрекеттерге тағы да жіктелуі мүмкін, ал ең соңында ары қарай жіктелмейтін қарапайым әрекеттер бейнеленеді. Әрекеттерді бейнелеу үшін тік

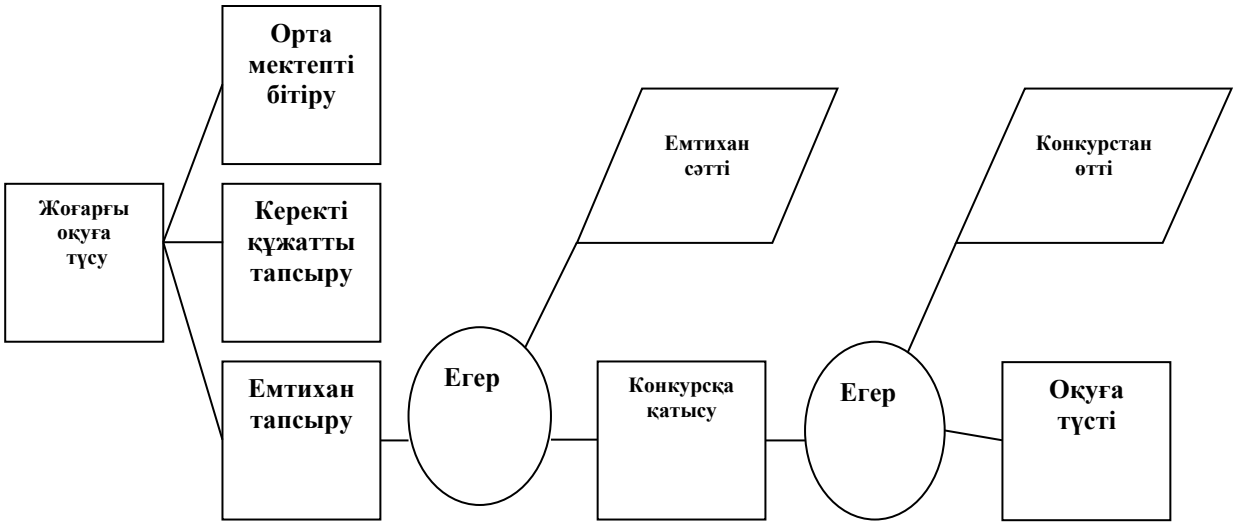
төртбұрыш қолданылады, олардың құрамдас әрекеттері шеңбермен басталады, алгоритмдегі әрекеттердің хронологиялық реті цифрлармен көрсетіледі, ал дарақты аралау жолдарын бағыттамалар көрсетеді. Дарақты аралау жолы алгоритмдегі әрекеттердің жіктелу реті мен олардың орындалу ретін көрсетеді. Әдетте, *жіктелу реті солдан оңға* қарай, ал *орындалу реті жоғарыдан төмен* қарай болады. Сондықтан алгоритмді осы әдіспен кескіндеген кезде бағыттамаларды көрсетпесе де болады.

Дарақ тәріздес тіл нұсқаулардың хронологиялық байланысымен қатар, олардың логикалық жіктелуін, декомпозициясын көрсете отырып, алгоритмнің иерархиялық құрылымын бейнелеуге мүмкіндік береді.

Бұл әдіспен шартты тармақталуды немесе қайталауды (циклді) да бейнелеуге болады. Ол үшін «Егер», «Әзірше» немесе «Шейін» деген кілт сөзді шеңбердің ішіне, ал тармақталу немесе қайталау шартын ромбының ішіне жазады.

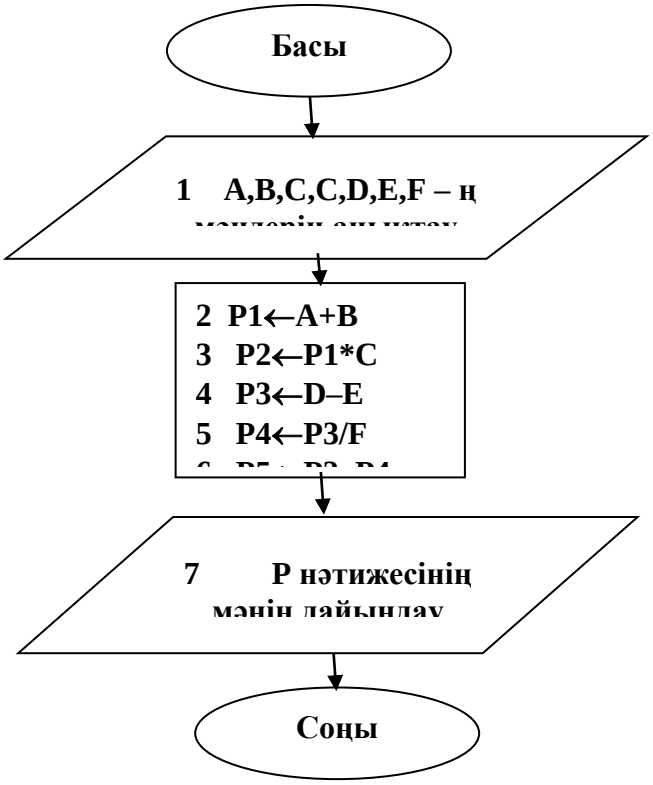
III.2.2. Мысалдар:

1. Талапкердің жоғарғы оқуға түсу үшін бірнеше әрекет жасауы керек және олардың кейбіреулері белгілі шарт орындалғанда ғана іске асады. Осылардың ішіндегі ең маңызды әрекеттердің тізбегі дарақ тәрізді тілде бейленген түрі төменгі суретте көрсетілген.



2. Деректердің атаулары A, B, C, D, E және F-ке әр түрлі мәндер бере отырып, мына өрнектің P ← (A+B)* C – (D – E)/ F мәнін

есептеу алгоритмінің блок-сұлба тілінде бейнелеуі төменде көрсетілген.



Мұндағы **P1, P2, P3, P4, P5** – қосалқы нәтижелердің атаулары. Осы алгоритмнің орындалу хаттамасы төмендегі кестеде көрсетілген.

Мысал нөмірі	**Нұсқаулар нөмірі**	**Нұсқаулардың орындалу барысы**	
I	1	A←2, B ←4, C ←2, D ←10, E ←2, F ←2	
	2	P1 ← 2 + 4	
	3	P2 ← 6 * 2	
	4	P3 ← 10 – 2	
	5	P4 ← 8/2	
	6	P5 ← 12 – 4	
	7	P ← 8	
II	1	A ←3, B ←5, C ←4, D ←15, E ←15, F ←2	
	2	P1 ← 3+5	
	3	P2 ← 8 * 4	
	4	P3 ← 15 – 5	
	5	P4 ← 10/2	

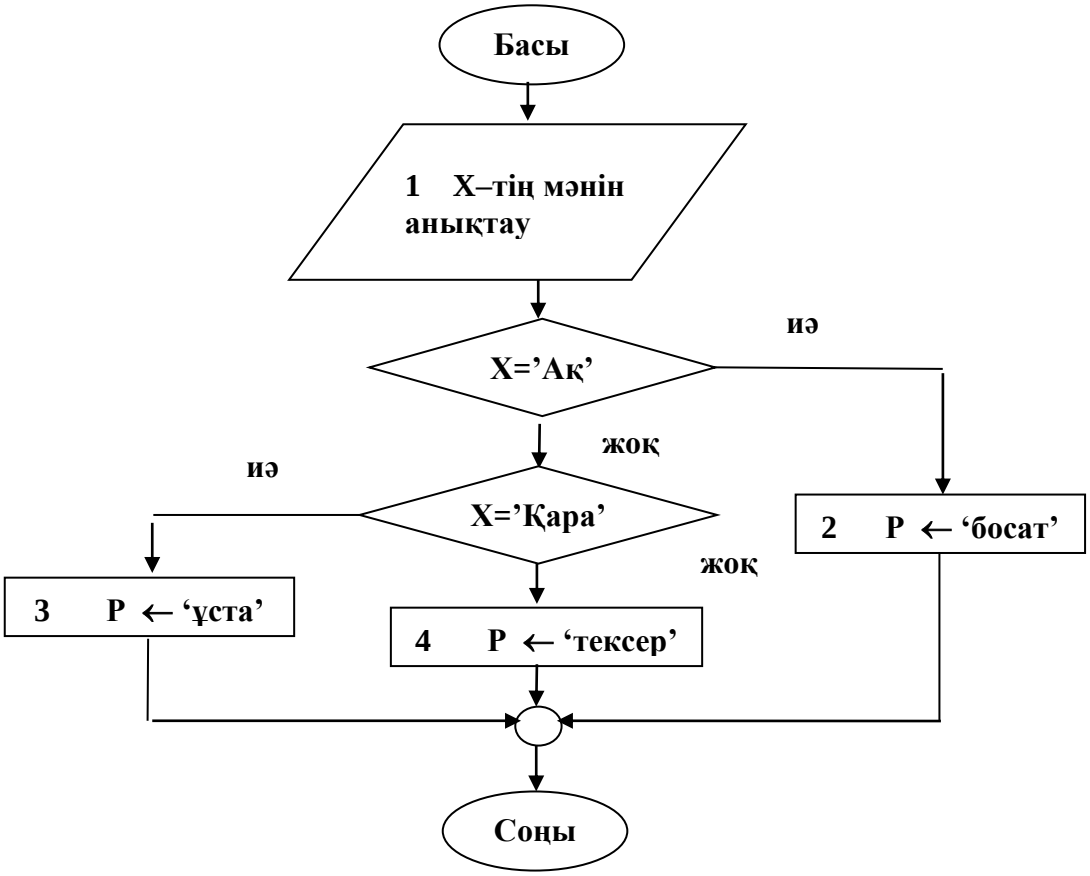
	6 7	P5 ← 32 – 5 P ← 27	

Осы кестеде блок-сұлбадағы енгізу және шығару амалдарының орнына 1-ші және 7-ші нұсқауларда меншіктеу амалдары қолданылып отыр. Себебі, енгізу және шығару амалдары меншіктеу амалының бір түріне жатады.

3. Берілген символдық шама **X** бірнеше мәнді болсын. Оның біреуі – ‘Ақ’, екіншісі – ‘Қара’, ал қалғаны белгісіз. Осы **X**-тің мәндеріне байланысты әр түрлі әрекеттер жасау керек. Оның біреуі – символдық **P** шамасына ‘босат’ деген мән беру, егер **X**=’Ақ’, екіншісі – символдық **P** шамасына ‘ұста’ деген мән беру, егер

4. **X** = ‘Қара’, ал үшіншісі – символдық **P** шамасына ‘тексер’ деген мән беру, егер **X**–тің мәні кез келген болса.

Блок-сұлбадан шартты тармақталуды белгілейтін ромбыдан шығатын бағыттаманың тек екеу болатынын және олардың жоғарыдан басқа, ромбының кез келген жағынан шығатынын байқауға болады. Жалпы айтқанда, блок-сұлбаны құрастырушы өзінің ыңғайына қарай жазықтықтың кез келген жағына жазуына болады. Осы айтылғандар үшін төмендегі блок-сұлбаны саламыз.

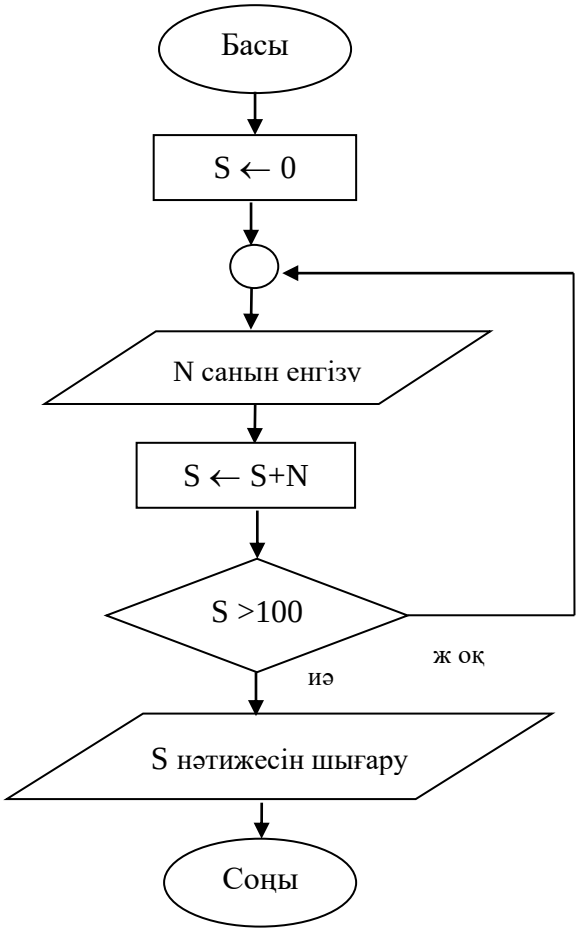


Осы алгоритмнің орындалу хаттамасы төменде көрсетілген.

Мысал нөмірі	**Нұсқаулар нөмірі**	**Нұсқаулардың орындалу барысы**
I	1	X ← ‘Ақ’
	2	P ← ‘босат’
II	1	X ← ‘Қара’
	3	P ← ‘ұста’
	1	X ← ‘Білмеймін’
III	4	P ← ‘тексер’

III.2.2. Тапсырмалар:

1. Мына X←(A + B) ∗ (C – D) өрнектегі амалдардың орындалу ретін дарақ тәрізді тілінде жазыңыз.
2. Компьютердің жұмыс ырғағының алгоритмін блок-сұлба тілінде құрыңыз**.**
3. Төмендегі алгоритмді вербалды тілде қайталап жазыңыз.



Көмек:

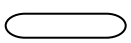
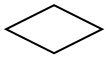
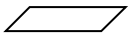
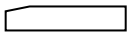
1. Алгоритмді II.1.2-те берілген классикалық құрылымдағы компьютердің жұмыс ырғағы туралы ережелерге сәйкес құру керек.
2. Дарақ тәрізді тілде құрастыратын алгоритмде ешқандай шарт бойынша тармақталудың қажеті жоқ.
3. Бұл тілде қазақ тілінің бір мағыналы бірнеше алдын ала бекітілген сөздері қолданылады.

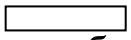
III.2.2. Сұрақтар:

1. Графикалық тілдің қанша түрі бар?
2. Блок-сұлба тілінің әліпбиіне не кіреді?
3. Дарақты аралау жолы нені көрсетеді?

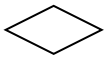
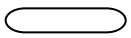
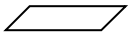
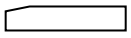
III.2.2. Тесттер:

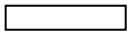
1. Блок-сұлбадағы қандай фигура алгоритмнің басын және соңын білдіреді?

A)  ;
B)  ;
C) ;
D)  ;
E)  .



2. Блок-сұлбадағы қандай фигура шартты білдіреді?

A)  ;
B)  ;
C) ;
D)  ;
E)  .



3. Блок-сұлбадағы қандай фигура енгізу немесе шығару амалын білдіреді?

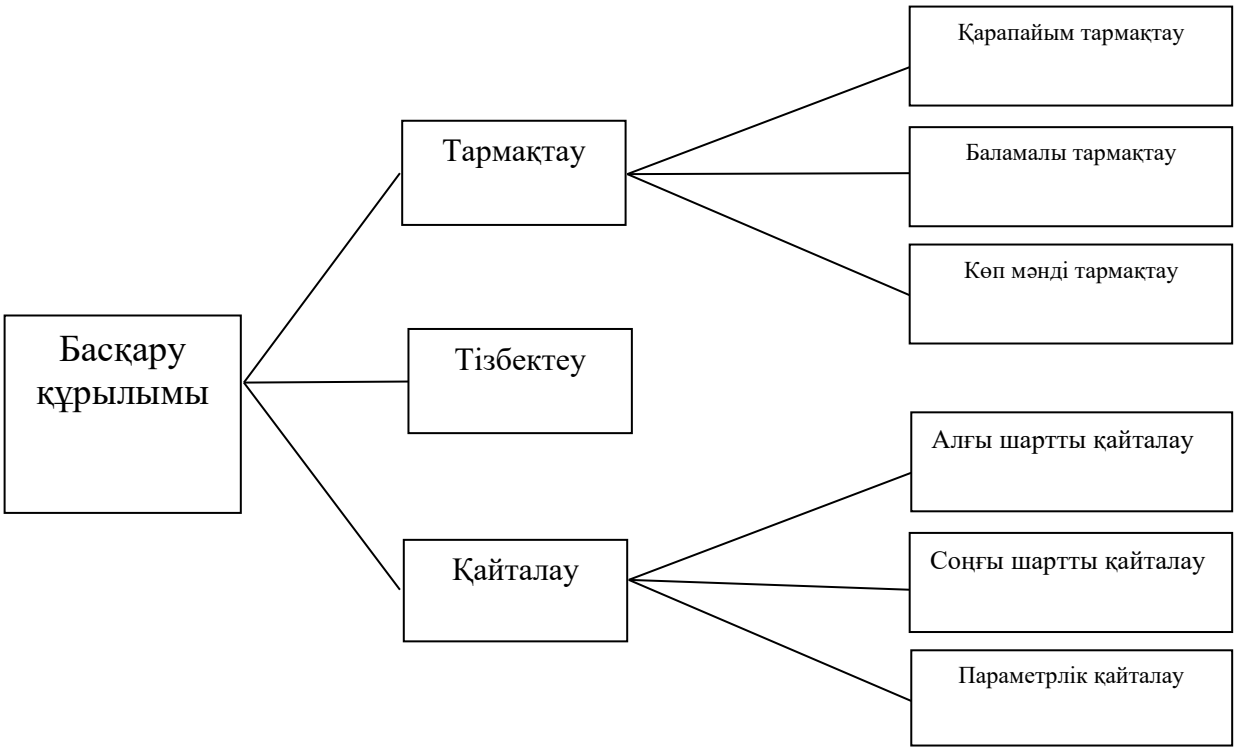
A) Параллелограмм;
B) Ромб;
C) Шеңбер;
D) Тіктөртбұрыш;
E) Жұмыр.

III.3. Басқару құрылымдары

Түйін сөздер: *тізбектеу*, *тармақталу*, *қарапайым тармақталу*, *баламалы тармақталу*, *көп мәнді тармақталу*, *қайталау*, *алғышартты қайталау*, *соңғышартты қайталау*, *параметрлі қайталау.*

Мақсат: *басқару құрылымдарымен танысу және олардың қасиеттерін ажырата білу, жасанды тілде басқару құрылымдары арқылы алгоритмдерді құруды үйрету.*

Құрылымы: «Басқару құрылымдары» атты оқу материалының құрылымы III.3-суретте көрсетілген.



III.3-сурет. Басқару құрылымдарының құрылымы.

III.3.1. Басқару құрылымдары және олардың мәні

Алгоритмдерді жазудың әртүрлі тәсілдері қарастырылғаннан кейін оларды құрудың әдіснамасы туралы сұрақтың тұруы заңды, яғни оларды құрудың технологиясы керек болды. Осыған байланысты 1970 жылдары IBM компаниясында белгілі ғалымдар Э. Дейкстра, Х. Милс, Э. Кнут, С. Хоор ұсынған *құрылымды программалау* идеясы туды.

Алгоритм ақпаратты өңдеудің ретін анықтағандықтан, ол бір жағынан, *өңдеу әрекеттерін*, ал екінші жағынан, *басқару ағымы* деп аталатын олардың орналасу ретін қамтуы керек. Басқару ағымы мынадай қасиеттерге ие болуы мүмкін:

(1) Әрбір блок бірден артық орындалмауы керек;

(2) Әрбір блок орындалады.

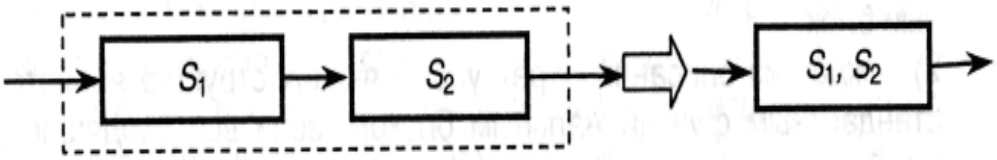
Мұнда деректерді өңдеуге қатысты блоктар жай және шартты болып екіге бөлінеді. Қарапайым әрекеттің, белгілі шарт ақиқат бола алатындығына байланысты екі шығысы бар тармақталудан ерекшелігі, оның бір кірісі және бір шығысы бар болады.

Қарапайым әрекет деген ол жалғыз дегенді білдірмейді, ол әрекеттердің кейбір тізбегі бола алады. Алгоритмнің қарапайым әрекет ретінде ұйымдастырылған бөлігі, яғни бір кірісі (орындалуы әрдайым бір ғана әрекеттен басталады) және бір шығысы (осы блоктың орындалуы аяқталғаннан кейін әрдайым бір ғана әрекеттің орындалуы басталса) *функционалдық блок* деп аталады.

Құрылымдық программалау шарты бойынша алгоритм әрекетін басқару ағымын ұйымдастырудың әртүрлі үш үлгісі болады. Оларды түсіндіру үшін екі S_1 және S_2 функционалдық блогын аламыз.

Басқару ағымының бірінші түрі *тізбектеу* деп аталады, ол екі S_1 және S_2 функционалдық блоғын орындауды ұйымдастырады және егер басқару ағымының екі қасиеті де орындалса, онда басқары ағымы *сызықтық* болады. Сызықтық басқару ағымында

бірнеше блок бір функционалдық блокка біріге алады. Тізбектеліп орындалатын S_1 және S_2 блоктарының бір S_1S_2 функционалдық блокка бірігуінің графикалық бейнесі III.3.1-суретте көрсетілген.



III.3.1-сурет. Екі блоктың бір блокка бірігуі

Басқару ағымының екінші түрі *тармақталу* деп аталады, ол тексерілетін логикалық шарттың мәніне байланысты екі S_1 және S_2 функционалдық блоктың бірін орындауды ұйымдастырады. Мұнда (1) қасиеті орындалады, ал (2) орындалмайды. Егер басқару ағымы S_1 және S_2 функционалдық блоктардың екеуін де қамтыса, онда тармақталу *баламалы* деп аталады. Егер басқару ағымы S_1 және S_2 функционалдық блоктардың біреуін қамтымаса, онда тармарталу *қарапайым* деп аталады. Жалпы тармақталу үш түрге бөлінеді: *қарапайым тармақталу*, *баламалы тармақталу*, *көп мәнді тармақталу*.

Басқару ағымының үшінші типі *қайталау* деп аталады, ол *қайталану денесі* деген функционалдық блоктың бірнеше рет қайталанып орындалуын ұйымдастырады. Қайталану саны алдын ала беріледі немесе қандай да бір шарттың орындалуына немесе орындалмауына байланысты анықталады. Қайталану ағымы үшін (2) қасиет орындалады, ал (1) қасиет орындалмайды. Белгілі шартқа байланысты қайталаудың екі түрі бар: *алғышартты қайталау*, *соңғышартты қайталау*. Алғышартты қайталауда шарт қайталау денесі орындалуының алдында тексеріледі, ал соңғышартты қайталауда шарт қайталау денесі орындалуынан кейін тексеріледі. Егер қайталау саны алдын ала белгілі болса, онда оны *параметрлі қайталау* деп атайды, ол соңғышартты қайталау тобына жатады.

Тармақталуда және қайталауда бір кіріс және бір шығыс болғандықтан, олар бүтіндей функционалдық блоктың анықтамасына келеді. Осыны пайдаланып *стандартты функционалдық блок* ұғымының рекурсивті анықтамасын берейік:

1) әрбір қарапайым әрекет стандартты функционалдық блок;

2) әрбір сипатталған үш басқару ағымы стандартты функционалдық блок;

3) басқа стандартты функционалдық блок болмайды.

Әдебиетте стандартты функционалдық блокты кейде *базалық басқару құрылымы* немесе *базалық алгоритмдік құрылым* деп атайды. Сондықтан біз қосымша түсіндірмесіз осы атауларды мәтіннің мағынасына сай еркін қолдана береміз.

Егер ол базалық басқару құрылымдарымен ғана бейнеленсе, құрылымды алгоритм деп айтуымызға болады. Басқаша айтқанда, құрылымды алгоритм үш қарастырылған басқару құрылымының комбинациясынан ғана тұрады.

Құрылымды алгоритмдердің құрылымсыз алгоритмдерге қарағанда бірталай артықшылықтары бар:

– алгоритмді қабылдаудың *түсініктілігі* мен *қарапайымдылығы* (себебі ол құрылған алғашқы құрылымдардың саны көп емес);

– *тексерімділік* (кез келген негізгі құрылымды тексеру үшін оның құрамындағы функционалдық блоктардың дұрыстығына көз жеткізсе болғаны);

– *жаңғыртушылық* (алгоритмнің құрылымын жай ғана өзгертсе болғаны, себебі оның құрамдас функционалдық блоктары салыстырмалы тәуелсіз).

Басқару құрылымдарының алгоритмді құруда маңызы зор. Сондықтан төменде басқару құрылымдарын әртүрлі алгоритмдік тілдеріндегі бейнелері және қасиеттері қарастырылады.

Басқару құрылымдарын бейнелеу үшін блок-сұлба тілі, дарақ тәрізді тіл және жасанды вербалды тіл қолданылады.

III.3.1. Ескертпе:

Жасанды вербалды тілдегі сөйлем «:» таңбасымен басталатын болса, онда ол алгоритмге қатысты түсініктеме болады және ондағы көрсетілген әрекет орындалмайды. Бекітілген сөздер осы тілдің меншікті (әліпби) таңбалары ретінде қолданылады. Ал бұрыштық жақша таңбалары «< » және « > » бұл тілдің меншікті таңбалары болмайды және оның ішіндегі өрнек әрекеттің осы тілдегі бейнесі емес, белгісі болады. Яғни егер әрекеттің бейнесі айқын жазылса, онда бұрыштық жақша қолданылмайды. Сол сияқты тік жақша таңбалары да « [» және «] » бұл тілдің меншікті таңбалары болмайды. Олардың ішіне кескіннің міндетті емес бөліктері жазылады, яғни бұл бөліктерді кейде көрсетпеуге де болады.

III.3.1. Мысалдар:

1.Тізбекті әрекеттер тізбектелу басқару құрылымы арқылы жазылады.

2. Айырылымды әрекеттер тармақталу басқару құрылымы арқылы жазылады.

3. Қайталымды әрекеттер қайталану басқару құрылымы арқылы жазылады.

III.3.1. Тапсырмалар:

1. Екі функционалды блоктың бір блокка бірігуін көрсетіңіз.
2. Тармақталудың түрлерін көрсетіңіз.
3. Құрылымды алгоритмнің қасиеттерін көрсетіңіз.

Көмек:

1. Басқару ағымының екі қасиетін де қанағаттандыратын функционалды болктарды алыңыз.
2. Логикалық шарттың мәніне байланысты бір немесе бірнеше функционалдық блоктың бірін орындауды ұйымдастырады.

3. Бұл қасиеттер ыңғайлылықты жүзеге асырады, құруды жеңілдетеді.

III.3.1. Сұрақтар:

1. Алгоритмдік басқару құрылымы тізбектеуді қалай анықтауға болады?
2. Алгоритмдік басқару құрылымы тармақталуды қалай анықтауға болады?
3. Алгоритмдік басқару құрылымы қайталауды қалай анықтауға болады?

III.3.1. Тесттер:

1. Алгоритмдік басқару құрылымы тармақталудың қандай түрлері болады?

A) қарапайым, баламалы, көпмәнді;

B) қарапайым, баламалы;

C) баламалы, көп мәнді;

D) қарапайым, күрделі;

E) алғышартты, соңғышартты, параметрлі.

2. Алгоритмде қайталауды ұйымдастырудың қандай түрлері болады?

A) қарапайым, баламалы, көпмәнді;

B) алғышартты, соңғышартты, параметрлі;

C) шартты, шартсыз;

D) қарапайым, күрделі;

E) орташа, күрделі.

3. Алгоритмдік басқару құрылымдарының бейнеленуі қалай беріледі?

A) қарапайым, баламалы, көпмәнді;

B) блок-сұлба тілінде, дарақ тәрізді тілде, вербалды тілде;

C) алгоритмде;

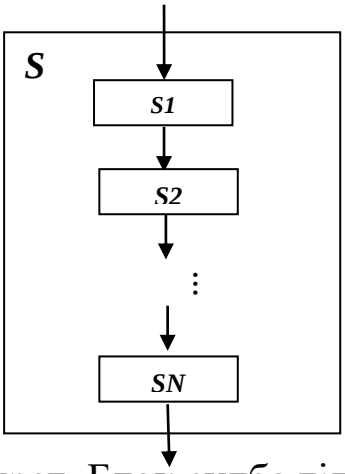
D) программада;

E) табиғи тілде.

III.3.2. Тізбектеу

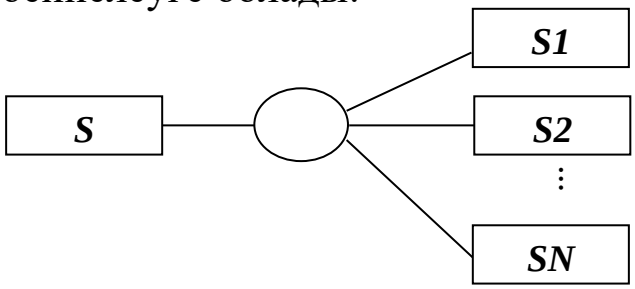
Тізбектеу басқару құрылымы бірнеше қарапайым әрекеттердің тізбегінен бір күрделі әрекет құрастыруға болатындығын көрсетеді.

Егер ***S1***, ***S2***,., ***SN*** қарапайым әрекеттер және олардың орындалу реті өздерінің нөмірлерінің өсу ретіне сәйкес болса, онда осы әрекеттерден құралған күрделі ***S*** әрекетінің блок сұлба тіліндегі бейнесі III.3.2.1-суретте көрсетілген:



III.3.2.1-сурет. Блок сұлба тіліндегі тізбектеу

Тізбектеу басқару құрылымын дарақ тәрізді тілде III.3.2.2-суреттегідей бейнелеуге болады:



III.3.2.2-сурет. Дарақ тәрізді тілде**гі** тізбектеу

Тізбектеудің вербалды тілдегі өрнектелуі III.3.2.3-суретте берілген:

: **S** күрделі әрекетінің денесі
БАСЫ
< ***S1*** >
< ***S2*** >
...
< ***SN***>
СОҢЫ

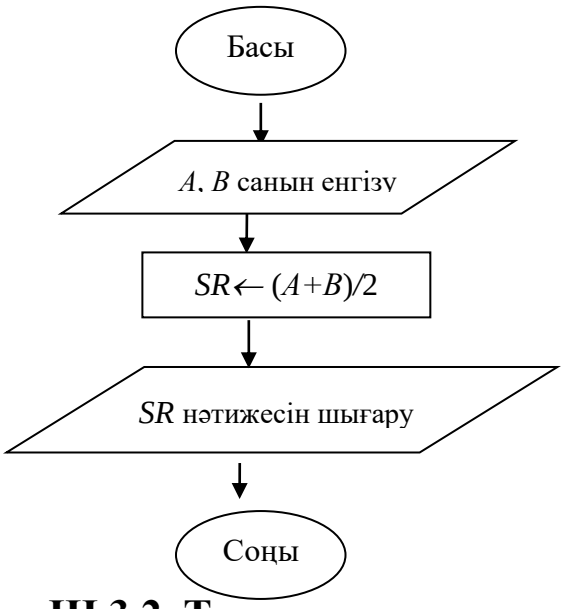
III.3.2.3-сурет. Вербалды тілдегі тізбектеу

III.3.2. Ескертпе:

Тізбектеуді қолданған кезде ***S*** әрекетінің алғашқы деректері ретінде қатардағы ең бірінші ***S1*** әрекетінің алғашқы деректері алынады, ал ***S*** әрекетінің нәтижесі ретінде қатардағы ең соңғы ***SN*** әрекетінің нәтижесі есептелінеді.

III.3.2. Мысалдар:

Екі санның арифметикалық ортасын табу алгоритмін блок-сұлба және вербалды тілде жазу.



Алг Арифметикалық ортасы

Нақты *A*, *B*, *SR*

A, *B*-ны енгізу

SR:=(*A*+*B*)/ 2

SR -ті шығару

Соңы

III.3.2. Тапсырмалар:

1. Квадрат теңдеудің дискриминантының мәнін есептейтін алгоритмді дарақ тәрізді тілде жазыңыз.

2. Табаны *B* және биіктігі *H* тік үшбұрыштың ауданын есептеп табатын алгоритмді жасанды вербалды тілде жазыңыз.

3. Әртүлі радиусты шеңберлердің аудандарын табатын алгоритмді блок-сұлба тілінде жазыңыз.

Көмек:

1. Тік үшбұрыштың ауданы *S* мына формуламен есептелінеді

$$S = B*H/2$$

2. Квадрат теңдеудің дискриминанты мына формуламен табылады $D = B^2 - 4AC$, мұнда *A*, *B*, *C* – коэфициенттер.

3. Егер шеңбердің радиусын *R* деп белгілесек, онда оның ауданы *S* мына формула арқылы табылады:

$$S = \pi*R^2, \text{ мұнда } \pi = 3{,}14.$$

III.3.2. Сұрақтар:

1. S1, S2,..., SN қарапайым әрекеттерінен құралған күрделі ***S*** әрекеті вербалды тілде қалай бейнеленеді?

2. S1, S2,..., SN қарапайым әрекеттерінің тізбегінен құралған ***S*** күрделі әрекеттің алғашқы деректері мен нәтижесі ретінде нелер алынады?

3. S1, S2,..., SN қарапайым әрекеттерінен құралған күрделі ***S*** әрекеті блок сұлба тілінде қалай бйнеленеді?

III.3.2. Тесттер:

1. Басқару құрылымдарының түрлері қандай?

A) Тізбектеу, тармақталу, қарапайым;
B) Қарапайым, тармақталу, қайталау;
C) Тізбектеу, күрделі, қайталау;
D) Тізбектеу, тармақталу, қайталау;
E) Қарапайым, күрделі, тізбектеу.

2. Тізбектеу басқару құрылымын құру үшін ең кемінде қанша әрекет керек?

A) 5;
B) 3;
C) 2;
D) 4;
E) 1.

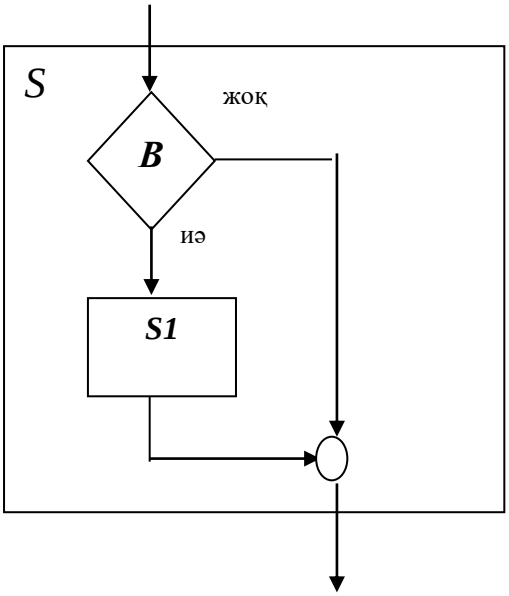
3. Екі санның қосындысын есептеу үшін қандай алгоритмдік басқару құрылымын қолдану керек?

A) Тізбектеу;
B) Қарапайым;
C) Қайталау;
D) Тармақталу;
E) Балама.

III.3.3. Қарапайым тармақталу

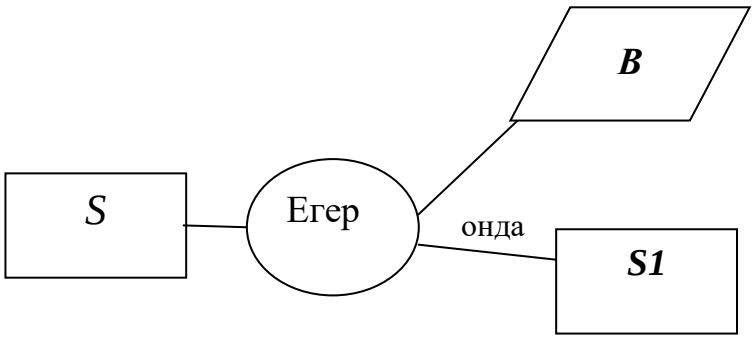
Қарапайым тармақталу күрделі ***S*** әрекеті белгілі ***B*** шартына байланысты берілген ***S1*** әрекетін орындау керек немесе еш нәрсе орындамау керек дегеннен құрылатындығын көрсетеді. Яғни ***B*** шарты қанағаттандырылған жағдайда ***S1*** әрекеті орындалады, ал басқа жағдайда еш нәрсе орындамайды.

Қарапайым тармақталу блок сұлба тілінде III.3.3.1-суретпен бейнеленген:



III.3.3.1-сурет. Блок-сұлба тіліндегі қарапайым тармақталу

Қарапайым тармақталудың дарақ тәрізді тілдегі бейнесі III.3.3.2-суретте көрсетілген:



III.3.3.2-сурет. Дарақ тәрізді тілдегі қарапайым тармақталу

Қарапайым тармақталудың вербалды тілдегі өрнектелуі III.3.3.3-суретте берілген:

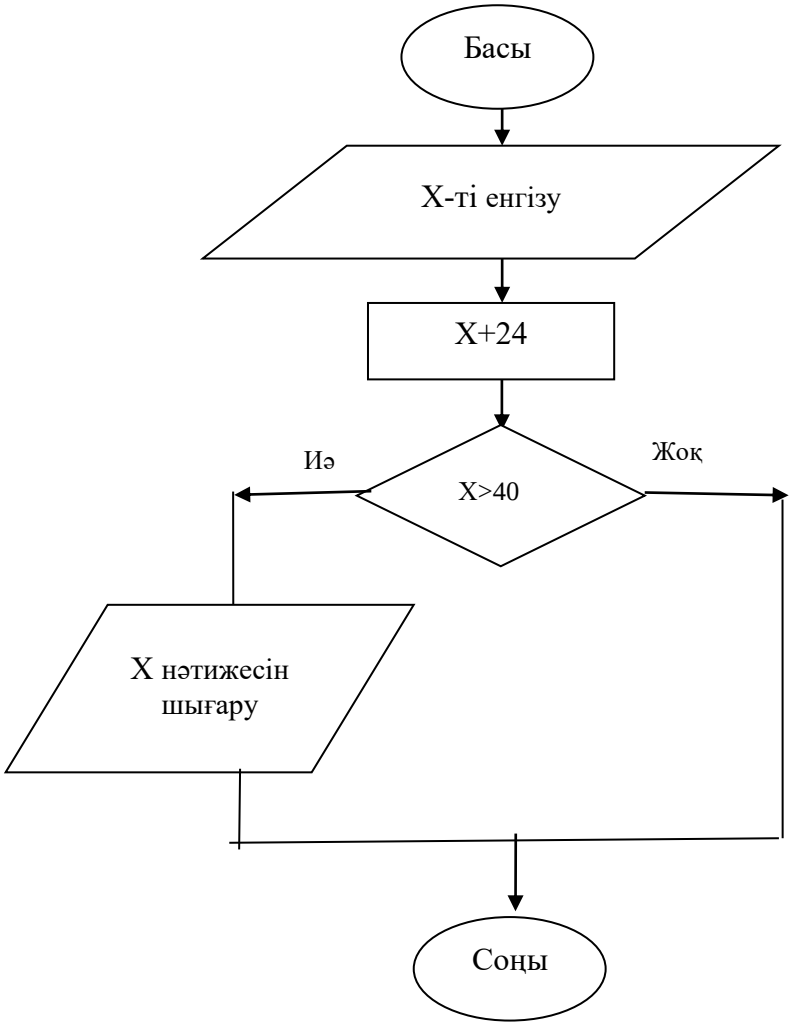
: ***S*** күрделі әрекетінің денесі

ЕГЕР <***B***> ОНДА <***S1***>

II.3.3.3-сурет. Вербалды тілдегі қарапайым тармақталу

III.3.3. Мысалдар:

X+24>40 шарттарын қанағаттандыратын X мәнін табудың блок-сұлбасын құру төмендегідей орындалады.



III.3.3. Тапсырмалар:

X және Y айнымалысын өсу ретімен орналастырудың блок-сұлбасын жазыңыз.

Көмек:

Егер айнымалылар X < Y шартын қанағаттандырса, онда оларды өзгеріссіз қалдырыңыз; егер X ≥ Y болса, онда олардың орнын ауыстыру керек.

III.3.3. Сұрақтар:

1. Қарапайым тармақталу вербалды тілде қалай белгіленеді?
2. Қарапайым тармақталу дарақ тәріздес тілде қалай болады?
3. Қарапайым тармақталу блок-сұлба тілінде қалай болады?

III.3.3 Тесттер:

1. Екі санның ең үлкенін табу алгоритмінде қандай алгоритм түрі қолданылады?

A) сызықты;
B) циклдік;
C) тармақталу;
D) иерархиялық;
E) рекурсиялық.

2. Алгоритмдік тармақталу басқару құрылымының түрлері қайсысы?

A) әзірге қайталау, қайталау шейін, қайталау дейін;
B) тармақтау, тізбектеу, қайталау;
C) жай тармақтау, баламалы тармақтау, көпмәнді тармақтау;
D) жай тармақтау, қайталау, қайталау дейін;
E) қайталау, тізбектеу.

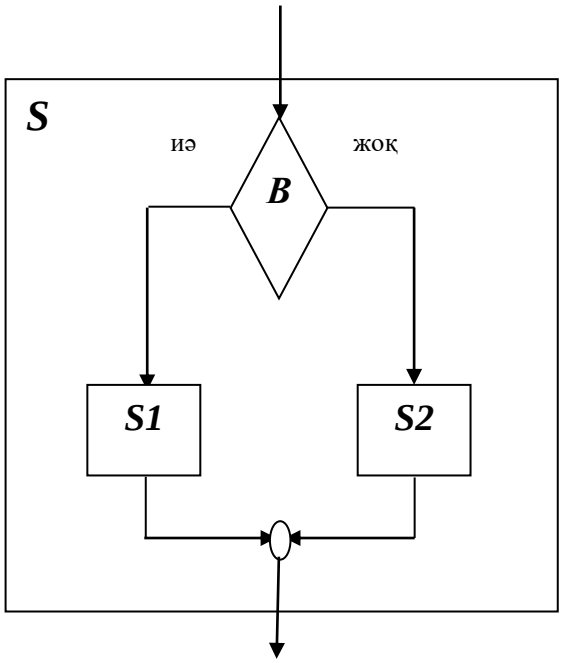
3. Алгоритмдік тармақталу басқару құрылымы деп нені айтамыз?

A) Кейбір пәрмендердің бірнеше рет орындалуын айтады;
B) Шартқа тәуелді әртүрлі пәрмендердің орындалуын айтады;
C) Кесте функциясын есептеуді айтады;
D) Ойындарда әртүрлі жағдайлар құруды айтады;
E) Екі әртүрлі алгоритмде тапсырма беруді айтады.

III.3.4. Баламалы тармақталу

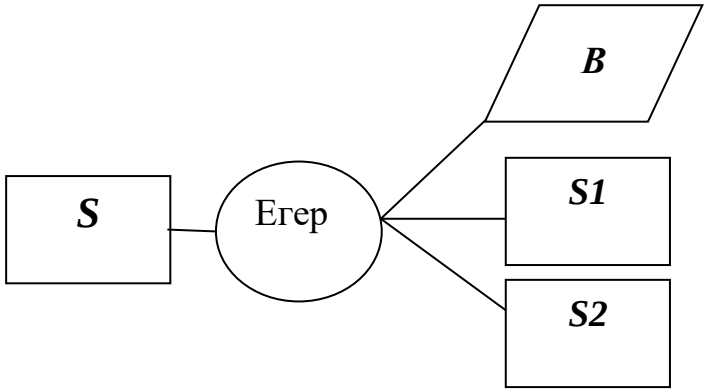
Баламалы тармақталу күрделі ***S*** әрекеті белгілі бір ***B*** шартына байланысты берілген ***S1*** әрекетін немесе ***S2*** әрекетін орындау керек дегенді көрсетеді. Яғни егер ***B*** шарты қанағаттандырылған жағдайда ***S1*** әрекеті орындалады, әйтпесе ***S2*** әрекеті орындалады.

Баламалы тармақталу блок сұлба тілінде III.3.4.1-суретте:



III.3.4.1-сурет. Блок сұлба тіліндегі баламалы тармақталу

Баламалы тармақталу басқару құрылымының дарақ тәрізді тілдегі бейнесі III.3.4.2-суретте көрсетілген:



III.3.4.2-сурет. Дарақ тәрізді тілдегі баламалы тармақталу

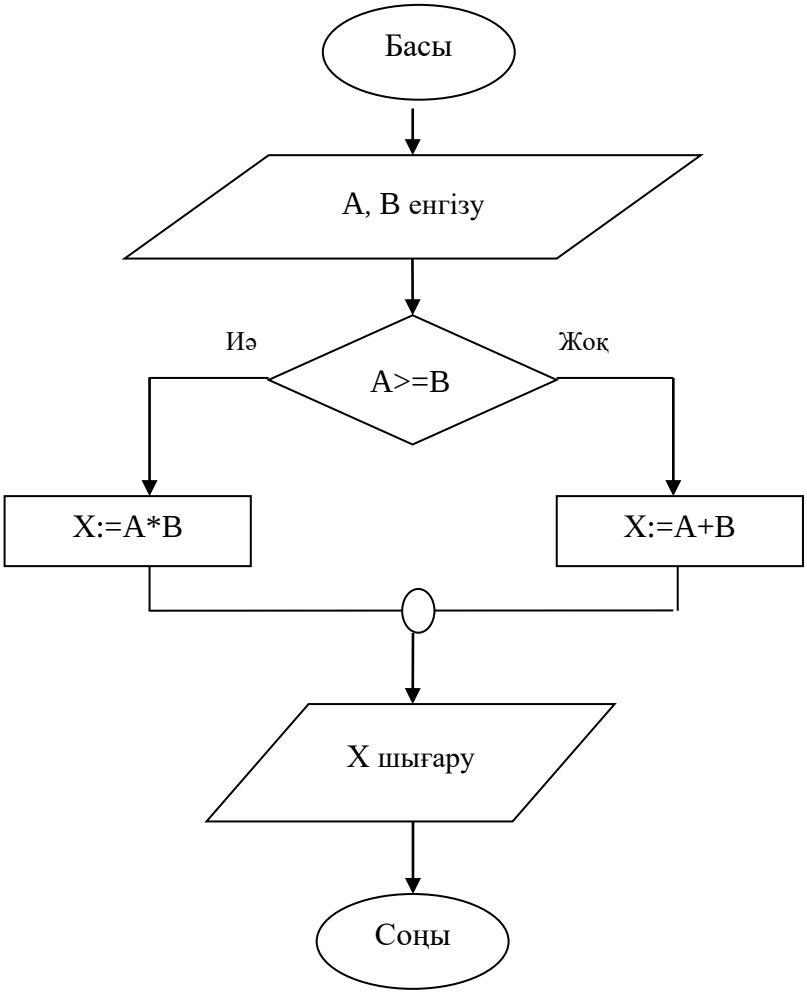
Баламалы тармақталу басқару құрылымының вербалды тілдегі өрнектелуі III.3.4.3- суретте берілген:

: ***S*** күрделі әрекетінің денесі
ЕГЕР <***B***> ОНДА <***S1***>
ӘЙТПЕСЕ <***S2***>

III.3.4.3-сурет. Вербалды тілдегі баламалы тармақталу

III.3.4. Мысалдар:

Алгоритмде *A* мен *B* – алғашқы деректер, *X* – нәтиже. Егер *A>=B* ақиқат болса, онда *X:=A*B* пәрмені орындалады, әйтпесе *X:=A+B* пәрмені орындалады. Нәтижесінде *X* мәні шығады.



III.3.4. Тапсырмалар:

1. Вербалды тілде екі санның ішіндегі үлкенін ***A*** есіміне кішісін ***B*** есіміне меншіктейтін тармақталуды бейнелеңіз.

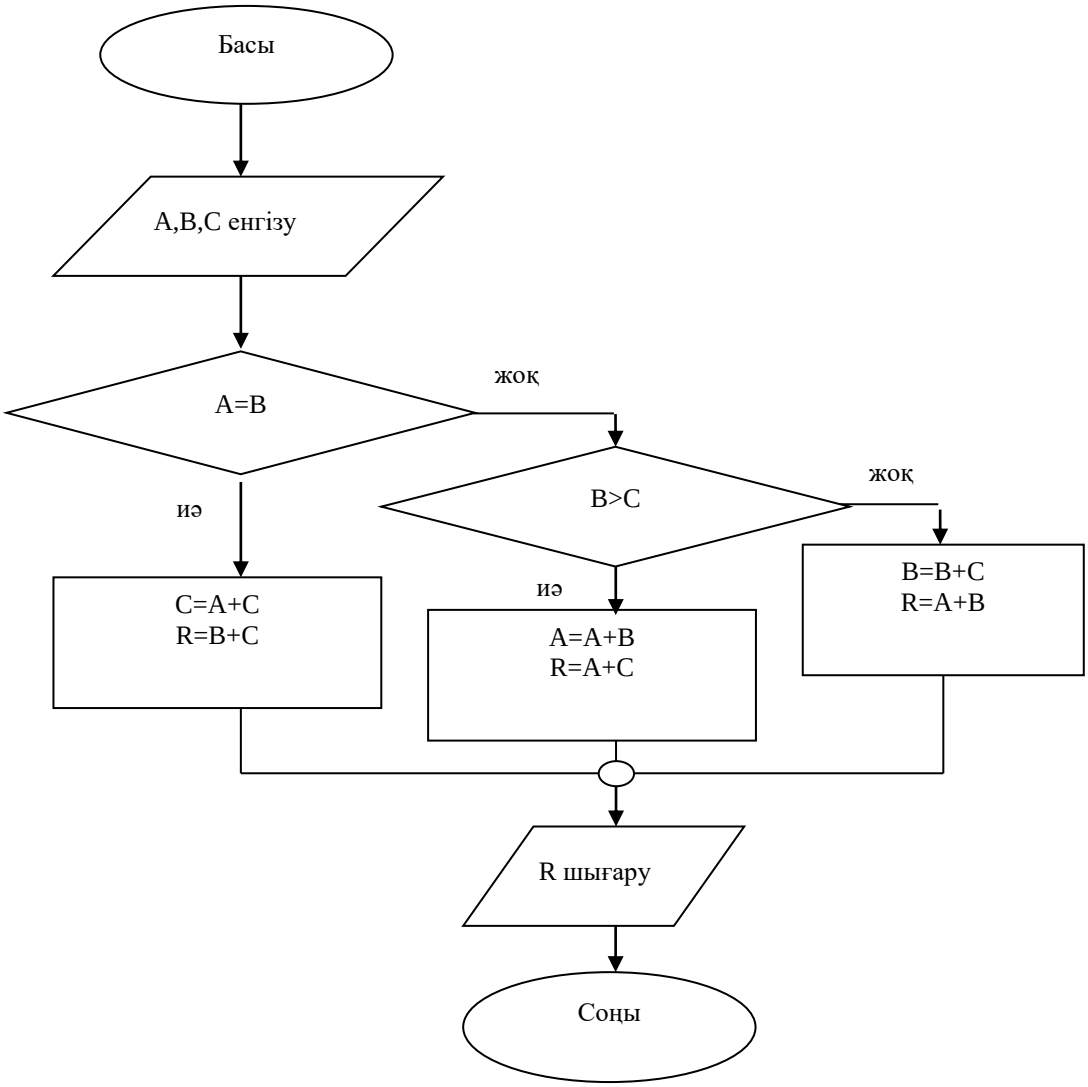
2. Мына тармақталудың қай түріне жататындығын анықтаңыз?

ЕГЕР <шарт>

ОНДА <1-пәрмен>

ӘЙТПЕСЕ <2-пәрмен >

3. Төмендегі алгоритмде егер A, B, C мәндері сәйкесінше 1, 2, 6 болса, R нәтижесінің мәнін анықтаңыз.



Көмек:

1. Шарт бойынша екі әрекетте баламалы орындалуы керек.

2. Баламалы тармақталуға жатады.

3. Алгоритмдегі баламалы тамақталуда көрсетілген әрекеттерді орындау керек.

III.3.4. Сұрақтар:

1. Алгоритмдік баламалы тармақталу құрылымы блок-сұлба тілінде қалай болады?
2. Алгоритмдік баламалы тармақталу құрылымы дарақ тәріздес тілде қалай болады?
3. Алгоритмдік баламалы тармақталу құрылымы вербалды тілде қалай белгіленеді?

III.3.4.Тесттер:

1. Алгоритмдік баламалы тармақталу құрылымында қанша тармақ бар?

A) 2;
B) 4;
C) 1;
D) 3;
E) 0.

2. Алгоритмдік баламалы тармақталу құрылымында баламалы әрекет қашан орындалады?

A) егер шарт анықталса;
B) егер шарт орындалса;
C) егер шарт ақиқат болса;
D) егер шарт жалған болса;
E) егер шарт қате болса.

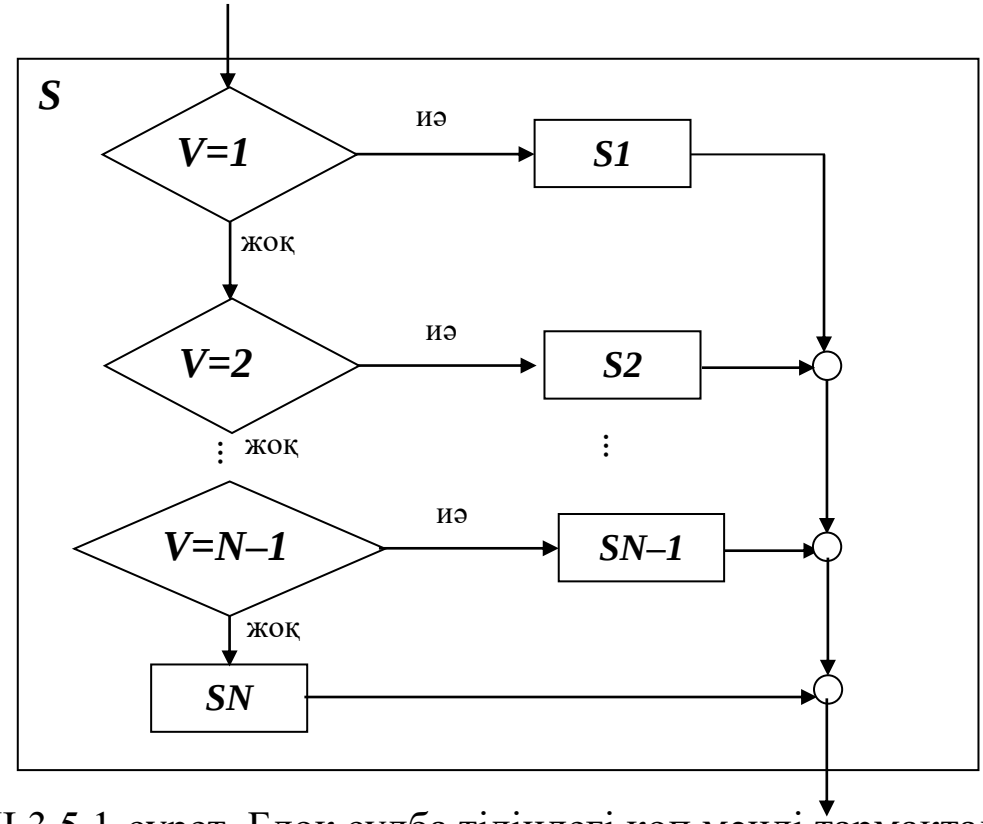
3. Алгоритмдік баламалы тармақталу құрылымын моделдеу үшін қарапайым тармақталудың саны қаншадан кем болмауы керек?

A) Төрттен;
B) Бірден;
C) Белгісіз;
D) Үштен;
E) Екіден көп.

III.3.5. Көп мәнді тармақталу

Көп мәнді тармақталу (таңдау) күрделі ***S*** әрекеті белгілі айнымалы шама ***V*** өзінің әр түрлі мүмкін ***1***, ***2***,., ***N*** мәндерінің біреуін міндетті түрде қабылдауына сәйкес берілген әр түрлі ***S1***, ***S2***,., ***SN*** әрекеттерінің біреуі міндетті түрде орындалуы қажет дегенді көрсетеді. Яғни егер ***V***=1 болса, онда ***S1*** орындалады, әйтпесе егер ***V=2*** болса, онда ***S2*** орындалады, ал егер ***V=N–1*** болса, онда ***SN–1*** орындалады, әйтпесе ***SN*** орындалады.

Көп мәнді тармақталу блок сұлба тілінде III.3.5.1-суретте берілген:



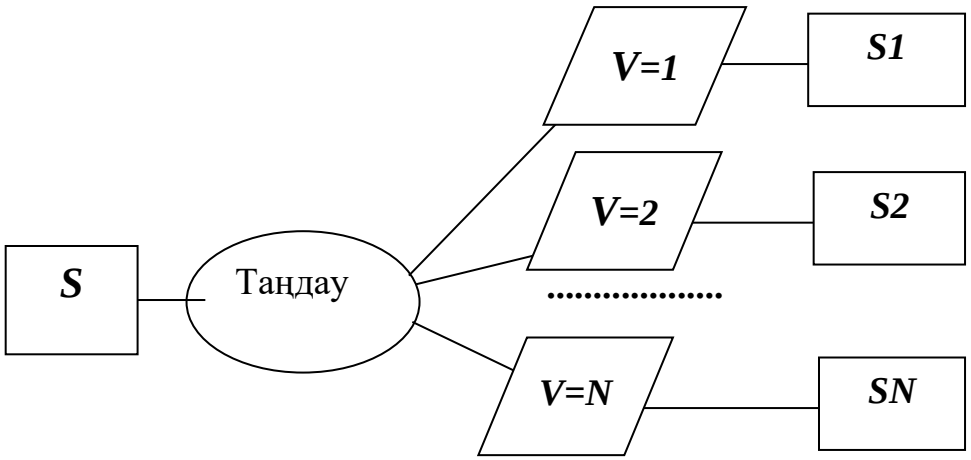
III.3.5.1-сурет. Блок сұлба тіліндегі көп мәнді тармақталу

Көп мәнді тармақталудың вербалды тілдегі өрнектелуі III.3.5.2-суретте берілген:

: ***S*** күрделі әрекетінің денесі
ТАҢДАУ
V= 1 БОЛСА ***S1***
V = 2 БОЛСА ***S2***
. . . . . .
V = N БОЛСА ***SN***

III.3.5.2- сурет. Вербалды тілдегі көп мәнді тармақталу

Көп мәнді тармақталудың дарақ тәрізді тілдегі бейнесі III.3.5.3-суретте көрсетілген:



III.3.5.3-сурет. Дарақ тәрізді тілдегі көп мәнді тармақталу

Егер вербалды тілде көп мәнді тармақталуға арнаулы өрнек – құрылым (мысалы, Таңдау) қарастырылмаса, онда оны баламалы таңдау құрылымы арқылы жазуға болады. Таңдаудың ондай жазылуы III.3.5.4-суретте көрсетілген:

: ***S*** күрделі әрекетінің денесі
ЕГЕР ***V = 1*** ОНДА <***S1*** >
ӘЙТПЕСЕ ЕГЕР ***V = 2*** ОНДА <***S2*** >
ӘЙТПЕСЕ ЕГЕР ***V = 3*** ОНДА <***S3*** >
. . . . . .
ӘЙТЕСЕ ЕГЕР ***V = N–1*** ОНДА <***SN–1*** >
< ***SN*** >

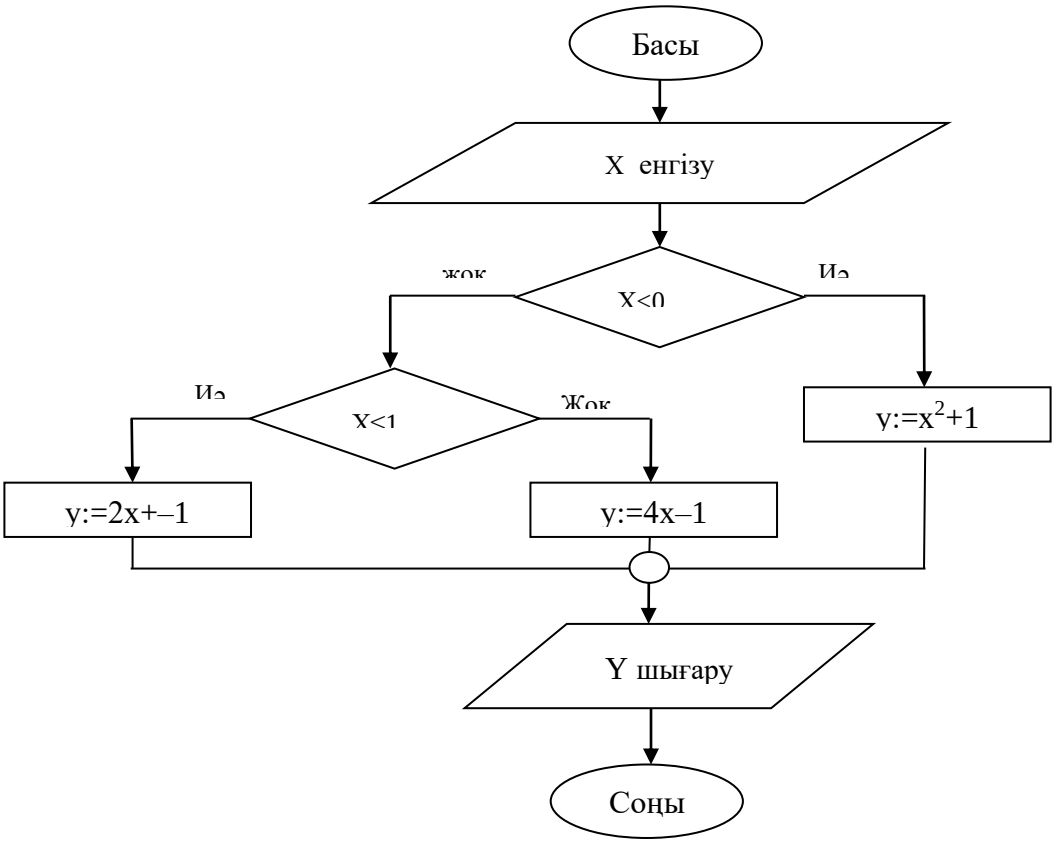
III.3.5.4-сурет. Таңдаудың баламалы тармақталу арқылы берілуі.

III.3.5. Мысалдар:

Берілген функцияның мәнін анықтайтын алгоритмнің құру жолдарын қарастырайық:

$$y = \begin{cases} x^2 + 1, & x < 0 \\ 2x + 1, & 0 \le x \le 1 \\ 4x - 1, & x > 1 \end{cases}$$

Бұл алгоритм блок-сұлба тілінде бейнесі былай кескінделеді:



III.3.5. Тапсырмалар:

1. Үш санның ең үлкенін анықтайтын блок-сұлба жазыңыз.

2. Берілген пәрмендер қандай тармақталуды сипаттайтынын көрсетіңіз:

ЕГЕР <1-шарт>
ОНДА <1-пәрмен>
ЕГЕР <2-шарт>
ОНДА <2-пәрмен>
.....
ЕГЕР <N-шарт>
ОНДА <N-пәрмен>

3. Енгізілген санның 1-ден 3-ке дейінгі мәндерінің атауларын шығаратын алгоритмнің блок-сұлбасын құрыңыз.

Көмек:

1. Үш түрлі тармақталудың ең күрделісін қолдану керек.

2. ***S*** әрекеті ***V*** айнымалы шамасының мәндерімен байланысты әр түрлі ***S1***, ***S2***,., ***SN*** әрекеттерінің біреуі болады.

3. Енгізілген санды 1-ден 3-ке дейінгі сандармен салыстыратын көпмәнді тармақталу құру керек.

III.3.5. Сұрақтар:

3. Көп мәнді тармақталу вербалды тілде қалай көрсетіледі?
4. Көп мәнді тармақталу блок сұлба тілінде қалай көрсетіледі?
5. Таңдау дарақ тәрізді тілде бейнесі қалай көрсетіледі?

III.3.5. Тесттер:

1. Тармақталудың қандай түрлері бар?

A) әзірге қайталау, қайталауға шейін, қайталауға дейін;

B) тармақталу, тізбектеу, қайталау;

C) жай тармақталу, баламалы тармақталу, көпмәнді тармақталу;

D) жай тармақталу, қайталау, қайталауға дейін;

E) қайталау, тізбектеу.

2. Қайсы тармақталуда тек бір ғана шарт тексеріледі?

A) Жай тармақтау;

B) Баламалы тармақтау;

C) Көп мәнді тармақтау;

D) Таңдау;

E) Кіріктірілген тармақтау.

3. Мейлі $F(x) = \begin{cases} x, & егер \quad x > 0 \\ 0, & егер \quad x = 0 \\ -x^2, & егер \quad x < 0 \end{cases}$

Сонда функцияның мәні қандай басқару арқылы табылады?

A) тармақталу;

B) параметрлі қайталау;

C) таңдау;

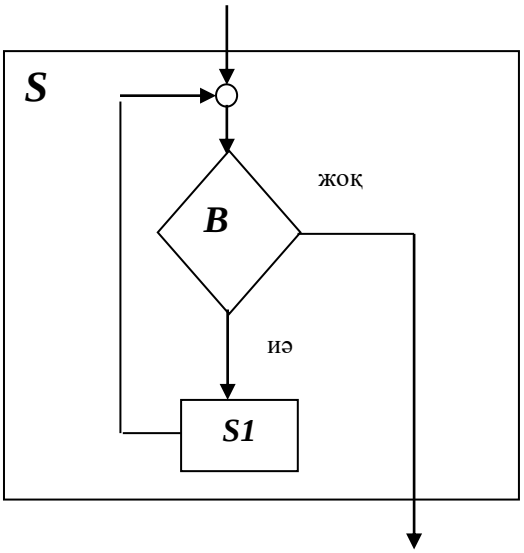
D) әзірге қайталау;

E) сызықты.

III.3.6. Алғышартты қайталау

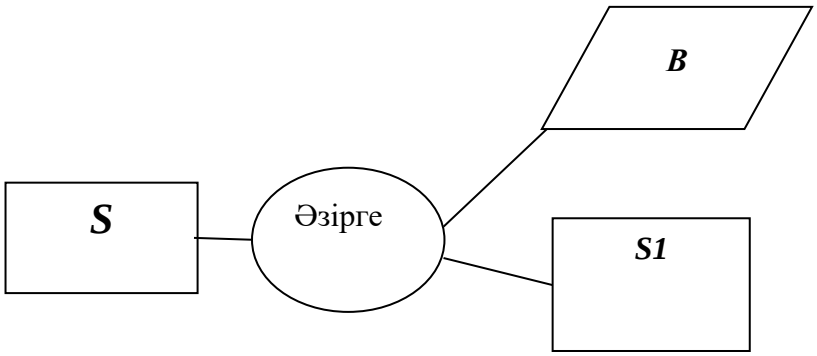
Алғышартты қайталау күрделі ***S*** әрекеті белгілі ***B*** шарты қанағаттандырылса, қайталау денесі ***S1*** әрекетін орындауды қайталау керек, ал егер осы ***B*** шарты қанағаттандырылмаса, еш нәрсе орындамау керек дегенді көрсетеді.

Алгоритмдік алғышартты қайталау құрылымы блок сұлба тілінде III.3.6.1-суретте бейнеленген:



III.3.6.1-сурет. Блок сұлба тіліндегі алғышартты қайталау

Алгоритмдік алғышартты қайталау құрылымының дарақ тәрізді тілдегі бейнесі III.3.6.2-суретте көрсетілген:



III.3.6.2-сурет. Дарақ тәрізді тілдегі алғышартты қайталау

Алгоритмдік алғышартты қайталау құрылымының вербалды тілдегі өрнектелуі III.3.6.3- суретте берілген:

: ***S*** күрделі әрекетінің денесі
ӘЗІРГЕ <***B***>
ҚАЙТАЛАУ <***S1***>

III.3.6.3- сурет. Вербалды тілдегі алғышартты қайталау

Бұл басқару құрылымы бойынша мына жағдайлар бар**:**

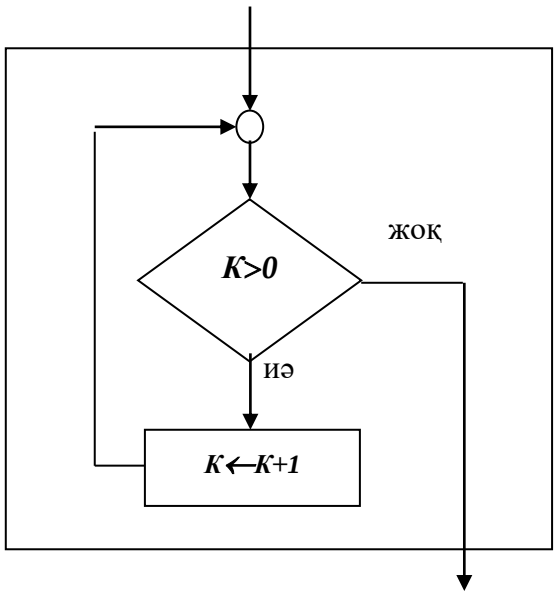
1. Егер күрделі ***S*** әрекетін орындауды алғаш бастағанда ***B*** шарты қанағаттандырылмаса, онда қайталау денесі ***S1*** мүлде орындалмайды;

2. Егер күрделі ***S*** әрекетін орындауды алғаш бастағанда ***B*** шарты қанағаттандырылса және қайталау денесі ***S1***–дің орындалуы ***B*** шартының мәніне әсер етпесе, онда ***S1*** әрекеті шексіз рет орындалуды талап етеді;

3. Егер күрделі ***S*** әрекетін орындауды алғаш бастағанда ***B*** шарты қанағаттандырылса және оның мәніне ***S1*** әрекетінің орындалуы әсер етсе, онда ***S1*** әрекеті ***B*** шартының мәні өзгергенше бірнеше рет орындалады (***S1*** әрекетінің орындалуы қайталанады).

III.3.6. Мысалдар:

1. Алғышартты қайталаудың дұрыс құрылмауы төмендегі суретте көрсетілген:

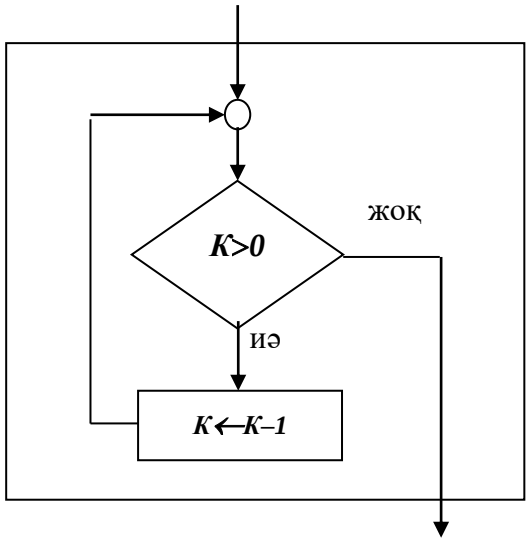


Бұл мысалда қайталау денесі ***К←К+1*** берілген ***К>0*** шартының мәніне ешқандай әсер етпейді. Сондықтан:

– егер ***К*** шамасының мәні алдын ала оң сан болмаса, яғни ***К<0***, онда қайталау денесі мүлде орындалмайды (1-ші жағдай);

– егер айнымалы ***К*** шамасының мәні алдын ала оң сан болса, яғни ***К>0***, онда қайталау денесінің орындалуы шексіз қайталана береді (2-ші жағдай).

2. Алғышартты қайталаудың дұрыс құрылуы төмендегі суретте көрсетілген:



Бұл мысалда қайталау денесі ***К←К–1*** берілген ***К>0*** шартының мәніне әсер етеді. Сондықтан, егер ***К*** шамасының мәні алдын ала оң сан болса, онда қайталау денесінің орындалуы ***К***-нің мәні нөл немесе теріс, яғни ***К≤0***, болғанша қайталана береді (3-ші жағдай);

III.3.6. Тапсырмалар:

1. Алгоритмдік алғышартты қайталау құрылымын пайдаланып 1-ден бастап 10-ға дейінгі бүтін сандарды экранға шығарыңыз.

2. Алгоритмдік алғышартты қайталау құрылымын пайдаланып 100-ге дейінгі сандардың қосындысын *S* деп белгілеп, оның мәнін есептеңіз.

3. Алгоритмдік алғышартты қайталау құрылымын пайдаланып берілген *n* санының факториалын *F* деп, оның мәнін есептеңіз.

Көмек:

1. Әзірше қайталау басқару құрылымын пайдаланыңыз.

2. Әзірше қайталау басқару құрылымын пайдаланып, $S \leq 100$ шарты орындалғанша $S = S+1$ есептеу керек.

3. Әзірше қайталау басқару құрылымын пайдаланып, $F \leq 100$ шарты орындалғанша $F = F*(F-1)!$ есептеу керек.

III.3.6. Сұрақтар:

1. Алғышартты қайталаудың мағынасы неде?

2. Алғышартты қайталауда қандай жағдайлар болады?

3. Алғышартты қайталауда тексерілетін шарт қайталау денесіне тәуелсіз орындалып тұрса не болады?

III.3.6. Тесттер:

1. Егер күрделі ***S*** әрекетін орындауды алғаш бастағанда ***B*** шарты қанағаттандырылмаса, онда ***S1*** әрекетімен не болады?

A) Орындалады;
B) Тармақталады;
C) Қайталанады;
D) Өзгереді;
E) Орындалмайды.

2. Кейбір бөлек пәрмендерді не пәрмендер тобын орындауды қайталайтын алгоритм қалай аталады?

A) Тармақталған;
B) Сызықты;
C) Циклдық;
D) Тізбектелген;
E) Рекурентті.

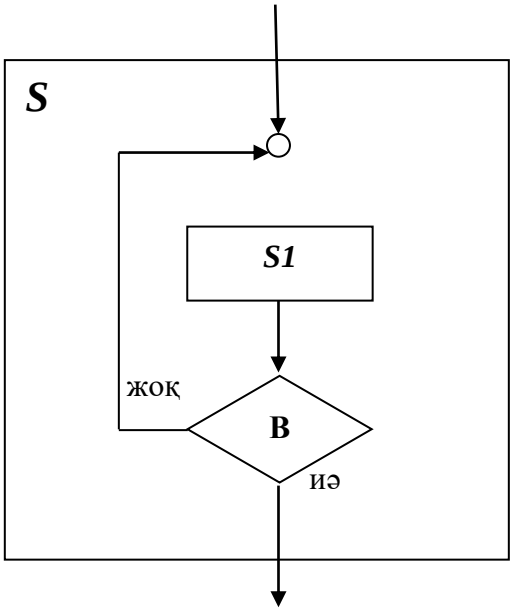
3. Евклид алгоритмі қандай алгоритмге жатады?

A) Тармақталған;
B) Сызықты;
C) Циклдық;
D) Тізбектелген;
E) Рекурсивті.

III.3.7. Соңғышартты қайталау

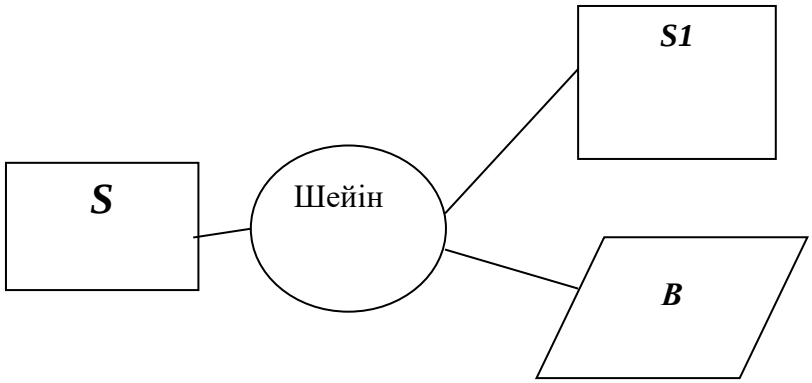
Соңғышартты қайталау күрделі ***S*** әрекеті қайталау денесі болатын берілген ***S1*** әрекетін орындауды белгілі ***B*** шарты қанағаттандырылғанға шейін қайталай беру керек, ал осы ***B*** шарты қанағаттандырылса қайталауды тоқтату керек дегенді көрсетеді.

Соңғышартты қайталау блок сұлба тілінде III.3.7.1-суретте бейнеленген:



III.3.7.1-сурет. Блок сұлба тіліндегі соңғышартты қайталау

Соңғышартты қайталаудың дарақ тәрізді тілдегі бейнесі III.3.7.2-суретте көрсетілген:



III.3.7.2-сурет. Дарақ тәрізді тілдегі соңғышартты қайталау

Соңғышартты қайталаудың вербалды тілдегі өрнектелуі III.3.7.3-суретте берілген:

: ***S*** күрделі әрекетінің денесі
ҚАЙТАЛАУ <***S1***>
<***B***> ШЕЙІН

III.3.7.3-сурет. Вербалды тілдегі соңғышартты қайталау

Соңғышартты қайталау бойынша мына жағдайлар бар**:**

1. Егер ***S*** әрекетін алғаш орындағанда ***S1*** әрекетінің орындалуы ***B*** шартының мәніне әсер етпесе және ***B*** шарты қанағаттандырылса, онда ***S1*** әрекеті бір рет қана орындалады;

2. Егер ***S*** әрекетін алғаш орындағанда ***S1*** әрекетінің орындалуы ***B*** шартының мәніне әсер етпесе және ***B*** шарты қанағаттандырылмаса, онда ***S1*** әрекеті шексіз рет орындалуды талап етеді;

3. Егер ***S*** әрекетін алғаш орындағанда ***S1*** әрекетінің орындалуы ***B*** шартының мәніне әсер етсе және ***B*** шарты қанағаттандырылмаса, онда ***S1*** әрекеті ***B*** шартының мәні өзгергенше бірнеше рет орындалады.

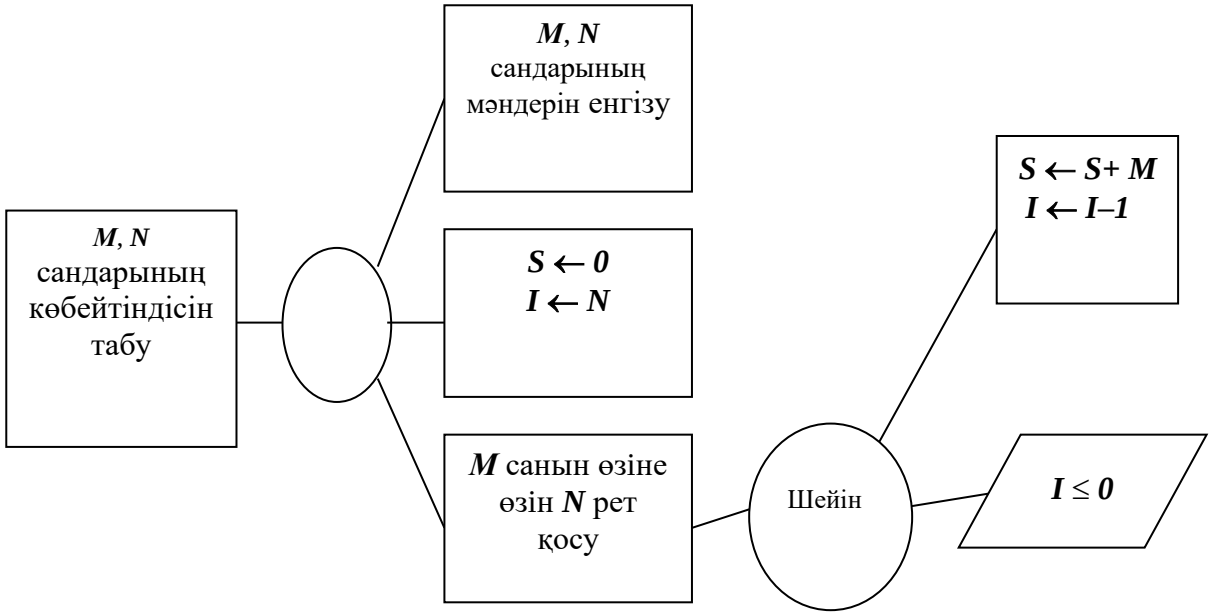
III.3.7. Мысалдар:

Берілген екі бүтін санның көбейтіндісі олардың біреуін өзіне-өзін бірнеше (екінші берілген санға тең) рет қосу арқылы табу алгоритмін құру керек. Ол үшін ***S*** айнымалы шамасын қосындыны жинау үшін, ал ***I*** айнымалы шамасын қосу амалының орындалу санын тексеру үшін тағайындаймыз.

Егер айнымалы шамасының алғашқы мәнін ***N*** айнымалы шамасының мәніне тең болсын деп алсақ, онда берілген ***M, N*** сандарының көбейтіндісін табуды мына формуланы пайдалану арқылы жүзеге асыруға болады:

$$S \leftarrow S+M$$

Мұнда қосу амалының орындалуы ***N*** рет қайталану қажет, яғни ***M*** саны өзіне өзі ***N*** рет қосылады. Осы алгоритм төменгі суретте.



III.3.7. Тапсырмалар:

1. Төмендегі алгоритм орындалған соң ***S***–тің мәні нешеге тең?

```
АЛГ ҚОСЫНДЫ
    БАСЫ
        N=3,  I=1, S=0
        M: S=S+(3*I+2)
        I=I+1
        ЕГЕР I<=N  ОНДА  КӨШУ М
    СОҢЫ
```

Көмек:

1. Егер күрделі ***S*** әрекетін орындауды алғаш бастағанда ***B*** шарты жалған болса, онда қайталау денесі ***S1*** орындалмайды;

2. Егер күрделі ***S*** әрекетін орындауды алғаш бастағанда ***B*** шарты қанағаттандырылса және қайталау денесі ***S1***–дің орындалуы ***B*** шартының мәніне әсер етпесе, онда ***S1*** әрекеті шексіз рет орындалуды талап етеді;

3. Егер күрделі ***S*** әрекетін орындауды алғаш бастағанда ***B*** шарты қанағаттандырылса және оның мәніне ***S1*** әрекетінің орындалуы әсер етсе, онда ***S1*** әрекеті ***B*** шартының мәні өзгергенше бірнеше рет орындалады (***S1*** әрекетінің орындалуы қайталанады).

III.3.7. Сұрақтар:

1. Алгоритмдік соңғышартты қайталау құрылымының қалай орындалады?
2. Алгоритмдік соңғышартты қайталау құрылымын блок-сұлбасы қандай?
3. Орындалу кезінде шарттың мәні өзгермесе не болады?

III.3.7. Тесттер:

1. Циклдік алгоритмнің базалық түрі:

A) «Дейін циклі», «Әзір циклі», «Үшін циклі»;
B) «Сыртқы цикл», «Ішкі цикл»;
C) « Рекурсивтік цикл», «Ақырсыз цикл»;
D) «Қайталау циклі», «Цикл ішіндегі цикл»;
E) «Қайталау циклі», «Процедуралық цикл».

2. Циклдық алгоритм деген не?

A) Пәрмендердің белгілі бір ретпен бірнеше рет қайталануы;
C) Кейбір пәрмендердің жиі-жиі қайталануын айтады;
C) Кесте функциясын есептеуді талап етуді айтады;
D) Ойындарда әртүрлі жағдайлар құруды айтады;
E) Екі әртүрлі алгоритмде тапсырма беруді айтады.

3. Егер Q=2 болса, баспаға не шығады?

```
ЕНГІЗУ Q
        S=0, I=1
        M    S=S+I; I=I+1
         ЕГЕР  S<=Q  ОНДА КӨШУ  M
ШЫҒАРУ  I-2
```

A) 0;
B) 1;
C) 2;
D) 3;
E) 4.

III.3.8. Параметрлік қайталау

Параметрлік қайталау басқару құрылымында қайталау денесінің орындалу саны *параметр* деп аталатын айнымалы шаманың *алғашқы мәні* мен *соңғы мәні* және *қадам* деп аталатын тұрақты шама арқылы анықталады: қайталау денесін әр орындау кезінде параметрдің алғашқы мәніне параметрдің соңғы мәніне жеткенше қадамның мәні (қадам көрсетілмесе 1) қосылып отырады.

Әдетте, параметрдің алғашқы мәні, соңғы мәні және қадам мәні бүтін сандар болады. Сондықтан қайталану саны параметрдің соңғы мәні мен алғашқы мәнінің айырмасын қадам мәніне бөлгендегі шыққан санның бүтін бөлігіне бірді қосқанға тең болады. Мысалы: параметрдің алғашқы мәні 5, соңғы мәні 20, ал қадам мәні 2 болса, онда қайталану саны мына өрнек [(20–5) : 2] + 1 = 8 арқылы есептелінеді, мұнда тік жақшаның ішінде санның бүтін бөлігі.

Жалпы параметрлік қайталау жоғарыда қарастырылған алғышартты қайталаудың мен соңғышартты қайталаудың дербес жағдайы болады. Сондықтан параметрлік қайталаудың графикалық тілдерінде арнаулы кескіні болмайды. Бірақ оның көптеген алгоритмдік (программалау) тілдерінде арнаулы бейнелері бар. Параметрлік қайталаудың вербалды тілдегі кескіні III.3.8.1-суретте:

: ***S*** күрделі әрекетінің денесі
<***V=I,N***> ҚАДАМ <***K***> ҮШІН
ҚАЙТАЛАУ <***S1***>

III.3.8.1-сурет. Параметрлік қайталау вербалды тілде

Мұнда ***V*** – параметр, ***I*** – параметрдің алғашқы мәні, ***N*** – параметрдің соңғы мәні, ***K*** – қадам, ***SI*** – қайталау денесі. болады.

Әр уақытта ***I*** **<** ***N*** болуы керек. Қадамды кейде көрсетпейді. Егер қадам көрсетілмесе, онда оның мәні бірге тең деп есептелінеді.

III.3.8. Мысалдар:

1. Егер 1-ден 100-ге дейінгі тақ сандарының қосындысын табу керек болса, онда оның алгоритмі төмендегі суреттегідей болады:

: Тақ сандарының қосындысы
S ← ***0***
V=1,100 ҚАДАМ ***2*** ҮШІН
ҚАЙТАЛАУ ***S*** ←***S+V***

2. Жоғарыда қарастырылған қайталау басқару құрылымдарын кәдімгі алгебралық өрнектегі жақшалар сияқты етіп бірінің ішіне бірін енгізіп орналастыруға болады, ондай мүмкіншілік төменгі суретте көрсетілген:

: Қабаттасқан қайталаулар
S ← ***0***
: Сыртқы қайталау
L=1,4 ҮШІН
ҚАЙТАЛАУ
    : Ішкі қайталау
    I = ***1,5*** ҮШІН
    ҚАЙТАЛАУ
    S ← ***S*** + ***L*** * ***I***

Бұл алгоритмде ***S*** ←***S+L*** * ***I*** өрнегі ішкі қайталау бойынша 5 рет, ал сыртқы қайталау бойынша 4 рет орындалады. Яғни барлығы 20 рет орындалады да мынадай із қалдырады:

I. ***L = 1***

1. ***I*** = 1, S ← 0 + 1;
2. ***I*** = 2, S ← 1 + 2;

3. $I = 3,\ S \leftarrow 2 + 3;$
4. $I = 4,\ S \leftarrow 5 + 4;$
5. $I = 5,\ S \leftarrow 9 + 5.$

II. $L = 2$

6. $I = 1,\ S \leftarrow 14 + 2;$
7. $I = 2,\ S \leftarrow 16 + 4;$
8. $I = 3,\ S \leftarrow 18 + 6;$
9. $I = 4,\ S \leftarrow 26 + 8;$
10. $I = 5,\ S \leftarrow 34 + 10.$

III. $L = 3$

11. $I = 1,\ S \leftarrow 44 + 3;$
12. $I = 3,\ S \leftarrow 47 + 6;$
13. $I = 3,\ S \leftarrow 53 + 9;$
14. $I = 4,\ S \leftarrow 62 + 12;$
15. $I = 5,\ S \leftarrow 74 + 15;$

IV. $L = 4$

16. $I = 1,\ S \leftarrow 89 + 3;$
17. $I = 3,\ S \leftarrow 92 + 6;$
18. $I = 3,\ S \leftarrow 98 + 9;$
19. $I = 4,\ S \leftarrow 107 + 12;$
20. $I = 5,\ S \leftarrow 119 + 15.$

III.3.8. Тапсырмалар:

Енгізілген 15 бүтін санның ішінен ең кішісін табыңыз.

Көмек:

Параметрлік қайталауды пайдалану керек.

III.3.8. Сұрақтар:

1. Параметрлік қайталау қандай жағдайда орындалмайды?
2. Параметрлік қайталау қалай жұмыс істейді?
3. Параметрлік қайталауды қандай жағадайда қолданады?

III.3.8. Тесттер:

1. Төмендегі алгоритмнің қайсысы циклдік емес?

A) квадрат теңдеуді шешу;

B) N сандарының ішінен ең үлкенін табу;

C) N элементін өсуі бойынша сұрыптау;

D) қораптан алған барлық шарды бір–бірден жою;

E) Баше ойыны.

2. 1-ден 100-ге дейінгі тақ сандарының қосындысын қандай алгоритммен табады?

A) тармақталу;

B) параметрлі қайталау;

C) таңдау;

D) баламалы тағдау;

E) көпмәнді таңдау.

3.Төмендегі жазу қайсы алгоритмге тиісті?

<***V=I,N***> ҚАDAM <***K***> ҮШІН
ҚАЙТАЛАУ <***S1***>

A) тармақталу;

B) параметрлі қайталау;

C) таңдау;

D) әзірге қайталау;

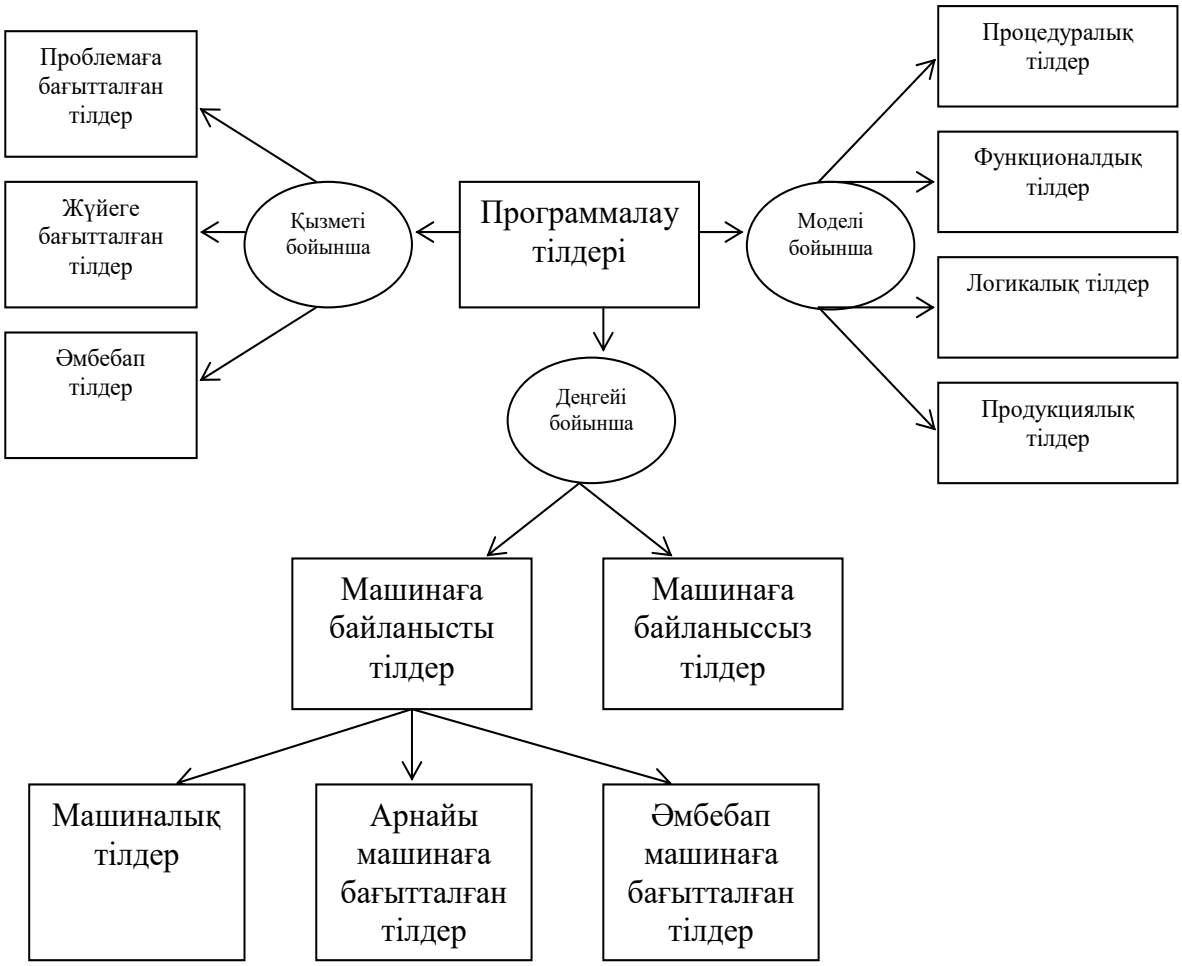
E) сызықты.

III.4. Программалау тілдері

Түйін сөздер: *программа, машиналық тіл, программалау тілі, транслятор, мнемокод, автокод, ассемблер тілі, ішпрограмма, макроамал, операциялық жүйе, басқарушы программа, қызметші программа, жұмысшы программа, тапсырмаларды басқару тілі, сұратым тілі, машинаға байланыссыз тіл, әмбебап тіл, проблемаға бағытталған тіл, процедуралық тіл, функционалдық тіл, логикалық тіл, продукциялық тіл, объектіге бағытталған тіл, сандық есептер, экономикалық есептер, символдық есептер, логикалық есептер, моделдеу есептері.*

Мақсат: *программалау тілдерінің жіктелуі, қасиеттері қарастырылады.*

Құрылымы: Программалау тілінің түрлері III.4.1-суретте берілген.



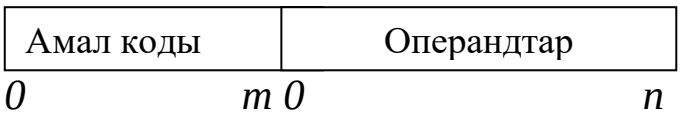
III.4.1- сурет. Программалау тілінің түрлері

III.4.1. Программа және программалау тілі ұғымы

Заманауи компьютерлерде ақпаратты өңдеуге қажетті деректер мен оларға қатысты алгоритмдер 0 мен 1-ден тұратын тізбелермен жазылуы керек екендігі белгілі. Ал кез келген алгоритмдік тілде жазылған алгоритмді және кез келген таңбалардың тізбектері арқылы берілген өңдеуге қажет деректерді мазмұндарын өзгертпей 0 мен 1-ден тұратын тізбектер арқылы кескіндеуге болады (бір ғана мазмұн әр түрлі хабар арқылы беріледі).

Алгоритмдегі *нұскаулар мен амалдарды* 0 мен 1-ден тұратын тізбелерден бейнеленуін *пәрмен* деп атайды.

Пәрмен құрылымы III.4.1-суреттегідей болады:



III.4.1-сурет. Пәрмен құрылымы.

Мұнда *амал коды* қандай да бір амалдың екілік коды болады (мысалы, қосу амалын былай $C1_{16} = 11000001_2$ кодтауға болады); *операндтар* – осы пәрменнің алғашқы деректері және нәтиже адрестері болады; *m*, *n* – компьютер жадында амал коды мен операндтарды орналастыруға қажет биттер санын сәйкес көрсететін бүтін сандар.

Әрбір пәрмен өзін компьютердің аппараттық жабдықтарымен орындалуын (өзінің жүзеге асырылуын) қолжетімді етуі керек. Компьютер орындай алатын пәрмендер жиынын *машиналық тіл* деп атайды, оның әліпбиі 0 мен 1-ден тұрады.

Алгоритмнің машиналық тілде бейнеленуін *программа* деп атаймыз. Осыдан әрбір компьютер өзінің тілінде жазылған кез келген программаны орындай алады.

Кез келген программа жазыла алатын формалды (бір мәнді) тілді *программалау тілі* деп атайды. Сондықтан машиналық тілді ең бірінші программалау тілі деп есептеуге болады.

Машиналық тілдегі барлық пәрмендер компьютердің құрамына кіретін функционалдық жабдықтар арқылы орындалады. Машиналық тілдің мүмкіншіліктері компьютердің алғашқы интеллектуалдық (зерделік) деңгейін анықтайды. Кейін неше түрлі интеллектуалды есептерді шешетін программаларды компьютердің жадына сақтай отырып, оның интеллектуалдық деңгейін көтеруге болады.

Машиналық тілде программаларды жазу адам үшін өте қиын екендігін ескерген жөн. Себебі, біріншіден, барлық амалдар мен нұсқаулардың екілік кодтарын және абсолют (физикалық) адрестерді есте сақтау керек, екіншіден, нұсқаулар мен амалдарды машиналық тілге аудару керек, үшіншіден, алғашқы деректер мен нәтижелерді екілік түрінде бейнелеу керек, төртіншіден, тармақталу мен қайталау нұсқауын жазу үшін басқару беретін адресті қолмен есептеу қажет. Сол сияқты бұрын істелген көптеген жұмыстарды басынан бастап қайтадан жасау керек. Мұның бәрі бірден көзге көрінбейтін көптеген қателер жіберуге соқтырады. Жазылған программалар оқуға және түсінуге қолайсыз болғандықтан, олардың ішіндегі жіберілген қателерді тауып түзету өте көп уақытты және шыдамдылықты қажет етеді. Сондықтан адамдар өздерінің жұмыстарын жеңілдету үшін программа жазуға ыңғайлы басқа жасанды тілдер ойластыра бастады.

Бірақ жасанды алгоритмдік тілдегі алгоритмді программаға айналдыру үшін оны машиналық тілге аудару керек. Осындай аударуды асыру барысында информатика ғылымында жаңа сала пайда болды. Ол жаңа формалды тілдерді, жасанды тілдердің синтаксисі мен семантикасын формалдау әдістерін, трансляциялау тәсілдерін жасаумен айналысады.

Аударуды «*транслятор*» деп аталатын программа орындайды. Сондықтан программаны жасанды тілді пайдаланып жазған кезде компьютердің жұмысы екі кезеңнен тұрады:

1. Алгоритмді жасанды тілден машиналық тілге трансляциялау;

2. Машиналық тілдегі программаны орындау.

Программаны орындаудағы осы екі кезеңде көрсетілген жұмысты жүзеге асырудың екі түрлі әдісі бар:

1. *Компиляция әдісі* – алдымен жасанды тілдегі программа түгелдей машиналық тілге аударылады, содан кейін бұл программа басынан басталып орындалады.

2. *Интерпретация әдісі* – жасанды тілдегі программаның әрбір әрекеті (нұсқауы немесе амалы) жеке алдын ала аударылмастан бірден машиналық тілде орындала бастайды.

Яғни, компиляция әдісінде жасанды тілдегі программа тек қана бір рет қарастырылады және аударылған программаны бірнеше рет орындауға болады, ал интерпретация әдісінде әр орындау алдында жасанды тілдегі программаны қайтадан қарастыру қажет. Сондықтан компиляциялау әдісі бойынша шыққан орындалуға дайын программа жылдам жұмыс істейді, бірақ программаны түгел орналастыру үшін жадтың көп орнын алады. Керісінше, интерпретациялау әдісі жадтың аз көлемін қажет етеді де, бірақ ақырын жұмыс істейді.

Сонымен осындай жасанды тілдер де компьютерге түсінікті болады, сондықтан оларды да біз программалау тілдері дейміз.

Қазіргі кезде компьютердің көмегімен әр түрлі есептерді шығаруға мүмкіндік беретін сан алуан программалау тілдері бар. Әр тілдің кемшілігі де, жетістігі де бар. Мысалы, кейбір тілде программаны жазу оңай болғанымен оны орындау өте көп уақытты талап етеді немесе жадтың көп орнын алады.

Сондықтан программалау үдерісіне кіріспен үшін, программалық өнімнің барлық өмірлік мезгілдері бойынша нормативті және жоба құжаттарының талатарын қанағаттандыратын, шығарылатын есепке сйкес келетін тілді мұқият таңдау керек.

III.4.1. Ескертпе:

Барлық программалау тілдерде айнымалылар, параметрлер, функциялар және басқа программалық бірліктер мен объектілердің атаулары кез келген әріптен немесе әріптен басталатып әріптер мен цифрлардан тұратын тізбек болатын «*идентификатор*» ұғым арқылы беріледі.

III.4.1. Мысадар

1. IBM/360 және ЕС ЭВМ компьютерлерінде амалдар кодының ұзындығы сегіз бит болды, яғни ол бір байтта жазылды.
2. Дүние жүзіндегі ең бірінші транслятор 1957 жылы АҚШ-та жасалды, ол 1954 жылы шыққан Fortran тілін машиналық тілге автоматты аударды.
3. Пәрмендегі операндтар саны компьютерлердің адрес санын білдіреді, мысалы, егер пәрмендердегі операнд екеу болса, онда компьютер екі адресті болады.
4. Мыналар NOM, ABC, A5, SIN, LOG, DIV идентификатор болады.
5. Мыналар 1NOM, A+B-C, NOT!, $100, |D|, (X+Y) идентификатор емес?

III.4.1. Тапсырмалар

1. IBM/360 және ЕС ЭВМ компьютерлерінде барлық мүмкін болатын пәрмендер санын анықтаңыз.
2. Компилятор мен интерпретатордың артықшылықтары мен кемшіліктерін көрсетіңіз.
3. Алгоритмді жазу үшін алгоритмдік тілдерді пайдаланғанда компьютердің жұмыс кезеңдерін көрсетіңіз.

Көмек:

1. Бір байтта қанша әртүрлі бір-бірінен бөлек ақпаратты орналастыруға болатындығын есептеу керек.
2. Уақыт мөлшері және жад көлемі бойынша салыстыру керек.
3. Машиналық тілге аударып, орындау керек.

III.4.1. Сұрақтар:

1. Транслятор дегеніміз не?

2. Программалау тілдерінің деңгейлері неге байланысты болады?

3. Жоғарғы деңгейлі қай программалау тілі ең бірінші жасалды?

III.4.1. Тесттер:

1. Программалау тілдерінің ішіндегі қай тілдің деңгейі ең төмен болады?

A) машиналық;
B) процедуралық;
C) функционалдық;
D) логикалық;
E) продукциялық.

2. Егер алдымен жасанды тілдегі программа түгелдей машиналық тілге аударылып, содан кейін бұл программа басынан басталып орындалатын болса, онда бұл қай тәсілмен жүзеге асырылған?

A) интерпретациялау;
B) конвертациялау;
C) модуляциялау;
D) компиляциялау;
E) интеграциялау.

3. Егер жасанды тілдегі программаның әрбір әрекеті (нұсқауы немесе амалы) жеке алдын ала аударылмастан бірден машиналық тілде орындала бастаса, онда бұл қай тәсілмен жүзеге асырылған?

A) интерпретациялау;
B) конвертациялау;
C) модуляциялау;
D) компиляциялау;
E) симуляциялау.

III.4.2. Программалау тілдерінің сыныпталуы

Жалпы, программалау тілдерін мынадай белгілері, қасиеттері бойынша жіктеуге болады:

1) Тілдің деңгейіне байланысты, яғни машиналық немесе табиғи тілге жақындығына байланысты;

2) Тілдің проблемаларға бағытталғанына байланысты, яғни белгілі сала есептерін оңай және ыңг

айлы шешу мүмкіндіктерінің болуына байланысты;

3) Тілдің моделіне байланысты, яғни программаларды құру стилін анықтайтын прінсіптеріне байланысты.

Ең төменгі деңгей машиналық тілдерде, ал ең жоғарғы деңгей табиғи тілдерде болады деп қабылданды. Сондықтан басқа тілдердің деңгейі деп олардың машиналық тілдерге немесе табиғи тілдерге жақындық өлшемін айтады. Бір жағынан, программалық тіл неғұрлым машиналық тілге жақын болса, соғұрлым оның деңгейі төмен болады және керісінше. Екінші жағынан, программалық тіл неғұрлым табиғи тілге жақын болса, соғұрлым оның деңгейі жоғары болады және керісінше. Сонымен, бірінші сыныптағы тілдер екі топтан тұрады:

– деңгейі төмен машинаға байланысты тілдер;

– деңгейі жоғары машинаға байланыссыз тілдер.

Деңгейі төмен машинаға байланысты тілдер компьютерлердің жеке маркаларының ерекшелігін немесе компьютерлер сыныптарының ерекшеліктерін қамтиды. Сондықтан олар тағы да екіге бөлінеді:

– машинаға бағытталған арнайы тілдер;

– машинаға бағытталған әмбебап тілдер.

Машинаға бағытталған арнайы тілдерге *мнемокодтар* мен *автокодтар* жатады.

Мнемокодтарда машиналық тілдердегі барлық пәрмендердің, деректердің және олардың адрестерінің екілік бейнелері (кодтары) *мнемоника* деп аталатын әріптер мен цифрлардан тұратын тізбектерге алмастырылып жазылады. Әдетте мнемоника ретінде

осы пәрмендердің табиғи тілдегі атауларының бас әріптерін алады. Мысалы, қосу (алу) амалының мнемоникасы ретінде амал атауынна сәйкес латын әліпбиінің үлкен әрпі ***A*** (***S***) алынады. Себебі ағылшын тілінде қосуды *Adition* (*Subtract*) деп атайды. Мнемокодтың машиналық тілге қатысы 1:1 болады, яғни әрбір машиналық тілдегі пәрменге бір мнемоника беріледі.

Автокодтардың мнемокодтардан айырмашылығы ол мнемоникаларды пайдаланумен қатар онда *ішпрограммаларды* ұйымдастыру және *макроамалдар* деп аталатын бір ғана амалдармен бірнеше қарапайым амалдардың тізбектерін атау мүмкіншіліктері бар. Мысалы, бір есепті шығарғанда программаның өзіндік логикалық бір тұтастығы бар кейбір бөлшек жиі кездессе, онда оны *ішпрограмма* немесе *макроанықтама* ретінде ұйымдастырып және қажеттілігі бар жерде оның атын және осы контекстке байланысты параметр мәнін көрсетіп қолдануға болады. Автокодтар мнемокодтардың біршама дамыған түрлері.

Жалпы, мнемокодтар мен автокодтарды біріктіріп *ассемблер тілі* деп атайды. Олай дейтіні, әрбір компьютерде осы тілді машиналық тілге аударатын программа бар. Сол программаны *ассемблер* деп атайды. Ассемблердің алғашқы деректері ретінде мнемокодта немесе автокодта жазылған программалар, ал оның нәтижесі ретінде осы программалардың машиналық тілдегі түрлері болады.

Машинаға бағытталған әмбебап тілдерде бір сыныпта жататын бірнеше компьютерлердің ерекшеліктерін қамтитын амалдар мен нұсқаулар болады. Бұлар машинаға бағытталған тілдердің дамығаны. Оларға, мысалы, *Алмо* немесе *Uncol* деген тілдер жатады. Кейін *C* және оның жетілдірілген түрі *C++* шықты, ал соңында Java тілі қосылды.

Бізге компьютердің жұмысын жоспарлау, ұйымдастыру және басқару үшін қатынас тілі қажет. Негізінде осы айтылғандардың барлығы белгілі программаларды орындау арқылы жүзеге асады. Оларды біріктіріп *операциялық жүйе* (ОЖ) деп атайды. ОЖ

программалары берілген есепті шығаратын *жұмысшы программа* орындалу кезінде оның алғашқы деректерін енгізу, сақтау және нәтижесін шығару сияқты қызметтер көрсетеді. Сонымен қатар, осы жұмысшы программаға қажетті компьютердің жабдықтарын алып береді, оның орындалу кезегін анықтайды, т.б. жұмыстар атқарады. Яғни, ОЖ программалары ішінде басқарушы және қызметші программалары бар. Қысқасы, компьютердегі программалар, былайша айтқанда «программалық қоғам» құрады. Бұл «қоғамдағы» программалар өзара қатынас жасауы үшін және олардың жұмысын алғаш бастап беру үшін немесе жұмыстың соңын белгілеу үшін жоғарыда айтылған қатынас тілі қолданылады. Әдетте, осындай қатынас тілдерін *операциялық жүйенің тілі* немесе *тапсырмаларды басқару тілі* деп атайды. Бұл тілдер, бір жағынан, операциялық жүйе белгілі компьютердің маркасына үйлескенде осы марканың ерекшеліктері мен қасиеттерін қамтитын болады. Сондықтан оларды машинаға байланыс тілдер класына жатқызуға болады. Ал екінші жағынан, кейбір операциялық жүйелер тілі машинаға байланыссыз болады (мысалы, *Unix* деген операциялық жүйеде).

Деңгейі жоғары, машинаға байланыссыз тілдер табиғи тілдерге жақын болады. Бұлар негізінен алгоритмді оңай және түсінікті жазу үшін арналып жасалынады. Оларда компьютерлердің ерекшеліктерін қамтитын нұсқаулар мен амалдар болмайды десе де дұрыс. Бірақ, бұл тілдерде жазылған программалар машинаға байланысты тілдерде жазылған программаларға қарағанда компьютердің көп ресурсын талап етеді. Мысалы, орындалуы көп уақыт және орналасуы көп жер алады.

Осы тілдер қандай есептерді шешуге ыңғайлылығына байланысты төмендегідей болып бөлінеді:

- сандық есептерге арналған тілдер;
- символдық есептерге арналған тілдер;
- логикалық есептерге арналған тілдер;
- экономикалық есептерге арналған тілдер;
- моделдеу есептеріне арналған тілдер, т.с.с.

Сандық есептерге негізінен ғылыми-техникалық есептер жатады. Әдетте, осы есептер үшін *Fotran, Алгол, Basic, Pascal, Ada* және т.б. пайдаланылады.

Символдық есептерге бір тілден екінші тілге аудару есептері, аналитикалық түрлендіру есептері сияқтылар жатады. Бұл есептер үшін *Снобол, Lisp, Refal* және т.б. қолданылады.

Логикалық есептерге теорема дәлелдеу, шахмат ойнау, программаны синтездеу, образдарды тану сияқты жасанды интеллекттің (зерденің) есептері жатады. Ол есептер үшін *Prolog, Лого* сияқты тілдер пайдаланылады.

Экономикалық есептерді шығару үшін, мысалы, *Cobol, РПГ* сияқты тілдер пайдаланылады. Бұл тілдерде неше түрлі кестелерді құрастыру үшін және оларды түрлендіру үшін амалдар мен нұсқаулар бар. Қазіргі кезде мұндай есептер үшін бүтін бір программалық жүйелер жасалынды. Олардың қатарына электрондық кестелер деп аталатын Lotus, Excel сияқтылар жатады. Сонымен қатар, бұл есептерді шығару үшін 1970–жылдардың орталарынан бастап пайда бола бостаған *деректер базасын басқаратын жүйе* деген арнаулы программалық жабдықтамалар қолданыла бастады. Оларда өзіндік сұрату тілі деп аталатын тілдер болады. Ол тілдер пайдалануға ыңғайлы және түсінуге қарапайым болады. Осындай жүйелерге мысалы, *dBase, Foxpro, Access, My SQL, Oracle, Informix* және тағы басқаларды жатқызуға болады.

Моделдеу есептерін шешуге арналған тілдерге *Simula, Dynamo* сияқты тілдерді жатқызуға болады. Мұндай есептерге имитациялық моделдеу әдісімен шығатын есептер жатады, мысалы, ауа райын болжау. Бұл тілдерде моделдеу жабдықтары жүйені бір–бірімен әрекеттесіп отыратын үдерістерді программаны трансляциялау кезінде анықтау көзделген.

Жалпы, осы айтылған тілдердің барлығын *проблемаға бағытталған тілдер* деп те атауға болады.

Жоғарыда келтірілген есептердің барлығын шығаруға мүмкіндік беретін тілдерді әмбебап тілдер деп атайды. Мысалы, ондай тілге *PL/I, Algol-68, Ada* тілдерін жатқызуға болады. Бұл

тілдерде шамалардың барлық типтері және оларға орындалатын амалдар анықталған. Әдетте, бұл тіл арқылы кез келген есепті шығаруға болғанымен ондағы программаның көлемі өте үлкен болады және олар шабан жұмыс істейді. Сондықтан ол компьютердің көп ресурсын қажет етеді.

Программалау тілдері, табиғи тілдері сияқты, өздерінің моделі (құрылу негізі) бойынша бөлінеді. Мысалы, табиғи тілдерді біз *түркі тілдер*і, *славян тілдері*, *роман тілдері*, *араб тілдері*, *парсы тілдері* және т.с.с. деп топқа бөлеміз. Ал программалау тілдерін процедуралық тілдер, функционалдық тілдер, логикалық тілдер және продукциялық (алмастырымдық) тілдер деп бөлуге болады.

Программалау тілдердің ішінде ең алғаш пайда болғандары және ең көп тарағандары процедуралық тілдер. Оларға *Fotran*, *Алгол*, *Basic*, *Pascal*, *Ada* сияқтылар жатады.

Жалпы, процедуралық тілде алгоритм әрекеттерді және оларды орындау үшін қажет нұсқауларды бейнелеу арқылы жазылады. Процедуралық тілдердегі программа бірлік *оператор* деп аталынады. Ең кіші программалық бірлік болып *меншіктеу операторы* есептелінеді. Оны, жалпы түрде, былай бейнелеуге болады:

<айнымалы> ← <өрнек>,

мұнда "←" – меншіктеу амалының белгісі, ол әр программалау тілінде әр түрлі белгімен берілуі мүмкін, мысалы, "=" – *Fotran* және *Basic* тілдерінде, ":=" – *Алгол* және *Pascal* тілінде. Меншіктеу операторының орындалуы екі кезеңде өтеді: алдымен өрнектің мәні есептелінеді; содан кейін осы есептелінген мән айнымалыға меншіктеледі.

Әрбір программалау тілінде кездесетін *деректерді енгізу операторы* және *деректерді шығару операторы* меншіктеу операторының дербес жағдайлары болады: деректерді енгізген кезде айнымалы ретінде компьютер оперативтік жадындағы адрес, ал өрнек ретінде енгізілетін деректер терілетін немесе орналасқан сыртқы құрылғының аты немесе сыртқы жадтағы адрес алынады: деректерді шығарған керісінше болады, айнымалы – сыртқы

құрылғы немесе сыртқы жад адресі, өрнек – оперативтік жад адрестері арқылы құрылады.

Процедуралық тілдерінде басқа программалық бірліктерге *тізбектеу операторы*, *тармақталу операторлары* және *қайталау операторлары* сияқтылар жатады.

Функционалдық программалау тілдері лямбда есептеуіне негізделеді, онда шешілетін есептің алгоритмі функцияларды бейнелеу арқылы жазылады. Функциялар қарапайым функция немесе күрделі функция болуы мүмкін. Күрделі функция қарапайым функциялардың композициясынан (суперпозициясынан) тұрады. Функционалдық тілдерде программа бірлігі ретінде функция қабылданады. Осы тілдердің мысалы ретінде *Lisp*, *Плэнер*, *Conniver*, *KRL*, *FRL* және *FP* сияқты тілдерді алуға болады.

Логикалық программалау тілдері логикалық есептеуге негізделген, онда алгоритм шығарылатын есептің алғашқы деректерінің және оларда орындалатын амалдардың қасиеттері мен қатынастары туралы құрылған тұжырымдарды бейнелеу арқылы жазылады. Әрбір тұжырымды логикалық ереже деп қарастыруға болады және әрбір ереже белгілі бір шартты береді. Есептің нәтижесі сұратым арқылы анықталады. Егер осы сұратым бойынша логикалық ережелердің орындалғанын дәлелдесек, онда есептің нәтижесінің болғаны, әйтпесе не басқа сұратым қою керек, не қойған сұратымға теріс жауап алғанымыз. Осыны жүзеге асыру үшін осы тілде белгілі бір формальды дедукциялық жүйені қолданады. Мұндай тілдерге мысал ретінде *Prolog* және *Logo* сияқтыларды алуға болады.

Продукциялық программалау тілдері нормалды алгорифмдерге негізделген, онда шешілетін есептің алгоритмі продукциялық ережелерін (алмастырымдарды) кескіндеу арқылы жазылады. Алмастырымдар «сол жағы» және «оң жағы» деп аталатын екі бөліктен тұрады. Алмастырымның сол жағында осы алмастырымның атауы және алғашқы деректердің үлгілері

орналасады, ал оң жағында үлгілерге сәйкес келген алғашқы деректермен не істеу керектігі көрсетіледі. Бұл үшін басқа алмастырымдарды шақыруға да болады. Ал алмастырымдарды құру «жоғарыдан төмен» алдымен берілген есептің жалпы мазмұнына сәйкес алмастырым құрылады. Содан кейін осы есепті шешу кезінде пайда болған жаңа ұғымдарға сәйкес алмастырымдар құрылады. Яғни бұл тілдегі программаның құрылымы иерархиялық (дарақ тәріздес) болады. Оның кез келген деңгейінде тексеруге немесе өзгерістер енгізуге болады. Осындай тілдердің мысалы ретінде *Snobol* және *Refal* тілдерін алуға болады.

Объектілі-бағытталған программалау тілдері (ОБПТ) деректердің абстрактілі типтерінің теориясына негізделген, шешілетін есептің алгоритмі деректер жиынынан және оларға орындалатын операциялардан тұратын деректер типтерін құру арқылы құрылады. Басында есептің жалпы болмысына сәйкес келетін глобалды тип (суперсынып) құрылады. Қойылған есепті шешу кезінде пайда болатын барлық басқа болмыстар глобалды типтің ішкі типі (ішкі класы) немесе объектісі (нақты кластың экземпляры) болып табылады.

ОБПТ *деректерді абстрактциялау*, *инкапсуляция*, *мұрагерлік*, *полиморфизм*, «кеш байланыстыру» принциптерін қолдайды.

Абстракциялау (Abstracting) – объектінің маңызды ғана сипаттамалар жиынын бөлуге мүмкіндік беретін принцип. Сәйкесінше, абстракция – осындай барлық сипаттамалар жиыны.

Инкапсуляция (*Encapsulation*) – деректер мен кластың ішінде жұмыс жасайтын әдістерді (код) біріктіруге және қолданушыдан жүзеге асырылуын жасыруға (деректерге тікелей сырттан қолжетімділік мен дұрыс емес қолданудан қорғау) мүмкіндік беретін принцип, бұл сынып деректеріне қолжетімділік тек сол кластың әдістері арқылы жүзеге асырылуды қамтамасыз етеді.

Инкапсуляция программа бөліктеріне өзгертулерді басқа бөліктеріне әсер етпейтіндей енгізуге мүмкіндік береді, бұл программалық қамтаманы қолдау және өзгертуді айтарлықтай жеңілдетеді.

Мұрагерлік (*Inheritance*) – жаңа класты бұрыннан бар сынып негізінде және оның функционалдылықтарының бір бөлігін немесе толық қайталап сипаттауға мүмкіндік беретін принцип. Нақтылағанда, сынып (объект) басқа кластың негізгі қасиеттерін мұра етіп, өзіне ғана тән қасиеттер мен әдістер қоса алады.

Мұрагерлік тарайтын сынып негізгі, аталық немесе суперсынып деп аталады. Жаңа сынып – ұрпақ, мұрагер немесе туынды сынып. Бір сыныпта басқа бірнеше кластың мүмкіндіктерін біріктіруге болады.

Мұрагерлік екі түрі бар:

Оңаша – сынып (сонымен қатар ішкі сынып болып табылады) бір ғана суперкласы бар;

Көптік – кластың тегі саны кез келген бола алады (Java тілінде рұқсат етілмеген).

Полиморфизм – екі немесе одан да көп ұқсас, бірақ айырмашылықтары бар есептерді шешу үшін интерфейстері бірдей объектілерді типі мен ішкі құрылымы туралы ақпаратсыз-ақ қолдануға мүмкіндік беретін принцип, яғни «бір интерфейс, көптеген әдістер » идеясы жүзеге асырылады.

Полиморфизм кезінде аталық кластың кейбір бөліктері (әдістер) берілген ұрпаққа тән әрекеттер жасайтын жаңа бөліктермен ауыстырылады. Осылайша, кластар интерфейсі бұрынғы болып қалады, ал атаулары және параметрлер жиыны бірдей әдістерді жүзеге асыру айрықша болып келеді. Полиморфизммен кеш байланыстыру тығыз байланысқан.

Кеш байланыстыру виртуалды функциялар мен туынды кластар көмегімен жүзеге асады және объекті тек программа орындалу кезінде ғана функция шақыруымен байланысады дегенді білдіреді. Оның артықшылығы жоғары иілгіштігі. Ол ортақ интерфейсті қолдау кезінде қолданыла алады және әртүрлі

объектілерде бұл интерфейстің өзіндік жеке жүзеге асырылуына мүмкіндік береді. Сонымен қатар, қайта қолдану мекн кеңейтуге рұқсат ететін кластар кітапханасын құруға көмектеседі. Кеш байланыстыруды қолдану программа құрылымы пен басқаруын жақсартатын жағдайда ғана орынды.

ОБПТ келесідей элементтер жиынтығынан тұрады:

- өрістер кластарын өрістерімен (деректерімен - сынып мүшелерімен) және әдістерін (функцияларымен- сынып мүшелерімен) жариялау;
- сыныпты кеңейту механизмі (мұрагерлік) - аталық кластың жүзеге асу ерекшеліктерін мұрагер сынып құрамына кіретін автоматты қосып, бар кластан жаңа сынып жасау;
- әртүрлі кластың экземплярларын бір айнымалыға меншіктеуге мүмкіндік беретін полиморфты айнымалылар мен функциялардың (әдістердің) параметрлері;
- виртуалды әдістерді қолданғанда кластар экземплярларының полиморфты іс ірекеті.

Объектіге бағытталған программалау тілдеріне қолданғанда сынып және объект түсініктері нақтыланады:

Сынып деректер сипаты мен оларда анықталған амалдардан тұрады. Класта туысқан, нақты бар болатын объектілердің кейбір жиынтығының жалпылама сипаты беріледі.

Объект - компьютер жадында физикалық орналасқан атаулары бар деректер (объект өрістері) және оларға қолжетімі бар әдістер жиынтығы. Атау объектіні құрайтын өрістер мен әдістерге қол жетекізу үшін пайдаланылады. Ақтық жағдайда объект өрісті немесе әдісті, сонымен қатар, атауды қамтымауы мүмкін.

Кез келген объект белгілі бір класқа жатады. Объект кластың нақты экземпляры болып табылады.

Мұрагерлік пен полиморфизмнің маңыздылығында келесі парадигма тұр: «суперсынып *объектісі қолданылатын жердің барлығында ішкластар объектілері қолданылуы мүмкін*».

Класты тәсілін шақырғанда, ол кластың өзінен ізделінеді. Егер әдіс бар болса, онда ол шақырылады. Егер ағымдағы класта әдіс

болмаса, онда аталық кластан ізделіне бастайды. Егер іздеу сәтсіз болса, онда ол иерархиялық ағаш бойынша жоғары иерархияның тамырына (жоғарғы класына) дейін жалғастырады.

Кейбір ОБПТ көрсетілген минималды жинаққа әртүрлі қосымша жабдықтарды қосты. Олардың санына:

конструкторлар, деструкторлар, финализаторлар.

- қасиеттер (аксессорлар);
- индексаторлар;
- сынып компоненттер көрерлерін басқару жабдықтары (интерфейстер немесе public, private, protected, feature сияқты қолжетімді жетілдіруіштері).

Кейбір тілдер ОБП прінсіптеріне толық қанды жауап береді – оларда барлық негізгі элементтер күй және байланысқан әдістері бар объектілер болады. Бұл тілдерге *Smalltalk, Eiffel, Ada* жатады. Объектік ішжүйені бүтіндей түрде басқа «екі және одан көп тілдер бір тілде» парадигма ішжүйелерімен қиыстырып, бір программада объектік моделдерді өзгелермен қиыстыруға, объекті-бағытталған программалау және басқа парадигмалар арсындағы айырмашылықты шаюға мүмкіндік беретін тілдер бар. Бұл тілдерге *CLOS, Dylan, Python, Ruby, Objective-C* жатады. Бірақ, объектік моделді эмуляциялау (бопсалау) жабдықтарын императивтік семантика үстіне қосатын тілдер көп тараған. Бұл қандай да бір парадигманы қамтитын «таза стиль» тілдерге қарсы «желімдеу» деп аталады. Осындай тілдерге *Visual Prolog, Visual Basic, Object Pascal, Modula,* C++, C#, *Java жатады.*

Объекті-бағытталған тілдер процедуралық және функционалдық тілдерге тән мүмкіншіліктерді біріктіреді және кеңейтеді. Олар объекті-бағытталған тәсілдің артықтышылығын программалық жүйелерді тек жобалау мен құрастыру кезінде ғана емес, оларды жүзеге асыру, тестілеу және сүйемелдеу кезінде де пайдалануға мүмкіндік береді. Осындай тілдерге ең алдымен *Simula, Smoltok* және *Ada* жатады. Кейінірек объектіге бағытталған концепция басқа да тілдерде жүзеге аса басады. Оларға *Visual Prolog, Visual Basic, Object Pascal* және C++, C# жатады.

III.4.2. Ескертпе:

Өзіңіздің есебіңізді шешу үшін программалау тілін таңдаған кезде программа құруды басқаратын екі жол (парадигма) бар екендігіне көңіл аудару керек. Бірінші жол үдеріске бағытталған модел деп аталады, оны деректерге кодтық әсер ету (code acting on data) деп айтуға болады. Үдеріске бағытталған программалау программаның жазылуы «осы үдеріс не нәрсеге әсер етеді» дегеннің айналасында жүрсін деп қалайды. Бұл жолда программаның көлемі мен күрделілігі өскен кезде проблемалар пайда болады. Екінші жол объектіге бағытталған модел деп аталады, оны деректермен басқарылатын кодқа қолжетім (data controlling access to code) деп сипаттауға болады. Объектіге бағытталған программалау программаны өзінің деректері (объектілері) және осы деректермен жақсы анықталған интерфейстер төңірегінде ұйымдастырады. Бұл жолда қандай да бір ұтымдылық алуға болады.

III.4.2. Мысалдар:

1. Simula программалау тілі моделдеу және процедуралық тілдер тобына жатады.
2. Prolog программалау тілі логикалық және проблемаға бағытталған тілдер тобына бірдей жатады.
3. Lisp программалау тілі функционалдық және жоғары деңгейлі тілдер тобына бірдей жатады.

III.4.2. Тапсырмалар:

1. Процедуралық тілдердегі меншіктеу операторының ***<айнымалы>*** ← ***<өрнек>*** семантикасын сипаттаңыз.
2. Программалаудағы замануи парадигмаларды атаңыз.
3. D ← B*B - 4*A*C өрнегі бойынша D-ның мәнін табатын кез келген тілде алгоритм жазыңыз.

Көмек:

1. Ол екі кезеңнен тұрады: алдымен өрнектің мәнін есептеу, кейін осы мәнді меншіктеу.

2. Бұл парадигмалардағы моделдерге, үдерістерге және объектілерге көңіл беру керек.

3. Айнымалылардың мәндерін енгізіп, меншіктеуді құру керек.

III.4.2. Сұрақтар:

1. Қазіргі кезде программалау тілдерін трансляциялаулардың қандай түрлері жүзеге асқан?

2. Объектіге бағытталған программалаудың негізгі прінсіптерін нелер құрайды?

3. Программалау тілдерінің қайсыларында алгоритмді кіріктірілген абстрактылы типтер түрінде бейнелеу жүзеге асырылған?

III.4.2. Тесттер:

1. Prolog тілінде программалау бірлігі не?

A) тұжырым;
B) функция;
C) оператор;
D) амал;
E) қатынас.

2. Lisp тілінде программалау бірлігі не?

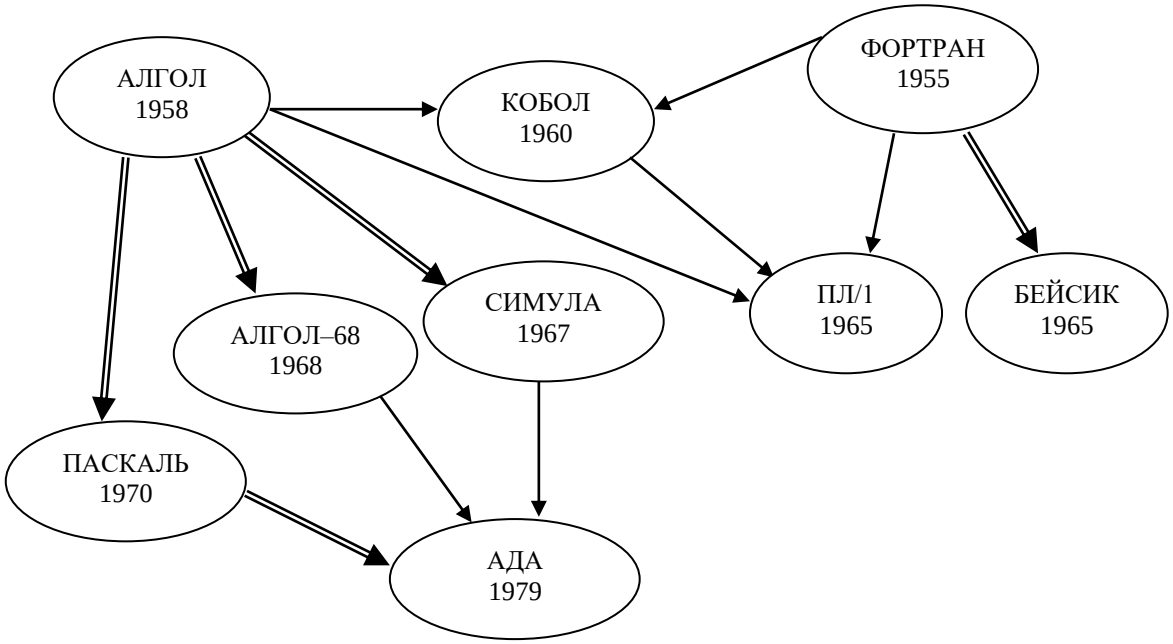
A) қатынас;
B) функция;
C) оператор;
D) тұжырым;
E) алмастырым.

3. Refal тілінде программалау бірлігі не?

A) оператор;
B) функция;
C) алмастырым;
D) тұжырым;
E) қатынас.

III.4.3. Процедуралық программалау тілдері

Қазіргі кезде мыңдаған процедуралық программалау тілдері бар. Бірақ бізді қызықтыратыны тек кең тараған классикалық процедуралық тілдер ғана, олардың даму тарихы III.4.3-суретте көсетілген.



III.4.2-сурет. Процедуралық программалау тілдерінің даму тарихы

Мұнда қалың бағыттамамен көп әсер еткендікті, ал жіңішке бағыттамамен аз әсер еткендікті белгіленген.

Процедуралық тілдер жөнінде функционалдық программалаудың негізін салушы американ оқымыстысы Дж.Бэкус өзінің Тьюринг сыйлығының (*информатика ғылымы бойынша ең жоғарғы сыйлық*) лауреаты болуына байланысты лекциясында *Ada* тілі процедуралық тілдердің даму шегі, бұдан ары қарай процедуралық тілдер дами алмайды, себебі *Ada* тілінде барлық теориялық сұрақтардың шешімдері орын тапқан» деп дәлелдейді (*Ada* тілі 1975 жылы АҚШ–ң Қорғаныс министрлігінің жариялаған конкурсты жеңіп алған Жан Ишбианың басқаруымен француз

ғалымдары 1979 жылы жасаған тіл. Бұл тілдің аты дүние жүзіндегі ең бірінші программалаушы әйел, лорд Байронның қызы, Чарльз Бэббидждің қызметкері ада Лавлейс есімімен аталған).

Процедуралық тілдердің ең көп тараған және оқуға қолайлы өкілі *Pascal* тілін Швейцария ғалымы Н. Вирт программалау әдістерін оқыту мақсатында 1970 жылы жасаған [Вирт, 1997]. Ол тілдің атына француяның ұлы ғалымы Б.Pascal есімін берді.

Төменде *Pascal* тілінің қысқаша түрі беріледі. *Pascal* тілінің қысқаша үлгісіндегі дұрыс құрылған программа, осы тілдің толық үлгісінде де дұрыс болады. Сондықтан бұл кітаптағы программаның мысалдары, соның ішінде сұрыптау алгоритмдері, осы *Pascal* тілінің қысқаша үлгісінде жазылады.

Pascal тіліндегі программа символдардың (латын әріптері, орыс әріптері, араб цифрлары, тыныс белгілері, амалдардың таңбалары, кетік таңбасы және жақшалар) тізбесі түрінде жазылады және әрбір сөйлемнің соңына «;» таңбасы қойылады.

Алғашқы деректермен нәтижелерді белгілеу үшін тұрақтылар мен айнымалылар қолданылады.

Тұрақты сандық шамалар ондық санау жүйесінде берілген бүтін, нақты сандар түрінде, ал тұрақты символдық шама қос дәйекшенің (деректерді шығарған кезде дәйекше көрінбейді және басылмайды) ішіндегі таңбалардың тізбегі түрінде беріледі.

Программада айнымалыларды жариялау кезінде *var* деген арнайы сөзді және олардың типтерін бейнелеу үшін тілдің арнайы сөздері қолданылады:

сандық тип: *integer* – бүтін сан, *real* – нақты сан;

символдық тип: *char* –таңбалар тізбегі.

Мысалы, айнымалылар *i* – бүтін сан; *a,b* – нақты сан; *x,y* – таңбалар тізбегі болсын десек, онда олар *Pascal* тілінде былай жазылады:

var *i, j* : *integer;*

var (*x, y*) : *real;*

var (*a, b*) : *char;*

Программалауда тип дегенімізді кейбір амалдар анықталған жиын деп түсініп, ал жиын элементтерін осы типтің мәндері деп қарастыру керек. Осы тұрғыдан, мысалы, типтер *integer* мен *real* – арифметикалық амалдар және салыстыру амалдары анықталған сандық жиындар, ал *char* – тіркеу амалы және салыстыру амалдары анықталған символдар жиыны. Бұл типтерді *стандартты типтер* дейді, себебі олар көптеген программалау тілдерінде анықталған.

Pascal тілінде бар типтен жаңа *стандартты емес типтер* алуға болады. Сондай мүмкіншіліктердің біреуін қарастырайық. Математикада кейбір жиынмен қатар осы жиынның сұрыпталған қосақтар жиынын, сұрыпталған үштіктер жиынын және т.с.с. жиі қарастырады. Pascal тілінде сұрыпталған қосақтар, үштіктер және т.с.с. *жиым* деп аталатын әрқайсысының *индексі* бар *бір типті элементтер жиыны* түрінде беріледі. Элементтер типі және индексті беру әдісі жиым жататын типтің анықтамасында беріледі. Мысалы, мынадай өрнек

t = **array**[1.20] **of** *real*

атауы *t* болатын, элементтері 20 қосақтан тұратын (индекс мәнінің өзгеру аралығы 1-ден 20–ға шейін) және типі *real* жаңа типтің

анықтамасы. Программада мұндай анықтаманың алдына арнау сөз **type** жазылады. Мысалы, айнымалы *a* программада *t* типті айнымалы ретінде сипатталса:

var *a* : *t*;

программа орындаған кезде *a* айнымалысының мәні типтері *real* 20 элементтен тұратын жиым болады, яғни, оларды *a*[1], *a*[2], …, *a*[20] деп жазуға болады. Мұндағы *a* айнымалысының типі *t*, ал *a*[1], *a*[2], …, *a*[20] айнымалыларының типі *real* болатынын және *t* объектісінде анықталған амал – ол индекстер арқылы әрбір элементтерге қол жеткізу, ал оның әрбір элементінде анықталған амал – ол *real* типінде анықталған сандық амалдар екенін ұмытпаған жөн. Сонымен айтылғанды программада толықтап мынадай түрде жазуға болады:

type *t*= **array**[1.20] **of** *real*;

var *a* : *t*;

Жалпы түрде жиым мына түрде анықталады:

u= **array**[n_1.n_2] **of** *r*;

мұнда *u* – жаңа типтің атауы, бүтін сан n_1 – индекс мәнінің төменгі шекарасы, бүтін сан n_2 – индекс мәнінің жоғарғы шекарасы және $n_1 \leq n_2$, ал *r* – стандартты *integer*, *real*, *char* деген типтердің біреуі немесе басқа бұрын анықталған типтің атауы.

Типтердің барлық анықтамаларының жиыны былай жазылады:

type T_1, T_2, …, T_m

мұнда T_1, T_2, …, T_m – жеке типтердің анықтамалары. Бұл жиын программада айнымалылар жариялауларының алдында жазылады.

Көп жағдайда берілген есепті шешетін программа құру үшін аралас типтер қажет болады: бір элементі символдық, екінші элементі сандық және т.б. Мысалы, оқушы деген типті анықтап оның элементтері мынадай бола алады деуге болады: оқушының *аты* – символдық, *жасы* – нақты сан, *бағасы* – бүтін сан және т.с.с.

Pascal тілінде аралас тип *жазба* арқылы беріледі. Жазбаның элементтерін *өріс* (бағана) деп атайды. Жазбадағы әрбір өрістің жеке атауы бар болады. Мысалы, егер өріс атаулары ретінде *аты, жасы, бағасы* дегендер таңдалса және егер *x* арқылы жазбаны белгілесек, онда осы жазбаның өрістерін *x.аты, x.жасы, x.бағасы* арқылы бейнелейміз. Жазбадағы өрістер саны, олардың типтері және атаулары жазбаға қатысты типтің анықтамасында жазылады. Мысалы, былай жазуға болады:

оқушы = **record** *аты* : *char*; *жасы* : *real*; *бағасы*: *integer* **end**

бұл элементтері үш өрісті жазба болатын *оқушы* деген типтің анықтамасы және өрістерінің атаулары *аты*, *жасы*, *бағасы*, ал олардың типтері сәйкес *char, real*, *integer* болады.

Егер программада *v* аралас типі **record** .. **end** құрылымымен анықталған болса, онда ол жалпы түрде былай жазылады:

$$v = \textbf{record}\ \ l_1 : r_1;\ l_2 : r_2;\ \ldots;\ l_k : r_k\ \textbf{end}$$

мұнда $l_1, l_2, \ldots, l_k$ – өрістердің атаулары, ал әрбір $r_1, r_2, \ldots, r_k$ – *integer*, *real*, *char* стандартты немесе басқа бұрын анықталған типтердің атаулары.

Меншіктеу амалы «**:=**» таңбасы арқылы белгіленеді, мысалы, $i := 0$; $x := 2.71$; *a*:= ‘алғашқы шарт’ .

Енгізу операторы мен *шығару операторы* үшін *read* және *write* деген арнайы сөздер сәйкес қолданылады. Мысалы, *u, v* және *w* айнымалылардың мәндерін енгізіп, *r, s t* айнымалылардың мәндерін шығару керек болса, онда оны былай жазуға болады:

read(*u,v, w*)*;*

write(*r, s, t*)*;*

Тізбектеу **begin** және **end** деген арнайы сөздердің арасында жазылады, мысалы, **begin** *j* := 1; *y* := 3.14; *b*:= ‘соңғы шарт’ **end**

Егер тармақталуға қажет шартты *B* арқылы, ал әр тармақтағы әрекетті *S1* және *S2* арқылы белгілесек, онда тармақталу мынадай:

if *B* **then** *S1* **else** *S2*

Мысалы, егер *max* айнымалысына *x1* және *x2* айнымалыларының үлкен мәнін меншіктеу керек болса, онда оны былай жазуға болады:

if *x1>x2* **then** *max* := *x1* **else** *max* := *x2*

Алғы шартты қайталау мына түрде жазылады:

while *B* **do** *S*

мұнда *B* – қайталау шарты, *S* – қайталау денесі.

Соңғы шартты қайталауға жататын параметрлік қайталаудың жазылуы мынадай болады:

for *i* := *K* **to** *N* **do** *S*

мұнда *i* – қайталау параметрі (бүтін типті айнымалы), *K* және *N* – қайталау параметрінің алғашқы мәні және соңғы мәні (бүтін типті айнымалылар), *K*≤*N*, *S* – қайталау денесі.

Параметрлік қайталаудың тағы да бір түрі бар:

for *i* := *K* **downto** *N* **do** *S*

мұнда $K \geq N$, сондықтан і біртіндеп $K, K–1, K–2, .., N$ мәнедерін қабылдап олардың әрқайсысы үшін *S* қайталау денесін орындайды, ал егер $K < N$ болса, онда *S* бір рет те орындалмайды.

Pascal тіліндегі программаның басын белгілеу үшін арнайы program деген сөз қолданылады. Егер программаға параметр қажет болса, онда осы сөзден соң жазылатын программаның атауынан кейін жақшаның ішіне бір–бірінен үтірмен айырылған айнымалылар тізімі жазылуы мүмкін. Мысалы,

program *аудан* (*input*, *output*) ;

мұнда *input* - *программа*да енгізу операторы кездесетінін, ал *output* - *программа*да шығару операторы кездесетінін білдіреді.

Программалау кезінде операторлардың бір ғана тізбегін бірнеше рет жазуға тура келеді. Егер осындай тізбекке қандай–да бір әдіспен атау беріп және оның шамаларының программа контекстісіне байланысты өзгеруі мүмкін мәндерін параметр арқылы бейнелесек, онда оны бірақ рет анықтап, көп рет оның қызметі керек жерде шақырып, програманы ықшамдап құруға болады. Бұл екі программалық жабдықтармен жүзеге асырылады:

1) *Процедура* – қандай–да бір әрекет жасайтын атау берілген операторлардың тізбегі арқылы;

2) *Функция* – қандай–да бір мәнді есептейтін атау берілген операторлардың тізбегі арқылы.

Операторлардың тізбегіне атау беру үшін программаға процедураның сипаттамасын қосу керек. Мысалы, егер S операторлардың тізбегі үшін P деген процедураның атауын программаға ендіру керек болса, онда сипаттаманың түрі мынадай:

procedure *P; S*

Параметрлі процедураны былай анықтауға болады:

procedure < атау> (<параметрлер тізімі>);

мұнда <параметрлер тізімі> – бір немесе бірнеше, бір–бірінен үтірмен айырылған параметрлер (айнымалылар).

Параметрмен бірге олардың типтері көрсетілуі мүмкін: егер параметр типі көрсетілсе, онда олардың арасына қос нүкте жазылады, мысалы, n : *integer*, u : *real* .

Функциялардың сипаттамасы **function** деген сөзден басталады, сонан кейін функцияның атауы және дөңгелек жақшаның ішіне оның параметрлер тізімі жазылады, ал соңында осы функция мәнінің типі көрсетіледі. Мысалы,

function *f1* (*k*: *integer; l* : *integer; m* : *integer*;) : *integer*;

function *f2* (*a*: *real*; *b* : *real*; *c* : *real*;) : *real*;

function *f3* (var *x, y*: *real*; *n* : *integer*;) : *real*;

Функциялардың шақыруы (атауы мен параметрлері–аргументтері) меншіктеу операторының оң жағында көрсетіледі. Сонымен қатар, функциялар қатынас амалдарында да кездеседі.

III.4.1. Ескертпе

Процедура мен функциялар анықталған кездегі параметрлерді (айнымалыларды) *формалды парамертрлер* деп, ал олардың шақыруларында көрсетілген параметрлерді (нақты мәндерді) *нақты параметрлер* деп атайды.

Pascal тілінде қолдануға дайын стандартты функциялар бар. Төмендегі III.4.2-кестеде осындай функциялар келтірілген.

№	Pascal тіліндегі түрі	Атауы
1	*abs*(*e*)	Абсолюттік шама
2	*arctan*(*e*)	Арктангенс
3	*cos*(*e*)	Косинус
4	*exp*(*e*)	Көрсеткіштік функция
5	*ln*(*e*)	Натурал логарифм
9	*mod* (*a*,*b*)	Бүтінді бүтінге бөлгендегі қалдық
6	*sin*(*e*)	Синус
7	*sqr*(*e*)	Квадратқа шығару
8	*sqrt*(*e*)	Квадрат түбір

III.4.2- кесте. Стандартты функциялар

Енді әртүрлі программалар құруға болады.

III.4.3. Мысалдар:

1. Үшбұрыштың ауданын есептейтін программаны мына фомула бойынша $p = \frac{a+b+c}{2}$; $s = \sqrt{p(p-a)(p-b)(p-c)}$ құрыңыз. Мұнда *a, b, c* - үшбұрыштың қабырғалары, *p* – периметрдің жартысы, *s* – ауданы.

Бізге қажет Pascal тіліндегі программа мынадай болады:

```
program аудан (input, output) ;
    var  a, b, c, p, s  : real;
        begin read (a, b, c);
            p:=(a+ b+ c)/2; s=sqrt(p*(p–a) *(p–b) *(p–a));
            write (s)
        end.
```

2. Берілген оң бүтін санның факториалының мәнін функцияны мына формула бойынша құрыңыз:

$$0! = 1$$

$$n! = n*(n–1)!$$

Сонда осы ереже Pascal тілінде былай жазылады:

```
function  fact (n:integer); integer;
    begin
        if  n = o then  fact: =1
                    elce fact: = n * fact (n–1)
    end.
```

III.4.3. Тапсырмалар:

1. Үшбырыштың берілген қабырғалары *a, b, c* бойынша оның биіктігін табу керек.

2. Мәтіндік файлда постфиксті пішінде жазылған өрнектің мәнін есептейтін программаны құрыңыз. Файлдың әрбір жолы тек бір ғана өрнекті қамтиды.

Көмек

1. Герон формуласын пайдаланыңыз.

2. Өрнек солдан оңға қарағанда сан кездессе, ол стекке жазылады; егер амал таңбасы кездессе, онда стектен соңғы екі элемент алынады, оларға амал қолданып, нәтижесі стекке жазылады. Стектің соңында тек жалғыз сан қалады, ол өрнек мәні.

III.4.3. Сұрақтар:

1. Pascal тілінде айнымалылар қалай сипатталады?
2. Pascal тілінде функциялар қалай сипатталады?
3. Pascal тілінде қандай стандартты функциялар бар?

III.4.3.Тестер:

1. Мына өрнек $\frac{e^x + a*b}{Cos^2 x + Sinx}$ Pascal тіліндегі қалай жазылады?

   A) (EXP(X)+A*B)/ (COS(X)*COS(X)+SIN(X))

   B) EXP^X+A*B)/ (COS(X)*COS(X)+SIN(X))

   C) (EXP(1)+A*B)/ (SQR(COS(X))+SIN(X))

   D) EXP(X)+A*B)/ (SQRT(COS(X))+SIN(X))

   E) (EXP(1)+A*B)/ (SQR(COS(X))+SIN(X)

2. Мына өрнектің 6 MOD 4 нәтижесі неге тең?

   A) 4

   B) 2

   C) 1

   D) 0

   E) 6

3.TRUNC(8.915) нәтижесі неге тең?

   A) 8.9

   B) 9

   C) 8

   D) 8.91

   E) 8.92

III.4.4. Функционалдық программалау тілдері

Төменде функционалдық тілдердің ең бұрын шыққан және ең көп тарағаны Lisp тілін 1960 жылдары америка ғалымы Д.Маккарти жасаған. Бұл тіл *тізім* деп аталатын арнаулы *деректер құрылымын* бейнелеу және өңдеу үшін арналған.

Lisp тіліндегі ең кіші бөлінбейтін өңделетін деректі *атом* дейді. Атом ретінде тұрақты шама мен айнымалы шама алынады. Көпшілік Lisp–жүйелер қандай–да бір арнаулы сөздерді айнымалылардың атаулары ретінде пайдалануға рұқсат етеді, бірақ ескерту жасайды. Мысалы, F, T, NIL деген сөздерді айнымалылардың атаулары ретінде қолдануға болмайды, себебі, олар логикалық шамалардың мәндері: F – жалған, T – ақиқат, NIL– жоқ немесе жалған.

Lisp тілінде: әрбір программалық құрылым тек функция болып келеді, программаны құрастыру деген күрделі функцияны қарапайым функциялар арқылы құрастыру болып табылады. Функциялар арасындағы қарым–қатынас тек программа жұмыс істеген кезде олардың шақырулары арқылы жүзеге асады. Кез–келген функция екі жақшаның «(» және «)» ішінде жазылады және оның түрі мынадай болады:

(<функция атауы> <аргументтер тізімі>)

Функция мен аргумент атаулары *идентификатор* арқылы беріледі. Аргумент ретінде тұрақты шама да алынады, бұл жағдайда осы тұрақты шаманың нақты бейнеленуі жазылады. Сонымен қатар, аргумент ретінде басқа функция қолдануы мүмкін, бұл жағдайда композиция (суперпозиция) амалы анықталды дейді, яғни, күрделі функция анықталады: ол өзінің аргументі ретінде функция–аргументке қолданылған *аппликация* амалының нәтижесін алады.

Мысалы, квадрат теңдеуді шешу үшін оның дискриминантын мынадай формула $D = B^2 - 4AC$ арқылы табылатындығы белгілі. Осы формуланы Lisp тілінде былай жазуға болады:

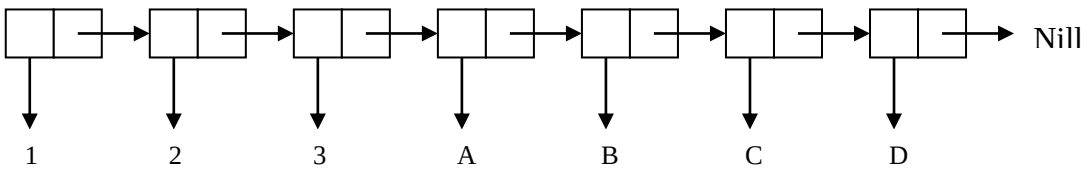
(**SETQ** D (**SUB** (**MUL** B B) (**MUL** 4 A C))),

мұнда **MUL** – көбейту амалын орындайтын функцияның аты, **SUB** – азайту функциясының аты, **SETQ** – айнымалыға мән беру функциясы (меншіктеу амалывна ұқсас). **MUL** және **SUB** сандық функция, яғни аргументтері де нәтижелеріде сандық шама болады, ал **SETQ** – кез келген типте анықталған.

Тізім деп сұрыпталған элементтер жиынын айтады. Тізімнің элементтері атомдар немесе тізімдер болады. Яғни, тізімдер *рекурсиялық құрылым*.

Әрбір элемент екі бөліктен тұрады: бірінші бөлікте элементің мәніне сілтеме, ал екінші бөлікте келесі элементке сілтеме орналасады. Осыған байланысты екі жағдайды қарастырайық:

1.Тізімнің элементі атом болса, онда осы элементтің мәні ретінде тұрақты сан, символ немесе логикалық мән (F – жалған, T – ақиқат, NIL– жоқ) алынады. Мысалы, тізім ретінде жеті атомдық элементтен тұратын (1 2 3 A B C D) өрнегін аламыз. Ол графикалық түрде былай бейнеленеді:



Егер осы тізімге SETQ функциясын қолдансақ, яғни жазсақ:

(**SETQ** X (1 2 3 A B C D))

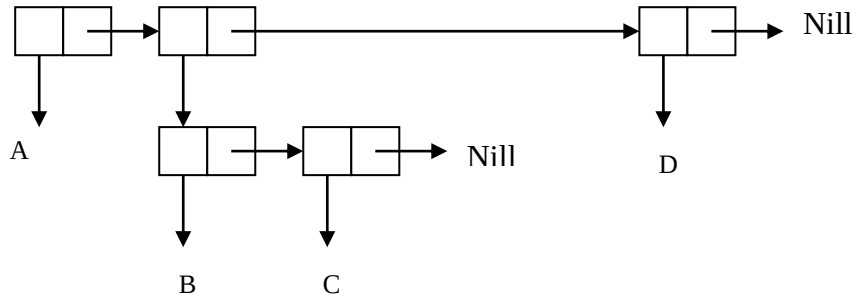
онда біз X айнымалысына тізімді меншіктеу кезінде үлкен қиындыққа ұшыраймыз. Себебі, (1 2 3 A B C D) тізіміндегі бірінші атом 1, ол функцияның атауы болуы керек. Бірақ Lisp тілінде

функцияның атауы 1 болуы мүмкін емес. Сондықтан өз аргументін өзінің нәтиже қылатын функция керек. Ондай функцияның атауы QUOTE. Яғни, мына өрнектің

(**SETQ** X (**QUOTE** (1 2 3 A B C D)))

мәні (1 2 3 A B C D) болады. Тізімді функцияның мәні ретінде алудың басқа да жолы бар. Ол үшін LIST функциясын қолдану керек. Мысалы, (LIST 1 2 3 A B C D) өрнегінің мәні (1 2 3 A B C D) болады.

2. Тізімнің элементі тізім болса, онда оны өңдеу үшін әртүрлі функцияны қолдануға болады. Мысалы, (A (B C) D) өрнегі берілсін. Бұл күрделі тізімде үш элемент бар және оның біреуі екі атомнан тұратын қарапайым тізім (B C). Осы күрделі тізімнің графикалық түрі мынадай болады:



Lisp тіліндегі негізгі деректер құрылымын *S–өрнек* деп атайды, мұнда *S* ағылшын тіліндегі Symbolic деген сөздің бірінші әріпі. *S–өрнек* атом, немесе тізім болады.

Lisp тілінде тізімдермен жұмыс істеу үшін мынадай стандартты функциялар анықталған:

1. **CAR** функциясының аргументі тізім, ал нәтижесі осы тізімнің бірінші элементі болады. Мысалы, X–тің әртүрлі мәнінде (**CAR** X) функциясы әртүрлі мән береді:

a) егер X мәні (1 2 3 4) болса, онда (**CAR** X) мәні 1 болады;

b) егер X мәні (1 (2 (3 (4)))) болса, онда (**CAR** X) мәні 1 болады;

c) егер X мәні ((1 2) (3 4)) болса, онда (**CAR** X) мәні (1 2) болады;

d) егер X мәні (((1 2) 3) 4) болса, онда (**CAR** X) мәні ((1 2)3) болады;

e) егер X мәні ((((1 2) 3) 4)) болса, онда (**CAR** X) мәні (((1 2) 3) 4) болады;

2. **CDR** функциясының аргументі тізім, ал нәтижесі осы тізімнің бірінші элементін жойғаннан кейін қалған қалдық болады. Мысалы, жоғарыдағы X–тің әртүрлі мәнінде (**CDR** X) мынадай нәтиже береді:

a) егер X мәні (1 2 3 4) болса, онда (**CDR** X) мәні (2 3 4) болады;

b) егер X мәні (1 (2 (3 (4)))) болса, онда (**CDR** X) мәні (2 (3 (4))) болады;

c) егер X мәні ((1 2) (3 4)) болса, онда (**CDR** X) мәні (3 4) болады;

d) егер X мәні (((1 2) 3) 4) болса, онда (**CDR** X) мәні (4) болады;

e) егер X мәні ((((1 2) 3) 4)) болса, онда (**CDR** X) мәні Nil болады;

3. **CONS** функциясы **CAR** және **CDR** функцияларының нәтижелерін қайтадан біріктіреді. Мысалы, (**CONS** X Y) функциясы X пен Y–тің жоғарыдағы мәндері бойынша мынадай біріктірулер алады:

a) егер X мәні 1 және Y мәні (2 3 4) болса,
онда (**CONS** X Y) мәні (1 2 3 4) болады;

b) егер X мәні 1 және Y мәні (2 (3 (4))) болса,
онда (**CONS** X Y) мәні (1 (2 (3 (4)))) болады;

c) егер X мәні (1 2) және Y мәні (3 4) болса,
онда (**CONS** X Y) мәні ((1 2) (3 4)) болады;

d) егер X мәні ((1 2)3) және Y мәні (4) болса,
онда (**CONS** X Y) мәні (((1 2)3)4) болады;

e) егер X мәні (((1 2) 3) 4) және Y мәні Nil болса,
онда (**CONS** X Y) мәні (((1 2) 3) 4) болады;

4. **EQUAL** функциясының екі аргументі болады. Бұл функцияның нәтижесі логикалық T – ақиқат болады, егер аргументтердің мәндері тең болса және кері жағдайда Nil – жалған болады.

5. **GREAT** функциясының екі аргументі болады. Бұл функцияның нәтижесі логикалық Т – ақиқат болады, егер бірінші аргументің мәні екінші аргументің мәнінен үлкен болса және кері жағдайда Nil болады.

6. **COND** функциясы тармақталуды жүзеге асырады. Бұл функцияның аргументтерінің саны анықталмаған. Бірақ оның әрбір аргументі қосақ, яғни, екі элементтен тұратын тізім болып келеді: бірінші элемент тармақталу шартын береді, ал екінші элемент не шаманы, не әрекетті көрсетеді. Функцияны орындау кезінде алдымен бірінші аргументің шарты тексеріледі: егер осы шарт орындалса, онда функцияның мәні болып оның қосағы алынады, ал осы шарт орындалмаса келесі аргументтің шарты тексеріледі және ары қарай жалғастырыла береді. Егер бір де бір аргументің шарты орындалмаса, онда функция мәні NIL болады.

7. **DEFINE** функциясы өзі программада кезкескеннен кейін ары қарай функцияның анықтамасы болады дегенді білдіреді, яғни, ол кез келген функцияның анықтамасының алдында тұрады. Бұл функцияның бір ғана аргументі бар және ол анықталатын функциялардың тізімі болады. Ал анықталатын әрбір функцияның өзі екі элементті тізім болады: бірінші элемент функцияның атауы, ал екінші элемент осы функцияны анықтайтын өрнектің сипаты. Мысалы, DEFINE функциясын мына түрде бейнелеуге болады:

(**DEFINE** ((**F1** X1) (**F2** X2) (**F3** X3))),

мұнда функцияның аргументі ((**F1** X1) (**F2** X2) (**F3** X3)) – анықталатын функциялардың тізімі; **F1, F2, F3** – анықталатын функциялардың атаулары; X1, X2, X3 – анықталатын функциялардың сипаттамалары. Егер жаңадан анықталатын функциялардың сипаттамалары ретінде бұрын белгілі функциялардың атауларын жазылса, онда жаңа функциялар сол бұрынғы функциялардың дәл көшірмесін береді. Мысалы, мына өрнектегі (**DEFINE** ((**F1** MUL) (**F2** SUB) (**F3** SETQ))) анықталатын функциялар **F1, F2, F3** бұрын белгілі болған **MUL, SUB, SETQ** функцияларына сәйкес болып, және тура солар құсап жұмыс істейді.

8. **LAMBDA** функциясы басқа функцияларды анықтау үшін қажет. Бұл функцияның әрқашан екі аргументі болады: бірінші аргумент анықталатын функциялардың аргументтерінің тізімі, ал екінші аргумент осы анықталатын функциялар пайдаланатын өрнектердің қызметін атқарады. Мысалы, $X^2 - Y^2$ өрнегіне сәйкес Lisp тілінде мына функцияны жазуға болады:

(**LAMBDA** (X Y) (**SUB** (**MUL** X X) (**MUL** Y Y))),

мұнда X және Y айнымалылары параметрдің қызметін атқарып тұр.

LAMBDA функциясының мәні сан, атау, тізім және т.с.с болмайды, оның мәні болып өзінің сипаттамасы есептелінеді және ол мән осы функциясының атауы қай жерде кездессе, сол жерде қолданылады. Яғни, **LAMBDA** функция емес, өрнек деуге болады. Бұл 1941 жылы ағылшын оқымыстысы Чёрч «λ – түрлендіру» деген атпен математикаға ендірген өрнекке негізделген: жоғарыда қарастырылған мысалды $\lambda xy(x^2 - y^2)$ өрнек түрінде жазуға болады және оның мәні мына $\lambda yx(x^2 - y^2)$ өрнектің мәнінен бөлек болатынын ескерген жөн.

LAMBDA функциясында параметрлерді нақты мәндермен байланыстыруға болады. Ол үшін осы функция анықтамасының соңында параметрлердің мәндерінің тізімін жазу керек. Мысалы, мынадай өрнектің

((**LAMBDA** (X Y) (**SUB** (**MUL** X X) (**MUL** Y Y))) 5,4)

мәні 9 болады, себебі $5^2 - 4^2 = 25 - 16 = 9$.

Енді осы айтылғандарды пайдаланып қарапайым функцияларды анықтап Lisp тілінде программа құруға болады.

III.4.4. Мысалдар:

1. Үшбұрыштың ауданын есептейтін программаны мына фомула бойынша $p = \frac{a+b+c}{2}$; $s = \sqrt{p(p-a)(p-b)(p-c)}$ құрыңыз.

Мұнда *a, b, c* - үшбұрыштың қабырғалары, *p* – периметрдің жартысы, *s* – ауданы.

Осыларға сәйкес Lisp тіліндегі программаны былай болады:

DEFINE(((**AUDAN** (**LAMBDA** (A B C)(**SETQ** P (**DIV** (**ADD** A B C) 2)) (**SETQ** S (**SQRT** (**MUL** P (SUB P A) (**SUB** P B) (**SUB** P C))))))))

2. Бүтін ***N*** санының факториалын есептейтін программаны жазу керек. Ол үшін факториал ұғымының рекурсиялық анықтамасын аламыз:

$$0! = 1$$

$$N! = N*(N-1)!$$

Осыларды ескеріп Lisp тілінде факториалды есептейтін программаны былай жазуға болады:

DEFINE(((**FACT** (**LAMBDA** (N) (**COND**

((**EQUAL** 0 N) 1) (T (**MUL** N (**FACT** (**SUB** N 1))))))))

3. Евклид алгоритмі арқылы N1 және N2 сандарының ең үлкен ортақ бөлгішін **EUOB** табу. Ол Lisp тілінде былай жазылады:

DEFINE(((**EUOB** (**LAMBDA** (N1 N2) (**COND**

((**EQUAL** 0 N2) N1) (T (**EUOB** (**DIV** N 1 N2)))))))

III.4.4. Тапсырмалар:

1) Егер X мәні (6 7 8 9) болса, онда (**CAR** X) мәні қанша болады;

2) Егер X мәні ((1 2) (7 8)) болса, онда (**CDR** X) мәні қанша болады;

3) Егер X мәні ((3 4)5) және Y мәні (6) болса, онда (**CONS** X Y) мәні қанша болады?

Көмек

1) **CAR** функциясының мәні аргументінің біріншісі.

2) **CDR** функциясының мәні аргументіндегі бірінші элементті алып тастағаннан кейін қалғаны.

3) **CONS** функциясы **CAR** және **CDR** функцияларының нәтижелерін қайтадан біріктіреді.

III.4.4. Сұрақтар:

1. Функцияналдық программалау тілдерде ең кіші программалық бірлік не?

2. Функцияналдық программалау Lisp тілінде қандай стандартты функциялар бар?

3. Функцияналдық программалау Lisp тілінде қандай деректер құрылымымен жұмыс ітейді?

III.4.4. Тестер:

1. Егер X мәні (1 2 3 4) болса, онда (**CAR** X)- ның мәні неге тең?

   A) 0

   B) 2

   C) 3

   D) 4

   E) 1

2. Егер X мәні ((1 2)3) және Y мәні (4) болса, онда (**CONS** X Y) мәні қаншаға тең?

   A) (((1 2)3)4)

   B) (1 2 3 4)

   C) ((1 2) (3 4))

   D) (1 2)3)4)

   E) (((1 2 3 4))

3. Lisp тілінің программалық бірлігі не?

   A) оператор

   B) тізім

   C) шарт

   D) функция

   E) процедура

III.4.5. Логикалық программалау тілдері

Логикалық программалау тілдері 1960 жылы Грин айтқан «математикадағы предикаттар логикасының тілін программалау тілі ретінде қолдануға болады» деген идеяны жүзеге асыру жолында пайда болды.

Бұл идеяның мағынасы мынадай: *программалаушы процедуралық тілдердегі сияқты есептің шешіміне алып келетін әрекеттердің тізбегін компьютерге тәптіштеп жазбай, логикалық тілде керекті есептің қойылымын* (*есепке қатысты объектілердің қасиеттерін және олардың бір–бірімен қатынасын*) *ғана сипаттайды, ал есептің шешімін логикалық қорытындылау ретінде компьютер өзі табады*.

Логикалық программалау тілдердің ішіндегі кең тарағаны 1972 жылы француз ғалымы А. Колмераурердің ғылыми жетекшілігімен 1-ші реттегі предикаттар логикасы тілінінің дербес (*хорнның дизъюнктілерімен шектелген және резолюция әдісімен жабдықталған*) жағдайына негіздеделіп жасалған *Prolog* тілі.

Prolog тілінде программалық бірлік болып логикалық тұжырым есептелінеді. Ең кіші программалық бірлік *факты* болады, ал келесі программалық бірліктер ретінде *ережелер* және *сұратымдар* алынады.

Факты деп $\boldsymbol{P(t_1, t_2, \ldots, t_k)}$ түрінде берілген логикалық формуланы айтады, мұнда $\boldsymbol{P}$ – предикаттық символдар, $\boldsymbol{t_1, t_2, \ldots, t_k}$ – тұрақты шамалар, айнымалы шамалар және функциялардан тұратын *термдер*. Программада $X_1, X_2, \ldots, X_r$ – айнымалылары кіретін $\boldsymbol{P(t_1, t_2, \ldots, t_k)}$ фактысының болуын, кез келген объектілер

$X_1, X_2, \ldots, X_r$ үшін предикат $\boldsymbol{P(t_1, t_2, \ldots, t_k)}$ ақиқат болады деп түсіну керек. Мысалы, программада ***кіші***(***x***, ***x+1***) фактысының болуы, ***x*** объектісі ***x+1*** объектісінен кіші болады деген тұжырымды білдіреді.

Ереже деп мына түрде $\boldsymbol{P_1, P_2, \ldots, P_n}$:– $\boldsymbol{P_0}$ жазылған өрнекті айтады, мұндағы $\boldsymbol{P_0, P_1, P_2, \ldots, P_n}$ – конъюнкция амалымен тіркескен фактылар, «:–» – қорытындылау белгісі. Егер ережеде $X_1, X_2, \ldots, X_r$ айнымалылары кездессе, онда ереженің мағынасы мынада: барлық объектілер $X_1, X_2, \ldots, X_r$ үшін $\boldsymbol{P_1, P_2, \ldots, P_n}$ тұжырымдарынан $\boldsymbol{P_0}$ тұжырымы шығады немесе барлық $X_1, X_2, \ldots, X_r$ үшін, егер $\boldsymbol{P_1, P_2, \ldots, P_n}$ тұжырымдары ақиқатты болса, онда $\boldsymbol{P_0}$ тұжырымы да ақиқатты болады.

Сұратым деп мына түрде ?– $\boldsymbol{Q_1, Q_2, \ldots, Q_m}$ жазылған өрнекті айтады, мұнда $\boldsymbol{Q_1, Q_2, \ldots, Q_m}$ – конъюнкция амалымен тіркескен фактылар, «?–» – сұратым белгісі.

Айнымалысы жоқ сұратым былай оқылады: рас па мыналар $\boldsymbol{Q_1, Q_2, \ldots, Q_m}$?

Егер сұратымда $X_1, X_2, \ldots, X_r$ айнымалылары кездессе, онда сұратымның мағынасы мынада: қайсы $X_1, X_2, \ldots, X_r$ объектілер үшін $\boldsymbol{Q_1, Q_2, \ldots, Q_m}$ рас па деген сұратылады.

Prolog тілінде компьютерге тек программада бар деректер, соның ішінде стандартты предикаттар (=, <, > сияқтылар) және стандартты функциялар (+, –, *, / сияқтылар) қасиеттері, ғана белгілі деп есептелінеді және сұратудың жауабы осы деректерден логикалық қорытындылады.

Сонымен логикалық программалардың қарапайым, түсінікті және табиғи семантикасы бар: ***P*** программасына X_1, X_2, ..., X_r айнымалылары бар ?– Q_1, Q_2, …, Q_m сұратымын осы ***P*** программасындағы тұжырымдардан Q_1, Q_2, …, Q_m логикалық қорындыланатындай етіп X_1, X_2, ..., X_r айнымалыларының барлық мәндерін есептеу талабы деп түсінеді.

Prolog тіліндегі программа деп фактылар мен ережелердің тізбегін айтады. Программада барлық фактылар ережелердің алдында жазылады. Әрбір программаның өзіндік сұратымдары болады. Фактылардың, ережелердің және сұратымдардың атаулары *идентификатор* (кез–келген әріптен немесе әріптен басталып, әріптер мен цифрлардан тұратын тізбек) арқылы беріледі.

Prolog тілінде деректер *атом* немесе *тізім* ретінде беріледі. Атом деп типтері сандық немесе символдық болып келетін тұрақтылар мен айнымалыларды айтады. Тізім тік жақшаның ішінде жазылған және бір–бірімен үтірмен айырылған атомдардан құрылады. Мысалы, [A, B, C, D] – төрт атомнан тұратын тізім. [A| B] сияқты жазба тізімнің басы A, ал құйрығы B дегенді көрсетеді.

Prolog тілінде мынадай амалдар қолданылады: *арифметикалық амалдар* (+ – қосу, - – алу, * – көбейту, / – бөлу); *қатынас амалдары* (= – тең, <= – кіші немесе тең, < – кіші, > = – үлкен немесе тең, > – үлкен); *логикалық амалдар*(not – теріске шығару, and – және, or – немесе);

Prolog тілінде көптеген қолдануға дайын тұрған, яғни, анықтамалары белгілі және сақталып қойған стандарттық предикаттар бар: **sin**(X) – синус, **cos**(X) – косинус, **tan**(X) – тангенс,

arctan(X) – арктангенс, **abs**(X) – абсолют шама, **exp**(X) – экспонента, **ln**(X) – натурал логарифм, **log**(X) – ондық логарифм, **sqrt**(X) – квадрат түбір, ***random*** (X) – нөлден бірге дейінгі кездейсоқ нақты сан. Сонымен қатар, жеке бит (ең кіші жад бірлігі) деңгейінде жұмыс істейтін бірнеше предикаттар бар: **bitnot**(A,B) – емес, **bitand** (A,B,E) – және, **bitor** (A,B,E) – немесе, **bitleft** (A,N,E) – N бит солға жылжыту, **bitright** (A,N,E) – N бит оңға жылжыту.

Осы айтылғандардың негізінде Prolog тілінде әртүрлі программалар құруға болады.

III.4.5. Мысалдар:

1. Мына логикалық программада

ғылым(*математик*а)**.**

ғылым(*информатик*а)**.**

абстрактылы(*математик*а)**.**

қолданбалы(*информатик*а)**.**

қиын(*X*) *:–* ***ғылым***(*X*), ***абстрактылы***(*X*)**.**

пайдалы(*X*) *:–* ***ғылым***(*X*), ***қолданбалы***(*X*)**.**

сұратым

?– ***қиын***(*X*)

мынадай жауап алады:

X= *математика*,

ал сұратым

?– ***пайдалы***(*X*)

мынадай жауап алады:

X= *информатика*,

яғни, ***қиын***(*математика*) және ***пайдалы***(*информатика*) деген тұжырымдар осы программадан логикалық қорындыланып тұр.

2. Prolog тілінде ***әке*** және ***шеше*** деген фактыларды құрып ***ата***, ***апа*** және ***әже*** деген тұжырымға ережелер құру:

әке(*Болат*, *Дина)*.

әке(*Дулат*,*Болат*).

әке(*Абай*, *Айжан*).

шеше(*Ұлжан*, *Айжан*).

шеше(*Айжан*, *Жанар*).

шеше(*Айжан*, *Жамал*).

шеше(*Жанар*, *Шолпан*).

шеше(*Жамал*, *Болат*).

ата(*X*, *Z*) :– ***әке***(*X*, *Y*), ***әке***(*Y*, *Z*).

ата(*X*, *Z*) :– ***әке***(*X*, *Y*), ***шеше***(*Y*, *Z*).

апа(*X*, *Z*) :– ***шеше***(*X*, *Y*), ***әке***(*Y*, *Z*).

апа(*X*, *Z*) :– ***шеше***(*X*, *Y*), ***шеше***(*Y*, *Z*).

әже(*X*, *Z*) :– ***шеше***(*X*, *Y*), ***ата***(*Y*, *Z*).

әже(*X*, *Z*) :– ***шеше***(*X*, *Y*), ***апа***(*Y*, *Z*).

әже(*X*, *Z*) :– ***апа***(*X*, *Y*), ***әке***(*Y*, *Z*).

әже(*X*, *Z*) :– ***апа***(*X*, *Y*), ***шеше***(*Y*, *Z*).

Осы программадағы кейбір сұратымдардың жауаптары мынадай:

?– ***ата***(*X*, *Дина)*, онда жауап ***ақиқат***, *X=Дулат*

?– ***әже***(*X*, *Дина)*, онда жауап ***жалған***

?– ***ана***(*X*, *Жанар*), онда жауап ***ақиқат***, *X=Ұлжан*

?– ***ана***(*X*, *Шолпан*), онда жауап ***ақиқат***, *X=Айжан*

?– ***әже***(*X*, *Шолпан*), онда жауап ***ақиқат***, *X=Ұлжан*

?– ***ата***(*Болат*,*Y*), онда жауап ***жалған***

?– ***ана***(*Жамал*,*Y*), онда жауап ***ақиқат***, *Y= Дина*

?– ***әже***(*Айжан*,*Y*), онда жауап ***ақиқат***, *Y=Дина*

?– ***әже***(*Ұлжан*,*Y*), онда жауап ***ақиқат***, *Y=Шолпан*

3. Теріс емес бүтін ***N*** санының факториалын есептейтін программаны жазу керек. Ол үшін рекурсияны қолданамыз:

fact (0, 1).

fact (N, R) : – ***sub*** (N, 1, N1), ***fact*** (N1, P), ***mul*** (P, N, R) .

немесе

fact (0,1).

fact (N, R) : – N1=N–1, ***fact*** (N1, P), R=P*N.

Мұнда N – алғашқы берілген сан, R – нәтиже, P – аралық нәтиже.

III.4.5. Тапсырмалар:

Prolog тілінде ***әке*** мен ***шеше*** деген фактыларды және ***ата*** мен ***ана*** деген ережелерді алдын ала анықтап алып, ***ұл***, ***қыз***, ***немере*** және ***жиен-немере*** деген тұжырымдарға ережелер құрыңыздар.

Көмек:

Ең үлкен буынның анықтамасы орта буынның анықтамасын қамтитын болуы керек, ол өз кезегінде кіші буынның анықтамасын қамтып тұруы керек.

III.4.5. Сұрақтар:

1. Логикалық тілдің негізгі программалау бірліктеріне нелер жатады?
2. Мына алгебралық өрнекті дифференциалдайтын Prolog тіліндегі программа дұрыс жұмыс жасай ма?

% ***dif*** (+Өрнек, +Айнымалы, –Нәтиже).

dif(*X*, *X*, *1*) :– !.

dif(*C*, *X*, *0*) :– atomic(C).

dif(*–U*, *X*, *–A*) :– ***dif***(*U*, *X*, *A*). % (*–u*)'=*–u*'

dif(*U+V*, *X*, *A+B*) :– ***dif***(*U*, *X*, *A*), ***dif***(*V*, *X*, *B*). % (*u+v*)'=*u'+v'*.

dif(*U–V*, *X*, *A–B*) :– ***dif***(*U*, *X*, *A*), ***dif***(*V*, *X*, *B*).

dif(*C*U*, *X*, *C*A*) :– atomic(C), C\=X, ***dif***(*U*, *X*, *A*), !.

dif(*U*V*, *X*, *U*B+A*V*) :– ***dif***(*U*, *X*, *A*), ***dif***(*V*, *X*, *B*). % (*uv*)'=*uv'+u'v*

dif(*U/V*, *X*, (*A*V–U*B*)*/V?2*) :– ***dif***(*U*, *X*, *A*), ***dif***(*V*, *X*, *B*).

dif(*U?C*, *X*, *C*A*U?*(*C–1*)) :– ***dif***(*U*, *X*, *A*).

dif(*log*(*U*), *X*, *A/U*) :– ***dif***(*U*, *X*, *A*).

?–***dif***(*x*x–2*, *x*, *R*).

*R=x*1+1*x–0*

III.4.5. Тестер:

1. Мына тілдердің қайсысы логикалық программалау тілі болады?

A) ML
B) Prolog
C) Lisp
D) Pascal
E) Parlog

2. Prolog тілінің мына сұратымына

Goal: ***random*** (X)

Prolog-жүйесінің жауабы қалай болады?

A) 0 ден 1-ге дейінгі кездейсоқ нақты сан
B) X санының бүтін бөлігі.
C) 0 ден X–ке дейінгі Y кездейсоқ бүтін саны.
D) Yes
E) X–ті жуық бүтін санға дейін дөңгелектеу.

3. «Баласы бар әркім бақытты» деген тұжырым Prolog тілінде қалай жазыдады?

A) бақытты(X) :– әке(X, _).
B) бақытты(X,Y) :– әке(X, Z).
C) бақытты(X).
D) әке(X, Z):–бақытты(X, Z).
E) бақытты(X):–бала(X).

III.4.6. Продукциялық программалау тілдері

Продукциялық тілдер алгоритм ұғымының математикалық анықтамасының (формалдау әдісінің) бірі болып есептелетін 1954 жылы ресей ғалымы А.Марков жасаған *нормалды алгорифмдер тілін* программалау тіліне айналдыруды көздеген мақсатты жүзеге асыру жолында жасалынған.

Продукциялық программалау тілдерінде негізгі программалық бірлік *продукция* деп аталатын *алмастырым ережесі* (ағылшынша – *rewriting*, орысша – *подстановочный*).

Продукциялық тілде программа құру деген ол қарапайым алмастырым ережелерін пайдалана отырып, күрделі алмастырым ережелерін құру болады. Алмастырым ережелері бір–бірімен топтасып *ережелер схемасын* құра алады. Әрбір ережелер схемасы бір ортақ атаумен аталып логикалық біртұтастығы бар әрекетке немесе қатынасқа бағытталып құрылады.

Продукциялық тілдердің ішіндегі ең кең тарағаны және қолайлысы 1968 жылы орыс ғалымы В. Турчин жасаған *Refal* тілі. Кезінде (1970–1985жж) бұл тілде көптеген программалалық жүйелер, бірнеше тілдердің трансляторлары жасалынды. Соның ішінде Algol, Fortran, Simula, Dynamo, ТРИКС және т.б.

Refal тіліндегі негізгі бірлік функция (ережелер схемасы) болады. Сондықтан осы тілдегі кез келген программа атаулары бөлек *функциялардың* тізбегінен тұрады. Әрбір функция бірнеше алмастырым ережелерінен құралады. Алмастырым ережесінің бір–бірінен арнаулы «=» таңбамен айрылған екі жағы болады. Сол жағы

осы ереженің қолдану шартын анықтайды, ал оң жағы жасалатын әрекетті немесе тексерілетін қатынасты көрсетеді. Сонымен қатар, оң жақта функциялардың шақыруы, соның ішінде рекурсиялық (өзін-өзі) шақыру жазылады.

Рекурсиялық шақырудың екі түрі қолданылады: тікелей шақыру, жанай шақыру. Тікелей шақырылғанда осы ереже кіретін функцияның атауы көрсетіледі, ал жанай шақыруда басқа функциялар арқылы осы функцияны өзін-өзі шақыру жүзеге асады.

Refal тіліндегі программаның әрбір жұмыс қадамында қандай функция шақырылатындығы және оның кірісі мен шығысы бірмәнді анықталады. Кез келген алмастырым ережесінің, функцияның және программаның кіріс және шығыс деректері белгілі бір ережемен құрылған арнаулы символдық өрнек арқылы беріледі. Функция атауы *идентификатор* (кез–келген әріптен немесе әріптен басталып, әріптер мен цифрлардан тұратын тізбек) арқылы беріледі.

Басқа программалау тілдер сияқты Refal тілінде стандарттық функциялар бар. Олар бірігіп бірнеше топ құрайды. Солардың кейбіреулерінің сипаттамасы төменде қарастырылады:

1. *Арифметикалық функцияларға* бүтін сандар үшін анықталған мыналар жатады:

ADD – *қосу*;

SUB – *алу*;

MUL – *көбейту*;

DIV– *бөлу*.

2. Егер амал қолданатын бүтін сандарды S1 және S2 деп белгілесек, онда бұл функциялардың шақырулары мен нәтижелері төмендегідей болады:

Шақыруы	**Нәтижесі**
<**ADD** (S1) S2>	S1+ S2
<**SUB** (S1) S2>	S1– S2
<**MUL** (S1) S2>	S1* S2
<**DIV** (S1) S2>	S1\ S2

мұнда қ*осу*, *алу* және *көбейту* функцияларының нәтижелері әдеттегідей болады да ешқандай сұрақ туғызбайды, ал бөлу функциясының нәтижесі N1(N2) түрінде беріледі, N1– бөлінді, N2 – қалдық және олар мына шартты S1 = S2*N1+N2 & N2≤S2 қанағаттандырады. Мысалы, егер:

<**DIV** (/5/) /3/> болса, онда нәтиже /1/(/2/);

<**DIV** (–/5/) /3/> болса, онда нәтиже –/1/(–/2/);

<**DIV** (–/5/) –/3/> болса, онда нәтиже /1/(–/2/);

<**DIV** (/4/) /2/> болса, онда нәтиже /2/(/0/),

мұнда бүтін сандар «/» екі таңбасының ішіне алынып жазылады.

2. *Символдық функциялар* символдық өрнектерде анықталған. Олардың қатарына **FIRST, LAST** және **TYPE** жатады.

FIRST және **LAST** екі аргументті функциялар: бірінші аргумент бүтін сан, екінші аргумент символдық өрнек. **TYPE** бір аргументті функция, ол аргумент символдық өрнек болады.

FIRST (**LAST**) функциясының екі аргументі болады және саны бірінші аргументпен көрсетілген санға тең болатын алдыңғы

- сол жағындағы (соңғы - оң жағындағы) термдерді дөңгелек жақшаның ішіне салып, екінші аргументі арқылы берілген символдық өрнекті екіге бөледі. Бұл функциялардың шақыруларының түрі мынадай <**FIRST** N E> және <**LAST** N E> болады, мұнда N – бүтін сан, E –символдық өрнек болуы керек. Ал олардың нәтижелері екі жағдайға байланысты:

1) Егер E символдық өрнегінің ұзындығы N санынан үлкен немесе N санына тең болса, онда (E1) E2 – **FIRST** үшін және E1(E2) – **LAST** үшін;

2) Егер E символдық өрнегінің ұзындығы N санынан кіші болса, онда *E – **FIRST** үшін және E* – **LAST** үшін.

TYPE функциясы мына түрде < **TYPE** E > шақырылады және ол E өрнегінің сол жақтағы бірінші бөлігін талдайды және соның мәніне байланысты әртүрлі сипаттау символын тудырады да, нәтижесі мына түрде ξE шығады, мұнда ξ – сипаттау символы. Сипаттау символының бірнеше мәндері мынадай болады:

№	E өрнегінің сол жақтағы бірінші бөлігінің мәні	ξ–ң мәні
1	Құрамды символ–ен	F
2	Құрамды символ–сан	N
3	Объектілі таңба – әріп	L
4	Объектілі таңба – цифр	D
5	Объектілі таңба – әріп емес және цифр емес	O
6	Сол жақ құрылымдық (дөңгелек) жақша	B
7	Бос тізбек	*

III.4.6. Мысалдар:

1. Кіріс дерегінің құрылымы «*идентификатор*» ұғымының құрылымына сәйкестігін анықтайдын программа мынадай болады:

* *Бірінші символды тексеру*

ID E1 = < **ID1** <**TYPE** E1>>

* *Бірінші символдың мәні әріп бола ма?*

ID1 ‘L’ SA E1 = < **ID2** (SA) <**TYPE** E1>>

* *Келесі символдың мәні әріп немесе цифр бола ма?*

ID2 (EX)‘L’ SA E1 = < **ID2** (EX SA) <**TYPE** E1>>

(EX)‘D’ SA E1 = < **ID2** (EX SA) <**TYPE** E1>>

(EX)‘*’ = (EX)

(EX) SA E1 = < **PRINT ‘*** Қате**:** SA – әріп те, цифр да емес’>

Бұл программаның шақыруының түрі <**ID** E>, ал нәтижесінің түрі (E) болады, және ол **ID2** функциясының үшінші ережесінен шығады, мұндағы E – кез–келген символдық өрнек, соның ішінде бос тізбек.

2. Егер Евклид алгоритмі арқылы N1 және N2 сандарының ең үлкен ортақ бөлгішін **EUB** деп белгілесек, онда мынадай ережені жазуға болады:

EUB (E1) E2 (/0/) = E1

(E1) E2 (E3) = < **EUB** (E3) <**DIV** (E1) E3>>

мұнда E1, E2, E3 – кез келген мәнді, соның ішінде бос тізбекті де, қабылдай алатын символдық өрнек.

Бұл функция мына түрде шақырылады: < **EUB** (N1) (N2) >.

3. Бүтін N санының факториалын есептейтін программа мынадай:

FACT /0/ = /1/

S1 = <**MUL** (S1) <**FACT** <**SUB** (S1) /1/>>>

III.4.6. Тапсырмалар:

Функциялардың шақырулары төмендегідей болса, нәтиже қандай болады?

a. <**ADD** (J1) S2>
b. <**SUB** (J1) J2>
c. <**MUL** (N1) N2>
d. <**DIV** (/8/) /2/>

Көмек

Мұнда *қосу, алу және көбейту* функцияларының нәтижелері әдеттегідей болады да ешқандай сұрақ туғызбайды, ал *бөлу* функциясының нәтижесі N1(N2) түрінде беріледі, N1– бөлінді, N2 – қалдық және олар мына шартты S1 = S2*N1+N2 & N2≤S2 қанағаттандырады.

III.4.6. Сұрақтар:

1. Продукциялық тілдің негізгі программалау бірліктері қандай?
2. Программалау тілдеріндегі қандай стандартты функциялар бар болады?

3. Синтаксистік анализ жасайтын *Refal* тіліндегі программада қандай стандартты функцияларды қолдануға болады?

III.4.6. Тестер:

1. Рефал продукциялық программалау тілінде бүтін сандар үшін анықталған арифметикалық функцияларға мыналардың қайсылары жатады?

A) **ADD** – *қосу*, **SUB** – *алу*, **MUL** – *көбейту*, **DIV**– *бөлу*.

B) **ADD** – *алу*, **SUB** – *қосу*, **MUL** – *көбейту*, **DIV**– *бөлу*.

C) **ADD** – *бөлу*, **SUB** – *алу*, **MUL** – *көбейту*, **DIV**– *қосу*.

D) **ADD** – *қосу*, **SUB** – *көбейту*, **MUL** – *алу*, **DIV**– *бөлу*.

E) **ADD** – *бөлу*, **SUB** – *алу*, **MUL** – *көбейту*, **DIV**– *қосу*.

2. Рефал тілінде рекурсиялық шақырудың қандай екі түрлерін қолдпнуға болады?

A) Символдық шақыру, жанама шақыру.

B) Жанама шақыру, жанай шақыру.

C) Тікелей шақыру, жанама шақыру.

D) Тікелей шақыру, жанай шақыру.

E) Жанама шақыру, символдық шақыру.

3. 1968 жылы орыс ғалымы В. Турчин нормалды алгорифм негізінде қандай тілді жасаған?

A) Dynamo

B) Algol

C) Fortran

D) Simula

E) Refal

III.4.7. Объектілі-бағытталған программалау тілдері

Объектілі-бағытталған программалау тілдері (ОБПТ) жарқын өкілдерінің бірі болып 1995 жылы Sun Microsystems компаниясы әзірлеген Java тілі табылады. Java қосымшалары арнайы байт-кодтарға аударылады (трансляцияланады), сондықтан олар компьютерлік архитектураға тәуелсіз кез келген виртуалды Java-машинада жұмыс жасай алады. Төменде Java тілі сипатталады.

Java *тілінің әліпбиі* әріптерден, ондық цифрлардан және арнайы символдардан тұрады. Әріптерге латын әріптері (ASCII стандартына кодталады), ұлттық әліпбилер әріптері (Unicode стандартында кодталады, UTF-16 кодталу), сонымен қатар басқарушы тізбектермен кодталатын, оларға сәйкес символдар (олар туралы кейінірек айтылады) символдар жатады.

Айнымалылардың, ішпрограмма-функциялардың және т.б. программалау тілі элементтерінің атаулары құрамында тек әріптер мен цифрларды қолдануға болатын және біріншісі әрдайым әріп (соның ішінде төменгі сызықша және доллар символдары) болатын, ал кейін кез келген әріптер мен цифрлар комбинациясы болатын кез келген ұзындықты идентификаторлар көмегімен белгіленеді. Ұлттық әліпбилердің кейбір символы әріп ретінде қарастырылады және оларды идентификаторларда қолдануға болады. Бірақ кейбірін ажыратқыш ретінде қолданады және оларды идентификаторларда қолдануға болмайды.

Java тілі *регистрге сезімтал* болып келеді. Бұл идентификаторлар символарды қандай (жоғарғы немесе кіші) регистрде терілгеніне сезімтал дегенді білдіреді. Мысалы, *i*1 және I1 атаулары әртүрлі идентификаторға сәйкес келеді.

Java тілінде а*йнымалы* құрамы өзгере алатын жад ұяшығын атайды. Қандай да бір айнымалыны қолданар алдында, осы айнымалы қолданылатын программа бөлігінің алдындағы облыста берілуі тиіс. Айнымалыны жариялау кезінде айнымалы типі, сонан кейін берілген айнымалының идентификаторы көрсетіледі. Java тілінде алдын ала анықталған бірнеше типтер бар: ***int*** – бүтін сан,

float – нақты сан, ***boolean*** – логикалық мән, Object – Java тіліндегі ең қарапайым объектілі тип (сынып), және т.б. Сонымен қатар өзіндік объектік типтер (кластар) беруге мүмкіндік бар, бұл туралы кейінірек айтылады.

Типі ***MyType***1 болатын *a1* және *b1* айнымалыларын жариялау келесідей жүзеге асырылады:

MyType*1 a1,b1*;

мұнда ***MyType***1 – осы айнымалылардың типінің атауы.

Басқа мысал – типі ***int*** болатын *j* айнымалысын жариялау:

int *j*;

Типтер алдын ала анықталған немесе қолданушы типі болады. Мысалы, ***int*** – алдын ала анықталған тип, ал ***MyType***1 – қолданушы типі. Айнымалыны жариялау үшін ешқандай резервтелінген сөз керек емес, ал типтер атаулары берілетін айнымалылар алдына жазылады.

Айнымалыны жариялауды оларды инициализациялаумен – бастапқы мәндерді берумен бірге жасауға болады. Бүтін санды *i*1 және *i*2 айнымалыларын жариялау мысалын келтірейік:

int *i*1=5;

int *i*2=-78;

немесе

int *i*1=5, *i*2=-78;

C/C++ тілдеріне тән ***int*** *i*1=*i*2=5; түріндегі меншіктеуге тыйым салынған. Бастаушы программистер үшін“=” символы көптеген басқа тілдердегідей Java тілінде де математикада қолданылатын теңдік символы емес, меншіктеу символы ретінде қолданытатынын атап өту керек. Бұл осы символдың оң жағында тұрған мән сол жақта тұрған айнымалыға көшіріледі дегенді білдіреді. Яғни, мысалы, b=a меншіктеуі b айнымалысына (ұяшыққа) a айнымалысы (ұяшығы) мәнін көшіру керек деген. Сондықтан математикалық тұрғыдан қате *x*=*x*+1 өрнегі программалауда толықтай дұрыс. Бұл *x* ұяшығында сақталған мәнді алып, оған 1-ді қосып (бұл *x* ұяшығынан бөлек жерде орындалады), алынған нәтижені *x* ұяшығына бұрынғы мәннің орнына жазу керек дегенді

білдіреді.

Айнымалыларды жариялағаннан соң, олар өрнектер мен меншіктеулерде қолданыла алады:

айнымалы=мән;

айнымалы =өрнек;

айнымалы1= айнымалы2;

және т.б. Мысалы,

*i*1=*i*2+5**i*1;

Қарапайым типтер деп деректер жадының бір ұяшығында орналасқан және бұл ұяшықтың ішұяшықтары жоқ болатын деректер типі аталады. *Сілтемелі типтер* деп жады ұяшығында (сілтемелі айнымалыда) деректердің өзі емес, тек олардың адрестері, яғни деректерге сілтемелер орналасатын деректер типі аталады. Меншіктеу кезінде сілтемелік айнымалыға деректердің өзі емес, жаңа адрес беріледі. Дегенмен сілтемелі айнымалыларда сақталатын адреске тікелей қолжетімділік жоқ. Бұл деректермен қауіпсіз жұмысты қамтамасыз ету үшін – байқаусыз қателіктердің алдын алу үшін де, қасақана ақпаратты бұзу мүмкіндігін болдыртпау үшін жасалған.

Егер сілтемелік айнымалыға сілтеме меншіктелмесе, онда нолдік адрес сақталады және оған символдық ат null беріледі. Сілтемені бір-біріне меншіктеуге болады, егер олар типтері бойынша үйлесімді болса, оған да null мәні меншіктеледі. Бұл ретте бір сілтемелік айнымалыдан екіншісіне адресі көшіріледі. Сілтемелік айнымалыны теңдікке салыстыруға болады, оның ішінде null теңдікке де. Мұнда деректер салыстырылмайды, олардың сілтемелік айнымалыларда сақталатын адрестері салыстырылады.

Java тілінде барлық типтер примитивті және сілтемелі болып бөлінеді. Примитивті типтерге келесі алдын-ала анықталған типтер жатады: бүтін типтер ***byte***, ***short***, ***int***, ***long***, ***char***, жылжымалы нүкте пішініндегі деректер типтері ***float***, ***double***, бульдік (логикалық) тип ***boolean*** және резервтелген сөз ***enum*** арқылы жарияланған санауланған-тип (қысқа аудару enumeration - «санаулым»). Қалған

Java типтері сілтемелі болады.

Көптеген Java конструкцияларында бір оператор пайдаланылады, бірақ жиі құрамдас оператор ретінде рәсімделген бірнеше операторлардан тұратын тізбекті пайдалануға болады.

Құрамдас операторлар бұл фигуралы жақшалар {} арасында орналасқан кодтар блогы. Фигуралық жақша қолдана отырып, мәтінді форматтайтын екі тәсіл бар:

Біріншісінде тәсілде жақша бір бірінің астына жазылады, ал олардың ортасында орналасқан мәтін оң жаққа 1-2 (кейде одан да көп) символға жылжытылады.

Мысалы:

```
оператор
{
  жай немесе құрамдас операторлар тізбегі
}
```

Екіншісінде ашылатын фигуралық жақша құрамдас оператор басталатын жолға жазылады және ол келесі жолға тасмалданбауы керек, ал жабылатын фигуралық жақша бірінші сөздің астында болуы қажет. Мысалы:

```
оператор
{
қарапайым немесе композициялық операторлар тізбегі
   }
```

Фигуралық жақшадан кейін Java ережелері бойынша, /C ++-тағыдай «;» символын қою қажет емес. Бірақ оны фигуралық жақшаданг кейін программадағы ештеңе істемейтін бос операторларды қоятын орындарға қоюға болады.

Шартты оператор *If. If* шартты операторлың екі түрі бар: ***if*** және ***if- else***.

Бірінші түрі:

if(шарт)

оператор1;

Егер шарт==*true* болса, онда оператор1 орындалады. Егер де шарт==*false* болса, операторда ешқандай әрекет орындамайды.

Екінші түрі:

if(шарт)

оператор1;

else

оператор2;

if операторының бұл нұсқасында, егер шарт==false, онда оператор2 орындалады. Мәттің форматталуына ерекше мән беріңіз. Мысалы:

if(a<*b*)

a=*a*+1;

else if(*a*==*b*)

a=*a*+1;

else{

a=*a*+1;

b=*b*+1;

};

Бұл ережеден ерекше жағдай бар: егер бір жолға азаятын, қатарынан көп ***if*** операторлары келсе, программалардың жеңіл оқылуын арттыру үшін операторлардың басқа бөліктерін бөлек жолдарға түсірмеген жөн. ***if*** операторында шарттан соң тұратын орындалу облысында, сонымен қатар ***else*** облысында операторлар тізбегі емес, тек бір оператор тұру керек екенін атап кету керек. Сондықтан келесідей операторды жазуға

if(шарт)

оператор1;

оператор2;

else

оператор3;

жол берілмейді.

Мұндай жағдайда фигуралы жақшалармен шектелген құрамдас

оператор қолданады. Олардың арасында, біз білетін, саны кездейсоқ операторлар орналасуы мүмкін:

if(шарт){
оператор1;
оператор2;
}
else
оператор3;

Егер де біз келесідей жазсақ:

if(шарт)
оператор1;
else
оператор2;
оператор3;

Компилятор қателігін ешқандай диагностика бермейді. Оператор3-тің бұл жағдайда else шартына ешқандай қатысы жоқ – мәтінді форматтау логикалық қателікке итермелейді. Компиляция кезінде алдындағысына эквивалентті келесі программаның мәтінін форматтауда, оператор3 ***else***:

if(шарт)
оператор1;
else
оператор2;
оператор3;

бөлігіне қатысы жоқ екендігі айқын көрінеді.

Оператор3 ***else*** бөлігіне қатысты болу үшін келесі құрамды операторды қолдану керек:

if(шарт)
оператор1;
else{
оператор2;
оператор3;
};

Операторлың мына түріндегі: ***if***(шарт1) ***if***(шарт2) оператор1

else оператор2; тізбегінде бар ***else*** соңғы ***if***-ке қатысты болады, сондықтан мәтінді былай форматтау дұрысырақ

if(шарт1)
if(шарт2)
оператор1;
else
оператор2;

Сонымен, егер сәйкес ***if*** және ***else*** бөліктерін бірінің астына бірін жазатын болса, программа жұмысының логикасы анық болады.

Таңдау операторы ***switch*** бірнеше шартты таңдау үшін ***if*** аналогты болады. Операторлардың синтаксисі келесідей:

switch(өрнек){
case мағына1: операторы1;
……
case мағынаN: операторы N;
default: операторы;
}

Бірдей операторлар сәйкес болатын мәндер диапазонын көрсетуге және мәндерді үтір арқылы тізуге болмайтындығы, шындығында, ыңғайсыз. Өрнектің түрі қандай да бір бүтін түрден болу керек. Анығында, нақты түр қолжетімсіз болады. Оператор келесідей жұмыс істейді: алдымен өрнек есептелінеді. Сосын есептелінген мән программаның компиляциялау кезінде анықталған мән нұсқасымен салыстырылады. Егер өрнек мәні қанағаттандыратын нұсқа табылса, онда осы нұсқаға сәйкес операторлар тізбегі орындалады, онан кейін ***case*** операторынан шығу ОРЫНДАЛМАЙДЫ. Мұндай шығу үшін ***break*** операторын қою керек. Java-ның бұл жағымсыз ерекшелігі C тілінен еншіленген. Егер бірде бір нұсқа табылмаса, ***default*** бөлігі міндетті емес және орындалады. Мысалы:

switch(*i*/*j*){

```
case 1:
i=0;
break;
case 2:
i=2;
break;
case 10:
i=3;
j=j/10;
break;
default:
i=4;
};
```

switch операторында 2 ерекшелік бар:

1. Әрбір ***case*** нұскасы үшін операторлардың кез келген санын жазуға болады, ол ыңғайлы, бірақ Java тілінің операторлар логикасынан толықтай шығады;

2. Орындалып жатқан операторлар тізбегінен шығу ***break*** оператор көмегімен жүзеге асырылады. Егер ол жоқ болса, өрнек мағынасы сәйкес келген нұсқаның опреатор блогы «сәтсіздікке» ұшырайды. Мұнда келесі мәнге сәйкестікке ешқандай тексеріс жасалмайды. Солай ***break*** операторы кездескенше немесе таңдалған нұсқаларда барлық операторлар бітпейінше жалғаса береді. Мұндай тексеру ережесі «ұмытылған ***break***» деген қатені тудырады.

Қайталау (цикл) операторы ***for*** (инициалдау блогы; цикл денесін орындау шарты; санауышты өзгерту блогы) оператор;

Инициалдау блогында анықталу облысы ***for*** опреаторымен шектелген локалды айнымалы беру операторлары үтір арқылы тізіледі. Сондай-ақ, циклдан тыс берілген айнымалыларға мәндер меншіктеуге болады. Бірақ инициалдау тек қана бір типті айымалы үшін болуы мүмкін.

Шарт блогында циклдің жалғасу шарты тексеріледі. Егер ол

орындалса, денін оператор атқаратын цикл денесі орындалады. Егер орындалмаса – цикл тоқтайды және программаның ***for*** операторынан кейінгі операторына қадам жасалынады. Әрбір орындалған цикл денесінен соң (кезекті цикл қадамы) санауышты өзгерту блогындағы операторлар орындалады. Олар үтір арқылы бөліну керек. Мысалы:

```
for(int i=1,j=5; i+j<100; i++,j=i+2*j){
...
};
```

Оператор ***for***-ның әрбір блогы міндетті емес, бірақ айыратын «;» таңбасы жазылуы қажет. Ең қолданымы көп ***for*** операторын пайдалану – 1-ге артатын немесе кемитін қандай да бір айнымалы мәндерін тізу үшін және осы мәндерді пайдаланатын операторлар тізбегін орындау.

Айнымалы циклдің санауышы деп аталады, ал операторлар тізімі – цикл денесі деп аталады. Мысал 1: тізбектеліп келіп жатқан сандардың қосындысын есептеу.

1-ден 100-ге дейін барлық сандардың қосындысын шығаратын цикл жазайық. Шешімді result айнымалысын сақтайық.

```
int result=0;
for(int i=1; i<=100; i++){
result=result+i;
};
```

III.4.7 Ескерту: Java тілінде циклда жиым және жинақ элементтерін тізу үшін ***for*** операторының арнайы пішімі жоқ. Соған қарамастан ***for*** операторы жиымның немесе жинақтың барлық элементтерін тізбектеп өңдейді.

Қосымшаның басты пішім компоненттерінің жиым элементтері болатын компоненттер қасиеттерінің мәндерімен диалогтарды кезектеп шығарудың мысалы:

```
java.util.List components=
```

```
java.util.Arrays.asList(this.getComponents());
    for (Iterator iter = components.iterator();iter.hasNext();) {
    Object elem = (Object) iter.next();
    javax.swing.JOptionPane.showMessageDialog(null,"Компонент: "+
    elem.toString()); }
```

Алғышартты цикл *операторы* ***while***

while(шарт)

оператор;

Әзірге шарт true мағынасын сақтап тұрғанда – циклде оператор орындалады, әйтпесе цикл әрекеті тоқтайды. Егер шарт алдын ала false болса, цикл бірден тоқтайды және цикл денесі бір ретте де орындалмайды.

Алғышартты ***while*** циклі ***for*** циклінің орнына қолданылады, егер оның шартының жалғасы жеткілікті күрделі болса. Мұнда ***for*** циклінен айырмашылығы, оның нақты формалды цикл санауышы жоқ және оны автоматты түрде өзгертпейді. Ол үшін программалаушы өзі жауап береді. Дегенмен ***while*** циклінің орнына ***for*** циклін де қолдануға болады. Көп программлаушылар ***for*** циклін қолдануды жөн көреді. Мысалы:

```
i=1;
x=0;
while(i<=n){
x+=i;//эквивалентті x=x+i;
i*=2;//эквивалентті i=2*i;
};
```

Дегенмен ***while*** операторында цикл шектерін анықтау үшін салыстыруда жылжымалы нүкте сандары қолданылса, жиі қателіктер жібереді. Бұл жылжымалы нүкте сандарының компьютерде дәл бейнеленбуіде.

Мысалы, егер ***for*** цикліне арналған бөлімде санауышты ***while***

операторының көмегімен нақты санды санауыш ұйымдастырса. Мысал: осындай циклды ұйымдастырғанда әдеттегі қателік төменде көрсетілген:

double *a*=…;
double *b*=…;
double *dx*=…;
double *x*=*a*;
while(*x*<=*b*){
…
x=*x*+*dx*;
};

Біз білетіндей, *a*-дан *b*-ға дейінгі интервалда бүтін санды қадам орныққанда бұл цикл тұрақсыздық жағдайға тап болады. Мысалы: *a*=0, *b*=10, *dx*=0.1 цикл денесі *x*=0, *x*=0.1, …, *x*=9.9 аралығында орындалады. Ал *x* = 10 жағдайында цикл денесі не орындалады, не орындалмайды. Мұның себебі, амалдардардың жылжымалы нүкте форматындағы сандарымен ақырғы дәлдікпен орындалуында. Екілік бейнелеуде *dx* қадамының шамасы 0.1-ден аздап айырмашылықта болады және әрбір циклде х-тің мәнінде жүйеленген қателіктер жинала береді. Сондықтан *x*=10 дәл мәні жетілмейді, *x* мәні бірде аздап кіші, немесе аздап көп болады. Бірінші жағдайда цикл денесі орындалады, ал екінші жағдайда – жоқ. Не 100 не 101 итерация (цикл денесінің орындалу саны) өтеді.

Соңғышартты цикл операторы ***do...while***

do
оператор;
while(шарт);

Егер шарт false мәнін қабылдаса, цикл тоқтайды. Цикл денесі шартты тексергенге дейін орындалады, сондықтан ол әрқашан кем дегенде бір рет орындалады. Мысалы:

int *i*=0;
double x=1;
do{

```
i++; // i=i+1;
x*=i; // x=x*i;
}
while(i<n);
```

Егер ***do…while*** оператор көмегімен нақты санды санауыш немесе типі ***float*** немесе ***double*** сандарды тең, тең емес екенін тексеретін циклі ұйымдастырылса, онда ***for*** және ***while*** циклдерінде сипатталғандай проблемалар пайда болады. Қажет жағдайда ақырсыз цикл (цикл денесінен үзілім операторы көмегімен шығумен) ұйымдастыруды келесі нұсқаны жиі пайдаланады:

```
do{
…
}
while(false);
```

Орындалу барысын өзгерту ***break*** және ***continue*** операторы.

Кейбір жағдайларда программаның орындалу барысын өзгерту керек болады. Дәстүрлі программалау тілінде осы мақсатпен ***goto*** операторы қолданылады. Java-да бұл қолданылмайды. Бұл мақсатпен ***break*** және ***continue*** қолданылады.

continue операторы тек ***while***, ***do***, ***for*** циклдеріне қолданылуы мүмкін. Егер есептеу ағымында ***continue*** операторы кездессе, онда ағымдағы операторлар (өрнектер) тізбегімен орындалуы тоқтатылады және басқару осы тізбекті қамтитын блоктың басына беріледі.

```
...
int x = (int)(Math.random()*10);
int arr[10]  = {....}for(int cnt=0;
cnt<10;cnt++) {
if(arr[cnt] == x) continue;
...}
```

Берілген жағдайда, егер *arr* массивінде *x*-ке тең мән кездесетін

болса, онда ***continue*** операторы орындалады және барлық операторлар блок соңына дейін өткізіліп жіберіледі, ал басқару цикл басына беріледі.

Егер ***continue*** операторы цикл операторы контекстінен тыс қолданылса, онда компиляция кезінде қате шығарады. Кірістірілген циклдерді қолданғанда ***continue*** операторы ауысу адресі ретінде осы операторлардың біреуіне қатысты тамғаға берілуі мүмкін.

Мысалды қарастырайын:

```
public class Test {
public Test() {}
public static void main(String[] args) {
Test t = new Test();
for(int i=0; i < 10; i++){
if(i % 2 == 0) continue;
System.out.print(" i=" + i);
} } }
```

Жұмыс нәтижесінде консольда шығатыны:

i=1 *i*=3 *i*=5 *i*=7 *i*=9

7-ші жолдағы шарттың орындауы кезінде операторлардың қалыпты орындалу тізбегі тоқтатылып, басқару цикл басына беріледі. Осылай, консольға тек қана тақ мәндер ғана шығарылатын болады.

break операторы ***continue*** операторы сияқты орындалу тізбегін өзгертеді, бірақ орындайды цикл басына бермейді, оны тоқтады.

```
public class Test {
public Test() {}
public static void main(String[] args) {
Test t = new Test();
int [] x = {1,2,4,0,8};
int y  = 8;
for(int cnt=0;cnt < x.length;cnt++) {
```

if(0 == x[***cnt***]) ***break***;
System.out.println("y/x = " + y/x[***cnt***]);
} } }
Консольге келесі нәтиже шығарылады:
y/x = 8
y/x = 4
y/x = 2

Сонымен қатар нөлге бөлумен байланысты қате шықпайды, өйткені егер массивтің элемент мәні 0-ге тең болса, онда 9-шы жолдағы шарт орындалады да, ***for*** циклының орындалуы тоқтайды.

Break аргументі ретінде белгі берілуі мүмкін. ***Continue*** жағдайы сияқты аргумент ретінде ***break*** операторы жоқ блоктар белгісін қолдануға болмайды.

Мысалдар III.4.7:

Берілген формулалар бойынша үшбұрыштың ауданын оның қабырғалардың белгілі мәндері арқылы есептейтін программа құру:

$$p = \frac{a+b+c}{2};$$

$$s = \sqrt{p(p-a)(p-b)(p-c)}$$

мұнда a, b, c – қабырғалары, p – периметр жартысы, s – үшбұрыш ауданы.

Java тілінде қажетті программа келесі түрде болады:

```
package kz.enu.inf;
import static java.lang.Math.sqrt;
import static java.lang.Math.pow;
public class Calculate {
   private int a = 20;
   private int b = 40;
   private int c = 50;
public void Find() {
        double p, s;
        p = (a+b+c)/2;
```

```
        s=sqrt(p*(p-a)*(p-b)*(p-c));
        System.out.println("s=" + s);
    }
     public static void main(String[] args) {
     Calculate obj = new Calculate();
        obj.Find();
     }
}
```

Тапсырмалар III.4.7:

1 Қиылған конустың толық бетінің ауданы мен көлемін берілген формулалар бойынша есептейтін программа құру:

$$S = (R + r)\lambda + R^2 + r^2;$$

$$V = \left(R^2 + r^2 + Rr\right)h / 3,$$

мұнда $R = 20$ см, $r = 15$ см, $\lambda = 13$ см, $h = 12$ см.

2 $A(x_1, y_1)$ және $B(x_2, y_2)$ нүктелерін қосатын AB кесіндісін $m:n$ қатынасында кесетін нүкте координаталарын берілген формулалар бойынша есептейтін программа құру:

$$x = \frac{x_1 + kx_2}{1 + k};$$

$$y = \frac{y_1 + ky_2}{1 + k},$$

мұнда $k = m / n.$

Сұрақтар III.4.7:

1. Java тілінің басқа программалау тілдерінен негізгі айырмашылықтарын атаңыз?
2. Айнымалыны жариялау кезінде қандай қасиеттері көрсетіледі?
3. Көбейту, бөлу, қосу және алу амалдарының приоритеттері қандай?

Тесттер

1. Шартты өту оператор қай нұсқалары дұрыс?

A) ***if*** (*i<j*) { System.out.print("-1-"); }
B) ***if*** (*i<j*) then System.out.print("-2-");
C) ***if*** *i<j* { System.out.print("-3-"); }
D) ***if*** (*i<j*) System.out.print("-5-");
E) ***if*** {*i<j*} then System.out.print("-6-").

2. Идентификаторлар ішінде қайсысы дұрыс?
A) 2***int***;
B) ***int***_#;
C) _***int***;
D) _2_;
E) $***int***;
6) #***int***.

3. Мына программалық кодты

```
public class Quest6 {
        public static void main (String [] args) {
                System.out.print (“A”);
                main (“java7”);
        }
        private static void main (String args) {
                System.out.print (“B”);
        }
    }
```

орындауға және компиляцияға жібергенде қандай нәтиже алуға блолады?

A) компиляция қате шығарады
B) BA
C) AB
D) AA
E) компиляция сәтті өтеді, орындау кезінде программа шексіз қайталана береді.

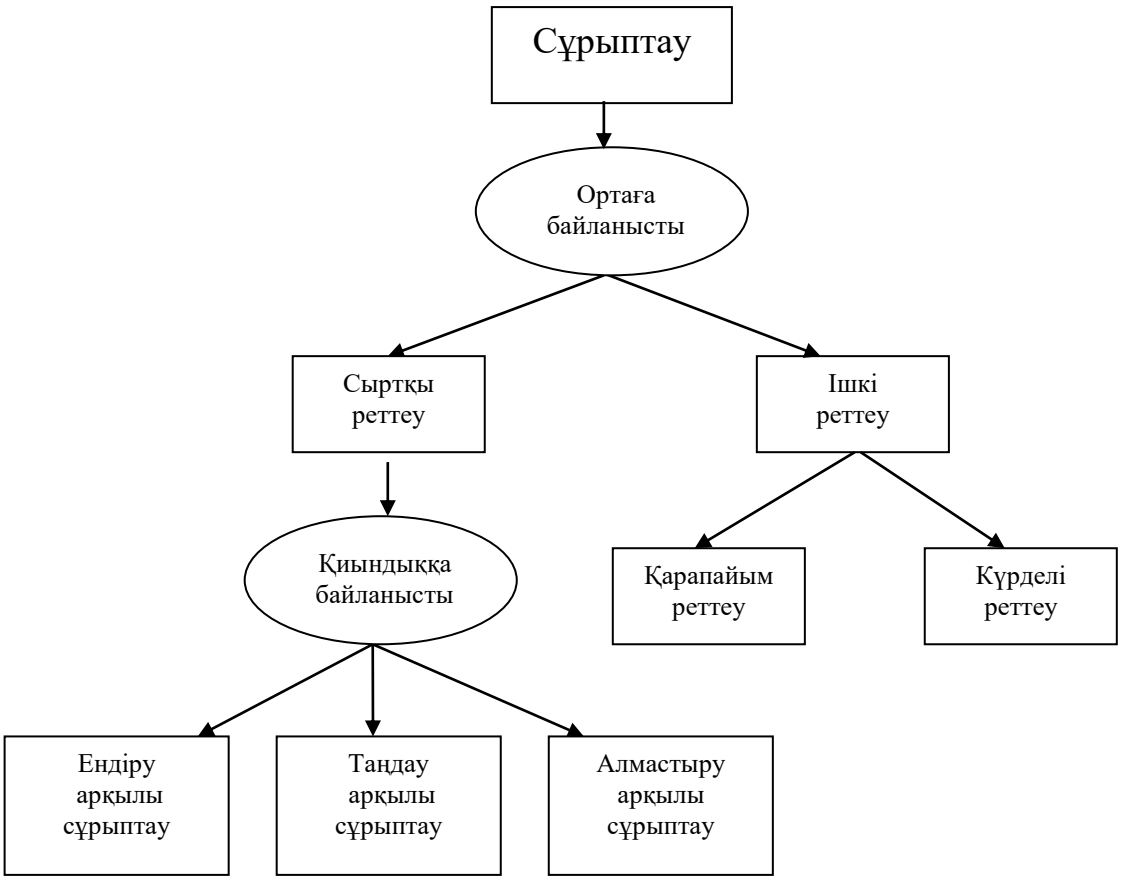
IV. ИНФОРМАТИКАНЫҢ АҚЫЛДЫ ҚАМТЫМЫ

IV.1. Сұрыптау әдістері

Түйін сөздер: *жазба, өріс, кілт, жиым, сұрыптау, сұрыптау әдістері, сұрыптау алгоритмдері, қиындық, деректер құрылымы, рет, кему, өсу, іздеу, тиімділік өлшемі, ендірумен сұрыптау, таңдаумен сұрыптау, алмастыру арқылы сұрыптау, элементтің кілті, элементтің индексі.*

Мақсат: *сұрыптау әдістерін сипаттау, сұрыптаудың алгоритмдерін құру, сұрыптау алгоритмдерінің қиындығын бағалау, олардың жетістіктері мен кемшіліктерін көрсету.*

Құрылымы: Сұрыптаудың түрлері IV.1-суретте көрсетілген.



IV.1-сурет. Сұрыптау түрлері

IV.1.1. Сұрыптау ұғымы

Сұрыптау – қандай да бір деректер құрылымының элементтерін өсу немесе кему реті бойынша орналастыру үшін олардың орнын ауыстыру. Сұрыптау керекті деректерді кез келген жерде іздегенде қажет болады, мысалы, анықтамаларда, сөздіктерде, энциклопедияларда, архивтерде және т.б.

Сонымен деректер құрылымын сұрыптаудың негізгі мақсаты қажетті элементті іздеу жұмысын жеңілдету болып табылады.

Сұрыптау есебін жалпы түрде былай анықтауға болады: бір типті жазбалар – жиым элементтері беріледі; олардың бір өрісі – мәні кілт болады, кілттің типі салыстыру ("=", ">", "<", ">=", "<=") амалын қамту керек; сұрыптау есебі алғашқы тізбекті осы жазбаларды қамтитын тізбекке түрлендіру, бірақ олардың кілттерінің мәндері өсу (немесе кему) бойынша орналасуы керек.

Информатикада сұрыптау әдісі өте маңызды болып табылады. Сонымен қатар, сұрыптау болашақтағы ақпараттық технологиядағы алгоритмдерді құрудың көптеген әдістерімен байланысты. Сондықтан сұрыптаудың әдістері мен алгоритмдері информатика мен ақпараттық технологияны үйренудің негізгі бөлігі болып табылады.

Сұрыптау әдістері мен алгоритмдерінің бірнеше түрлері бар. Әрбір сұрыптау программа болып табылады. Сондықтан программалау әдісіне байланысты екі сұрыптау алгоритмдерінің арасындағы айырмашылық бір ғана сұрыптау алгоритмін жүзеге асыратын және бір программалау тілде жазылған екі түрлі программалар арасындағы айырмашылықтан бірнеше рет аз болуы мүмкін. Бұның сондай–ақ, бір ғана алгоритмнің төменгі деңгейдегі және жоғарғы деңгейдегі программалау тілдеріндегі әртүрлі нұсқаларының өнімділігіне де қатысты.

Сұрыптаудың жалғыз ғана тиімді алгоритмі жоқ, себебі оның тиімділігін бағалайтын бірнеше параметрлер бар: *уақыт*, *жад*, *тұрақтылық*.

Сұрыптау әдістері екі үлкен топқа бөлінеді: *ішкі сұрыптау* және *сыртқы сұрыптау*.

Ішкі сұрыптау – бұл әдіс оперативтік жадқа толығымен орналасқан және элементтеріне емін–еркін қолжетімі бар деректер құрылымына (мысалы, жиымға) ғана қолданылады.

Сыртқы сұрыптау – бұл әдіс үлкен көлемді себепті оперативтік жадқа сыймайтын, сыртқы жадта орналасқан және элементтеріне емін–еркін емес, тек тізбекті қолжетімі бар деректер құрылымдарына (мысалы, файлдарға) қолданылады.

Сыртқы сұрыптау әдісінде файлдың белгілі бір бөлігі оперативтік жадта сұрыпталып, содан соң ғана сыртқы жадқа көшіріледі. Бұл үдеріс файл толық сұрыпталып болғанша бірнеше рет қайталанады. Сыртқы сұрыптаудың алгоритмдік мәселесіне қарағанда, қолданылатын жүйеге байланысты техникалық мәселері көбірек. Сондықтан біз сыртқы сұрыптауды қарастырмаймыз.

Сұрыптаудың ішкі әдісіне қойылатын негізгі талап, ол пайдалатын жад көлемін үнемдеу болып табылады. Бұл дегеніміз элементтердің орнын ауыстыруды көмекші деректер құрылымын пайдаланбай өз орнында жасау деген сөз. Сондықтан қосымша деректер құрылымын қажет ететін әдістермен жұмыс істемейміз.

Жад көлемін үнемдеу қағидасы бойынша сұрыптау әдістерінің сыныпталуы уақыт пен жылдамдықты үнемдеумен сәйкес келеді. *Тиімділік өлшем*і сұрыпталатын *n* элементтер функциясымен берілетін кілттерінің қажетті *C салыстырым санымен* және элементтердің *M алмастырым санымен* анықталады.

Әдетте сұрыптаудың қиын әдістеріне қарағанда, оңай әдістер көп салыстырымдарды қажет етеді. Оңай әдістердің мынадай қасиеттері бар:

– оңай әдістер сұрыптаудың көптеген прінсіптерінің қасиеттерін талдауға қолайлы;

– оңай әдістер қиын әдістерге қарағанда, әлдеқайда, оңай операциялар арқылы жүзеге асады;

– оңай әдістегі алгоритмдер қысқаша, әрі түсінуге оңай;

– оңай әдістегі программалар аз ғана операциялардан тұрады және программаның мәтінін сыйғызу үшін жадтың аз ғана мөлшерін қажет етеді;

– сұрыпталатын элементтердің саны n аз болғанда оңай әдіс тез жұмыс істейді, бірақ бұл әдісті n үлкен болғанда қолдануға болмайды.

Сұрыптаудың үш прінсіпі бар:

- *ендірумен сұрыптау*;
- *таңдаумен сұрыптау*;
- *алмастырумен сұрыптау*.

Ары қарай осы үш прінсіпті талдаймыз. Ол үшін мынадай белгілеулер енгіземіз:

1) Сұрыпталатын деректер құрылымы ретінде бір өлшемді сандар жиымын аламыз.

2) Сұрыпталатын жиымды a арқылы белгілейміз, ал оның элементтері мына түрде болады: $a[1], a[2], ..., a[n]$, мұндағы тік жақшалар ішінде элементтердің *индекстері* көрсетілген.

Осы a жиымын сұрыптау берілген сұрыптау функциясы F бойынша n–орынды $F(a[k_1]) \le F(a[k_2]) \le ... \le F(a[k_n])$ қатынасы орындалатындай мынадай орын ауыстыру $a[k_1], a[k_2], ..., a[k_n]$ түрінде болады:

Жалпы айтқанда, берілген $1 \le i \le n$ және $1 \le k_i \le n$ индекстері бойынша $a[i] \ne a[k_i]$ болуы мүмкін және "$\le$" белгісінің орнына "$\ge$" белгісін қолдануға болады.

Әдетте, сұрыптау функциясы арнайы бір ережемен анықталмайды, ол әрбір элементте мағынасы элементтің кілті деп аталатын нақты компонент ретінде болады. Яғни, F сұрыптау функциясын индексті $a[k_1], a[k_2], ..., a[k_n]$ айнымалыларға мәндер меншіктейтін функция ретінде есептеуге болады.

3) Барлық алгоритмдер VI–да қарастырылған *Pascal* тілінің қысқаша үлгісінде жазылады. Онда элементтерді, кілттерді, жиымды және индекстерді бейнелеу үшін *item*, *key*, *a* және *index* деген деректер типі қолданылады. Олар төмендегідей анықталады:

```
type  item = record key : integer;
 type  index = 0.n;
 var  a : array[1. n] of  item
end
```

IV.1.1. Мысалдар:

1. Жиымның көпіршіктік сұрыптауда көрші тұрған екі элементі қарастырылады: егер кіші индексті элементтің мәні индексі үлкен элемент мәнінен үлкен болса, онда олардың орындарын ауыстырамыз. Осындай қосақ бар болса бұл әрекет қайталанады. Нәтижесінде жиым элементтері сұрыпталған болады. Алгоритмнің әрбір қадамында ең болмағанда бір элемент орнына келеді:

```
Program BubbleSort;
    var a : array[1.1000] of integer;
    n, i, j, p : integer;
    begin
    for i:=1 to n do
        for j:=1 to n–i do
        if a[j]>a[j+1] then
            begin  {Элементтермен алмастыру}
                p:=a[j]; a[j]:=a[j+1]; a[j+1]:=p;
            end;
    end.
```

2.Жылдам сұрыптауда жиым элементтері бірінші бөліктің барлық элементтері екінші бөліктің кез келгенінен кіші болатындай етіп екіге бөлінеді. Содан кейін әрбір бөлік жеке сұрыпталады. Екіге бөлу жиымның қандайда бір элементіне қатысты, яғни, бұл саннан бірінші бөліктің барлық элементтері үлкен емес, ал екінші бөліктің барлық элементтері үлкен болуы керек.

```
 Program QuickSort;
var a  : array[1.1000] of integer;
n, t : integer;
procedure Sort(p, q : integer);
    {p,q – сұрыпталатын топтың басы мен аяғы}
        var i, j, r : integer;
      begin
            if p<q then
```

```
begin
    r:=a[p]; i:=p–1; j:=q+1;
    while i<j do
    begin
        repeat
            i:=i+1;
        until a[i]>=r;
        repeat
            j:=j–1;
        until a[j]<=r;
        if i<j then
        begin
            t:=a[i]; a[i]:=a[j]; a[j]:=t;
        end;
    end;
    Sort(p, j); Sort(j+1, q);
end;
end;
```

Екі индекс екі жақтан өз бөлігіне түспеген элементтті іздейді. Егер ондай элементтер табылса, онда оларды орындарымен ауыстырады. Қай элементте осы индекстер қиылысса, сол элемент бөліктердің айырмасы болады.

IV.1.1. Тапсырмалар:

1. Сұрыптау есебінің алғы шарттарын, орындалатын әрекеттерін және нәтижесін көрсетіңіз.
2. Ішкі сұрыптау мен сыртқы сұрыптаудың айырмашылықтарын көрсетіңіз.
3. Сұрыптаудың қандай жұмысты жеңілдететін және не үшін қажет екендігін көрсетіңіз.

Көмек:

1. Элементтері бір типті және олардың арасында реттеу

қатынасы бар деректер құрылымын қарастыру керек.

2. Ішкі сұрыптау жылдам, сыртқы сұрыптау баяу жұмыс істейтіндігі неден болатындығын білу керек.

3. Интернетте негізгі жұмыс деректерді сұрыптау екендігін еске түсіріңіз.

IV.1.1. Сұрақтар:

1. Сұрыптау дегеніміз не?
2. Сұрыптау функциясы қалай анықталады?
3. Сұрыптаудың қандай топтары бар?

***IV.1.1.* Тесттер:**

1. Сұрыптаудың қандай прінсіпі бар?

A) ендіру, таңдау, алмастыру арқылы сұрыптау;
B) шығару, айырбас, бөлу арқылы сұрыптау;
C) талдау, таңдау, қысу арқылы сұрыптау;
D) ендіру, алмастыру, жинастыру арқылы сұрыптау;
E) салыстыру, біріктіру, таңдау арқылы сұрыптау.

2. Сұрыптаудың қандай түрлері бар ?

A) ішкі сұрыптау, сыртқы сұрыптау;
B) бөліп сұрыптау, біріктіріп сұрыптау;
C) баяу сұрыптау, жылдам сұрыптау;
D) тізбекті сұрыптау, параллель сұрыптау;
E) тік сұрыптау, көлденең сұрыптау.

3. Сұрыптаудың тиімділік өлшемі немен анықталады?

A) қарастыру санымен, ығыстыру санымен;
B) енгізу санымен, шығару санымен;
C) қайталау санымен, тізбектеу санымен;
D) элемент санымен, уақыт санымен;
E) салыстырым санымен, алмастырым санымен.

IV.1.2. Ендірумен сұрыптау

Ендірумен сұрыптаудың негізгі идеясы қарапайым: қандайда бір элемент таңдалынады, басқа элементтерді сұрыптайды, таңдалған элементті «ендіреді», яғни оны басқа элементтер арасындағы орнына қояды.

Элементтерді Ендірумен сұрыптау әдісінің мағынасы мынадай болады:

1) Сұрыпталатын ***a*** жиымын *алғашқы тізбек* $a[1], a[2], \ldots, a[n]$ және *дайын тізбек* $a[1], a[2], \ldots, a[i-1]$ деп екі тізбекке бөлеміз;

2) *сұрыптау* қадам жасау арқылы жүзеге асады, $i=2$ бастап әрбір қадамда i–дің *мәнін* бірге арттыра отырып, алғашқы тізбектен i-ші элементті алып дайын тізбектегі қажетті орынға апарып қояды.

Дайын $a[1], a[2], \ldots, a[i]$ тізбегіндегі қажетті орынды іздеу кезінде салыстырымды орын ауыстырумен кезектестірген ыңғайлы болады. Яғни, көмекші айнымалы x-ті, кезектегі белгілі элемент $a[i]$-мен салыстыра және x-ті орнына қоя отырып, немесе $a[i]$-ді оңға жібере және солға жылжи отырып, «сүзеді».

«Сүзу» екі жағдайда аяқталатынын ескерген жөн. Олар:

1) х–тің кілтіне қарағанда кілтті кіші $a[i]$ элементі табылды.

2) дайын тізбектің сол жақ шетіне қол жетті.

Ендірумен сұрыптаудың алгоритмі келесі болады:

```
for t:=2 to n do
    begin x:=a[i];
      « x–ті  a[1], a[2],..., a[i] тізбегіндегі өз орнына қою »
    end
```

Бұл екі шарт арқылы аяқталатын циклдің, мысалы, бізге тосқауыл ретінде баламалы (фиктив) элемент әдісін қарастыруға мүмкіндік береді. Бұл жағдайда тосқауылды $a[0] = x$ деп алып оны оңай қолдануға болады. Мұнда *a* үшін индекс 0-ден басталады, яғни *а-ны* сипаттағанда индекстер аралығын *0-ден n-ге* дейін кеңейту керек.

Ендірумен сұрыптаудың нақты алгоритмі IV.1.2.1 - программа

түрінде берілген.

```
program jear;
var i,j : index; x : item;
begin
    for i:=2 to n do
        begin  x:=a[i]; a[0]:=x; j:=i–1;
            while x.key < a[j].key do
                begin a[j+1]:=a[i]; j:=j–1;
                end ;
            a[j+1]:= x
        end
end.
```

IV.1.2.1- программа. Ендірумен сұрыптаудың алгоритмі.

Ендірумен сұрыптауге талдау. Егер барлық n кілттерді орын ауыстыру орташа тең ықтималды $i/2$-ге тең болса, онда 1-ші сүзу кезінде кілттердің C_i салыстырым санының ең үлкені i–1, ал ең кішісі 1 болады. Алмастырым (меншіктеу) саны M_i, тоскауылды есептегенде, C_i+2 -ге тең. Салыстырым мен алмастырымның жалпы саны IV.1.2.2-кестеде көрсетілген.

IV.1.2.2-кесте. Салыстырым мен алмастырымның жалпы саны.

Салыстырым саны	**Алмастырым саны**
$C_{\min} = n-1$	$M_{\min} = 2(n-1)$
$C_{mid} = (n^2+n-2)/4$	$M_{mid} = (n^2+9n-10)/4$
$C_{\max} = (n^2+n)/2-1$	$M_{\max} = (n^2+3n-4)/2$

Мұнда *min* – ең кіші, *mid* – орта, *max*– ең үлкен дегенді білдіреді. Егер элементтер басынан сұрыпталған болса, ең кіші сан, ал егер элементтер кері ретте орналасса, онда ең үлкен сан пайда болады. Осы алгоритм тұрақты сұрыптауды сипаттайтыны түсінікті: ол кілттері бірдей элементтердің ретін өзгертпейді.

Жаңа элемент қосатын дайын тізбектің $a[1]$, $a[2]$, ..., $a[i–1]$

сұрыпталғандығын пайдаланып Ендірумен сұрыптау алгоритмін оңайлатуға болады. Өйткені ендіру орнын табу оңайға түседі. Мұнда дайын тізбектен ортадағы элементті тауып алып, оны ендіру орны табылғанша жартыға бөлетін бинарлық іздеу әдісін қолдануға болады.

Бинарлық ендірумен сұрыптауды талдау. Ендіру орны табылғаны, егер $a[l].key \le x.key < a[r].key$ орындалса. Іздеу аралағы соңында *1-ге* тең болуы керек, ол *i* кілттері бар аралық $[\log_2 i]$ рет жартыға бөлінуі керек дегенді білдіреді. Сонымен, $C = \sum_{i=1}^{n} [\log_2 i]$. Осы қосындыны интеграл арқылы жуықтап табайық

$$\int_1^n \log x dx = x(\log x - c)\Big|_1^n = n(\log n - c) + c,$$

мұнда $c = \log e = 1/\ln 2 = 1.45269\text{K}$.

Салыстырым саны элементтердің алғашқы ретіне тәуелді емес. Бірақ, іздеу аралығын бөлгенде жуықтау салдарынан, салыстырымның нақты саны күткен нәтижеден 1-ге үлкен болады. Осы ауытқушылықтың табиғаты мынада: нәтижесінде төменгі жақтағы ендіру орындары, жоғарғы жақтағыларға қарағанда бірталай тезірек табылады. Бұл алғашқыда элементтер дұрыс реттен алыс болған жағдайда тиімді болады. Турасында, егер алғашқыда элементтер кері ретпен орналасса, онда салыстырымның ең кіші саны талап етіледі, ал олар дұрыс ретпен орналасқан жағдайда салыстырымның ең үлкен саны шығады. Сондықтан, бұл сұрыптау алгоритмінің *табиғи емес* тәртібі:

$$C = n(\log n - \log e \pm 0.5).$$

Өкінішке орай, бинарлық іздеу әдісін қолдану арқылы алынған тиімділік, қажетті алмастырым санына емес, тек салыстырым санына қатысты. Нақтысында элементтердің орын алмасуы, әдетте екі кілттің салыстырымына қарағанда, әлдеқайда көп уақытты талап етеді. Сондықтан бұл тиімділік еш уақытта шешуші емес: маңызды көрсеткіш ***M*** бұрынғыдай n^2 ретінде қала береді. Расында, сұрыпталып қойған жиымды қайта сұрыптау, тізбекті іздеумен

Ендірумен сұрыптаудан гөрі көп уақыт алады.

IV.1.2. Мысалдар:

Кездейсоқ алынған сегіз бүтін сан үшін *Ендірумен сұрыптау үдерісі* кілттің өсу реті бойынша IV.1.2.1 - кестеде көрсетілген. Әрбір қадамда сұрыптау тәртібін бұзатын сандардың асты сызылады.

IV.1.2.1-кесте. Ендірумен сұрыптауге мысал

Бастапқы кілттер	**45**	**57**	**13**	**33**	**96**	**21**	**07**	**74**
i=2	***45***	**57**	**13**	**33**	**96**	**21**	**07**	**74**
i=3	**13**	**45**	**57**	**33**	**96**	**21**	**07**	**74**
i=4	**13**	**33**	**45**	**57**	**96**	**21**	**07**	**74**
i=5	**13**	**33**	**45**	**57**	**96**	**21**	**07**	**74**
i=6	**13**	**21**	**33**	**45**	**57**	**96**	**07**	**74**
i=7	**07**	**13**	**21**	**33**	**45**	**57**	**96**	**74**
i=8	**07**	**13**	**21**	**33**	**45**	**57**	**74**	**96**

IV.1.2. Тапсырмалар:

1. Элементті сүзу қандай жағдайда аяқталатынын көрсетіңіз.
2. Ендірумен сұрыптау әдісінің мағынасын ашыңыз.
3. Бинарлық сұрыптауды пайдаланатын жағдайды көрсетіңіз.

Көмек:

1. Сүзу екі жағдайда аяқталатынын ескерген жөн
2. Бұл әдісте алғашқы тізбек және дайын тізбек алынады.

3. Алғашқы тізбекті екіге бөлінетінін ескеру керек.

IV.1.2. Сұрақтар:

1. Ендірумен сұрыптау әдісін қолдану үшін қандай тізбектер керек?

2. Қандай жағдайда 1-ші сүзу кезінде кілттердің салыстырым санының ең үлкені $i-1$, ал ең кішісі 1 болады?

3. Сұрыптау әдісінде салыстырым саны элементтердің алғашқы ретіне тәуелді ме?

IV.1.2. Тесттер:

1. Бинарлық сұрыптау әдісін қолдану арқылы алынған тиімділік неге қатысты?

A) салыстырым санына;

B) алмастырым санына;

C) элементтер санына;

D) амалдар санына;

E) ендіру санына.

2. Сұрыптау алгоритмінде салыстырым санының қандай мәндері бар?

A) берілген, есептелген, кездейсоқ;

B) алғашқы, есептеу, нәтиже;

C) бірінші, ортаңғы, соңғы;

D) ең кіші, орта, ең үлкен;

E) өспелі, тұрақты, кемімелі.

3.Сұраптау алгоритмінде алмастырым санының қандай мәндері болады?

A) алғашқы, есептеу, нәтиже;

B) бірінші, ортаңғы, соңғы;

C) ең кіші, орта, ең үлкен;

D) өспелі, тұрақты, кемімелі;

E) берілген, есептелген, кездейсоқ.

IV.1.3. Таңдаумен сұрыптау

Таңдаумен сұрыптау әдісі мына ережелерге негізделген:

1. Ең кіші кілті бар элемент таңдалады.

2. Ол бірінші элемент *a*[*1*]-мен орын ауыстырады.

Бұл амалдар қалған *n–1* элементтер үшін, кейін *n–2* элементтер үшін, соңында бір ғана ең үлкен элемент табылғанша қайталананып жалғаса береді. Осы әдісті жүзеге асыратын алгоритмді келесі түрде жазуға болады:

for *i:=1* **to** *n–1* **do**

begin

«a[*1*],...,*a*[*n*] *ішіндегі ең кіші элемент индексін k–ға меншіктеу»*;

«a[*i*] *мен a*[*k*]*–ның орнын ауыстыру»*

end

Таңдаумен сұрыптау әдісі *Ендірумен сұрыптау* әдісіне қарама–қайшы болып табылады; қарапайым ендіру арқылы сұрыптаған кезде әрбір қадамда *берілген тізбектің* тек *бір ғана* кезектегі элементін қарастырады және ендіру орнын табу үшін *дайын тізбектің барлық* элементтері қарастырылады; қарапайым таңдау арқылы сұрыптағанда ең кіші кілті бар элементті табу үшін *берілген тізбектің барлық* элементтері қарастырылады және осы *бір* элемент *дайын тізбекке* жіберіледі.

Таңдаумен сұрыптауды талдау. Кілттердің салыстырым саны ***C*** кілттердің бастапқы ретіне тәуелсіз екендігі айқын. Осы тұрғыдан қарағанда бұл әдіс Ендірумен сұрыптау әдісінен қолайлырақ екенін байқаймыз. Кілттердің салыстырым саны:

$$C = (n^2 - n)/2$$

Кілттер алғашқыда тура реттелген болса, алмастырым саны ең кіші:

$$M_{\min} = 2(n-1)$$

Ал кілттер алғашқыда кері ретте орналасқан болса, алмастырым саны ең үлкен мән қабылдайды:

$$M_{\max} = trunc(n^2/4) + 3(n-1)$$

Алгоритмнің қарапайымдылығына қарамастан орта мәнді M_{mid} табу оңайға түспейді. Оны табу үшін k_1,., k_n тізбегін қарағанда алдыңғы k_1,., k_{i-1} элементтерден кіші болатын қанша k_i барын анықтау керек. Алмастырым санының орта мәні

$$M_{mid} = n(\gamma + 1) + \sum_{i=1}^{n} \ln i,$$

мұндағы γ == 0.577216 – Эйлер константасы.
Қосындыны келесі интегралдың көмегімен

$$\int_1^n \ln x dx = x(\ln x - 1)\Big|_1^n = n \ln n - n + 1$$

жуықтап, мынадай өрнек аламыз

$$M_{mid} = n(\ln n + \gamma)$$

IV.1.3. Мысалдар:

Таңдаумен сұрыптау әдісі бойынша нақты мысал сегіз кілтті тізбек үшін IV.1.3 - кестеде көрсетілген.

IV.1.3 - кесте. Қарапайым таңдау арқылы сұрыптауға мысал

Бастапқы кілттер	**45**	**57**	**13**	**33**	**96**	**21**	**07**	**74**
i=2	**07**	**57**	**13**	**33**	**96**	**21**	**45**	**74**
i=3	**07**	**13**	**57**	**33**	**96**	**21**	**45**	**74**
i=4	**07**	**13**	**21**	**33**	**96**	**57**	**45**	**74**
i=5	**07**	**13**	**21**	**33**	**96**	**57**	**45**	**74**
i=6	**07**	**13**	**21**	**33**	**45**	**57**	**96**	**74**
i=7	**07**	**13**	**21**	**33**	**45**	**57**	**96**	**74**
i=8	**07**	**13**	**21**	**33**	**45**	**57**	**74**	**96**

IV.1.3. Тапсырмалар:

1. Ең кіші кілтті элементтің мағынасын түсіндіріңіз.
2. Әрбір қадамда қанша элемент қарастырылады.

3. Кілттерді салыстыру саны неге тәуелділігін көрсетіңіз.

Көмек:

1. Дайын тізбекті еске алу керек.
2. Бұл тізбектің түріне тәуелді
3. Кілттің бастапқы ретін ескеру керек.

IV.1.3. Сұрақтар:

1. Сұраптау алгоритміндегі таңдаумен реттеу қандай ережеге негізделген?

2. Таңдаумен сұрыптауда ең кіші кілті бар элемент қалай табылады?

3. Таңдаумен сұрыптауда кілттердің салыстырым саны кілттердің бастапқы ретінде тәуелді ме?

IV.1.3. Тесттер:

1. Қай жағдайда алмастырым саны ең кіші болады?

A) алғашқыда кілттер тура реттелген болса;
B) алғашқыда кілттер кері реттелген болса;
C) алғашқыда кілттер сұрыпталған болса;
D) алғашқыда кілттер реттелмеген болса;
E) алғашқыда кілттер тура реттелмеген болса.

2. Кілттер кері ретте орналасқан болса не болады?

A) алмастырым саны ең үлкен мән қабылдайды;
B) алмастырым саны ең кіші мән қабылдайды;
C) алмастырым саны мән қабылдамайды;
D) алмастырым саны ең үлкен мән қабылдамайды;
E) алмастырым саны ең кіші болады.

3. Қай жағдайда алмастырым саны ең үлкен болады?

A) алғашқыда кілттер тура реттелмеген болса;
B) алғашқыда кілттер реттелмеген болса;
C) алғашқыда кілттер жартылай реттелген болса;
D) алғашқыда кілттер тура реттелген болса;

Е) алғашқыда кілттер кері реттелген болса.

IV.1.4. Ауыстыру арқылы сұрыптау

Ауыстыру арқылы сұрыптау әдісі жиымдағы барлық элементтер сұрыпталғанша көршілес екі элементті салыстыру мен ауыстыру прінсіпіне негізделген. Ол сұрыпталатын жиымдағы барлық элементтердің жеке салмақтары (кілттері) болуын және тік (вертикаль) орналасуын қалайды.

Ауыстырумен сұрыптау әдісі сұрыпталатын жиымды бірнеше рет қаралымнан өтеді және әрбір қаралымда қалған жиынның ең кіші элементі сүзіліп алынып өзінің кілтінің мәніне (салмағына) сәйкес деңгейге шығарылады. Яғни, әрбір қаралымда көпіршік (ең жеңіл элемент) пайда болады. Сондықтан, бұл әдіс кейде *көпіршік әдісі* деп аталады. Осы әдісті жүзеге асыратын алгоритм барлық тізбектегі элементтер сұрыпталып болғанша жұмыс істейді.

Көпіршік әдісімен сұрыптау IV.1.4.1-программада көрсетілген.

```
procedure jaar;
    var i, j: index; x: item;
    begin for i:=2 to n do
        begin for j:= n downto i do
            if a[j-1].key > a[j].key then
            begin x:=a[j-l]; a[j-1]:=a[j]; a[j]:= x
            end
        end
    end
```

IV.1.4.1-программа. Көпіршік әдісімен сұрыптау

Алгоритм жұмысын жеңілдету үшін қандай ауыстырулар жүргізілгенін есте сақтап отыру керек. Егер ауыстырулар болмаса, онда алгоритм жұмысын аяқтай алады деген сөз. Ауыстырулар орындалған жағдайда соңғы ауыстырудың орнын (индексін) *k-ны* бақылау керек. Себебі осы *k* индексінен кіші индексті элементтер жұбы сұрыпталып қойғандығы айқын. Сондықтан келесі қарауда бастапқыда белгіленген төменгі шекара *i-ге* жетпей-ақ, осы *k* индексіне келгенде жұмысты тоқтатуға болады.

Көпіршік әдісі арқылы сұрыптау алгоритмін талдағанда мынадай *ассиметрия* бар екендігін ескерген жөн болады: сұрыпталған тізбектің «ауыр» жағының соңында орналасқан бір «жеңіл» элемент (көпіршік) өзінің орнын тек бір қаралымда табады, ал «жеңіл» жағының алдында орналасқан бір «ауыр» элемент әрбір қаралымда бір ғана қадам арқылы өз орнына келеді. Мысалы, көпіршік әдісі арқылы сұрыптағанда мына 21 33 45 57 74 96 07 тізбегі бір ғана қаралымда сұрыпталады да, ал мына 96 07 13 21 33 45 57 74 тізбегін сұрыптау үшін жеті қаралым қажет.

Бұл ассиметрия көпіршік әдісі арқылы сұрыптауды жетілдіруге болатындығын көресетеді. Ол үшін әрбір келесі қаралымда көпіршік табу үшін бағытты (жоғары, төмен) өзгертіп отыру керек. Осылай пайда болған алгоритмді *шейкер-сұрыптау* деп атайды, ол IV.1.4.2-программада көрсетілген.

```
procedure shakersort;
    var j,k,l,r: index; x: item;
    begin l:=2; r:=n; k:=n;
        repeat
            for j:=r downto l do
                if a[j-1].key>a[j].key then
                begin x := a[j-1]; a[j-1]:=a[j]; a[j]:=x;
                k:=j
                end ; l:= k+1;
            for j :=l  to r do
                if a[j-1].key > a[j].key then
                begin x:=a[j-1]; a[j-1]:=a[j];a[j] :=x; k:=j
                end ;
            r:=k-1;
        until  l > r
    end
```

IV.1.4.2-программа. шейкер-сұрыптау.

Қазір жоғарыда көрсетілген көпіршік әдісі арқылы сұрыптау мен шейкер-сұрыптауға талдау жасайық.

Қарапайым ауыстыру (көпіршік) әдісіндегі салыстырым саны

$$C = (n^2 - n)/2,$$

ал ауыстырудың ең кіші, орта және ең үлкен сандары мынадай:

$$M_{\min} = 0,\ M_{mid} = 3(n^2 - n)/4,\ M_{\max} = 3(n^2 - n)/2.$$

Шейкер-сұрыптау әдісі үшін салыстырымның ең кіші саны

$$C_{\min} = n - 1$$

Тізбектің орта қаралым саны $n - k_1\sqrt{n}$ -ге пропорционал, ал салыстырымның орта саны $[n^2 - nk_2 + \ln n]/2$ -ге пропорционал.

Сұрыптау кезінде *n* элементтер ішіндегі әрқайсысының жылжитын орта қашықтығы *n/3*-ке тең. Бұл сұрыптаудың тиімді әдістерін жасау үшін тиек болады. Негізінде барлық қарапайым әдістер әр қадамда әрбір элементті бір орынға (позицияға) жылжытады. Сондықтан олар қадамдар саны n^2 ретінде болғанды талап етеді. Кез келген жақсарту бір циклда элементтерді үлкен қашыққа жіберу прінсіпіне негізделуі қажет.

Енді бұрын қарастылған барлық (ендірумен, таңдаумен, ауыстырумен) сұрыптау әдістерін өзара салыстырайық. Ол үшін осы әдістерге байланысты *салыстырым сандары* мен *алмастырым сандарының* ең кіші, орта және ең үлкен мәндерін есептеуеге мүмкіндік беретін формулаларды IV.1.4.1-кестеде көрсетейік.

IV.1.4.1-кесте. Қарапайым сұрыптау әдістерін салыстыру

Әдістер аты	**Ең кіші мән**	**Орта мән**	**Ең үлкен мән**
Қарапайым ендіру	$C_{\min} = n - 1$ $M_{\min} = 2(n-1)$	$C_{mid} = (n^2 + n - 2)/4$ $M_{mid} = (n^2 + 9n - 10)/4$	$C_{\max} = (n^2 + n)/2 - 1$ $M_{\max} = (n^2 + 3n - 4)/2$
Қарапайым таңдау	$C_{\min} = (n^2 - n)/2$ $M_{\min} = 3(n-1)$	$C_{mid} = (n^2 - n)/2$ $M_{mid} = n\ (\ln n + 0.57)$	$C_{\max} = (n^2 - n)/2$ $M_{\max} = n^2/4 + 3(n-1)$
Қарапайым ауыстыру	$C_{\min} = (n^2 - n)/2$ $M_{\min} = 0$	$C_{mid} = (n^2 - n)/2$ $M_{mid} = (n^2 - n) * 0.75$	$C_{\max} = (n^2 - n)/2$ $M_{\max} = (n^2 - n) * 1.5$

Осы көрсетілгендерден мынадай қорытынды шығады:

1. Қарапайым ауыстыру (көпіршік) әдісі барлық салыстырылған

әдістердің ішіндегі ең жаманы болады. Оның жетілдірілген түрі шейкер-сұрыптау да Ендірумен сұрыптау және Таңдаумен сұрыптау әдістерінен тиімсіздеу.

2. Таңдаумен сұрыптау осы әдістердің ішіндегі ең ұтымдысы.

Сонымен бұл талдаудан ауыстырумен сұрыптау ендірумен және таңдаумен сұрыптаудан анағұрлым қиын екендігі шығады.

IV.1.4. Мысалдар

1. Мейлі алғашқы тізбек 45 57 13 33 96 21 07 74 болсын. Осыны көпіршік әдісі арқылы сұрыптап, мынадай тізбек алу керек 07 13 21 33 45 57 74 96, ол үшін IV.1.4.2-кестеде көрсетілген сегіз қаралымды жасау керек.

IV.1.4.2-кесте. Көпіршік әдісімен сұрыптауға мысал.

Бастапқы кілттер	***i=2***	***i=3***	***i=4***	***i=5***	***i=6***	***i=7***	***i=8***
45	**07**	**07**	**07**	**07**	**07**	**07**	**07**
57	**45**	**13**	**13**	**13**	**13**	**13**	**13**
13	**57**	**45**	**21**	**21**	**21**	**21**	**21**
33	**13**	**57**	**45**	**33**	**33**	**33**	**33**
96	**33**	**21**	**57**	**45**	**45**	**45**	**45**
21	**96**	**33**	**33**	**57**	**57**	**57**	**57**
07	**21**	**96**	**74**	**74**	**74**	**74**	**74**
74	**74**	**74**	**96**	**96**	**96**	**96**	**96**

Көпіршік әдісін жетілдіру оңай. IV.1.4.2-кестеде соңғы үш қаралым элементтердің ретіне ешқандай әсер етпейді.

Мейлі алғашқы тізбек 45 57 13 33 96 21 07 74 болсын. Осыны *шейкер-сұрыптау* әдісі арқылы мәндердің өсуіне қарай сұрыптап, мынадай тізбек алу керек 07 13 21 33 45 57 74 96.

Ол үшін IV.1.4.2-программадағы *шейкер-сұрыптау* алгоритмін қарастырылған сегіз кілті бар тізбек үшін жұмыс істету керек. Осы алгоритмнің жұмыс нәтижесі IV.1.4.3-кестеде көрсетілген.

IV.1.4.3-кесте. Шейкер-сұрыптауға мысал.

i=2 ***r=8*** ↑	***i=3*** ***r=8*** ↓	***i=3*** ***r=7*** ↑	***i=4*** ***r=7*** ↓	***i=4*** ***r=4*** ↑
45	**07**	**07**	**07**	**07**
57	**45**	**45**	**13**	**13**
13	**57**	**13**	**45**	**21**
33	**13**	**33**	**21**	**33**
96	**33**	**57**	**33**	**45**
21	**96**	**21**	**57**	**57**
07	**21**	**74**	**74**	**74**
74	**74**	**96**	**96**	**96**

Мұнда алма-кезек үш рет жоғары және екі рет төмен бығыттарда қарау жасалғандығы байқалады.

IV.1.4. Тапсырмалар:

1. Ауыстырумен сұрыптау қандай прінсіпке негізделгенін және ол не қалайтынын көрсетіңіз

2. Берілген мына тізбекті 25 37 42 54 75 91 06 көпіршік әдісімен сұрыптағанда қанша қарау болатындығын анықтаңыз.

3. Берілген мына тізбекті 19 31 43 52 69 87 05 шейкер-сұрыптаумен сұрыптағанда қанша қарау болатындығын анықтаңыз.

Көмек:

1. Ауыстырумен сұрыптау көршілес екі элементтің салыстыруы

мен орын алмасуына прінсіпіне негізделген және ол берілген тізбекті тік (вертикаль) орналастыруды қалайды.

2. Құрылатын программа IV.1.4.1-программадағы Ауыстырумен сұрыптау алгоритміне негізделуі және оның жұмыс хатаммасы IV.1.4.2-кестедегі хаттамадай болуы керек.

3. Құрылатын программа IV.1.4.2-программадағы *шейкер-сұрыптау* алгоритміне негізделуі және оның жұмыс хатаммасы IV.1.4.3-кестедегі хаттамадай болуы керек

IV.1.4. Сұрақтар:

1. Ауыстырумен сұрыптау қандай прінсіпке негізделген?
2. Көпіршік әдісінде қандай ассиметрия бар ?
3. Көпіршік әдісінің алгоритмі қай жағдайда тоқтайды?

IV.1.4. Тесттер:

1. Шейкер–сұрыптау әдісі үшін салыстырымның ең кіші саны?
   A) $C_{\min} = n-1$;
   B) $C_{\min} = (n^2 - n)/2$;
   C) $M_{\min} = 2(n-1)$;
   D) $M_{\min} = 3(n-1)$;
   E) $M_{\min} = 0$.
2. Сұрыптау әдістерінің ішіндегі ең тиімдісі қайсысы ?
   A) Шейкер-сұрыптау әдісі;
   B) Ауыстырумен сұрыптау;
   C) Ендірумен сұрыптау;
   D) Таңдаумен сұрыптау;
   E) Барлық әдістер тиімді.
3. Сұрыптау әдістерінің ішіндегі ең жаманы қайсысы ?
   A) Таңдаумен сұрыптау;
   B) Ендірумен сұрыптау;
   C) Ауыстырумен сұрыптау;
   D) Шейкер-сұрыптау әдісі;
   E) Барлық әдістер жаман.

ӘДЕБИЕТТЕР

1. Colmeraurer J., Kanoui V., Prolog bases et developpements actuels, T.A.1,vol.2, №4, 1983, pp. 112–120.
2. McCarty J. A LISP programmer’s manual. –Cambridge, Mass.: M.T.T. Press, 1963.
3. Бауэр Ф. Гооз Г. Информатика –М.: Мир, том 1,2, 1990. – 742 с.
4. Вирт Н. Алгоритмы структуры данных = программы. –М.: Мир, 1985.
5. Грэхем Иан. Объектно-ориентированные методы. Принципы и практика. –3-е изд. –М.: Вильямс, 2004. – 880 с.
6. Кнут Д. Искусство программирования. Сортировка и поиск. –М.: Мир, 2000, т.3. – 822 с.
7. Лорин Г. Сортировка и системы сортировки. –М.: Наука, 1983, –384 с.
8. Пратт Т., Зелковиц М. Языки программирования. Разработка и реализация. 4-е издание. –М.:, С.П.: Питер, 2002. –688 с.
9. Турчин В. Программирование на языке Рефал. –М.: Препринт ИПМ АН СССР, 1971, № 41, № 43, № 44, №48.
10. Шәріпбаев А. Информатика. –Алматы: Қазақ университеті, 1992. –72 б.
11. Шәріпбаев А., Ыскакова А., Рахлетова А. Информатика пәнінен қазақша, орысша, ағылшынша түсіндірме сөздігі. –Алматы: Сөздік-Словарь, 2002. –156 б.
12. Шәріпбаев А. Информатика. –Астана: Нұржол, 2003. –168 б.
13. Шәріпбаев А. және басқалар СТ ҚР 1048 – 2002. Ақпараттық технологиялар. Қазақ әліпбиінің 8-битті кодтық кестесі.
14. Шәріпбай А. және басқалар. Информатика және есептеу техникасы. Казақша-орысша, орысша-қазақша сөздік. ҚР Үкіметінің жанындағы Республикалық терминологиялық комиссиясы мақұлдаған, КАЗақпарат, –Алматы:,2014,–452 б.

ЖАУАПТАР

Тапсырмалардың жауаптары

I.1.1. Тапсырмалар:

1. Табиғи тілдер;
2. Математикалық тіл;
3. Мимика тілі.

I.1.2. Тапсырмалар:

1. Бір хабар арқылы бір мазмұн береді;
2. Бір хабар арқылы бірнеше мазмұн береді;
3. Бірнеше хабар арқылы бір мазмұнды береді.

I.1.3. Тапсырмалар:

1. Көлемдік тәсіл және ықтималдық тәсіл;
2. 24 бит немесе 3 байт;
3. $2^8 = 256$.

I.1.4. Тапсырмалар:

1. 4 бит;
2. 6 бит;
3. 8 бит.

I.2.1. Тапсырмалар:

1. Идентификатор емес;
2. Идентификатор;
3. Идентификатор.

I.2.2. Тапсырмалар:

1. Тұрақты нүктелі нақты;
2. Комплекс сан;
3. Жылжымалы нүктелі нақты.

I.2.3. Тапсырмалар:

1. 1/6;
2. -16;
3. 15.

I.2.4. Тапсырмалар:

1. $10000001_2=129_{10}$;
2. $129_{10}=201_8$;
3. $1952_{10}=7A0_{116}$.

I.2.5. Тапсырмалар:

1. Қазақ әліпбиінің алғашқы бас әріптерінен құрылған тізбек.
2. Цифрлардан құрылған тізбек
3. Аралас таңбалардан құрылған тізбек

I.2.6. Тапсырмалар:

1. АЛАТАУ;
2. ЖЕР;
3. X ⊃ Y.

I.2.7. Тапсырмалар:

1. жалған;
2. жалған;
3. ақиқат.

I.2.8. Тапсырмалар:

1. 1;
2. 1;
3. 0.

I.3.1. Тапсырмалар:

1. Жолдар, векторлар, жиымдар;

2.Байланысқан тізімдер, графтар, дарақтар, стектер, кезектер, дектер;

3. Сызықтық, сызықтық емес байланысқан тізімдер.

I.3.2. Тапсырмалар

1. $L=\{a,b,c,d,e,f,h,g,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,y,z\}$;
2. $D = \{0,1,2,3,4,5,6,7,8,9\}$;
3. $X = \{x \mid x > 0 \ \& \ x \in N\}$.

I.3.3. Тапсырмалар:

1. Бұл теңдеудің түбірлерінің жиыны бос жиын болады.
2. *B* жиыны *A* жиынының ішжиыны.
3. A ⊂ B.

I.3.4. Тапсырмалар:

1. АСТАНА, мұндағы барлық әріптер қазақ әліпбиінен.
2. Информатикадығы жиым математикада матрицаға сәйкес.

Мейлі екі өлшемді $A=\begin{pmatrix}1 & 3\\ 5 & 7\end{pmatrix}$, $B=\begin{pmatrix}2 & 4\\ 6 & 8\end{pmatrix}$ матрицалары

берілсін. Осы екі матрица элементтерін қосу анықталған:

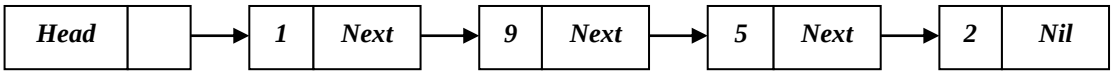
$$A+B=\begin{pmatrix}1 & 3\\ 5 & 7\end{pmatrix}+\begin{pmatrix}2 & 4\\ 6 & 8\end{pmatrix}=\begin{pmatrix}1+2 & 3+4\\ 5+6 & 7+8\end{pmatrix}=\begin{pmatrix}3 & 7\\ 11 & 15\end{pmatrix}$$

3. Кестенің алты бағаны болады, ал жолдар саны топтағы студенттер санына бірді қосқанға тең, себебі ол жолда бес бағанның атаулары жазыдады және ол мынадай болады:

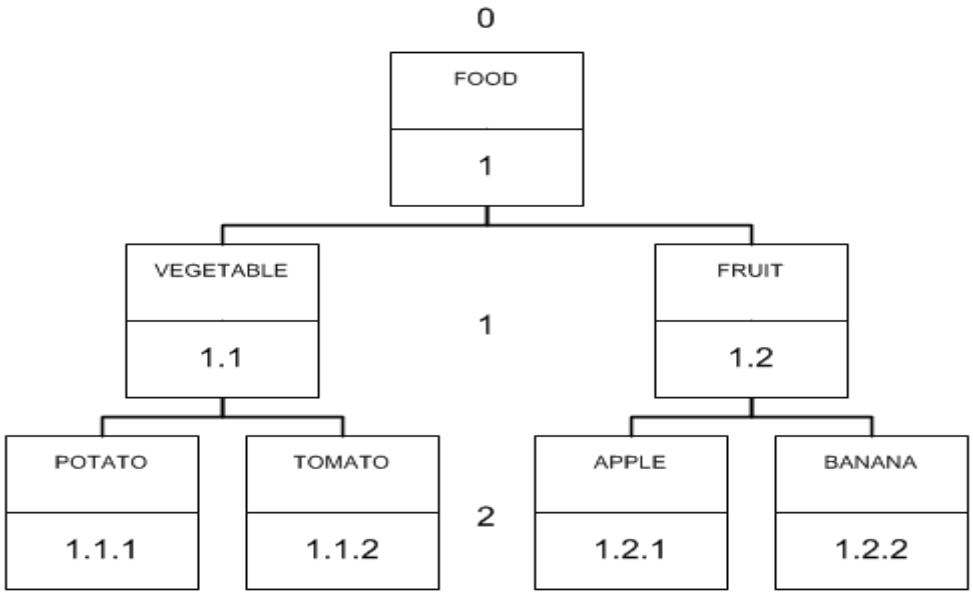
№	Тегі	Аты	Жасы	Жынысы	Ұлты

I.3.5. Тапсырмалар:

Алдымен Data атауымен қажетті элементтердің деректер типі өрісін қамтитын құрылымды анықтаймыз, содан кейін List атауымен Data өрісін және келесі элемент next адресін қамтиды. Бұл негізгі List құрылымын тиіспей Data құрылымды деректерімен өзгертуге мүмкіндік береді. Құрылатын бірбайланысты сызықтық тізімнің графикалық кескіні төмендегідей болады:

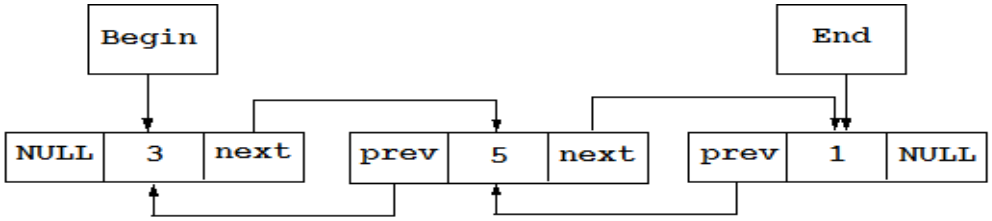


2. Иерархиялық тізімді бейнелеу және өңдеу үшін ең жоғарғы деңгейден ең төменгі деңгейге дейін жолды көрсететін тәсіл пайдаланады. Жол тереңдігі – иерахия деңгейі. Есептің графикалық шешімі төменде көрсетілген:



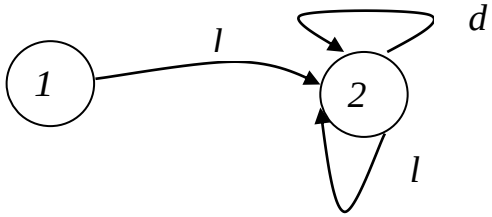
3. Екібайланысты тізімде әрбір элемент деректер өрісін және екі нұсқағышты қамтиды.: бір нұсқағыш алдыңғы элемент адресін, ал

екіншісі келесі элемент әдрірісін қамтиды. Тізімнің басы мен соңының адрестерін сақтайтын нұсқағыштар бар. Төменде екібайланысты тізімнің графикалық кескіні көрсетілген:



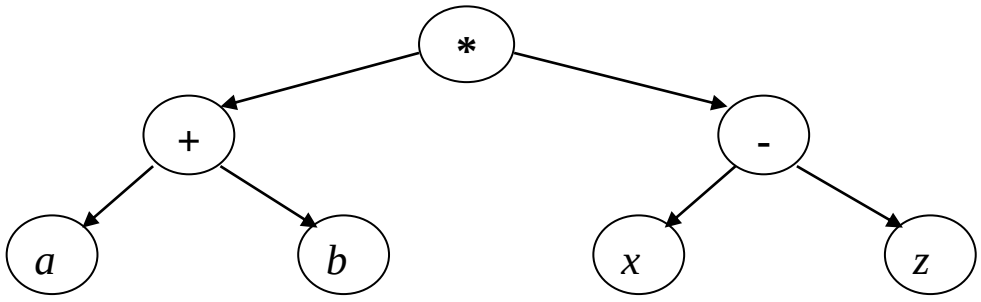
I.3.6. Тапсырмалар:

1. Егер *l* арқылы әріпті, ал *d* арқылы цифрды белгілесек, онда идентификатор құрылымы мына графпен сипатталады.



Мұндағы 1-ші төбеден 2-ші төбеге жүретін *l* салмақты идентификатордың әріптен басталатынын білдіреді. Ары қарай, 2-ші төбеден 2-ші төбеге *l* және *d* салмақты жолдар идентификатордың басқа орындарында тек әріп немесе тек цифр болатындығын және оның ұзындығы ақырсыз екендігін көрсетеді.

2. (*a* + *b*) * (*x* -*z*) өрнегі үшін дарақ былай салынады.



I.3.7. Тапсырмалар:

1. Тапаншаның оқ салғышы: соңғы салынған оқ, бірінші атылады.

2.Темір жолдағы тіркескен вагондар: алғашқы стансадан шыққан бірінші вагон, келесі сатансаға бірінші болып келеді.

II.1.1. Тапсырмалар:

1. 2^{24};

2. 32 бит;

3. Жад сыйымдылығының өлшем бірліктері екінің дәрежесі болады, әрбір келесі бірлік алғашқыдан 1024 байтқа үлкен.

II.1.2. **Тапсырмалар:**

1. Жадтың адрестік прінсіпі, деректер мен пәрмендердің бірдей болу прінсіпі, программалық басқарылу прінсіпі, компьютер жұмысының дискретті болу прінсіпі.

2. Пәрмен адресінің регістрі, пәрмен регістрі, алғашқы және нәтиже деректер регістрі, нәтиже белгісінің регістрі, ерекше жағдайдың регістрі.

3. Компьютердің жұмыс ырғағы деп берілген программадағы бір пәрменді орындау кезінде туатын жұмыстарды айтады.

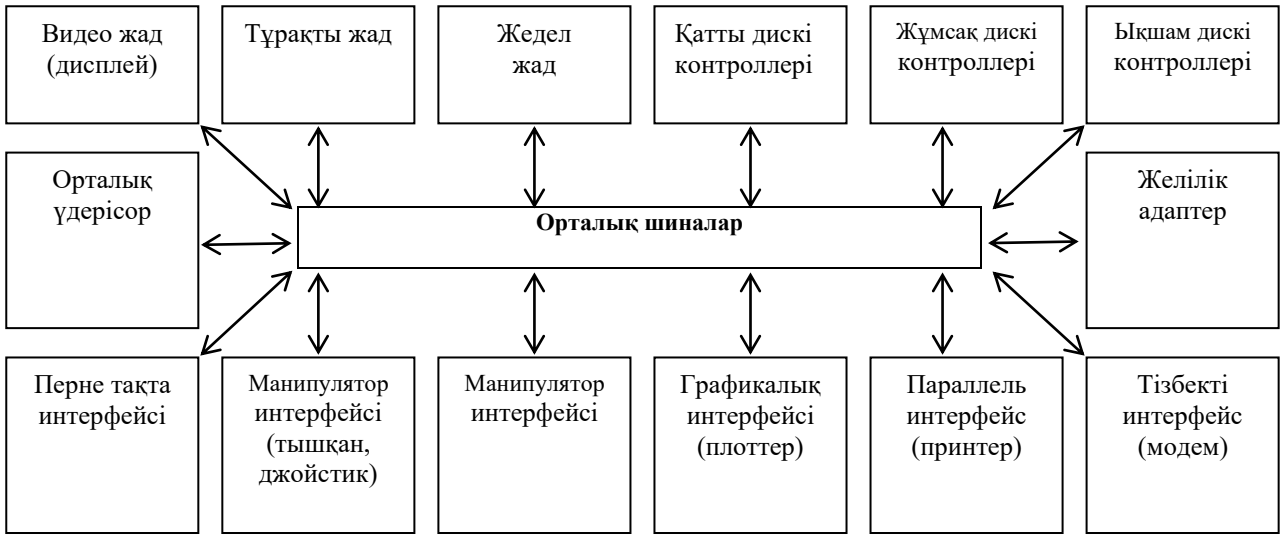
II.1.3. Тапсырмалар:

1. Бірінші буын компьютерлерінің элементтік базасы электронды лампылар.

2. Екінші буын компьютерлерінің программалық қамтымына мониторлық жүйе, транслятор, ассемблер, программалық дестелер, макростар жатады.

3. Компьютердің бесінші буынында Prolog аппаратты түрде жүзеге асырып, машиналық тіл ретінде жасанды зерде жасау үшін қолданылады.

II.1.4. Тапсырмалар:



II.2.1. Тапсырмалар:

1. $1_4 2_3 4_3 \quad 1_4 2_3 4_3 \quad 1_4 2_3 4_3$;
   A E A

2. $1_4 2_3 4_3 \quad 1_4 2_3 4_3 \quad 1_4 2_3 4_3 \quad 1_4 2_3 4_3$;
   2 0 1 4

3. $1_4 2_3 4_3 \quad 1_4 2_3 4_3 \quad 1_4 2_3 4_3$.
   D E F

II.2.2. Тапсырмалар:

1. $63_{10} = 111111_2$, сондықтан

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

2. $-0{,}11101 * 2^5$:

1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	1	0	0	0	0	1	0	0	1
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

3. $X =$ 1111110111001110.

III.1.1. Тапсырмалар:

1. Теңдеудің коэфициенттерін ендіру; дискриминанты есептеу; егер дискриминант нөлден кіші болса, онда теңдеудің шешімі жоқ; егер дискриминант нөлге тең болса, онда теңдеудің бір шешімі бар; қалған жағдайда теңдеудің екі шешімі бар.
2. Шай демдеулің алғашқы деректері: шай демдейтін ыдыстағы қайнаған су, құрғақ шай; нәтижесі қайнаған суға салынған шай.
3. Әр семестр сайын тапсырылған емтиханның нәтижелері.

III.1.2. Тапсырмалар:

1. A, B мәндерін беріңіз
   Егер A>B, онда A – үлкен, әйтпесе B – үлкен;
2. A, B мәндерін беріңіз
   Егер A<B, онда A – кіші, әйтпесе B – кіші;
3. A, B мәндерін беріңіз; C (A+B)/2.

III.2.1. Тапсырмалар:

1. **БАСЫ**
   S ← 0
   M, N сандарын енгізу
   ҚАЙТАЛАУ

S ← S+N

M← M-1

M=0 **ШЕЙІН**

S нәтижесін шығару

СОҢЫ

2. **БАСЫ**

S ← 1

M, N сандарын енгізу

ҚАЙТАЛАУ

S ← S*N

M← M-1

M=0 **ШЕЙІН**

S нәтижесін шығару

СОҢЫ

3. **БАСЫ**

S ← 1

N санын енгізу

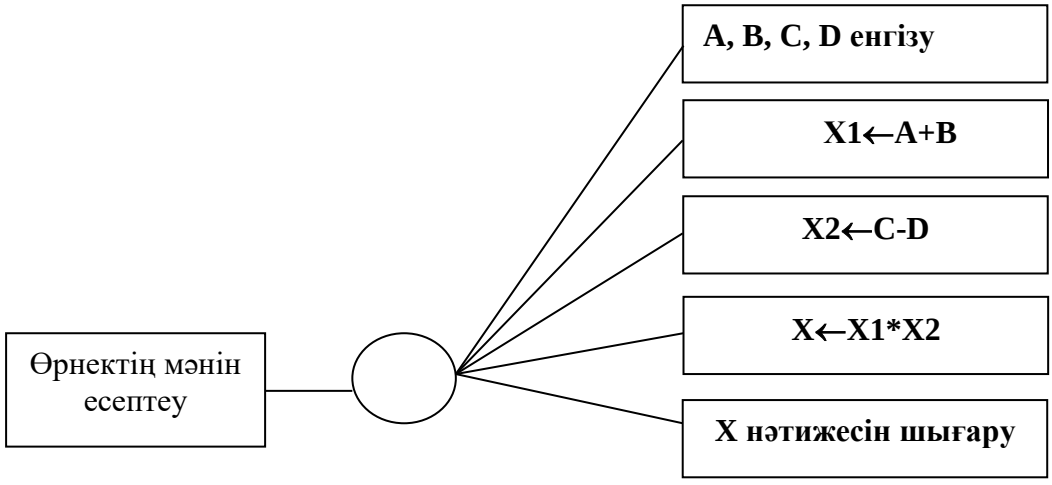
S ← (N*N) / 2

S нәтижесін шығару

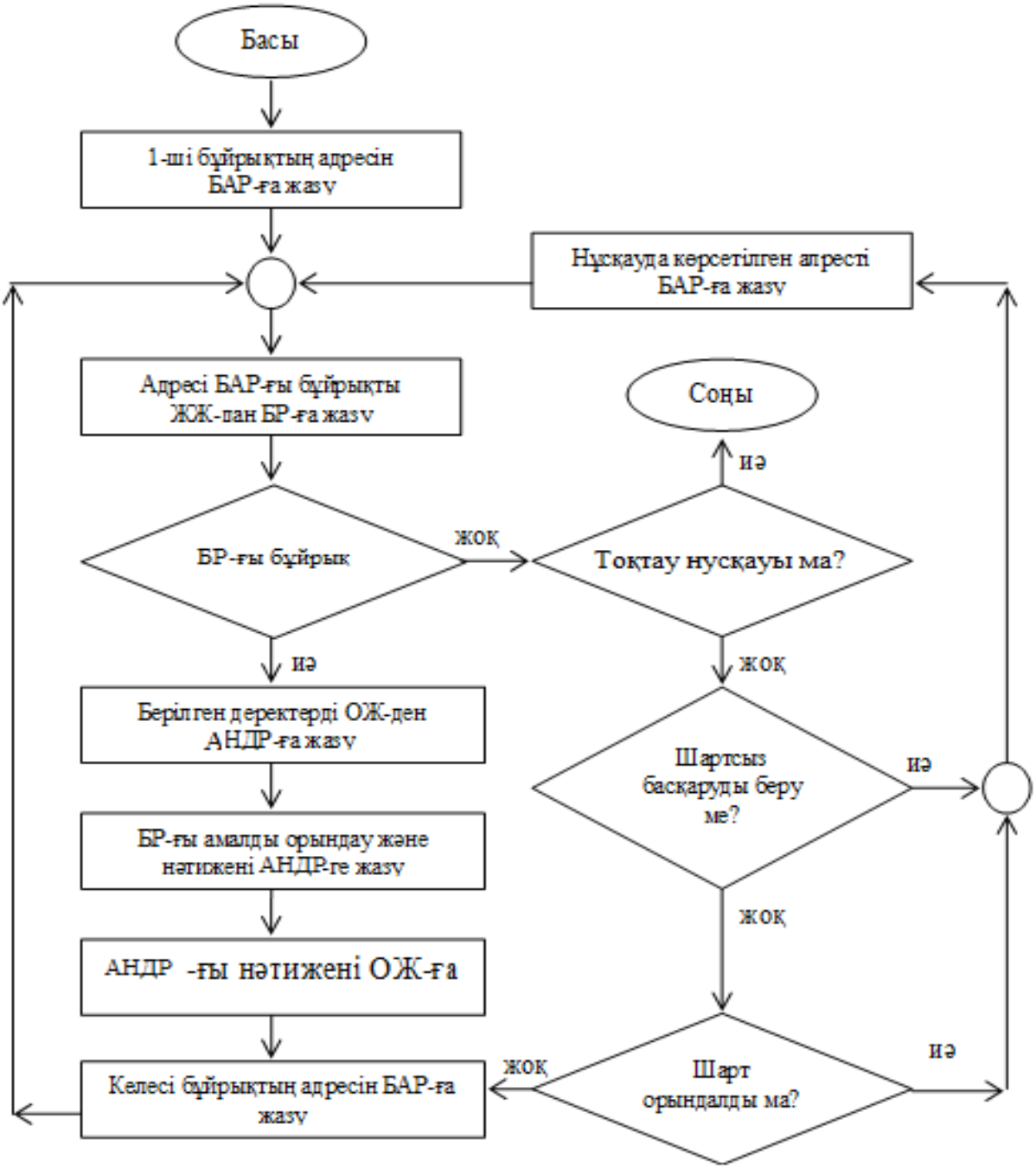
СОҢЫ

III.2.2. Тапсырмалар:

1.



2.



3. **БАСЫ**

S ← 0

N санын енгізу

ҚАЙТАЛАУ

S ← S+N

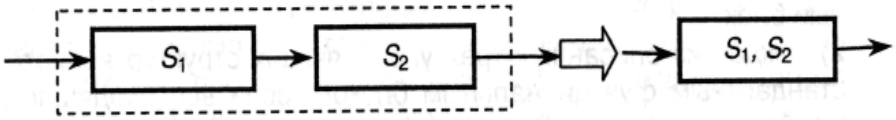
S >100 **ШЕЙІН**

S нәтижесін шығару

СОҢЫ

III.3.1. Тапсырмалар:

1.

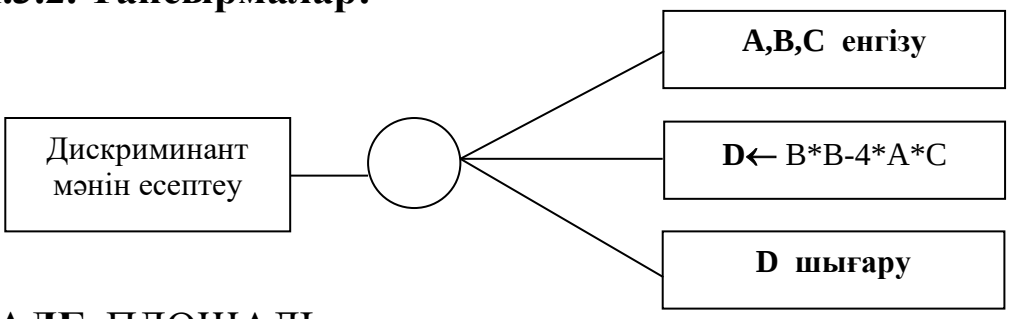


2. Қарапайым тармақталу, баламалы тармақталу, көп мәнді тармақталу.

3. Түсініктілік, қарапайымдылық, тексерімділік, жаңғыртулылық.

III.3.2. Тапсырмалар:

1.



2. **АЛГ** ПЛОЩАДЬ

БАСЫ

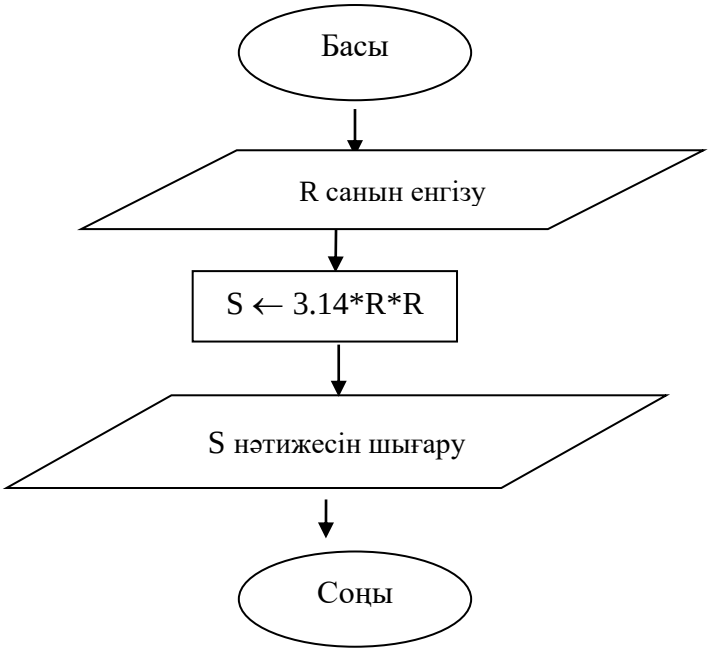
H, B-ны енгізу

S = B* H/ 2

S-ті баспаға шығару

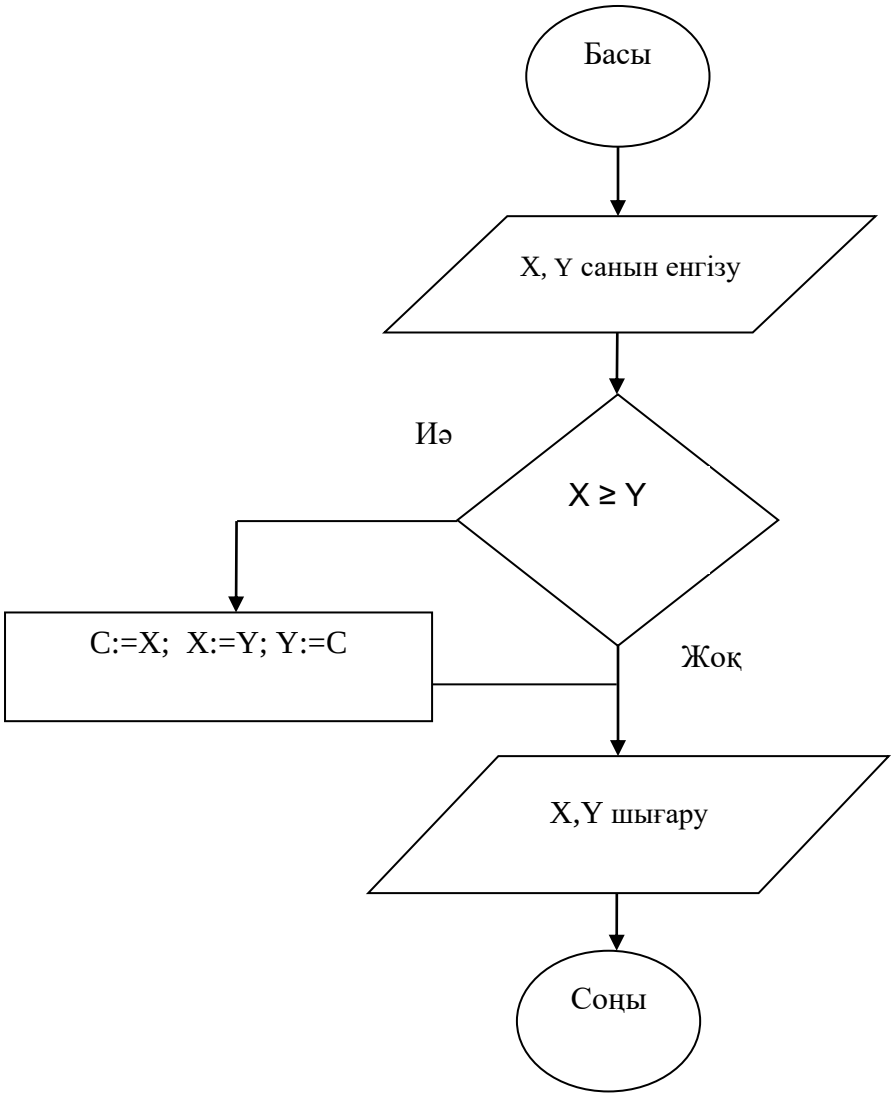
СОҢЫ

3.



III.3.3. Тапсырмалар:

1.

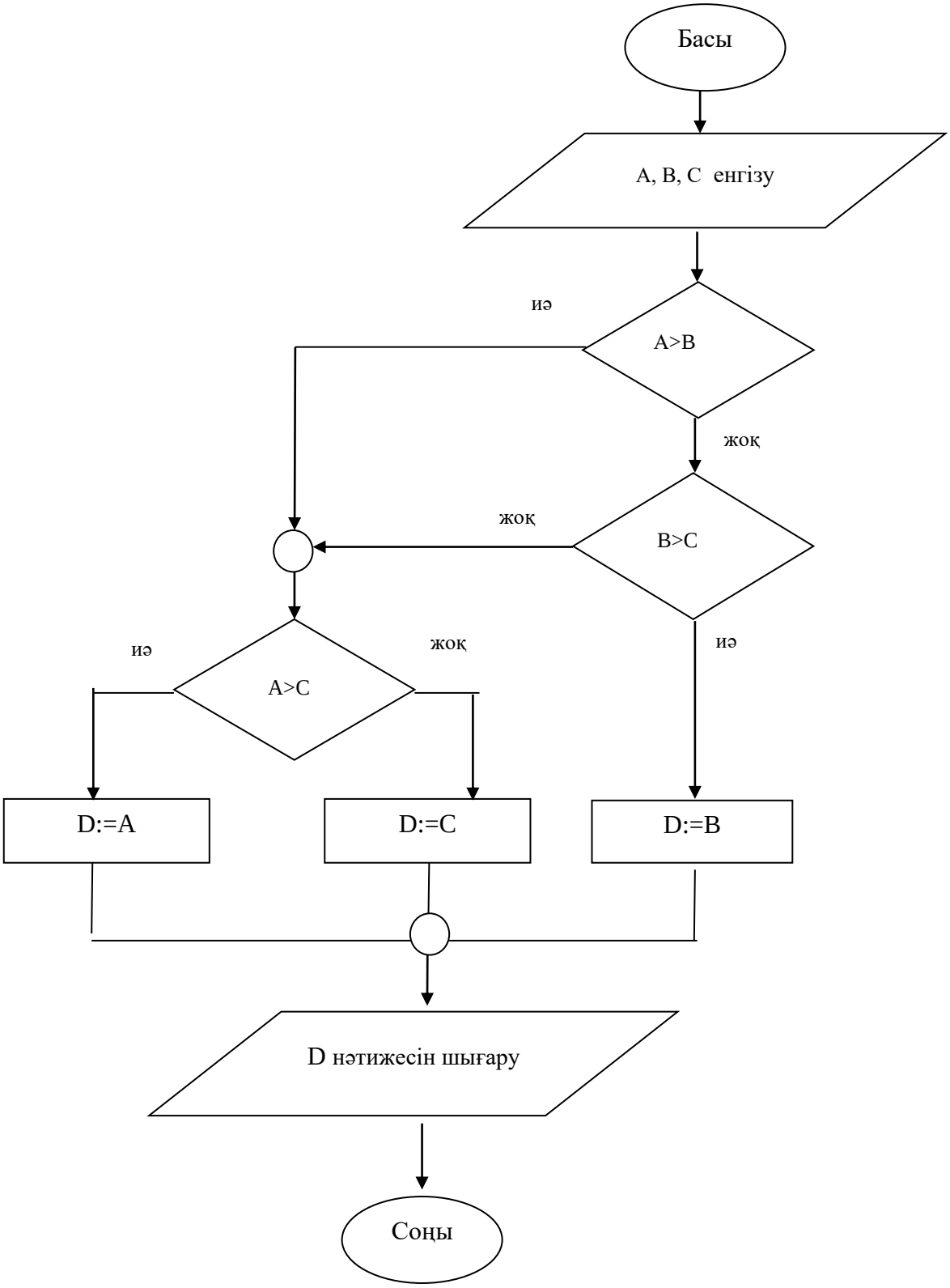


III.3.4. Тапсырмалар:

1. АЛГ МИНМАХ
   ЕНГІЗУ *X*, *Y*
   БАСЫ
      ЕГЕР *X*>*Y* ОНДА *A*=*X*
         ӘЙТПЕСЕ *B*=*Y*
   ШЫҒАРУ *A*, *B*
   СОҢЫ
2. Баламалы тармақталуға жатады.
3. R=9.

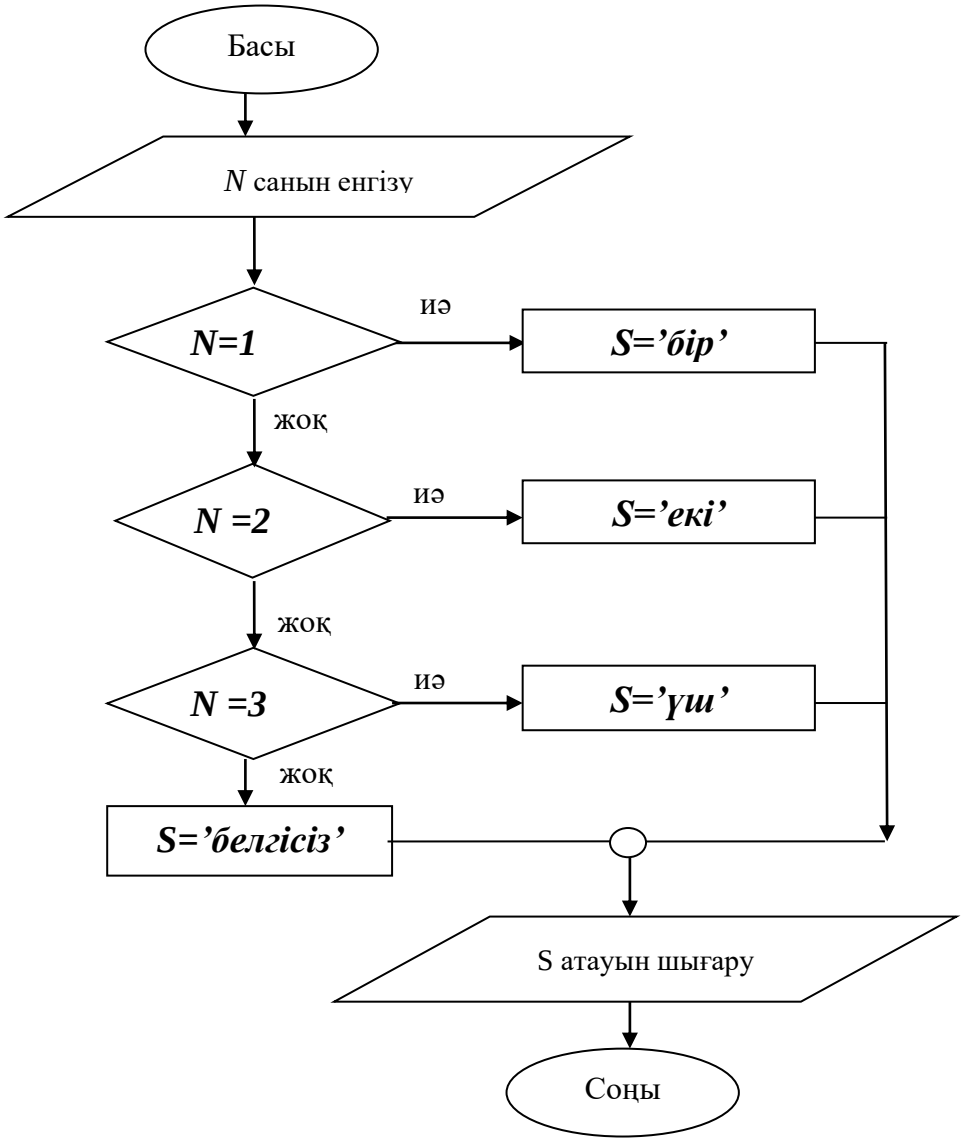
III.3.5. Тапсырмалар:

1.



2. Көпмәнді тармақтау.

3.



III.3.6. Тапсырмалар:

1. АЛГ ОН
 БАСЫ
  I:=1
  ӘЗІРШЕ I < 11
   ШЫҒАРУ I
   I:=I+1
  СОҢЫ

2. АЛГ ЖҮЗ
 БАСЫ
  S:=1

ӘЗІРШЕ $S \leq 100$
$S = S+1$
СОҢЫ

3. АЛГ FACT
БАСЫ
F:=1
ӘЗІРШЕ $F \leq 100$
$F= F*(F-1)!$
СОҢЫ

ІІІ.3.7. Тапсырмалар:
24.

ІІІ.3.8. Тапсырмалар:
АЛГ МАХ
$M \leftarrow 0$
V=1,15 ҚАДАМ ***1*** ҮШІН
ҚАЙТАЛАУ
ЕНГІЗУ X
ЕГЕР М<Х ОНДА М:=Х
СОҢЫ
ШЫҒАРУ М
БІТТІ

ІІІ.4.1. Тапсырмалар:
1. $2^8 = 256$;
2. Компиляциялау әдісі бойынша программа жылдам жұмыс істейді, бірақ жадтың көп орнын алады, ал интерпретациялау әдісі жадтың аз көлемін қажет етеді де, бірақ ақырын жұмыс істейді.
3. Алдымен алгоритмді тілден машиналық тілге аударылады, содан кейін машиналық тілдегі программаны орындау.

III.4.2. Тапсырмалар:

1. Алдымен меншіктеу амалының оң жағындағы өрнектің мәні есептеледі, содан кейін ол мән сол жағындағы айнымалыға меншіктеледі.

2. Процедуралық, функционалдық, логикалық, продукциялық және объектілік.

3. АЛГ ДИСКР
   ЕНГІЗУ *A, B, C*
   БАСЫ
      D ← B*B - 4*A*C
   ШЫҒАРУ D
   СОҢЫ

III.4.3. Тапсырмалар:

```
program u;
var a,b,c,ha,hb,hc,p,s:real;
begin
read(a,b,c);
p:=(a+b+c)/2;
s:=sqrt(p*(p–a)*(p–b)*(p–c));
ha:=2*s/a;
hb:=2*s/b;
hc:=2*s/c;
writeln('ha=',ha:3:3);
writeln('hb=',hb:3:3);
writeln('hc=',hc:3:3);
end.
```

III.4.4. Тапсырмалар:

1. 6
2. (7 8)
3. (((3 4)5)6)

III.4.5. Тапсырмалар:

ұл(*X*, *Y*) :– ***әке***(*Y*, *X*)**.**
ұл(*X*, *Y*) :– ***шеше***(*Y*, *X*)**.**
қыз(*X*, *Y*) :– ***әке***(*Y*, *X*)**.**
қыз(*X*, *Y*) :– ***шеше***(*Y*, *X*)**.**
немере(*X*, *Z*) :– ***ұл***(*X*, *Y*), ***ұл***(*Y*, *Z*)**.**
немере(*X*, *Z*) :– ***ұл***(*X*, *Y*), ***қыз***(*Y*, *Z*)**.**
жиен_немере(*X*, *Z*) :– ***ұл***(*X*, *Y*), ***қыз***(*Y*, *Z*)**.**
жиен_немере(*X*, *Z*) :– ***қыз***(*X*, *Y*), ***қыз***(*Y*, *Z*)**.**

III.4.6. Тапсырмалар:.

1. J1+ S2
2. J1–J2
3. N1* N2
4. /4/(/0/)

III.4.7. Тапсырмалар:

1. Қиылған конустың толық бетінің ауданы мен көлемі мына программамен есептеледі:

```
package kz.enu.inf;
import static java.lang.Math.*;
public class Calculate {
  private int r1 = 20;
  private int r2 = 15;
       private int l = 13;
       private int h = 12;
```

```
public void Find() {
        double S, V;
        S = PI*(pow(r1,2)+(r1+r2)*l+pow(r2,2));
        V=PI*h*(pow(r1,2)+pow(r2,2)+r1*r2)/3;
        System.out.println("S=" + S+" V=" + V );
 }
   public static void main(String[] args) {
   Calculate obj = new Calculate();
        obj.Find();
   }
}
```

2 AB кесіндісін $m:n$ қатынасындай бөлетін екі $A(x_1, y_1)$ және $B(x_2, y_2)$ нүктелерінің координаттары мына программамен анықталады:

```
package kz.enu.inf;
import static java.lang.Math.*;
public class Calculate {
 private int x1 = 5;
 private int y1 = 15;
      private int x2 = -5;
      private int y2 = 20;
      private int m = 2;
      private int n = 3;

public void Find() {
        double x, y, k;
        k=m/n;
            x=(x1+k*x2)/(1+k);
            y=(y1+k*y2)/(1+k);
            System.out.println("x=" + x+" y=" + y );
 }
   public static void main(String[] args) {
   Calculate obj = new Calculate();
        obj.Find();
   }
}
```

IV.1. 1. Тапсырмалар:

1. Сұрыптаудың алғашқы шарты бір типті жиым элементтері,

олардың мәндері үшін салыстыру ("=", ">", "<", ">=", "<=") амалы орындалады; нәтижесінде осы элементтер, бірақ олардың мәндері өсу (кему) бойынша орналасуы керек.

2. Ішкі сұрыптау оперативтік жадта, сыртқы сұрыптау сыртқы жадта жасалады.

3. Сұрыптау ақпарат іздеуді жеңілдетеді, деректер құрылымындағы элементтерді реттейді.

IV.1.2. Тапсырмалар:

1. Элементті сүзу x–тің кілтіне қарағанда кілтті кіші $a[i]$ элементі табылғанда және дайын тізбектің сол жақ шетіне қолжеткенде аяқталады.

2. Сұрыпталатын $\boldsymbol{a}$ жиымын алғашқы тізбек $a[1], a[2], \ldots, a[n]$ және дайын тізбек $a[1], a[2], \ldots, a[i-1]$ деп екіге бөлінеді, сұрыптау қадам жасау арқылы жүзеге асады.

3. Алғашқы тізбект екіге бөлінеді: бір бөлігі реттелген, ал екішісі реттелмеген болып.

IV.1.3. Тапсырмалар:

1. Сұрыптау ең кіші кілтті элементті таңдаумен басталады және бірінші элементпен алмасады.

2. Әрбір қадамда берілген тізбектің тек бір ғана кезектегі элементін қарастырады.

3. Кілттерді салыстыру саны кілттердің бастапқы ретінен тәуелсіз.

IV.1.4. Тапсырмалар:

1. Жиымдағы барлық элементтер сұрыпталғанша көршілес екі элементті салыстыру мен ауыстыру прінсіпіне негізделген.

2. Мына тізбек 25 37 42 54 75 91 06 көпіршік әдісімен сұрыптағанда бір қаралымда сұрыпталады.

3. Берілген мына тізбекті 19 31 43 52 69 87 05 шейкер-сұрыптаумен сұрыптағанда бір қаралымда сұрыпталады.

Сұрақтардың жауаптары

I.1.1. Сұрақтар:

1. Ақпарат – бізді қоршаған дүниедегі объектілердің, жағдайлардың, үдерістердің немесе құбылыстардың кейбір қасиеттерінің және қатынастарының бейнесі.

2. Ақпарат ұғымының дәл (математикалық) анықтамасы жоқ.

3. Әрбір ақпараттың хабары және мазмұны болуы керек.

4. Хабар ақпаратың формасы (пішімі).

5. Ақпарат материалды энергетикалық түрде беріле алады.

6. Табиғи тілдер, математика тілі, әуез тілі, мылқаулар мен кереңдер тілі.

7. Программалау тілдерін қатынас тілдеріне жатқызуға болады.

8. Хабарды жіберуші мен алушының арасында байланыс арнасы болады.

9. Хабарды жіберу (қабылдау) кезінде жіберушінің (қабылдаушының) жағдайы уақытқа байланысты өзгереді.

10. Аналогты ақпарат деген уақытқа үздіксіз болады.

11. Цифрлы ақпарат деген уақытқа үздікті болады.

12. Аналогты ақпаратты дискреттеп цифрлайды.

I.1.2. Сұрақтар:

1. Хабарды жіберуші мен қабылдаушы арасында хабардың түрі (бейнесі) жайында және оның мазмұны туралы алдын ала анықталған келісім болуы керек.

2. Хабарды белгілі бір тіл арқылы беруге болады.

3. Хабар мен мазмұнның арасында байланыс бірмәнді емес.

I.1.3. Сұрақтар:

1. Бит (bit).

2. Джон фон Нейман және Клод Шеннон.

3. 2-ге еселі.

I.1.4. Сұрақтар:

1. Анықталмағандықты өлшейтін сандық өлшем *энтропия*.

2. Бірмәнді сәйкестік бар.

3. Энтропияны анықтауға керек.

I.2.1. Сұрақтар:

1. Сандық, символдық және логикалық.

2. Әріптен басталатын және әріптер мен цифрлардан құралған ұзындығы шектелген тізбекті айтады.

3. Айнымалы шаманың мәні көп, ал тұрақты шаманың мәні біреу ғана.

I.2.2. Сұрақтар:

1. Бүтін, нақты және комплекс сандар болып үшке бөлінеді.

2. Тұрақты нүктелі нақты сандар және жылжымалы нүктелі нақты сандар.

3. Комплекс сандар нақты бөлік және жорамал бөліктен тұрады.

I.2.3. Сұрақтар:

1. Амалдар операндтардың алдында жазылады.

2. Амалдар операндтардыңортасында жазылады.

3. Амалдар операндтардың артында жазылады.

I.2.4. Сұрақтар:

1. Санау жүйесі деп сандардың цифрлар арқылы жазылу әдістері мен ережелер жиынын айтады.

2. Позициялық және бейпозициялық санау жүйесі.

3. Кез келген сандық мән әртүрлі санау жүйесінде жазыла береді.

I.2.5. Сұрақтар:

1. Белгілі бір қатынас тілінің әліпби таңбаларынан тұратын тізбектердің жиыны символдық типті құрайды.

2. Символдық шаманың мәні символдық жақшаға алынған әріптер, цифрлар немесе арнаулы таңбалардан құрылған тізбекпен кескінделеді.

3. Информатикада қолданылатын қатынас тілдерінің әліпбилері әріптерден, цифрлардан және арнаулы таңбалардан тұрады.

4. Информатикада цифрлар ретінде араб цифрлары қабылданады.

5. Информатикада арнайы символдарға пайдаланылатын амалдардың таңбалары, тыныс белгілері, топтастыру таңбалары, кетік таңбасы және т.б. жатады.

6. Информатикада әріптер ретінде латын әріптері қабылданады.

7. |ABCD| – символдық шаманың ұзындығы төртке тең.

8. Кетікт бос тізбе емес, ол бір таңба.

9. Бос тізбенің ұзындығы нөлге тең.

10. Кетіктің ұзындығы бірге тең.

I.2.6. Сұрақтар:

1. Символдық шамаларда анықталған амалдар екі топқа бөлінеді: құрастыру амалдары, бөлу амалдары.

2. Конкатенация және дизъюнкция амалдары арқылы анықталады.

3. Бос тізбе деген құрамында ештеңе болмайтын абстракты тізбек.

4. Берілген символдық тізбектен белгілі бір шарт бойынша таңбаны немесе осы тізбектің бөлігін алып тастауға мүмкіндік беретін амал.

5. Бөлу амалдарының ішіндегі ең қарапайымы – берілген тізбектен оның белгілі орнынан (позициясынан) бастап саны белгілі таңбаларды алып тастау амалы

6. $\alpha \Subset \beta \rightleftharpoons \exists \xi(\beta = \alpha\xi)$.

7. $\alpha \Supset \beta \rightleftharpoons \exists \zeta(\beta = \zeta\alpha)$.

I.2.7. Сұрақтар:

1. Логикалық мәндерге «ақиқат» және «жалған» кіреді.

2. «ақиқат» немесе «иә» немесе « 1 » немесе «+».
«жалған» немесе «жоқ» немесе « 0 » немесе «–».

3. Ақиқат.

I.2.8. Сұрақтар:

1.

A	B	A∨B
0	0	0
0	1	1
1	1	1

1	1	1

2. «Коньюнкция» («және») амалын.

I.3.1. Сұрақтар:

1. Элементтер арасында байланыстарды ұйымдастыру бойынша сызықтық деректер құрылымы және сызықтық емес деректер құрылымы деп бөлінеді.

2. Сызықтық емес деректер құрылымы элементтерінің өзара байланысы бойынша иерархиялық, торлық, кестелік деректер құрылымына бөлінеді.

3. Бейнелену тәсілі бойынша деректер құрылымы физикалық және логикалық деректер құрылымдарына бөлінеді.

I.3.2. Сұрақтар:

1. Жиынға кіретін элементтердің санын жиынның қуаты дейді

2. Жиын барлық элементтерін көрсету арқылы немесе сипаттау арқылы анықталады

3. Бос тізбелер жиынының ұзындығы бірге тең.

I.3.3. Сұрақтар:

1. Екі жиынның бірігуі;

2. Екі жиынның қиылысуы;

3. Екі жиынның айрымы.

4. Екі жиынның симметриялық айырымы.

I.3.4. Сұрақтар:

1. Бір өлшемді кесте бір ғана жиынның элементтерінен құрылған статикалық байланыссыз сызықтық деректер құрылымы.

2. Екі өлшемді кесте екі жиынның тік көбейтіндісі элементтерінен құрылады.

3. Көп өлшемді жиымда өлшем саны элементтер индексінің санына тең.

I.3.5. Сұрақтар:

1. Сызықтық бірбайланысты тізім деген бір-бірімен нұсқағыштар арқылы тізбекті баланысқан бір типті элементтерден тұратын динамикалық сызықтық деректер құрылымы.

2. Сызықтық екібайланысты тізім деген әрбір элементі екі нұсқағышты қамтиды: келесіге және алдындағысына динамикалық сызықтық байланысқан деректер құрылымы.

3. Сақиналық тізім деген сызықтық бірбайланысты немесе екібайланысты тізімдерде соңғы элементтің нұсқағышы бірінші элементке нұсқайды.

4. Иерархиялық тізім – сызықтық тізіммен дарақтың қосындысы.

5. Нөлге тең.

I.3.6. Сұрақтар:

1. Кілттер саны кесте өлшемінен көп болғанда қайшылық кпайда болады.

2. Егер біз хеш-кестедегі элементті жай ғана жоятын болсақ.

3. Ұяшықта қайшылық болғанда бір ұяшыққа сызықтық ығыстырудың орнына, басқа хеш-функцияны қолдану.

I.3.7. Сұрақтар:

1. Графтағы жол деп көрші екі қабарғаның ортақ төбесі болатындай және ешбір қабырға бір реттен артық кездеспейтіндей етіп бір төбеден басқа төбеге жүргізілген тізбекті айтады.

2. Дарақ деген барлық төбелері байланысқан, ал жолдары тұйық емес граф, яғни тұйықсыз және тұзақсыз байланысқан граф.

3. Екілік дарақ деген әрбір төбесінің екіден аспайтын ұрпағы болатын дарақ.

I.3.8. Сұрақтар:

1. Стек деген динамикалық сызықтық біршекті деректер құрылымы.

2. Стекте стекті құру, элементті стекке қосу, элементті стектен жою, стек төбесіндегі элементті жоймай қарау, стектің күйін тексеру, стекті тазалау амалдары анықталған.

3. Кезек деген динамикалық сызықтық екішекті деректер құрылымы.

4. Кезекте кезекті құру, элементті кезектің соңына қосу, элементті кезектің басынан жою, стекті тазалау амалдары анықталған.

5. Дек деген динамикалық сызықтық екішекті деректер құрылымы.

6. Декте амалдар элементті дектің соңына қосу, элементті дектің басына қосу, элементті дектің соңынан жою, элементті дектің басынан жою, дектің мөлшерін анықтау, декті тазалау амалдары анықталған.

II.1.1. Сұрақтар:

1. Компьютер ақпараттық хабардың түрі бойынша аналогты компьютерлер немесе дискретті компьютерлер болып екі топқа бөлінеді.

2. Компьютердің классикалық сәулетіндегі қос бағыттама деректер мен нұқаулардың жолын көрсетеді.

3. Компьютердің адрестік сөзі адрестерді сақтайды, ал машиналық сөзі деректерді сақтайды.

II.1.2. Сұрақтар:

1. Процессоррдың разрядтылығы дегеніміз процессоррдың бір амалды орындау үшін алынатын ақпараттың көлемдік мөлшері. Разрядтылық әрқашан 2–нің дәрежесі болады.

2. Процессоррдың жиілігі деп процессоррдың бір секундта қанша такты жасай алатындығын айтады.

3. Процессоррдың жылдамдығы дегеніміз процессоррдың бір секундта қанша амал орындай алатындығын айтады.

II.1.3. Сұрақтар:

1. Компьютердің элементтер базасы оның негізгі құрылғыларын (процессорр және т.б.) жасауға қолданылған электрондық элементтер (лампалар, транзистрлер, интегралды сұлбалар).

2. Қазір компьютерлердің алты буыны белгілі.

3. Қомпьютерлердің алтыншы буыныны жасанды нейрон желісіне, бұлдыр логикаға және кванттық есептеу теориясы мен генетикалық алгоритмдердерге негізделіп жасалынады.

II.1.4. Сұрақтар:

1. Монитор, жүйелі блок, пернетақта.
2. Микропроцессорр.
3. Бірінші әмбебап Intel–8080 микропроцессорры 1974 жылы

жасалынды.

4. Дербес компьютерге қажет барлық байланысты қамтамасыз етеді.

5. Сыртқы дүниемен алмасуды контроллер басқарады.

II.2.1. Сұрақтар:

1. Логикалық деректер компьютерде бір битпен бейнеленеді: 0 – жалған, 1 – ақиқат.

2. Символдық деректер компьютер жадында 0 мен 1-ден тұратын ұзындығы 8-бит немесе 16-бит тізбектермен бейнеленеді.

3. ANSI 8-битті, ал Unicode 16-битті стандарт.

II.2.2. Сұрақтар:

1. Бүтін сан екілік санау жүйесінде бір машиналық сөзде бейнеленеді, бірінші битте санның таңбасы орналасады: 0 – таңба «+», а 1 – таңба «-».

2. Жылжымалы нүктелі нақты сан екілік санау жүйесінде бір машиналық сөзде бейнеленеді: ең сол жақ битте мантиссаның таңбасы жазылады, кейін мантисаның өзі орналасады (байттар саны машиналық сөздің разрядтығына тәуелді), содан кейін бір битте рет таңбасы түседі, ал ең соңында реттің өзі бір байтта бейнеленеді.

3. Бекітілген нүктелі нақты сан компьютерде бейнелену үшін жылжымалы нүктелі нақты санның нормалды пішініне түрлендіріледі де жылжымалы нүктелі нақты сан ретінде бейнеледі.

III.1.1. Сұрақтар:

1. Алдын ала анықталған мақсатқа жету үшін берілген шамаларды түрлендіру.

2. Өңдеудің алдында алғашқы деректер беріледі.

3. Нәтиже өңдеудің орындалуы біткеннен кейін алынған деректер.

III.1.2. Сұрақтар:

1. Алгоритм деп ақырлы қадам жасау арқылы белгілі бір сыныптағы кез–келген есепті шешуге арналған түсінікті және дәл қай амалды қандай ретпен орындалатынын көрсететін нұсқаулар тізбегін айтамыз.

2. Түсініктілік, дәлдік, дискреттілік, ақырлылық, жалпыламалық.

3. Алгоритмнің орындалу хаттамасы деп оны орындаған кезде әрбір қадамның алғашқы деректері мен нәтижелерінің тізімі.

III.2.1. Сұрақтар:

1. Вербалды алгоритмдік тіл және графикалық алгоритмдік тіл.
2. Вербалды алгоритмдік тілдің әліпбиі латын және қазақ әліпбилері таңбаларынан, араб және рим цифрларынан, жақшалардан, амалдар таңбаларынан, бекітілген сөздерден тұрады.
3. БАСЫ, СОҢЫ, ЕНГІЗУ, ШЫҒАРУ, ЕГЕР, ОНДА, ӘЙТПЕСЕ, ТАҢДАУ, БОЛСА, ОРЫНДА, ҚАЙТАЛАУ, ӘЗІРГЕ, ШЕЙІН, ҮШІН, ҚАДАМ, АЛГОРИТМ, БІТТІ.

III.2.2. Сұрақтар:

1. Графикалық алгоритмдік тілдің екі түрі бар: блок-сұлба тілі және дарақ тәрізді тіл.
2. Блок-сұлба тілінің әліпбиі жұмыр, параллелограмм, тік төртбұрыш, ромб, шеңбер және бағыттамалардан тұрады.
3. Дарақты аралау жолы алгоритмдегі әрекеттердің жіктелу реті мен олардың орындалу ретін көрсетеді.

III.3.1. Сұрақтар:

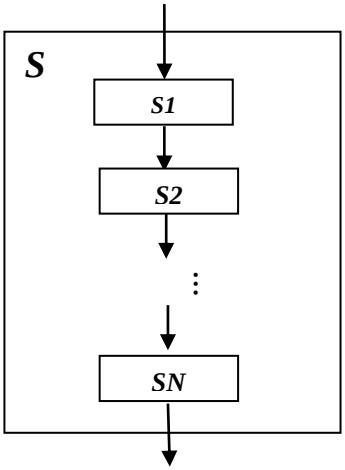
1. Тізбектеу бірнеше функционалдық блокты бір блокка біріктіреді.
2. Тексерілетін логикалық шарттың мәніне байланысты екі S_1 және S_2 функционалдық блоктың бірін орындауды ұйымдастырады.
2. Тексерілетін логикалық шарттың мәніне байланысты функционалдық блоктың бірнеше рет қайталанып орындалуын ұйымдастырады.

III.3.2. Сұрақтар:

1.
: **S** күрделі әрекетінің денесі
БАСЫ
< ***S1*** >
< ***S2*** >
...
< ***SN***>
СОҢЫ

2. Тізбектеудегі ***S*** күрделі әрекеттің алғашқы деректері ретінде ***S1*** әрекеттің деректері, ал нәтижесі ретінде ***SN*** әрекеттің нәтижесі алынады.

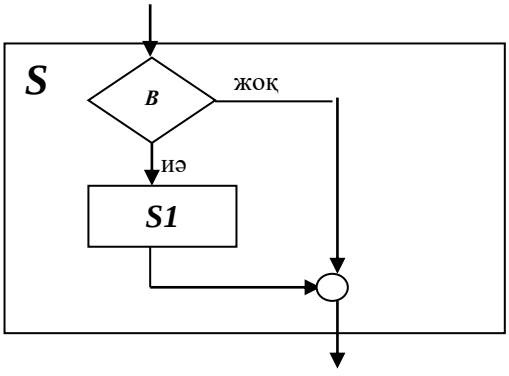
3.



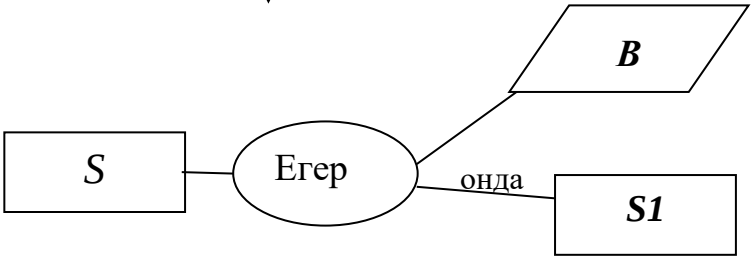
III.3.3. Сұрақтар:

1. : ***S*** күрделі әрекетінің денесі

2. ЕГЕР <***B***> ОНДА <***S1***>

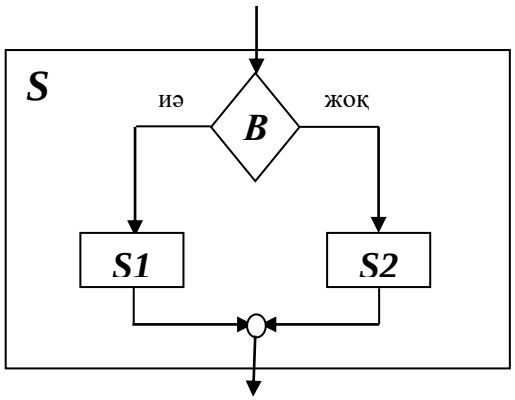


3.

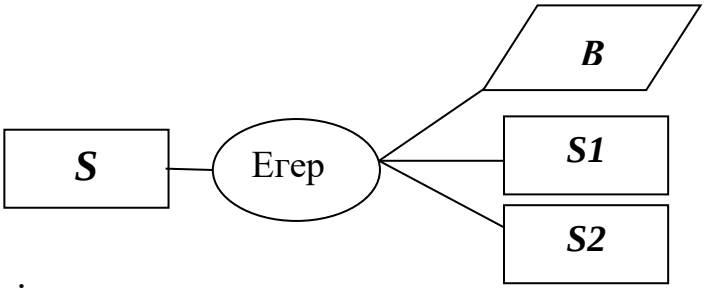


III.3.4. Сұрақтар:

1. Блок сұлба тілі:



2. Дарақ тәріздес тіл:



3. Вербалды тіл:

ЕГЕР <**B** > ОНДА < **S1** >
ӘЙТПЕСЕ < **S2** >

III.3.5. Сұрақтар:

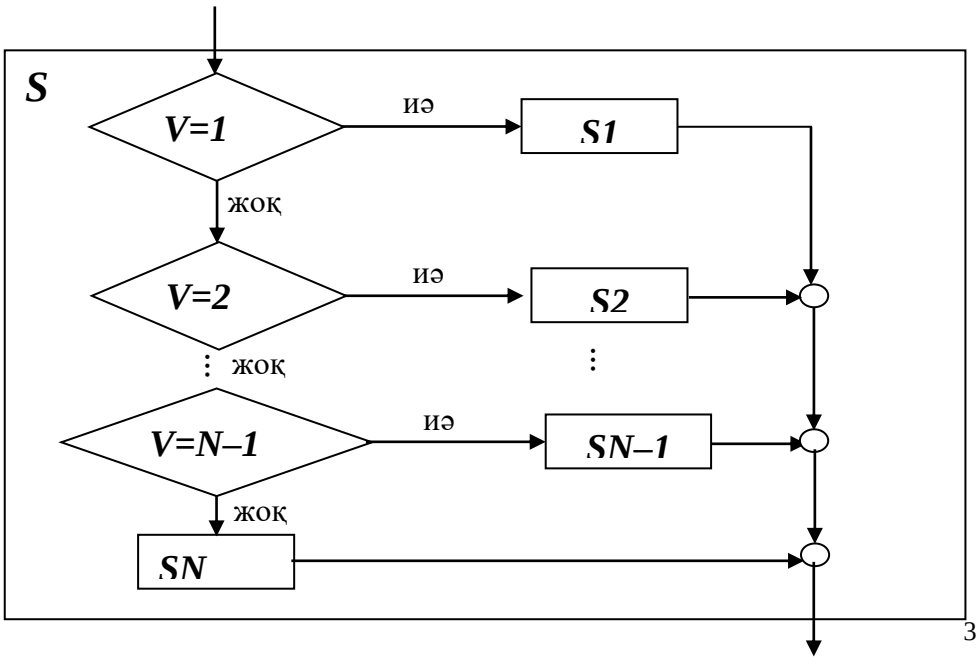
1. Вербалды тіл: ТАҢДАУ

V= 1 БОЛСА ***S1***

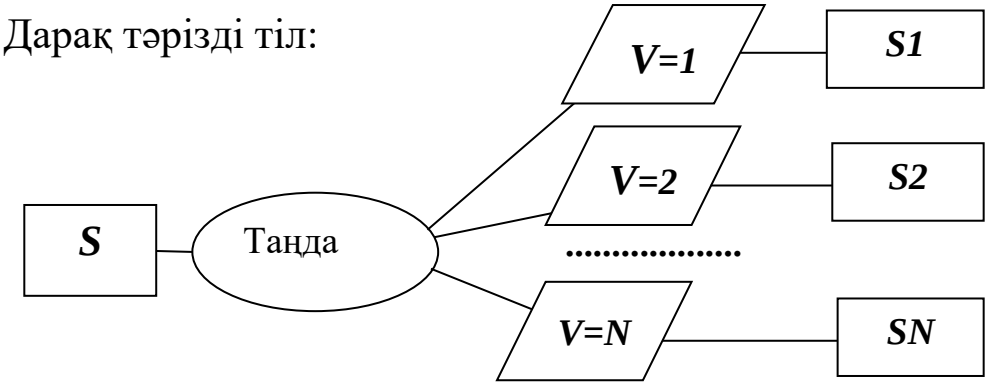
V = 2 БОЛСА ***S2***

. . . . . .

2. Блок-сұлба тілі:



3. Дарақ тәрізді тіл:



III.3.6. Сұрақтар:

1. Алғышартты қайталау күрделі **S** әрекеті белгілі **B** шарты қанағаттандырылса, қайталау денесі **S1** әрекетін орындауды қайталау керек, ал егер осы **B** шарты қанағаттандырылмаса еш нәрсе орындамау керек дегенді көрсетеді.

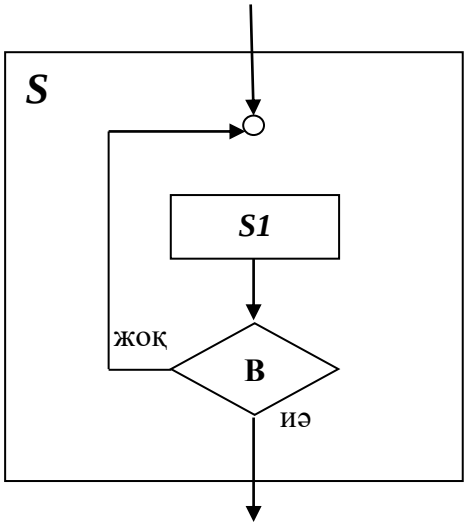
2. Алғышартты қайталауда тексерілетін шарт орындалады немесе орындалмайды деген екі жағадай болуы мүмкін.

3. Егер алғышартты қайталауда тексерілетін шарт қайталау денесіне тәуелсіз орындалып тұрса, онда қайталау денесі ақырсыз қайталана береді.

III.3.7. Сұрақтар:

1. Соңғышартты қайталау күрделі **S** әрекеті **S1** әрекетін орындауды **B** шартына тәуелді қайталай беру керек немесе тоқтату керек дегенді көрсетеді.

2. Блок-сұлба тілі:



3. Орындалу кезінде шарттың мәні өзгермесе екі жағдай болады: егер ол орындау алдында ақиқат болса, онда қайталау денесі бір-ақ рет орындалады; егер ол орындау алдында жалған болса, онда қайталау денесі ақырсыз орындала береді.

III.3.8. Сұрақтар:

1. Егер параметрдің алғашқы мәні параметрдің соңғы мәнінен кіші болмаса, параметрлік қайталау орындалмайды.

2. Параметрлік қайталауда қайталау денесінің орындалу саны параметрдің алғашқы мәні мен соңғы мәні және қадам арқылы анықталады: қайталау денесін әр орындау кезінде параметрдің алғашқы мәніне параметрдің соңғы мәніне жеткенше қадамның мәні қосылып отырады.

3. Параметрлік қайталауды қайталанатын сан және алдын ала белгілі болған жағадайда қолданады.

III.4.1. Сұрақтар:

1. Жасанды алгоритмдік тілді компьютердің машиналық тіліне автоматты түрде аударатын программа.

2. Машиналық тілге қаншалықты жақын екендігіне байланысты блолады.

3. Фортран тілі.

III.4.2. Сұрақтар:

1. Трансляциялаулардың интерпретация, компиляция деген түрлері жүзеге асқан.

2. Абстракция, полиморфизм, инкасуляция, еншілеу.

3. Объектігі бағытталған программалау тілдер.

III.4.3. Сұрақтар:

1. Процедуралық тілдердің ең көп тараған және оқуға қолайлы өкілі Pascal тілі программалауды оқыту мақсатында жасалған.

2. Мысалы, айнымалылар i – бүтін сан; a,b – нақты сан; x,y – таңбалар тізбегі болсын десек, онда олар Pascal тілінде былай жазылады:

```
var  i, j  :  integer;
var  (x, y)  : real;
var  (a, b)  : char;
```

3. *abs*(*e*), *arctan*(*e*), *cos*(*e*), *exp*(*e*), *ln*(*e*), *mod* (*a*,*b*), *sin*(*e*), *sqr*(*e*), *sqrt*(*e*).

III.4.4. Сұрақтар:

1) Функционалдық тілдердің ең бұрын шыққан және ең көп тарағаны LISP (Lisp) тілі. Бұл тіл тізім деп аталатын арнаулы деректер құрылымын бейнелеу және өңдеу үшін арналған.

2) Lisp функциялар тілінде: әрбір программалық құрылымы тек функция болып келеді, программаны құрастыру деген күрделі функцияны қарапайым функциялар арқылы құрастыру болып табылады. Функциялар арасындағы қарым–қатынас тек программа жұмыс істеген кезде олардың шақырулары арқылы жүзеге асады.

III.4.5. Сұрақтар:

1) Логикалық тілде есепке қатысты объектілердің қасиеттерін және олардың бір–бірімен қатынасын сипаттайды, ал есептің шешімін логикалық қорытындылау ретінде компьютер өзі табады.

2) Prolog тілінде программалық бірлік болып логикалық тұжырым есептелінеді. Ең кіші программалық бірлік факты болады, ал келесі программалық бірліктер ретінде ережелер және сұратымдар алынады.

III.4.6. Сұрақтар:

1) Продукциялық программалау тілдерінде негізгі программалық бірлік продукция деп аталатын алмастырым ережесі. Сондықтан бұларды алмастырымдық программалау тілі деп те атауға болады.

2) Басқа программалау тілдер сияқты Refal тілінде стандарттық функциялар бар. Олар бірігіп бірнеше топ құрайды.

III.4.7. Сұрақтар:

1. Java тілінің басқа программалау тілдерден айырмашылығы оның объектіге бағытталғандығында.

2. Айнымалыны жариялау кезінде оның атауы, типі егер керек болдса алғашқы мәні көрсетіледі.

3. Алдымен жақшаның ішіндегісі, сонан кейін көбейту және бөлу, ал ең соңынан қосу және алу амалдары орындалады.

IV.1. 1 Сұрақтар:

1. Сұрыптау дегеніміз қандай да бір деректер құрамындағы элементтерді белгілі бір ретпен орнын ауыстыру.

2. Сұрыптау функциясыеың мәні элементтің кілті болады.

3. Сұрыптау әдістері ішкі сұрыптау және сыртқы сұрыптау деген екі топқа бөлінеді.

IV.1.2. Сұрақтар:

1. Ендірумен сұрыптау жасау үшін алғашқы тізбек *a[1], a[2], ..., a[n]* және дайын тізбек *a[1], a[2], ..., a[i–1]* керек.

2. Егер барлық *n* кілттерді орын ауыстыру орташа тең ықтималды *i/2*-ге тең болса, онда 1-ші сүзу кезінде кілттердің C_i салыстырым санының ең үлкені *i*–1, ал ең кішісі 1 болады.

3. Салыстырым саны элементтердің алғашқы ретіне тәуелді емес.

IV.1.3. Сұрақтар:

1. Таңдаумен сұрыптау әдісі мына ережелерге негізделген: ең кіші кілті бар элемент таңдалады, ол бірінші элемент а[1]-мен орын ауыстырады.

2. Қарапайым таңдау арқылы сұрыптағанда ең кіші кілті бар элементті табу үшін берілген тізбектің барлық элементтері қарастырылады және осы бір элемент дайын тізбекке жіберіледі.

3. Таңдаумен сұрыптауда кілттердің салыстырым саны кілттердің бастапқы ретіне тәуелсіз.

IV.1.4. Сұрақтар:

1. Ауыстырумен сұрыптау көршілес екі элементтің салыстыруы мен орын алмасуының прінсіпіне негізделген.

2. Көпіршік әдісінде мынадай ассиметрия бар: сұрыпталған тізбектің «ауыр» жағының соңында орналасқан бір «жеңіл» элемент (көпіршік) өзінің орнын бір қаралымда табады, ал «жеңіл» жағының алдында орналасқан бір «ауыр» элемент әрбір қаралымда бір ғана қадам арқылы өз орнына келеді.

3. Ш*ейкер-сұрыптау әдісі* әрбір келесі қаралымда көпіршік табу үшін бағытты (жоғары, төмен) өзгерту қажеттігінен пайда болды.

Тесттер жауабы:

Дәрістер нөмірі	1-тест	2-тест	3-тест
I.1.1	B	D	A
I.1.2	C	A	B
I.1.3	C	B	C
I.1.4	E	B	A
I.2.1	D	D	A
I.2.2	C	B	E
I.2.3	A	B	B
I.2.4	E	A	B
I.2.5	A	D	D
I.2.6	B	B	E
I.2.7	A	A	B
I.2.8	B	A	C
I.3.1	A	C	E
I.3.2	B	C	C
I.3.3	A	A	A
I.3.4	A	B	A
I.3.5	A	B	A
I.3.6	A	A	A
I.3.7	A	A	A
II.1.1	B	A	B
II.1.2	A	A	A
II.1.3	A	A	A
II.1.4	A	B	A
II.2.1	C	A	A
II.2.2	A	A	A
III.1.1	A	D	A
III.1.2	A	A	E
III.2.1	A	E	A
III.2.2	A	A	A
III.3.1	A	B	B

III.3.2	D	C	A
III.3.3	B	C	B
III.3.4	A	D	A
III.3.5	C	A	C
III.3.6	E	C	C
III.3.7	A	A	C
III.3.8	A	B	B
III.4.1	A	D	A
III.4.2	A	B	C
III.4.3	A	B	C
III.4.4	E	A	D
III.4.5	B	A	A
III.4.6	A	C	E
III.4.7	A	E	C
IV.1.1	A	A	E
IV.1.2	A	D	C
IV.1.3	A	C	E
IV.1.4	A	D	C

ИНФОРМАТИКА
(Оқулық)

Пішімі 60х80 1/8 Ағартылған қағаз,
Тығыздығы 80 гр./ м2. 95%.
РИЗО басылымы.
Шартты баспа таб. 15,5. Көлемі 312 бет.

Оқулық мазмұнына баспа бөлімі жауап бермейді

“Эверо” баспада басылымға деген
баптаулы және басып шығарылды.
ҚР, Алматы, Байтұрсынұлы көшесі,
22
Тел.: +7 /727/ 233 83 89, 233 83 43,
233 80 45, 233 80 42
e-mail: evero08mail.ru

ШӘРІПБАЙ Алтынбек Әмірұлы, техника ғылымдарының докторы, профессор, ҚР педагогика ғылымдары академиясының академигі, Халықаралық ақпаратандыру академиясының академигі, ҚР мемлекеттік сыйлығының лауреаты. Л.Н.Гумилев атындағы ЕҰУ «Жасанды интеллект» ҒЗИ-ның директоры (www.e-zerde.kz). Оның ғылыми қызығушылығының ғылымға белгілі информатика мен ақпараттық технологиялардың теориялық және қолданбалы проблемалары: программалардың тиімділігі мен технологиялары, автоматтандырылған жүйелер, жасанды интеллект, компьютерлік лингвистика және электрондық оқыту. Ол бұл салада алгоритмдемелік және логикалық программалау тілдерінің семантикаларын формалдау, компьютерлердің программалық және ақпараттық құралдарын верификациялаудың жаңа әдістерін жасады. Осы зерттеулер нәтижелері негізінде ол «05.13.13 (05.13.11) – Есептеу машиналары, жүйелері және торлары (Математикалық, программалық и техникалық қамтамасыз ету саласы бойынша «Есептеу машиналары мен жүйелерінің программалық және ақпараттық құралдарын верификациялау» тақырыбында докторлық диссертация қорғады. Оның кейбір ғылыми нәтижелері ірі ғылыми орталықтарында енгізілді: 1976-1981жж. - «Ғарыштық жүйелерді имитациялау тілінің транслиторы» Ғыш-сынау институты. Жүкелік; 1988-1989жж. - «Цифрлық сүрбаттарды верификациялау жүйесі» ғылыми-өндіріс орталығы. Зеленоград қ.; 1990-1991жж. - «Көппроцессорлы есептеу комплексі үшін параллелді программалау жүйесі» Электрондық есептеу техникасының ғылыми-зерттеу орталығы. Москау қ. Он 300-ге жуық ғылыми еңбектер, 5 оқулық, 10 оқу құралдары және 2 монография жарияланды. информатика мен есептеу техникасы бойынша 5 терминологиялық және түсіндірме сөздіктер шығарды; 30- дан астам зерделі меншікті мемлекеттік тіркеу куәліктерін алды. көптеген мемлекеттік стандарттарды жасауға қатысты: 10 – ақпараттық және коммуникациялық технологиялар саласында, 9 – білім беру саласында. А. Шәріпбай «Информатика, есептеу техникасы және басқарудағы математикалық әдістер» бойынша 5 ғылым докторын, 8 ғылым кандидатын және 4 PhD докторын даярлап шығарды. Оның ғылыми мектебі қазақ тілінің математикалық теориясын құрды, қазақ тілінің дауысты және жалғаушы дәуірі мен сөйлемдерін автоматтандырылған талдау мен синтездеу әдістерін жасады, электрондық оқыту бағдарламын дайындаудың технологиясын ұсынды. Бұл нәтижелер әртүрлі мемлекеттік және мемлекеттік емес құрылымдар мен ұйымдарда енгізілген көптеген электрондық оқыту бағдарламын, қазақ тілін қашықтан оқыту жүйесін, қазақша сөйлеуді тану мен синтездеу жүйесін және басқа автоматтандырылған жүйелерді жасау үшін қолданылды.