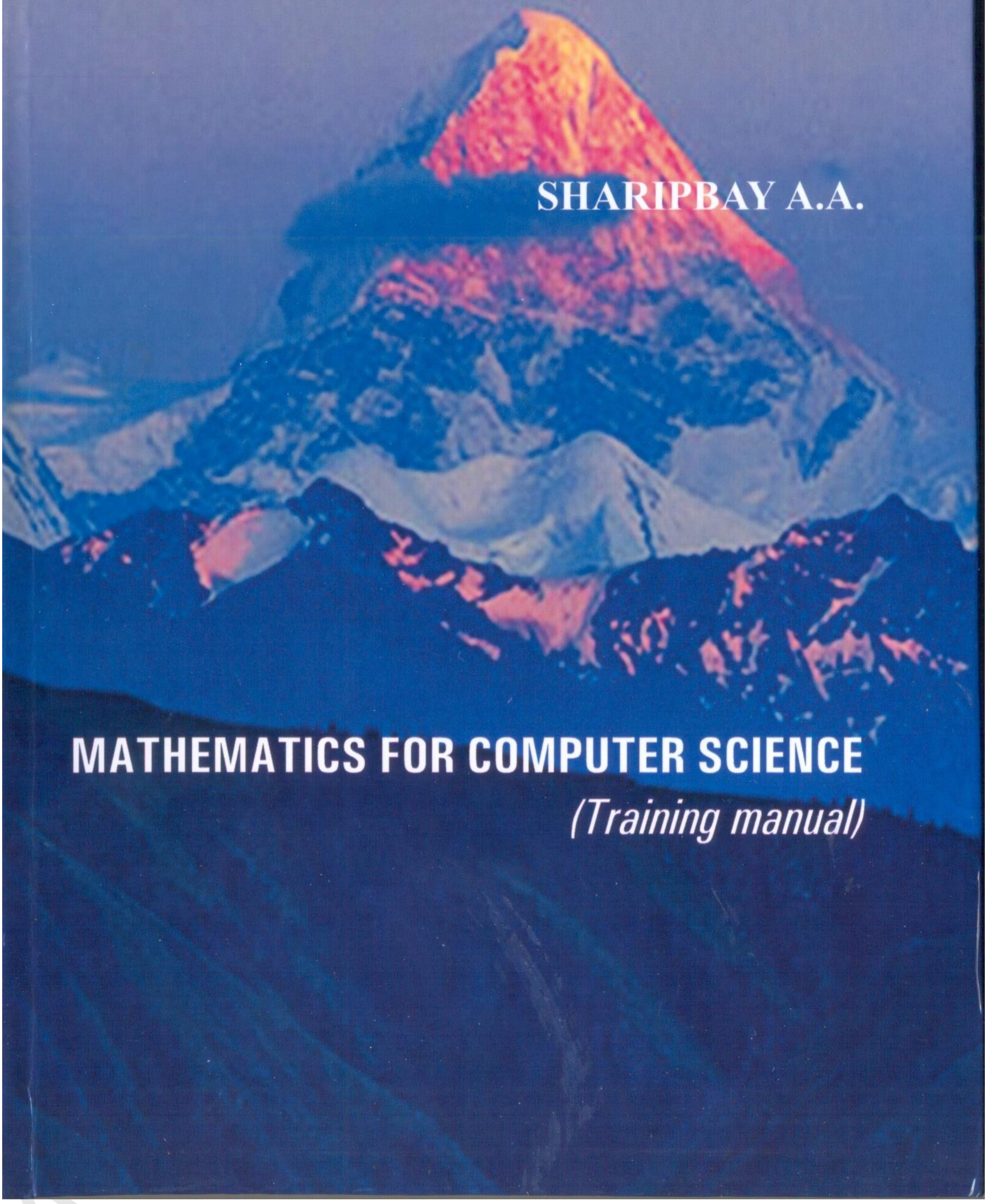
Co-funded by the
Tempus Programme
of the European Union

PROMIS
PROFESSIONAL MASTER'S DEGREES



SHARIPBAY A.A.

MATHEMATICS FOR COMPUTER SCIENCE

(Training manual)

Co-funded by the
Tempus Programme
of the European Union

PROMIS

SHARIPBAY A.A.

MATHEMATICS FOR COMPUTER SCIENCE
(Training manual)

Astana 2017

UDC 004.4(075.8)
LBC 32.973.26-018.1я73
SH-25

Reviewers:

Kaziev Galym Zulkhurnaevich – D.T.S., Professor, Director of the National Information Technologies JSC.

Atanov Sabyrzhan Kubeisinovich – D.T.S., Professor of the Chair of Computers Science and Software of the Lev Gumilyov Eurasian National University.

Raisa Kabievna Uskenbayeva – D.T.S., Professor, prorector of the International IT University;

Author:

SH-25 **Sharipbay A.A** – D.T.S., Professor, Laureate of State prize of the Republic of Kazakhstan, Director of the SRI Artificial Intelligence, Professor of the Chair of Computer Science and Information Security of the LN Gumilyov Eurasian National University,

SH-25 Sharipbay A.A. Mathematics for Computer Science – Training manual, -Astana, 2017, -153 pages.

ISBN 978-601-326-012-9

The content of the training manual 'Mathematics for Computer Science' is compiled in accordance with Silabus of the compulsory discipline of the same name for the professional master's study program 'Informatics as the Second Competence' in the speciality 6M060200-Informatics.

The manual is intended for studying the mathematical foundations of the computer science by undergraduates of an oriented educational program.

The manual can also be used by students, undergraduates and by all those, who independently wishes to study mathematical foundations of the computer science.

The training manual is published with the help of the International Project TEMPUS - 544319-TEMPUS-1-2013-1-FR-TEMPUS-JPCR (2013-4530/001/00) "Professional Master's Degree in Computer Science as a second Competencein Central Asia" .

UDC 004.4(075.8)
LBC 32.973.26-018.1я73

ISBN 978-601-326-012-9

CONTENT

I. INFORMATION AND DATA

I.1. Notation, types and measurement of information

I.1.1. Form and content of information

Information is the main object of Computer Science and comes from the Latin word "Information", which means to report something (face, subject, fact, event, phenomenon, process), regardless of the form of its presentation.

The concept of "Information" does not have an exact (mathematical) definition. If we try to give him an exact definition, then we again come to another vague notion and this can continue indefinitely. However, without knowing its exact definition, we can accept, understand, store, process and, if necessary, transfer to another entity. Taking into account what has been said, it is possible to give the following intuitive (not exact) definition of the concept "Information".

Information is a reflection of the properties and relationships of material and non-material objects and subjects of the surrounding world. This means that each information is characterized by its form (message) and its content (meaning).

In general, when we talk about a message, we need to remember the existence of its transmitter and receiver. They can be living organisms (people, animals, birds, fish, insects, etc.) and technical devices (telephone sets, radio stations and radio receivers, etc.). For example, if the receiver is a person, then he receives the message with his senses organ.

The message from the transmitter to the receiver is transmitted through a special medium, called the communication channel. For example, as such, an environment for an audio message, one can take air in which sound waves can propagate, and a communication channel for a written message can be paper on which you can write text.

The message is the bearer of the information value, i.e. the value of information is determined together with its message.

The scheme of information transfer is shown in Figure I.1.1.1.

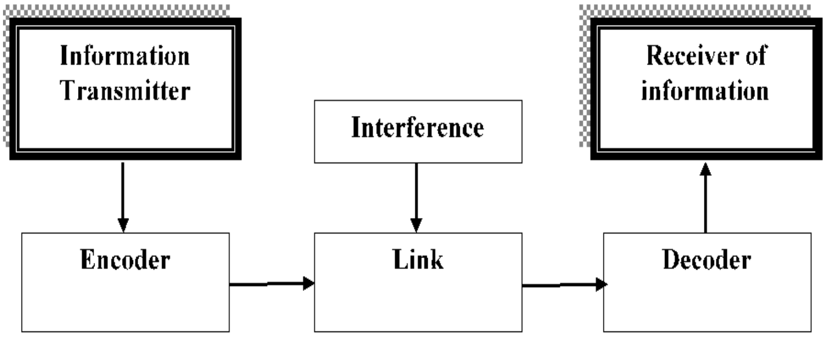


Figure I.1.1.1. Information transfer scheme

Note I.1.1:

1. ***A*** differently interpreted the same message is a bearer of different meanings. For example, readers, depending on their points of view, can perceive the same journal article in different ways.

2. The same value can be transmitted by different messages. For example, the report of the same person on the same topic presented, in different languages.

Therefore, between the transmitter and the receiver there must be a preliminary agreement on the form and the meaning of the message. Such an agreement is called the rule of interpretation.

Interpretation of a given message is taken from a general rule that can be applied to a set of messages built according to a single law. For example, if a message is given as a sentence in a known natural language, then the rule for interpreting such a sentence can be taken from the general rule of interpretation applied to all sentences of that language. To the general rule of interpretation is also the rule of determining the value of a number by its form of recording. Also, it can include to the rule of obtaining an assertion or denial from a message in the form of an assertion that has the meaning "true" or "false", or from a question that requires a "yes" or "no" answer. For example, "Snow has a white color", "A person never dies", "Is it interesting to study computer science?", "Is it clear?". Interpretation of such messages also depends on the experience and knowledge of the receiver.

When transmitting (receiving) a message, the state of the transmitter (receiver) varies with time. Therefore, we can consider the description of the material-energy state of the transmitter (receiver) as a function of $\boldsymbol{x}(\boldsymbol{t})$, depending on the time t. This function $\boldsymbol{x}(\boldsymbol{t})$ can be both continuous and discrete (discontinuous). Depending on this, the message of the information can be continuous (analog) and discrete (digital). For example, to the continuous message of the information is the graph of a continuous function of the temperature of the medium that changes in time, and to a discrete message of the information, an information message using the symbols of a certain communication language. Any continuous information can be turned into discrete information, this is called discretization. To digitize a function among its infinitely many values, a limited number of values are taken that can characterize the remaining values:

1) The axis of the graph's abscissa (domain of definition) of the function is divided into equal segments using a finite number of points

$t_1, t_2, \ldots, t_n$ and it is assumed that in each segment the value of the function is constant, for example, is equal to the average value in this segment;

2) Having designed the value of the function in each segment to the ordinate axis of the graph (area of change) of the function, it is necessary to find the points $x_1, x_2, \ldots, x_n$.

The points $x_1, x_2, \ldots, x_n$ found in this way will be considered as a discrete approximate representation of the continuous function $x(t)$. Its accuracy can be infinitely improved by reducing the lengths in the function definition area, until the required requirement is met.

The graphs of the continuous function $x(t)$ and its sampling are shown in Figure I.1.1.2.

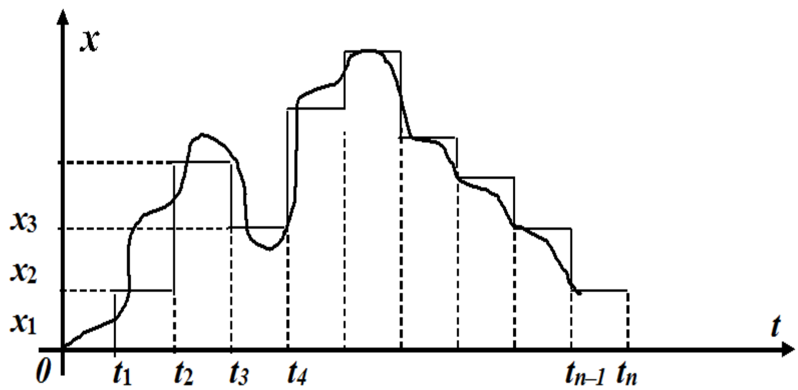


Figure I.1.1.2. Discretization of a continuous function

The possibility of discretization of continuous information makes it possible to represent a message of this information with the help of symbols of the alphabet of some communication language. This is very important for the reception, presentation, storage, processing and transmission of information using a computer.

The message is given in a material-energy form (for example: light, sound, movement, symbol, etc.). In other words, the form is the expression of some language. These languages include:

• Natural languages (Kazakh, Russian, English, etc.);

• Mathematical language (a set of expressions containing conditional designations of properties and relations of mathematical objects);

• Musical language (notes - many expressions, containing conditional designations of properties and relations of sounds in the sound series);

• Language of deaf-mutes (many expressions containing conditional movements of the face and hands);

• Artificial languages (programming languages, specification languages, design languages, etc.);

Since we are interested in the ways of presenting and processing information only on digital (discrete) computers, we will only consider discrete information types and corresponding notations (artificial languages), which are a system of conditional symbols used to represent concepts and their relationships, And also the rules of their application in any field of knowledge or activity.

The notation has its own alphabet, consisting of characters (letters) ordered in a certain sense, which will be used to construct words - a sequence of letters that has a certain meaning. The construction of words was carried out in accordance with the syntactic rules.

Examples I.1.1.

To see the relationship between a message and a value, consider a few examples. They are listed in Table I.1.1.

Table I.1.1. Examples of the message.

№	1-st message	2–nd message
1.	Мен жақсы оқимын.	Я учусь хорошо.
2.	From the flame ice will turn out	1>3
3.	Ты меня понял?	X=0?

4.	XXI	21
5.	Tomorrow it will snow	The lamp went out
6.	☼♀♂♥	© ®
7.	I will say.	Мен айтамын.

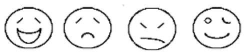
Examples 1,2,3,4 show that you can send the same value to two different messages: in the first example, the sentences in Kazakh and Russian have the same value, in the second example the value is "false" Have two messages. In the third example, questions that require "yes" or "no" answers are listed. In the fourth example, the same number is represented in two ways.

In the fifth example, the possibility of transmitting several values by the same message is shown: the first message means that it will get colder tomorrow or you can ride a sleigh.

In the sixth example, it is shown the possibility of communicating according to preliminary symbols. The values of such messages are only accessible to those who know the meaning of these signs.

In the seventh example, one meaning is conveyed by the sentences of two natural (English and Kazakh) languages.

Tasks I.1.1. What language are these expressions:

1. The Republic of Kazakhstan is an independent country.
2. a + (b + c) = (a + b) + c.
3. 

Questions I.1.1.

1. What is characterized by the information?
2. What are the types of communication languages?
3. What is the relationship between the message and the value?

Tests I.1.1.

1. Which language is the given expression (a+b+c)?

A) Mathematical;
B) Logical;
C) Mimicry;
D) Natural;
E) Chemical.

2. Between the transmitter and the receiver, what is the preliminary agreement about the type of message and its meaning?

A) Interpretation;
B) Integration;
C) Intervention;
D) Informatization.
E) Interpolation;

3. What is the relationship between the messages represented by the Roman numerals XIII and Arabic numerals 13 and their meanings?

A) The same value of a number is written in two ways;
B) Multivalued message;
C) ***A*** single-valued message;
D) One value and one message;
E) There is no connection.

I.1.2. Volumetric units and amount of information

The definition of the "Amount of information" is very difficult. To do this, you first need to determine the unit of measurement of information. It can be defined in two ways: volumetric and probabilistic. Both of these methods became known simultaneously in the 40s of the XX century. They were created by US scientists, one of the founders of the science of computer science John von Neumann and Claude Shannon.

John von Neumann was the first to show the possibility of building computers, this led to the definition of a measure of information in a three-dimensional way, and Claude Shannon defined the measures of information in a probabilistic way.

The minimum indivisible unit of information is called a ***bit***; it comes from two English words: ***bi****nary digi****t***. The reason for this was the convenience for computer developers to work with binary numbers while storing and processing information in the computer: a physical element having only multiple devices can implement two stable states, two stable states are denoted by binary digits 0 and 1. For example, it is easy to create a device showing the presence or absence of an electric current that measures a low or high voltage level, reveals the polarity of magnetization, etc.

The amount of information written in binary digits 0 and 1 is the number of binary digits used in this record; this number will always be an integer.

The next volumetric unit of information is called ***byte***, it consists of 8 bits, i.e. 1 byte = 2^3 bit. In a single byte, you can write = 256 different characters from each other, i.e. The capacity of one byte is 256 bits. This means that with a single byte, you can represent 256 different (non-repeating) messages.

The voluminous amount of information is a very large number. Therefore, large volumetric units of measurement are defined for the convenience of use. These units of measure are multiples of two, i.e. they must be powers of two. The list of voluminous units of information is given in Table I.1.2:

Table I.1.2. List of voluminous units of information.

№	Name	Symbol	Capacity
10	*byte*	B	10^0
11	*kilobyte*	Kb	10^3
12	*megabyte*	Mb	10^6
13	*gigabyte*	Gb	10^9
14	*terabyte*	Tb	10^{12}
15	*petabyte*	Pb	10^{15}
16	*exabyte*	Eb	10^{18}
17	*zettabyte*	Zb	10^{21}
18	*yottabyte*	Yb	10^{24}

Examples I.1.2.

1. If there are 50 pages in the book, 50 lines in each page, and 50 symbols in each line, then the volume of the book in bytes will be 400 * 50 * 50 = 1 000 000 Byte = 1 MB, i.e. In a disk of 1 GB you can save 1000 of these books.

2. If the size of the electronic book is 10 megabytes and the capacity of the electronic library is 100 gigabytes, then it can store
100 * 1024/10 = 102400 electronic books.

3. If the size of the electronic book is 10 megabytes and the capacity of the electronic library is 100 terabytes, then it can store 100 * 1024 * 1024/10 = 104857600 electronic books.

Tasks I.1.2.

1. What are the ways to measure the amount of information?
2. Find the volumetric size in the word "RIM".
3. Specify the capacity of one byte.

Help:

1. Ways by John von Neumann and Claude Shannon.
2. Information stored in the computer (word, number, figure, computer program) is written in binary digits.
3. It is necessary to specify the number of bits in one byte.

Questions I.1.2.

1. What is the minimum unit of information called?
2. Who determined the amount of information?
3. What should be the multiples of large units of information measurement?

Tests I.1.2.

1. How many bits are in one byte?
A) 8;
B) 10;
C) 2;
D) 0;
E) 17

2. How many megabytes will there be one GB?
A) 1024;
B) 1040;
C) 10024;
D) 124;
E) 102400.

3. How many gigabytes will one TB be?
A) 10024;
B) 1040;
C) 1024;
D) 124;
E) 102400.

I.1.3. Probabilistic units and amount of information

Before the introduction and discussion of the concept of "Probabilistic amount of information," let us consider one experiment relating to probability theory. As an example, you can take a throw N of a dice (the most common is $N = 6$). The result is the falling of the face, with the inscription of the digits 1, 2, ..., N.

We introduce a numerical measure-to-measure uncertainty, calling it ***entropy*** and denote by H. The data N and H will be in the following functional ratio:

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

Where the function f is nonnegative and increasing for our $N = 1, 2, ..., 6$.

Let us consider in more detail the throwing of a dice:

1) Preparation for throwing a bone: its outcome is unknown, i.e. there is some uncertainty, and we denote it by $H1$;

2) The dice are thrown: information on the outcome of the experiment is obtained; I will denote the number of this information;

3) We denote the uncertainty of this experiment after its realization through $H2$;

As the amount of information in the course of the experiment, it is possible to take the difference of uncertainties obtained before the experiment and after the experiment:

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

Obviously, in the case of obtaining a concrete result, the previously obtained uncertainty disappears, i. e. $H2 = 0$. Thus, the amount of information after the experiment will coincide with the initial entropy, i.e. $I = H1$. In other words, the uncertainty in experience coincides with information about the outcome of this experiment. Here, the value of $H2$ may not be zero, for example, during the experiment; the inscription of the next dropping edge is greater than 3.

The next important circumstance is the determination of the form of the function f in the formula (1.1). If N denotes the number of faces and M denotes the number of dice tossed, the total number of outcomes determined by the vectors of length M and consisting of N characters will be N to the power M

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

For example, in the case of two dice with six faces, we have $X = 6^2 = 36$. In fact, each outcome of *X* is a pair (*X1*, *X2*), where *X1* and *X2* are the outcomes of the first and second casts respectively, and *X* is the total number of such pairs.

The situation with throwing *M* times can be considered as a kind of complex system consisting of independent subsystems - "single casting of a bone". The entropy of such a system is *M* times larger than the entropy of one system (the principle of additivity of entropy):

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

This formula can be extended to the case of any N:

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

Now let us logarithm the left and right parts of the formula (1.3):

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

Then *M* is as follows:

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

We substitute the value obtained for *M* into formula (1.4):

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

Denoting by *K* a positive constant $\frac{f(N)}{\ln N}$, we obtain:

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

Or, taking into account (1.1), $H = K \cdot \ln N$.

Usually accept $K = \frac{1}{\ln 2}$. This gives the *Hartley formula:*

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

Important for the introduction of any data is the question of what to take as a unit of its measurement. Obviously, $H = 1$ for $N = 2$. In other words, as a unit, the amount of information is taken to carry out an experiment consisting in obtaining one of two equiprobable outcomes (an example of such an experience is the coin toss, in which two outcomes are possible: "eagle "," Tails "). This unit of information is called a "bit".

All N outcomes of the above experiment are equally probable, and therefore we can assume that the "fraction" of each outcome is one N-th part of the total uncertainty of the experiment:

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

Moreover, the probability of the i-th outcome of Pi is obviously 1 / N. Thus, the entropy is found by Shannon's formula as follows:

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

The same formula (1.6) is taken as the measure of entropy in the case when the probabilities of different outcomes of the experiment are unequally probable (that is, P. may be different).

Note I.1.3:

The relationship between the volumetric and probabilistic amounts of information is ambiguous. Not every text written with binary symbols allows measuring the amount of information in a probabilistic sense, but certainly admits it in a voluminous sense. Further, if some message allows measurability of the amount of information in both senses, then they do not necessarily coincide, while the probabilistic amount of information cannot be greater than the volume one.

Examples I.1.3.

1. Consider an alphabet consisting of two signs 0 and 1. If we assume that the identical probability of their occurrence is associated with the signs 0 and 1 in the binary alphabet ($P(0) = P(1) = 0.5$), then by the formula (1.5) the amount of information per character with binary coding will be equal to

$$H = log_2 2 = 1 \text{ Bit}.$$

Thus, the amount of information (in bits), enclosed in a binary word, is equal to the number of binary signs in it.

2. Define the amount of information associated with the appearance of each character in messages written in Russian. We will assume that the Russian alphabet consists of 33 letters and a "space" sign for the separation of words. By the formula (1.5)

$$H = \log 34 = 5 \text{ bits.}$$

However, in the words of the Russian (as well as in the words of other languages), different letters occur unevenly often. By the formula (1.6): H = 4.72 bits.

Tasks I.1.3.

1. Let in a box there are 16 different colored balls. You need to calculate the amount of information using entropy to pull out the white ball.
2. Determine the amount of information associated with the appearance of each character in messages written in Kazakh in the 42-letter alphabet.
3. Determine the amount of information associated with the appearance of each binary digit in messages recorded in one megabyte.

Help:

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

2. To determine the amount of information, use the formula (1.5).
3. It should be noted that there are 10^6 bits in one megabyte.

Questions I.1.3.

1. What are the properties of the function for computing entropy?
2. What is the principle of additivity of entropy?
3. What is the term that measures uncertainty?
4. Is the relationship between volume and probability units of information unambiguous?
5. Why the Hartley formula?

Tests I.1.3.

1. What is the minimum indivisible unit of volumetric measurement of information?

A) Terabyte;

B) Byte;

C) Megabyte;

D) Gigabyte;
E) Bit.

2. What is the magnitude of the uncertainty measurement for Chenon?
A) Ectropion;
B) Entropy;
C) Value;
D) Information;
E) Byte.

3. What is the Hartley formula?
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. Types and types of data

I.2.1. Constant and variable types of data

In Computer Science, the concept of "Information" is often replaced by the concept of "Data", while the information message is treated as a data name, and the value of information is treated as a data value.

The data, depending on the ways of giving values to their names, are divided into constant data (constant) and variable data (variables).

If data values are determined during the construction of common rules for interpreting the used communication language, then such data will be permanent. In other words, when naming a constant, its values are simultaneously determined. Together with the value, the type of constant becomes known. For example, if the chain 135, formed from digits 1, 3, and 5, is treated as a constant name, then this value will be an integer "one hundred and thirty-five". If you construct another chain 315 from the same numbers, then it will be the name of the constant with the value "three hundred and fifteen", which are an integer.

This implies the following statement: the names, values and types of constants do not change, they are determined simultaneously.

If data values change during their processing, then such data will be variable. The type of the value of the variable must not be changed. Otherwise, when processing this variable, there will be difficulties associated with the incompatibility of the data types involved in the processing.

In computer science, the term "identifier" is used to name a variable.

An identifier is a chain with a limited length that starts with a letter and consists of letters and numbers.

The value of a variable is described by its type, which uniquely identifies:

A) the admissible values that an object of the described type can have;

B) permissible operations that can be applied to an object of the type described.

In general, data values are divided into numerical, symbolic and logical types.

To process data, you must apply operations to this data. And in order to apply operations you need to know their definitions, notations and properties.

The operations, depending on what types they are defined, are divided into numeric operations, character operations, and logical operations.

It should be noted that any of these operations is defined and investigated in various sections of mathematics. For example, all symbolic operations are defined in the section "Mathematical Linguistics", which explores the composition and properties of languages, all logical operations are defined in the section "Mathematical Logic", and all numerical operations are in other sections of mathematics (arithmetic, algebra, etc.).

You can define the properties of the operations defined on each data type. These properties are grouped and form an axiomatic with respect to these data.

In the proposed axiomatics intended for data of various types, there are many similarities. They show equivalence, including regularities such as *commutativity*, *associativity and distributivity*.

Such patterns simplify the complex expression, reducing the number of operations, and facilitate its calculation.

Examples I.2.1.

1. N12B is the identifier.
2. 7X is not an identifier, because it starts with the number 7.
3. $A + B$ is not an identifier, since there is a "+" sign in it.

Tasks I.2.1. Will these data be identifiers?

1. A-B;
2. XY;
3. C6R7.

Help:

The identifier begins with a letter and consists of letters and numbers.

Questions I.2.1.

1. What types of data are available?
2. Why do I need an identifier?
3. What is the difference between constant and variable data?

Tests I.2.1.

1. What is the concept of replacing information?
A) Data;
B) Identifier;
D) The message;
C) Presentation;
E) Definition.

2. Which data does the name, value, and type do not change?
A) Constant data;
B) Certain data;
D) Variable data;
C) Not exact data;
E) Uncertain data.

3. How is the data divided according to the way the value is assigned?
A) Constant and variable data;
B) Persistent and non-permanent data;
C) Variables and non-persistent data;
D) Unstable and accurate data;
E) Valid and non-permanent data.

I.2.2. Numeric data types

A *numeric type* is formed from a set of numbers, operations on these numbers and the properties of these operations. They are divided into three: integers, real numbers and complex numbers.

Note I.2.2:

In computer science, the processing of numeric data may require different number systems. The basis of such systems can be 2,3,4, ... The difference between them is only in the methods of denoting the values of numbers, but the types of operations on numbers and their properties will be the same. Therefore, first consider a well-known to us the notation of numbers in the decimal number system, operations on them and the properties of these operations, since everything said, connected with the decimal number system, also applies to numbers in another number system. The integers are represented by Arabic numerals, before their negative values the sign "-" is written, and before positive values the sign "+" can be written.

Real numbers, depending on the mode of representation, are divided into two groups: real fixed-point and real floating-point.

The representation of real with a fixed point consists of a whole and fractional parts. The whole part is placed before (on the left side) by a fractional part and they are separated by a "." Sign, called a decimal point. Before them, to indicate the positivity or negativity of their values is written "+" or "-". Both parts are represented by Arabic numerals.

The representation of real floating points consists of parts called mantissa, the basis of the number system and order. The values of the mantissa, the base of the number system and the order can be positive or negative. To indicate them, "+" or "-" is written before the values. The order is represented as an integer, and the mantissa is represented as a real number with a fixed point.

If we denote the mantissa by $\boldsymbol{M}$, the order of $\boldsymbol{p}$, the basis of the system of contraction by $\boldsymbol{q}$, then the real numbers are as follows:

$$\boldsymbol{M} * \boldsymbol{q}^{\boldsymbol{p}} .$$

To understand what was said in Table I.2.2, examples of real numbers with floating point are examined.

Table I.2.2. Examples of floating-point real numbers.

№	Example	Mantissa	The order of	Value
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

The same number in a floating-point form can be represented by many records. For example, the same number 3.14 can have the following entries: $314*10^{-2} = 31.4*10^{-1} = 3.14*10^0 = 0.314*10^1 = 0.0314*10^2 = \ldots$To have a single entry to represent a real floating-point number, you need to normalize it. For this, the following condition must be fulfilled:

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

where $|M|$ is the absolute value.

For example, real numbers with a floating point$13.64*10^2$ and $0.00617*10^{-5}$ in the normalized form will be:

$$0.1364*10^4 \text{ and } 0.617*10^{-7}.$$

Complex numbers $0.1364*10^4$ and $0.617*10^{-7}$ are represented in the form of an algebraic sum of their parts: the first (left) summand is the real part, the second (right) summand is the complex part. The real part and complex part of the complex number are written in the form of real numbers. To distinguish them after the complex part, the lowercase Latin letter "i" is put as its sign.

Examples I.2.2.

1. 3.14 is a positive real number with a fixed point, the whole part is 3, and the fractional part is 14.

2. +5 is a positive integer 5.

3. 0.2 is a positive real number with a fixed point, the whole part is 0, and the fractional part is 2.

4. -1.001 - a negative real number with a fixed point, an integer part of 1, and a fractional part of 001.

5. 0.0 - positive real number, integer part 0 and fractional part 0.

6. 0 + 3*i* is a positive complex number, the real part is 0, and the imaginary part is 3.

7. -3.14 + 2*i* - negative complex number, real part -3.14, and imaginary part 2.

8. - 0.12*i* is a positive complex number, the real part, and the imaginary part is 0.12.

Tasks I.2.2.

Determine real floating-point numbers and complex numbers:

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

Help:

1) ***A*** real number with a fixed point consists of an integer and fractional parts.

2) The complex number has a real part and an imaginary part, at the end of which the lowercase Latin letter i is written.

3) ***A*** real number with a floating point has a base of the number system, a mantissa and an order.

Questions I.2.2.

1. What types of numerical data are available?

2. Which groups are divided into real numbers, depending on the form of their presentation?

3. What parts consist of complex numbers?

Tests I.2.2.

1. What should I use to normalize real numbers?

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

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

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

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

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

2. What is the real part for 0 + 3i, and what is the imaginary part?
A) Actual part i, imaginary part 3;
B) Real part 0, imaginary part 3;
C) Actual part 3, imaginary part i;
D) The real part i, the imaginary part 0;
E) The real part 0, the imaginary part of i.

3. What numbers represent any integer?
A) Latin numerals;
B) Greek numerals;
C) Kazakh numerals;
D) Russian figures;
E) Arabic numerals.

I.2.3. Numerical operations and their properties

Operations defined over numeric types are known to us from school as arithmetic operations: addition, subtraction, multiplication, division. They are indicated by the signs "+", "-", "*", "/", respectively. Using these operations, you can build numerical expressions. Usually, for writing numeric expressions, we use an infix entry in which operation characters are written between operands (arguments). For example, if for any numbers a and b there is a third number that is the sum of these numbers, then we will write it as a + b.

When calculating the values of numerical expressions, it is necessary to take into account the priorities (execution order) of these operations: first, high-priority operations, multiplication and division, then low-priority operations addition and subtraction. Sometimes, parentheses are used to change the order of operations. In one expression, you can use several parentheses and you can write them inside each other while the operation in the innermost bracket and in the leftmost bracket is performed first.

There are also non-skewed ways of writing numeric expressions, called prefixes and postfix records. In the prefix record, operation signs are written before their operands, and in postfix entries, the operation signs are after their operands. For example, the numeric expression in the infix record (x+3)*(y-2) is written in the prefix entry as * + x 3 - y 2, and in the postfix record as x 3 + y 2 - *.

Note I.2.3:

1. The rules (but not the order) of performing these operations on different types will be different, even though they are designated identically. For example, the rule of summing integers is not suitable for adding real numbers.

2. Prefix record and postfix recording of an arithmetic expression, in honor of their author, the Polish mathematician Jan Lukashevich, are called a direct Polish record and a reverse Polish record, respectively.

Let a, b, and c be any numbers and let a be the inverse of a. Then operations on the numbers "+" - addition and "*" - multiplication have the properties shown in Table I.2.3.

Table I.2.3. The properties of numerical operations.

№	Axiom	Description
1	$a+b=b+a$	Commutative law
2	$a*b=b*a$	
3	$a+(b+c)=(a+b)+c$	Associativity law
4	$a*(b*c)=(a*b)*c$	
5	$a*(b+c)=a*b+a*c$	Distributivity law
6	$(a+b)*c=a*c+b*c$	
7	$a+0=0+a=a$	Addition properties
8	$a+(-a)=(-a)+a=0$	
9	$a*1=1*a=a$	Multiplication properties
10	$a \neq 0, a*a^{-1}=a^{-1}*a=1$	

Axiom 8 says that for every number a there exists an opposite number -*a*, and axiom 9 - for every nonzero number *a* there is an inverse number.

Examples I.2.3.

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.

Tasks I.2.3.

1. Find the opposite number to the number 6;
2. Calculate 2**a*+*3***b* for *a*=5 and *b*=8;
3. Calculate (a + 5) * (3 * b) for a = 10 and b = 2.

Help:

1. $a*a^{-1}=a^{-1}*a=1$;
2. Take into account the order of execution of transactions;
3. Take into account the order of execution of operations and brackets.

Questions I.2.3.

1. What is a prefix entry?
2. What is an infix entry?

3. What is postfix recording?

Tests I.2.3.

1. Find the law of commutativity?

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. Find the law of distributivity?

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. Find the law of associativity?

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. Character data type

A symbolic type is formed from a set of alphabet character chains of a given communication language. In computer science, the alphabets are composed of a group: *letters*, *numbers* and *special signs*.

Among the letters can be lowercase and uppercase letters. Sometimes to the alphabet of one language can be added letters of the alphabet of another language . For example, in the alphabet of modern Kazakh language, there are letters of the Russian: *ё,э,ю,я*, *ц*, *ч*, *щ*, *ь*, *ъ*.

As digits are taken arabic numerals: *0*, *1*, *2*, *3*, *4*, *5*, *6*, *7*, *8*, *9*.

Special signs include *signs of used operations*, *punctuation marks*, *grouping marks* (brackets), a *space mark*, etc.

The values of the character data are enclosed in character brackets. The symbol for the apostrophe « ' ». is used as the symbol bracket. For example, 'ABC', '2015', 'X+1 > 0', '(3,14)'.

So, the value of the character data is represented by a chain of letters, numbers or special characters enclosed in character brackets. You can determine the length of each chain: it is equal to the number of characters in this chain. The length is denoted between two vertical lines "| ". For example, the lengths above the given symbol values are represented as:

||ABC| = 3, |2015| = 4, | X+1 > 0| = 5, |(3,14)| = 6.

The abstract value "empty chain" is included in the set of character data. As part of the empty chain, there is not a single symbol. Usually an empty chain is denoted by the sign "ε". The length of the empty chain is zero, |ε | = 0.

Note I.2.4.

A space is not an empty chain, it is considered a real symbol. The length of the space is one, | | = 1.

Examples I.2.4:

1) 'ABCDE' is a chain formed from the initial letters of the Latin alphabet.

2) 'X + Y = 100' is a chain formed from mixed signs.

3) 'Al' is a chain formed from a Latin letter and a digit.

Tasks I.2.4.

Determine the type of the following chains:

1. 'ABC';

2. '2013';
3. 'X + Y> 1'.

Questions I.2.4.

1. What does the length of character data mean?
2. What is the length of the symbol chain 'ABCD'?
3. What is the length of the empty chain?

Tests I.2.4.

1. From which groups are alphabets of communication languages used in computer science?

A) Consists of a group of letters, numbers and special signs.
B) Does not consist of a group of letters, numbers and special signs.
C) Consists of a group of numbers and special signs.
D) Consists of only a group of special signs.
E) Consists of only a group of letters and numbers.

2. Which chain consists of mixed signs?

A) '210';
B) 'ABC';
C) 'X + 1> 0';
D) '( )';
E) '96'.

3. What is the length of the chain |ABCDFHTRY| ?

A) 0;
B) 1;
C) 10;
D) 9;
E) 11.

I.2.5. Symbolic operations and their properties

All operations on character data are basically divided into two groups:

1) *Construction* operations that allow one character chain to be obtained from the given two character chains;

2) *Division* operations that allow a symbol or part of this chain to be removed from a given character chain, depending on the specific condition.

The simplest operation of construction is the operation concatenation (coupling), it is denoted by the sign "•". For chains X and Y, it is defined as follows: after the right value of X is bound to the value of Y, we obtain a chain Z and write it as X • Y = Z. For example, if the value of X is 'KAZAKH', and the value of Y is 'STAN', then the value of Z is ' KAZAKH'STAN'. In some languages, the sign of this operation is not recorded.

The operation of construction involves the operation disjunction (choice), denoted by the sign "|". For example, if A = 'ALMATY', B = 'ASTANA', then C = A|B = 'ALMATY' | 'ASTANA'.

Another operation of construction is the iteration operation, it is denoted by the "*" symbol. This operation is a derivative (complex), which is defined through operations concatenation and disjunction. For example, for any character data X, you can define:

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

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

To properly construct a symbolic expression using construction operations, you need to know the order in which they are executed. The following order is established: the first is the iteration operation "*", the second is the concatenation operation "·", the third is the operation disjunction "|". Parentheses are sometimes used to indicate the order of operations. For example, for these two expressions *0|10** and (*0*|(*1*(*0**))) the values are the same.

The simplest operation of division is the operation of removing from a given chain a certain number of characters starting at the specified location. If the name of the operation division is denoted by ***DEL***, then we can write the deletion of the symbols from the chain X in the number M, starting from the position K in the form of the function:

$$\boldsymbol{DEL}(X, K, M) = Y,$$

Where the succession of the chain Y is obtained after removing M symbols from the value of X, starting from the position of K, then we can write

Let α, β and γ be arbitrary nonempty character data and ε an empty chain. Then the properties of the construction operations defined over the symbol data, "•" - concatenation, "|" - disjunction and "*" - iteration are shown in Table I.2.5.

Table I.2.5. Properties of character operations.

№	Axiom	Description
1	$\alpha\cdot\varepsilon = \varepsilon\cdot\alpha = \alpha$	The commutativity law for empty chains
2	$\alpha \mid \beta = \beta \mid \alpha$	The commutativity law for non-empty chains
3	$\alpha\cdot\beta \neq \beta\cdot\alpha$	The noncommutativity law for non-empty chains
4	$\alpha\cdot(\beta\cdot\gamma) = (\alpha\cdot\beta)\cdot\gamma$	The law of associativity
5	$\alpha \mid(\beta \mid \gamma)=(\alpha \mid\beta) \mid \gamma$	The law of associativity
6	$\alpha\cdot(\beta \mid \gamma)=\alpha\cdot\beta \mid \alpha\cdot\gamma$	The law of distributivity
7	$(\alpha \mid\beta)\cdot\gamma=\alpha\cdot\gamma \mid \beta\cdot\gamma$	The law of distributivity
8	$\alpha\mid\alpha = \alpha$	The reduction law
9	$(\alpha^*)^*=\alpha^*$	The reduction law
10	$\alpha^*= \varepsilon\mid\alpha^2\mid\alpha^3\|\ldots\mid\alpha^n\mid\ldots$	The iteration law
11	$\alpha^+ =\alpha \mid\alpha^*$	The iteration law

1. The chain α is a prefix (the beginning) of the chain β if there is a chain ξ such that the equality $\beta = \alpha\xi$ holds. If we denote by $\alpha\sqsubset\beta$ the relation "the chain α is a prefix of the chain β", then its formal definition can be written as

$$\alpha\sqsubset\beta| \rightleftharpoons \exists\xi(\beta = \alpha\xi).$$

2. The chain α is the suffix (end) of the chain β if there is a chain ζ such that $\beta = \zeta\alpha$. If we denote by $\beta\sqsupset\alpha$ the relation "chain α is a suffix of the chain β", then its formal definition can be written as

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

3. The chain α is a subchain of the chain β if there are chains ζ and ξ such that the equality β = ζαξ holds. If we denote by α⊆β the relation "chain α is a subchain of β", then its formal definition can be written as

$$\alpha \subseteq \beta \rightleftharpoons \exists\zeta\exists\xi(\beta = \zeta\alpha\xi|).$$

Examples I.2.5.

1. If the value of *X* is 'ALTYNBEK', *K* = 1, and *M* = 5, then
DEL(*X*, *K*, *M*) = ***DEL***('ALTYNBEK', 1, 5) = 'BEK'

2. If the value of *X* is 'ALA', and the value of *Y* is 'TAU', then *X* • *Y* = 'ALATAU' ≠ *Y* • *X* = 'TAUALA'.

3. If the value of *X* is ' LONDON', *K* = 4, *M* = 3, then
DEL(*X*, *K*, *M*) = ***DEL***(' LONDON', 4, 3) = 'DON'.

4. $\varepsilon \subset abcd$, $a \subset abcd$, $ab \subset abcd$, $abc \subset abcd$, $abcd \subset abcd$.

5. $abcd \supset \varepsilon$, $abcd \supset d$, $abcd \supset cd$, $abcd \supset bcd$, $abcd \supset abcd$.

Tasks I.2.5.

1. If ***A*** = 'ALA', ***B*** = 'TAU', then ***A*** • ***B*** = ?,

2. If *X* = 'STRASBOURG', *K* = 6, *M* = 5, then ***DEL*** (*X*, *K*, *M*) = ?

3. If *X* = 'BERLIN', *Y* = 'BER' then *X* ⊃*Y* or *Y* ⊂*X*?

Help:

1. You need to know the construction operations;

2. It is necessary to know the operations of division;

3. You need to know the relationship over the chains.

Questions I.2.5.

1. How many groups of operations are defined over the symbol data?

2. What operations determine the operation of iteration over the symbol data?

3. What is such an empty chain?

Tests I.2.5.

1. If ***A*** = 'SHYM', ***B*** = 'KENT', then *A*•*B* =?

A) SHYMKENT;

B) SHYM;
C) KENT;
D) KENTSHYM;
E) THE SHYN.

2. If X = 'COPENHAGEN', $K = 6$; $M = 5$, then ***DEL***$(X, K, M) = ?$
A) HAGEN;
B) COPEN;
C) COP;
D) HAG;
E) OPEN.

3. What kind of axiom is $\alpha \mid \beta = \beta \mid \alpha$?
A) Non-commutativity for non-empty chains;
B) Commutativity for empty chains;
C) The law of distributivity for empty chains;
D) The associativity law for empty chains;
E) Commutativity for non-empty chains.

I.2.6. Boolean data type

A logical type consists of logical values. The logical values are "true" and "false".

Boolean values are given in the following form:

1) "true" or "yes" or "1" or "+";

2) "false" or "no" or "0" or "-".

These values are the results of certain conditions. Such conditions include statements that take the values "truth" or "lie," and questions that require answers "yes" or "no." Examples of logical values are the values of messages from rows 1, 2 and 3 in Table I.1.2.

In different literary sources, instead of a logical value (logical constant), "false" and "no" use "false", "non" or "0", and instead of a logical value (a logical constant), "true" and "yes" - "true" , "Yes" or "1". Therefore, in the sequel, for convenience only 0, 1 can be used as logical values. Then we assume that any logical variables take values from the set {0, 1}.

The simplest condition is formed by the comparison operation (relationship): "<" - "less"; «=» - «equal»; ">" - "more"; "≤" is "less than or equal to"; "≥" is "greater than or equal to".

To build a complex condition, you need to use logical operations defined over logical values. Types of such operations and their properties will be considered below in the section "Logical operations and their properties".

Examples I.2.6.

1. The simple condition "2 less than 5" has the meaning "true".
2. The simple condition "1 equals 3" has the meaning "lie".
3. The simple condition "A is less than or equal to 7" will have the value "true" or "false" depending on the value of the numerical variable A.

Tasks I.2.6.

Determine the value of "true" or "false":

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

Help:

A logical value is the result of a certain statement that takes the values "true" or "false", or questions that require "yes" or "no" answers.

Questions I.2.6.

1. Values that define a logical type?
2. How is the opposite logical value defined?
3. What is the meaning of the statement "It's not true that the computer is smarter than a person"?

Tests I.2.6.

1. Determine the logical value 45 = 60?

A) Lies;
B) Truth;
C) Approximately;
D) Equal to;
E) Imaginary.

2. Determine the logical value 17 <48?

A) Truth;
B) Falsehood;
C) Approximately;
D) Equal to;
E) Imaginary.

3. Determine the logical value -12≥0?

A) Truth;
B) Falsehood;
C) Approximately;
D) Equal to.
E) Imaginary.

I.2.7. Logical operations and their properties

Logical operations are diverse. We among them will consider the most simple. This operation is not, and, or. Using these operations, you can define any complex logical operation, i.e. They allow you to build any complex logical expression (condition).

Depending on the language of communication, the logical value, comparison operations (relations), and the designations of logical operations may be different. For example,

1) Comparison operations (relations):

"," - "less";

«=» - «equal»;

">" - "more";

2) Logical operations:

"¬" - "inversion" or "no";

"&" or "∧" - "conjunction" or "and";

"|" or "∨" - "disjunction" or "or".

In any logical expression, logical relationships must be performed before logical operations, and among the logical operations of the first one, the operation "inversion", then the operation "conjunction", and at the end the "disjunction" operation.

Values (models) of logical operations can be determined using truth tables. For example, if the logical variables ***A*** and ***B*** are given, then the logical operations can be defined as follows:

1. *Inversion* (*no*):

A	$\neg A$
0	1
1	0

If the value of the logical variable ***A*** is "false", then its negation will be "true" and vice versa.

2. *Conjunction* (*and*):

A	B	$A \wedge B$
0	0	0

1	0	0
0	1	0
1	1	1

The result value will be "true" if and only if both variables ***A*** and ***B*** take the value "true", otherwise the result has the value "false".

3. *Disjunction* (*or*):

A	***B***	***A∨B***
0	0	0
0	1	1
1	0	1
1	1	1

The result value will be "false" if and only if both variables ***A*** and ***B*** take the value "false", otherwise the result has the value "true".

Before discussing the properties of logical operations, consider comparison operations that are defined over any types of data. ***A*** comparison operation can be defined between quantities or data measurements and their distinguishable characteristics.

Comparison operations have many kinds. But, despite this, the results of all comparison operations can be represented using the logical values "true" and "false". In addition, all comparison operations have common properties.

Let ***R*** be a comparison operation defined over the data *a*, *b*, *c*. Then the operation *R* has the following properties:

1) *reflexivity* if for each *a*, *aRa*;

2) *transitivity*, if for each *a*, *b* and *c* of *aRb* and *bRc*, *aRc* follows;

3) *symmetry*, if for each *a* and *b* from the execution of *aRb* follows the execution of *bRa*.

Now, instead of ***R***, we will use well-known notations "<" - less, "=" - equal and ">" - more and write the following properties:

1. For any data *a* and *b*, the operation $a < b$, $a = b$ or $a > b$ will be in only one comparison:

2. If $a > b$ and $b > c$, then $a > c$.

3. If $a = b$ and $b=c$, then $a=c$.

4. If $a < b$ and $b < c$, then $a < c$.

Let p, q, r be arbitrary logical data and 0 - false, 1 - true. The properties of logical operations are given in Table I.2.7.

Table I.2.7. Properties of logical operations.

№	**Axiom**	**Description**
1	$p \wedge q \sim q \wedge p$	The law of commutativity
2	$p \vee q \sim q \vee p$	The law of commutativity
3	$p \wedge (q \wedge r) \sim (p \wedge q) \wedge r$	The law of associativity
4	$p \vee (q \vee r) \sim (p \vee q) \vee r$	The law of associativity
5	$p \wedge (q \vee r) \sim (p \wedge q) \vee (p \wedge r)$	The law of distributivity
6	$p \vee (q \wedge r) \sim (p \vee q) \wedge (p \vee r)$	The law of distributivity
7	$\neg(p \vee q) \sim \neg p \wedge \neg q$	De Morgan's Law
8	$\neg(p \wedge q) \sim \neg p \vee \neg q$	De Morgan's Law
9	$\neg(\neg p) \sim p$	The law of double negation
10	$p \sim p$	The law of identity
11	$p \vee \neg p \sim 1$	The law of exclusion of the third
12	$p \wedge \neg p \sim 0$	The law of contradiction
13	$p \wedge p \sim p$	Conjunction property
14	$p \wedge 1 \sim p$	
15	$p \wedge 0 \sim 0$	
16	$p \wedge (p \vee q) \sim p$	
17	$p \vee p \sim p$	Disjunction property
18	$p \vee 1 \sim 1$	

19	$p\vee 0 \sim p$	
20	$p\vee(p\wedge q) \sim p$	

Examples I.2.7.

1. In the expression A∨¬B ∧ C, first ¬B, then ¬B ∧ C, and at the end A ∨ ¬B ∧ C.

2. In the expression A = 0 ∨ B > 1, the comparison operation ***A*** = 0 and ***B*** > 1 is performed first, then the logical operation ∨.

3. In the expression ¬(A = 0) ∧ B = 1, the comparison operation ***A*** = 0 and ***B*** = 1 is performed first, then the logical operations ¬, ∧.

Tasks I.2.7.

In this expression, perform operations and determine its logical value.

1) (1 < 2) ∧ 1=3 ∨ 2 <5;

2) 1>3 ∨ 1<2;

3) 1<3 ∨ 5=7;

Help:

Among the logical operations of the very first operation is the operation "inversion", then the operation "conjunction", and at the very end the "disjunction" operation.

Questions I.2.7.

1. What logical operation is denoted by the signs «&», «∧»?
2. What values have empty places in this table?

A	B	$A\vee B$
0	0	
0	1	
1	0	
1	1	

Tests I.2.7.

1. Which logical value (0 or 1) will have a logical expression 2>5 ∨ 2<6?

A) 2;
B) 1;
C) 5;
D) 6;
E) 0.

2. What order of operations in the expression
D∨¬F ∧ G?
A) first ¬F, then ¬F ∧ G, and at the end D ∨¬F ∧ G.
B) first ¬F ∧ G, and at the end D ∨¬F ∧ G.
C) first ¬F, and at the end D ∨¬F ∧ G.
D) first ¬F, then ¬F ∧ G.
E) first ¬G, then ¬G ∧ D, and at the end D ∨¬F ∧ G.

3. Which of them is De Morgan's law?
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. Coding Information

I.3.1. Coding of logical and symbolic information

In paragraph I.1.1, a representation of a discrete information message was shown through the alphabet symbols of a certain language (notation), which is very important for receiving, presenting, storing, processing and transferring information using a computer.

It should be noted that a person can distinguish characters by their outlines, and a computer only by their codes consisting of sequences 0 and 1, since the physical storage devices in the computer (memory cells and registers) can only be in two states that are correlated 0 or 1. Using a number of similar physical devices, you can store any information in the computer's memory with the help of binary code in the form of sequences 0 and 1. Therefore, the notation of any (numerical, text, logical, graphic, audio, etc.) The computer with which modern computers operate is coded (converted) into binary code, and decoded (converted back) into a notation to facilitate human perception.

In a general sense, the encoding of information can be defined as the translation of information represented by a message in the primary alphabet into a sequence of codes. It must be understood that any data is somehow encoded information. Information can be presented in various forms: in the form of numbers, text, picture, sound, etc. Translation from one form to another is coding.

Logical information that has only two values of "False" and "True", regardless of their notation, is represented in computer as 0 and 1, respectively.

From paragraph 1.3.8 it is shown that using logical operations $\neg$, &, $\vee$, denoting the words "*not*", "*and*" and "*or*" ", you can build any complex logical expression.

In modern computers, all three logical operations $\neg$, & and $\vee$ are implemented in hardware with the help of basic logical elements of the computer.

The basic logical elements of the computer, indicating their input and output, are shown in Figure 1.3.1.1:

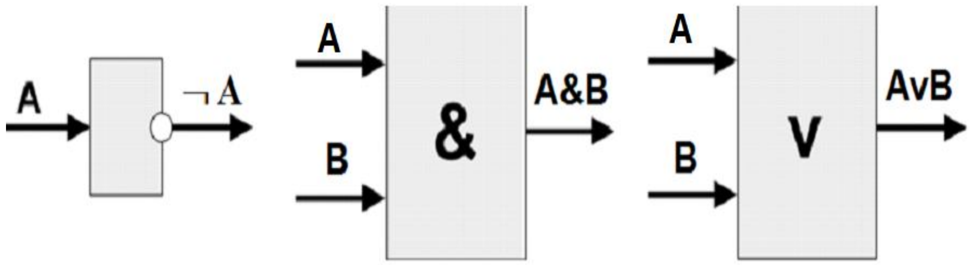


Figure 1.3.1.1. Basic logical elements.

When you enter symbolic information into the computer, its binary coding takes place, the character code is stored in the RAM, and when the symbol is output to the printer or to the computer screen, decoding takes place, i.e. The conversion of the symbol code into its image.

The code for each character is specified by the index values in the code table of a two-dimensional array containing this symbol. In the code table, the encoding order is called the coding standard. That is, each standard defines its code table. Such standards include widespread ASCII, ANSI, Unicode and others. In the ASCII standard, each character takes 8 bits, i.e. = 256 characters or the corresponding binary code from 00000000 to 11111111. Values from 0 to 127 are constant and form the main part of the table, which includes decimal digits, Latin letters (uppercase and lowercase), punctuation marks (dot, comma, brackets, etc.), as well as a space character and various service symbols (tabulation, translation of a line, etc.).The values from 128 to 255 form an additional part of the table, where it is customary to encode symbols of the national alphabets. To circumvent this limitation, the International Organization for Standardization (ISO) created a new character encoding standard, called Latin-1, which contained characters of European languages that were not part of the ASCII set. Microsoft expanded Latin-1 and named this ANSI standard. But ANSI still remained 8-bit encoding. Many languages have thousands of characters, especially languages such as Chinese, Korean and Japanese.

In Kazakhstan, in 2002, the state standard for encoding the letters of the Kazakh alphabet in the 8-bit code table was adopted. Table 1 shows the second part of the 8-bit encoding table, where the letters of the Kazakh alphabet for Windows are located.

Table 1.3.1. 8-bit table of the encoding of the letters of the Kazakh alphabet for Windows.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
8			‚		„	…	†			‰		‹		Қ	Һ	
	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143
9		‘	’	“	”	•	–	—		™		›		қ	һ	
	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159
A		Ұ	ұ	Ә	¤	Ө		§		©	Ғ				®	Ү
	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175
B		±	І	і	ө	µ	¶			№	ғ		ә	Њ	њ	ү
	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191
C	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П
	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207
D	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я
	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223
E	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п
	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239
F	р	с	т	у	ф	х	ц	ч	ш	щ	ъ	ы	ь	э	ю	я
	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255

Here, in each cell, its decimal number and the character's outline are indicated, if it has an 8-bit encoding, and in uppercase and lowercase letters that indicate specific letters of the Kazakh language.

To overcome the limitations of the 8-bit character encoding standard, Microsoft, in cooperation with companies such as Apple Computer, Inc., and

IBM, created a non-commercial Unicode consortium whose goal was to define a new standard for character encoding for international character sets. The work done in Unicode was combined with the work that was conducted in ISO, and as a result, in the early 90s of the XX century, a standard for encoding symbols, called Unicode, was developed.

In Unicode, 31 bits are provided for character encoding (4 bytes, minus one bit). The number of possible combinations gives a prohibitive number: 231 = 2,147,483,644 (that is, more than two billion). Therefore, Unicode describes the alphabets of all known languages, even "dead" and fictitious, includes many mathematical and other special symbols, even archaic symbols, such as the ancient Turkic runes, Sanskrit and Egyptian hieroglyphs, i.e. Unicode allows you to use almost any languages and symbols in the text. In its coding table there are codes of signs used in various branches of science and various decorative signs. However, the information capacity of the 31-bit Unicode still remains too large. Therefore, the abbreviated 16-bit version (216 = 65 536 values) is more often used, where the first 128 codes coincide with the ASCII table.

The first 16-bit version of Unicode (1991) was a 16-bit character set with a fixed character width. In the second version of Unicode (1996) it was decided to significantly expand the code area; To maintain compatibility with those systems where 16-bit Unicode has already been implemented, and UTF (Unicode Transformation Format) -16 was created, which is one of the ways to encode characters from Unicode as a sequence of 16-bit words. This encoding allows you to write Unicode characters. One character of the UTF-16 encoding is represented by a sequence of two bytes. Which of the two goes ahead, the older or the younger, depends on the order of the bytes. To determine the order of bytes, the byte order mark is used.

Now consider an alphabetical approach to measuring information, in which any character sequence is considered a message of information. To determine the amount of such information, the length of its message is counted, without taking into account its value (content).

Definition 1.3.1. The information volume of the message is the number of binary digits that is used to encode this message.

Let M be the number of characters of the initial alphabet in which the message is written, N is the number of characters in the message record. Then the information volume of the message is calculated by the formula:

$$I = N \cdot log_2 M \qquad (1)$$

If $log_2 M$ is not an integer, then it must be rounded up or found $log_2 \widetilde{M}$, where $\widetilde{M}$ is the nearest integer power of 2 and$\widetilde{M} > M$.

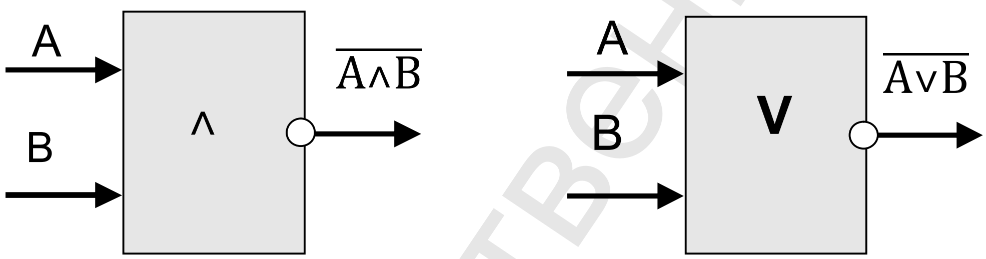
The information volume of the message, expressed in bits, and the minimum number of bits required to write a message in a binary alphabet, are the same.

Using n binary digits, you can encode all the elements of a set of 2^n by a binary code. The information volume of one character of the alphabet denoting the element of the given set is *n*.

Examples 1.3.1.

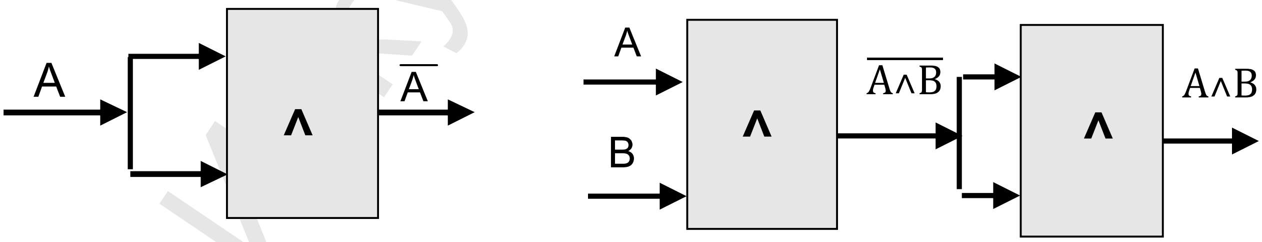
1. With the help of these basic logical elements, **NOT**, **AND**, **OR**, you can build new logical elements:

AND-NOT **OR-NOT**



2. Any Boolean expression can be implemented on elements **NOT-AND** (**NAND**).

NOT: $\bar{A} = \bar{A} \vee \bar{A} = \overline{A\wedge A}$ AND: $A\wedge B = \overline{\overline{A\wedge B}}$



3. Determine the information volume of the word "ASTANA", if we assume that the alphabet of the message consists of 10 letters.

Decision. The length of this message is *N* = 6, the power of its alphabet is *M* = 10. By formula (1), we find $I = 6 \cdot log_2 10$. Since the number 10 is not equal to an integer power of 2, then the value of 〖log〗 _2 10 is rounded up or we find the value $log_2 10$, where $\widetilde{M}$ - the nearest integer power of 2 and $\widetilde{M} > 0$, that $\widetilde{M} = 16$. Then $I = 6\ log_2 16 = 6 \cdot 4 = 24$. Answer: 24.

Questions 1.3.1.

How is the logical information encoded?

What is meant by a logical element?

What are the standards for character encoding?

How is the information volume of the information message calculated?

Tasks 1.3.1.

1.Construct a new scheme for the realization of the expression $X = \bar{A} \& B \vee A \& \bar{B} \& \bar{C}$ from the logical elements of **NAND**.

Help: the implementation of the operation ∨ is performed by a parallel connection with the same inputs, but different outputs.

2. How much information is the message: "ASTANA - THE CAPITAL OF KAZAKHSTAN!".

3. Determine the maximum number of pages containing 80 characters per line and 64 lines per page, which can contain a file of 10 KB in 8-bit encoding.

4. Two texts contain the same number of characters. The first text is made up of 8 character alphabets, and the second one is 16. Determine how many times the amount of information differs in them.

Help.

Denote by x the number of characters in both texts, then the amount of information in the first text is 8x, and in the second text 16x.

Tests 1.3.1.

1. Determine the minimum number of bits for encoding 16 words in a 6-character alphabet, if you use an 8-bit encoding.

A) 768;

B) 800;

C) 1024;

D) 516;

E) 1540.

2. Determine the size of the alphabet, if the volume of the message compiled on it is 1024 characters and occupies 1 kilobyte.

A) 256;

B) 512;

C) 1024;

D) 2,048;

E) 128.

3. Determine how many times the amount of information differs in two texts, if they contain the same number of characters, but the first is written in 16-bit, and the second is in 4-bit encoding.

A) 4;

B) *8*;

C) 24;

D) 32;

E) 2.

I.3.2. Number systems

In general, the number system is a way of writing numbers using numbers and a set of rules. There are several ways to write numbers using numbers.

Any number system must satisfy the following rules:

- the ability to write the values of numbers in a given interval;
- each sequence of digits defines only one numeric value;
- ease of executable of the operations.

All number systems are divided into: positional number systems and non-position number systems.

1. In the positional number system, the values of the digits depend on their place (position) in the number entry. If in a record of a number the same figure occurs several times, then it determines a different value. For example, the number is written in Arabic numerals: in the three-digit number 333, the leftmost digit 3 will be defined by three hundreds, the middle digit 3 - three tens, and the rightmost digit 3 - three units.

2. In the non-position number system, the values of the digits do not depend on their place in the number entry. For example, writing a number using Roman numerals: in the record of the number LXXXVIII (eighty eight), the digit L stands for fifty, X for ten, V for five, and I for the number.

The positional number system is characterized by its base of calculation. The basis determines the number of digits used in this system. For example, the number of digits in the decimal system is ten, in the octal system - eight, and in the binary system - two, etc.

The capabilities of all positioning systems are the same. However, among them the decimal number system is the most common. That's the reason - in both hands there are ten fingers, which makes it easy and convenient to count to ten. Counting to ten and exhausting all the possibilities of the "computing tool", it is reasonable for the next position (second digit) to take the number 10 (ten) as a new unit. Further, the decimal number ten will be the unit of the next position (the third digit) and so, continuing the calculations, the decimal number system appeared.

However, the decimal number system did not immediately occupy a priority place in the calculations. In different historical periods, many peoples used non-decimal systems of scraping. For example, in ancient Turks the basis of the number system is seven (1 week = 7 days, 1 girth = 7 span), in ancient Babylonians - sixty (1 minute = 60 seconds, 1 hour = 60 minutes, 1 degree = 60 minutes), Englishmen - twelve (1 year = 12 months, 1 foot = 12 inches, 1 shilling = 12 pence).

In any positional number system with a base q, a given number ***A*** can be represented as follows:

$$A_{(q)} = a_{n-1}q^{n-1} + \cdots + a_1 q^1 + a_0 q^0 + a_{-1}q^{-1} + \cdots + a_{-m}q^{-m} \quad (1)$$

Where ai is the number of digits used in the number system, n is the number of digits in the integer part, m is the number of digits in the fractional part (i = -*m*, ..., -1, 0, 1, ..., *n*-1).

Among these number systems we need a decimal system of scaling and a binary number system. Table I.3.2 gives a binary equivalent for each decimal digit.

Table I.3.2. Binary equivalent of decimal digits.

Decimal digit	**Binary digit**
0	0
1	1
2	10
3	11
4	100
5	101
6	110
7	111
8	1000
9	1001

From this table, you can notice that when you write the same number in a different number system, you will need a different number of characters (digits). For example, the two-digit number 16 in the decimal number system in the binary system will be 10 000, i.e. will require five characters.

To translate a given integer with base *p* to a base *q*, it is necessary to divide this number several times by *q* until the remainder becomes less than *q*. The resulting quotient takes the number with the base *q* as the most significant digit, and as the values of the remaining digits, we need to take the remainders in the direction starting from the last remainder to the first remainder and form the chain from left to right.

To translate a given correct fractional number with a base p to a base q, multiply this number by q several times until the value of the fractional part bit is zero or until the specified precision is obtained. As the value of the bits of a regular fraction with a new basis q, it is necessary to form a chain from left to right in the direction starting from the first whole that appears to the last whole that appears.

When translating mixed numbers, it is necessary to translate separately the whole and fractional parts into the new system according to the rules for translating integers and proper fractions, and then combine the two results into one mixed number in the new number system.

The conversion of binary, octal and hexadecimal numbers to decimal notation is done according to the rule:

To translate the number of the P system to decimal, the following decomposition formula should be used:

$$a_n a_{n-1} \dots a_1 a_0 = a_n P^n + a_{n-1} P^{n-1} + \dots + a_1 P^1 a_0 P^0$$

The translate of octal and hexadecimal numbers to the binary number system and back is performed according to the rule:

1. To translate a number from octal to binary, you need to write each digit of that number with a three-digit binary number (triad). In this case, insignificant zeros on the left for integers and on the right for fractions are not written. To reverse translate the translation of a binary number into an octal number system, the original number must be divided into the triads to the left and right of the comma and each group represented in the octal number system. The extreme incomplete triads are complemented by zeros.

2. To convert a number from hexadecimal to binary, you must write each digit of this number with a four-digit binary number (tetrad). Note: insignificant zeros on the left for integers and on the right for fractions are not written. To reverse translate the binary number in hexadecimal notation, it is necessary to divide the original number into tetrads to the left and right of the comma and present each group with a digit in hexadecimal notationThe extreme incomplete tetrads complete with zeros.

Notes I.3.2:

1) The number that is the basis of the smallest number system is equal to two, it is called the binary number system. The number in binary notation consists only of digits 0 and 1.

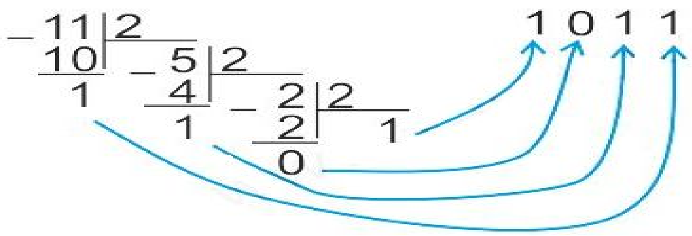
2) The most common number system is the decimal number system, which consists of ten Arabic numerals: 0,1, 2, 3, 4, 5, 6, 7, 8, 9.

3) Rules for performing operations in the binary system and in the decimal notation are similar.

4) Properties of operations on binary numbers are identical with the properties of operations on decimal numbers.

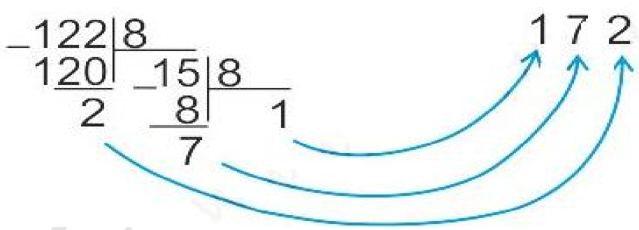
Examples I.3.2:

1. Translate the number $11_{(10)}$ into a binary number system.



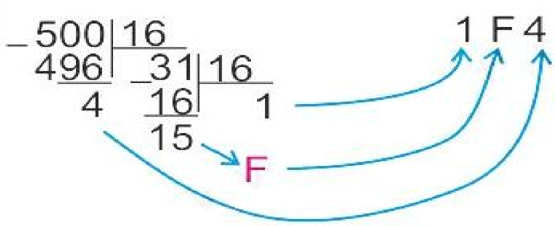
Answer: 11 (10) = 1011 (2).

2. Translate the number $122_{(10)}$ into the octal number system



Answer: $122_{(10)} = 172_{(8)}$.

3. Translate the number $500_{(10)}$ to the hexadecimal number system.



Answer: $500_{(10)} = 1F4_{(16)}$.

4. The conversion of the fractional number of 0.625 in the decimal number system to the binary number system will look like this:

0, 625

	* 2
1,	250
	* 2
0,	500
	* 2
1,	000
	* 2
0,	000

Now, collecting from top to bottom, the obtained units of transition to the highest digits from the results of the execution of the chain of operations of multiplying the fractional part of a given number by the base of the translated number system (in our case 2), we get 1010, which means.

5. The four-digit decimal number *1952* is expressed as follows:

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

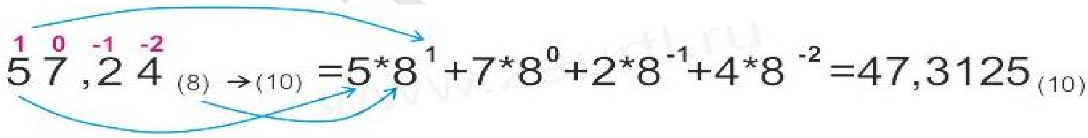
6. *A* decimal number with a three-digit integer part and a three-digit fractional part $596.174_{(10)}$ is expressed as follows:

$$596174_{(10)} = 5*10^2 + 9*10^1 + 6*10^0 + 1*10^{-1} + 7*10^{-2} + 4*10^{-3}\,7.$$

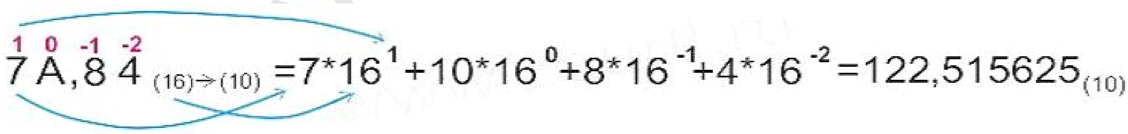
7. The number of a binary system with a four-digit integer part and a three-digit fractional part $1010.101_{(2)}$ is expressed as follows:

$1010101_{(2)} = 1*2^3 + 0*2^2 + 1*2^1 + 0*2^0 + 1*2^{-1} + 0*2^{-2} + 1*2^{-3}$ 8. Translate the number 57,24 (8) into the decimal number system.

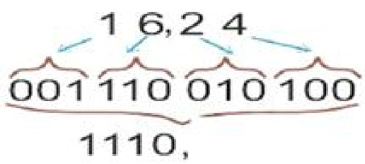
8. Translate the number $57{,}24_{(8)}$ into the decimal number system.



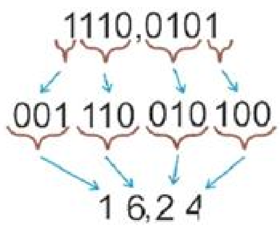
9. Translate the number 7A, $84_{(16)}$ into the decimal number system.



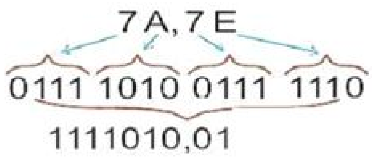
10. Write the number $16.24_{(8)}$ in the binary system.



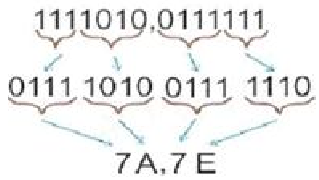
11. Write the number $1110.0101_{(2)}$ in the octal number system.



12. Record the number 7A, 7E$_{(16)}$ in the binary system.



13. Write the number 1111010,0111111 (2) in the hexadecimal system.



Tasks I.3.2.

1. Translate the specified number of 10000001 binary system in decimal.
2. Translate the specified number 129 of the decimal system to the octal number system.
3. Translate the specified number 7A0 of the hexadecimal system to the decimal number system.

Questions I.3.2.

1. What is the number system?
2. Which groups are divided into all the number systems?
3. Is it possible to represent the same numerical value in a different number system?

Tests I.3.2.

1. What is the relationship between numbers in the positional number system?

A) The value of the digits depends on their place in the record
B) The meaning of the letters depends on their place in the record;
C) The value depends on their form of writing the number;
D) The place of the digits depends on their meaning;
E). The meaning of the numbers depends on their recording;

2. What is the dependency in the non-positional number system?
A) The value of the digits does not depend on their place in the record;
B) The meaning of the letters does not depend on their place in the record;
C) The value depends on their form of writing the number;
D) Does not depend on the recording of a given number;
E) The value of the digits depends on their place in the recording.

3. What is the basis of the positional binary number system?
A) 0 and 1;
B) 2; 0
C) 1 and 2;
D) 10;
E) 0.

II. SETS AND RELATIONS

II.1. Types of sets and operations on them

II.1.1. The concept of a set and operations on them

The set is one of the primary concepts of mathematics, that is, those that are at the heart of the logical system and are no longer defined through other concepts. The set was developed at the end of the 19th century. Set theory is now the main part of mathematics, and can be used as a basis from which almost all mathematics can be obtained.

When we talk about a set, we understand that a set is a set or set of elements united by some common attribute (property). Elements can be real physical or abstract mathematical objects. On the basis of this, we can give the following intuitive definition of the concept of a set:

A set is the union of individual (discrete) elements selected by some criterion (criterion, type).

The names of sets are denoted by uppercase Latin letters, and their elements by lowercase (small) Latin letters or Arabic numerals. In both cases, you can use indexes, including multi-level ones.

The notation a∈A (a∉A) means that the element a belongs (does not belong) to the set A. For example, let $A = \{1, 2, 3\}$: if a = 2, then we can write a∈A, and if = 5, then - a∉A.

Sets can be described in various ways: *by enumerating elements*, *by notifying the developer of a set*, *by interval recording*, *by plotting a graph on a number line and / or Venn diagrams*.

By enumeration of elements: Enumeration of elements is a list of elements in the set, separated by commas and surrounded by curly braces. For example:

1) {2, 3, 4, 5, 6} is a list for the set of numbers from 2 to 6 inclusive;

2) {1, 2, 3, 4, ...} is a list for the set of natural numbers, where three points indicate that their number continues in the same scheme by an indefinite number;

3) a set of small Latin letters denoting the vowels of the English language is described as $\{a, e, i, o, u\}$.

According to the developer's notation of the set: The notation of the builder of a set is a mathematical abbreviation for the exact declaration of all elements of a certain set that have a specific property. In addition, you can use a colon (:) to represent the words "so, that". For example:

1) The statement "all x, which are elements of the set of integers, such that x is between 2 and 6 inclusive." $\{x \in Z: 2 \leq x \leq 6\}$ is a set of numbers from 2 to 6 inclusive, Z is the set of integers;

2) The statement "all n, which are elements of the set of natural numbers, such that n is less than the number 100". $\{n \in N: n < 100\}$ is a set of natural numbers smaller than the number 100, N is the set of integers;

3) The statement "all x, which are elements of the set of integers, such that their values are greater than 0, positive. $\{x \in Z \mid x > 0\}$ is the set of all positive integers.

On the recording interval: the interval is a connected subset of numbers. Interval designation is an alternative expression for the answer in the form of an inequality. Unless otherwise specified, we will work with real numbers.

Among all sets there are two special sets:

1. *Ø* is an empty set that does not contain any elements.

2. ***U*** is a universal set (universe) containing all elements of the type (subject domain) under consideration.

Concerning the theory of the universe, this set contains all the objects considered in this theory as elements.

For example, the universe is:

1) in the theory of numbers, the set of all integers;

2) in the theory of languages, the set of all words in a given alphabet;

3) in geometry, the set of all points of an n-dimensional geometric space.

The cardinality of ***A*** is equal to the number of its elements and is denoted by |***A***|. Now you can define operations on sets:

Suppose that two sets A and B are given, then the following operations can be defined over them:

1. The union of the sets A and B consists of the elements A or B:

$$A \cup B = \{x: x \in A \vee x \in B\}.$$

2. The intersection of the sets A, B consists of the elements A and B:

$$A \cap B = \{x: x \in A \ \& \ x \in B\}.$$

3. The complement to the set A consists of the elements of the universe U and does not include elements of A:

$$\overline{A} = \{ x \mid \quad x \in U \ \& \ x \notin A \}.$$

4. The difference between the sets A and B consists of the elements of the set A and does not include the elements B:

$$A \setminus B = \{x: \ x \in A \ \& \ x \notin B\}.$$

5. The symmetric difference of sets A and B consists only of elements of A or only of elements of B:

$$A \triangle B = \{x: \ (x \in A \ \& \ x \notin B) \lor (x \in B \ \& \ x \notin A)\}.$$

6. The Cartesian (direct) product of sets A and B consists of all possible ordered pairs of elements A and B:

$$A \times B = \{(a, b): \ a \in A \ \& \ b \in B\}.$$

Operations (1) - (3) can be represented using the Euler-Venn diagram (Figure I.2.1), in which the universe U is represented by a rectangle, and the sets A and B are circles. Hatching is used to highlight the result.

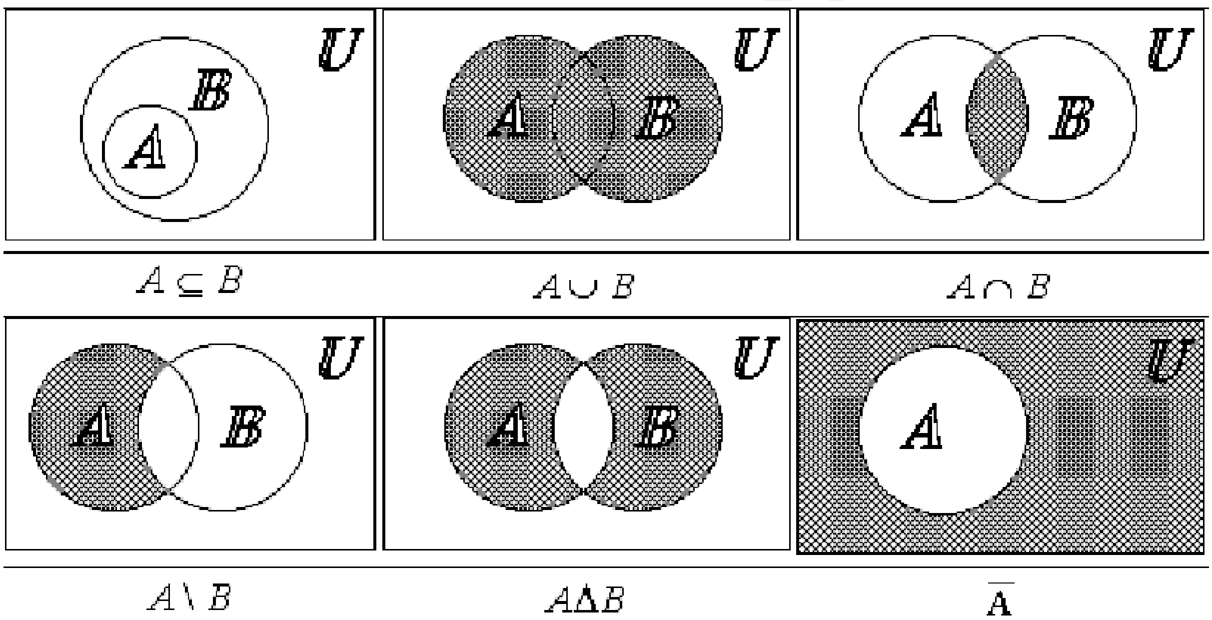


Figure I.2.1. Euler-Wenn diargam.

It is shown here that the sets A and B are subsets of U, and they are written as $A \subseteq U$ and $B \subseteq U$ (see I.2.2.).

The operations (1) - (3) can be defined not only over two sets, but also over n sets $A_1, A_2, \dots, A_n$ where $n \in N$ & $n > 2$.

The union over the sets $A_1, A_2, \dots, A_n$ is defined as:

$$A_1 \cup A_2 \cup \dots \cup A_n = \bigcup_{i=1}^{n} A_i \,.$$

The intersection over the sets $A_1, A_2, \dots, A_n$ is defined as:

$$A_1 \cap A_2 \cap \dots \cap A_n = \bigcap_{i=1}^{n} A_i \ .$$

The direct product over the sets $A_1, A_2, \dots, A_n$ is defined as the set of tuples of the form $(a_1, a_2, \dots, a_n)$, $a_1 \in A_1, a_2 \in A_2, \dots, a_n \in A_n$, i.e.

$$A_1 \times A_2 \times \dots \times A_n = \{(a_1, a_2, \dots, a_n): a_1 \in A_1, a_2 \in A_2, \dots, a_n \in A_n)\}.$$

If If $A_1 = A_2 = \cdots = A_n = A$, then $\underbrace{A \times A \times \dots \times A}_{n} = A^n$ is a power.

Now we can show the tabular method of specifying sets and operations on them. Let U, $A \subseteq U$ and $x \in U$ be given.

An indicator (characteristic) function for a set A is a function $I_A(x)$, define U, $A \subseteq U$ и $x \in U$ as:

$$I_A(x) = \begin{cases} 1, & if \quad x \in A \\ 0, & if \quad x \notin A \end{cases}$$

In this way, $I_A\colon\ U \to \{0,1\}$.

For $A \subseteq U$ and $B \subseteq U$, the following properties hold:

$$I_A(x) = I_B(x) \Leftrightarrow A = B;$$

$$I_A(x) \le I_B(x) \Leftrightarrow A \subseteq B;$$

$$I_{\overline{A}}(x) = 1 - I_A(x);$$

$$I_{A \cup B}(x) = I_A(x) + I_B(x) - I_A(x) \cdot I_B(x);$$

$$I_{A \cap B}(x) = I_A(x) \cdot I_B(x);$$

$$I_{A \setminus B}(x) = I_A(x) - I_A(x) \cdot I_B(x);$$

$$I_{A \Delta B}(x) = I_A(x) + I_B(x) - 2 I_A(x) \cdot I_B(x).$$

Indicators can be conveniently set using Table II.1.1.

Table II.1.1. Indicators.

$x \in A$	$x \in B$	$x \in A \cup B$	$x \in A \cap B$	$x \in A \setminus B$	$x \notin A$	$x \in A \Delta B$
0	0	0	0	0	1	0
0	1	1	0	0	1	1
1	0	1	0	1	0	1
1	1	1	1	0	0	0

Operations on sets have the following properties:

I. Unification*, *intersection and difference:

1) $A \cup \varnothing = A$ – the property of zero;

2) $A \cup A = A$ – idempotency;

3) $A \cup B = B$, if all elements of A are contained in B;

4) $A \cup B = B \cup A$ is commutativity;

5) $(A \cup B) \cup C = A \cup (B \cup C) = A \cup B \cup C$ – associativity;

6) $A \cap \varnothing = A$ – the property of zero;

7) $A \cap A = A$ – idempotency;

8) $A \cap B = A$, if all elements of A are contained in B;

9) $A \cap B = B \cap A$ is commutativity;

10) $(A \cap B) \cap C = A \cap (B \cap C) = A \cap B \cap C$ – associativity;

11) $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$ is distributive;

12) $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$ - distributivity;

13) $A \cap (B \setminus C) = (A \cap B) \setminus (A \cap C)$ is distributive;

14) $A \cup \overline{A} = U$ – the complement property;

15) $A \cap \overline{A} = \varnothing$ – the complement property;

16) $\overline{A \cup B} = \overline{A} \cap \overline{B}$ - the law of de Morgan;

17) $\overline{A \cap B} = \overline{A} \cup \overline{B}$ is de Morgan's law;

18) $\overline{\overline{\mathbf{A}}} = \mathbf{A}$ is involutive;

19) $A \setminus \varnothing = A$ the property of the difference;

20) $A \setminus A = \varnothing$ – the property of the difference;

21) $A \setminus B = A \cap \overline{A} = \varnothing$ the property of the difference;

22) $B \setminus A = B \cap \overline{A} = B \setminus (B \cap A)$ is the property of the difference;

II. **Symmetric difference and direct product:**

1) $A \triangle \varnothing = A$ – the property of zero;

2) $A \triangle A = \varnothing$ – idempotency;

3) $A \triangle B = (A \cup B) \setminus (A \cap B)$ – the property of the symmetric difference;

4) $A \triangle B = B \triangle A$ is commutativity;

5) $(A \triangle B) \triangle C = A \triangle (B \triangle C) = A \triangle B \triangle C$ – associativity;

6) $(A \cup B) \times C = (A \times C) \cup (B \times C)$ is distributive;
7) $A \times (B \cup C) = (A \times B) \cup (A \times C)$ is distributive;
8) $(A \cap B) \times C = (A \times C) \cap (B \times C)$ is distributive;
9) $A \times (B \cap C) = (A \times B) \cap (A \times C)$ is distributive;
10) $(A \backslash B) \times C = (A \times C) \setminus (B \times C)$ – is distributive;
11) $A \times (B \backslash C) = (A \times B) \setminus (A \times C)$ – is distributive;

Examples II.1.1.
1. All non-negative integers form a set of natural numbers.
2. $D = \{0, 1\}$, where only the listed constants 0, 1 are elements of the set D.
3. $X = \{x: x > 0\}$, where only positive variables of x are elements of X.
4. $| \varnothing | = 0$.
5. If $A = \{a, b, c, d, e\}$, then $| A | = 5$.
6. Let $A = \{a, b, c, d, e, f\}$, $\mathrm{B} = \{c, d\}$, then:
   • $B \times B = \{(c, c), (c, d), (d, c), (d, d)\}$;
   • $A \setminus B = \{a, b, e, f\}$;
   • $A \triangle B = \{a, b, e, f\}$;
   • *depends* on what will be the universe U. Suppose that
   if $U = \{a, b, c, d, e, f, h\}$, then = $\{h\}$.

1.7. Let $A = \{1, 2\}$, $B = \{a, b\}$, $\mathrm{C} = \{+, -\}$. Then, using the distributivity $(A \cup B) \cup C = A \cup (B \cup C) = A \cup B \cup C$, one can obtain $(\{1,2\} \cup \{a,b\}) \cup \{+,-\} = \{1,2\} \cup (\{a,b\}) \cup \{+,-\}) = \{1,2\} \cup \{a,b\} \cup \{+,-\}$.

Tasks II.1.1. Let $A = \{1, 2, 4\}$, $B = \{3, 4, 5, 6\}$. Then perform the following operations:
1) A∪t;
2) A∪A;
3) A∩B;
4) A × B;
5) A \ B;
6) A∪;
7) A △ B;
8) (A∪B) × C;
9) (A \ B) × C.

Questions II.1.1:

1. How are sets given?
2. What is a universal set?
3. How are subsets determined?
4. What is the Euler-Venn diagram?
5. How is the direct product of sets determined?
6. How is the difference of sets determined?
7. How is the symmetric difference of sets determined?
8. How is the indicator function of a set defined?
9. Find out which of the following distributive laws are valid for any sets A, B, C:

1) $A \setminus (B \cup C) = (A \setminus B) \cup (A \setminus C)$;
2) $A \setminus (B \cap C) = (A \setminus B) \cap (A \setminus C)$;
3) $A \triangle (B \cup C) = (A \triangle B) \cup (A \triangle C)$;
4) $A \triangle (B \cup A) = (A \triangle B) \cup (A \triangle C)$;
5) $A \setminus (B \triangle A) = (A \setminus B) \triangle (A \setminus C)$;
6) $A \cup BC = (A \cup B)(A \cup C)$;
7) $A \cup (B \setminus C) = (A \cup B) \setminus (A \cup C)$;
8) $A (B \setminus C) = AB \setminus AC$;
9) $A \cup (B \triangle C) = (A \cup B) \triangle (A \cup C)$;
10) $A (B \triangle C) = AB \triangle AC$;
11) $A \triangle (B \setminus C) = (A \triangle B) \setminus (A \triangle C)$?

Tests II.1.1:

1. If D = {d | D is an integer and $0 \le d \le 9$, then what is $|\boldsymbol{D}|$?

A) 10
B) 9
C) 10
D) 0
E) 1

2. What result will be obtained after performing the operation $\boldsymbol{A} \cup \boldsymbol{B}$ for given sets $\boldsymbol{A} = \{a, b, c\}$ and $B = \{b, d, e, f\}$?

A) $\{a, b, c, d, e, f\}$
B) $\{a, b, c\}$

C) $\{b, d, e, f\}$
D) Ø
E) $\{c, b, d, e\}$

3. What result will be obtained after the operation A \ B for the given sets A = {a, b, c, r, q, e, x} and B = {z, e, x}?
A) {a, b, c, d}
B) {a, b, c, d, d, e}
C) {r, e, x}
D) {r, e}
E) Ø

II.1.2. Relations and ways of representing them

The concept of "Attitude" plays a very important role in many branches of science, especially in computer science, as it is used in the construction of any algorithm with which to solve a particular problem, for organizing branching or repetition.

Definition II.1.2.1. If two sets A and B are given the same type, then the following relations can be made:

1) $A = B$: A is equal to B if A and B consist of the same elements, i.e. A and B are subsets of one another;

2) $A \subseteq B$: A is contained in B if all elements of A belong to B or A is equal to B, this means that A is a subset of B;

3) $A \subset B$: A is strictly contained in B if all elements of A belong to B and A is not equal to B, that is, some elements of B do not belong to A, this means that A is a proper subset of B.

Similarly, a relationship can be defined including $A \supseteq B$ and strictly includes $A \supset B$.

It is not difficult to see that the above relations =, ⊆ and ⊂ are subsets of the direct product $A \times B$, that is, We can assume that any relation is a subset in the direct product, allocated by a certain law.

Note that the empty set t is a proper subset of any finite set.

Examples II.1.2.1:

1) if $A = \{a, b, c\}$, $B = \{b, a, c\}$, then $A = B$;

2) if $A = \{1,2,3,4\}$, $B = \{3,1,4,2\}$, then $A \subseteq B$;

3) if $A = \{1,2,3\}$, $B = \{3,1,4,2\}$, then $A \subset B$

Definition II.1.2.2. Let $A_1, A_2, \dots, A_n$ be arbitrary sets, not necessarily distinct. Then the n-ary relation on the sets $A_1, A_2, \dots, A_n$ is a subset

$$R^n = A_1, A_2, \dots, A_n,$$

where $n \geq 1$, R^1 is a unary relation to A_1, R^2 is a binary relation on A_1, A_2, R^3 is a ternary relation to A_1, A_2, A_3, etc.

Every unary relation on the set A is a characteristic property of some subset of it. The set of all unary relations on A coincides with the set of all subsets of A.

Examples II.1.2.2:

1. ***A*** unary relation $R_1^1 = \{n:\ n \in N \,\&\, n < 100\}$ defines a set of natural numbers less than the number 100;

2. ***A*** unary relation $R_2^1 = \{n:\ \forall k \in N(n = 2*k)\}$ defines a set of even natural numbers;

3. ***A*** unary relation $R_2^1 = \{n:\ \forall k \in N(n = 2*k)\}$ defines a set of odd integers.

Definitions II.1.2.3. A binary relation is defined over pairs of sets and can be represented in one of three ways:

1) *prefix record* - the sign of the relation is inserted before the participants of the binary relation;

2) *infix record* - the sign of the relation is inserted between the participants of the binary relation;

3) *postfix record* - the sign of the relation is inserted after the participants of the binary relation.

Binary relations over pairs of elements are often represented using tables: the rows correspond to the first elements of the pair, the columns correspond to the second elements of the pair, and the relationship between the specific elements of the row and column is marked with a special sign, for example, with a "1" or other symbol.

Examples II.1.2.3:

1. If $a \in A$ and $b \in B$ are in the binary relation R, then this can be written as:

Rab - prefix entry;

aRb - infix entry;

abR - postfix recording.

2. If the sets $A = \{a_1, \ldots, a_r\}$ and $B = \{\{b_1, \ldots, b_s\}$ are in the binary relation R, then it can be represented using table I.2.2, in which the elements ai are represented by chains, the elements b_j – columns, and the ratio a_iRb_j is marked "1":

Table II.1.2 shows the binary (binary) relation defined between two given sets $A = \{a_1, \ldots, a_r\}$ and $B = \{b_1, \ldots, b_s\}$.

Table II.1.2. Binary relation.

R	b_1	b_2	…	b_{s-1}	b_s
a_1	1		…	1	
a_2		1	…		
…	…	…	…	…	…
a_{r-1}	1		…	1	
a_r			…		1

Definitions II.1.2.4. It is said that the binary relation R on the set S:

1) is reflexive if sRs holds for each $s \in S$;
2) is transitive if for any s, t, $u \in S$ from sRt and tRu follows sRu;
3) is symmetric if for any s, $t \in S$ from sRt follows tRs;
4) is antisymmetric if from aRb and bRa follows $a = b$.

Examples II.1.2.4:

1. The ratio of numbers over numbers is not reflexive;
2. Relations =, ≥,> over numbers are transitive;
3. The ratio = over the numbers is symmetric.

Definition II.1.2.5. A binary relation R is called an equivalence relation if it satisfies the properties of reflexivity, transitivity, and symmetry.

To every equivalence relation on the set S there corresponds a unique partition of the given set into adjacent classes.

Examples II.1.2.5. The ratio = in any numeric set is an equivalence relation, i.e. For any k, m, $n \in \boldsymbol{N}$:

1. Reflectivity: $n = n$.
2. Transitivity: from $k < m$ and $m < n$ follows $k < n$.

3. Symmetry: from $k = m$ follows $m = k$.

Definitions II.1.2.6. *A* binary relation *R* on some set *S* that satisfies the properties of reflexivity, *transitivity*, and *antisymmetry* is called *a partial order relation*.

Examples II.1.2.6. Partial relations are:
1. The relation ⊆ for subsets of a set;
2. The relation ⊇ for subsets of a set;
3. Relation = on the set of integers;
4. The relation ≤ on the set of natural numbers;
5. The relation ≥ on the set of integers.

Definitions II.1.2.7:

1. *A* partially ordered set is a set *A* with a partial order relation defined on it. More precisely, a partially ordered set is a pair <*A*, *R*>, where *A* is a set, and *R* is a partial order relation on *A*.

2. Elements *a* and *b* of a partially ordered set *A* are said to be congruent with respect to the partial order *R* on this set if *aRb* or *bRa* holds.

3. *A* partial order on a set *A* is called a linear order if any two elements *a* and *b* of *A* are congruent with respect to the partial order *R*.

4. *A* linearly ordered set or chain is a partially ordered set in which the elements of each pair are comparable.

5. The elements *a* and *b* of a partially ordered set *A* are called incomparable if no partial order relation is fulfilled between them. The possibility of the existence of incomparable elements explains the meaning of the term "partially ordered set".

Examples 1.2.2.7:

1. The set of natural numbers *N* with the relation "≤" is partially ordered, all natural numbers will be comparable with respect to the relation "≤", and *N* is a chain.

2. The set of all real numbers with the relation "=" is a linearly ordered set if all real numbers are comparable with respect to the relation "=".

3. Let A be the set of real-valued functions on the closed interval [0,1] with the partial order relation <, =,> defined on it, then the elements $f(x) = x$ and $g(x) = 1-x$ will be incomparable.

Definitions II.1.2.8. Let A be a partially ordered set, B its subset, i.e. $A \supseteq B$. Then:

1) the lower bound (the supremum) of a set B in a set A is an element $a \in A$, such that $a \leq b$ ($b \leq a$) for any $b \in B$;

2) the element $a \in A$ is called the smallest (largest) in the set A if a is the lower (upper) face of A itself;

3) An element $a \in A$ is called minimal (maximal) in the set A, if there is no $b \in A$ such that $b < a$ ($a < b$).

The smallest (largest) element of A is its unique minimal (maximal) element.

Examples II.1.2.8:

1. The set of all subsets of A has the smallest element t and the largest element of A itself.

2. The set N of natural numbers has the smallest element 1 and does not have the largest element.

3. The set Z of all integers does not have the smallest, largest, minimal, and maximal elements.

The elements of sets A and B are in one-to-one correspondence if each element $a \in A$ by some law is associated with the same element $b \in B$, and each $b \in B$ is mapped to the same $a \in A$.

The sets A and B are equivalent (equipotent) if one can establish a one-to-one correspondence between their elements.

Note II.1.2.2. A binary relation can be given by a triple of sets $<R, A, B>$, where $R \subseteq A \times B$ is a graph of the relation and written $(a, b) \in R$ or aRb. Then you can define:

Domain of definition: Dom $R = \{x \in A\colon\ \exists y \in B (x, y) \in R\}$;

Value range: Run $R = \{y \in B\colon\ \exists x \in A (x, y) \in R\}$;

The inverse relation:

$R^{-1} = \{(y,x): B \times A: (x,y) \in R\}$;

The composition of the ratio: $R \subseteq A \times B$, $S \subseteq B \times C$,

$R \cdot S = \{(x, z) \in A \times C: \exists y \in B\ [(xRy)\ \&(ySz)]\}$.

Definition II.1.2.9. A binary relation $f \subseteq X \times Y$ is called a function from X to Y if Dom $R = X$ and $(x, y) \in f$, $(x, z) \in f \Rightarrow y = z$.

The function $f: X \rightarrow Y$ is called:

1) surjective if for any $y \in Y$ there exists $x \in X$ such that $y = f(x)$, i.e. $\forall y \in Y$ $\exists x \in X (y = f(x))$;

2) injective, for any x_1, $x_2 \in X$, from the fact that x $x_1 \neq x_2$ it follows that $f(x_1) \neq f(x_2)$, i.e.

$\forall x_1 \in X\ \forall x_2 \in X\ (x_1 \neq x_2 \Rightarrow f(x_1) \neq f(x_2))$;

3) *bijective* if it is surjective and injective.

Any binary function can be associated with a ternary relation, for example, if a binary function $f(x,y)$ is given, then it can be associated with the ternary ratio $R^3(x,y,z)$ so that $z = f(x,y)$.

Examples II.1.2.9. Let x, y, $z \in N$ be positive integers and let $f(x,y) = z$ be a binary function, then:

1) if x is 3, y is 5, z is 8, and f is an operation of addition +, then instead of writing $f(x,y) = z$, we can write 3 + 5 = 8 and associate the ternary relation Add (3, 5, 8), for which 8 = 3 + 5 holds.

2) if x is 8, y is 2, z is 4, and f is the division operation ":", then instead of writing $f(x,y) = z$, we can write 8:2 = 4 and associate the ternary relation *Dev*(8, 2, 4), for which 4 = 8: 2 is valid.

Clearly, for some x, $y \in N$, the result of the division operation is not an integer, and for them the ternary relation *Dev*(x, y, z) does not hold. Therefore, for such cases it is necessary to determine additional conditions for the results.

Definition II.1.2.10. *A* transitive closure of a relation R on a set A is the intersection of all transitive relations containing R as a subset (otherwise, the minimal transitive relation containing R as a subset).

A transitive closure exists for any relation. For this we note that the intersection of any set of transitive relations is transitive. Moreover, there necessarily exists a transitive relation containing R as a subset.

Transitive closure has the following properties:

1) The transitive closure of a reflexive relation is reflexive, because The transitive relation contains the original ratio;

2) The transitive closure of a symmetric ratio is symmetric. Indeed, suppose there is a transitive relation aRb, then there exist $x_1, x_2, ..., x_n$ such that $aRx_1, x_1Rx_2, \ldots, x_nRb$. But from the symmetry of the relation R, it follows that $bRx_n, x_nRx_{n-1}, \ldots, x_1Ra$, hence bRa.

3) Transitive closure does not preserve *antisymmetry*, for example, for the ratio $\{(a,b), (b,c), (c,a)\}$ on the set $\{a, b, c\}$.

4) The transitive closure of a transitive relation is itself.

The relation $R^*=R^+\cup R^0$, where $R^0=\{(\varepsilon, \varepsilon): \varepsilon\in A\}$ is sometimes called a reflexive-transitive closure, although often by "transitive closure" is meant R *. Usually the differences between these relations are not significant.

Examples II.1.2.10:

1. For any a, b, $c\in N$, the relation $a<b$ and $b<c$ implies that $a<c$.
2. For any x, $y\in N$, the relation $y = x$ follows from the validity of the relation $x = y$.
3. If A is a set of cities and the relation xRy is given, meaning "there is a bus route from x to y", then the transitive closure of this relation is the relation "there is an opportunity to get by bus from x to y".

Tasks II.1.2.

1. Let $A=\{a,b,c,d\}$, $B=\{b,d\}$, $C=\{c\}$. Determine:

$A \triangle B$;

$A \times (B\cup C)$;

$A \times (B\cap C)$;

$A\cup (B\cap C)$.

$A \setminus (B\cup C)$;

$(A \setminus B) \cap C$.

$(A \triangle B) \cap C$;

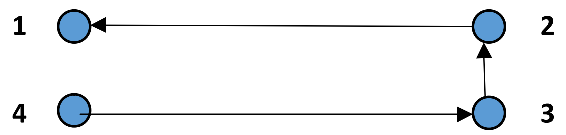
2. Write out the ordered pairs belonging to the following binary relations on the sets $A = \{1,3,5,7\}$ and $B = \{2, 4, 6\}$:

(a) $U = \{\{x, y): x + y = 9\}$;

(б) $V = \{(x, y): x< \ y\}$.

3. Find all ordered pairs belonging to a relation that is defined on the set $A = \{1, 2, 3, 4, 5, 6\}$ by the following set $R = \{(x, y): x$ is a divisor of $y\}$.

3. List the ordered pairs belonging to R if the ratio R is represented as in the lower figure.



4. Determine which of the following relationships on the set of people are reflexive, symmetric or transitive:

(A) "... has the same parents as ...";

(B) "... is a brother ...";

(C) "... older or younger than ...";

(D) "... not higher than ...".

(D) "... older than his son ...".

Questions II.1.2:

1. What is the partial ordering of the set?
2. How is the function determined?
3. How is the composition of the relationship determined?
4. What is the power of the union of two sets?
5. What sets are equal?
6. What elements are incomparable?
7. What is the equivalence of two sets?
8. What is the transitive closure of a relationship?

Tests II.1.2:

1. How many sets participate in a binary relation?

A) 2

B) 3

C) 4
D) 0
E) 1

2. How does the relationship and direct work relate to each other?
A) A relation is a subset of a direct product.
B) The relation includes the direct product.
C) The relation and the direct product intersect.
D) ***A*** relation is an element of a direct product.
E) The relation describes the direct product.

3. What binary relation is an equivalence relation?
A) If it is reflexive, transitive, and symmetric;
B) If it is reflexive and symmetric;
C) If transitive and symmetric;
D) If it is reflexive and transitive;
E) If it is reflexive, nontransitive, and not symmetric.

II.1.3. Numerical sets and intervals

If only numeric values are elements of a set, then such sets are called numerical sets. Depending on the type of values of the elements, we will distinguish between numerical sets: the set of natural numbers, the set of integers, the set of primes, the set of rational numbers, the set of irrational numbers, the set of real (real) numbers:

1.*Natural numbers* are the very first numbers that people began to use. Natural numbers can be used to count items and use as their numbers. The set of natural numbers is bounded from below and is not bounded from above. The smallest natural number is 1-unit. The set of natural numbers is denoted by the letter N;

$N = \{1, 2, 3, ...\}$.

2. Integers include natural numbers. The set of integers is denoted by the letter Z:

$Z = \{..., -3, -2, -1, 0, 1, 2, 3, ...\} = \{0, \pm 1, \pm 2, \pm 3, ...\}$.

The set of integers consists of three subsets of negative numbers Z^-, zero 0 and a subset of positive integers Z^+, which coincides with the set of natural numbers, i.e. $Z^+ = N$.

3. *Simple numbers*. The set of prime numbers is denoted by the letter $\boldsymbol{P}$. A prime number is an integer that is divisible only by itself and by one. Examples of prime numbers are 3, 5, 7, 11, 13, 17. Simple numbers are widely used in cryptography.

5.R*ational numbers*. The set of rational numbers is denoted by the letter Q. A rational number is a fractional number, which is represented as p / q (where p is an integer and q is a natural number). For example, 1/3 is "one part of three", 0.25 is twenty-five hundredths. Decimal fractions can also be written as p / q. For example, 0.25 = 25/100 = 1/4. ***A*** rational number can have several different fractional representations. For example, 1/2 is equivalent to 2/4 or 132/264. In the decimal representation, rational numbers take the form of finite or infinite periodic fractions. Integers (positive and negative) can also be written in the form p / q, i.e. In the form of a fraction with denominator 1, for example:

2 = 2/1, 0 = 0/1, -5 = -5/1.

Thus, the set of rational numbers includes the set of integers.

6. Irrational numbers. The set of irrational numbers is denoted by the letter *I*. Any real number that is not rational is irrational. These numbers can be written in decimal form, but not in fractions. They are infinite non-periodic decimal fractions. Some examples of irrational numbers:

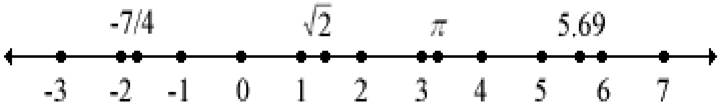
$$\pi = 3.1415926535..., \sqrt{2} = 1.4142135623..., \sqrt{7} = 2.6457513110...$$

Comment. Any root that is not a perfect root is an irrational number. Thus, any roots, such as the following examples, are irrational.

$$\sqrt{2}, \sqrt{3}, \sqrt{5}, \sqrt{6}, \sqrt{7}, \sqrt{8}, \sqrt{10}, ..., \sqrt[3]{2}, \sqrt[3]{13}, \sqrt[5]{20}, \sqrt[9]{10002229}$$

7. *Real* (*Real*) *numbers*. The set of real numbers is denoted by the letter *R*. Each number (with the exception of complex numbers) is contained in the set of real numbers. When the general term "number" is used, it refers to a real number. All the following types or numbers can also be considered as real numbers.

8. ***A*** real-number line. Each real number can be connected with one point on a real-number line



For each set *M* one can construct a new set whose elements are all subsets of M and only these. Then the set *M* is called the universe *I*, and the set of all its subsets is a Boolean.

If the cardinality of the universe is m, then the power of its Boolean

$$\boldsymbol{B(I) = 2^m}.$$

Examples II.1.3.

1. The nesting of the set of natural numbers *N* into the set of integers *Z* and the nesting of the latter into the set of real numbers *R* are shown in Figure II.1.3.

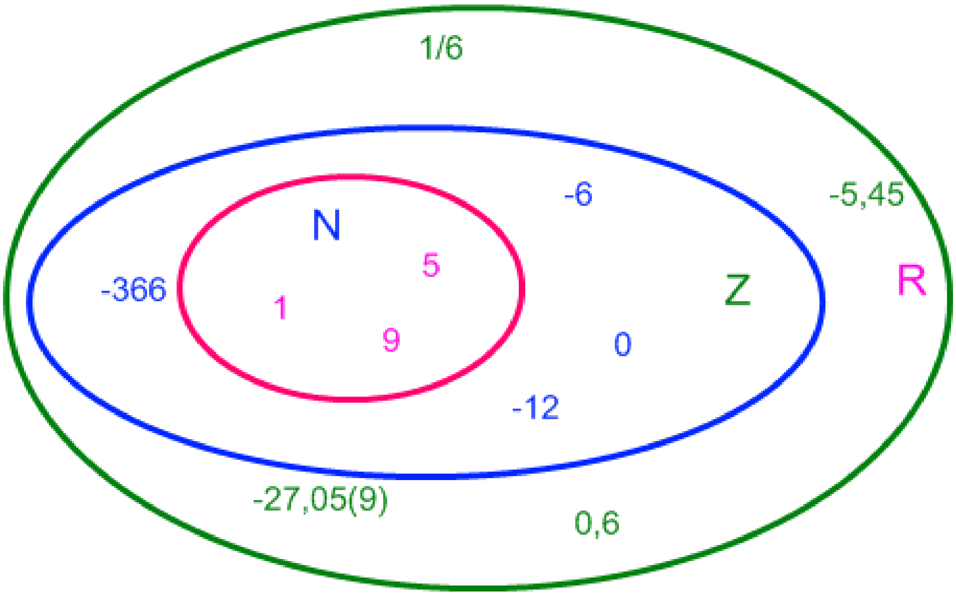


Figure II.1.3. Nesting of numerical sets in each other.

2. We take as the universe $\boldsymbol{I}$ the set of natural numbers on the interval [1, 3], $\boldsymbol{I} = \{1, 2, 3\}$, then boolean

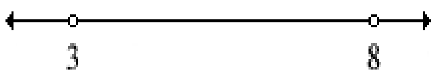
$B(I) = \{\{1\}, \{2\}, \{3\}, \{1, 2\}, \{1, 3\}, \{2, 3\}, \{1, 2, 3\}\}$

If the elements of a numerical set consist of all real numbers between a given pair of numbers, then such a set is called an interval. An interval can be considered as a segment of a real-number line. The end point of the interval is one of two points that mark the ends of the segment.

An interval can include either one endpoint, both endpoints, or no endpoint. To distinguish between these different intervals, we use the interval notation.

An open interval does not include endpoints. An exception from endpoints is indicated by parentheses (,) in interval notation. When the interval is represented by a real-number line segment, the end point exception is shown by an open point. For example, the range of numbers between numbers 3 and 8, with the exception of 3 and 8, is written in interval notation as: $(3, 8) = \{x: 3<x<8\}$.

In the segment of a real number line, this interval would be represented by a line like this:



A closed interval includes endpoints. The inclusion of the endpoints is indicated in square brackets [ ] in the interval notation. When the interval is represented by a segment of a real-number line, the inclusion by an end point is shown by a closed point. For example, the range of numbers between numbers 1 and 11, including both 1 and 11, is written in interval notation as: [1, 11] = $\{x: 1 \leq x \leq 11\}$.

In the segment of a real-number line, this interval is represented as follows:

One of the ends of the interval can be turned on, while the other is excluded. The interval [*a*, *b*) represents all numbers between a and b, including *a*, but not *b*. Similarly, the interval (*a*, *b*] will represent all numbers between *a* and *b*, including *b*, but not *a*. These intervals are presented in more detail in the table below.

Infinite intervals are those that do not have a finite interval, either in the positive or negative direction, or both. The interval is always expanding in this direction. Infinite intervals are shown in the table:

Table 7.1. Intervals.

Name	Old notation	New notation	Set constructor
Open interval	(a,b)	$]a,b[$	$\{x \mid a<x<b\}$
Semi open (closed) interval	$[a,b)$	$[a,b[$	$\{x \mid a\leq x<b\}$
	$(a,b]$	$]a,b]$	$\{x \mid a<x\leq b\}$
Closed interval	$[a,b]$	$[a,b]$	$\{x \mid a\leq x\leq b\}$
Open numerical beam	$(a,+\infty)$	$]a,+\infty[$	$\{x \mid x>a\}$
	$(-\infty,b)$	$]-\infty,b[$	$\{x \mid x<b\}$
Closed numeric ray	$[a,+\infty)$	$[a,+\infty[$	$\{x \mid x\geq a\}$
	$(-\infty,b]$	$]-\infty,b]$	$\{x \mid x\leq b\}$
Numeric axis	$(-\infty,+\infty)$	$]-\infty,+\infty[$	***R***

In mathematics, the interval is the set of real numbers with the property that any number that lies between two numbers in the set is also included in the set. For example, the set of all numbers *x* satisfying
$0\leq x\leq 1$ is an interval that contains 0,1and all numbers between them.

Other examples of intervals are the set of all real numbers *R*, the set of all negative numbers, and the empty set.

Intervals are also defined on an arbitrary linearly ordered set, for example, on a set of integers or rational numbers.

End points of the interval.

The interval of numbers between *a* and *b*, including *a* and *b*, is often denoted by [*a*, *b*]. The numbers *a* and *b* are called the endpoints of the interval.

To show that one of the ends must be excluded from the set, the corresponding square brackets can either be replaced in parentheses, or vice versa:

$$(a,b) =]a,b[= \{x \in R: \quad a < x < b\},$$

$$[a,b) = [a,b[= \{x \in R: \quad a \le x < b\},$$

$$(a,b] =]a,b] = \{x \in R: \quad a < x \le b\}, \ [a,b] = [a,b] = \{x \in R: \quad a \le x \le b\}.$$

Note that (*a*, *a*), [*a*, *a*) and (*a*, *a*) each represent an empty set, while [*a*, *a*] denotes the set {*a*} .For *a*> *b*, all four notations are It is customary to represent an empty set.

1.Endless ends. In both record styles, you can use an infinite endpoint to indicate that there is no boundary in this direction. In particular, we can use *a* = -∞ or *b* = + ∞ (or both). For example, (-∞, 0), (0, + ∞) and (-∞, + ∞) represent the set of all negative, the set of all positive and the set of all real numbers, respectively.

2.*Intervals are an integer*. The notation [*a* .. *b*] when *a* and *b* are integers, or {*a* .. b} or simply *a* .. *b* is sometimes used to indicate the range of all integers between *a* and *b*, including both. This designation is used in some programming languages.

The number of intervals that has a finite bottom or upper end always includes that endpoint. Thus, the elimination of finite ones can be explicitly denoted as *a* .. *b* - 1, *a*+1 .. *b* or *a* + 1 .. *b* - 1. The alternative-bracket designation as [*a* .. *b*) or [*a* .. *b* [Rarely used for integer intervals.

An open interval does not include its endpoints and is indicated in parentheses. For example (0,1) means that the end point is greater than 0 and less than 1. The closed interval includes its end points and is denoted in square brackets. For example, [0,1] means that the end point is greater than or equal to 0 and less than or equal to 1.

3. *Classification of intervals*. The intervals of real numbers can be divided into eleven different types, listed below; Where *a* and *b* are real numbers,:

Empty: $[b, a] = (a, a) = [a, a) = (a, a] = \{\} = \emptyset$,

Degeneracy: $[a, a] = \{a\}$,

In fact, it is limited:

Open: $(a, b) = \{x \mid a < x < b\}$,

Closed: $[a, b] = \{x \mid a \leq x \leq b\}$,

Left is closed, on the right is open: $[a, b) = \{x \mid a \leq x < b\}$,

Open left, right closed:,

Left is bounded and right unbounded:

Open to the left:,

Left closed:,

Left and right unlimited limited:

Right open:,

Cl $[a, b) = \{x \mid a \leq x < b\}$ osed right:,

Unlimited at both ends:

In some contexts, the interval can be defined as a subset of extended real numbers, the set of all real numbers complemented by -∞ and + ∞. In this interpretation, the notation [-∞, b], [-∞, b), [a, + ∞], and (a, + ∞) are all significant and distinct. In particular, (-∞, + ∞) denotes the set of all Simple real numbers, while [-∞, + ∞] denote extended real numbers.

Task II.3.1:

Let U = {1,2,3,4,5}; *A* = {1,3,5}; B = {2,4}; C = {2,3,4}; D = {5}. Calculate; $\overline{(A \cap B)}$, (B \ D) \ (A∪C), (U \ B) ∪D

Let U = {1, 3, 4, 5, 7, 9}; A = {1, 3, 9}; B = {5, 7, 9}; C = {4, 5}; D = {9}. Calculate (U \ D) \ C; A∩D; $(A \cap \overline{B})$; $(D \cap \cap \overline{B})$;

Let U = {2, 4, 6, 8, 10}; A = {2, 4}; B = {4, 6, 8}; C = {2, 6, 10}; D = {4}. Calculate $A \cap \overline{D}$; (A \ B) ∩ (U \ D); $(\overline{B} \cup C.)$

Let A = {1,2, 4}, B = {3, 4, 5, 6}, C = {5, 7} Calculate: A × C; A∪; A △ B; (A∩B) × C; (A \ B) × C.

Questions II.3.1:

1. If $A = \{1,2,3\}$, $B = \{3,4\}$, then what does $\{1,2,3,4\}$ mean?
2. If $A = \{1,2,4\}$, $B = \{4,3,2\}$, then what does $\{2,4\}$ mean?
3. If $A = \{1,2,3,4\}$, $B = \{3,4,5,6\}$, then what does $\{2,4\}$ mean?
4. How many elements in each of the sets: $\{\varnothing, \{\varnothing\}, \{\varnothing, \{\varnothing\}\}\}$.

Tests II.3.1:

1. Determine | D |, if D consists only of Arabic numerals,

A) 10

B) 26

C) 28

D) 0

E) 30

2. What result will be obtained after the operation $A \cap B$ for the given sets $A = \{0, 2, 4, 6\}$ and $B = \{-2, -1, 0, 1, 2\}$?

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) $\varnothing$

3. What result will be obtained after the operation $A \cup B$ for the given sets $A = \{1, 3, 5\}$ and $B = \{2, 4, 6, 8\}$?

A) $\{1, 2, 3, 4, 5, 6, 8,\}$

B) $\{1, 3, 5\}$

C) $\{2, 4, 6, 8\}$

D) $\varnothing$

E) $\{1, 3, 8\}$.

III. FOUNDATIONS OF MATHEMATICAL LOGIC

III.1. Logical Calculus

III.1.1. Proposional logic

The application of mathematical methods in logic becomes possible when judgments are formulated in some precise language. Such exact languages have two sides: syntax and semantics. A syntax is a set of rules for constructing language objects (usually called formulas). Semantics is the totality of agreements describing our understanding of formulas (or some of them) and allowing one formulas to be considered correct, while others are not.

A calculus is a set of inference rules that allow us to consider some statements - formulas that are deducible. Formulas are formed with the help of logical connections.

Saying (affirmation) is a narrative sentence, in relation to which one can say, it is true-1 or false-0.

The content of any science is made up of statements about the objects of its subject area. The logic of statements abstracts from the concrete content of statements and studies the structure of complex statements and their logical connections.

Statements can communicate with each other with the help of logical links, which are given in Table III.1.1.1.

Table III.1.1.1. Logical connectives.

Logical links	**Logical operations**	**Marking**	**Terrain**
It is not true that	inversion	$\neg$	unary
Or	disjunction	?	binary
And	conjunction	?	binary
If ..., then	mplication	>	binary
If and only if the	equivalence	~	binary

We will study mathematical logic with the help of the metalanguage, which differs from the objective language of the studied logic. If a natural language can be used as a subject language with a combination of the language of mathematics, then to determine the metalanguage - the language of the logic of utterances - it is necessary to define the alphabet, syntax and semantics:

Alphabet of the language of propositional logic.

The alphabet of the language of propositional logic consists of logical constants and a countable set of utterances denoted by lowercase Latin (prepositional) letters with or without indices, and also from the designation of the five logical operations listed in Table III.1.1.1 and parentheses to indicate the priority of these operations :

(1) 0, 1 - logical constants;

(2) p, q, r, ..., - lowercase Latin letters with or without indices (propositional letters) are used to denote atomic utterances;

(2), $\wedge$, $\vee$, $\rightarrow$, $\neg$, $\leftrightarrow$ are signs of logical operations for designating logical bundles of statements;

(3) (,) - parentheses to indicate the priority of logical operations.

Syntax of the language of propositional logic.

The syntax of the utterance language is the rules that allow you to build complex statements from the elements of the alphabet, and inductively define the concepts "formula" as follows:

1) ***Induction basis:*** every propositional letter denoting some atomic statement is a formula;

2) ***Induction step***: A and B are formulas, then:

$\neg A$ is the formula;

$A \wedge B$ is the formula;

$A \vee B$ is the formula;

$A \rightarrow B$ is a formula;

$A \leftrightarrow B$ is a formula.

3) ***Induction restriction***: the formula is obtained only with the help of the rules described in 1) - basis of induction and 2) - induction step.

In the definition of formulas, meta-letters A, B are used; Symbols that do not belong to the subject language.

If in the formula the operation ¬ is applied only to atoms, then such a formula is called a formula with close negations.

A subformula is part of a formula itself that is a formula.

Semantics of the language of propositional logic.

The semantics of the utterance language are the rules for interpreting formulas that give certain logical meanings to formulas.

The value of the truth of the formula depends only on the structure of this formula and on the truth values of its components, i.e. The value of a formula is a function of the values of its components.

First in the table I.2.3.2. We define propositional letters and logical operations in a domain of two elements {0, 1}:

Table I.2.3.2. Table of truth for logical operations.

p	q	$\neg p$	$p \wedge q$	$p \vee q$	$p \rightarrow q$	$p \leftrightarrow q$
1	1	0	1	1	1	1
1	0	0	0	1	0	0
0	1	1	0	1	1	0
0	0	1	0	0	1	1

Interpretation of the formula is a mapping of ι, that associates with each atomic statement of this formula a certain truth value. The interpretation of ι, given on the set of atomic statements can be extended (extended) to the set of formulas (complex sentences) by induction.

The value of the formula $F[p_1, ..., p_n]$ for a given interpretation of the propositional letters ι: $\{p_1, ..., p_n\} \Rightarrow \{0,1\}$ in it can be determined by induction on the construction of the formula:

$F = p$: $F[\iota] = \iota(p)$;

$F = A \wedge B$: $F[\iota] = (A \wedge B)[\iota] = A[\iota] \wedge B[\iota]$;

$F = \neg A$: $F[\iota] = \neg A[\iota]$;

$F = A \vee B$: $F[\iota] = (A \vee B)[\iota] = A\,[\gamma] \vee B[\iota]$;

$F = A \rightarrow B$: $F[\iota] = (A \rightarrow B)[\iota] = A\,[\iota] \rightarrow B[\iota]$;

$F = A \leftrightarrow B$: $F[\iota] = (A \leftrightarrow B)[\iota] = A\,[\iota] \leftrightarrow B[\iota]$.

Interpretation, in which the truth value of the formula is 1, is called the model of this formula.

Thus, the logic of propositions expresses certain functional-truth relations between statements that are given by a truth table. The truth table for n operands has 2n variants of the operand values represented by the rows of the table. Each of these lines has its own meaning of truth.

Example III.1.1.

1. Let the formula $(p \wedge q) \vee r \to \neg(p \vee q)$ be given. It is necessary to determine the semantics of this formula. For this, a truth table will be constructed for it, using the rules of interpretation.

The table of truth formulas $(p \wedge q) \vee r \to \neg(p \vee q)$: will be composed of chains that set the values of all of its subformulas. Below is the truth table of the formula $(p \wedge q) \vee r \to \neg(p \vee q)$:

p	q	r	$p \wedge q$	$(p \wedge q) \vee r$	$p \vee q$	$\neg(p \vee q)$	$(p \wedge q) \vee r \to \neg(p \vee q)$
1	1	1	1	1	1	0	0
1	1	0	1	1	1	0	0
1	0	1	0	1	1	0	0
1	0	0	0	0	1	0	1
1	1	1	0	1	1	0	0
0	1	0	0	0	1	0	1
0	0	1	0	1	0	1	1
0	0	0	0	0	0	1	1

Examples III.1.1. Examples of statements are the following narrative sentences;

1) The number three is less than five;

2) It is not true that the computer is smarter than a human;

3) The programmer did not work or the computer was broken;

4) He attended classes and successfully passed the exam;

5) He speaks then and only when he hears.

Tasks III.1.1.

1. Compare the truth tables of the formulas $\neg A \vee B$ and $A \to B$.

2. Find the tautology among the following formulas:

a) $(A \& \neg A) \to (B \vee C \to (C \to \neg A))$;

b) $(A \to \neg A) \sim \neg A$;

3. After calculating only one line of the truth table, find formulas that are not tautologies:

a) $A \vee B \to A \& B$;

b) $(A \to B) \to (B \to A)$.

4. Find the value of the formula $(\neg B \to A)[\gamma]$, if there are $(\mathrm{A} \vee B)[\gamma]=1$ and $(A \to B)[\gamma]=1$.

Questions III.1.1.

1. Will the formula $(\neg A \vee B) \& (C \to (A \leftrightarrow B))$ be a tautology?

2. Are there any statements A, B and C such that the following conditions are satisfied for them simultaneously in some interpretation:

$\gamma(\mathrm{A} \to \mathrm{B})\ [\gamma] = \mathrm{T}$; $(B \vee C)[\gamma] = \mathrm{F}$; $(B \leftrightarrow (A \& \neg C))[\gamma] = \mathrm{F}$?

3. Are the following statements equivalent?

A⊃B and if A, then B?

A⊃B and as soon as A, then B?

A⊃B and in case A there is a B?

A⊃B and A implies B?

4. What does the alphabet of logic of statements consist of?

5. How is the subformula defined?

6. How is the value of the formula calculated for a given interpretation of the propositional letters in it?

7. How is the semantics of the language of utterances determined?

Tests III.1.1.

1. What is tautology?

A) A formula that is true for all interpretations of its propositional letters.

B) The column of values of which contains one false values.

C) A formula that can be true or false.

D) A formula that can only be false.

2. Which of them is the law of De Morgan?

A) $\neg(\neg p) \sim p$

B) $p \sim p$

C) $\neg(p \vee q) \sim \neg p \wedge \neg q$

D) $p \wedge \neg p \sim 0$

E) $p \vee \neg p \sim 1$

3. What is the order of the operations in the expression $D \vee \neg F \wedge G$?

A) first ¬F, then $\neg F \wedge G$,, and at the end $D \vee \neg F \wedge G$

B) first $\neg F \wedge G$, and at the end $D \vee \neg F \wedge G$.

C) first ¬F, and at the end $D \vee \neg F \wedge G$.

D) first ¬F, then $\neg F \wedge G$.

E) first ¬G, then $\neg F \wedge G$, and at the end $D \vee \neg F \wedge G$

4. What is a formula with close negatives?

A) A formula in which ¬ applies only to atoms.

B) A formula that is false for all interpretations.

C) A formula that is true for all interpretations.

D) A formula that is undefined for all interpretations.

E) A formula in which the operation ¬ is applied to all subformulas.

III.1.2. Logic of predicates

Predicates are a language expression that denotes a property or relation.

The calculus of predicates is a formal calculus admitting a statement about variables, fixed logical functions, and predicates.

Before we consider the meta-language of the predicate calculus, we give definitions of the concept of a term, a logical function, and a perdicate.

We define the concept of a logical function as an n-place operation on the set {F, T}.

Alphabet:

(1) $x, y, \dots, x_1, x_2,$ ...are subject variables;

(2) $f, g, \dots, f_1, f_2,$ ... are function symbols.

Term:

(1) $x, y, \dots, x_1, x_2, \dots$ are subject variables are terms;

(2) if $f^{(n)}$is a functional symbol, $t_1, t_2, \dots, t_n$ are terms, then $f^{(n)}(t_1, t_2, \dots, t_n)$ is a term.

Term meaning:

(1) if t is an object variable x, then Val $t = \gamma(x)$;

(2) if $t = f^{(n)}(t_1, t_2, \dots, t_n)$, then $Val\ t = f^{(n)}(t_1, t_2, \dots, t_n)$

Function:

$t = f^{(n)}(t_1, t_2, \dots, t_n)$ is representable by the term

$t = f^{(n)}(v_1, v_2, \dots, v_n)$ if

$\{v_1, v_2, \dots, v_n\} \subseteq \{x_1, x_2, \dots, x_n\}$ and $t[\gamma] = f^{(n)}[\gamma]$ for all interpretations

$$\gamma: \{x_1, x_2, \dots, x_n\} \Rightarrow \{\mathrm{F}, \mathrm{T}\}$$

The predicates are the logical functions $J^{(n)}(x_1, x_2, \dots, x_n)$, which are given on the non-empty domain *D* and take the values 0, 1.

The predicate $J^{(n)}(x_1, x_2, \dots, x_n)$ becomes a statement after the signification of the variables entering it on the elements of the set *D*.

Alphabet:

(1) $x, y, \dots, x_1, x_2,$ ... are subject variables;

(2) $P^{(n)}(x_1, x_2, \dots, x_n)$ are predicate letters (n = 0,1, ...);

(3) &, ∨, ¬, →, ↔, ∀, ∃ are logical connectives and quantifiers;

(4) (,) are auxiliary symbols.

Formulas:

(1) $P^{(n)}(x_1, x_2, \ldots, x_n)$ are elementary formulas or atoms;

(2) if A, B are formulas, then A&B, A∨B, ¬A, A→B, A↔B are formulas;

(3) if A(x) is a formula with free variable x, then $\forall x$A(x) and $\exists x$A(x) are the formulas.

Free and bound variables.

Variables in the range of the quantifier with respect to this variable are called bound, otherwise they are free.

Interpretation of the formula.

The value of the formula $E[P_1, P_2, \ldots, P_m;\ x_1, x_2, \ldots, x_n]$ in the interpretation of the predicate letters $\tau: P^{(n)} \Rightarrow J^{(n)}$ and the notation $\gamma: \{x_1, x_2, \ldots, x_n\} \Rightarrow D$ ($D \neq \varnothing$) of subject variables, is denoted by E[τ,γ], we define by induction on the construction of the formula E:

1) $E = P^n(x_1, x_2, \ldots, x_n)$, then E[τ,γ] = P[γ];
2) E = (A&B) $[P_1, P_2, \ldots, P_m;\ x_1, x_2, \ldots, x_n]$, then

$$E[\tau,\gamma] = A[\tau,\gamma]\ \&\ B[\tau,\gamma].$$

Similarly for the remaining logical connectives.

3) $E = \forall x_1 A[P_1, P_2, \ldots, P_m;\ x_1, x_2, \ldots, x_n]$, then

$$E[\tau,\gamma] = \forall x_1 A[\tau, x_1, \gamma] = T],$$

where $\gamma: \{x_1, x_2, \ldots, x_n\} \Rightarrow D$, if A [τ, a, γ] = T for any $a \in D$.

4) $E = \exists x_1 A[P_1, P_2, \ldots, P_m;\ x_1, x_2, \ldots, x_n]$, then

$$E[\tau,\gamma] = \exists x_1 A[\tau, x_1, \gamma] = T],$$

where $\gamma: \{x_1, x_2, \ldots, x_n\} \Rightarrow D$, if A [τ,a,γ] = T for some $a \in D$.

The formula E$[P_1, P_2, \ldots, P_m;\ x_1, x_2, \ldots, x_n]$ is said to be valid or tautological if for any domain $D \neq \varnothing$, for any interpretations τ of predicate letters and any valuation γ of subject variables in domain D, E[τ,γ] = T.

Examples III.1.2

1. Let P be a false statement 1 = 5, let Q be a false statement 3 = 7, and R a true statement 4 = 4. Show that conditional statements: "if P, then Q" and "if P, then R", both Are true.

Decision. If 1 = 5, then, adding 2 to both sides of the equality, we get that 3 = 7. Consequently, the saying "if P, then Q" is true. We subtract now the number 3 from both sides of the equality 1 = 5 and we come to -2 = 2. Therefore (-2) 2 = 22, that is, 4 = 4. Thus, "if P, then R" is also true.

2.We show that the formula P (x, y) → Q (x) is not 1-valued, hence, not universally valid.

Help. D = {1} is a singleton region, I_1 and I_2 are interpretations of the letter P, and J_1 and $J_2$are interpretations of the letter Q:

x	y	I_1	I_1	J_1	J_1
1	1	T	F	T	F

The truth table of the formula P (x, y) → Q (x):

x	y	P(x,y)	Q(x)	P(x,y)→Q(x)
1	1	T	T	T
1	1	T	F	F
1	1	F	T	T
1	1	F	F	T

3. We show that the formula $\forall x(\exists x\ P(x) \to P(x))$ is not 2-valued.

Decision. D = {1, 2}, J_1, J_2, J_3, J_4 are the interpretations of the letter P:

x	J_1	J_2	J_3	J_4
1	T	T	F	F
2	T	F	T	F

The truth table of the formula $\forall x(\exists x\ P(x) \to P(x))$:

x	P(x)	$\exists x$ P(x)	$\exists x$ P(x)→P(x)	$\forall x(\exists x P(x) \to P(x))$
1	J_1	T	T	T

2	J_1		T	
1 2	J_2 J_2	T	T F	F
1 2	J_3 J_3	T	F T	F
1 2	J_4 J_4	F	T T	T

4. We show that the formula $\forall x\ \exists y$ P(x, y)→$\exists y\ \forall x$ P(x, y) is not valid.

Decision. Let D = {1, 2}, then the interpretation of the predicate letter P (x, y) is given by the following table:

X	Y	J_1	J_2	J_3	J_4	…	J_7	…
1	1	T	T	T	T	…	T	…
1	2	T	T	T	T	…	F	…
2	1	T	T	F	F	…	F	…
2	2	T	F	T	F	…	T	…

In particular, for the interpretation of J_7 we obtain:

For x=1: $\exists y J_7(1,y)$=T, for x=2: $\exists y J_7(2,y)$=T, then $\forall x \exists y J_7(x,y)$=T.

For y=1: $\forall x\ J_7(x,1)$=F, for y=2: $\forall x J_7(x,2)$=F, then $\exists y \forall x J_7(x,y)$=F.

Hence $\forall x \exists y J_7(x,y) \rightarrow \exists y \forall x J_7(x,y)$ = F.

Tasks III.1.2. Will the following expressions be formulas, and if they are formulas, what are the variables in them that are free and which are related:

1) $\forall x_1 \exists x_2 \forall x_3 P(x_1, x_2, x_3, x_4)$;

2) $\forall x_1 P(x_1, x_2) \supset \exists x_2 P(x_1, x_2)$;

3) $\exists x_1 \exists x_2 P(x_1, x_2) \wedge Q(x_1, x_2)$.

2. Let $\boldsymbol{M} = < Z^+, f >$, where $Z^+$is the set of non-negative integers, f is the correspondence that for predicate symbols $S^{(3)}(x, y, z)$, $P^{(3)}(x, y, z)$ defines the following predicates:

$$S^{(3)}(x, y, z) = 1 \ \Leftrightarrow x + y = z;\ P^{(3)}(x, y, z) = 1 \ \Leftrightarrow x \cdot y = z.$$

Write in the model ***M*** formulas that express the following statements:

1) $x = 0$;
2) $x = 1$;
3) $x = 2$;
4) x is an even number;
5) x is an odd number;
6) $x = y$;
7) $x \leq y$;
8) $x < y$;
9) x divides y;
10) commutativity of addition;
11) associativity of addition;
12) the commutativity of multiplication;
13) associativity of multiplication;
14) the distributivity of addition with respect to multiplication;

Questions III.1.2.

1. What are the similarities and differences between the classical calculi of the logic of predicates and the logic of propositions?

2. In what interpretation does the statement take on a true value?
1) Any prime number is greater than zero;
2) There is a planet that rotates around the earth;
3) There is a season that is colder than summer.

3. How to write sentences with the help of predicates?
1) Happy hours are not observed;
2) Only a fool can be happy;
3) There are happy people in the world;
4) Some fool does not watch the clock.

4. Record with the help of predicates and quantifiers the condition "All animals are mortal. All people are animals. "Conclusion: All people are mortal" and to prove it is fair.

Tests III.1.2.

1.The predicate is:

A) A language expression denoting a property or relation.

B) A term used in mathematical logic with respect to a particular logical system, to indicate what is for a given logic.

C) The main section of modern logic, which describes the conclusions that take into account the internal structure of utterances.

D) The result of some reconstruction of the natural language.

E) An expression that takes the value F or T.

2.Which of the following statements are true, and which ones are false?

A) All cats have a tail;

B) There exists an integer x satisfying the condition $x^2 = 2$;

C) Every utterance is a negation of oneself;

D) Every utterance is a logical consequence of oneself;

E) That is, every utterance is a union of oneself.

3.The predicate calculus is:

A) A formal calculus admitting a proposition with respect to variables, fixed functions, and predicates.

B) Formal theory, the main object of which is the concept of logical utterance.

C) A term used in mathematical logic with respect to a particular logical system, to indicate that all laws of the propositional calculus are valid for a given logic.

D) Rule of inference in the propositional calculus.

E) Axiomatic system using expressions and predicates.

III.2. Logical laws and conclusions

III.2.1. Tautologies and equivalence properties

Before, to proceed to logical reasoning, we must master the logical laws that allow us to compare two statements to equivalence, in order, if necessary, to replace one another. Also, we need a technique for detecting tautologies, more powerful than a truth table. And, finally, we will consider methods of reasoning that can be useful for solving logical problems formulated in natural language.

Definition III.2.1.

1. Two statements are said to be equivalent if they take the same values on the same states of their variables.

2. Tautology is a statement whose meaning is true for all interpretations of propositional letters that enter into it and is denoted by a symbol.

3. Contradiction is a statement whose meaning is false for all interpretations of propositional letters entering into it.

4. Two statements are said to be equivalent if they take the same values on the same states of their variables.

Thus, one way to establish the equivalence of two statements is to calculate their truth tables and compare them. We will, however, use a different method.

Two statements A and B are equivalent if and only if A ~ B is a tautology (generally valid).

Below is a list of the main tautologies in the propositional calculus:

1a. $\models A\to(B\to A)$

1b. $\models (A\to B)\to((A\to(B\to C))\to(A\to C))$

2. $\models A\to(B\to A\&B)$

3a. $\models A\&B\to A$

3b. $\models A\&B\to B$

4a. $\models A\to A\vee B$

4b. $\models B\to A\vee B$

5. $\models (A\to C)\to((B\to C)\to(A\vee B\to C))$

6. $\models (A\to C)\to((A\to\neg C)\to\neg A)$

7. $\models \neg\neg A \rightarrow A$

8. $\models (A \rightarrow B) \rightarrow ((B \rightarrow A) \rightarrow (A \leftrightarrow B))$

9a. $\models (A \leftrightarrow B) \rightarrow (A \rightarrow B)$

9b. $\models (A \leftrightarrow B) \rightarrow (B \rightarrow A)$

10. $\models (A \rightarrow (\neg A \rightarrow C)$

To simplify expressions, we define what it means for two expressions to be equivalent and replace the more complex statement by the less complex one.

Basic, frequently used equivalence properties:

I. Commutativity

1. $A \wedge B \sim B \wedge A$
2. $A \vee B \sim B \vee A$

II. Associativity

1. $A \wedge (B \wedge C) \sim (A \wedge B) \wedge C$
2. $A \vee (B \vee C) \sim (A \vee B) \vee C$

III. Distributivity

1. $A \wedge (B \vee C) \sim (A \wedge B) \vee (A \wedge C)$
2. $A \vee (B \wedge C) \sim (A \vee B) \wedge (A \vee C)$

IV. De Morgan's Law

1. $\neg(A \vee B) \sim \neg A \wedge \neg B$
2. $\neg(A \wedge B) \sim \neg A \vee \neg B$

V. The law of implication

1. $A \sim B \sim (A \rightarrow B) \wedge (B \rightarrow A)$

VI. The law of forward and backward conditions

1. $A \sim B \sim (A \rightarrow B) \wedge (B \rightarrow A)$

VII. The property of negation

1. $\neg(\neg A) \sim A$

VIII. The law of identity

1. $A \sim A$

IX. The law of exclusion of the third

1. $A \vee \neg A \sim T$

X. Law of Contradiction

1. $A \wedge \neg A \sim F$

XI. Disjunction Properties

$1.\ A\vee A \sim A$

$2.\ A\vee T \sim T$

$3.\ A\vee F \sim A$

$4.\ A\vee(A\wedge B) \sim A$

XII. Conjunction Properties

$1.\ A\wedge A \sim A$

$2.\ A\wedge T \sim A$

$3.\ A\wedge F \sim F$

$4.\ A\wedge(A\vee B) \sim A$

XIII. Laws of absorption

$1.$ A&A~A, A∨A~A

$2. A\&(A\vee B) \sim A, A\vee(A\&B) \sim A$

$3.\ A\&T \sim A, A\&F \sim F, A\vee T \sim T, A\vee F \sim A.$

$4.\ A\to(B\to C) \sim A\&B\to C.$

5. $(A\to B)\&(B\to A) \sim (A\leftrightarrow B)$

XIV. Law of Contraposition

1. $(A\to B) \sim (\neg B\to\neg A)$

XV. The law of identity

$1. A\to A$

We will use these properties for different purposes. Commutativity, for example, allows us to swap elements of an utterance, in order to simplify it. Associativity allows you to remove brackets. Distributivity allows us to collect similar terms, just as we do in arithmetic terms. The law of implication allows you to leave the operation → using only operations □, ∨,. In order to verify the correctness of these properties, it is sufficient to construct their truth tables.

Using the equivalence properties, we can simplify utterances.

Example III.2.1:

1. Simplify the expression $(p\vee\neg q)\wedge r\wedge(\neg p\vee q)$. We apply the rules:

I.1 we obtain $(p\vee\neg q)\wedge(\neg p\vee q)\wedge r$

I.2 we obtain $(\neg q\vee p)\wedge(\neg p\vee q)\wedge r$

V.1 we obtain $(q\to p)\wedge(p\to q)\wedge r$

VI.1 we obtain$(p\sim q)\wedge r$

Finally, we obtain $(p\vee\neg q)\wedge r\wedge(\neg p\vee q) \sim (p\sim q)\wedge r$

2. Simplify the expression $p \vee (\neg q \rightarrow p) \vee \neg q$. We apply the rules:

V.1, we obtain $p \vee (\neg(\neg q \vee p) \vee \neg q$

VII.1 we obtain $p \vee (q \vee p) \vee \neg q$

I.2 we obtain $p \vee (q \vee p) \vee \neg q$

I.2 we obtain $(p \vee p) \vee (q \vee \neg q)$

XI.1 we obtain $p \vee (q \vee \neg q)$

IX.1 we get $p \vee 1$

XI.2 we get 1

In summary, we have proved that $p \vee (\neg q \rightarrow p) \vee \neg q \sim 1$ is a tautology.

3. Simplify the expression $((p \rightarrow q) \rightarrow p) \rightarrow p$.

V.1 we obtain $(\neg(p \rightarrow q) \vee p) \rightarrow p$

V.1 we obtain $(\neg(\neg p \vee q) \vee p) \rightarrow p$

IV.1 we obtain $((\neg(\neg p) \wedge \neg q) \vee p) \rightarrow p$

VII.1 we obtain $((p \wedge \neg q) \vee p) \rightarrow p$

I.2 we obtain $(p \vee (p \wedge \neg q)) \rightarrow p$

XI.4 we obtain $p \rightarrow p$

V.1 we obtain $\neg p \vee p$

I.2 we obtain ppp

IX.1 we get 1

As a result, it is proved that $((p \rightarrow q) \rightarrow p) \rightarrow p$ is a tautology.

Task III.2.1:

Suppose we are given a composite proposition $(p \vee q) \wedge (p \vee \neg q)$. Prove that this compound statement is equivalent to the statement p.

Help.

It is necessary to fill with the values in the truth table the column for the statement $(p \vee q) \wedge (p \vee \neg q)$.

Then the resulting column of values is compared with the column for p. If they coincide, then we can conclude that the expression $(p \vee q) \wedge (p \vee \neg q)$ is equivalent to p and denote it as $(p \vee q) \wedge (p \vee \neg q) \sim p$. This means that wherever we meet the expression $(p \vee q) \wedge (p \vee \neg q)$, we can replace it with p.

Tests III.2.1.

1. Will the following statements be equivalent

A) $A \wedge B$ and A and B?

B) $A \wedge B$ and not only A, but also B?

C) $A \wedge B$ and B, although A?

D) $A \wedge B$ and B, despite A?

E) $A \wedge B$ and both A and B?

2. Will the following statements be equivalent?

A) $A \vee B$ and A or B?

B) $A \vee B$ and A or B?

C) $A \vee B$ and A, if not B?

D) $A \vee B$ and A and B?

E) $A \vee B$ and A or B?

3. Will the following statements be equivalent?

A) $A \sim B$ and A, if and only if B?

B) $A \sim B$ and if A, then B, and back?

C) $A \sim B$ and A, if B, and B, if A?

D) $A \sim B$ and A is equivalent to B?

E) $A \sim B$ and A if and only if B?

III.2.2. Logical conclusions

Proof based on inference rules.

The main purpose of any reasoning is to establish the truth in the form of some universally valid statement, i.e. Tautology. For simple cases, we have a method of truth tables. However, it becomes cumbersome when the number of variables is greater than four.

There is another method called evidence, which is a sequence of logical inferences, the correctness of each of which is strictly logically justified. Thus, the reasoning in this method takes the form of a sequence of logical conclusions.

The process of proof in the propositional logic is essentially a sequence of transformations of the statement p in order to show that p is valid and is a development of the method of simplifying utterances. However, the proof includes an important additional component: a derivation from the assumption. The conclusion in the proof is based on a small number of inference rules. Each step in the proof is either an already proven statement, or a statement that is true by assumption and introduced for subsequent steps. Each step that is a guess is in brackets [,]. These rules establish that some statements can follow from others (axioms), the truth of which has already been established, or are considered as such by assumption. All other steps must be proven. The last step in the proof must be the very statement p:

Let us prove the proposition $p \Rightarrow p$

1. $[p]$
2. p
3. $p \Rightarrow p$ Rule I

The first step is to make the assumption that p is generally valid. Then the second step directly follows from the first. Since we have assumed the validity of p at the first step, we use this fact on the second step. In the third step we use the rules of inference I, which establishes the validity of a statement.

The proof with the help of inference rules is more flexible than the proof with the help of the truth table. In the first case, we can analyze each step in the chain of evidence. At the same time, an unlimited growth of the truth table will not allow us to do this.

When you look closely at the rules of inference, you can see that they are in good agreement with our intuition. For example, take rule VIII. If the validity of the statements p and q was proved at the previous steps, then obviously the statement $p \wedge q$ is also valid.

So, in the sequel, in the proof, we will use either the equivalence rules (in this case each step will be a replacement of the right occurrence in the statement on the left-hand side of the rule) or the inference rules.

I. Introduction →

$$[\boldsymbol{p}] \qquad \frac{q}{p \Rightarrow q}$$

II. Introduction ~

$$\boldsymbol{p \rightarrow q} \qquad \frac{q \Rightarrow p}{p \Leftrightarrow q}$$

III. Removing →

1. $\boldsymbol{p} \qquad \dfrac{p \Rightarrow q}{q}$ (*Modus ponens*)

2. $\neg \boldsymbol{q} \qquad \dfrac{p \Rightarrow q}{\neg p}$ (*Modus tollens*)

IV. Removing ~

1. $\dfrac{p \Leftrightarrow q}{p \Rightarrow q}$

2. $\dfrac{p \Leftrightarrow q}{q \Rightarrow p}$

V. Introduction

$$[\boldsymbol{p}] \quad \frac{F}{\neg p}$$

VI. Removal

1. $\boldsymbol{p} \qquad \dfrac{\neg p}{F}$

2. $\dfrac{F}{p}$

VII. Introduction

1. $\boldsymbol{p} \quad \frac{q}{p \wedge q}$

VIII. Introduction

1. $\frac{p}{p \vee q}$

2. $\frac{q}{p \vee q}$

IX. Uninstalling

1. $\frac{p \wedge q}{p}$

2. $\frac{p \wedge q}{q}$

X. Removal

$[\boldsymbol{p}]\ [\boldsymbol{q}] \qquad \frac{p \vee q \quad r \quad r}{\boldsymbol{r}}$

Deductive output.

Deductive inference is a method of reasoning in which a private conclusion is derived from a general conclusion. A chain of reasoning, where statements are related by logical conclusions.

The beginning (premisses) of deduction are axioms or simply hypotheses that have the character of general statements ("general"), and the end - corollaries from premises, theorems ("private"). If the premises of deduction are true, then its consequences are also true. Deduction is the opposite of induction.

Using deductive inference to prove $p \Rightarrow (q \Rightarrow p)$

[p] - Assumption

[q] - Assumption

1. p – 1.
2. $q \Rightarrow p$ – I, 2, 3
3. $p \Rightarrow (q \Rightarrow p)$ – I, 1, 4

We have assumed the validity of the statements of p and q and using rule I. Introduction.

Using the rule Modus Ronens.

This rule works well when you need to prove statements like "If the show shows" Gioconda - Gioconda ", then I'll buy tickets." If someone made this statement and you saw that the exhibition shows "Gioconda - Gioconda", then you You can conclude that this person bought tickets.

Prove it $((p \Rightarrow q) \wedge (r \Rightarrow p) \wedge r) \Rightarrow q$

1. $[(p \Rightarrow q) \wedge (r \Rightarrow p) \wedge r]$ – Assumption
2. $r \Rightarrow p$ – IX. Removal $\wedge$, 1
3. r – IX. Removal $\wedge$, 1
4. p – III. Modus Ponens, 2, 3
5. $p \Rightarrow q$ – IX. Removal $\wedge$, 1
6. q – III. Modus Ponens. 4, 5
7. $((p \Rightarrow q) \wedge (r \Rightarrow p) \wedge r) \Rightarrow q$ – I. Introduction $\Rightarrow$, 1, 6

Using Modus Tollens.

Prove $((p \Rightarrow q) \wedge \neg q) \Rightarrow \neg p$

1. $[(p \Rightarrow q) \wedge \neg q]$ – Assumption
2. $p \Rightarrow q$ – IX. Removal $\wedge$, 1
3. $\neg q$ – IX. Removal $\wedge$, 1
4. $\neg p$ – III.2. Modus Tollens, 2, 3
5. $((p \Rightarrow q) \wedge \neg q) \Rightarrow \neg p$ – I. Introduction $\Rightarrow$, 1, 4

Using Introduction $\neg$ *and* Removal $\neg$.

Prove $((p \Rightarrow q) \wedge \neg q) \Rightarrow \neg p$

1. $[(p \Rightarrow q) \wedge \neg q]$ – Assumption
2. $p \Rightarrow q$ – IX. Removal $\wedge$, 1
3. $\neg q$ – IX. Removal $\wedge$, 1
4. $[p]$ – Assumption
5. q – III. Modus Ponens, 4, 2

6. $\neg q$ – 3
7. 0 – VI. Removal $\wedge$, 5, 6
8. $\neg p$ – V. Introduction $\neg$ 4, 7
9. $((p \Rightarrow q) \wedge \neg q) \Rightarrow \neg p$ – I. Introduction $\neg$ 1, 8

The proof is by contradiction.

On the use of the rule V. Introduction □ the frequently used method of proof is based - the proof is by contradiction. We have already used it several times. His idea is as follows.

Let us want to prove the general validity of the proposition Q:

"Triangle with sides 2, 3, 4 - not rectangular."

Suppose that $\neg Q$ is universally valid, that is, a triangle with sides 2, 3, 4 - rectangular. Then, using the Pythagorean theorem, we can assert that 4 + 9 = 16, but 4 + 9 = 16. Hence, using the rule VI.1 Removal of $\neg$, we obtain 0. Having 0 and the assumption of general significance $\neg Q$, using the rule V , We obtain the general validity of $\neg(\neg Q)$. Whence, using rule VII from Section III.2.1, we obtain the general validity of Q.

Prove it $((p \Rightarrow q) \wedge p) \Rightarrow q$

1. $[\neg(((p \Rightarrow q) \wedge p) \Rightarrow q)]$ – Assumption
2. $\neg(\neg((p \Rightarrow q) \wedge p) \vee q)$ – The law of implicationV.1, 1
3. $\neg(\neg((p \Rightarrow q) \wedge p)) \wedge \neg q$ – De Morgan's Law IV.1
4. $\neg \boldsymbol{q}$ – IX.2. Removal $\wedge$, 3
5. $((p \Rightarrow q) \wedge p)$ – IX.1. Removal $\wedge$, 3
6. $p \Rightarrow q$ – IX.1. Removal $\wedge$, 5
7. p – IX.2. Removal $\wedge$, 6
8. q – III.1. Modus Ponens, 6, 7
9. 0 – V.1 Removal $\neg$, 4, 8
10. $((p \Rightarrow q) \wedge p) \Rightarrow q$ – V. Introduction $\neg$, 1, 9

The duality principle.

Suppose that the formulas E, F do not contain other operations other than $\neg$, $\wedge$, $\vee$ and are formulas with close negations. Formulas E' and F obtained from E and F by simultaneous replacement everywhere & to $\vee$ and & to $\vee$, are said to be dual to the formulas E and F, respectively. Then the following assertions hold:

a) If $\models \neg E$, then $\models E'$. b) If $\models E$, then $\models \neg E'$.

c) If $\models E \sim F$, then $\models E' \sim F'$. d) If $\models E \rightarrow F$, then $\models F' \rightarrow E'$.

A logical consequence.

Let the formulas $A_1, A_2, \dots, A_n$ and B be given. If the truth of the formula B follows from the simultaneous truth of the formulas $A_1, A_2, \dots, A_n$, then formula B is a logical consequence of the formulas $A_1, A_2, \dots, A_n$ and is denoted by $A_1, A_2, \dots, A_n \models$ B, ($n \geq 1$), here $A_1, A_2, \dots, A_n$ are the premises, and B is the conclusion.

Rules of inference.

In the propositional logic, a single output rule, called modus ponens, is used, which is the procedure for passing from two formulas of the form A, $A \rightarrow B$ (premisses) to the formula B (conclusion):

$$\frac{A,\ A \rightarrow B}{B} \quad \text{(modus ponens)}$$

The requirements that the rules of inference must satisfy are that true conclusions must be obtained from true premises.

Reasoning with the help of predicates.

Reasonings with the help of predicates are analogous to arguments using sentences. The rules of inference, which we introduced in III.2.2 for utterances, are also true for predicates. Replacing all variables with their current values and performing the necessary calculations, we obtain a statement to which the known rules of inference apply. However, there are two special cases for which we do not have the appropriate inference rules: this is the use of quantifiers. Table III.2.2 summarizes the derivation rules for the case of predicates with quantifiers.

Table III.2.2. The inference rules for predicates with quantifiers.

Introduction $\forall$	*Removal* $\forall$

$\dfrac{R \Rightarrow P}{\forall i : R(i) : P(i)}$	$\dfrac{\forall i : R(i) : P(i)}{R(i_0) \Rightarrow P(i_0)}$
Introduction $\exists$	*Removal*$\exists$
$\dfrac{\forall i : R(i) : P(i)}{\neg \exists i : R(i) : \neg P(i)}$	$\dfrac{\exists i : R(i) : P(i)}{\neg \forall i : R(i) : \neg P(i)}$

For example, consider the rule of introduction $\forall$. "It says that if $R \Rightarrow P$ is satisfied on all the states in question, then we can state that for all values of $\forall i : R(i) : P(i)$, where

Consider for example the rule of introduction $\forall$, which states that if $R \Rightarrow P$ is satisfied on all the states considered, then we can state that for all values of i, where $\forall i : R(i) : P(i)$ is satisfied. The second rule for removing $\forall$ is that if the predicate $\forall i : R(i) : P(i)$ is satisfied, then for arbitrary i is satisfied $R(i_0) \Rightarrow P(i_0)$,

As an example demonstrating the use of these rules, let us consider the justification of the proof method by induction. We have used this method more than once to prove the properties of algorithms. With him, we will meet many times when justifying the correctness of programs.

To prove the feasibility of the predicate

$\forall i : R(i) : P(i)$, where $R(i)$ has the form $i \in \{1, 2, ...\}$

Let us consider two cases:

The initial step: We prove $P(1)$

Induction step: We prove $P(i) \Rightarrow P(i+1)$ for any $i \geq 1$.

If we have shown the feasibility of these steps, then we have

$$P(1)$$

$$P(1) \Rightarrow P(2)$$

$$P(2) \Rightarrow P(3) \text{ etc.}$$

Then from the first two lines and the *Modus Ponens rule* we obtain the feasibility of $P(2)$. From $P(2)$ from the third line, applying again Modus Ponens, we get $P(3)$, etc. to infinity.

In this way

$$\forall i \in \{1, 2, ...\} : P(i) .$$

Example III.2.2.

On Wednesday, when the robbery occurred, either Pitts was in the bank's operating room, or Elena in the bank's accounting department. Pitts was never seen in the operating room without Irwin. Irwin left the bank on Wednesday only when he and Helen went to a meeting with clients. If Korn was involved in the robbery, Irvina would not be in the bank. The robbery took place on Wednesday. Could Korn be a robber?

Denote by:

p = Pits was in the operating room;

q = Elena was in the accounting department;

s = Irwin was in the operating room;

h = Corn participated in the robbery;

u = obbery happened on wednesday.

Then the original statements can be written as follows:

1. $u \to (p \vee q)$
2. $p \to s$
3. $\neg s \to \neg q$
4. $h \to \neg s$
5. u
6. From 1, 5 items of Modus ponens we obtain $p \vee q$
7. Suppose that [q]
8. From 3, 7 points of Modus Tollens we obtain s
9. From 7, 8 and "introduction $\to$" we obtain $q \to s$
10. From 4, 10 items of Modus Tollens $\neg h$

So, Korn could not participate in the robbery.

Tasks III.2.2.

1. Translate each of the following reasonings into logical symbols and analyze the result for correctness:

1) I would pay for a computer repair job only if it started working. It does not work. Therefore, I will not pay for repairs.

2) If he had not told her, she would not have known for sure. And if she did not ask him, he would not have told her. But she found out. So she asked him.

3) He said that he would come if there was no rain. But it's raining. So he will not come.

2. Check the correctness of the reasoning: Irvine will not do this work, if it makes Pits. Pitts and Sidorov will do this work if and only if Irwin does it. Sidorov, this work will do, but Irvine does not. Therefore, Pitts will not do this job.

3. Let M be the set of points, lines, and planes of a three-dimensional space. Let us consider the model $D = <M, f>$, where f is the correspondence that determines predicates for the predicate symbols $Q(x, y), R(x, y)$, $P_1(x), P_2(x), P_3(x)$ define the predicats:

$P_1(x) = 1 \Leftrightarrow$ "x *is point*",

$P_2(x) = 1 \Leftrightarrow$ "x *is line*",

$P_3(x) = 1 \Leftrightarrow$ “x is *plane*",

$Q(x, y) = 1 \Leftrightarrow$ "x lies on y",

R (x, y) = 1 ⇔ "x coincides with y".

Using the above predicates, write the following statements in this model:

1) Through every two points one can draw a straight line and, moreover, unique if these two points are different;

2) Through every three points not lying on one line, we can draw a single plane;

3) The definition of parallel lines;

Questions III.2.2.

1. The formulas from which formulas are the following sequences of formulas: A⊃ (B⊃C), A, B⊃C, B, C?

2. Find out which of the statements of each pair are the negatives of each other?

A) In the book more than 100 pages;
B) The book does not exceed 100 pages;
C) This clove is red;
D) This pink is pink;
E) This word is a noun;
F) This word is an adjective.

3. Which of the following sentences are compound statements?

A) In 1 m 100 cm or 10 dm;
B) 27 times 3 and less than 3;
C) It is not true that 45 is an even number;
D) Today is Monday;
E) If the triangle is equilateral, then it is isosceles;

4. Suppose two statements are given in the language of first-order logic. A - $\forall x\ \exists y\ (x \geq y)$ and B - $\exists y\ \forall x\ (x \geq y)$, $x \in N$, $y \in N$, and N is the set of natural numbers.

A) How will these statements be written in a natural language?
B) Is saying A true in this interpretation?
C) Is statement B true in this interpretation?
D) Is B a logical consequence of A?
E) Is A a logical consequence of B?

Tests III.2.2.

1. Statements A and B are equivalent if:

A) If and only if the truths of propositions A and B coincide;
B) If and only if A is true, and B is false;
C) If and only if A is false and B is true;
D) When the conjunction of statements A and B is false;
E) Then, when the disjunction of propositions A and B is true;

2. Find out in what cases the inferences are true:

A) If the number is natural, then it is an integer; The number 6 is an integer; Hence, it is natural;

B) If the number is odd, then it is not divisible by 2;

B) The number 15 is odd; Therefore, the number 15 is not divisible by 2;

C) If the triangle is isosceles, then there are at least two equal sides in it; Triangle ABC - non-isosceles; Hence, there is not a single pair of equal sides in it;

D) If the number is divided by 3, then the sum of the digits in the record of this number is divided by 3; The number 32 is not divisible by 3; Then the sum of the digits in its record is not divisible by 3;

E) If the number is even, then it is divisible by 3; The number 13 is odd; Therefore, the number 13 is not divisible by 2.

3. Choose deductive reasoning:

A) All honors students of the I-course; Student II-course John - an excellent student; Therefore, John is an athlete;

B) Not all excellent students II-course athletes; Sophomore Jack is an excellent pupil; Consequently, he is not an athlete;

C) All honors students of the I-course; Sophomore Elen is not an excellent pupil; Therefore, Elen is not an athlete;

D) All honors students of the I-course; Sophomore Cohen is an athlete; Therefore, he is an excellent student.

E) Not all excellent students II-course athletes; Freshman Bob is an excellent pupil; Therefore, he is not an athlete.

IV. GRAPHS

IV.1. The concept and types of graphs

IV.1.1 Elements and methods for representing graphs

A graph is a dynamic network-connected data structure, represented by a set of pairs called *vertices* and *edges*. Each vertex can be connected with several vertices or with itself by means of edges and a vertex that does not form a hierarchy.

Formally, the graph is defined as the set of pairs $G = (V, E)$, where V is a set of vertices, E is a set of edges, in fact there is a relation on V i.e. $E \subseteq V \times V$.

The two vertices that are connected bty the edge of a graph are called the *boundary vertices* of this edge and are *adjacent* , so as $v_i \in V$ and $v_j \in V$ - boundary vertices ($i = \overline{1,n}; j = \overline{1,m}$), $(v_i, v_j) \in E$ are edges.

The edge whose boundary vertices coincide is called a *loop*, i.e. $(v_i, v_i) \in E$ - a loop. There is a single vertex in the loop, i.e. the edge originates from one vertex and directly enters this vertex back (see Figure IV.1.1.1).

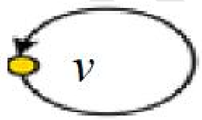


Figure IV.1.1.1. The loop.

The edges with the coinciding boundary vertices are called *fold*. For example, if 3 edges have the same boundary vertices, then they will be 3-fold edges (see Figure IV.1.1.2).

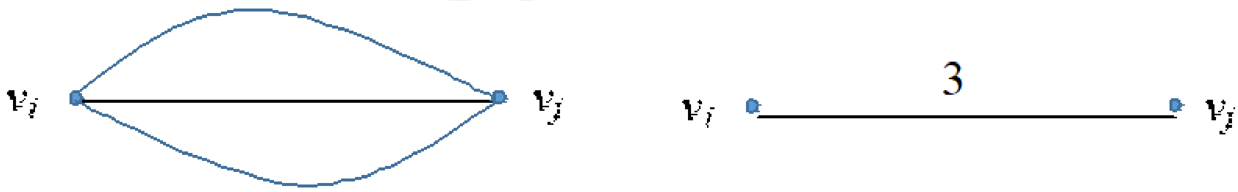


Figure IV.1.1.2. Multiple edges

A graph without loops and multiple edges is called a simple graph.

Vertices that do not have incident edges are called *isolated vertices*.

The number of edges incident to a vertex is called a *degree of* a *vertex*.

Vertices whose degrees are even numbers are called *even vertices*.

Vertices of odd degree are called *odd vertices*.

A vertex whose degree is 1 is called an *end vertex* (the degree of an isolated vertex is 0).

The sum of the degrees of all its vertices is twice the number of edges, i.e. is an even number. The number of odd vertices of any graph is even.

Depending on the properties of vertices and edges, and also on the type of relations between them, the graphs are divided into several types.

A graph consisting only of isolated vertices is called an empty or null-graph and is denoted by O_n, where n is the number of vertices of the graph.

If every two vertices in a graph are connected, then such a graph is called a *connected graph*.

If there is at least one pair of disconnected vertices in the graph, then the graph is called *disconnected*.

If in a graph the vertex sets V and edges E are finite, then it is called a *finite graph*.

A finite graph containing n vertices and m edges is called an $(n; m)$ graph.

If an equal number of edges originates from each vertex of the graph and an equal number of edges enter each vertex, then such a graph is called a *regular graph*.

If a graph does not contain loops and multiple edges, then such a graph is called a *simple graph*. A simple graph in which any two vertices are connected by an edge is called a *complete graph* and is denoted by: U_n, where n is the number of vertices

A graph without loops, but with multiple edges, is called a *multigraph*.

A graph that contains at least one loop is called a *pseudograph*. Multiple edges can be in the pseudograph.

If a direction is defined for each edge of the graph, then such a graph is called an *oriented graph*.

If all pairs of vertices of the graph are connected in both directions, then such a graph is a *strongly connected graph*.

If every edge of a graph has weight, then such a graph is called a *weighted graph*, i.e. We can define a function $w: E \to R$, where R is the set of real numbers, w is the weight of the edge, and $w \geq 0$.

If the set of vertices of a simple graph V admits such a partition into two disjoint subsets V_1 and V_2 (that is, $V_1 \cap V_2 = \emptyset$) that there are no edges connecting vertices of the same subset, then it is called *bigraph* or *bipartite graph*.

A graph is called *flat*, if its edges do not intersect at points other than the vertices of a given graph.

There could be a relationship between two graphs, including multidimensional ones. For example, for given two graphs, one can determine the ratio of their complete coincidence (isomorphism) or the inclusion relation (to be part) of one graph in another graph.

Two graphs are said to be *isomorphic*, if they have one and the same number of vertices, and for any 2 vertices of the first graph connected by an edge, the corresponding vertices of the second graph are also connected by an edge and back.

The graph $G' = (V', E')$ is called part of the graph $G = (V, E)$, if $V' \subset V$ and $E' \subset E$.

A part of a graph that does not contain isolated vertices is called a *subgraph*.

Part of the graph, which, along with some subset of the edges of the graph, contains all the vertices of the graph, is called a *sigraph*.

In graphs, you can solve the following problems: *finding the shortest path from one vertex to another*, *finding the number of closed paths*, etc. To do this, we introduce the following definitions:

A path in a graph is a sequence of edges leading from one vertex to another vertex, such that every two adjacent edges have a common vertex, and no edge occurs more than once, i.e. Formally the path in the graph is a sequence of vertices $(v_1, v_2, v_3, \ldots, v_{m-1}, v_m)$ such that the pairs $\{(v_1, v_2), (v_2, v_3), \ldots, (v_{m-1}, v_m)\}$ become edges. This path can be in both directions.

A path without repeating edges is called a *chain*, and a chain without repeating vertices is called a *simple* one.

A chain in which ending vertices coincide is called a *cycle*, and the cycle in which there are no repeated vertices except the terminal vertices, is called a *simple* one. If the path back enters the same vertex, then such path is called a *closure* (cycle), i.e. in the closure, the staring and ending vertices coincide. If the closure does not pass through one of the vertices of the graph more than once, then it is called a *simple closur*e.

The *length of the path* is the number of edges of this path. If the weights of the edge are their lengths, then the path length is calculated as:

$$w(v_1, v_2, v_3, \dots, v_{m-1}, v_m) = \sum_{i=1}^{m-1} w(v_i, v_{i+1})$$

The graph can be represented using an adjacency matrix. To do this, we introduce the ratio of the contiguity between the vertices.

Having a vertex set *V*, we can determine the adjacency relation by representing each edge as a pair of adjacent vertices, i.e.

$$e_k = (v_i, v_j) = (v_j, v_i)$$

The vertex set *V* together with the defined adjacency relation completely determines the graph *G*.

A graph *G* with the number of vertices *n* can be represented by an *adjacency matrix* by a square matrix *A* of size *n* × *n* whose rows and columns correspond to vertices of the graph v_i and v_j, its element a_{ij} is equal to the number of multiple edges connecting the vertices v_i and v_j, and $a_{ij} = 0$, if is not exist the edge going from the vertex v_i to vertex v_j. The adjacency matrix of an empty graph that does not contain a single edge consists of zeros. The adjacency matrix of a simple graph consists of zeros and ones whose main diagonal contains only zeros. Sometimes, a loop is counted as two edges, that is, the value of the diagonal element in this case is equal to twice the number of loops around the i-th vertex.

A graph can be represented by an *incidence matrix*.

The incidence matrix of graph G is a matrix of size *n* × *m*, in which the number of rows *n* corresponds to the number of vertices, and the number of columns of *m* is the number of edges.

Elements of this matrix are defined by the rule: the element (*i*; *j*) is equal to 1 if the vertex $v_i$is incident to the edge e_j, and is equal to 0 if v_i and e_j are not incident. The column corresponding to the edge $(v_i, v_j) \in E$ contains -1 in the line corresponding to the vertex v_j and 1, in the line corresponding to the vertex v_i. In all other cases 0.

Let two parts of the graph G be given: G_1 and G_2, then we can define:

1. The union of graphs is the graph $G = G_1 \cup G_2$, the sets of vertices and edges of which are defined: $V = V_1 \cup V_2$, $E = E_1 \cup E_2$.

2. The intersection of graphs is the graph $G = G_1 \cap G_2$_2, the sets of vertices and edges of which are defined: $V = V_1 \cap V_2$, $E = E_1 \cap E_2$.

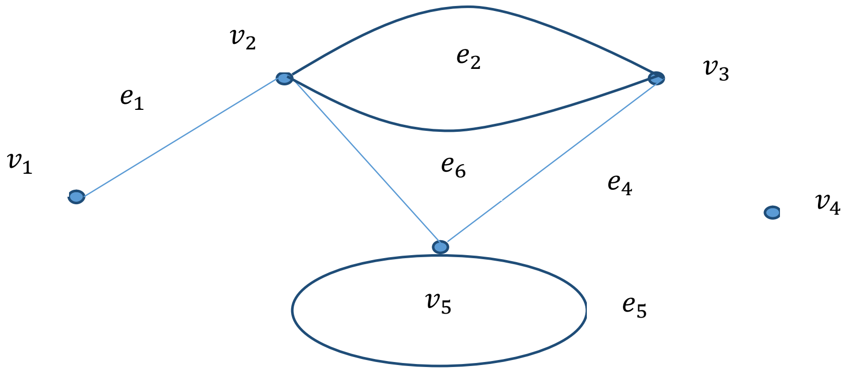
3. If $V_1 \cap V_2 = \emptyset$ and $E_1 \cap E_2 = \emptyset$, then the graphs are said to be disjoint.

3. The addition of two graphs G_1 and G_2 modulo 2 - this is the graph $G = G_1 \oplus G_2$, the set of vertices of which is $V = V_1 \cup V_2$, and the set of edges $E = (E_1 \cup E_2) \backslash (E_1 \cap E_2)$

If A is the adjacency matrix of the graph G, then the matrix A^n has the following property: the element in the i-th row, the j-th column is equal to the number of paths from the i-th vertex to the j-th, consisting of exactly n edges.

Examples IV.1.1.

1. We construct an adjacency matrix for the graph shown in the figure below.



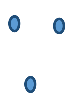
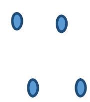
The adjacency matrix of this graph has the form:

	v_1	v_2	v_3	v_4	v_5
v_1	0	1	0	0	0
v_2	1	0	2	0	1
v_3	0	2	0	0	1
v_4	0	0	0	0	0
v_5	0	1	1	0	2

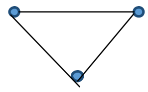
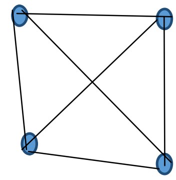
For the graph depicted in the previous example, the incidence matrix has the form:

	e_1	e_2	e_3	e_4	e_5	$\mathbf{e_6}$
v_1	1	0	0	0	0	0
v_2	1	1	1	0	0	1
v_3	0	1	1	1	0	0
v_4	0	0	0	0	0	0
v_5	0	0	0	1	2	1

2. Zero graphs:

$\boldsymbol{O_1}$ - $\boldsymbol{O_2}$ - $\boldsymbol{O_3}$ -  $\boldsymbol{O_4}$ - 

3. Complete graphs:

U_2 - U_3 -  U_4 - 

Task IV.1.1.

1. Construct a graph corresponding to a given adjacency matrix.

	v_1	v_2	v_3	v_4	v_5	v_6
v_1	1	0	1	0	0	0
v_2	0	0	0	1	1	0
v_3	0	0	1	0	0	2
v_4	2	0	1	0	0	0
v_5	0	0	0	1	0	1
v_6	0	1	0	0	1	0

2. Construct a graph corresponding to a given adjacency matrix.

	v_1	v_2	v_3	v_4	v_5	v_6
v_1	0	0	1	2	1	0
v_2	0	2	0	0	1	0
v_3	1	0	0	2	0	0
v_4	2	0	2	0	1	0
v_5	1	1	0	1	2	1
v_6	0	0	0	0	1	0

Questions IV.1.1.

1. Which edges are called fold?
2. What is a loop?
3. Which two vertices are said to be adjacent?
4. Which vertex is called isolated?
5. Explain what the incidence of vertices and edges means?

6. What is the difference between the adjacency matrix and the incidence matrix?

7. What is a path in the graph?

8. What is the difference between a zero graph and a complete graph?

9. What is a cycle?

10. How is a path length determined?

Tests IV.1.1.

1. What are the points of a graph called?

A) vertices of the graph;

B) points of the graph;

C) arcs of the graph;

D) nodes of the graph;

E) edges of the graph.

2. The graph is ...

A) the set of points, two of which are necessarily connected by lines;

B) the set of points that are never connected by lines;

C) only two points that are connected by lines;

D) the set of points that can be connected by lines;

E) the set of points that can be connected to each other;

3. The lines that connect the vertices are called ...

A) edges of the graph;

B) the sides of the graph;

C) vertices of the graph;

D) segments;

E) by a cycle.

V.1.2. Trees

A *tree* is a graph in which all vertices are connected, and the paths are not closed, i.e. connected graph without cycles and without loops.

In the tree, the vertices are divided into the following types:

1) The *root* is the vertex from which one or several edges emanate, but no one edge that enters, i.e. a vertex that does not have any ancestor, but can have many descendants;

2) A *branch* is a vertex in which one edge enters, but many edges can emanate from it, i.e. a vertex that has a single ancestor and can have many descendants;

3) A *sheet* is a vertex in which only one edge enters, but does not emanate any edge, i. e. a vertex that has a single ancestor, but does not have a single descendant.

In the tree, the direction of the path passes from the root through the branches to the leaves. Inside the tree there can be several trees, which we will call *subtrees*.

Now we can give the following recursive definition (with reference to itself):

1. *A recursive basis*: the set $\{v\}$ consisting of only one vertex v is a tree where its only vertex is both a root and a leaf.

2. *A recursive step*: if *v* is a vertex and $A_1, A_2, ..., A_n$ are trees, then we can construct a new tree in which the vertex *v* is the root, and the edges are the outgoing from this vertex and entering the tree roots $A_1, A_2, ..., A_n$.

3. *Recursive conclusion*: Trees are obtained only via rules 1 and 2.

From this definition it is clear that a tree is a hierarchical connected dynamic data structure, represented by a single root vertex and its descendants. The maximum number of descendants of each vertex determines the dimension of a tree.

The tree definition can be represented as follows in Figure IV.1.2.1:

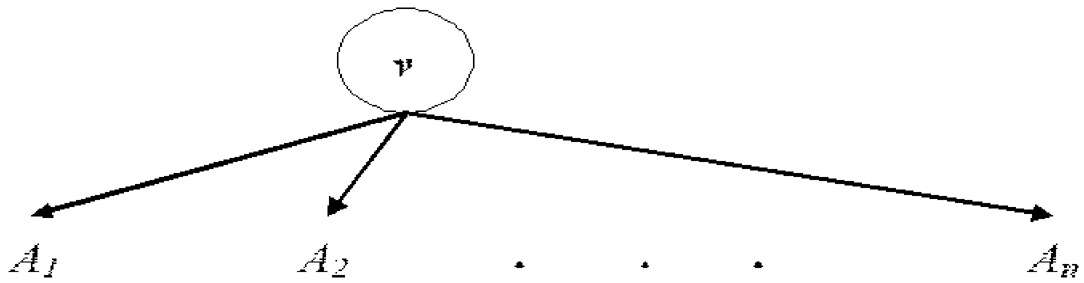


Figure IV.1.2.1. Definition of a tree.

There are especially distiguised trees among the trees that are called *binary trees*. They can be defined as follows:

A binary tree is a tree in which each vertex has at most two descendants. This vertex is called the parent vertex, and the descendants are called the *left heir* and the *right heir*. We give a recursive definition of a binary tree. A set of vertices is called a binary tree, if this set:

- either contains nothing (empty set);
- or consists of a root that connects to two binary trees, called a left-sided subtree and a right-side subtree.

Thus, a binary tree is either empty, or consists of data and two subtrees, each of which can be empty. If both subtrees are empty at some vertex, then it is a leaf. Formally, a binary tree is defined as follows:

<Bintree> :: = nil | (<Data> < Bintree > < Bintree >),
where nil is empty.

The trees solve the following tasks: *bypassing of trees*, *searching a tree*, *adding a new vertex to the tree*, *destroying the tree top*, *comparing trees*, etc.

Binary trees are used in search algorithms: each vertex of the binary search tree corresponds to an element from some sorted set, all its left descendants - to smaller elements, and all its right descendants - to large elements. Each vertex in the tree is uniquely identified by a sequence of non-repeating vertices from the root and up to it – which is a path. The path length is the level of the vertex in the tree hierarchy. For practical purposes, two subsets of binary trees are usually used: *a binary search tree* (BST) and *a binary heap*.

The binary search tree has the following properties:

- the left subtree and the right subtree are binary search trees;
- for all vertices of the left subtree of an arbitrary vertex *v*, the values of the data keys are less than the value of the data key of the vertex *v* itself;

- for all vertices of the right subtree of the same vertex *v*, the values of the data keys are greater than the value of the data key of the vertex *v*.

Obviously, the data at each vertex must have the keys on which the comparison operation is defined.

A binary heap or sorting tree has the following properties:

- the value at any vertex is not less than the values at the vertices of its descendants;

- the depth of the leaves (the distance to the root) differs by not more than one layer;

- the last layer is filled from left to right.

The heap of this type is called *max-heap*. There are also heaps where the value at any vertex, on the contrary, is not greater than the values of its descendants. Such heaps are called *min-heap*.

Examples IV.1.2.

1. A binary relation can be represented as an oriented graph as in figure IV.1.2.2, where the divisibility ratio over integers from 1 to 12 is shown: 2 and 3 is divided by 1; 4 and 6 is divided by 2; 6 is divided into 2 and 3; 12 is divided into 4 and 6.

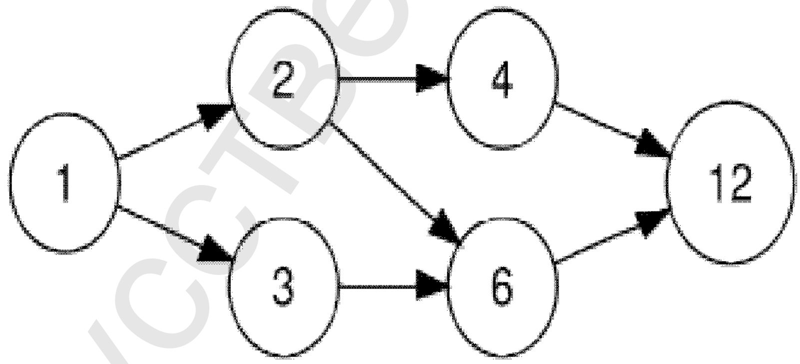


Figure IV.1.2.2. Representation of a binary relation

2. The representation of the binary tree is shown in Figure IV.1.2.3.

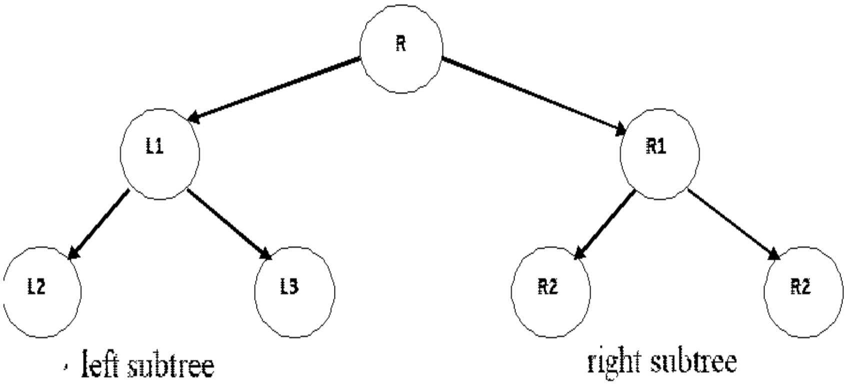


Figure IV.1.2.3. Binary tree.

3. Bypassing the binary tree of the arithmetic expression ((3 + 1) * 3 / (9-5) + 2- (3 * (7-4) +6) from below downwards and from left to right is shown in Figure IV.1.2.4.

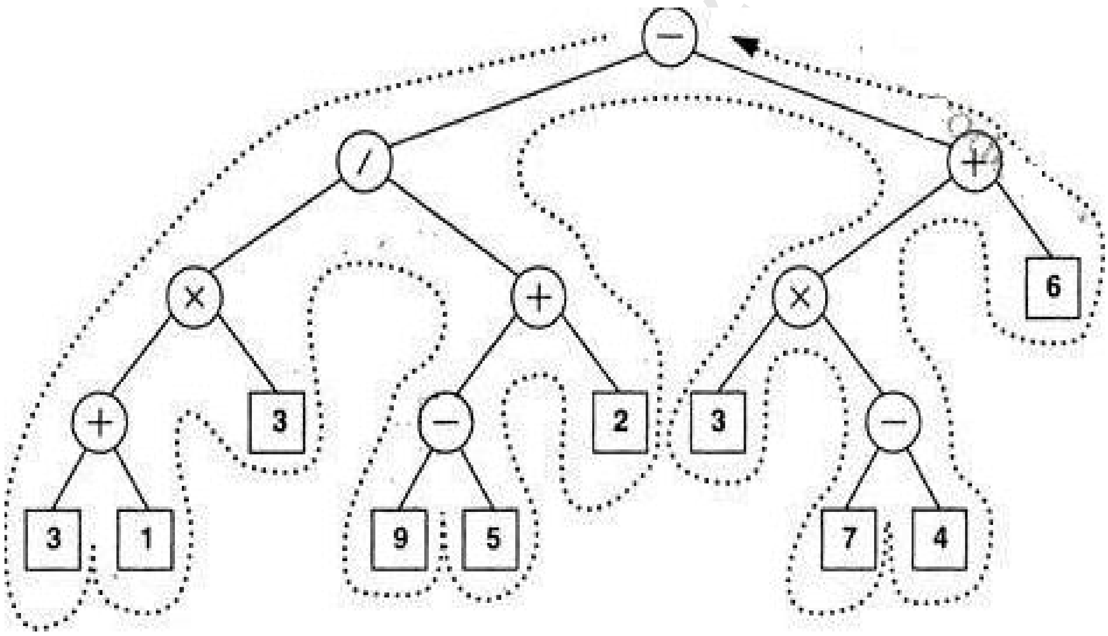


Figure IV.1.2.4. Tree traversal

Tasks IV.1.2.

1. Construct an oriented weighted graph to describe the structure of the identifier.

2. Build a tree for the expression ((*a* / (*b* + *c*)) + (*x* * (*y* - *z*))).

3. Determine the adjacency matrix *A* for the undirected graph, which is shown in Figure IV.1.2.5 and contains a loop around vertex 1 in which the application-specific element can be considered equal to either one (as shown below) or two.

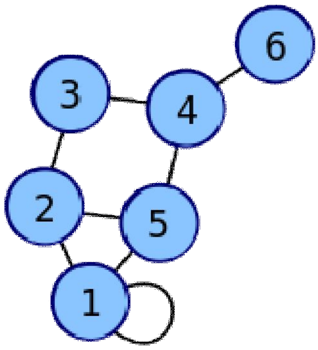


Figure IV.1.2.5. Unoriented graph.

4. On a finite set N = {1, 2, 3, 4, 5}, a relation

R = {(1,2), (1,4), (1,5), (2,3), (3,2), (3,4), (4,4), (4,5) , (5.3), (5.4)}.

For this relationship, write down the domain of definition and domain of values. Draw a graph of this relationship. Make for him an adjacency matrix and an incidence matrix.

Help:

1. Not derogating a generality, to facilitate the construction of the required graph, we will not consider letters, but only one letter and not digits, only one digit that will serve as the weights of the required weighted graph.

2. In the corresponding binary tree, the leaves are the operands, and the remaining vertices are the operations.

3. Matrix of adjacency

$$\begin{pmatrix} 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

Questions IV.1.2.

1. How is the path formed in the graph?
2. Which edges are called fold?
3. Which vertex is called isolated?
4. What is the degree of an isolated vertex?

5. What does the degree of the vertex mean?
6. What graph is called cyclic?
7. What is an incidence matrix?

Tests IV.1.2:

1. What are the types of graphs?
A) an oriented graph, an undirected graph;
B) an oriented graph, a certain graph;
C) a certain graph, an undirected graph;
D) a certain graph, an indefinite graph;
E) an indefinite graph, an undirected graph.

2. What is a tree?
A) a graph without loops and without cycles;
B) a graph without weights;
C) a graph without networks and cycles;
D) a weighted graph;
E) a oriented graph.

3. What is a binary tree?
A) a tree in which each vertex has at most two children;
B) a tree in which there are two vertices;
C) a tree in which there is no cycle;
D) a tree in which there is no loop;
E) a tree in which one vertex has no direct descendants.

V. LANGUAGES, GRAMMAR AND AUTOMATONS

V.1. Mechanisms of the generation of language

V.1.1. Basic concepts of formal languages

The basic concepts associated with formal languages are reviewed in this paragraph.

Definitions V.1.1.1.

1. *An alphabet* is a finite non-empty set of elements, called symbols (letters). Hereinafter, the alphabet will be denoted by uppercase (capital) Latin letters, and the elements – by lowercase (small) letters.

2. *A word (chain)* in a given alphabet is a finite sequence of elements of this alphabet. Hereinafter, the words will be denoted by lowercase Greek letters.

3. A word that does not contain any symbols is called *an empty word*, and is denoted by the Greek letter ε.

4. The length of the word ω is defined by the number of its symbols and is denoted by $|\omega|$. Each symbol is counted as many times as it occurs in ω. The length of the empty word is zero, $|\varepsilon|=0$.

5. If α and β are words in the alphabet T, then the word $\alpha \cdot \beta$ is called the *concatenation* of the words α and β, which is obtained by appending the word β to the end of the word α. In recordings the sign · is sually omitted and written simply $\alpha\beta$.

6. If α is a word and n≥0 are positive integers, then $\alpha^0 \rightleftharpoons \varepsilon$ и $\alpha^n \rightleftharpoons$ $\underbrace{\alpha \cdot \alpha \cdot \ldots \cdot \alpha}_{n}$ (the sign of $\rightleftharpoons$ is "equal by definition").

7. If τ and ω are words in the alphabet $\boldsymbol{T}$ and for some word ξ in $\boldsymbol{T}$ the equality $\omega = \tau\xi$ is fair, then the word τ is called the prefix (the beginning) of the word ω and is denoted by $\tau \Subset \omega$, i.e. $\tau \Subset \omega$ ∃ $\exists\xi\ (\omega = \tau\xi)$.

8. If τ and ω are words in the alphabet $\boldsymbol{T}$ and for some word ζ in $\boldsymbol{T}$ the equality $\omega = \zeta\tau$ is fair, then the word is called the postfix (end) of the word ω and is denoted by $\tau \Supset \omega$, i.e. $\tau \Supset \omega$ ∃ $\exists\zeta\ (\omega = \zeta\tau)$.

9. If τ and ω are words in the alphabet $\boldsymbol{T}$ and for some words ζ and ξ the equality $\omega = \zeta\tau\xi$ is fair, then the word τ is called the subword of the word ω and is denoted by $\tau\subseteq\omega$, i.e. $\tau\subseteq\omega \; \exists \; \exists\zeta \; \exists\xi \; (\omega = \zeta\tau\xi)$.

10. If τ and ω are words in the alphabet $\boldsymbol{T}$ and τ is a subword of the word ω, then $|\omega|_\tau$ denotes the number of occurrences of the word τ in the word ω.

Note V.1.1.1.

For any non-empty word ω in the alphabet $\boldsymbol{T}$ and empty word ε, the relations $\varepsilon\Subset\omega$ and $\varepsilon\Supset\omega$ are satisfied.

If $\boldsymbol{T}$ is an alphabet, then the set of all chains (words) of finite length in this alphabet is defined as

$$\boldsymbol{T}^* \rightleftharpoons \bigcup_{k=0}^{\infty} T^{k},$$

Where $\boldsymbol{T}^0 \rightleftharpoons \{\varepsilon\}$ is the set of chains of length 0, $\boldsymbol{T}^k \rightleftharpoons \underbrace{\boldsymbol{T} \cdot \boldsymbol{T} \cdot \ldots \cdot \boldsymbol{T}}_{\boldsymbol{k}}$ is the set of chains of length k, and $k \geq 1$ are natural numbers.

We denote by $\boldsymbol{T}^{+}$ the set of all possible non-empty chains, i.e. $\boldsymbol{T}^{+} = \boldsymbol{T}^{*} \setminus \{\varepsilon\}$. For example, if $\boldsymbol{T} = \{a\}$, then the set $\boldsymbol{T}^{+}$ is defined as $\boldsymbol{T}^{+} = \{a, aa, aaa, \ldots\}$.

Obviously, not all chains from the set $\boldsymbol{T}^{+}$ can be meaningful units (words, phrases, phrases, sentences and texts) of some language. Note that meaningful units of the language can only be those chains that satisfy grammatical rules and have semantic meanings. Therefore, any concrete language $\boldsymbol{T}^{+}$ is a proper subset of the set $\boldsymbol{T}^{+}$; $\boldsymbol{L} \subset \boldsymbol{T}^{+}$. For example, if you take Kazakh letters and various special signs as elements of the alphabet $\boldsymbol{T}$, then the Kazakh language will be a proper subset of the set $\boldsymbol{T}^{+}$, which contains only words, word combination, phrases, phrases, sentences and texts meaningful in the Kazakh language.

Since each language is a set of chains of finite length in a given alphabet, we can consider operations of union, intersection, difference, complement, direct product, symmetric difference, concatenation and iteration of languages given over the same alphabet.

Definitions V.1.1.2. Suppose that two languages $\boldsymbol{L_1}$ and $\boldsymbol{L_2}$ are given in the alphabet $\boldsymbol{T}$, that is, $\boldsymbol{L_1} \subseteq \boldsymbol{T}^+$ and $\boldsymbol{L_2} \subseteq \boldsymbol{T}^+$, and also $\boldsymbol{U}$ is a universe, then we can define:

1. The union: $\boldsymbol{L_1} \cup \boldsymbol{L_2} \rightleftharpoons \{x:\ x \in \boldsymbol{L_1} \vee x \in \boldsymbol{L_2}\}$;
2. Intersection: $\boldsymbol{L_1} \cap \boldsymbol{L_2} \rightleftharpoons \{x:\ x \in \boldsymbol{L_1}\ \&\ x \in \boldsymbol{L_2}\}$;
3. Difference: $\boldsymbol{L_1} \setminus \boldsymbol{L_2} \rightleftharpoons \{x:\ x \in \boldsymbol{L_1}\ \&\ x \notin \boldsymbol{L_2}\}$;
4. Addition: $\overline{L} \rightleftharpoons \{x:\ x \in \boldsymbol{U}\ \&\ x \notin \boldsymbol{L}\}$;
5. The direct product: : $\boldsymbol{L_1} \times \boldsymbol{L_2} \rightleftharpoons \{(a, b):\ a \in \boldsymbol{L_1}\ \&\ b \in \boldsymbol{L_2}\}$;
6. Symmetrical difference: $\boldsymbol{L_1} \triangle \boldsymbol{L_2} \rightleftharpoons \{x:\ x \in (\boldsymbol{L_1} \setminus \boldsymbol{L_2}) \vee x \in (\boldsymbol{L_2} \setminus \boldsymbol{L_1})\}$;
7. Concatenation: $\boldsymbol{L_1} \cdot \boldsymbol{L_2} \rightleftharpoons \{a \cdot b:\ a \in \boldsymbol{L_1}\ \&\ b \in \boldsymbol{L_2}\}$;
8. Iteration - Kleene star: $\boldsymbol{L}^* \rightleftharpoons \bigcup_{k=0}^{\infty} L^k$,

where $\boldsymbol{L}^0 \rightleftharpoons \{\varepsilon\}$, $\boldsymbol{L}^k \rightleftharpoons \underbrace{L \cdot L \cdot \ldots \cdot L}_{k}$, $k \geq 1$.

Notes V.1.1.2.

1. The set of words of finite length in the alphabet $\boldsymbol{T}$ is a partially ordered set with the relation $\leq$ ($\geq$) and all its chains are comparable in length with respect to $\leq$ ($\geq$).

2. The set of words of finite length in the alphabet $\boldsymbol{T}$ is a partially ordered set with the relation $\subseteq$ ($\supseteq$) and all subsets of words in the alphabet $\boldsymbol{T}$ are comparable with respect to $\subseteq$ ($\supseteq$).

Definitions V.1.1.3. Let $L \subseteq \boldsymbol{T}^*$, then we can introduce the following notations and concepts:

1. $\boldsymbol{L}^R$ - the operation of the language *reference* $\boldsymbol{L}$ is defined as $\boldsymbol{L}^R = \{\tau^R:\ \tau \in \boldsymbol{L}\}$.

2. $\boldsymbol{P}$ref($\boldsymbol{L}$) - the set of prefixes of $\boldsymbol{L}$ is defined as $\boldsymbol{P}\mathrm{ref}(\boldsymbol{L}) \rightleftharpoons \{\tau:\ \exists\xi(\xi \in \boldsymbol{L}\ \&\ \tau \Subset \xi)\}$, where $\Subset$ is the prefix relation.

3. Suf ($\boldsymbol{L}$) - $\{\tau:\ \exists\zeta\ (\zeta \in \boldsymbol{L}\ \&\ \tau \Supset \zeta)\}$, where $\Supset$ is the postfix ratio.

4. Substr ($\boldsymbol{L}$) - the set of all subwords (subchains) of the language $\boldsymbol{L}$ is defined as Substr ($\boldsymbol{L}$) $\rightleftharpoons \{\tau:\ \forall\xi\ (\xi \in \boldsymbol{L}\ \&\ \xi \neq \varepsilon \Rightarrow \tau \subseteq \xi)\}$.

5. A function $\boldsymbol{f}$: $\boldsymbol{K} \to \boldsymbol{L}$ is called a *bijection* if every element of the set $\boldsymbol{L}$ is the image of exactly one element of the set $\boldsymbol{K}$ with respect to the function $\boldsymbol{f}$.

3. The sets $\boldsymbol{K}$ and $\boldsymbol{L}$ are said to be *equipotent* if there is a bijection from $\boldsymbol{K}$ to $\boldsymbol{L}$.

Examples V.1.1. Let the alphabet $\boldsymbol{T} = \{a, b, c, d\}$ be given. Then:

1) $|a| = 1$, $|bb| = 2$, $|ccc| = 3$, $|abcd| = 4$.

2) If $\alpha = ccc$ and $\beta = dddd$, then $\alpha\beta = cccdddd$.

3) ab2 = abb, (ab) 3 = $ababab$.

4) If $\alpha = ab$, $\beta = abcd$, $\xi = cd$, then $\beta = \alpha\xi$, i.e. $\alpha \Subset \beta$

5) $\varepsilon \Subset abcd$, $a \Subset abcd$, $ab \Subset abcd$, $abc \Subset abcd$, $abcd \Subset abcd$.

6) $abcd \Supset \varepsilon$, $abcd \Supset d$, $abcd \Supset cd$, $abcd \Supset bcd$, $abcd \Supset abcd$.

7) If $\tau = ca$ and $\xi = da$, then $\tau \Subset \omega$ and $\omega = cada$.

8) If $\tau = bbb$, $\zeta = aaa$ and $\xi = ccc$, then $\tau \subseteq \omega$ and $\omega = aaabbbccc$.

9) If $\omega = abcdabcd$ and $\tau =$ cd, then $\tau \subseteq \omega$ and $|\omega|_\tau = 2$.

Tasks V.1.1:

1. Let $\boldsymbol{T} = \{a, b\}$ and $\boldsymbol{L} = \{aa, ab\}$ be given. Find $\boldsymbol{L}^3$.

2. List the words of the language $\boldsymbol{L_1} \cap \boldsymbol{L_2}$, where $\boldsymbol{L_1} = \{(ab)\ n\text{: } n \geq 0\}$ and $\boldsymbol{L_2} = \{ a^m b^m\text{: } m \geq 0\}$.

3. Let $\boldsymbol{T} = \{a, b, c\}$. $\boldsymbol{L_1} = \{\tau \in \boldsymbol{T}\text{: } |\tau| = 4\}$ and $\boldsymbol{L_2} = \{\tau \in \boldsymbol{T}^*\text{: } |\tau|_c = 1\}$. Calculate the number of chains of the language $\boldsymbol{L_1} \setminus \boldsymbol{L_2}$.

4. Let the languages $\boldsymbol{L_1}$ and $\boldsymbol{L_2}$ be given. Find the equivalence of languages

$\boldsymbol{L}_1 \cdot (\boldsymbol{L}_2^{\boldsymbol{R}})$ and $\boldsymbol{L}_1^{\boldsymbol{R}} \cdot \boldsymbol{L}_2$.

5. For $\boldsymbol{L_1}$ and $\boldsymbol{L_2}$, determine the result of their concatenation and join:

$\boldsymbol{L_1} = \{d, de, dee\}$ and $\boldsymbol{L_2} = \{\varepsilon, d, e, de, d\}$;

$\boldsymbol{L_1} = \{\varepsilon, d, de, dee\}$ and $\boldsymbol{L_2} = \{d, e, dee, d\}$;

$\boldsymbol{L_1} = \{\varepsilon, d, e, de, ded\}$ and $\boldsymbol{L_2} = \{\varepsilon, d, e, de, ed\}$;

$\boldsymbol{L_1} = \{d, e, dd, de, ee, ded\}$ and $\boldsymbol{L_2} = \{d, e, dd, de, ed\}$.

$\boldsymbol{L_1} = \{d, e, ded, ddde, eedd\}$ and $\boldsymbol{L_2} = \{d, e, ddd, ded, eee\}$.

6. Let $\boldsymbol{L} = \{abcd, ad\}$ be a language in the alphabet $\{a, b, c, d\}$. Find the set of all subwords of this language Substr ($\boldsymbol{L}$).

7. Let $\boldsymbol{L} = \{ a^k b^m c^n\text{: } k \leq m \leq n\}$ be a language in the alphabet $\{a, b, c\}$. Find the set of all subwords of this language Substr ($\boldsymbol{L}$).

Questions V.1.1:

1. Is there such a language $\boldsymbol{L}$ that

$\boldsymbol{L}^* \neq \{x^n: \ x \in \boldsymbol{L}, n \geq 0\}$?

3. Is there such a language $\boldsymbol{L}$ that

$(\boldsymbol{L}^R)^* \neq (\boldsymbol{L}^*)^R$?

3. Let the alphabet $\boldsymbol{T} = \{a, b, c, d\}$ be given and the language $\boldsymbol{L} = \{\tau \in \boldsymbol{T}^*: |\tau|_a = 1 \ \& \ |\tau|_b = 1\}$. Is it true that $abcdcacdcabbacba \in \boldsymbol{L}^*$?

Tests V.1.1:

1. Let the languages $\boldsymbol{L_1} = \{\text{aa, bb}\}$, $\boldsymbol{L_2} = \{\text{cc, dd}\}$ be given, then what will the concatenation of these languages be $\boldsymbol{L_1} \cdot \boldsymbol{L_2}$?

A) $\boldsymbol{L_1} \cdot \boldsymbol{L_2} = \{aacc, aadd, bbcc, bbdd\}$.

B) $\boldsymbol{L_1} \cdot \boldsymbol{L_2} = \{aaaa, dddd, bbbb, cccc\}$.

C) $\boldsymbol{L_1} \cdot \boldsymbol{L_2} = \{aabb\ aacc, bbaa, ddaa\}$.

D) $\boldsymbol{L_1} \cdot \boldsymbol{L_2} = \{aadd, bbdd, ccdd, dddd\}$.

E) $\boldsymbol{L_1} \cdot \boldsymbol{L_2} = \{aacc, aabb, aadd, aacc\}$.

2. Let the language $\boldsymbol{L} = \{\text{amban: m} \leq \text{n}\}$ be a language in the alphabet {a, b}, then which contains the set of all its sub-chains?

A) chains *b*, *ba*, *aba*, *baa*, *abaa*, *baaa*, *aabaa*, *abaaa*, and so on.

B) the chains *a*, *ab*, *bab*, *abb*, *abab*, *aaaa*, *aaaab*, *abbaa*, and so on.

C) the chains *aa*, *ab*, *ba*, *aaa*, *aabb*, *bdaa*, *aabba*, *abbba*, and so on.

D) chains *b*, *bb*, *ba*, *baba*, *abba*, *bbaa*, *abbaa*, *abbba*, *and so on*.

E) chains *b*, *aa*, *ab*, *baab*, *abba*, *baab*, *abbba*, *abbbb*, and so on.

3. Let the language $\boldsymbol{L} = \{ab^n: \ n \geq 0\}$ be given in the alphabet $\{a, b\}$, then which contains the set of all its sub-chains?

A) chains *a*, *ab*, *abb*, *abbb*, *abbbb*, *abbbbb*, and so on.

B) chains *a*, *ab*, *abb*, *abbb*, *abbbb*, *abbbbb*, and so on.

C) chains *a*, *ab*, *abb*, *abbb*, *abbbb*, *abbbbb*, and so on.

D) chains *a*, *ab*, *abb*, *abbb*, *abbbb*, *abbbbb*, and so on.

E) chains *a*, *ab*, *abb*, *abbb*, *abbbb*, *abbbbb*, and so on.

V.1.2. Formal grammars

Formal grammar is an important class of mechanisms for generating languages. The American linguist Chomsky firstly introduced formal grammar in 1959.

In the formal grammar generating the language ***L***, two disjoint sets of symbols are used:

1) a finite set of terminals that are constant values of ***T***, from which chains of the language ***L*** are formed;

2) the finite set of nonterminals variables ***N*** that do not intersect with the set ***T*** and denote grammatical concepts, categories, etc. Language of ***L***.

3) The process of generating chains of language ***L*** is described by a finite set of rules of substitution (rewriting rule) P, each of which consists of pairs of chains (α, β). In such a pair, the first component α is a chain containing at least one nonterminal, and the second component can be any chain formed from terminal and / or nonterminal symbols. It can be an empty chain.

Agreements V.1.2. The following agreements are accepted:

(1) lowercase Latin italic letters a, b, ..., z and the Arabic numerals 0, 1, ..., 9 denote terminals;

(2) uppercase Latin italic letters A, B, …, X, Y, Z denote nonterminals, where S denotes the initial nonterminal symbol;

(3) the lowercase Greek letters α, β,..., ω denote chains that can contain both terminals and nonterminals; here ε is an empty chain;

(4) the substitution rule, which is a pair of chains (α, β) from the set P, can be written as $\alpha \rightarrow \beta$;

(6) rules of the form $\alpha \rightarrow$ ε are called ε (epsilon) -rules;

(7) these agreements also apply to letters with lower and upper indices;

(8) rules of the form $\alpha_1 \mid \alpha_2 \mid \dots \mid \alpha_m \rightarrow \beta$ is an abbreviation of m rules of the form $\alpha_1 \rightarrow \beta$, $\alpha_2 \rightarrow \beta$, …, $\alpha_m \rightarrow \beta$ or rules written in the column as follows:

$$
\begin{gathered}
\alpha_1 \rightarrow \beta \\
\alpha_2 \rightarrow \beta \\
\dots \\
\alpha_m \rightarrow \beta
\end{gathered}
$$

(9) rules of the form $\alpha\to\beta_1 \mid \beta_2 \mid \ldots \mid \beta_n$ is an abbreviation of n rules of the form $\alpha \to \beta 1$, $\alpha \to \beta 2$, ..., $\alpha \to \beta$n or:

$$\alpha\to\beta_1$$
$$\alpha\to\beta_2$$
$$\ldots$$
$$\alpha\to\beta_n$$

(10) a rule of the form $\alpha_1 \mid \alpha_2 \mid \ldots \mid \alpha_{\mathrm{m}}\to\beta_1 \mid \beta_2 \mid \ldots \mid \beta_n$ is an abbreviation of $\boldsymbol{m} \times \boldsymbol{n}$ rules obtained from the agreement (6) and (7).

Definition V.1.2.1. The formal quadrangle G = <***T***, ***N***, ***P***, S> is called the formal grammar, where:

T is a non-empty finite set of terminal symbols (terminals);

N is a non-empty finite set of nonterminal symbols (nonterminals), where $\boldsymbol{T} \cap \boldsymbol{N} = \varnothing$, t is an empty set;

P is a non-empty finite set, the substitution rule is of the form $\alpha \to \beta$, where $\alpha \in (\boldsymbol{T} \cup \boldsymbol{N})^* \times \boldsymbol{N} \times (\boldsymbol{T} \cup \boldsymbol{N})^*$, $\beta \in (\boldsymbol{T} \cup \boldsymbol{N})^*$, that is,

$\boldsymbol{P} \subseteq \{(\alpha,\beta)\colon \alpha \in (\boldsymbol{T} \cup \boldsymbol{N})^* \times \boldsymbol{N} \times (\boldsymbol{T} \cup \boldsymbol{N})^* \;\&\; \beta \in (\boldsymbol{T} \cup \boldsymbol{N})^*\}$;

S - initial nonterminal, $S \in \boldsymbol{N}$.

The rules for deriving grammar can be considered as elementary operations, which, when applied in a certain sequence to the original chain, generate only right chains. The sequence of rules used in the process of generating a certain chain is its derivation.

A language defined by a (generated) grammar is a set of finite chains that consist only of terminals. All these terminal chains are derived starting with one particular chain consisting of only one initial nonterminal S.

To define a language using grammar, the concept of a *deducible chain* and the *relation of direct derivability* are applied.

Definitions V.1.2.2.

1. In the grammar ***G*** = <***T***, ***N***, ***P***, *S*> the resulting chains are recursively defined as follows:

1) S - deducible chain of grammar ***G***;

2) If $\alpha\beta\gamma$ is a deducible grammar chain of ***G*** and there is a rule $\beta\to\sigma$ in ***P***, then $\alpha\sigma\gamma$ is also a deducible grammar chain ***G***.

2. A derivable chain of ***G*** that does not contain nonterminal symbols from ***N*** is called a terminal chain generated by the grammar ***G***.

3. If $\alpha = \gamma\xi\delta$, $\beta = \gamma\eta\delta$ and $\alpha \to \beta$, $\xi \to \eta$ are the rules for deriving the grammar ***G*** then we say that there is an established direct derivability between the chains α and β. This means that in the grammar ***G*** the chain β is directly deduced from chains α via replacing ξ by η and denoting this relation via $\alpha \Rightarrow_G \beta$. If the grammar is known in advance, the index ***G*** with respect to the direct derivability is omitted and this relation is written as $\alpha \Rightarrow \beta$.

A record of the form $\alpha \Rightarrow^k \beta$ is the kth power of the ratio $\alpha \Rightarrow \beta$, if there are k + 1 chains α_0, α_1, ...,α_k such as $\alpha = \alpha_0$, $\alpha_k = \beta$ и $\alpha_{i-1} \Rightarrow \alpha_i$ $(1 \le i \le k)$. This sequence of chains is called the derivation of the length k of the chain β from the chain α in the grammar ***G***.

This sequence of chains is called the derivation of the length k of the chain β from the chain α in the grammar ***G***.

If there exists $i \ge 1$ (or $i \ge 0$) the relation $\alpha \Rightarrow^i \beta$ is satisfied, then it is written as $\alpha \Rightarrow^+ \beta$ (or $\alpha \Rightarrow^* \beta$). Here $\Rightarrow^+$denotes the transitive closure of the relation $\Rightarrow$, and by $\Rightarrow^*$ the reflexive and transitive closure of the relation $\Rightarrow$. In this case, a record of the form $\varphi \Rightarrow^+ \psi$ ($\varphi \Rightarrow^* \psi$) is read as: "$\psi$ is derivable from φ in a nontrivial way" ("ψ is deducible from φ").

Note V.1.2.: $\alpha \Rightarrow^* \beta$ if and only if $\alpha \Rightarrow^i \beta$ for some $i \ge 0$, and $\alpha \Rightarrow^+ \beta$ if and only if $\alpha \Rightarrow^i \beta$ for some $i \ge 1$.

Definitions V.1.2.3.

1. Each chain, which is derived from the initial nonterminal of the grammar, is called the sentential form.

2. Derived chains that do not contain nonterminal symbols are called terminal chains. Therefore, the language ***L***(***G***) can be defined as the set of terminal chains deducible in the grammar ***G***.

3. The language ***L***(***G***) generated by the grammar ***G*** is the set of terminal chains that are derived from one initial nonterminal S by applying a substitution rule from the set ***P***, that formally is written as

$$\boldsymbol{L}(\boldsymbol{G}) \rightleftharpoons \{\tau:\ \tau \in \boldsymbol{T}^*,\ S \Rightarrow^* \tau\}.$$

This means that any chain $\boldsymbol{L}(\boldsymbol{G})$ belonging to the language is a sentential form.

Examples V.1.2.

1. Let the parent algebraic expression in the infix record be generated using the grammar $\boldsymbol{G}$ = <$\boldsymbol{T}$, $\boldsymbol{N}$, $\boldsymbol{P}$, S>, where

$\boldsymbol{T} = \{+, -, /, *, (,), a\}$, $\boldsymbol{N} = \{S, E, T, F\}$,

$\boldsymbol{P} = \{S \to E,\ E \to E+T \mid E-T \mid T,\ T \to T*F|T/F|F,\ F \to a|(E)\}$.

2. Grammar with rules $\boldsymbol{P_1} = \{S \to 01S,\ S \to 0\}$ and a grammar with rules $\boldsymbol{P_2} = \{S \to 0A,\ A \to 10A,\ A \to \varepsilon\}$ are equivalent.

3. Two grammars for generating algebraic expressions formed by the operands i, n and operations +, * with the same terminal symbols $\boldsymbol{T}$ = {i, n, (,), +, *} and nonterminal symbols $\boldsymbol{N}$ = {S, F, H}, but with different rules:

P1 = {S → S + F, S → F, F → F * H, F → H, F → H, H → i, H → n, H → (S)} and

$\boldsymbol{P_1} = \{S \to S+F,\ S \to F,\ F \to F*H,\ F \to H, F \to H,\ H \to i,\ H \to n,\ H \to (S)\}$are equivalent.

Tasks V.1.2:

1. Construct all sentential forms for grammar with rules:

$S \to A+B|B+A,\ A \to a,\ B \to b$

2. Construct the output of a given chain a-$b * a + b$ for a grammar with rules:

$$S \to K|F+S|K-S,\ K \to F|F*K,\ F \to a|b.$$

3. Construct the output of a given chain *aaabbbcc*c for a grammar with rules:

$S \to aSBC|abC,\ CB \to BC,\ bB \to bb,\ bC \to bc,\ cC \to cc.$

4. Describe the language generated by grammar

$S \to FF,\ F \to aFb,\ F \to ab.$

5. Describe the language generated by grammar

$S \to Sc,\ S \to A,\ A \to aAb,\ A \to \varepsilon.$

6. Describe the language generated by grammar

$$S \to \varepsilon,\ S \to a,\ S \to b,\ S \to aSa,\ S \to bSb.$$

V.1. Describe the language generated by grammar

$S \to SA$, $SAA \to ASb$, $ASA \to b$, $A \to a$.

8. Describe the language generated by grammar

$S \to aSA$, $S \to abc$, $bA \to bbc$, $cA \to Aa$.

9. Describe the language generated by grammar

$S \to aAS$, $S \to B$, $Aa \to aaA$, $AB \to B$, $B \to a$.

Questions V.1.2.

1. What is a chain prefix?
2. What is a postfix chain?
3. If $\omega = abcdabefabhgabik$ and $\tau = ab$, then what is $|\omega|_\tau$?
4. How is the set of all chains determined?
5. What is a language in a given alphabet?
6. How is it determined?
7. How is $\boldsymbol{L_1} \setminus \boldsymbol{L_2}$ defined?
8. How is $\boldsymbol{L_1} \triangle \boldsymbol{L_2}$ defined?
9. How is $\boldsymbol{L_1} \cdot \boldsymbol{L_2}$ defined?
10. How is $\boldsymbol{L}^+$
11. What is $\boldsymbol{L}^0$?
12. What is $\boldsymbol{L}^k$?
13. How is the $\boldsymbol{L}^R$ language reference constructed?
14. Does there exist a language $\boldsymbol{L}$ such that $(\boldsymbol{L}^{\boldsymbol{R}})^* \neq (\boldsymbol{L}^*)^R$?
15. How is the prefix Pref ($\boldsymbol{L}$) of the language defined?
16. How is the postfix Suf ($\boldsymbol{L}$) of the language defined?
17. How is Substr ($\boldsymbol{L}$) defined?

Tests V.1.2.

1.How is the alphabet of formal languages correctly defined?

A) some finite set of symbols;

B) any finite sequence of symbols;

C) a subset of chains of finite length;

D) a set of tokens that can represent a particular token in the source program

E) the set of all comparison operators

2. How to correctly define chains in the alphabet V?

A) some finite set of symbols;

B) any finite sequence of symbols of the alphabet V;

C) a subset of chains of finite length of the alphabet V;

D) a set of tokens that can represent a particular token in the source program;

E) the set of all comparison operators.

3. What is denoted by $|\alpha|$?

A) length of the alphabet line;

B) a finite set of symbols;

C) a sequence of symbols;

D) the language of the alphabet V;

E) the length of the chain.

V.1.3. Regular sets and regular expressions

Let $\varnothing$ be an empty set, $\{\varepsilon\}$ be the set of empty chains, ***T*** a finite alphabet, and t a symbol from ***T***, that is, $\boldsymbol{T} \in \boldsymbol{T}$. Then we can give the following definition.

Definition V.1.3.1. The regular set in the alphabet ***T*** is defined recursively as follows:

Recursion basis:

(1) *t* is a regular set in the alphabet ***T***;

(2) $\{\varepsilon\}$ is a regular set in the alphabet ***T***;

(3) $\{t\}$ is a regular set in the alphabet ***T***;

Recursive extension: If ***P*** and ***Q*** are regular sets in the alphabet ***T***, then the regular ones are:

(4) $\boldsymbol{P} \cdot \boldsymbol{Q}$ is the concatenation of the sets ***P*** and ***Q***;

(5) $\boldsymbol{P} \cup \boldsymbol{Q}$ is the union of the sets ***P*** and ***Q***;

(6) $\boldsymbol{P}$ * is the iteration of the set ***P***;

Recursive conclusion:

The regular set in the alphabet ***T*** is determined only by the rules (1) - (6).

Thus, a set in the alphabet ***T*** is regular if and only if it is either t, or $\{\varepsilon\}$, or $\{t\}$ for some $t \in \boldsymbol{T}$, or it can be obtained from these sets by applying a finite number of join, concatenation, and iteration operations.

Definition V.1.3.2. Regular expressions describing regular sets in a finite alphabet ***T*** are defined recursively as follows:

Recursion basis:

(1) ϕ is a regular expression that defines a regular set t in the alphabet ***T***;

(2) ε is a regular expression that defines a regular set $\{\varepsilon\}$ in the alphabet ***T***,

(3) if $t \in \boldsymbol{T}$, then t is a regular expression that specifies

A regular set $\{t\}$ in the alphabet ***T***;

Recursive extension:

If p and *q* are regular expressions denoting regular sets ***P*** and ***Q*** in the alphabet ***T*** respectively, then:

(4) $p \cdot q$ is a regular expression defining $\boldsymbol{P} \cdot \boldsymbol{Q}$ in the alphabet ***T***;

(5) $p \vee q$ is a regular expression that specifies $\boldsymbol{P} \cup \boldsymbol{Q}$ in the alphabet ***T***;

(6) p^* is a regular expression defining $\boldsymbol{P}$ * in the alphabet ***T***;

Recursive conclusion:

Regular expressions are defined only by rules (1) - (6).

To build a regular expression, the operation concatenation ·, operation disjunction ∨ and operation iteration * were used. The highest priority is given by the operation *, then goes to ·, and the operation ∨ is the last. Usually, the sign of the operation · is omitted in the record of the regular expression.

We will use the notation p^+ for the abbreviated designation of the expression *pp** and, in addition, we will remove the excess brackets from the regular expressions where this can not lead to misunderstandings. For example, the regular expression 0∨10 * means (0∨ (1 (0 *))).

For each regular set, you can find at least one regular expression that specifies this set, and conversely, for each regular expression, you can construct a regular set, which is denoted by this expression.

Note that for every regular set there are infinitely many regular expressions that designate it.

We say that two regular expressions are equal if they denote the same set. For an equivalent transformation there are algebraic laws.

In the practical description of lexical structures, it is useful to match certain names to regular expressions, and to refer to them by these names. To determine such names, we will use a record of the form

$d_1 = r_1$

$d_2 = r_2$

...

$d_n = r_n$

where d_i are different names, and each r_i is a regular expression over the symbols $\boldsymbol{T} \cup \{d_1, d_2, ..., d_{i-1}\}$; Symbols of the main alphabet and previously defined symbol-names. Thus, for any ri, we can construct a regular expression over $\boldsymbol{T}$, repeatedly replacing the regular expression names with the regular expressions they designate.

If α, β, γ are regular expressions other than φ and ε, then:

(1) $\phi \cdot \alpha = \alpha \cdot \phi = \phi$

(2) $\varepsilon \cdot \alpha = \alpha \cdot \varepsilon = \alpha$

(3) $\alpha\cdot\beta\neq\beta\cdot\alpha$

(4) $\phi\vee\alpha=\alpha\vee\phi=\alpha$

(5) $\alpha\vee\alpha=\alpha$

(6) $\alpha\vee\beta=\beta\vee\alpha$

(7) $\alpha\cdot(\beta\cdot\gamma)=(\alpha\cdot\beta)\cdot\gamma=\alpha\cdot\beta\cdot\gamma$

(8) $\alpha\vee(\beta\vee\gamma)=(\alpha\vee\beta)\vee\gamma=\alpha\vee\beta\vee\gamma$

(9) $\alpha\cdot(\beta\vee\gamma)=\alpha\cdot\beta\vee\alpha\cdot\gamma$

(10) $(\alpha\vee\beta)\cdot\gamma=\alpha\cdot\gamma\vee\beta\cdot\gamma$

(11) $\varepsilon^* = \varepsilon$

(12) $\phi^* = \varepsilon$

(13) $\alpha^*=\alpha\vee\alpha^*$

(14) $(\alpha^*)^*=\alpha^*$

Examples V.1.3.

1. The regular expression $(0\vee1)*011$ defines a regular set of all chains consisting of 0 and 1, ending in the chain 011.

2. The regular expression $a(a\vee 0)*$ defines a regular set of all chains from $\{0, a\}*$ starting with a.

3. The regular expression $(a\vee b)(a\vee b\vee 0\vee 1)*$ defines a regular set of all chains from $\{0, 1, a, b\}*$ starting with *a* or *b*.

4. The regular expression $((00\vee 11)* \cdot ((01\vee 10) \cdot (00\vee 11)* \cdot (01\vee 10) \cdot (00\vee 11)*)*$ defines the regular set of all chains of zeros and ones containing even Number of zeros and an even number of ones.

5. The regular expression $a(a\vee b)^*$ defines the set of all possible chains consisting of *a* and *b*, starting with *a*.

6. The regular expression $(a\vee b)^*(a\vee b)(a\vee b)^*$ defines the set of all non-empty chains consisting of *a* and *b*, i.e. Set $\{a, b\}^+$.

7. The regular expression defines the set of all chains consisting of 0 and 1, whose lengths are divided by 3.

8. The regular expression $\varepsilon \vee \varphi$ defines the set $\{\varepsilon\} \cup \varnothing$.

9. Sets of numbers in decimal notation:

$Digit$ = 0∨1∨2∨3∨4∨5∨6∨7∨8∨9;

$Integer = Digit^+$;

$Fraction = Integer \vee \varepsilon$;

$Exponent = (E(+\vee - \vee \varepsilon)Integer) \vee \varepsilon$.

Tasks V.1.3.

1. Find the regular expression that specifies the set $\{a, b\}^*$.
2. Find the regular expression that specifies the set $\{a, bc^*\}^*$.
3. Find the regular expression that specifies the set $\{ab, cd\}^*$.
4. Find the regular expression that specifies the set $\{ab, b\}^*$.
5. Find a regular expression that specifies the set $\{a^*, b^*\}^*$
6. Find a regular expression for the language $\{ \omega \in \{a, b\}^* : (|\omega|_a - |\omega|_b) ...3\}$.
7. Find a regular expression for the language $\{ \omega \in \{a, b\}^* : (|\omega|_a - |\omega|_b) ...4\}$.
8. Find a regular expression for the language $\boldsymbol{L}_1 \cap \boldsymbol{L}_2 \cap \boldsymbol{L}_3$, where $\boldsymbol{L}_1 = (aaab \vee c \vee d)^*$, $\boldsymbol{L}_2 = (a^*ba^*ba^*bc \vee d)^*$, $\boldsymbol{L}_3 = ((a \vee b)^*c(a \vee b)^*cd)^*$

1. Simplify the regular expression $((a \vee b \vee ab)^*$.
2. Simplify the regular expression $(a^*b)^* \vee (b^*a)^*$.
3. Simplify the regular expression $(ba \vee a * ab)^*$.
4. Simplify the regular expression $a(\varepsilon \vee a) \vee b$.

Questions V.1.3.

1. How is the regular expression defined in the alphabet $\boldsymbol{T}$ defined?
2. Which set defines the regular expression $(ab) +$?
3. Which set defines a regular expression $(aa \vee bb)$?
4. What set defines a regular expression $a(\varepsilon \vee a) \vee b$?

5. Which set defines a regular expression $a(a\vee b)^*$?

6. Which set defines the regular expression ε^*?

7. Which set defines a regular expression $((a\vee bc)^* \cdot a)$?

8. Which set defines a regular expression $(c\cdot(ab\vee cd)^*)$?

9. Which set defines a regular expression ϕ?

10. Are the regular expressions $((a\vee b)^*\vee aa)^*$ and $(aa\vee b\vee ab)^*$ equal?

11. Are the regular expressions $(aa\vee b\vee ab)^*$and $(a\vee b)^*(a\vee b)^*$ equal?

12. Are the regular expressions $(b\vee cd^*a)^*cd^*$ and $b^*c(d\vee ab^*c)^*$ equal?

13. How can I simplify the following regular expressions:

$(00^*)0 \vee (00)^*$?

$(0 \vee 1)(\varepsilon \vee 00)^+ \vee (0\vee 1)$?

$(a \vee b)(\varepsilon \vee ab)^+ \vee (b\vee a)$?

$(aa \vee \varepsilon)(a \vee b)\ ab$?

$ab\ (a \vee b)\ (aa \vee \varepsilon)$?

14. Will the given regular expressions $(p^*q^*)^* = (q^*p^*)^*$ be equivalent?

15. Will the given regular expressions $p(qp)^*$ и $(pq)^*p$ be equivalent?

16. Will the given regular expressions $p^*(p\vee q)^*$, $(p \vee qp^*)^*$ и $(p\vee q)^*$ be equivalent?

Tests V.1.3.

1. What regular set defines a regular expression $(a \vee b)^*$?

A) $\{a, b\}^*$

B) $\{aa, b\}^*$

C) $\{aaa, b\}^*$

D) $\{a, bb\}^*$

E) $\{a, bbb\}^*$

2. What regular set determines the regular expression $a(\varepsilon \vee a) \vee b$?

A) $\{a, b, aa\}$

B) $\{a, a, aa\}$

C) $\{b, b, bb\}$

D) $\{a, b, bb\}$

E) $\{aa, bb, aa\}$

3. What regular set defines a regular expression ϕ^*?

A) $\{\varepsilon\}$

B) $\{\varepsilon, \varepsilon\}$

C) $\{\varepsilon, \varepsilon, \varepsilon\}$

D) $\{\varepsilon, \varepsilon, \varepsilon, \varepsilon\}$

E) $\{\varepsilon\ \varepsilon, \varepsilon, \varepsilon, \varepsilon\}$

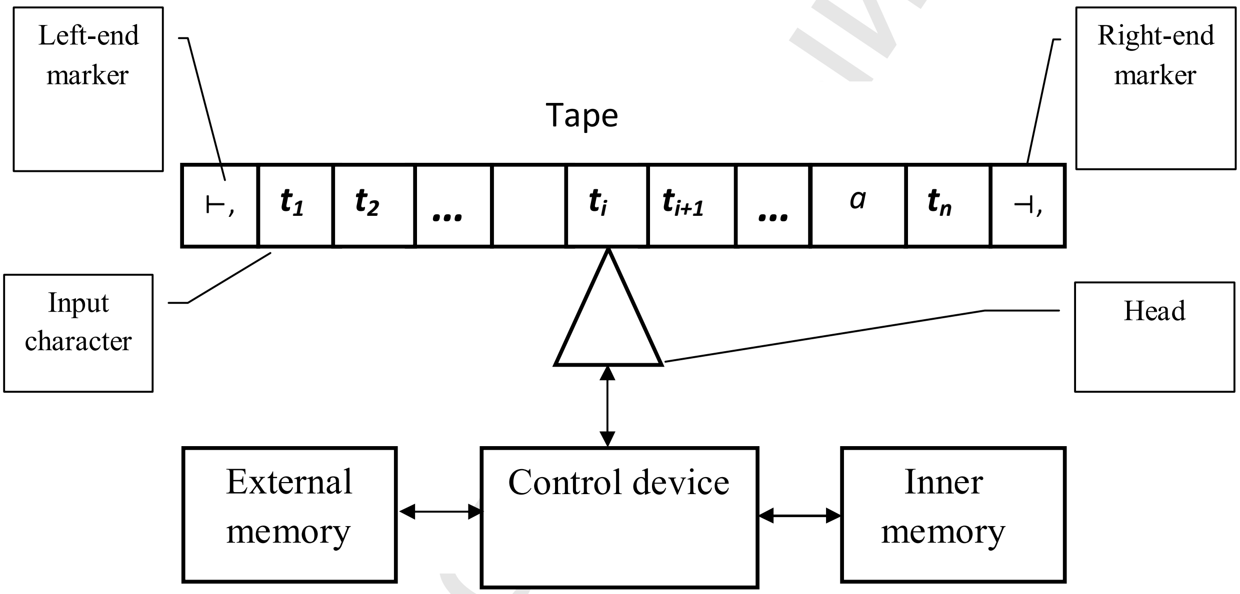
V.2. Mechanisms of language recognition

V.2.1. Finite automatons

Usually under the term “automaton” we understand a device which, once turned on, can perform a number of given operations on its own. However, we deal with an abstract automaton used as a mathematical model of any digital (discrete) devices in which all signals are quantized in level, and all actions are quantized in time.

An abstract automaton (hereinafter – automaton) can distinguish a set or transform a set into another set; it consists of a tape, a head unit and a controller device; it may also have working memory (Figure V.2.1).

может иметь рабочую память (см. рисунок V.2.1).



F V.2.1 Composition and structure of the finite automaton.

Tape – a linear sequence of cells, each of which can store only one symbol from a certain finite input (output) alphabet.

The tape is infinite, but at each given moment only a finite number of cells is occupied. Special markers denoting the beginning and end of the tape may occupy the boundary regions to the left and right of the occupied cell area. The marker may be just at one end of the tape or be absent altogether.

Input (*output*) *head unit* – a device which can view only one tape cell at any given moment of time. The head unit can shift one cell to the left or to the right, or remain immobile. It is generally assumed that the head unit is read-only, i.e. during the work of the automaton the symbols on the tape do not change. But it is also possible to consider automatons that possess a head unit which both reads and writes. Thus, the head unit may perform both reading and writing operations.

Working memory – an auxiliary storage for reading and writing data. Working memory may be organized as a dynamic data structure (queue or stack).

Controlling unit – a device which governs the automaton's behavior and has a finite internal memory for storing a finite number of states. It governs the automaton's behavior by means of a function (relation) which describes how the states change depending on the current state and current input symbol that is red by the head unit, and the current information extracted from the working memory, if available. The controlling unit also determines the direction of the shift of the head unit and the information to be entered in the working memory.

The automaton is determined by the input of a finite set of states of the controlling unit, finite set of accepted input symbols, the source state and the set of final states, as well as the state of transition function, which by using the current state and current input symbol as its arguments indicates all possible next states or values of this function. The work of the automaton may be conveniently described by means of its configuration. The automaton's configuration includes:

- controlling unit's state;
- contents of the input tape and the position of the input head unit;
- contents of the working memory and the position of the working head unit if available;
- contents of the output tape if available.

The automaton's configuration can be initial, current and final.

In its *initial configuration* the internal memory contains a previously entered symbol denoting the initial state of the controlling unit; the controlling unit is in the initial state; the head unit reads the leftmost input symbol on the tape; if working memory is available, it contains preconfigured initial contents.

In its *current configuration* the internal memory contains previously entered symbols of current states of the controlling unit; the controlling unit is in one of its current states; the head unit reads neither the leftmost nor the rightmost current input symbol; if working memory is available it has preconfigured current contents.

In its *final configuration* the internal memory contains previously entered symbols denoting the final states of the controlling unit; the controlling unit is in one of its final states; the head unit views the right end marker or, if the marker is not available, it leaves the input tape; if working memory is available then it satisfies certain conditions.

Prior to its inception the automaton is its initial configuration, i.e. the symbol denoting the initial state of the controlling unit is entered in the internal memory, the input chain is entered in the input tape; if working memory is available, corresponding data is entered in the memory.

The automaton uses a program consisting of a finite sequence of *steps*. Each step consists of the current (initial) and next (final) configuration.

At the step's beginning the memory reads the symbol of the current state of the controlling unit, the input tape reads the current input symbol; the information in the working memory, if available, is also read. Then, depending on the current state and read information the automaton's actions are determined:

(1) Input head unit moves to the right, left or remains in place;

(2) A new symbol is entered in the current cell of the input tape or the previous symbol is not changed;

(3) Some information, if available, is entered in the working memory;

(4) A symbol is entered in the output tape, if the tape is available.

(5) The controlling unit moves into another state and the number (symbol) of this state is entered in the internal memory.

As a result, during one step of the automaton the input head unit can move one cell to the left, right or remain in its place. As the automaton functions, the contents of the input tape cells do not change, but the contents of the output tape cells and the working tape cells can.

If the automaton views the input chain and executes a sequence of steps starting from the initial configuration and finishing in a final configuration, then it recognizes the chain.

A *language* recognized by the automaton is a set of chains that the automaton recognizes.

Examples V.2.1.1:

1. A public pay telephone may serve as an example of automaton: it recognizes the input of a coin and enters the dial number state.
2. An ATM is an automaton: it recognizes an inserted card and enters the pin-code input state.
3. A subway ticket gate is an automaton: it recognizes a token and enters the open gate state.

Finite automatons recognize regular languages. First, formal definitions of indeterminate and determinate finite automatons are given, then the languages they recognize are described, followed by the proof of their equivalency.

Finite automatons are among the simplest and most widespread recognizing machines. A finite automaton contains *output tape*, *internal memory*, *external memory*, *head unit and controlling unit*.

Finite automaton may be indeterminate or determinate, but its head unit must be one-way only and move only to the right. Their formal definitions are as follows:

Definition V.2.1.1. *Indeterminate finite automaton* (*IFA*) is determined by the seven element set $\boldsymbol{M} = <\boldsymbol{Q},\boldsymbol{T},\boldsymbol{I},\boldsymbol{F},\vdash,\dashv,\Delta>$ where:

$\boldsymbol{Q}$ – finite set of states of the controlling unit;

$\boldsymbol{T}$– finite set of input symbols, $\boldsymbol{Q}\cap\boldsymbol{T}=\varnothing$;

$\boldsymbol{I}$ – set of initial states of the controlling unit, $\boldsymbol{I}\subseteq\boldsymbol{Q}$;

$\boldsymbol{F}$– set of final states of the controlling unit indicating that the input chain is recognized,$\boldsymbol{F}\subseteq\boldsymbol{Q}$;

$\vdash$, $\dashv$ – tape start and end markers $\vdash, \dashv\notin\boldsymbol{T}$;

Δ–set of relations of transition $\Delta\subseteq\boldsymbol{Q}\times\boldsymbol{T}^*\times\vartheta(\boldsymbol{Q})$, $\vartheta(\boldsymbol{Q})$ – set of all subsets of the set $\boldsymbol{Q}$.

The determined finite automaton (DFA) is a special case of IFA.

Definition V.2.1.2. Finite automaton $\boldsymbol{M} = <\boldsymbol{Q},\boldsymbol{T},\boldsymbol{I},\boldsymbol{F},\vdash,\dashv,\Delta>$ is called *determined*, if:

(1) The set of initial states $\boldsymbol{I}$ contains exactly one element;

(2) For each transition $<q, \tau, p>\in\Delta$ $|\tau|=1$ holds true;

(3) For each state $q \in \boldsymbol{Q}$ and for each symbol $t \in \boldsymbol{T}$ there exists no more than one state $p \in \boldsymbol{Q}$ with an attribute $<q, t, p> \in \Delta$;

(4) Other symbols are identical to IFA.

Notes V.2.1.1:

1. Sometimes instead of the set of relations of transition Δ taking logical values "true" or "false", the function of transition δ is used which takes value as a symbol of the set $\boldsymbol{Q}$, where δ: $\boldsymbol{Q} \times \boldsymbol{T}^* \rightarrow \vartheta(\boldsymbol{Q})$ – in the case of IFA and δ: $\boldsymbol{Q} \times \boldsymbol{T}^* \rightarrow \boldsymbol{Q}$– in the case of DFA. From the function δ it is easy to arrive at the relation Δ by assuming

$\Delta = \{<q, \tau, \delta(q, \tau)>: \; q \in \boldsymbol{Q}, \tau \in \boldsymbol{T}^*\}$

2. Henceforth we shall use both relations of transition and functions of transition depending on the context without making particular mention. For any $q \in \boldsymbol{Q}, p \in \boldsymbol{Q}$ и $\tau \in \boldsymbol{T}^*$ we may use:

1)For relations of transition: $<q,\tau,\{p\}>$–for IFA, $<q,\tau,p>$–for DFA;

2)For function of transition: $\delta(q,\tau)=\{p\}$–for IFA,$\delta(q,\tau)=p$ – for DFA.

3. If we want to use the function of transition instead of the relation of transition, then in the formal definition FA it is necessary to substitute the symbol Δ with δ, and leave other symbols unchanged at their previous values, i.e. we obtain $\boldsymbol{M} = <\boldsymbol{Q}, \boldsymbol{T}, \boldsymbol{I}, \boldsymbol{F}, \vdash, \dashv, \delta>$.

The FA transition may be illustrated as a *diagram*, in which each state is denoted with a circle and transition with an arrow. An arrow from the state $q \in \boldsymbol{Q}$ to the state $p \in \boldsymbol{Q}$ denoted with a chain $\tau \in \boldsymbol{T}^*$ indicates that $<q, \tau, p>$ (or $\delta(q, \tau) = p$) is a transition within the given IFA. Each initial state may be recognized by a short arrow leading to it. Each final state is indicated with a double circle.

1. Are the following grammars equivalent?

$$S \rightarrow ab,\ S \rightarrow aKSb,\ K \rightarrow bSb,\ KS \rightarrow b,\ K \rightarrow \varepsilon$$

and

$$S \rightarrow aAb,\ A \rightarrow \varepsilon,\ A \rightarrow b,\ A \rightarrow S,\ A \rightarrow bSbS$$

2. Are the following grammars equivalent?

$$S \rightarrow aD,\ D \rightarrow bba,\ D \rightarrow baDa,\ D \rightarrow aDaDa$$

and

$$S \rightarrow aaE,\ S \rightarrow abD,\ E \rightarrow bDD,\ D \rightarrow aaEa, D \rightarrow abDa, D \rightarrow ba?$$

3. What class does the following grammar belong to?

$$S \to abba,\ S \to baa?$$

4. What class does the following grammar belong to?

$$S \to AD,\ A \to aA,\ A \to \varepsilon,\ D \to bDc,\ D \to \varepsilon$$

5. Is the grammar with the rules

$$S \to AB,\ A \to a|Aa, A \to a|Aa$$

equivalent to the grammar with the rules

$$S \to AS|SB|AB,\ A \to a,\ B \to b?$$

6. Is the grammar with the rules

$$S \to cE,\ E \to ddc, E \to dcEc, E \to cEcEc$$

equivalent to the grammar with the rules

$$S \to ccA,\ S \to cdB, A \to dBB, B \to ccAc, B \to cdBc, B \to dc?$$

How should one describe in unambiguous grammar a language generated by the ambiguous grammar $E \to E+E|E*E|(E)|i$?

Examples V.2.1.2:

1. For FA $\boldsymbol{M}_1$ with one transition and parameters: $\boldsymbol{Q}=\{q,p\}$; $\boldsymbol{T}^*=\{\tau\}$, $\boldsymbol{I}=\{q\}$, $\boldsymbol{F}=\{p\}$, $\delta(q,\tau)=p$ the diagram is shown in the figure V.2.1.2.

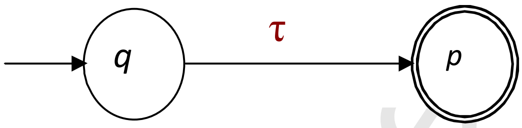


Figure V.2.1.2. Diagram FA $\boldsymbol{M}_1$ with one transition.

}, $\boldsymbol{I}$ $=\{1\}$, $\boldsymbol{F}=\{2\}$, $\Delta=\{<1,\ aaa,\ 1>,\ <1,\ ab,\ 2>,\ <1,\ b,\ 2>,\ <2,\ \varepsilon,\ 1>\}$. Figure V.2.1.3 shows a diagram of transitions of IFA $\boldsymbol{M}_2$, in which regular expressions *aaa*, *ab*, *b*,ε are used as arc markings. Such conception makes construction of the diagram easier and renders it compact and intuitive.

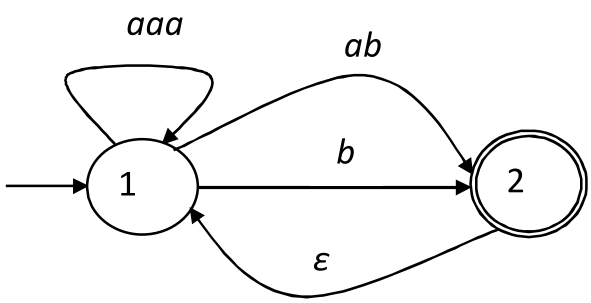


Figure V.2.1.3. Diagram FA $\boldsymbol{M}_2$ with regular expressions.

КА $\boldsymbol{M}_3$ for recognition of identifiers consisting only of letters and numbers and starting with a letter will have the following parameters:

$\boldsymbol{Q}=\{1,2\}$, $\boldsymbol{T}=\{b,d\}$, $\boldsymbol{I}=\{1\}$, $\boldsymbol{F}=\{2\}$, $\delta(1,b)=2, \delta(2,b)=2, \delta(2,d)=2$, where b – letter, d – number. The diagram FA $\boldsymbol{M}_3$ is shown in the figure V.2.1.4.

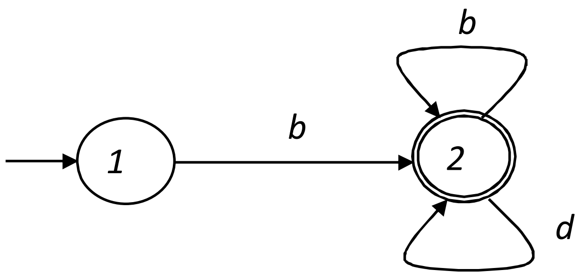


Figure V.2.1.4. Diagram FA $\boldsymbol{M}_3$ for identifier.

Note V.2.1.3. If a diagram contains several transitions with the same starting and ending point, they are called *parallel transitions*. Parallel transitions are indicated in a diagram with a single arrow. The markings of transitions are separated with commas. In figure V.2.1.5 a diagram FA $\boldsymbol{M}_4$ is shown with parallel transitions for chains *ab*, *b*.

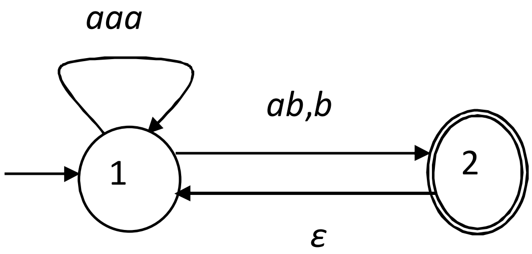


Figure КА V.2.1.5. Diagram FA $\boldsymbol{M}_4$. with parallel transitions.

The FA transitions may be represented as functions by means of a table or commands.

Convention V.2.1.1.Among all FA states the initial state q_s and final state q_f stand out; here s and f are understood not as numeral variables but as mnemonic marks of start (*start*) and end (*final*).

Examples V.2.1.3. In the table V.2.1.1 the function of transition δ FA $\boldsymbol{M}_5$ is shown determined by the sets $\boldsymbol{Q} = \{q_s, q_1, q_2, q_3\}$ and $\boldsymbol{T}= \{t_1, t_2, t_3\}$.

Table V.2.1.1. Values of the function of transition δ FA $\boldsymbol{M}_5$.

δ		Input		
		t_1	t_2	t_3
	q_s	q_2	q_2	q_2

State	q_1	q_3	q_s	q_s
	q_2	q_2	q_2	q_2
	q_3	q_3	q_2	q_s

The function of transition in the table V.2.1.1 may be represented as commands in the following way:

$$\delta(q_s, t_1) = q_2,\ \delta(q_s, t_2) = q_2, \delta(q_s, t_3) = q_2,$$
$$\delta(q_1, t_1) = q_3,\ \delta(q_1, t_2) = q_s, \delta(q_1, t_3) = q_s,$$
$$\delta(q_2, t_1) = q_2,\ \delta(q_2, t_2) = q_2, \delta(q_2, t_3) = q_2,$$
$$\delta(q_3, t_1) = q_3,\ \delta(q_3, t_2) = q_2, \delta(q_3, t_3) = q_s.$$

Let FA $\boldsymbol{M}$ be given with initial state $q_s \in \boldsymbol{Q}$,current state $q \in \boldsymbol{Q}$, final state $q_f \in \boldsymbol{Q}$ and unused current input chain $\tau \in \boldsymbol{T}^*$. Then the following description may be given.

Definitions V.2.1.3:

1. If the head unit views the leftmost symbol of the input chain, then the pair $(q_s,\tau) \in \boldsymbol{Q} \times \boldsymbol{T}^*$ is called *initial configuration* FA ;
2. If the head unit views the current symbol of the input chain τ, then the pair $(q,\tau) \in \boldsymbol{Q} \times \boldsymbol{T}^*$ is called *current configuration FA* ;
3. If the input chain τ has been read completely, then the pair $(q_f, \varepsilon) \in \boldsymbol{Q} \times \boldsymbol{T}^*$ is called *final configuration* FA ;

Note V.2.1.4. By its contents the configuration is an "instantaneous description" of FA . Assuming that the initial chain whose belonging to the language under discussion is to be verified is in the tape, then in the configuration (q,τ) the chain τ is the part of the initial chain which remains in the tape.

The step of FA is determined by the state of the controlling unit and the input symbol being viewed at that moment. The step itself consists in the change of state of the controlling unit and the shift of the head unit one cell to the right.

The Step FA $\boldsymbol{M}$ is yielded by the binary relation $\models_{\boldsymbol{M}}$, determined over its configurations in the set $\boldsymbol{Q} \times \boldsymbol{T}^*$. If the automaton is known, then the letter $\boldsymbol{M}$ in the relation $\models_{\boldsymbol{M}}$ may be omitted.

Let $t \in \boldsymbol{T}$ be the leftmost symbol of the input chain still not read and both for $q \in \boldsymbol{Q}$ and $p \in \boldsymbol{Q}$ $<q, t, p> \in \Delta$ holds true; then for the chains $\tau \in \boldsymbol{T}^*$ the relation $(q, t\tau) \vDash (p, \tau)$ is true which determines the step of the automaton; this means that the automaton is in the state q and the state unit is viewing the symbol t in the input tape; then FA $\boldsymbol{M}$ moves into the state p and the head unit moves one cell to the right. If $\tau = \varepsilon$, then the input chain is considered to have been *read completely*.

Examples V.2.1.4. Let τ = *abba*. Then in the diagram FA $\boldsymbol{M_2}$ in the figure V.2.1.2 there is a step determined as relation $(1, abba) \vDash (2, ba)$.

Definition V.2.1.4. $\vDash^{k}$ is the k–th degree of relation $\vDash$, if a chain of k+1 configurations exist

$$(q_0,\tau_0), (q_1,\tau_1), (q_2,\tau_2),\ldots, (q_{k-1},\tau_{k-1}), (q_k,\tau_k)$$

so that for any i $(1 \leq i \leq k)$ the relation is true

$$(q_{i-1},\tau_{i-1}) \vDash (q_i,\tau_i), \text{ where } q_0=q_s,\ \tau_0=\tau,\ q_k= q_f, \tau_k=\varepsilon.$$

If for any $i \geq 1$ or $i \geq 0$ $(q_0,\tau) \vDash^{i} (q_i,\varepsilon)$ holds true, then we may write $(q_0,\tau) \vDash^{+} (q_i,\varepsilon)$ or $(q_0,\tau) \vDash^{*} (q_i,\varepsilon)$ correspondingly. Here by $\vDash^{+}$ is denoted the transitive closure of relation $\vDash$, and by $\vDash^{*}$ – the reflexive and transitive closure of relation.

Definition V.2.1.5. Automaton $\boldsymbol{M}$ recognizes input chain τ, if the relation $(q_s,\tau) \vDash^{*} (q_f,\varepsilon)$ holds true.

Examples V.2.1.5. Let τ = *aaaab*. Then in FA $\boldsymbol{M}_2$ in the figure V.2.1.3 following relations $(1, aaaab) \vDash (1, ab)$ and $(1, ab) \vDash (2, \varepsilon)$ hold true.

Definition V.2.1.6. If the language $\boldsymbol{L}$ consists only of input chains recognized by automaton $\boldsymbol{M}$, then this language is recognized by automaton $\boldsymbol{M}$ and is denoted as $\boldsymbol{L}(\boldsymbol{M})$, i.e.

$\boldsymbol{L}(\boldsymbol{M}) \rightleftharpoons \{\tau\colon\ \tau \in \boldsymbol{T}^*\ \&\ (q_s,\tau) \vDash^{*} (q_f,\varepsilon)\}$.

Examples V.2.1.6. Let for $\boldsymbol{M}_6=<\{q_s,q_1,q_f\},\{0,1\},q_s,\{q_f\},\vdash,\dashv,\delta>$

there exist the following transition relations:

$<q_s,0,\{q_1\}>,<q_s,1,\{q_s\}>,<q_1,0,\{q_f\}>,<q_1,1,\{q_s\}>,<q_f,0,\{q_f\}>,<q_f,1,\{q_f\}>$

FA $\boldsymbol{M}_6$ recognizes all chains of zeroes and ones in which there are two zeroes in a row. The conditions may be interpreted in the following way:

q_s–initial condition indicates that “two zeroes in a row have not been detected and the initial symbol is a zero”;

q_1–state indicates that “two zeroes in a row have not been detected and the initial symbol is a zero”

q_f– final condition shows that “two zeroes in a row have been detected”.

It may be noted that FA $\boldsymbol{M}_6$, once entering the state q_f, remains in that state.

For the initial chain 01001 the only possible chain of configurations starting from configuration (q_0, 01001) will be (q_s,01001) $\models$ (q_1,1001) $\models$ (q_s,001) $\models$ (q_1,01) $\models$ (q_f,1) $\models$ (q_f, ε).

Thus, $01001 \in \boldsymbol{L}(\boldsymbol{M}_6)$.

The diagram of this automaton is shown in the figure V.2.1.6.

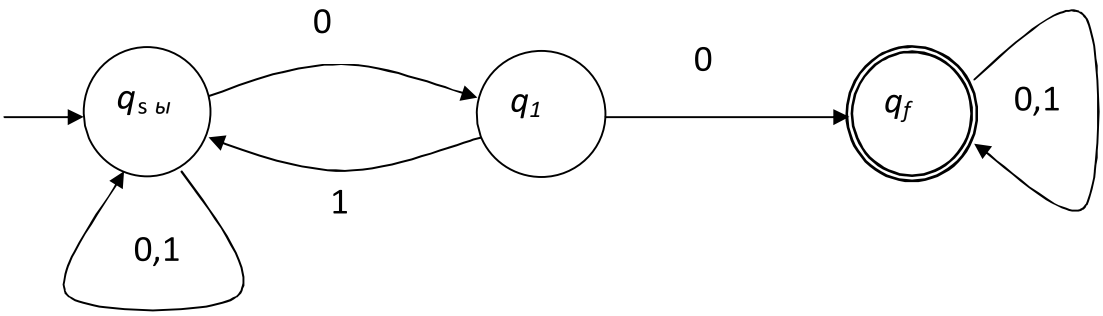


Figure V.2.1.6. Diagram FA $\boldsymbol{M}_6$.

***Definitions V.2.1.7*.**

1. Path FA is a tuple $<q_0, r_1, q_1, r_2,..., q_n>$, where $n\geq 0$ and $r_i = <q_{i-1}, \tau_i, q_i> \in \Delta$ for each i, $1\leq i\leq n$. Here q_0 – *beginning of the path*,q_n – *end of the path*, $\tau_1...\tau_n$ – *mark of the path*, n – *length of the path*.

2. A path is called *successful* if its beginning belongs to $\boldsymbol{I}$ and its end belongs to $\boldsymbol{F}$.

Note V.2.1.5. For any state $q\in\boldsymbol{Q}$ there exists a path $\langle q\rangle$. Its mark ε, beginning and end coincide.

Examples V.2.1.7. Let us consider FA $\boldsymbol{M}_2$ in the figure V.2.1.2 Let τ = *baaab*. Then the path

$<1,<1,b,2>,2,<2,\varepsilon,1>,1,<1,aaa,1>,1,<1,b,2>,2>$ is successful. Its mark is *baaab*, and its length is 4, i.e.: $q_0=1$, $q_1=2$, $q_2=1$, $q_3=1$, $q_4=2$;

$r_1=<1,b,2>$, $r_2=<2,\varepsilon,1>$, $r_3=<1,aaa,1>$, $r_4=<1,b,2>$;

$\tau_1=b$, $\tau_2=\varepsilon$, $\tau_3= aaa$, $\tau_4= b$.

Using the concept “path” it is possible to give alternative definitions to already introduced concepts of recognized chain and language.

Definitions V.2.1.8:

1. Chain $\tau \in \boldsymbol{T}^*$ *is recognized* FA $\boldsymbol{M}$, if it is the mark of a successful path.

2. FA $\boldsymbol{M}$ *recognizes* a language $\boldsymbol{L}(\boldsymbol{M})$, if it consists only of marks of all successful paths.

Note V.2.1.6. If $\boldsymbol{I} \cap \boldsymbol{F} \neq \varnothing$, then the language recognized by FA $\boldsymbol{M} = <\boldsymbol{Q}, \boldsymbol{T}, \vdash, \dashv, \boldsymbol{I}, \boldsymbol{F}, \Delta>$ contains an empty chain ε.

Examples V.2.1.8. If FA $\boldsymbol{M}_7 = <\boldsymbol{Q}, \boldsymbol{T}, \vdash, \dashv, \boldsymbol{I}, \boldsymbol{F}, \Delta>$ is given as $\boldsymbol{Q} = \{q_1, q_2\}$, $\boldsymbol{T} = \{a, b\}$, $\boldsymbol{I} = \{q_1\}$, $\boldsymbol{F} = \{q_1, q_2\}$, $\Delta = \{<q_1, a, q_2>, <q_2, b, q_1>\}$, then it is determined and recognizes the following language:

$\boldsymbol{L}(\boldsymbol{M}_7) = \{(ab)^n: n \geq 0\} \cup \{(ab)^n a: n \geq 0\}$.

The diagram of this automaton is shown in the figure V.2.1.7.

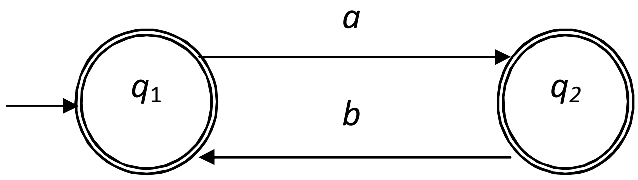


Figure V.2.1.7. Diagram FA $\boldsymbol{M}_7$.

Definition V.2.1.9. DFA $\boldsymbol{M} = <\boldsymbol{Q}, \boldsymbol{T}, \vdash, \dashv, \boldsymbol{I}, \boldsymbol{F}, \Delta>$, is called *full*, if for any state $q \in \boldsymbol{Q}$ and for any symbol $t \in \boldsymbol{T}$ there exists such state $p \in \boldsymbol{Q}$ that $<q, t, p> \in \Delta$, i.e. $\delta(q, t) = p$.

Examples V.2.1.9. The diagram of full automaton $\boldsymbol{M}_8$ with the following parameters $\Delta = \{<1,a,2>, <1,b,3>, <2,a,3>, <2,b,1>, <3,a,3>, <3,b,3>\}$, $\boldsymbol{Q} = \{1,2,3\}$, $\boldsymbol{T} = \{a, b\}$, $q_s = \{1\}$, $\boldsymbol{F} = \{1,2\}$ is shown in the figure V.2.1.8.

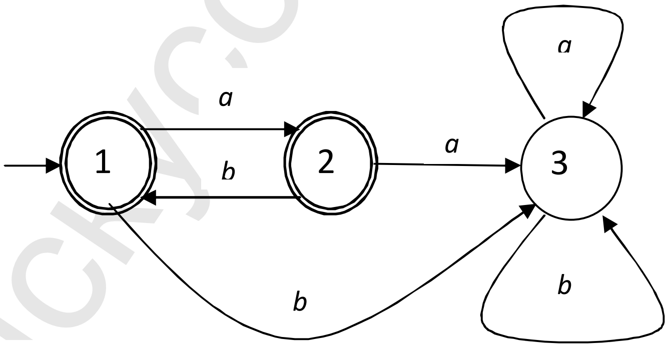


Figure V.2.1.8. Diagram FA $\boldsymbol{M}_8$

Tasks V.2.1.

1. Find a FA recognizing language $\{\alpha\beta: \alpha \in \{a,b\}^*, \beta \in \{a,b\}^*\}$.

2. Find a FA recognizing language $\{a,b\}^* \setminus (\{a^n\colon\ n\geq 0\} \cup \{ b^n\colon\ n\geq 0\})$.

3. Find a FA recognizing language $\{a\xi b\colon \xi\in\{a,b\}^* \cup \{b\xi a\colon\ \xi\in\{a,b\}^*\}$.

4. Find a FA recognizing language $\{\tau\in\{a,b\}^*\colon\ |\tau|_a\geq 3\}$.

5. Find a FA recognizing language $\{a^m b^n a^m b^n\colon m,n\geq 1\}$.

6. List all configurations $(q,\ \tau)$, satisfying the condition $(1,\ abaacdcc) \models^* (q,\ \tau)$, in FA $\boldsymbol{M_9}$ shown in the figure V.2.1.9.

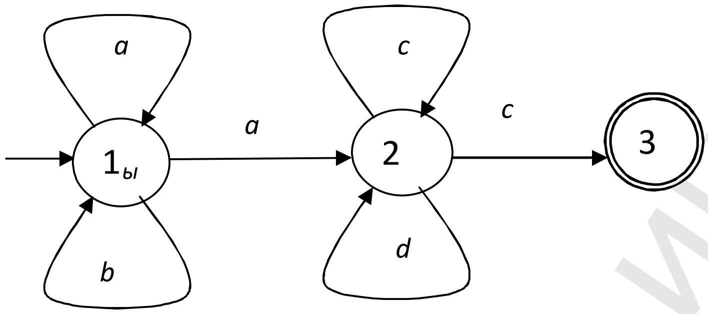


Figure V.2.1.9. Diagram FA $\boldsymbol{M_9}$.

7. Find the step of the automaton if it is determined as

$\boldsymbol{M} = \langle\{ q_0, q_1, q_2, q_f\}, \{a, b, c\}, \delta, q_0, \{q_f\}\rangle$,

where $\delta(q_0,a)=\{ q_1,q_2\}$, $\delta(q_1,a)=\{q_1\}$, $\delta(q_1,b)=\{q_f\}$, $\delta(q_2,c)=\{q_f\}$,

$\boldsymbol{L}(\boldsymbol{M}) = \{ac\}\cup\{a^n b\colon n\geq 1\}$.

8. Find the full determined finite automaton for language $(a\lor b)^*(aab\lor abaa\lor abb)(a\lor b)^*$.

9. Find the full determined finite automaton for language $(b\lor c)((ab)^*c\lor(ba)^*)^*$.

10. Find the full determined finite automaton for language $(b\lor c)^*((a\lor b)^*c(b\lor a)^*)^*$.

Questions 8:

1. Is FA $\boldsymbol{M_{10}}$ shown in the figure рисунке V.2.1.10. determined?

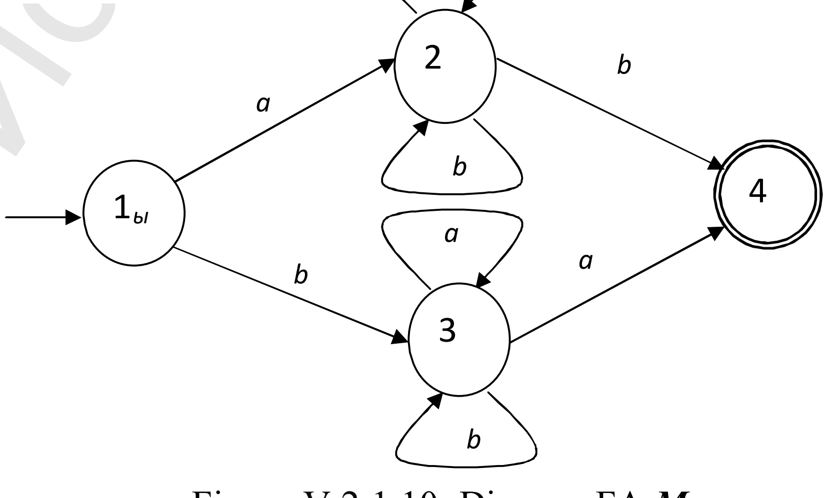


Figure V.2.1.10. Diagram FA $\boldsymbol{M_{10}}$.

2. Do FA states q_1, q_2 and chains α,β,δ exist such that the relations $(q_1, \alpha\beta) \models^* (q_2, \beta)$ и $\neg (q_1, \alpha\delta) \models^* (q_2, \delta)$ hold true?

3. How are $|\boldsymbol{Q}|$, $|\boldsymbol{T}|$, $|\Delta|$,$|\tau|$ and the number of configurations attainable from (q,τ) related in the sense of $\models^*$?

4. What automaton can recognize the language generated by the regular expression $(abab)\vee(aba)^*$?

5. What contains the input tape?

6. What determines the direction of the shift of the head unit?

7. What does the automaton configuration consist of?

8. What types of configurations exist?

9. What does an automaton – recognized language consist of?

10. Is the determined finite automaton $\boldsymbol{M_{11}}$ with alphabet $\boldsymbol{T} = \{a, b, c\}$ shown in the figure V.2.1.11 full?

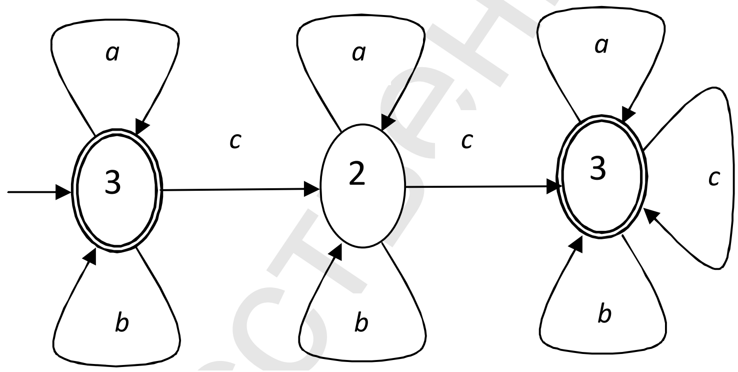


Figure V.2.1.11. Diagram FA $\boldsymbol{M_{11}}$.

11. Is the determined finite automaton $\boldsymbol{M_{12}}$ with alphabet $\boldsymbol{T} = \{a, b\}$ shown in the figure V.2.1.12. full?

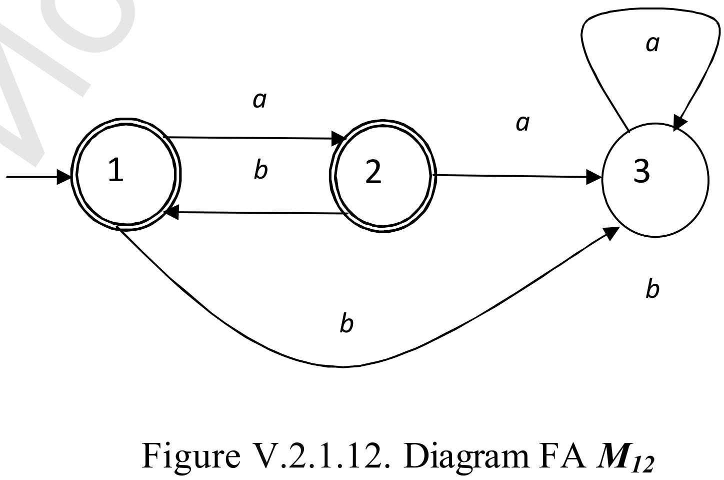


Figure V.2.1.12. Diagram FA $\boldsymbol{M_{12}}$

12. What does the graph of transition of finite automaton satisfying a given grammar look like?

$$G = < \{+, -, *\}, \{A, B, C\}, P, C > .$$
$$\begin{aligned} P: A &\to + \mid A+ \\ B &\to - \mid B- \mid A- \mid A+ \\ C &\to * \mid C* \mid B* \mid B- \mid A+ \end{aligned}$$

Tests V.2.1.

1. Finite automatons move to a distinct state in accordance with:

A) transition table in the automaton’s memory;

B) given task;

C) figures;

D) directions;

E) contents.

2. Which automaton is called determined?

A) if for any acceptable configuration of the identifier arising at one of the steps of its operation there exist two configurations in one of which the identifier will move in the following step;

B) if for any acceptable configuration of the identifier arising at one of the steps of its operation there exists a uniquely possible configuration in which the identifier will move in the following step;

C) if the identifier has an acceptable configuration for which there exists a finite set of configurations possible at the next step of operation;

D) if the identifier allows reading input symbols in one direction only (“from the left to the right”);

E) if the identifier allows that the reading device move in both directions with respect to the chain of input symbols – both forwards from the beginning of the tape to its end and backwards going back to previously read symbols.

3.Finite automaton is a five – element set $\boldsymbol{M} = <Q, T, \delta, q_0, F>$, where Q is:

A) a finite set of acceptable input symbols;

B) a finite set of states;

C) transition function;

D) initial state;

E) final state.

REFERENCE

1. Urginovich N. The Practical work on information technologies.- Moscow: BINOM, 2004.
2. Kuk D. The computer mathematics. - Moscow: Nauka, 1990.
3. Kozlov V.N. Mathematics and computer science.- Piter, 2004.
4. Meduna, Alexander (2014), Formal Languages and Computation: Models and Their Applications, CRC Press, p. 233. For more on this subject, see undecidable problem.
5. Ginsburg, Seymour (1975). Algebraic and automata theoretic properties of formal languages. North-Holland. pp. 8–9.
6. Akimov O.E. The discrete mathematics: logic, groups, graphs.- Moscow: The Laboratory of base knowledge, 2001.
7. Novikov F.A. The discrete mathematics for programmers: the Textbook for high schools. –St-Petersburg: Piter, 2008.
8. Meyer B., Baudoin C. Programming methods. In two parts: Part 2. Translate from French Y.A. Pervina - Moscow: Mir,1982.
9. http://www.regentsprep.org/regents/math/algebra/AP1/Interval
10. Хопкрофт, Дж., Мотвани, Р., Дж. Ульман. Введение в теорию автоматов, языков и вычислений. — М.: Вильямс, 2002 (пер. издания Addison Wesley). — 528 с. — ISBN 5-8459-0261-4
11. Harrison, Michael A. (1978). Introduction to Formal Language Theory. Reading, Mass.: Addison-Wesley Publishing Company.p.13.
12. Шарипбай А.А. Информатика, Алматы, Эверо, 2015, -313 с.
13. Шарипбай А.А. Теория языков и автоматов, Астана, ЕНУ им. Л.Н. Гумилева, 2015, -207 с.

Co-funded by the
Tempus Programme
of the European Union

PROMIS

SHARIPBAY A.A.

MATHEMATICS FOR COMPUTER SCIENCE
(Training manual)

Подписано к печати 29.05.2017 г. Бумага офсетная №1.
Формат 60X84 1/16 Объем 9,5 усл.печ.л.
Тираж 50 экз. Заказ0168
Отпечатано в типографии ТОО «Мастер ПО»
010008, г. Астана, ул. Пушкина, 15-76
(ул. Кенесары, 93-76) тел.: 8 (7172) 223-418
e-mail.: masterpo08@mail.ru

SHARIPBAY Altynbek Amiruly, Doctor of technical sciences, professor, Academician of the International Academy of Informatization, Academician of the Academy of Pedagogical Sciences of the Republic of Kazakhstan. Laureate of the State Prize of the Republic of Kazakhstan in the field of science, technology and education, Director of the Research Institute "Artificial Intelligence" and Professor of the Department of Computer Science & Information Security LN Gumilyov Eurasian National University (www.e-zerde.kz).

His scientific interests are theoretical and applied problems of informatics and information technologies: theory and technology of programming, automated systems, artificial intelligence, computer linguistics and e-learning. In this area, he developed methods for formalizing the semantics of rewriting and logical programming languages, methods for automated verification of software and hardware. Based on the results of these studies, he defended his doctoral thesis on "Verification of software and hardware of computers and systems" on the specialty "05.13.13 (05.13.11) - Computing machines, systems and networks (mathematical, software and hardware). Some of his scientific results were introduced into major scientific centers: "The translator from the language of simulation of space systems" in the Flight Test Institute, Zhukovsk; "System of verification of digital circuits" in the Research and Production Center, Zelenograd; "Parallel Programming System for Multiprocessor Computing Complex" in the Scientific Research Center of Electronic Computer Engineering, Moscow. In the field of information technology, he created various automated information systems. In particular, the "Automated Depository System of Securities" for the Department of State Property and Joint Stock Companies of Almaty; "Automated accounting system for utility payments" for the Department of Public Utilities and KSK in Almaty; "Automated system of elections"; "Database of Information Control" for the Accounts Committee of the Republic of Kazakhstan; "Automation system for the creation of electronic textbooks" for schools and universities of Kazakhstan.

He published more than 400 scientific papers, 10 textbooks and more than 10 teaching aids, 4 monographs and 5 terminology and explanatory dictionaries on computer science and computer technology, received more than 30 certificates of state registration of intellectual property, participated in the development of many states standards: 10 - in information technology, 9 - in the field of education.

Under the scientific guidance of A.Sharipbay 5 doctors and 8 candidates of sciences, 6 doctors of PhD on group of specialties "Computer science, computer facilities and management" are prepared. His scientific school created a mathematical theory of the Kazakh language, developed methods for automated analysis and synthesis of oral and written words and suggestions of the Kazakh language, proposed a technology for creating electronic educational publications, etc. These scientific results are used to create many electronic text books, a system of distance learning of the Kazakh language, a system for recognizing and synthesizing Kazakh speech and other automated systems, including accounting and expert systems in the public and private agencies and organizations of Kazakhstan.