Ministry of Education and Science of Ukraine

Odessa National Academy of Telecommunications Named After A.S. Popov

Sub-faculty of information technologies

ALGORITHMIZATION AND PROGRAMMING

Part 1
BASIC ALGORITHMS

Methodical instructions for laboratory training and exercises

ODESSA 2018

УДК 004.43 План НМВ 2018 р.

Compilers: **Y. V. Prokop, L. N. Bukata**, **E. G. Trofimenko**

These instructions for laboratory training and exercises contain theoretical information with examples of programs in Visual C++ and variants of individual problems. Methodical instructions will be useful for students of the Academy of Telecommunications who are studying in English, fixing theoretical material, preparing to laboratory training and exercises in the discipline of Algorithmization and programming during the first semester.

It is intended for the acquisition of skills in operation on a personal computer and programming by students of the academy studying in English, with the purpose of further usage of these skills in daily professional work. Also, it will be useful for users of personal computers wishing to learn programming in Visual C++ environment.

APPROVED
by the sub-faculty IT meeting and
recommended for publication.

Minutes № from

APPROVED
by the Methodology Council
the Academy of Communications.

Minutes № from

Introduction

The Visual C++ language and development tools help you to develop native Universal Windows apps, native desktop and server applications, cross-platform libraries that run on Android and iOS as well as Windows, and managed apps that run on the .NET Framework.

Visual C++ supports two distinct, but closely related flavors of the C++ language, ISO/IEC standard C++ (native C++) and C++/CLI. We'll cover both of them in this course.

There are two principal types of applications, which we are going to create: console application (it executes in "black screen" in character mode) and application with graphical interface with Windows forms.

There's a lot of code even in a simple Windows program, and it's very important not to be distracted by the complexities of Windows while learning the ins and outs of C++.

These instructions present a brief theoretical information, examples of how to create software projects using Visual C++ for the calculation of linear, branched and cyclic structures, control questions and variants of individual problems to the nine laboratory exercises. Each of the proposed laboratory exercises contains problems of different complexity levels. The student himself or under the teacher's instructions selects problem of this or that complexity level according to a student's log number. Problems of basic level are obvious.

Before realization of the laboratory training the student should do the following:

- to select individual tasks with the teacher;
- to study relevant sections of the theoretical course in accordance with lecture notes and academic literature;
- to develop flowcharts for solving the individual problems;
- to write code in C++;
- to prepare a report of the laboratory exercise and submit it to the teacher for checking.

The students may perform Laboratory exercise only after they prepared a report.

The content of the report of the laboratory exercise:

- the title of the topic and the purpose of the laboratory exercise;
- answers to control questions;
- flowcharts for solving of the individual problems;
- programming code of his problems in C++;
- form of the project with elements (in the case of Windows Forms project);
- results of calculations on the computer.

The correctness of execution of the program and obtained results should be checked by the teacher.

Lab # 1

Introduction to Visual C++. Programs with linear algorithm

Purpose: to become familiar with the interface of Microsoft Visual C++ and to purchase skills in creation and debugging of programming projects, in writing arithmetical expressions on C++ language and creation of programming projects with linear structure in Visual C++.

Theoretical information

1. Input/Output

There are several different ways to input and output data in Visual C++.

C++-style of input/output streams `cin>>` (input) and `cout<<` (output), which are defined in `iostream.h` library. For example:

```
cout<< "Input a number: ";
cin>> x;
cout<< "This number squared is: " << x*x << endl;
```

First `cout<<` outputs the string and puts the cursor at the end of that line. The second command allows you to input the value of variable `x`. The third command outputs the string with the value of `x` squared and places the cursor at the beginning of the next line of the screen due to `endl`.

The manipulator `setw` specifies the width of the displayed field for the next element in the stream. It only sets the width for the next element in the stream and must be inserted before each element which width you want to specify. It is necessary to include `iomanip` header to use manipulator `setw`. Example:

```
#include <iomanip>
. . . . .
cout<< "x=" << setw(1) << 155 << endl;
cout<< "x=" << setw(3) << 155 << endl;
cout<< "x=" << setw(5) << 155 << endl;
```

Result:

```
x= 155
x= 155
x=  155
```

There are two spaces before the value 155 in the last line, and the number occupies 5 positions.

Manipulator `setprecision(int count)` sets the format of real numbers:

```
cout << fixed << setprecision(3) << (13.5 / 2) << endl;  // 6.750
cout << fixed << setprecision(2) << 24.16425 << endl;    // 24.16
```

C-style

1) Functions **scanf** and **printf** from stdio.h header.

scanf (<*format*>, < *list of variables*>);

printf (<*format*>, < *list of variables* >);

where: *format* is a string in quotes, which may contain: %i – for integer numbers, %f – for float numbers, %s – for strings;

variables in the list are separated with commas.

Example: the function printf("x= %7.3f\n", x) outputs a real number x with 3 digits after the decimal point, and the function scanf("%i %f", &kol, &price) inputs values for integer variable kol and real variable price.

2) Functions **gets** and **puts** for input/output of strings, example:

```
puts("Hello, Dolly");
gets(s);
```

Function puts(s) outputs the string s to the screen.

Function gets(s) inputs the string from the keyboard.

.NET-style of input/output

Class System.Console is a part of CLR library and contains functions of input/output. This class belongs to namespace System. Methods Read and ReadLine are used for input, and methods Write and WriteLine – for output.

Examples:

```
double x=54.321, y=0.0125; int n=1234;
Console::WriteLine();
Console::WriteLine("n= {0:d}", n );          // n= 1234
Console::WriteLine("n= {0:d5}", n );         // n= 01234
Console::WriteLine("x={0:f} y={1:f2}", x, y);   // x=54.32   y=0.01
Console::WriteLine("Price: \t{0,7:c}",x);       // Price:  $54.32
Console::WriteLine("y={0:e} x={0:E2}",y,x);     // y=1.250000e-002 x=5.43E+001
Console::WriteLine("n= {0:n}", n );          // n= 1,234.00
```

Class **Convert** (namespace System) converts a value of one basic type to another with the help of some methods:

Method	**Conversion**
ToDouble()	String to double
ToString()	double to String
Parse()	String to double
ToString()	int to String
Parse()	String to int

Examples:

```
double x = Convert::ToDouble(textBox1->Text);
textBox2->Text = Convert::ToString(x*x);
int n = 3; double x;
Console::WriteLine(n.ToString());
String *s = S"3.14";
x = Double::Parse(s);
```

2. Creation of projects in Visual C++

Console Application Win32

1. Run Visual Studio and choose *Create Project / Visual C++ / Win32 / Console Application Win32*. Input the name of your project and select folder for it. Click *OK* and then *Ready*. After this you'll see the window of cpp-file with automatically generated code of the main function:

```
#include "stdafx.h"
int _tmain(int argc, _TCHAR* argv[])
{  return 0;
}
```

2. Write the program code.

Example: program calculates the sum of two numbers.

```
#include "stdafx.h"
#include <iostream>          // for cin and cout
using namespace std;         // the same

int _tmain(int argc, _TCHAR* argv[])
{ double a,b;
  cout << " Input 2 numbers (separated with a space) \n";
  cin >> a >> b;
  cout <<" The sum of these numbers - "<< a + b << endl;
  system ("pause>>void");
  return 0;
}
```

3. Run the program: *Ctrl + F5* or *Debug – Start Without Debugging*.
4. Input two numbers separated with a space in the black screen. Result:

```
Input 2 numbers (separated with a space)
45.15  8.12
The sum of these numbers - 53.27
```

Other ways to run the program are:

– *Debug – Start Debugging*;
– button  ;
– *F5*.

5. Save project: *File / Save All*, or press *Ctrl+Shift+S*, or click button  .

CLR console application

1. Run Visual Studio and choose *Create Project / Visual C++ / CLR / Console Application CLR*. Input a name of your project and select a folder for it. Click *OK* and then *Ready*. After this you'll see the window of cpp-file with automatically generated code of the main function:

```
#include "stdafx.h"
using namespace System;
int main(array<System::String ^> ^args)
{  Console::WriteLine(L"Hello, world!");
   return 0;
}
```

2. Type the program code:

```
#include "stdafx.h"
using namespace System;
int main(array<System::String ^> ^args)
{ double a,b;
  Console::Write("Input 2 numbers (with Enter)\n");
  a = Convert::ToDouble(Console::ReadLine());
  b = Convert::ToDouble(Console::ReadLine());
  Console::WriteLine("The sum of these numbers - {0:f}", a+b);
  Console::Read();
  return 0;
}
```

```
Input 2 numbers (with Enter)
45,12
23,48
The sum of these numbers - 68,60
```

3. Types of algorithms. Linear algorithms

We solve many trivial problems every day without even thinking about it, for example making breakfast, travelling to the workplace etc.

*An **algorithm** is a procedure consisting of a finite set of unambiguous rules (instructions) which specify a finite sequence of operations that provides the solution of a problem, or to a specific class of problems for any allowable set of input quantities (if there are inputs). In other word, an **algorithm** is a set or list of instructions for carrying out some process step by step.*

A recipe in a cookbook is an excellent example of an algorithm. The recipe includes the requirements for the cooking or ingredients and the method of cooking them until you end up with a nice cooked dish.

In the same way, algorithms executed by a computer can combine millions of elementary steps, such as additions and subtractions, into a complicated mathematical calculation.

One of the obstacles to overcome in using a computer to solve your problems is that of translating the idea of the algorithm to computer code (program). People cannot normally understand the actual machine code that the computer needs to run a program, so programs are written in a programming language such as C or Pascal, which is then converted into machine code for the computer to run.

In the problem-solving phase of computer programming, you will be designing algorithms. These algorithms can be designed through the use of **flowcharts** or **pseudocode**.

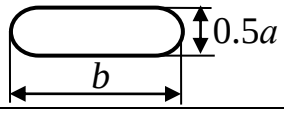
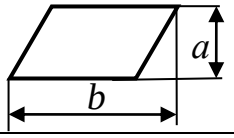
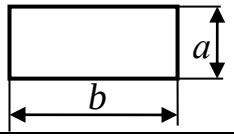
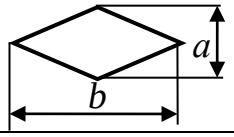
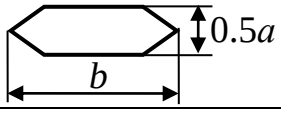
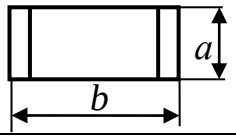
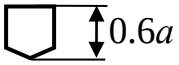
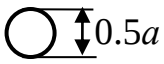
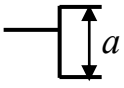
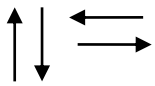
Flowcharting is a tool developed in the computer industry, for showing the steps involved in a process. A flowchart is a diagram made up of *boxes*, *diamonds* and other shapes, *connected by arrows* - each shape represents a step in the process, and the arrows show the order in which they occur. Flowcharting combines symbols and flowlines, to show figuratively the operation of an algorithm.

Flowcharting symbols

There are 6 basic symbols commonly used in flowcharting: Terminal, Process, Input / Output (I/O), Decision, Connector and Predefined Process. This is not a

complete list of all the possible flowcharting symbols, it is the ones used most in the structure of programs.

Generally, there are standard flowcharting symbols.

Name	Symbol	Function
Terminal (Begin / End)		Indicates the starting or ending of the program, process, or interrupt program
Input / Output		Used for any Input / Output (I/O) operation. Indicates that the computer is to obtain data or output results
Process		Indicates any type of internal operation inside the Processor or Memory
Decision		Used to ask a question that can be answered in a binary format (Yes / No, True / False)
Modification (Header cycle)		Sets the initial and final value and step for the cycle
Predefined Process		Used to invoke a subroutine or an interrupt program
Link to another page		Sets the connection between flowcharting symbol, which located on different pages
Connector		Allows the flowchart to be drawn without intersecting lines or without a reverse flow
Comment		Connects the flowcharting symbol and the explanation
Flow Lines		Shows direction of flow

General rules for flowcharting (ДСТУ 19.701-90, ISO 5807-85):

1) All boxes of the flowchart are connected with lines or arrows.

2) Flowchart symbols have an entry point on the top of the symbol with no other entry points. The exit point for all flowchart symbols is on the bottom except for the Decision symbol.

3) The Decision symbol has two exit points; these can be on the sides or the bottom and one side.

4) Generally a flowchart will flow from top to bottom.

5) Connectors are used to connect breaks in the flowchart. Examples are:

– from one page to another page;

– from the bottom of the page to the top of the same page.

6) Subroutines and Interrupt programs have their own and independent flowcharts.

7) All flow charts start with a Terminal or Predefined Process (for interrupt programs or subroutines) symbol.

8) All flowcharts end with a terminal or a contentious loop.

There are **three basic types of algorithms**:

– linear (Sequence);
– branching (Decision Structure or Selection Structure);
– loops (Repetition or Iteration Structure).

They all are usually combined in programs.

The linear structure (Sequence)

The first type of control structure is called the linear or sequence structure. It is the most elementary structure. The sequence structure is a case where the steps in an algorithm are constructed in such a way that, no condition step is required. The sequence structure is the logical equivalent of a straight line.

Standard sequence of linear algorithm:

– to input data;
– to make calculations according to the problem;
– to output the result.

Example: This pseudo-code requires to input three numbers from the keyboard and to output the result.

Use variables: sum, number1, number2, number3 of integer type.
Input number1, number2, number3.
Sum = number1 + number2 + number3.
Output the sum.
End of the program.

4. Basic data types in C++

C++ supports a large number of data types. The built in or basic data types supported by C++ are integer, floating point and character. C++ also provides the bool data type for variables that can only hold the values of true and false.

Variable is a location in the computer memory which can store data and is given a symbolic name for easy reference. The variables can be used to hold different values at different times during the execution of a program.

There are three basic types of variables: numeric-integer, numeric-real and character.

Numeric variables can either be of the type integer (`int`) or of the type real (`double`). Integer (`int`) values are integer numbers (like 1, 3, 10 or -15.) Real (`double`) values are numbers with decimal point (like 1.25, -0.25 or 3.0).

Character variables are letters of the alphabet, ASCII characters or digits (‘0’–‘9’). If you declare a character variable you must always put the character between single quotes (like ‘a’, ‘!’, ‘*’, ‘6’).

The **basic fundamental data types** in the C++ language are:

Type	**Keyword**
Boolean	`bool`
Character	`char`
Integer	`int`
Floating point	`float`
Double floating point	`double`
Valueless	`void`
Wide character	`wchar_t`

Declaration of variables

Before a variable is used in a program, it must be declared. This activity enables the compiler to make available the appropriate type of location in the memory.

```
double z;
```

You can declare more than one variable of the same type in a single statement:

```
int x, y;
```

Initialization of variables

A default value of declared variable is undetermined. We can initialize a variable in declaration:

```
int a = 20;
```

The other way to initialize variables, known as constructor initialization, is done by enclosing the initial value between parentheses (), for example:

```
int a (20);
```

Both ways of initializing variables are equivalent in C++.

Constants

A variable which does not change its value during execution of a program is known as a constant variable. Any attempt to change the value of a constant will result in an error message. A constant in C++ can be of any of the basic data types, const qualifier can be used to declare constant as shown below:

```
const double PI = 3.1415;
```

The above declaration means that `PI` is a constant of `double` type having a value `3.1415`.

Examples of valid constant declarations are:

```
const int RATE = 50;
const double PI = 3.1415;
const char CH = 'A';
```

Type conversion

The process in which one pre-defined type of expression is converted into another type is called **conversion**.

Data type can be mixed in the expression, for example:

```
double a; int b = 5; float c = 8.5;
a = b * c;
```

Function `sizeof(` *type* `)` – queries size of the object or type. Examples:

```
a = sizeof(int);                // a = 4
b = sizeof(long double);        // b = 10
```

For math constants `M_PI` ($\pi \approx 3.14159$), `M_E` ($e \approx 2.71828$) you have to write:

```
#define _USE_MATH_DEFINES
#include <math.h>
```

Another way is to use constants from platform .NET

```
Math::PI
Math::E
```

Basic math functions C++ and Math class (.NET) (required `#include <math.h>`)

Function C++	Description	Math Class
abs	module (absolute value) of an integer number x – $\|x\|$	`Math::Abs(x)`
fabs	module of a real number x – $\|x\|$	`Math::Abs(x)`
sqrt	square root – $\sqrt{x}$	`Math::Sqrt(x)`
pow	involution – x^y	`Math::Pow(x,y)`
exp	exponent e^x	`Math::Exp(x)`
log	natural logarithm – $\ln(x)$	`Math::Log(x)`
log10	logarithm – $\lg(x)$	`Math::Log10(x)`
–	logarithm x to the base N – $\log_N(x)$	`Math::Log(x,N)`
cos	cosine – $\cos(x)$	`Math::Cos(x)`
sin	sinus – $\sin(x)$	`Math::Sin(x)`
tan	tangent – $\mathrm{tg}(x)$	`Math::Tan(x)`
acos	arc cosine – $\arccos(x)$	`Math::Acos(x)`
asin	arc sinus – $\arcsin(x)$	`Math::Asin(x)`
atan	arc tangent – $\mathrm{arctg}\ (x)$	`Math::Atan(x)`
cosh	giperbolichny cosine – $(e^x + e^{-x})/2$	`Math::Cosh(x)`
sinh	giperbolichny sinus – $(e^x - e^{-x})/2$	`Math::Sinh(x)`
ceil	rounding to top: the smallest integer, not less than x	`Math::Ceiling(x)`
floor	rounding down: the largest integer, not bigger than x	`Math::Floor(x)`

For logarithm function: `Math::Log(x,N)` or `log(x)/log(N)` in C++.

Arithmetic operations: `+`, `-`, `*`, `/`, `%`, `++`, `--`.

Examples:

```
int n = 49, m = 10, x, y;
x = n / m;    // x = 4
y = n % m;    /* y = 9  */
```

Text after `//` or inside `/*  */` means comments, it is ignored by compiler.

Rules for writing math expressions in C++:

– each instruction (command) ends with semicolon (;);
– C++ language is case sensitive, variables `x` and `X` are different;
– function arguments always must be written in parentheses;
– remember about symbol of multiplication (`3ab -> 3*a*b`);

– if a denominator or numerator has operations (+, –, *, /), then it must be written in parentheses;
– if the numerator or the denominator of rational fractions contains numeric constants, at least one of these constants should be written as a real number indicating the decimal point, for example: the fraction $2/k$ should be written as `2.0 / k`.

Assignment operator

Assignment operator is a main operator in programming. Its format is:

<*variable*> = <*expression*>;

This operator calculates the value of expression and assigns this value to the variable.

Other variants of assignment operator: `+=` `-=` `*=` `/=` `%=` and some other.

You can do multiple assignments:

```
a = b = c = x * y;
```

This command applies from the right to the left (at first `x * y`, then `c`, then `b` and after this – a).

Examples of type conversion

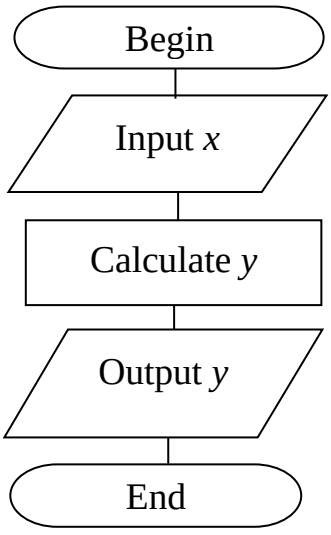
1. Result of `2/3` is 0, and result of `2.0/3` (or `2./3`) is `0.666(6)`.
2.
```
double a = 5.1, b = 1.5;
int c = a * b; // c=7, digits after decimal point will be lost
```
3.
```
int m = 2, n = 3;
double a = (double) m/n;      // a=0.66(6)
```

Another way to get the same result:

```
double a = (m*1.0)/n;         // a=0.66(6)
```

Examples of programs with linear structure (Sequence)

Example 1.1. Draw the flowchart and create the project to input variable x and calculate: $y = \dfrac{0.2x^2 - x}{(\sqrt{3}+x)(1+2x)} + \dfrac{2(x-1)^3}{\sin^2 x + 1}$.



Solution.

1) Run Visual Studio.

2) Choose *Create project / Visual C++ / CLR / Console application CLR*. Click *Browse*, select folder and write the name of your project. Click *OK* and write the program code.

3) Save your project: *File / Save All* (or click  , or press *Ctrl+Shift+ S* keys).

4) Run your program: *Debug / Start Debugging*, or press *F5* key.

The program code:

```
#include <iostream>
#include <math.h>
using namespace std;
int main ()
{ double y, x;
  cout<<"Input a value of variable x= ";
  cin>>x;
  y=(0.2*x*x-x)/((sqrt(3.)+x)*(1+2*x)) +2*pow(x-1,3)/(pow(sin(x),2)+1);
  cout<<"Result y= "<< y <<endl;
  system ("pause");
  return 0;
}
```

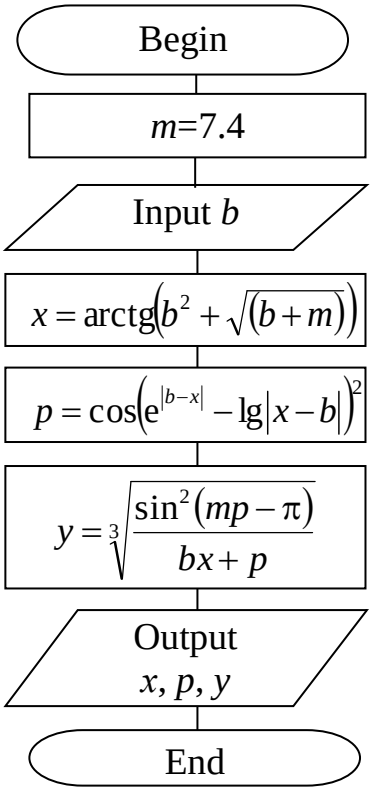
Result of the project:

```
Input a value of variable x= 2.15
Result y= 1.72928
```

Example 1.2. Draw the flowchart and create the project to input variable b and calculate: $y=\sqrt[3]{\dfrac{\sin^2(mp-\pi)}{bx+p}}$; $x=\operatorname{arctg}\left(b^2+\sqrt{(b+m)}\right)$ and $p=\cos\left(e^{|b-x|}-\lg|x-b|\right)^2$. Constant m=7.4.

The program code and the flowchart:

```
#include <iostream>
#include <math.h>
using namespace std;
int main ()
{ system("color F0");  // Set white background and black text
  const double m=7.4, pi=3.1415;
  double b, x, p, y;
  cout<<"Input a value of variable b= ";
  cin>>b;
  x=atan(b*b+sqrt(b+m));
  p=cos(pow(exp(fabs(b-x))-log10(fabs(x-b)),2));
  y=pow(pow(sin(m*p-pi),2)/(b*x+p),1./3);
  cout<<"Results:\n x= "<< x <<endl;
  cout<<" p= "<< p <<endl;
  cout<<" y= "<< y <<endl;
  system ("pause>>void");
  return 0;
}
```



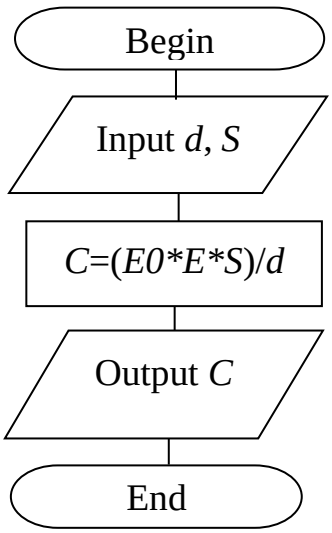
Result of the project:

```
Input a value of variable b= 8.4
Results:
 x= 1.55738
 p= -0.128333
 y= 0.370956
```

Example 1.3. Calculate the condenser capacity by the formula $C = \frac{\varepsilon_0 \cdot \varepsilon \cdot S}{d}$, where $\varepsilon_0 = 8.85 \cdot 10^{-12}$ F/m, $\varepsilon = 2.8$ F/m, S – area of each plate of condenser, d – distance between the plates, input their values from the screen.

The program code and the flowchart:

```
#include <iostream>
#include <math.h>
using namespace std;
int main ()
{ system("color F0");     // Set white background and black text
  double d, S, C;
  const double E0 = 8.85e-12, E = 2.8;
  cout<< " Input the value of distance between
the plates d= "  ;
  cin>>d;
  cout<< " Input the value of area of plate of
          condenser S= ";
  cin>>S;
  C=(E0*E*S)/d;
  cout<<" Condenser capacity C= "<< C <<endl;
  system ("pause>>void");
}
```



Results:

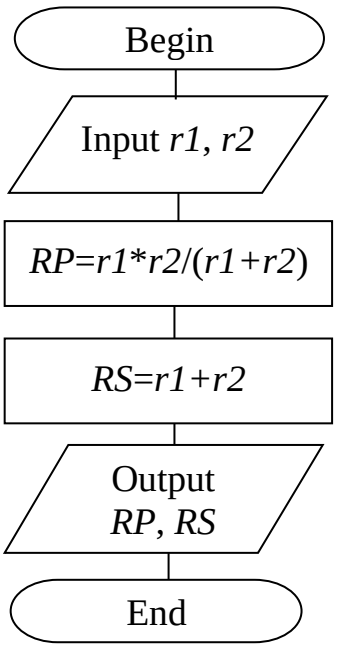
```
Input the value of distance between the plates d= 0.0004
Input the value of area of plate of condenser S= 50000
Condenser capacity C= 0.0030975
```

Example 1.4. Create the project to calculate resistance of electric chain, which consists from two united resistances, in two ways: 1) parallel connection and 2) serious connection of resistances in the electric chain.

Solution. Formulas: 1) $R = \frac{r1 \cdot r2}{r1 + r2}$ and 2) $R = r1 + r2$.

The program code and the flowchart:

```
#include "stdafx.h"
#include <iostream>
using namespace std;
int main(array<System::String ^> ^args)
{ double r1, r2, RP, RS;
  cout<<"Enter the 1-st resistance r1 "; cin>>r1;
  cout<<"Enter the 2-nd resistance r2 "; cin>>r2;
  RP = r1*r2/(r1+r2);
  RS = r1+r2;
  cout<<"Parallel connection: RP = "<<RP<<endl;
  cout<<"Serious connection: RS = "<<RS<<endl;
```



```
  system("pause");
  return 0;
}
```

Results of the program:

```
Enter the 1-st resistance r1 10
Enter the 2-nd resistance r2 12
Parallel connection: RP = 5.45455
Serious connection: RS = 22
```

Control questions

1) What different facilities of creation of programming projects in Visual C++ do you know?
2) What is a console program?
3) Designate the sequence of actions to create the console project in Visual C++.
4) What input/output functions in the console mode do you know?
5) How can some header file be included in the project?
6) What header must be included to use input/output functions?
7) What ways to run the program do you know?
8) How can you save the project? What is the difference between *Save*, *Save As* and *Save All*?
9) What is linear process?
10) What basic types in C++ do you know?
11) What are the values of `j` and `i` after calculation for `int j, i=2;`
    a) `j = i++ + i++;`
    b) `j = ++i + ++i;`
    c) `j = i++ + ++i;`
    d) `j = ++i + i--;`
12) Write the constant `0.2731e3` in fixed format.
13) Write the number `0.0001` in exponential form.
14) Write down expression `y = x3 + sin 2x` with C++.
15) What is the value of variable `y` after calculation:
    a) `y = (1 + 2) / (3 + 2);`
    b) `y = 1 + 2 / 3 + 2;`
    c) `y = 1 + 2.0 / 3 + 2;`
    d) `y = (1 + 2) / (3.0 + 2);`

Individual task

1) Answer the control questions in your report.
2) Create and execute two console projects (item 2 of theoretical information) to calculate the sum of two numbers. Write down the program code in your report.
3) Write in your report the following math expressions in C++ (tables 1.1–1.3) according to your individual variant.
4) Draw the flowcharts and create the projects to input variables and calculate values of expressions ++ (tables 1.4–1.6) according to your individual variant. Create projects on your computer and write down results to the report.

Variants of individual problems for the math expressions

Table 1.1 (basic level)

№ var.	Arithmetical expression	№ var.	Arithmetical expression
1	$Z = \frac{2t + y\cos t}{\sqrt{y + 4.831}}$	2	$D = y^2 + \frac{0.5n + 4.8}{\sin y}$
3	$Q = \frac{\sqrt{k + 2.6p\sin k}}{x - d^3}$	4	$F = \ln(d) + \frac{3.5d^2 + 1}{\cos(2y + 2.3)}$
5	$R = \frac{\sin(2t+1)^2 + 0.3}{\ln(t + y)}$	6	$L = \cos^2 c + \frac{3t^2 + 4}{\sqrt{c + t}}$
7	$U = \frac{\ln(k - y) + y^4}{e^y + 2.355k^2}$	8	$A = \frac{\sin(2y + h) + h^2}{e^h + y}$
9	$R = \frac{\sin^2 y + 0.3d}{e^y + \ln(d)}$	10	$G = \frac{9.33w^3 + \sqrt{w}}{\ln(y + 3.5) + \sqrt{y}}$
11	$P = \frac{e^{y+2,5} + 7.1h^3}{\ln\sqrt{y + 0.04h}}$	12	$U = \frac{\ln(2k + 4.3)}{e^{k+y} + \sqrt{y}}$

Table 1.2 (basic level)

№ var.	Arithmetical expression	№ var.	Arithmetical expression
1	$F = \cos(x^2 + 2) + \frac{3.5x^2 + 1}{\cos^2 y}$	2	$T = \frac{\sqrt{x + b - a} + \ln(x)}{\mathrm{ctg}(b + a)}$
3	$R = \frac{\sin(x^2 + a)^3 + 4.3^a}{\cos^3 x^4}$	4	$D = \frac{K^{-arx} - a\sqrt{6} - \cos(3ab)}{\sin^2(a\arcsin x + \ln y)}$
5	$L = \mathrm{ctg}^2 c + \frac{2x^2 + 5}{\sqrt{c + t}}$	6	$U = \frac{\ln(x^3 + y) - y^4}{e^y + 5.4k^3}$
7	$P = \frac{a^5 + \arccos(a + x^3) - \sin^4(y - c)}{\sin^3(x + y) + \lvert x - y\rvert}$	8	$A = \frac{\mathrm{tg}(y^3 - h^4) + h^2}{\sin^3 h + y}$
9	$F = \frac{\sqrt{(2 + y)^2 + \sqrt[7]{\sin(y + 5)}}}{\ln(x + 1) - y^3}$	10	$R = \frac{\cos^2 y + 2.4d}{e^y + \ln(\sin^2 x + 6)}$
11	$G = \frac{\mathrm{tg}(x^4 - 6) - \cos^3(z + xy)}{\cos^4 x^3 c^2}$	12	$F = \frac{\sqrt{\lvert x\rvert + \cos^3 x + z^4}}{\ln x - \arcsin(bx - a)}$

Table 1.3

№ var.	Arithmetical expression
1	$V1 = \sqrt{x^2 + \left(\sqrt{\operatorname{arctg}\lvert x\rvert - e^2} \,/\, \sin^2\left(x^3 + 1.8\right)\right)^4} + 2.8^{\sqrt{x}}$
2	$V2 = \left(\sin x - 5.4\right)^{3x} + \sqrt[3]{\left\lvert \lg\left(x - 1.5\right)^2 \right\rvert + x^{3.5}}$
3	$V3 = x^{2.8} \,/\, \left(\cos^2\left(x^3 - 1.5\right)^4 + \sqrt{\lvert x\rvert}\right) - \operatorname{arctg}\left(x / \ln\lvert x\rvert\right)^5$
4	$V4 = \sin^5\left(x^4 - \sqrt[3]{\lg^4\left(x^2 - \ln^2\left(x - 1.8\right)\right)}\right) + \operatorname{arctg}^2 x$
5	$V5 = \left(15.4^x - x^{3.9}\right) \Big/ \sqrt{x^2 + \lg^2 \ln^3 \lvert x^3 - 1.8\rvert} + 9^{5.3}$
6	$V6 = e^{\sqrt{\lg^3\left(x^2 - 1.8\right)^3}} + x^{4.5} \,/\, \operatorname{arctg}(x^2 + a^2)^4 - \sqrt{x^{3.2}}$
7	$V7 = \left(\cos^3 x^{1.5} + \sin^2 x^3\right)^4 \,/\, \left(\lg^2\left(\lvert x\rvert + e^{\sqrt[3]{x+1.8}}\right) + \sqrt{x}\right)$
8	$V8 = \lg^4\left(\ln^3\left(x^2 + \sqrt{\lvert x\rvert}\right) \Big/ \left(x^3 + e^x\right)^3\right)^5 - x^{3.5} \,/\, \sin^2\left(x^3 + 1.8\right)$
9	$V9 = \cos^5\left(x^2 + \operatorname{arctg}\sqrt{\lvert x - 1.8\rvert}\right) \Big/ \sin^2\left(x^2 + 1.5\right)^5 + \sqrt[3]{x^{3.5}}$
10	$V10 = \operatorname{arctg}^5\left(\sin^3\left(x^2 + 1.8\right)^5 - \sqrt{x}\right)^4 - e^{3.8} \,/\, \left(x^{4.5} + \sqrt{\lvert x\rvert}\right)$
11	$V11 = \cos^3\left(x^4 + \lg^2 \ln^3\left(\sqrt{x} - \sqrt[3]{\lvert x\rvert}\right)^2\right) + \left(4.8^{\sqrt[3]{x}} - \sqrt{\lvert x\rvert}\right)^5$
12	$V12 = \sin^8\left(\sqrt{x} + \sqrt[3]{\lvert x^2 + 1.8\rvert} \,/\, \cos^2\left(x^3 - 1.5\right)^6\right) - x^{3.7}$

Individual variants of problems with linear structure

Table 1.4 (basic level)

Input the value of variable *x*.

№ var.	Function	№ var.	Function
1	$y = \dfrac{2x^2 - \sin^2 x}{\cos(2x) + x^2} - \dfrac{x+1}{\ln x}$	2	$y = \dfrac{\ln x^2 + \cos^2 x}{\cos(2x) + x^2} + \dfrac{\sqrt[3]{x}}{x}$
3	$y = \dfrac{\ln x^2 + 2\cos^2 x}{\cos(2x)^2} + \dfrac{\sqrt[3]{x}}{x}$	4	$y = \dfrac{2\cos^2 x}{1 + x\cos(2x)} + \dfrac{0.3^x}{x \ln x - 2\sin^2 x}$
5	$y = \dfrac{x + 2x + \sin x}{\cos^2 x + x^2} + \dfrac{0{,}3^x}{\ln x}$	6	$y = \dfrac{2x + \sin x}{\cos^2 x + x^2} + \dfrac{0.5^x}{\sqrt{x}}$
7	$y = \dfrac{\sin x - x^2}{2x + 1} + \dfrac{(1+x)^x}{1 + 3x}$	8	$y = \dfrac{x - \ln x}{2x - 1} + \dfrac{2x - 1}{x^2 + 3x}$
9	$y = \dfrac{\ln x + 2x}{x^2 + 1} + \dfrac{x+1}{2x^2 + 1}$	10	$y = \dfrac{3x^2 + 2x}{\sin x + x^2} - \dfrac{2x}{(1+x^2)(1+2x)}$
11	$y = \dfrac{4x^2 + 3x}{(1+x)(1+2x)} + \dfrac{2x+1}{\sin x + 1}$	12	$y = \dfrac{(2x^2 - 1)}{x^2 + \sin} - \dfrac{2x+1}{(x+2)(x+3)}$

Table 1.5 (basic level)

Input the value of the first variable, the second value is a constant.

№ var.	Function $y = f(x)$	Parameters
1	$y = a\sin^2 b + b\cos^2 a;\ a = \sqrt[3]{\lvert b + c\rvert};\ b = \sqrt{x}$	x = 1.52; c = 5
2	$y = a^2 + b^2;\ a = \ln\lvert x\rvert;\ b = \mathrm{e}^k + a$	x = 5.3; k = 3
3	$y = \mathrm{e}^x + 5.8^c;\ c = a^2 + \sqrt{b};\ a = b^3 + \ln\lvert b\rvert$	x = 2.5; b = 7
4	$y = \sqrt[3]{\lvert a - b\rvert};\ a = \lg x;\ b = \sqrt{x^2 + t^2}$	x = 1.7; t = 3
5	$y = a^3 / b^2;\ a = \mathrm{e}^{\sqrt{\lvert x\rvert}};\ b = \left(\sin p^2 + x^3\right)$	x = 2.1; p = 2
6	$y = p^2 + t^4;\ p = x^2 - \sqrt{\lvert x\rvert};\ t = \sqrt[3]{x + a^2}$	x = 4; a = 3.7
7	$y = c^3 / \cos c;\ c = a^2 + b^2;\ a = \sqrt{\lvert x\rvert} + \mathrm{e}^{\sqrt{b}}$	x = –11; b = 12.5
8	$y = \sin^3(a + b);\ \ a = t^3 + \sqrt{b};\ b = \lg^2\lvert x\rvert$	x = 10.9; t = 2
9	$y = \mathrm{arctg}^3 x^2;\ \ x = p + k;\ k = \sqrt{p + t^2}$	t = 4.1; p = 3
10	$y = \cos^2(a + \sin b);\ a = \sqrt{\lvert x\rvert};\ b = x^4 + m^2$	m = 2; x = 1.1
11	$y = \sin^3 a + \cos^2 x;\ a = c + k^2;\ c = \mathrm{arctg}\lvert x\rvert$	k = 7.2; x = 5
12	$y = \mathrm{e}^{\sqrt{\lvert x\rvert}} + \cos x;\ \ x = a + c^3;\ a = \sin^5 b$	b = 3; c = 1.7

Table 1.6

№ var.	Problems
1	Triangle is given by coordinates of its vertexes. Calculate its area with Heron formula: $S = p(p–a)(p–b)(p–c)$, where $p = (a+b+c)/2$; a, b and c – sides of triangle. Input coordinates of vertex from the screen. Use the formula $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$ to calculate the length of segments (x_1, y_1), (x_2, y_2).
2	Find a period T and a frequency ν of oscillations in a contour. $T = 2\pi\sqrt{LC}$, $\nu = 1/T$, C is a capacity of condenser, L is inductance. Input values of C and L from the screen.
3	Calculate a length and area of a circle. Input the value of radius from the screen.
4	Calculate a hypotenuse and area of a right-angled triangle with two given catheti. Input the lengths of the catheti from the screen.
5	Find cosine of angle between vectors $\vec{a}$ = (a1, a2) and $\vec{b}$ = (b1, b2) using formula $\cos\alpha = \left(\vec{a} \cdot \vec{b}\right) / \left(\lvert\vec{a}\rvert \cdot \lvert\vec{b}\rvert\right)$. Absolute value of vector can be calculated with formula $\lvert\vec{a}\rvert = \sqrt{a_1^2 + a_2^2}$. Scalar product of vectors can be calculated with formula $\vec{a} \cdot \vec{b}$ = $a_1b_1 + a_2b_2$.

End of table 1.6

№ var.	Problems
6	Calculate the distance between point M and planes $22x-4y-20z-45=0$ and $3x-y+5z+1=0$, use the formula $\rho=\frac{\lvert ax_0+y_0+cz_0+d\rvert}{\sqrt{a^2+b^2+c^2}}$. Input coordinates of point M from the screen.
7	Calculate the roots of equation $ax^2+bx+c=0$, where $a\neq 0$, b, c are coefficients. Values a = 2, b = −8, c = −10 input from the screen.
8	Calculate the root of equation $2x/a+b-12=0$ with various values of a, b. Values of a, b input from the screen.
9	Input a number of sides of a regular polygon n and a length from center to a corner r. Calculate an area of the polygon S = (1/2)·n·sin(360°/n)·r^2 and a length of its side.
10	The rectangle is specified by the coordinates of its vertices. Calculate its perimeter. Input coordinates of vertices from the screen.
11	Calculate a value of function $W=\mathrm{sh}(x)\cdot\mathrm{tg}(x+1)-\mathrm{tg}^2(2+\mathrm{sh}(x-1))$, where $\mathrm{sh}(x)=\left(e^x-e^{-x}\right)/2$. Input x from the scree.
12	Input Cartesian coordinates of a point (x, y). Convert them to polar coordinates (ρ,ϕ), where $\rho=\sqrt{x^2+y^2}$, tg ϕ = y/x.

Lab # 2

Branching (Selection Structure). Switch statement

Purpose: to acquire practical skills in using a conditional operator `if`, multi-selection structure `switch`, jump statements and conditional operation `?:` in creation of projects in C++.

Theoretical information

1. Branching

The **selection structure** is the construct where statements can executed or skipped depending on whether a condition evaluates to `true` or `false`.

There are three selection structures in C:

1) the short form `if`;
2) the full form `if – else`;
3) selection among multiple sections `switch`.

2. Relational operators for comparing

There are six **fundamental operators for comparing** two values available: <, >, == (equal to), <=, >=, != (not equal to).

Result of comparing is a value of the `bool` type: 1 (`true`) or 0 (`false`).

3. Boolean operators

&& – the logical AND operator returns the boolean value true if both operands are true and returns false otherwise. The operands are implicitly converted to type `bool` prior to evaluation, and the result is of type `bool`. Logical AND has left-to-right associativity.

|| – the logical OR operator returns the boolean value true if either or both operands are true and returns false otherwise. The operands are implicitly converted to type `bool` prior to evaluation, and the result is of type `bool`. Logical OR has left-to-right associativity.

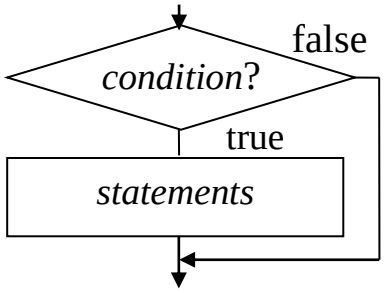
! – the logical negation operator reverses the meaning of its operand. The operand must be of arithmetic or pointer type (or an expression that evaluates to arithmetic or pointer type). The operand is implicitly converted to type `bool`. The result is true if the converted operand is false; the result is false if the converted operand is true. The result is of type `bool`.

4. IF selection control statement

There are two types of if-selection:

Single-selection IF (the short form)

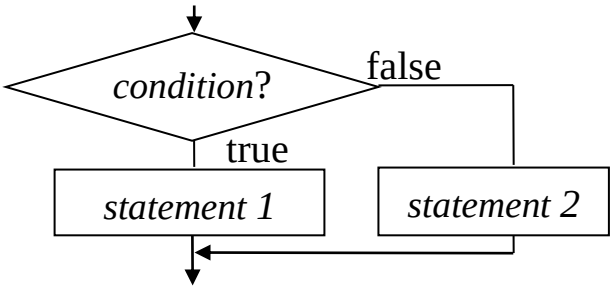
```
if ( condition )
{ statement 1;
  statement 2; etc.
}
```



If *condition (boolean expression)* is `true`, then subordinate *statement 1*, *statement 2*, *etc* are executed. If *condition* is `false`, then statements are ignored. If there is one statement only, it is not necessary to write braces `{}`. But if there are two or more statements, braces are obvious.

Double-selection IF (the full form)

```
if ( condition )
{ statement 1; etc.
}
else
{ statement 2; etc.
}
```



If *condition (boolean expression)* is `true`, then subordinate *statement 1*, *etc* are executed. If *condition* is `false`, then *statement 1* is ignored and *statement 2*, *etc* under `else` is executed.

Example. To calculate the value of function $y = \begin{cases} 1 + b^x & \text{за } x = b; \\ \dfrac{x+b}{b-x} & \text{за } x \neq b. \end{cases}$

We can use the short form of selection statement:

1)
```
if(x == b) y = 1 + pow(b, x);
if(x != b) y = (x + b)/(b - x);
```

But this way is not the best. It's better to use the full form:

2)
```
if(x == 0) y = 1 + pow(b, x); else y = (x + b)/(b - x);
```

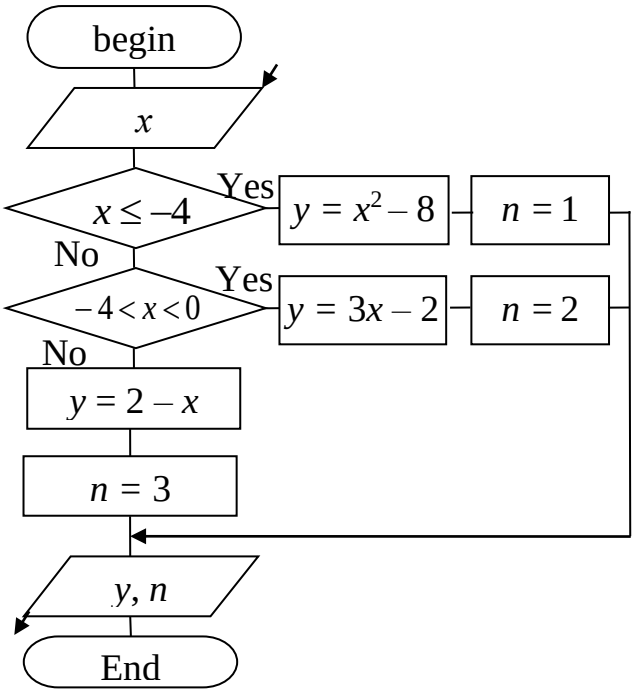
Examples of programs with branching

Example 2.1. Input x and calculate the value of function $y = \begin{cases} x^2 - 8 & \text{when } x \le -4; \\ 3x - 2 & \text{when } -4 < x < 0; \\ 2 - x & \text{when } x \ge 0. \end{cases}$

Flowchart and program code:

```
#include <iostream>
using namespace std;
int main ()
{
```

```
 double y, x; int n;
 cout<<"Input a value of x=";
 cin>>x;
 if (x <= -4) { y=x*x-8; n=1;}
 else
   if (-4<x && x<0) { y=3*x-2; n=2;}
   else  { y=2-x;   n=3;}
 cout<<"Result  y="<< y <<endl;
 cout<<"Number of condition"<<n<<endl;
 system ("pause");
 return 0;
}
```



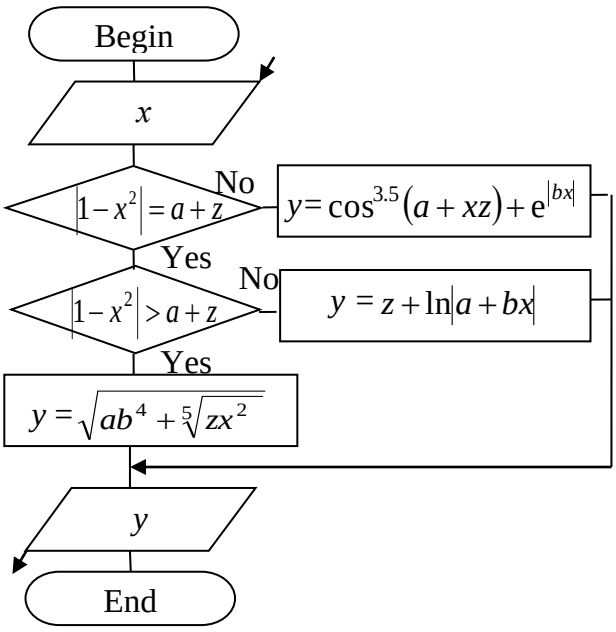
Results:

```
Input a value of x=0.4
Result  y=1.6
Condition number  3
```

Example 2.2. Input x and calculate the value of function y:

$$y=\begin{cases}\cos^{3.5}(a+xz)+e^{|bx|} & \text{when } |1-x^2|=a+z;\\ z+\ln|a+bx| & \text{when } |1-x^2|>a+z;\\ \sqrt{ab^4+\sqrt[5]{zx^2}} & \text{when } |1-x^2|<a+z,\end{cases}$$

where $a = 0.3$; $b = 1.25$; $z = (x + b)^2$.



Flowchart and program code:

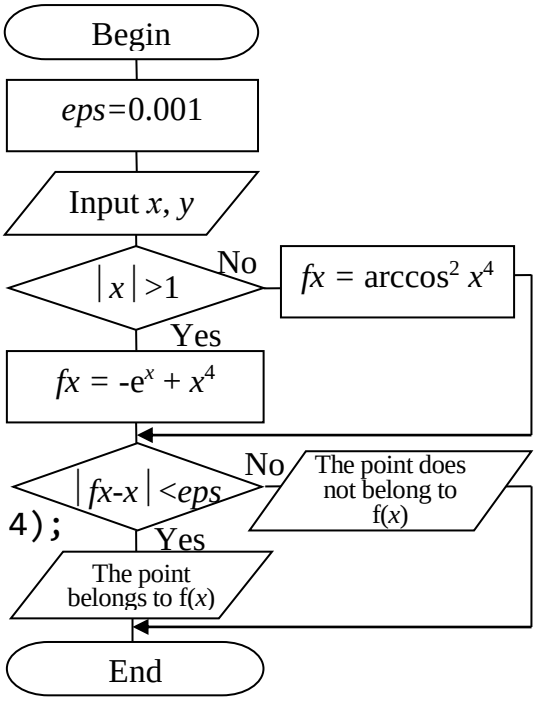
```
#include <iostream>
using namespace std;
int main ()
{
  double a=0.3, b=1.25, x, y, z;
  cout<<" Input a value of x= ";
  cin>>x;
  z = pow(x + b, 2);
  if (fabs(1-x*x) == a+z) y = pow(cos(a+x*z),3.5) + exp(fabs(b*x));
  else
    if (fabs(1-x*x) > a+z) y = z+log(fabs(a+b*x));
    else y = sqrt(a * pow(b,4) + pow(z*x*x,1./5));
  cout<<"\ Result  y="<< y <<endl;
  system ("pause");
  return 0;
}
```

Results:

```
Input a value of x=2.45
Result  y=1.77414
```

Example 2.3. Input coordinates of point B (variables x and y) and define whether this point belongs to the curve $f(x) = \begin{cases} -e^x + x^4 & \text{when } |x| > 1; \\ \arccos^2 x^4 & \text{when } |x| \le 1. \end{cases}$

Accuracy eps = 10^{-3} (that is $|f(x) - y| < \text{eps}$).



Flowchart and program code:

```
#include <iostream>
#include <math.h>
using namespace std;
int main()
{
 double x, y, fx;
 const double eps=1e-3;
 cout<<"Enter coordinate x = "; cin>>x;
 cout<<"Enter coordinate y = "; cin>>y;
 if (fabs(x) > 1) fx = -exp(x) + pow(x, 4);
 else fx = pow(acos(pow(x,4)),2);
 if (fabs(fx - y) < eps)
  cout<<"The point belongs to f(x)"<<endl;
 else
  cout<<"The point does not belong to f(x)"<<endl;
 system ("pause");
 return 0;
}
```

Results of the program:

```
Enter coordinate x = 3
Enter coordinate y = 4
The point does not belong to f(x)
```

```
Enter coordinate x = -1
Enter coordinate y = 0
The point belongs to f(x)
```

5. Conditional operator ?:

Last time you got acquainted with selection structure `if`. Another selection structure is **the inline conditional operator** (`?:`).

The conditional operator (`?:`) is a ternary operator (it takes three operands). The conditional operator works as follows.

operand1 `?` *operand2* `:` *operand3*;

The first operand is implicitly converted to `bool`. If the first operand evaluates to `true` (1), the second operand is evaluated. If the first operand evaluates to `false` (0), the third operand is evaluated. The result of the conditional operator is the result of whichever operand is evaluated – the second or the third. Only one of the last two operands is evaluated in a conditional expression.

Example 1. The absolute value of a number `i`:

```
j = (i<0) ? (-i) : (i);
```

Example 2. The biggest of two numbers `x` and `y`:

```
max = (x > y) ? x : y;
```

Example 3. You can change the button label by clicking on it:

```
button1->Text = (button1->Text == "on") ? "off" : "on";
```

*Example 4. E*xpression $(-1)^i$ for alternating series can be written without mathematical function:

```
(i%2) ? -1 : 1
```

6. SWITCH selection control statement

Switch allows selection among multiple sections of code, depending on the value of an integer expression.

```
switch ( expression )
{ case constant-expression1 : statements1; break;
  case constant-expression2 : statements2; break;
  case constant-expression3 : statements2; break;
  [default : statement;]
}
```

7. Jump Statements

C++ jump statement performs an immediate local transfer of control. The are:

```
break;    continue;    return [expression];    goto identifier;
```

The **break** statement ends execution of the nearest enclosing loop or conditional statement in which it appears. Control passes to the statement that follows the end of the statement, if any.

The **return** statement terminates the execution of a function and returns control to the calling function (or to the operating system if you transfer control from the main function). Execution resumes in the calling function at the point immediately following the call.

The **goto** statement unconditionally transfers control to the statement labeled by the specified identifier:

```
goto identifier;
```

The labeled statement designated by identifier must be in the current function. A statement label is meaningful only to a goto statement; otherwise, statement labels are ignored. Labels cannot be redeclared.

It is a good programming style to use the break, continue, and return statements instead of the goto statement whenever possible. However, because the break statement exits from only one level of a loop, you might have to use the goto statement to exit a deeply nested loop.

Examples of programs with switch

Example 2.4. Input a number of a week day and output the name of this day.

The program code:

```
#include <iostream>
#include <locale.h>
using namespace std;
int main()
{
  system("color F0");
  int n; char *s="";
  cout << "Input integer number from 1 to 7: ";  cin >> n;
  switch (n)
  {  case 1: s="Monday";    break;
     case 2: s="Tuesday";   break;
     case 3: s="Wednesday"; break;
     case 4: s="Thursday";  break;
     case 5: s="Friday";    break;
     case 6: s="Saturday";  break;
     case 7: s="Sunday";    break;
     default: { cout<<"\nError! The number is not a value from 1 to 7\n";
                system("pause>>void");return 0;
              }
   }
   cout << endl << n << " week day - " << s << endl;
   system("pause>>void");
   return 0;
}
```

```
Input integer number from 1 to 7: 3

3 day of the week - Wednesday
```

```
Input integer number from 1 to 7: 9

Error! The number is not a value from 1 to 7
```

Example 2.5. Simple calculator.

The program code:

```
#include <iostream>
using namespace std;
int main()
{system("color F0");
 double a,b,res;  char op;
 cout<< "\n Input first operand  "; cin>> a;
 cout<< "\n Input sign operations ";cin>> op;
 cout<< "\n Input second operand "; cin>> b;
 bool f=true;
 switch (op)
 { case '+' : res = a+b; break;
   case '-' : res = a-b; break;
   case '*' : res = a*b; break;
   case '/' : res = a/b; break;
   default: cout<<"\n Unknown operation!\n"; f=false;
 }
 if (f) cout<< "\n Result : " << res << endl;
 system("pause>>void");
 return 0;
}
```

```
Input first operand  25

Input sign operations  +

Input second operand  12.5

Result : 37.5
```

```
Input first operand  87

Input sign operations  /

Input second operand  3

Result : 29
```

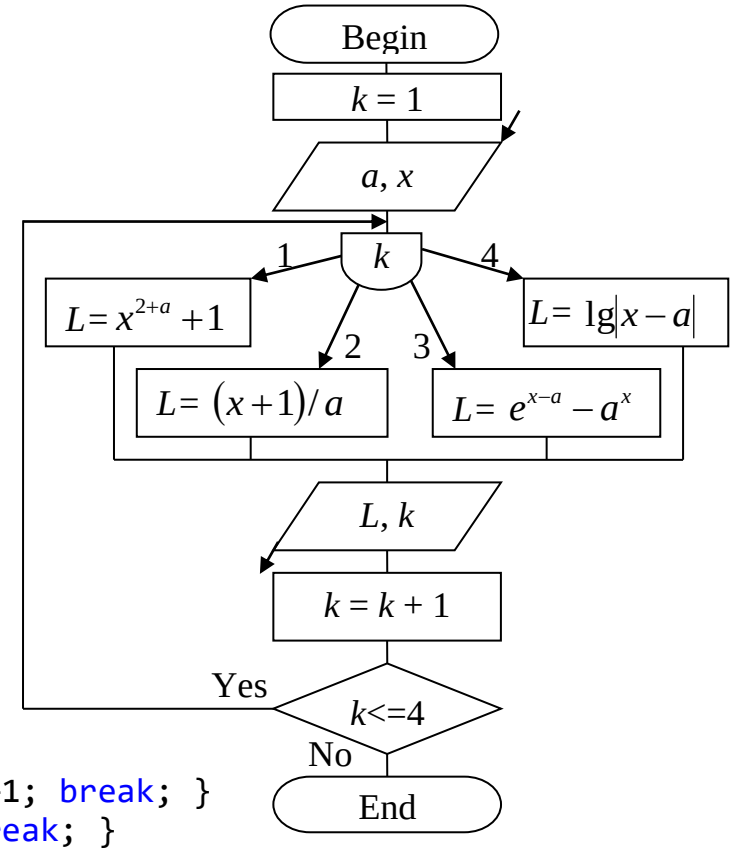
```
Input a= 0.8
Input x= 10.5

 Results:
L1 = 724.319
L2 = 14.375
L3 = 16317.5
L4 = 0.986772
```

Example 2.6. Calculate the value of function $L = \begin{cases} x^{2+a}+1 & \text{when } k=1; \\ (x+1)/a & \text{when } k=2; \\ e^{x-a}-a^x & \text{when } k=3; \\ \lg|x-a| & \text{when } k=4 \end{cases}$

for all values of parameter k.

The program code and the flowchart:




```
#include "stdafx.h"
#include <iostream>
#include <math.h>
using namespace std;
int main()
{
 system("color F0");
 double x, a, L;
 int k = 1;
 cout<< "Input a= ";
 cin>> a;
 cout<< "Input x= ";
 cin>> x;
 cout<< "\n Results:\n";
M1:
  switch (k)
  { case 1: { L = pow(x,2+a)+1; break; }
    case 2: { L = (x+1)/a; break; }
    case 3: { L = exp(x-a) - pow(a,x); break; }
    case 4: { L = log10(fabs(x-a)); break; }
  }
 cout<< "L" << k << " = " << L << endl;
 k++;
 if (k <= 4) goto M1;
 system("pause>>void");
 return 0;
}
```

Example 2.7. Input x and calculate the value of function

$$y = \begin{cases} \sin\left(e^{|a+c|}\right) + x^{2+b} & \text{when } x < a; \\ \sqrt[3]{a+|5.3b|}+c & \text{when } a \le x < b; \\ \cos^2(a+b)+\sin x^2 & \text{when } x > b \end{cases}$$

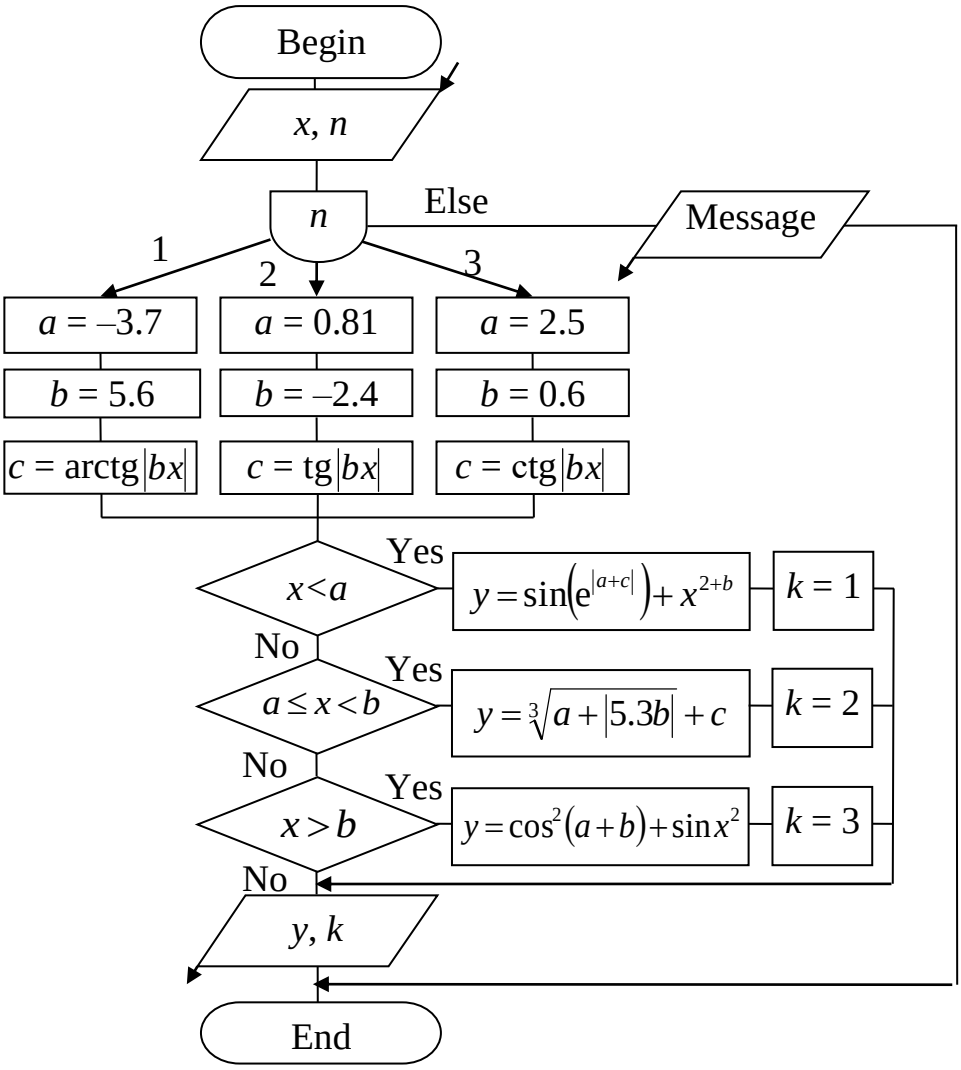
for one of three options for parameters whose number must be inputted:

1) $a = -3{,}7$; $b = 5{,}6$; $c = \operatorname{arctg}|bx|$;

2) $a = 0{,}81$; $b = -2{,}4$; $c = \operatorname{tg}|bx|$;

3) $a = 2{,}5$; $b = 0{,}6$; $c = \operatorname{ctg}|bx|$.

The flowchart:



The program code:

```
#include "stdafx.h"
#include <iostream>
#include <math.h>
using namespace std;
int main()
{
  system("color F0");
  double x, y, a, b, c;
  int k, n;
  cout<< "Input a number of the variant (int value) 1, 2 or 3: ";
  cin>> n;
  cout<< "Input a value of x= ";
  cin>> x;
  switch (n)
  {
    case   1:  a=-3.7;  b=5.6;  c=atan(fabs(b*x)); break;
    case   2:  a=0.81;  b=-2.4; c=tan(fabs(b*x)); break;
    case   3:  a=2.5;   b=0.6;  c=1/tan(fabs(b*x)); break;
```

```
    default: { cout<< "Incorrect value of the variant!"<<endl;
               system("pause>>void");
               return 0;
             }
  }
  cout<< endl << "Results:" << endl;
  if (x<a)  { y=sin(exp(fabs(a + c)))+pow(x+b,2);  k=1;}
  if (x>=a && x<b)  { y=pow(a+fabs(5.3*b),1.0/3)+c;  k=2;}
  if (x>=b) { y=pow(cos(a+b),2)+sin(x*x);  k=3;}
  cout<< "y= " <<y<< " condition #"<<k<<" is true"<<endl;
  system("pause>>void");
  return 0;
}
```

Results:

```
Input a number of the variant (int value) 1, 2 or 3: 3
Input a value of x= 3.45

Results:
y= 0.382101 condition #3 is true
```

```
Input a number of the variant (int value) 1, 2 or 3: 6
Input a value of x= 14
Incorrect value of the variant!
```

Control questions

1) What is the selection?
2) What instructions/commands are used in C++ for selection?
3) What logical operations do you know?
4) Calculate the logical expression: `(-3 >= 5) || (7 < 9) && (0 < 3).`
5) What is the value of variable `w` after the following instruction:
   `bool w = 2*5 <= 17 % 3; .`
6) Write the following function in C+, use: 1) the short form of `if`; 2) the full form of `if`

$$y = \begin{cases} \sin x^2 & \text{за } x > 0.5; \\ \cos^2 x & \text{за } x \le 0.5. \end{cases}$$

7) Which instructions do not contain errors?
   a) `if (x<=6) y=2*x; else y=cos(x);`
   b) `if y<=x then y:=exp(x*y);`
   c) `if (a<>0) if(b<>0) y=2*x;`
   d) `if (x>0) y=ln(x) else y=x;`
8) What is the value of variable x after the program code?
   a)
```
float x=1.5;
if (x<=0.5) x=7.7;
```
   b)
```
float x=1.5;
if (x<=0.5) x=7.7; else x=3;
```
9) What is the value of variable z after the program code?
```
float z, x=2.5;
if (x>=0.5) z=7.7; else z=5.5;
```

10) What is the value of `f` after the execution of the following code:

```
int f = 1, n = 3, i = 2;
M1: if (i > n)  goto PP;
    f = f * i;
    i++;
    goto M1;
PP: ;
```

11) What is the value of y after the execution of the following code:

a)
```
double y=0;
int n=1;
switch (n)
{
  case 1:  y=n/4.; break;
  case 2:  y=n*n;  break;
  case 3:  y=n;    break;
}
```

b)
```
double y=0;
int n=3;
switch (n)
{
  case 1:  y=n/4.;
  case 3:  y=n*n;
  case 5:  y=n+1;
}
```

c)
```
double y=0;
int n=4;
switch (n)
{
  case 2:  { y=n/4.; break; }
  case 5:  { y=n*n;  break; }
  case 9:  { y=n;    break; }
}
```

d)
```
double y=0;
int n=1;
switch (n)
 {
  case 1: { y=n/4; }
  case 3: { y=n*n; }
  case 5: { y=n+1; }
}
```

12) In the following program code `switch` statement evaluates an expression … (write down this expression).

```
int  nom = pow(2, 3);
switch (nom)
{
  case  2 : y=d;     break;
  case  8 : y=d*exp(x); break;
  case 10 : y=d*x;   break;
}
```

Individual task

1) Answer the control questions in your report.

2) Draw flowcharts and create projects to input variables and calculate values of expressions in C++ according to your individual variant (table 2.1–2.3). Create projects on your computer and write down results.

3) Draw the flowcharts and create the projects to input variable *x* and calculate the following functions in C++ according to your individual variant (table 2.4–2.6). Create projects on your computer and write down results to the report.

Individual variants

Table 2.1 (basic level)

№ var.	Function	№ var.	Function
1	$Y=\begin{cases}x^2+1 & \text{when } x<0;\\ x^2-1 & \text{when } 0\le x<2;\\ x & \text{when } x\ge 2\end{cases}$	2	$Y=\begin{cases}2x+2 & \text{when } x<-3;\\ 2x-2 & \text{when } -3\le x\le 0;\\ x^2 & \text{when } x>0\end{cases}$
3	$Y=\begin{cases}6x+8 & \text{when } x\le -5;\\ x-2 & \text{when } -5<x\le 3;\\ 2x^2 & \text{when } x>3\end{cases}$	4	$Y=\begin{cases}2x-1 & \text{when } x\le -4;\\ x^2+2 & \text{when } -4<x\le 5;\\ x+3 & \text{when } x>5\end{cases}$
5	$Y=\begin{cases}6x^3-8 & \text{when } x\le -8;\\ x^3-8 & \text{when } -8<x<0;\\ 2x^2 & \text{when } x\ge 0\end{cases}$	6	$Y=\begin{cases}2x^3+3x & \text{when } x\le -1;\\ x^2-4 & \text{when } -1<x<0;\\ x^3 & \text{when } x\ge 0\end{cases}$
7	$Y=\begin{cases}4x^2+2x & \text{when } x\le -12;\\ 2x^2+2x & \text{when } -12<x<3;\\ x+1 & \text{when } x\ge 3\end{cases}$	8	$Y=\begin{cases}x^3-1 & \text{when } x\le -4;\\ 2x-1 & \text{when } -4<x\le 3;\\ 3x^3 & \text{when } x>3\end{cases}$
9	$Y=\begin{cases}4x+3 & \text{when } x\le -2;\\ 2x^2-4 & \text{when } -2<x<4;\\ x^2-2 & \text{when } x\ge 4\end{cases}$	10	$Y=\begin{cases}2x+4 & \text{when } x\le -1;\\ x-4 & \text{when } -1<x<0;\\ x^3+4 & \text{when } x\ge 0\end{cases}$
11	$Y=\begin{cases}4x^2+2x & \text{when } x<-2;\\ 2x-1 & \text{when } -2\le x<3;\\ x^3+3 & \text{when } x\ge 3\end{cases}$	12	$Y=\begin{cases}3x^2+2x & \text{when } x<-3;\\ 2x+1 & \text{when } -3\le x<8;\\ 3x & \text{when } x\ge 8\end{cases}$

Table 2.2

№ var.	Function	№ var.	Function
1	$y=\begin{cases}a^3+\arcsin\left(\cos^3 bx\right) & \text{when } x\le a;\\ \sqrt{(a+bx)-2}+\sin x & \text{when } a<x<b;\\ \operatorname{tg}^2(a+bx+z) & \text{when } x\ge b,\end{cases}$ where $a=2.5;\ b=3.5;\ z=\sin(bx)$	2	$y=\begin{cases}a^{2b}x^2+\sqrt{b^4+2.7} & \text{when } \lvert x\rvert<0.7;\\ \operatorname{arctg}\left(3^x-px\right) & \text{when } \lvert x\rvert=0.7;\\ \sqrt[3]{\ln\lvert a-px\rvert+4.3} & \text{when } \lvert x\rvert>0.7,\end{cases}$ where $a=0.54;\ b=0.34;\ p=ax+b$
3	$y=\begin{cases}(a+z)\cos^2\left(bx+x^3\right) & \text{when } x<a;\\ a\ln(zx)+\sin^2\left(b^2\right) & \text{when } a\le x\le b;\\ \sqrt[3]{0.3b+\sqrt{\lvert(a-z^2)\rvert}} & \text{when } x>b,\end{cases}$ where $a=0.1;\ b=3.25;\ z=\cos^2(x)$	4	$y=\begin{cases}\operatorname{ctg}\left(x^2e^{3k}\right)+\ln\lvert rx\rvert & \text{when } x=rs;\\ \sqrt[5]{x^2}+\sqrt{\lvert\sin k\rvert} & \text{when } x>rs;\\ \operatorname{tg}(kx+\operatorname{tg}(rs)) & \text{when } x<rs,\end{cases}$ where $r=2.4;\ s=5;\ k=0.5$

End of table 2.2

№ var.	Function	№ var.	Function
5	$y=\begin{cases}\dfrac{(2a+1)^2}{3.71-x^2} & \text{when } z>-0.5;\\ \sin^3\sqrt{bz}-ax & \text{when } -0.5\le z\le 10^{-3};\\ \dfrac{\operatorname{tg}(z+x)-e^x}{3.5abx} & \text{when } z>10^{-3},\end{cases}$ where $a=0.3;\ b=0.7;\ z=\cos(x+2)$	6	$y=\begin{cases}e^{ax}+f\cos^5 bx & \text{when } x\le a;\\ \cos^2\sqrt{bx}-\ln(fx) & \text{when } a<x\le b^2;\\ \cos^2(a+bfx) & \text{when a } x>b^2,\end{cases}$ where $a=1.5;\ b=1.44;\ f=\sqrt{bx}$
7	$y=\begin{cases}a\cos^2 x+b\sin^2 zx & \text{when } x\le a;\\ a\cdot\operatorname{tg}(\sin^2 bx+z) & \text{when } a<x\le 4.5b;\\ \ln(ax-b)+z^2 & \text{when } x>4.5b,\end{cases}$ where $a=1.5;\ b=0.7;\ z=\operatorname{tg}\lvert\operatorname{tg}(bx)\rvert$	8	$y=\begin{cases}\ln\lvert bzx\rvert+za^{2.5} & \text{when } a^3<x\le b;\\ ax^2+bz^a+\sin^2 zx & \text{when } x\le a^3;\\ \cos(ax+b)+\ln\lvert zx\rvert & \text{when } x>b,\end{cases}$ where $a=0.5;\ b=0.7;\ z=0.2$
9	$y=\begin{cases}\sin\left(e^{a+b}\right)+x^2 & \text{when } a+b>x;\\ \operatorname{arctg}(abc)+\sqrt[3]{x} & \text{when } a+b=x;\\ \arcsin\left(\cos^2\left(\sqrt{\lvert x\rvert}\right)\right) & \text{when } a+b<x,\end{cases}$ where $a=0.5;\ b=1.5;\ c=3.2$	10	$y=\begin{cases}\ln\left(\lvert\lg\lvert kx+mn\rvert\rvert\right) & \text{when } x>\lvert m+n\rvert;\\ \sin(kmx)+\sqrt{\lvert nx\rvert} & \text{when } x=\lvert m+n\rvert;\\ e^{\cos x}+e^{m+n} & \text{when } x<\lvert m+n\rvert,\end{cases}$ where $m=2.1;\ n=1.9;\ k=8.5$
11	$y=\begin{cases}a\sin^2 x+\cos(zx) & \text{when } x<\ln(a);\\ \cos^3(a+zx) & \text{when } \ln(a)\le x\le b;\\ \sqrt{2.5a^3+(b-zx^2)^6} & \text{when } x>b,\end{cases}$ where $a=0.1;\ b=3.25;\ z=\cos^2(x)$	12	$y=\begin{cases}\sin(bm+\cos(nx)) & \text{when } \lvert bm\rvert>x^2;\\ \cos(bm-\sin x) & \text{when } \lvert bm\rvert<x^2;\\ \sqrt{e^{\lvert\cos x\rvert}+\sqrt{\lvert bmx\rvert}} & \text{when } \lvert bm\rvert=x^2,\end{cases}$ where $m=0.5;\ b=-2;\ n=0.2$

Table 2.3

№ var	Problems
1	Input two numbers and define what is bigger: the sum of squares or the squared sum of these numbers. Output the answer in the form of a message
2	Input an angle in radians and define what is bigger: sine or cosine value of the angle. Output the answer in the form of a message
3	Input three numbers and define which value of them is between two other
4	Input three numbers and define the smallest of them
5	Input three sides of a triangle *a*, *b* and *c*, and define whether this triangle is rectangular. Output the answer in the form of a message
6	Input coordinates of points *A* (x_0, y_0) and *B* (x_1, y_1) and define which of these points – *A* or *B* – is farthest from the origin *O*(0,0). Output the answer in the form of a message
7	Input coordinates of point *B* (variables *x* and *y*) and define whether this point belongs to the curve $f(x) = 6x^7–4.5x^5+4x^2$. Accuracy $eps = 10^{-3}$ (that is $\lvert f(x) – y\rvert < eps$)
8	Input three numbers and raise positive numbers to the second power (remain negative numbers unchanged)

End of table 2.3

№ var	Problems
9	Input coordinates of point *A* (*x* and *y*) and define to which coordinate quarter this point belongs. Output the answer in the form of a message
10	Input coordinates of point A (*x* and *y)* and define whether this point is inside the circle with the radius *R*. Center of circle is the origin *O*(0,0). Output the answer in the form of a message
11	Enter three integers (the lengths of the sides of the triangle) and determine if it is possible to construct a triangle with these sides.
12	Enter the value of the side of the square A and the radius of the circle R and determine which of these figures has larger area.

Table 2.4 (basic level)

№ var.	Function	№ var.	Function
1	$y=\begin{cases}7.8x^3-\operatorname{tg}(3.1x^2+4x) & \text{where } k=1;\\ e^{0.85\sqrt{x}}(x^2+3) & \text{where } k=2;\\ \sin(2x+\pi)+e^{4x} & \text{where } k=3;\\ x\dfrac{\sqrt[3]{x+\cos(\pi/2+x)}}{x^{2^x}+0.1\cdot10^{-3}} & \text{where } k=4\end{cases}$	2	$y=\begin{cases}\dfrac{\operatorname{tg}\lvert 1+e^{x+1.2}\rvert}{x+\arcsin x} & \text{where } n=1;\\ \sqrt[3]{\cos\lvert\pi+x\rvert} & \text{where } n=2;\\ \dfrac{1+x^{x+1}-\lg x}{x^3+\ln\lvert x\rvert} & \text{where } n=3\end{cases}$
3	$y=\begin{cases}\dfrac{4x^2t}{2x-3t+2} & \text{where } n=1;\\ 6.2x-\dfrac{\ln\sqrt{x^2+0.1}}{\sqrt{\lvert 2x-\cos x\rvert}} & \text{where } n=2;\\ 8.3t^3+x-0.2 & \text{where } n=3\end{cases}$	4	$y=\begin{cases}2x+1 & \text{where } k=1;\\ \sqrt[3]{1-x^4} & \text{where } k=2;\\ \lg\lvert x+5\rvert & \text{where } k=3;\\ \ln\left\lvert\dfrac{1+x}{x^3+\cos x}\right\rvert & \text{where } k=4\end{cases}$
5	$y=\begin{cases}\operatorname{arctg}(2x+1)+1 & \text{where } k=1;\\ \sqrt[3]{1+x^4} & \text{where } k=2;\\ \cos\left(\dfrac{\pi}{2}-x^x\right)+e^{\lvert x+5\rvert} & \text{where } k=3;\\ \lg\dfrac{1+x}{x^3+\sqrt{\lvert x\rvert}} & \text{where } k=4\end{cases}$	6	$y=\begin{cases}\sin e^{x+1.2} & \text{where } n=1;\\ \sqrt[5]{\lg\lvert 1+x\rvert}, & \text{where } n=2;\\ \operatorname{tg}\lvert\cos x+5\pi/4\rvert & \text{where } n=3;\\ \dfrac{1+x^{x+1}-x}{x^3+\ln\lvert x\rvert} & \text{where n}=4\end{cases}$
7	$y=\begin{cases}1/x+\operatorname{arctg}^2x^3 & \text{where } M=1;\\ 2^{x-1}+\sin^2 x+\lg\lvert x\rvert & \text{where } M=2;\\ \sqrt{\lvert 1+x\rvert}-\sqrt[3]{x} & \text{where } M=3\end{cases}$	8	$y=\begin{cases}10^{-3}+\sin x^3 & \text{where } z=1;\\ \sqrt{1+x}+\sin^2 x & \text{where } z=2;\\ \lg(1/x+\sqrt{x}) & \text{where } z=3\end{cases}$

End of table 2.4

№ var.	Function	№ var.	Function
9	$y=\begin{cases}\sqrt[5]{x+1} & \text{where } k=1;\\ \operatorname{tg}(\cos x+\pi/2) & \text{where } k=2;\\ e^{2x^2}+\sqrt{\lvert 1-x\rvert} & \text{where } k=3;\\ \sin^2(x^2+3) & \text{where } k=4;\\ \cos 3x^2 & \text{where } k=5\end{cases}$	10	$y=\begin{cases}2x^2+\lg\lvert x\rvert & \text{where } n=1;\\ \cos^2 x+2.8\sqrt[3]{x} & \text{where } n=2;\\ \sin^2\sqrt{\lvert x\rvert} & \text{where } n=3;\\ \ln\left\lvert\dfrac{x+1}{4}\right\rvert & \text{where } n=4\end{cases}$
11	$t=\begin{cases}\sqrt{\lvert 2^x-x^2\rvert+0.5} & \text{where } k=1;\\ 1+\operatorname{arctg}(x) & \text{where } k=2;\\ \sqrt[5]{\pi^2+x^2} & \text{where } k=3;\\ \lg\lvert 6.5-x^4\rvert & \text{where } k=4\end{cases}$	12	$y=\begin{cases}2^{x+1}+1 & \text{where } k=1;\\ \sqrt[3]{e^{x^2}+x^4} & \text{where } k=2;\\ \lg\lvert\sin(\pi-x)\rvert & \text{where } k=3;\\ \operatorname{tg}\dfrac{1+x}{x^3+x^x} & \text{where } k=4\end{cases}$

Table 2.5

№ var.	Function	Variants of parameters
1	$y=\begin{cases}\ln\left(\lvert\lg\lvert kx+mn\rvert\rvert\right) & \text{where } 3x>\lvert m+n\rvert\\ \sin(kmx)+\sqrt{\lvert nx\rvert} & \text{where } 3x=\lvert m+n\rvert\\ e^{\cos x}+e^{m+n} & \text{where } 3x<\lvert m+n\rvert\end{cases}$	1 $k=4;\ m=-14.7;\ n=-0.6$ 2 $k=3;\ m=6.5;\ n=3.15$ 3 $k=5;\ m=-12;\ n=0.45$
2	$y=\begin{cases}abx-\cos^2(zx) & \text{where } x<3.5a;\\ (a-x)^2-\ln(z+x) & \text{where } 3.5a\le x\le b;\\ \sqrt{bx-a+zx^2} & \text{where } x>b\end{cases}$	1 $a=0.4;\ b=2.3;\ z=e^{2x}$ 2 $a=0.2;\ b=0.8;\ z=e^{x}$ 3 $a=0.7;\ b=8.1;\ z=0.8$
3	$y=\begin{cases}e^{ax}-3.5\cos^2(z+bx) & \text{where } x\le a;\\ a+\ln\lvert a+bx\rvert-2x & \text{where } a<x\le b^{3.5};\\ a+\cos^{3.5}(a+bxz) & \text{where } x>b^{3.5}\end{cases}$	1 $a=-1;\ b=3.4;\ z=\operatorname{tg} bx$ 2 $a=-3.2;\ b=5.5;\ z=\operatorname{tg} bx^2$ 3 $a=-5.2;\ b=7.2;\ z=\operatorname{tg} bx^3$
4	$y=\begin{cases}a\sin^2 x+b\cos(zx) & \text{where } x<-\ln(a);\\ a^b-\cos^3(a+zx) & \text{where } -\ln(a)\le x\le b;\\ \sqrt{2.5a^3+(b-zx^2)^6} & \text{where } x>b\end{cases}$	1 $a=0.2;\ b=0.5;\ z=e^{ax}$ 2 $a=0.15;\ b=0.2;\ z=e^{2ax}$ 3 $a=0.9;\ b=5;\ z=e^{2.5ax}$
5	$y=\begin{cases}a^2x^3+\sqrt{b^4+1.7} & \text{where } \lvert x\rvert<0.2;\\ \operatorname{arctg}\left(2^x-\lvert p\rvert\right) & \text{where } \lvert x\rvert=0.2;\\ \sqrt[3]{\ln\lvert a\rvert+4.3}+x & \text{where } \lvert x\rvert>0.2\end{cases}$	1 $a=0.5;\ b=1.5;\ p=-4$ 2 $a=-1;\ b=0.5;\ p=-4$ 3 $a=-2;\ b=0;\ p=-4$

End of table 2.5

№ var.	Function	Variants of parameters
6	$y=\begin{cases}2.8\sin^2 ax-bx^3z & \text{where } x<a;\\ z\cos(ax+b)^2+\ln(z) & \text{where } a\le x\le b^2;\\ e^{2.5ax}+zabx & \text{where } x>b^2\end{cases}$	1 $a=-5;\ b=2.5;\ z=\ln\lvert bx^3\rvert$ 2 $a=3;\ b=5;\ z=\ln\lvert bx\rvert$ 3 $a=-10;\ b=3,\ z=\ln\lvert bx^2\rvert$
7	$y=\begin{cases}a\cos^2 x+b\sin zx & \text{where } x\le a;\\ \operatorname{tg}(ax+z)+\sin^2 bx & \text{where } a<x\le 1.5b;\\ \ln(ax-b)+z^2 & \text{where } x>1.5b\end{cases}$	1 $a=4.5;\ b=8.4;\ z=\operatorname{tg}(bx)^2$ 2 $a=8.2;\ b=15.2;\ z=\operatorname{tg}(bx)^2$ 3 $a=1.7;\ b=0.5;\ z=\operatorname{tg}(bx^2)$
8	$y=\begin{cases}\ln\lvert mx+n\rvert & \text{where } x^2>m+n\\ e^{\cos\lvert mx-n\rvert} & \text{where } x^2=m+n\\ \sqrt[3]{k^2+\cos^2 x} & \text{where } x^2<m+n\end{cases}$	1 $k=3.1;\ m=5.15;\ n=-1.15$ 2 $k=0.78;\ m=-2.4;\ n=4.36$ 3 $k=1.1;\ m=0.8;\ n=0.41$
9	$y=\begin{cases}a\sin^2 x+b\cos(zx+a) & \text{where } x<a^3;\\ (a+bx)^2-\sin(a+zx) & \text{where } a^3\le x\le b;\\ \sqrt{x-(\sin(bx+z))} & \text{where } x>b\end{cases}$	1 $a=1.2;\ b=7.2;\ z=e^x$ 2 $a=-1.5;\ b=3.2;\ z=e^{2x}$ 3 $a=1.7;\ b=5.5;\ z=e^3$
10	$y=\begin{cases}\sin\left(e^{a+b}\right)+x^2 & \text{where } a+b>x;\\ \operatorname{arctg}(abc)+\sqrt[3]{x} & \text{where } a+b=x;\\ \arcsin\left(\cos^2\left(\sqrt{\lvert x\rvert}\right)\right) & \text{where } a+b<x\end{cases}$	1 $a=7{,}2;\ b=-1.3;\ c=2.5$ 2 $a=1.47;\ b=3.81;\ c=2.8$ 3 $a=4.8;\ b=10.6;\ c=2.7$
11	$y=\begin{cases}\sqrt{\lvert ax-\cos^2 b^3x+5.1c^2\rvert} & \text{where } \lvert 1-x^2\rvert=a+c\\ e^{0.04x}+\ln\lvert b^5\cos x\rvert & \text{where } \lvert 1-x^2\rvert>a+c\\ \cos^2\left(b^3x^2\right)+\ln\lvert bx-a^2\rvert & \text{where } \lvert 1-x^2\rvert<a+c\end{cases}$	1 $a=3.5;\ b=-0.73;\ c=2.5$ 2 $a=15{,}4;\ b=-5.6;\ c=3.5$ 3 $a=5.1;\ b=4;\ c=2.7$
12	$y=\begin{cases}\dfrac{(2z+1)^2}{3.71-x^2} & \text{where } z>-0.5;\\ \sin^3 z-\sin\dfrac{z}{3\pi} & \text{where } -0.5\le z\le 10^{-3};\\ \dfrac{\operatorname{tg}(z+x)-e^x}{3.5x} & \text{where } z>10^{-3}\end{cases}$	1 $z=\arcsin x^3$ 2 $z=\arccos^2 x$ 3 $z=\operatorname{tg} x$

Table 2.6

№ var.	Problems	Variants of parameters
1	Define, whether the point A with given coordinates lies inside the area bounded by the parabola $y = 2 - x^2$ and abscissa. Output the result as a message	1 $x = 3.5$; $y = 7.2$ 2 $x = -0.5$; $y = 1.2$ 3 $x = 0.72$; $y = -3.12$
2	Output the number, which value is between two other	1 $a = 3$; $b = 3.5$; $c = -2.1$ 2 $a = 2.1$; $b = -6.55$; $c = 0.1$ 3 $a = -9$; $b = -3.7$; $c = -0.1$
3	Define whether the point with coordinates x and y lies inside the circle with radius R and center (0, 0)	1 $x = 3$; $y = -7$; $R = 5$; 2 $x = 12$; $y = 11$; $R = 16$; 3 $x = -9$; $y = 6$; $R = 11$.
4	Three sides of triangle a, b and c are given. Is this triangle right-angle?	1 $a = 3$; $b = 3.5$; $c = -2.1$ 2 $a = 2.1$; $b = -6,55$; $c = 0.1$ 3 $a = -9$; $b = -3,7$; $c = -0.1$
6	The values of three numbers A, B, C are given. Multiply by 2 those of them, for which $A + B + C > 0$, otherwise change them with zero	1 $A = -3$; $B = 3,5$; $C = 0,1$ 2 $A = 58$; $B = 27$; $C = -87$ 3 $A = -8$; $B = -35$; $C = 42$
7	Define, which of points $A(x_0, y_0)$ or $B(x_1, y_1)$ is nearer to point $O(0,0)$	1 $x_0 = 2$; $y_0 = 2$; $x_1 = -4$; $y_1 = 0$ 2 $x_0 = 8$; $y_0 = 9$; $x_1 = 12$; $y_1 = 1$ 3 $x_0 = -3$; $y_0 = 0.9$; $x_1 = 2$; $y_1 = 3$
8	For triangles with sides a, b and c define whether they are isosceles	1 $a = 3$; $b = 3.5$; $c = 1.1$ 2 $a = 3$; $b = 6.55$; $c = 6.55$ 3 $a = 0.9$; $b = 0.9$; $c = 0.9$
9	The values of three integer numbers (a, b, c) are given. Define, whether they make Pythagorean triple ($c^2 = a^2 + b^2$)	1 $a = 3$; $b = 5$; $c = 4$ 2 $a = 3$; $b = 8$; $c = 11$ 3 $a = 13$; $b = 5$; $c = 12$
10	The values of three points $A_1(x_1, y_1)$, $A_2(x_2, y_2)$ and $A_3(x_3, y_3)$ are given. Define, whether these points belong to one straight line	1 $x_1 = 2$; $y_1 = 2$; $x_2 = 4$; $y_2 = 0$; $x_3 = -2$; $y_3 = 6$ 2 $x_1 = 8$; $y_1 = 9$; $x_2 = 4$; $y_2 = 0$; $x_3 = 5$; $y_3 = 1$
11	Three numbers - x, y, and z are given. Find and display on the screen those of these numbers, which by module are larger than π	1 $x = -7.2$; $y = 3.14$; $z = -2.5$ 2 $x = -4$; $y = -3$; $z = 9.15$ 3 $x = 3.14$; $y = -3.4$; $z = 0.59$
12	The lengths of the sides of the triangle are given. Determine if it is possible to construct a triangle with these sides	1 $a = 8$; $b = 13.5$; $c = 1.1$ 2 $a = 3$; $b = 3.56$; $c = 0.55$ 3 $a = 1.9$; $b = 0.9$; $c = 0.9$

Lab # 3

Iteration statements. For loop statement

Purpose: to get practical skills in organization of cyclic calculations in C++ with the use of `for` iteration.

Theoretical information

Loops

Loops are used to repeat (iterate) statements a certain number of times or while a condition is true.

C++ provides next iteration statements: `for`, `while` and `do-while`. Each of these iterates while the condition of continuation evaluates to true, or until loop termination is forced with a break statement.

For loop

For loop is a C++ construction that allows to repeat a statement, or set of statements, for a known number of times or while some condition is true. The syntax of **for loop** is:

```
for ( initialization; condition of continuation; modification of parameters )
{
    statementblock;
}
```

where: *initialization* declares and initializes the loop control variable, for example:

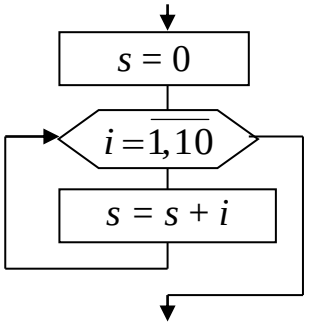
```
int i = 0;
```

condition of continuation is a test that will stop the loop as soon as it is false. Or, in other words, the repetition will continue as long as the condition is true;

modification of parameters is a statement that modifies the loop control variable appropriately, for example: `i++;`

statementblock is either a single statement or a group of statements inside the braces { ... }.

The simple example for calculation of a sum of first 10 natural numbers $S = 1 + 2 + 3 + ... + 10 = \sum_{i=1}^{10} i$:




```
int  s = 0;
for(int i = 1; i <= 10; i++)  s += i;
```

The algorithm of calculation:

1) to initialize the sum and loop parameter (index) (`s = 0, i = 1`);

2) to test the loop-termination criteria (`i <= 10`);

3) to execute the next iteration if the condition is `true`: add the current value of `i` to sum (`s += i`);

4) loop parameter increases by 1 (`i++`).

Then return to step 2. If the condition in 2) is false the loop terminates.

Syntax name	When executed	Description
Initialization (init-expression)	Before any other element of the for statement, init-expression is executed only once. Control then passes to cond-expression	Often used to initialize loop indices. It can contain expressions or declarations
condition of continuation (cond-expression)	Before execution of each iteration of statement, including the first iteration. Statement is executed only if cond-expression evaluates to true (nonzero)	An expression that evaluates to an integral type or a class type that has an unambiguous conversion to an integral type. Normally used to test for loop-termination criteria
modification of parameters (loop-expression)	At the end of each iteration of statement. After loop-expression is executed, cond-expression is evaluated	Normally used to increment loop indices

The following examples show different ways to use the for loop statement.

```
#include <iostream>
using namespace std;

int main()
{
    // The counter variable can be declared in the init-expression.
    for (int i = 0; i < 2; i++) cout<< i;   // Output: 01
    // The counter variable can be declared outside the for loop.
    int i;
    for (i = 0; i < 2; i++) cout<< i;       // Output: 01
    // These for loops are equivalent to following while loop:
    i = 0;
    while (i < 2) cout<< i++;               // Output: 01
}
```

Init-expression and loop-expression can contain multiple statements separated by commas. For example:

```
int main()
{   system("color F0");
    int i, j;
    for (i = 5, j = 10; i + j < 20; i++, j++)
       cout<< "i + j = " << (i + j) << '\n';
    system ("pause");
    return 0;
}
```

```
i + j = 15
i + j = 17
i + j = 19
```

Loop-expression can be incremented or decremented, or modified in different ways.

```
int main()
{ system("color F0");
```

```
  for (int i=10; i>0; i--) cout<< i << ' ';
  cout<< endl;
  for (int i=10; i<20; i+=2) cout<< i << ' ';
  system ("pause>>void");
  return 0;
}
```

```
10 9 8 7 6 5 4 3 2 1
10 12 14 16 18
```

For loop terminates when a `break`, `return`, or `goto` (to a labeled statement outside the `for loop`) within statement is executed. A `continue` statement in `for loop` terminates only the current iteration.

If cond-expression is omitted, it is considered true and the `for loop` cannot be terminated without a `break`, `return`, or `goto` statement.

A variable declared inside `for loop` go out of scope after the `for loop` ends. For example:

```
for (int i = 0 ; i < 5 ; i++)
{
   // do something
}
   // i is now out of scope
```

A loop can be nested inside of another loop. C++ allows 256 levels of nesting.

The syntax for a nested for loop statement in C++ is as follows:

```
for ( init; condition; increment )
{
   for ( init; condition; increment )
   {
      statements1;
   }
   statements2; // you can put more statements
}
```

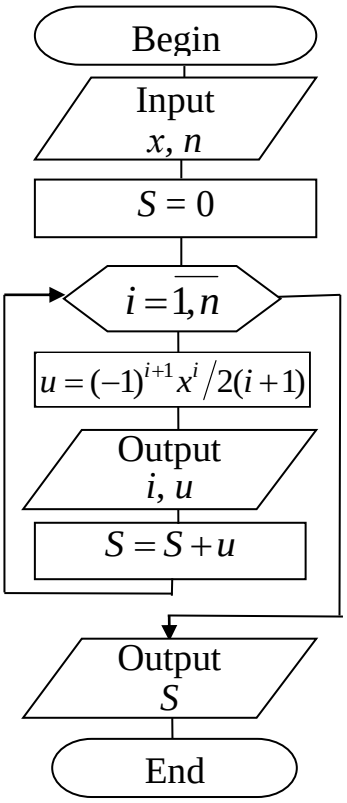
Examples of programs with FOR loop

Example 3.1. To calculate the sum of series.
Input integer number n and real number x, calculate

$$s = \frac{x}{4} - \frac{x^3}{6} + \frac{x^5}{8} - \ldots = \sum_{i=1}^{n} \frac{(-1)^{i+1} x^{2i-1}}{2(i+1)}.$$

The program code and the flowchart:

```
#include <iostream>
#include <math.h>
using namespace std;
int main()
{ system("color F0");
  double x, u, s=0;  int i, n;
  cout<<" Input integer number n= "; cin>> n;
  cout<<" Input value x= ";          cin>> x;
  cout<< "\n Resalts:\n";
```



```
  for (i=1; i<=n; i++)
  { u = pow(-1,i+1.0)*pow(x,i)/(2*(i+1));
    cout<<" Summand "<<i<<" = "<<u<<endl;
    s+=u;
  }
  cout<< " Sum = " << s << endl;
  system("pause>>void");
  return 0;
}
```

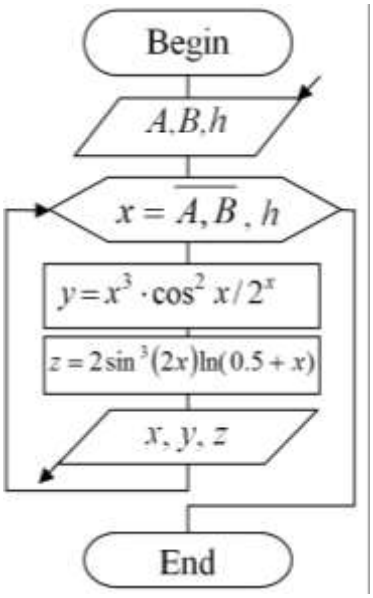
```
Input integer number n= 5
Input value x= 2.4

Resalts:
Summand 1 = 0.6
Summand 2 = -0.96
Summand 3 = 1.728
Summand 4 = -3.31776
Summand 5 = 6.63552
Sum = 4.68576
```

Example 3.2. Tabulation of function. Draw the flowchart and write the program with for loop for tabulation of functions $y(x)=\dfrac{x^3\cdot\cos^2 x}{2^x}$ and $z(x)=2\sin^3(2x)\ln(0.5+x)$, changing x in the interval [1, 11] with step h = 0,2.

Under the term tabulation of functions we understand the procedure of calculating discrete values of function in case the argument changes in accordance with the arithmetic sequence law. The results of tabulation are usually shown in tabular format accompanied with plot of the function.

The program code and the flowchart:



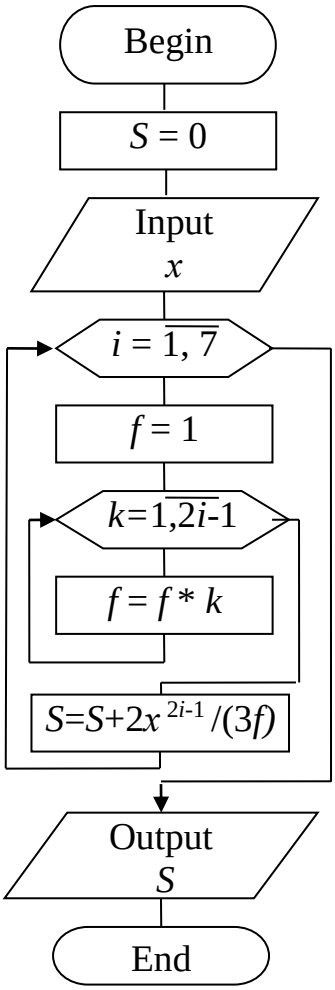

```
#include <iostream>
#include <math.h>
using namespace std;
int main()
{ double x, y, z, A, B, h;
  cout<<" A= "; cin>> A;
  cout<<" B= "; cin>> B;
  cout<<" h= "; cin>> h;
  cout<< "\n Results:\n";
  for(x=A; x<=B+0.1*h; x+=h)
  { y=x*x*x*pow(cos(x),2)/pow(2,x);
    z=2*pow(sin(2*x),3)*log(0.5+x);
    cout<< "x = " << x << " \t y(x) = " << y << "\t z(x) = " << z << endl;
  }
  system("pause>>void");
  return 0;
}
```

Results:

```
A= 1
B= 4
h= 0.3

Results:
x = 1    y(x) = 0.145963          z(x) = 0.609679
x = 1.3          y(x) = 0.0638462         z(x) = 0.161042
x = 1.6          y(x) = 0.00115203        z(x) = -0.000295161
x = 1.9          y(x) = 0.192082          z(x) = -0.401072
x = 2.2          y(x) = 0.802595          z(x) = -1.71181
x = 2.5          y(x) = 1.77282           z(x) = -1.93744
x = 2.8          y(x) = 2.79832           z(x) = -0.600682
x = 3.1          y(x) = 3.46849           z(x) = -0.00146958
x = 3.4          y(x) = 3.48022           z(x) = 0.328368
x = 3.7          y(x) = 2.80342           z(x) = 2.08336
x = 4    y(x) = 1.709     z(x) = 2.91314
```

Example 3.3. Calculate the sum of series $S = \sum_{i=1}^{7} \frac{2x^{2i-1}}{3(2i-1)!}$, where $i = 1, 2, \ldots, 7$.



Solution. *The program sums* the seven addends, each of them contains nested loop to calculate factorial $(2i-1)!$. Each addend is calculated in a separate variable `u`.

The program code and the flowchart:

```
#include <iostream>
#include <math.h>
using namespace std;
int main()
{
 double s=0, u, x;
 int i, k, f;
 cout<<"Input x = ";
 cin>>x;
 for (i=1; i<=7; i++)
 { f = 1;
   for (k=1; k<=2*i-1; k++) f *= k;
   u = 2*pow(x,2*i-1)/(3*f);
   s += u;
 }
 cout<<"\nSum = "<<s;
 cin.get();
 cin.get();
 return 0;
 }
```

Result:

```
Input x = 2.45
Sum = 3.83416
```

Example 3.4. Input m and calculate $S = \sum_{i=1}^{m} \frac{i}{i-2} \prod_{k=1}^{i+1} \frac{k+3}{k}$. Exclude zero addends and zero multipliers in the numerator or denominator.

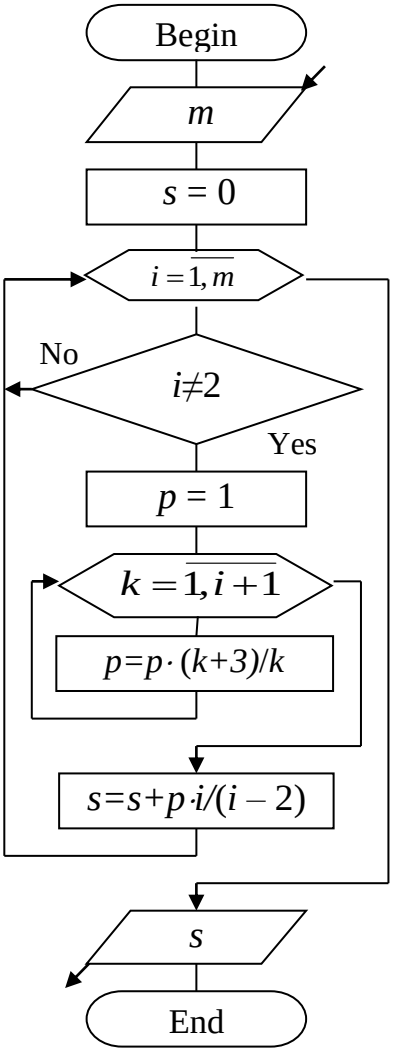
Solution. Internal loop parameter k depends on the external loop parameter `i` (`k` varies from `1` to `i+1`). The product $\prod_{k=1}^{i+1} \frac{k+3}{k}$ is calculated in the inner loop in variable `P`. It is a multiplier of the summand in the outer loop. Since the inner loop consists of one operator, the operator brackets `{ }` are optional.

It is necessary to initialize the variable `S` before the external loop (`S=0`). And it is necessary to initialize the variable `P` (`P=1`) inside the external loop before the internal one.

Do not forget to convert the nominator of type int to real type (to avoid losing digits after the decimal point): `(k+3.0)/k`.

The program code and the flowchart:

```
#include <iostream>
using namespace std;
int main()
{
 int i, k, m;
 cout<< "Input m = ";
 cin>>m;
  double S = 0, p;
  for (i = 1; i <= m; i++)
    if ( i != 2)
    { p = 1;
      for(k = 1; k <= i+1; k++)
        p*=(k+3.)/k;
      S += i/(i-2.) * p;
    }
 cout<< "\nSum = "<<S;
 cin.get();
 cin.get();
 return 0;
}
```



Result:

```
Input m = 10
Sum = 1874.05
```

Example 3.5. The program uses a nested for loop to find the prime numbers from 2 to 100.

The program code:

```
#include "stdafx.h"
#include <iostream>
using namespace std;

int main()
{ system("color F0");
   int i, j;
   for (i=2; i<100; i++)
   {
     for (j=2; j <= (i/j); j++)
       if (!(i%j)) break; // if factor found, not prime
       if (j > (i/j)) cout<< i << " is prime\n";
   }
  system ("pause>>void");
  return 0;
}
```

```
2 is prime
3 is prime
5 is prime
7 is prime
11 is prime
13 is prime
17 is prime
19 is prime
23 is prime
29 is prime
31 is prime
37 is prime
41 is prime
43 is prime
47 is prime
53 is prime
59 is prime
61 is prime
67 is prime
71 is prime
73 is prime
79 is prime
83 is prime
89 is prime
97 is prime
```

Control questions

1) What is the iteration process?
2) What iteration statements in C++ do you know?
3) How many times does the following loop repeat? What is the value of s:
```
for (int k = -1, s=0; k <= 5; k++) s++;
```
4) Are there any errors in the following code?

а)
```
int k, m=2, n=3;
for (k=1; k<=n; k++) n=n+m;
```

б)
```
int n=-7, m=2;
for (int k=n; k<=m; k--) k++;
```

5) What is the value of m after the following code:

а)
```
int k, m=1;
for (k=1; k<=5; k++) m++;
```

б)
```
int m=1, n=5;
for (int k=n; k>=1; k--) m*=k;
```

6) What are the rules for programs with nested loops?
7) Which code contains nested loops?

a)
```
for(k=1;k<=10;k++)
  p=k;
 for(j=1;j<=5;j++)
  s+= p*j;
```

b)
```
for(k=1;k<=10;k++)
{ p=k;
  for(j=1;j<=5;j++)
   s+= p*j;  }
```

c)
```
for(k=1;k<=10;k++)
{ p=k; }
for(j=1;j<=5;j++)
{ s+= p*j;  }
```

8) What is the value of variable s after the code:

a)
```
for(s=0,k=1; k<=3; k++)
  for(j=1; j<=k; j++) s+=j;
```

b)
```
for(s=0,k=1; k<=2; k++)
  for(m=k; m<=2; m++) s+=m;
```

Individual task

1) Answer the control questions.

2) Draw the flowchart and write the program with for loop for problems given in the table 3.1.

3) Draw the flowchart and write the program for function tabulation (table 3.2).

4) Create the project and draw the flowchart for problems with nested loops given in tables 3.3 – 3.4.

Table 3.1 (basic level)

№ var.	Problems
1	Input integer n and real x, calculate: $s = 1 + x + \frac{x^2}{2} + \frac{x^3}{3} + \ldots = \sum_{i=0}^{n} \frac{x^i}{i}$
2	Input integer n, calculate: $s = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \ldots = \sum_{i=1}^{n} \frac{(-1)^{i+1}}{2i-1}$
3	Input integer n and real x, calculate: $s = \cos(x) + \frac{\cos(2x)}{2} + \frac{\cos(3x)}{3} + \ldots = \sum_{i=1}^{n} \frac{\cos(ix)}{i}$
4	Input integer n and real a, calculate: $s = 1 - a + a^2 - a^3 + \ldots = \sum_{i=0}^{n} (-a)^i$

End of table 3.1

№ var.	Problems
5	Input integer n, calculate: $s = 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \ldots = \sum_{i=1}^{n} \frac{1}{i}$
6	Input integer n and real x, calculate: $s = -\frac{1}{x} + \frac{3}{x^2} - \frac{5}{x^3} - \ldots = \sum_{i=1}^{n} \frac{(-1)^i (2i-1)}{x^i}$
7	Input integer n and real x, calculate: $s = -x + \frac{x^3}{2} - \frac{x^5}{3} + \ldots = \sum_{i=1}^{n} \frac{(-1)^i x^{2i-1}}{i}$
8	Input integer n and real x, calculate: $s = \cos(x) + \cos(3x^3) + \ldots = \sum_{i=1}^{n} \cos((2i-1)x^{2i-1})$
9	Input two natural numbers n and m ($n < m$), calculate $s = \sum_{i=n}^{m} i$
10	Input integer n and real x, calculate $s = x - \frac{x^3}{3} + \frac{x^5}{5} - \ldots = \sum_{i=0}^{n} \frac{(-1)^{i-1} x^{2i+1}}{2i+1}$
11	Input integer n and real x, calculate $s = \sin(x) + \frac{\sin^2(x)}{4} + \frac{\sin^3(x)}{7} + \ldots = \sum_{i=1}^{n} \frac{\sin^i(x)}{3i-2}$.
12	Input integer n and real x, calculate $s = (x+1) + \frac{(x+2)^3}{4} + \frac{(x+3)^5}{9} - \ldots = \sum_{i=1}^{n} \frac{(x+i)^{2i-1}}{i^2}$.

Table 3.2 (basic level)

№ var.	Function $y = f(x)$	Function $z = f(x)$	Interval and step
1	$\sin(x)/x^2$	$\cos(x)/x$	$x \in [0{,}5;\ 11]$, $h = 0{,}3$
2	$\operatorname{arctg}(x + 3{,}1)$	e^x	$x \in [-6;\ 1]$, $h = 0{,}2$
3	$e^{3(x-0.6)}$	$\arcsin(x)$	$x \in [-1;\ 1]$, $h = 0{,}05$
4	$\sqrt{\lvert\sin(x + \pi/4)\rvert}$	$\sin x^2 + \cos x$	$x \in [-4;\ 10]$, $h = 0{,}4$
5	$\operatorname{tg}\sqrt{x}$	$x/(x-3)^2$	$x \in [4{,}5;\ 18{,}5]$, $h = 0{,}4$
6	$1/e^x$	$\lg(x/2 + 0{,}1)$	$x \in [0;\ 7]$, $h = 0{,}2$
7	$\operatorname{tg}(x/3) \cdot \sin(x - 1{,}2)$	$2{,}5\sin(x/2)$	$x \in [-2;\ 5]$, $h = 0{,}2$
8	$1/x$	$(x/3)^2$	$x \in [0{,}5;\ 4]$, $h = 0{,}1$
9	$\cos(1{,}5x) \cdot \lg(2{,}5x)$	$e^{\frac{1}{\sqrt{x}}} \sin(x)$	$x \in [3{,}5;\ 10{,}5]$, $h = 0{,}2$
10	$\cos(x)/x$	$\cos(x/2)$	$x \in [0{,}3;\ 7{,}3]$, $h = 0{,}2$
11	e^x	$1{,}5\cos(x - \pi/4 \cdot e^x)$	$x \in [-6;\ 11]$, $h = 0{,}2$
12	$\arcsin(x)$	$\cos(1/(x + \pi/3))$	$x \in [-1;\ 1]$, $h = 0{,}05$

Table 3.3 (basic level)

Input the value of *x* and calculate the value of function *y*.

№	Function	№	Function	№	Function	№	Function
1	$y=\prod_{k=1}^{10}\frac{x}{(k+1)!}$	2	$y=\sum_{i=1}^{6}\frac{(-1)^i x^{2i}}{(3i-1)!}$	3	$y=\sum_{i=1}^{10}\frac{(-1)^{i+1} i!}{2^{2i-1}\sin x}$	4	$y=\sum_{i=1}^{5}\frac{(2i-1)!}{x^{2i-1}}$
5	$y=\sum_{k=1}^{7}\frac{k!\cos(\pi k-x)}{\ln x}$	6	$y=\prod_{k=1}^{10}\frac{x^k}{(2k)!}$	7	$y=\sum_{k=1}^{7}\frac{(2k-1)!}{2^k x^{k-1}}$	8	$y=\sum_{i=1}^{5}\frac{(-1)^i x^{2i}}{(i+1)!\cos x}$
9	$y=\sum_{k=1}^{5}\frac{(-1)^{k-1}\cdot x^{k+2}}{k!}$	10	$y=\sum_{i=1}^{11}\frac{(-1)^i x^{3i}}{(2i-1)!}$	11	$y=\sum_{k=1}^{8}\frac{(-1)^k x^{2k-1}}{2^k k!}$	12	$y=\sum_{k=1}^{7}\frac{(-1)^k\cdot k!}{\sqrt{x}+\sin x}$

Table 3.4

Input the value of *x* and calculate the value of function *y*.

№	Function	№	Function	№	Function
1	$S=\sum_{k=1}^{n}\frac{(k+1)}{(k-5)}\prod_{m=1}^{k+1}\frac{m-2}{m^2-9}$	2	$Z=\prod_{j=-4}^{k}\frac{(j+2)}{j-3}\sum_{i=j}^{k+5}\left(\frac{\sqrt[5]{i+5}}{i-11}\right)$	3	$S=\sum_{k=0}^{n}\frac{(-2)^{k+1}}{(k-5)}\prod_{i=1}^{k+1}\frac{i}{i^2-16}$
4	$Z=\prod_{j=-2}^{k}\frac{j}{j-1}\sum_{i=j}^{k}\frac{i}{i+5}$	5	$W=\sum_{i=1}^{k}\frac{(-1)^i(i+1)!}{i^2-4}$	6	$Y=\sum_{n=1}^{k}\frac{(-1)^{2-n}(n^2-9)^2}{(n+1)!}$
7	$W=\sum_{i=1}^{k}\frac{(-1)^i}{(i-4)^2}\prod_{n=1}^{i+2}\frac{n^2-4}{n+2}$	8	$L=\prod_{j=1}^{k}\frac{(j-5)}{j-2}\sum_{i=k}^{12}\frac{\sqrt{i+5}}{i-1}$	9	$Q=\sum_{k=1}^{n}\frac{(k-1)^{k+1}(k-3)}{(k+1)!}$
10	$Z=\prod_{t=1}^{k}\frac{k-t-1}{\cos(t)-3}\sum_{i=1}^{t}\left(\frac{3i-2}{i-7}\right)$	11	$P=\prod_{j=1}^{k}\frac{(j-6)j}{j-3}\sum_{i=j}^{12}\frac{\sqrt[3]{i+5}}{i-11}$	12	$A=\prod_{j=1}^{k}\frac{j-3}{(j-4)j}\sum_{i=0}^{j}\frac{\sqrt{i+4}}{i-1}$

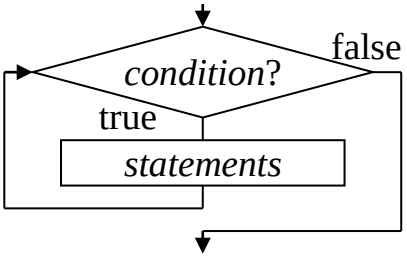
Lab # 4

While and do-while iteration statements. Processing with sequences of numbers

Purpose: to get practical skills in organization of cyclic calculations in C++ with the use of while and do-while iteration statements, and to get practical skills in processing of sequences of numbers.

Theoretical information

While loop is used when we do not know, how many times we need to repeat statements, but we do know the condition for repeating.

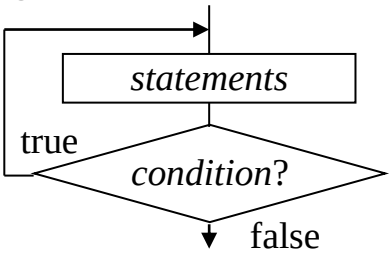


Its format is:

```
while ( condition )
{ statements;
}
```

Its function is simply to repeat *statements* while *condition* is true.

The compiler first examines the *condition*. If the *condition* is true, then it executes the *statements*. After executing the *statements*, the *condition* is checked again. As long as the *condition* is true, it will keep executing the *statements*. When or once the *condition* becomes false, the loop stops.



The format of **do-while** loop (loop with postcondition):

```
do {
    statements;
  } while  ( condition );
```

The do-while loop is used when it is conveniently to check up the condition after *statements*. The do-while loop executes *statements* first. After the first execution of the *statements*, it examines the *condition*. If the *condition* is true, then it executes the *statements* again. It will keep executing the *statements* as long as the *condition* is true. Once the *condition* becomes false, the looping (the execution of the *statements*) would stop.

Like the `if` and the `while` statements, the *condition* being checked must be included between parentheses. The whole `do-while` statement must end with a semicolon. The `do-while` statement can be used to insist on getting a specific value from the user. The differences of the various loop statements are shown in the following examples of calculation of the sum of all odd numbers in the range of 10 to 100:

1) using `for` loop

```
int i, s=0;
for (i=11; i<100; i += 2) s += i;
```

2) using `while` loop

```
int s=0, i=11;
while (i<100) { s += i; i += 2; }
```

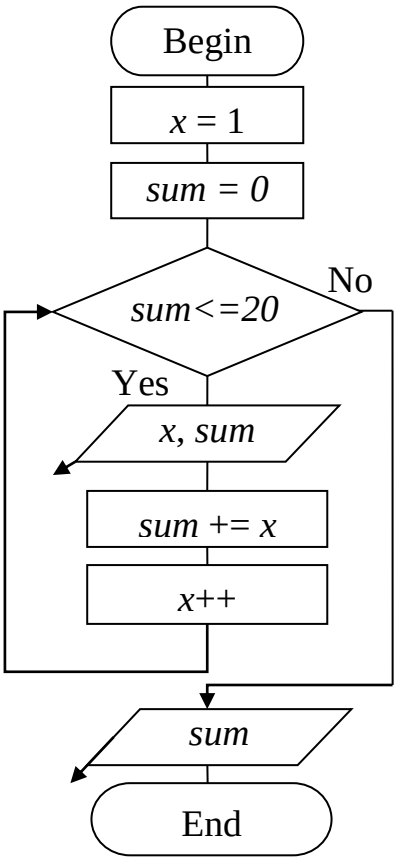
3) using `do-while` loop

```
int s=0, i=11;
do { s += i; i += 2;} while (i<100);
```

Examples of programs

Example 4.1. Make a program to print out positive integer numbers from 1 until their sum will become greater than 20.

We will add a number *x* to sum while the sum will be less than or equal 20.



Solution 1. Using `while` loop.

The program code and the flowchart:

```
#include <iostream>
using namespace std;
int main()
{
  system("color F0");
  int x=1;      // the first number is 1
  int sum=0;
  cout<< "x\tsum\n";
  while (sum<=20)
  {
    sum += x;  // sum=sum+x;
    cout<< x <<"\t"<<sum<<"\n";
    x++;
  }
  cout <<"\nEndSum = "<< sum ;
  system ("pause>>void");
  return 0;
}
```

The algorithm:

1) at first `x=1` and `sum=0`. As `sum<20`, then the condition is true and while loop starts to work;
2) `sum` becomes 1;
3) number 1 and the first sum are added to output;
4) `x` becomes 2;
5) loop returns to condition. It is true. And we do steps 2 … 4 again and again.

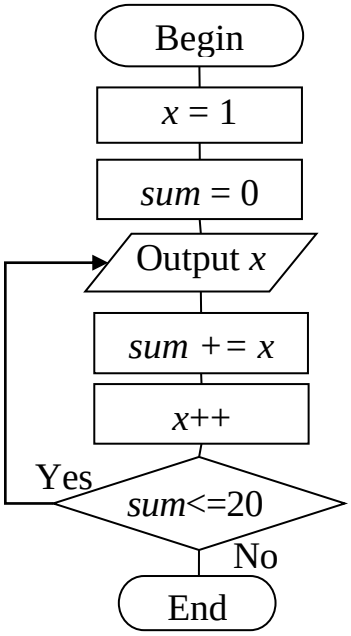
```
x       sum
1       1
2       3
3       6
4       10
5       15
6       21

EndSum = 21
```

From results window we see that when `x=6` the sum becomes 21. When we come to condition it is false and while stops.

Solution 2. Using `do-while` loop.

The program code and the flowchart:




```
int main()
{ system("color F0");
  int x=1;      // the first number is 1
  int sum=0;
  cout << "x\tsum\n";
  do
  {
    sum+=x;     // sum=sum+x;
    cout << x <<"\t"<<sum<<"\n";
    x++;
  }while (sum<=20);
  cout <<"\nEndSum = "<< sum ;
  system ("pause>>void");
  return 0;
}
```

```
x       sum
1       1
2       3
3       6
4       10
5       15
6       21

EndSum = 21
```

1) At first `x=1` and `sum=0`.
2) `sum` becomes 1.
3) Number 1 and first sum are added to output (without any conditions).
4) `x` becomes 2
5) Check up the condition. It is true. And we repeat until the `sum` becomes 21. Then the loop stops.

Example 4.2. Calculate the sum of series $S=\sum_{k=1}^{\infty}\frac{x^{2k+1}}{3^k(2k-1)!}$. Sum the addends, the absolute value of which is bigger than the required accuracy $\varepsilon=10^{-4}$. Define the number of addends.

Input value x $(-2 < x < 2)$ with keyboard.

Solution. It is inappropriate to use the `for` loop in this program, as the number of repetitions of the loop is unknown. The better way is to use the `do-while` loop, because it is necessary to know the value of the first addend at the time of the first verification of condition.

The program code and the flowchart:

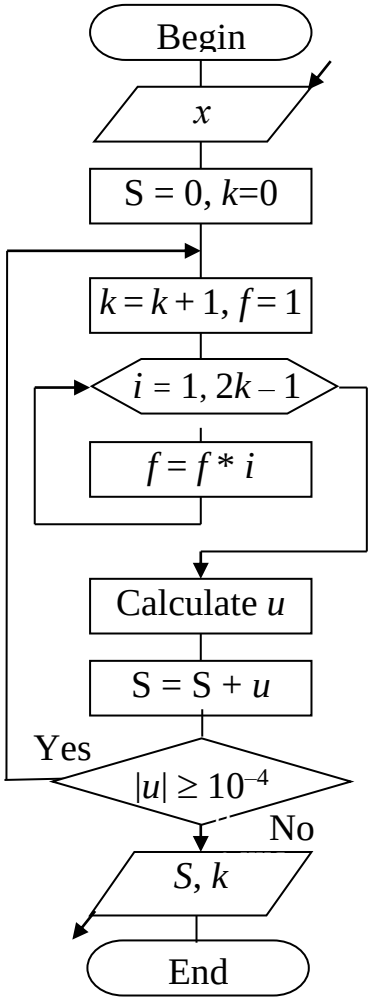
```
#include <iostream>
#include <math.h>
using namespace std;
int main()
{
 double x, f, u, s=0;
 int i, k=0;
 cout<< "Input x= ";   cin>> x;
```

```
 cout<< "\nResults:\n";
 do            // do-while loop
 {
   k++;        // Increasing variable k on 1
  // Calculation of factorial
  for(i=1,f=1; i<=2*k-1; i++) f *= i;
  // Calculation of k-summand
  u=pow(x, 2*k+1)/(pow(3.,k)*f);
  s+=u;        // Add the new addend to the sum
 }
 while (fabs(u)>=1e-4);
 cout << "sum = " << s << endl;
 cout << "count of summands = " << k << endl;
 system("pause>>void");
 return 0;
}
```

Results:

```
Input x= 2.5

Results:
sum = 7.21478
count of summands = 6
```



Example 4.3. Calculate the sum of series $y = \sum_{k=1}^{\infty} \frac{(-1)^{k+1} x^{2k-1}}{(k+3)(2k)!}$. Sum the numbers, the absolute value of which is bigger than the required accuracy $\varepsilon = 10^{-4}$. Determine the number of summands. Input value x $(-2 < x < 2)$ with keyboard.

Solution. The multiplier $(-1)^{k+1}$ is 1 for odd values k = 1, 3, 5, …, and it is (-1) for paired values k = 2, 4, … So this is alternating series, where all even summands are negative and all odd summands are positive. All summands are outputted separately for illustration and control of the correctness of the solution.

The algorithm of the program can be improved; it should calculate the recurrent multiplier (recurrent factor) first. This will allow the program to get rid of the inner loop to calculate the factorial and of checking value of k.

```
    u = x/8;  s = u;
     do
     {
       k++;
       cout<< "summand " << k <<   ":  " << u << endl;
       r = -x*x*(k+2)/((3+k)*(2*k) *(2*k-1));
       u *= r ;
       s += u;
     }
     while(fabs(u) >= eps);
```

More details on creating of recurrent formula. The sum of series $y=\sum_{k=1}^{\infty}\frac{(-1)^{k+1}x^{2k-1}}{(k+3)(2k)!}$ can be represented as $y=\sum_{k=1}^{\infty}u_k$, where $u_k=\frac{(-1)^{k+1}x^{2k-1}}{(k+3)(2k)!}$.

Each next summand (u_k) can be obtained by multiplying the previous summand (u_{k-1}) by some multiplier, which is called recurrent multiplier or recurrent factor:

$u_2=R\cdot u_1,\ u_3=R\cdot u_2,\ \ldots,\ u_k=R\cdot u_{k-1},$

That is why recursive multiplier *R* is a ratio of two nearby terms:

$$\boxed{R=\frac{u_k}{u_{k-1}}}.$$

It is necessary to substitute $u_k=\frac{(-1)^{k+1}x^{2k-1}}{(k+3)(2k)!}$ and u_{k-1} into this formula to calculate the value of *R*. To define u_{k-1} put $(k-1)$ instead of *k* into u_k:

$$u_{k-1}=\frac{(-1)^{k}x^{2(k-1)-1}}{(k+2)(2k-2)!}=\frac{(-1)^{k}x^{2k-3}}{(k+2)(2k-2)!};$$

$$R=\frac{u_k}{u_{k-1}}=u_k=\frac{(-1)^{k+1}x^{2k-1}}{(k+3)(2k)!}\cdot\frac{(k+2)(2k-2)!}{(-1)^{k}x^{2k-3}}=\frac{(-1)^{k+1}\cdot x^{2k-1}\cdot(k+2)\cdot(2k-2)!}{(-1)^{k}\cdot x^{2k-3}\cdot(k+3)\cdot(2k)!}=$$

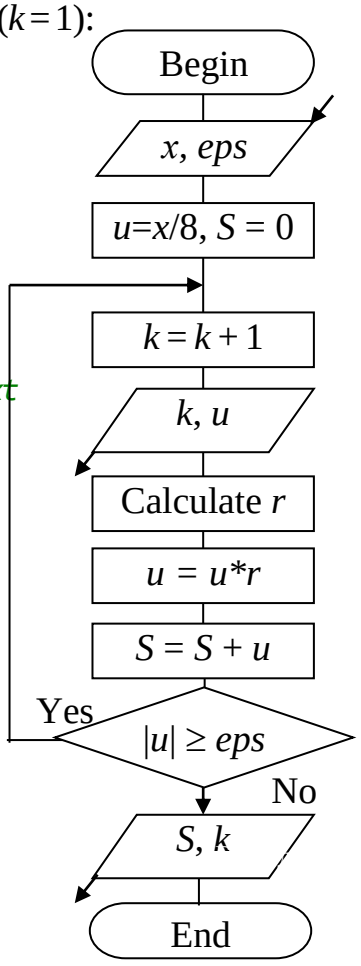
$$=\frac{(-1)^{k+1-k}\cdot x^{2k-1-(2k-3)}\cdot(k+2)}{(k+3)}\cdot\frac{1\cdot2\cdot\ldots\cdot\cancel{(2k-2)}}{1\cdot2\cdot\ldots\cdot\cancel{(2k-2)}\cdot(2k-1)\cdot2k}=-\frac{x^2(k+2)}{2k(k+3)(2k-1)}.$$

It is also necessary to calculate the first term of the series ($k=1$):

$$u_1=\frac{(-1)^2x}{4\cdot(2)!}=\frac{x}{8}.$$

The program code and the flowchart:

```
#include <iostream>
#include <math.h>
using namespace std;
int main()
{ system("color F0");//Set white background and black text
  double x, u, r, s=0, eps;
  int k=0;
  cout<<"Input value x=";   cin>> x;
  cout<<"Input value required accuracy="; cin>>eps;
  cout<<"\nResults:\n";
  u = x/8;  s = u;
  do
   { k++;
     cout<<"summand "<< k <<":  "<< u << endl;
     r = -x*x*(k+2)/((3+k)*(2*k) *(2*k-1));
     u *= r;
     s += u;
   } while(fabs(u) >= eps);
```



```
 cout<< "sum = " << s << endl;
 cout<< "number of summands = " << k << endl;
 system("pause>>void");
 return 0;
}
```

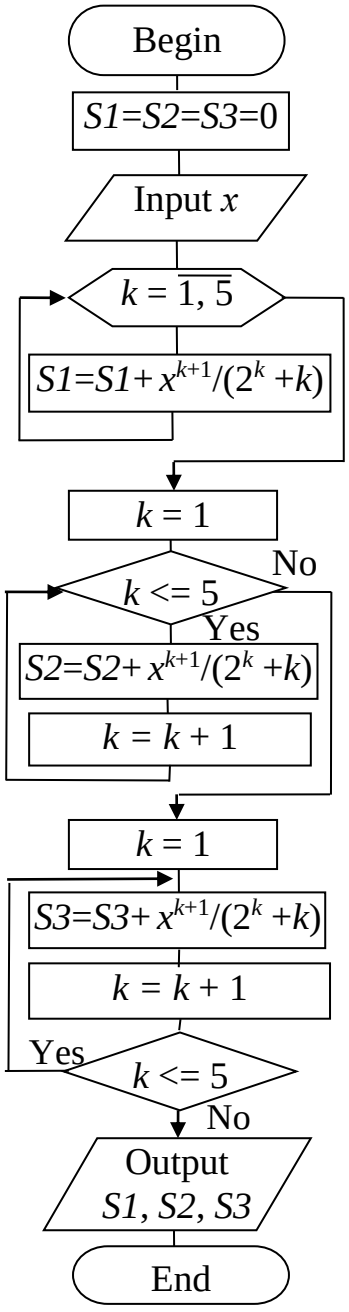
Results:

```
Input value x= 2.4
Input value required accuracy = 0.0001

Results:
summand 1:  0.3
summand 2:  -0.648
summand 3:  0.248832
summand 4:  -0.0398131
summand 5:  0.00351005
summand 6:  -0.000196563
sum = -0.13566
number of summands = 6
```

Example 4.4. Calculate the sum of series $f(x)=\sum_{k=1}^{5}\frac{x^{k+1}}{2^k+k}$ in three ways, use different loop statements.

The flowchart and the program code:




```
#include "stdafx.h"
#include <iostream>
#include <math.h>
using namespace std;
int main()
{ double S1=0, S2=0, S3=0, x;   int k;
  cout<<"Input x: "; cin>>x;

  // for loop
  for (k=1; k<=5; k++)
       S1 += pow(x,k+1)/(pow(2.0,k)+k);

  // while loop
  k = 1;
  while (k<=5)
  {    S2 += pow(x,k+1)/(pow(2.0,k)+k);
       k++;
  }

  // do..while loop
  k = 1;
  do {
       S3 += pow(x,k+1)/(pow(2.0,k)+k);
       k++;
     } while (k<=5);
```

```
  cout<< "Results:\n";
  cout<< "for loop: S1=" << S1 << endl;
  cout<< "while loop: S2=" << S2 << endl;
  cout<< "do..while loop: S3=" << S1 << endl;
  system("pause");
  return 0;
}
```

Results of the program:

```
Input x: 1.5
Results:
for loop: S1=2.46027
while loop: S2=2.46027
do..while loop: S3=2.46027
```

Example 4.5. To define and output all divisors of inputted natural number. The program code:

```
#include <iostream>
int main()
{ system("cls");          // Clear screen
  system("color F0");     // Set white background and black text
  unsigned int x, i;
  std::cout << "Input x: ";  std::cin  >> x;
  std::cout << std::endl << "Divisors: ";
  for (i=1; i<=x; i++)
   if (x%i == 0)              // If x is divided without the rest,
     std::cout << i << "\t"; // output another divider.
  std::cin.get();
  std::cin.get();
  return 0;
}
```

```
Input x: 121

Divisors: 1     11      121
```

```
Input x: 42

Divisors: 1     2       3       6       7       14      21      42
```

Example 4.6. To define the number of digits of entered natural number.

```
#include <iostream>
using namespace std;
int main()
{ unsigned int N, kol;
  cout << "N=";   cin >> N;
  for (kol=1; N/10>0; kol++,N/=10);
  cout << "kol=" << kol << endl;
  system("pause>>void");
  return 0;
}
```

```
N=50026
kol=5
```

Example 4.7. To define the smallest and the biggest digits of entered natural number.

```
#include <iostream>
using namespace std;
int main()
{ unsigned int N, z, min, max;
  cout<<"N="; cin>>N;
  for (max=min=N%10; N>0; N/=10)
  {  z=N%10;
     if (z>max) max=z;
     if (z<min) min=z;
  }
  cout << "min=" << min << endl << "max=" << max << endl;
  system("pause>>void");
  return 0;
}
```

```
N=550308
min=0
max=8
```

Usually a problem requires handling a sequence of numbers. For example, you can sum 100 numbers or calculate their arithmetic average. The best way to solve these problems is to use loops.

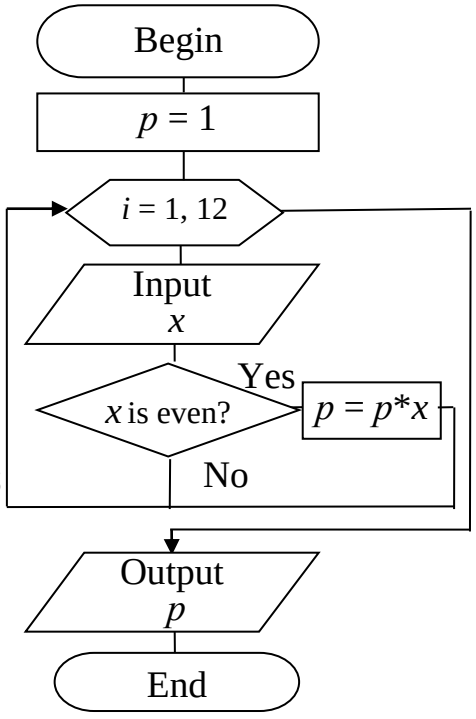
Example 4.8. Input 6 integer numbers and calculate the product of even numbers.

Solution. In the cycle, which is repeated 12 times, do the following:

– to prompt to enter next number;
– to input a new number;
– to check entered number whether it is even and, if so, the product is multiplied by the number.

The program code and the flowchart:

```
int main()
{
 // Initially the product is 1.
 int i, x, p=1;
 // To repeat the actions 12 times:
 for (i=1;i<=12;i++)
{
   // To output the prompt,
   cout<< "Input " << i << " number: ";
   cin >> x;     // to input a number, and,
   // if the number is even, multiply it.
   if (x%2==0 && x!=0) p *= x;
 }
 cout<< "The product of even numbers: "<< p << endl;
 system("pause>>void");
 return 0;
}
```



Results:

```
Input 1 number: 3
Input 2 number: 1
Input 3 number: -7
Input 4 number: 6
Input 5 number: 3
Input 6 number: -2
Input 7 number: 4
Input 8 number: 7
Input 9 number: 11
Input 10 number: -4
Input 11 number: -5
Input 12 number: 3
The product of even numbers: 192
```

This program has one significant drawback. If there are no even numbers, then the result of the product is 1. Instead, it is desirable to display a message that even numbers were not inputted. The counting of the number of entered even numbers should be organized for this. If after the loop this quantity will be 0, the message should be displayed.

The program code:

```
#include <iostream>
using namespace std;
int main()
{ int i, x, p=1, k=0;
  for (i=1; i<=12; i++)
  { cout<< "Input " << i << " number: ";   cin>> x;
    if (x%2==0 && x!=0)       // If the number is even
    { p *= x;                 // calculate the product
      k++;            // and increase the quantity even numbers.
  } }
if (k > 0) // If the quantity of even numbers is greater than 0, the product is
    cout << "The product of even numbers: " << p << endl; // displayed,
 else        // otherwise - message displays that there are no even numbers.
    cout << "There are no even numbers." << endl;
 system("pause>>void");
 return 0;
}
```

Results:

```
Input 1 number: 3
Input 2 number: 1
Input 3 number: -7
Input 4 number: 5
Input 5 number: 3
Input 6 number: -7
Input 7 number: 3
Input 8 number: 7
Input 9 number: 11
Input 10 number: -17
Input 11 number: -5
Input 12 number: 3
There are no even numbers.
```

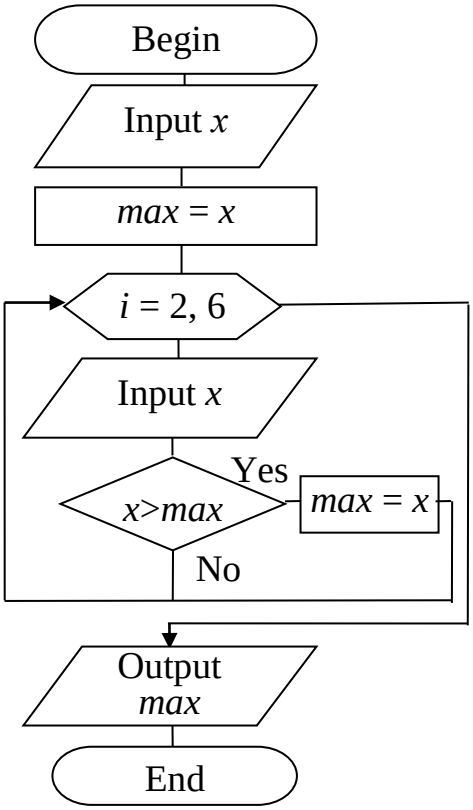
Example 4.9. Input 6 real numbers and find the biggest of them.

Solution. The algorithm of searching the maximum number in the sequence:
1) to input the first number x;
2) to assume that it is the biggest: `max = x;`
3) to input successively the rest of the numbers in the loop and compare each of the numbers with the value of `max` and if the number `x` is bigger than `max`, assign it to `max`: `if (x>max) max=x;`

The program code and the flowchart:

```
#include <iostream>
using namespace std;

int main()
{
 int i;  double x, max;
 cout<< "Input 1st number: ";
 cin>> x;
 max=x;
 for (i=2; i<=6; i++)
 { cout<< "Input the"<< i <<" number ";
   cin>> x;
   if (x>max) max=x;
 }
 cout<< "The biggest number is " << max << endl;
 system("pause>>void");
 return 0;
}
```



Results:

```
Input the 1 number: 5.5
Input the 2 number: 0.2
Input the 3 number: -7
Input the 4 number: 3.1
Input the 5 number: 15.2
Input the 6 number: -1.4
The biggest number is 15.2
```

Example 4.10. Input a sequence of 5 integer numbers and define the first negative element.

The program code:

```
#include <iostream>
using namespace std;
int main()
{ int v=0, x;
 for (int i=0; i<5; i++)
 { cout<< "Input " << i << " number: ";    cin>> x;
   if (x<0 && v==0) v=x;
 }
```

```
 if (v<0)
   cout<< "The first negative number = " << v << endl;
 else
   cout<< "There is no negative number = " << endl;
 system("pause>>void");
 return 0;
}
```

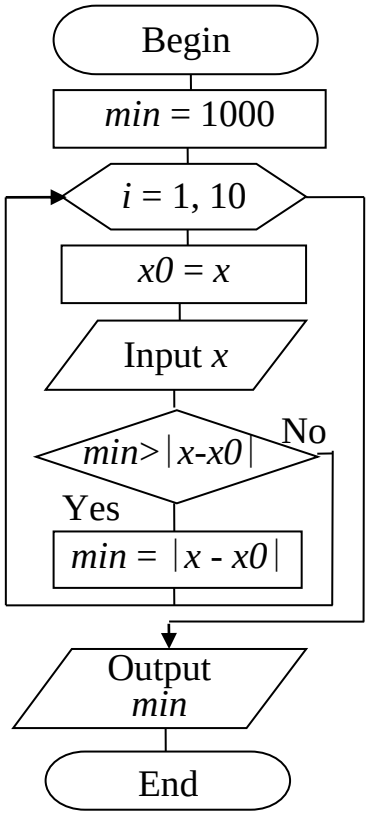
Results:

```
Input 0 number: 3
Input 1 number: 7
Input 2 number: -2
Input 3 number: 0
Input 4 number: -4
The first negative number = -2
```

Example 4.11. Input a sequence of 10 real numbers (a_1, a_2, a_3, ...) and calculate $\min(|a_2 - a_1|, |a_3 - a_2|, \ldots)$.

The program code and the flowchart:

```
#include <iostream>
using namespace std;
int main()
{
 double min=1e3, x0, x;
 for (int i=0; i<10; i++)
 {
   x0=x;
   cout<< "Input " << i << " number: ";
   cin>> x;
   if (min>fabs(x-x0)) min=fabs(x-x0);
 }
 cout<< "Min = " << min << endl;
 system("pause");
 return 0;
}
```



Results:

```
Input 0 number: 7
Input 1 number: 9
Input 2 number: 3
Input 3 number: -5
Input 4 number: 2
Input 5 number: -7
Input 6 number: 2
Input 7 number: 5
Input 8 number: 1
Input 9 number: 4
Min = 2
```

Control questions

1) What iteration statements in C++ do you know?
2) What is the value of n after execution of the following code:
   a)
```
int k=0, n=17;
while(k<7){ k++;  n--; }
```
   b)
```
int m=1, n=1;
while(m<5){ m+=2; n=n*m; }
```
3) What is the value of y after execution of the following code :
   a)
```
int i=1, y=1;
do {y*=i++;} while(y<7);
```
   b)
```
int k=1;  float y=0;
do{k+=2;y+=1./k;}while(k<5);
```
4) Write the calculation of the sum with each of iteration statements: $S=\sum_{i=1}^{6} i^2$.
5) What is the value of s after the code:
```
s=0.5; i=0;
while (i<5) i++;
s+=1.0/i;
```

Individual task

1) Create the project (with flowchart) to calculate the infinite sum from the table 4.1. Accuracy $\varepsilon = 10^{-4}$. Input x from the screen ($-2 < x < 2$).
2) Create the project (with flowchart) to calculate the sum (table 4.2) with each of iteration statements you know.
3) Draw the flowcharts and write the programs with loop for problems given in the tables 4.3 and 4.4.

Table 4.1 (basic lewel)

№	Function $f(x)$	№	Function $f(x)$	№	Function $f(x)$	№	Function $f(x)$
1	$\sum_{k=1}^{\infty}\frac{(-1)^k x^{2k+1}}{(2k+1)!}$	2	$\sum_{k=1}^{\infty}\frac{(-1)^{k+1} x^{2k}}{k!2^{k-1}}$	3	$\sum_{k=1}^{\infty}\frac{(-1)^{k-1} x^{k}}{k!}$	4	$\sum_{k=1}^{\infty}\frac{(-1)^k x^{2k+1}}{k(2k+1)!}$
5	$\sum_{k=1}^{\infty}\frac{(-1)^k x^{2k-1}}{k(k+3)!}$	6	$\sum_{k=1}^{\infty}\frac{(-1)^k x^{2k+1}}{2k(k+1)!}$	7	$\sum_{k=1}^{\infty}\frac{(-1)^{k+1} x^{2k}}{(2k-1)!}$	8	$\sum_{k=1}^{\infty}\frac{(-1)^{k-1} x^{3k-1}}{(2k)!}$
9	$\sum_{k=1}^{\infty}\frac{(-1)^{k+1} x^{k+3}}{k^2(k+2)!}$	10	$\sum_{k=1}^{\infty}\frac{(-1)^{k+1} x^{2k}}{(2k+1)!}$	11	$\sum_{k=1}^{\infty}\frac{(-1)^{k-1} x^{3k-1}}{(k+2)k!}$	12	$\sum_{k=1}^{\infty}\frac{(-1)^k x^{3k-1}}{(k+3)(3k)!}$

Table 4.2 (basic level)

№	Function $f(x)$	№	Function $f(x)$	№	Function $f(x)$	№	Function $f(x)$
1	$\sum_{k=1}^{7}\frac{2^k \sin(x+k)}{(x+1)^k}$	2	$\sum_{k=1}^{9}\frac{x^{k+1}}{(k+1)^x}$	3	$\sum_{k=1}^{12}\frac{\sin(kx)+k}{\sqrt[k]{x+0.1}+6k}$	4	$\sum_{k=1}^{9}\frac{\ln(x+1)}{(x+k)^k}$
5	$\sum_{k=1}^{9}\frac{\sin(2kx)+0.2}{2k+5}$	6	$\sum_{k=1}^{7}\frac{kx\cos(x+k)}{\ln(2+x)+2k}$	7	$\sum_{k=2}^{6}\frac{\sin(0.17x^k)}{2k+x}$	8	$\sum_{k=1}^{8}\frac{5\ln(2kx)}{\operatorname{arctg}(2x)+k^2}$
9	$\sum_{k=2}^{9}\frac{\operatorname{tg}(x)-x^2/k}{k^2-1}$	10	$\sum_{k=1}^{7}\frac{\sin(x^k-\pi)}{\ln k^2+0.3}$	11	$\sum_{k=1}^{12}\frac{\cos(kx)}{k}$	12	$\sum_{k=1}^{8}\frac{\sin x^k}{4k}$

Table 4.3

№	Problems
1	Input 8 real numbers and calculate an average of positive numbers
2	Input 7 real numbers and sum the squares of the numbers, which absolute value is not bigger than 3
3	Input 14 integer numbers and calculate the quantity of non-zero numbers
4	Input 9 real numbers and find the smallest of them
5	Input 6 integer numbers and calculate the product of non-zero numbers
6	Input 10 integer numbers and calculate an average of the numbers from the interval [10, 20]
7	Input 8 real numbers and calculate the quantity of the numbers from the interval [5, 10]
8	Input 7 integer numbers and sum the absolute values of the even numbers
9	Input 9 real numbers and calculate the product of positive numbers, which value is not bigger than 4
10	Input 12 real numbers and calculate the quantity of positive numbers and the quantity of negative numbers
11	Input 8 integer numbers and calculate an average of the absolute values of the numbers
12	Input 12 integer numbers and calculate the quantity of two-digit numbers

Table 4.4

№	Problems
1	Input a sequence of real numbers and calculate the quantity of numbers, which are bigger than the previous number
2	Input a sequence of real numbers and sum the numbers, which values are smaller than the first number
3	Input a sequence of integer numbers and calculate the difference between the smallest and the biggest numbers
4	Input a sequence of integer numbers and calculate the difference between the biggest number and the first number
5	Input a sequence of real numbers and calculate an average of the numbers, which values are bigger than the first number
6	Input a sequence of real numbers and calculate the difference between the biggest number and the last number
7	Input a sequence of integer numbers and calculate the quantity and the sum of the numbers, which are multiple to 5 and are not multiple to 7
8	Input a sequence of integer numbers and calculate the double sum of the positive numbers
9	Input a sequence of real numbers (a_1, a_2, a_3, ...) and calculate $\min(a_1, a_3, a_5, ...) + \max(a_2, a_4, a_6, ...)$
10	Input a sequence of real numbers (a_1, a_2, a_3, ...) and calculate $a_1*a_2 + a_2*a_3 + ... + a_{n-1}*a_n$
11	Input the sequence of integer numbers and calculate the arithmetic mean of the sequence elements whose values are smaller than the last element
12	Input a sequence of integers and define the first zero element

Lab # 5
Functions in C++

Purpose: to get practical skills in creation and use of functions in C++.

Theoretical information

A function is a block of code that performs some operation. A function can optionally define input parameters that enable callers to pass arguments into the function. A function can optionally return a value as output. Functions are useful for encapsulating common operations in a single reusable block, ideally with a name that clearly describes what the function does.

Function definition:

type name (*parameter1*, *parameter2*, ...)
{ *statements*;
}

where:

- *type* is the data type specifier of the data returned by the function;
- *name* is the identifier by which it will be possible to call the function (name of function);
- *parameters* (as many as needed): Each parameter consists of a data type specifier followed by an identifier, like any regular variable declaration (for example: `int x`) and which acts within the function as a regular local variable. They allow to pass arguments to the function when it is called. The different parameters are separated by commas;
- *statements* are the function's body. It is a block of statements surrounded by braces `{ }`.

The following function accepts two integers from a caller and returns their sum; `a` and `b` are parameters of type `int`.

```
int addition(int a, int b)
{
    int r = a + b;
    return r;
}
```

The function can be invoked, or called, from any number of places in the program. The values that are passed to the function are the arguments, whose types must be compatible with the parameter types in the function definition.

```
int main()
{
    int x = addition (10, 32);
    int y = addition (x, 66);
    cout<< "The value of y is" << y << endl; // 108
}
```

We called function `addition` passing two values: 10 and 32, that correspond to the `int a` and `int b` parameters declared for function addition.

At the point at which the function is called, the control is lost by the `main` function and passed to the function `addition`. The value of both arguments passed in the call (10 and 32) are copied to the local variables `int a` and `int b` within the function.

Function `addition` declares another local variable (`int r`), and by means of the expression `r=a+b`, it assigns to `r` the result of `a plus b`. Because the actual parameters passed for `a` and *b* are 10 and 32 respectively, the result is 42.

The following line of code:

```
return r;
```

finalizes function `addition`, and returns the control back to the function that called it in the first place (in this case, `main`). At this moment the program follows it regular course from the same point at which it was interrupted by the call to `addition`. And again function `addition` is called with parameters 42 and 66.

In we do not need it to return any value we should use the `void` type specifier for the function. This is a specifier that indicates absence of type: `void`.

```
void printmessage ()
{
  ShowMessage("I'm a function!");
}

… button1_Click …
{
  printmessage ();
}
```

Scope of Variables in Function

The scope of the variables can be broadly classified as:

– Local Variables;
– Global Variables.

The variables defined **local** to the block of the function would be accessible **only within the block** of the function and not outside the function. Such variables are called **local variables**. In other words the scope of the local variables is limited to the function in which these variables are declared.

Let us see this with a small example:

```
int ex(int x,int y)
{
  int z = x + y;
  return z;
}
private: System::Void button1_Click(System::Object^ sender,System::EventArgs^ e)
{
  int b, s=5, u=6;
  b = ex(s,u);
  textBox1->Text = b.ToString();
}
```

In the above program the variables x, y, z are accessible only inside the function ex() and their scope is limited only to the function ex() and not outside the function. Thus the variables x, y, z are local to the function ex(). Similarly one would not be able to access variable b inside the function ex() as such. This is because variable b is local to button1_Click.

Global variables are one which are visible in any part of the program code and can be used within all functions and outside all functions used in the program. The method of declaring global variables is to declare the variable outside the function or block.

For instance:

```
int x,y,z;        // Global variables
float a,b,c;      // Global variables

int sum()
{
  int S = x + y;
  return S;
}

private: System::Void button1_Click(System::Object^ sender,System::EventArgs^ e)
{
  int s, u;         // Local variables
  float w, q;        // Local variables
  s = sum();
  textBox1->Text = s.ToString();
}
```

In the above the integer variables x, y and z and the float variables a, b and c which are declared outside all functions are global variables and the integer variables s, S and u and the float variables w and q which are declared inside the function block are called local variables.

Thus the scope of global variables is between the point of declaration and the end of compilation unit whereas scope of local variables is between the point of declaration and the end of innermost enclosing compound statement. We can use global variables (x, y, z, a, b, c) in both sum()and button1_Click functions and local variable S is only known in sum(), we cannot use it in button1_Click. Local variables s, u, w, q are only known in button1_Click and we cannot use them in sum().

Let us see an example which has number of local and global variable declarations with number of inner blocks to understand the concept of local and global variables scope in detail.

```
int g;   // Global variable
void ex( )
{
  // Scope of g is throughout the program and it can be used inside functions
  g = 30;
}
```

```
private: System::Void button1_Click(System::Object^ sender,System::EventArgs^ e)
{
 int a=Convert::ToInt32(textBox1->Text);//Local in button1_Click, global in if block
 if (a!=0)
 { int b=25;      // Local in if block
   a=45;          // Global in if block
   g=65;          // Global in program
 }
 a=50;
 ex( );
 . . . .
}
```

Scope of b is till the first braces shaded.

Scope of a is till the end of button1_Click brace.

Arguments passed by value and by reference

Until now, in all the functions we have seen, the arguments passed to the functions have been passed *by value*. This means that when calling a function with parameters, what we have passed to the function were copies of their values but never the variables themselves. For example, suppose that we called our function addition using the following code:

```
int x=5, y=3, z;
z = addition ( x, y );
```

What we did in this case was to call the function addition() passing the values of x and y, i.e. 5 and 3 respectively, but not the variables x and y themselves.

This way, when the function addition is called, the value of its local variables a and b become 5 and 3 respectively, but any modification to either a or b within the function addition will not have any effect in the values of x and y outside it, because variables x and y were not themselves passed to the function, but only copies of their values at the moment the function was called.

But there might be some cases where you need to manipulate from inside a function the value of an external variable. For that purpose we can use arguments passed by reference, as in the function duplicate of the following example:

```
void duplicate (int& a, int& b, int& c)
{
  a*=2;   b*=2;  c*=2;
}
private: System::Void button1_Click(System::Object^ sender,System::EventArgs^ e)
{
  int x=1, y=3, z=7;
  duplicate (x, y, z);
  textBox1->Text="x="+x.ToString()+", y="+y.ToString()+", z="+ z.ToString();
}
```

The first thing that should call your attention is that in the declaration of duplicate the type of each parameter was followed by an ampersand sign (&). This

ampersand is what specifies that their corresponding arguments are to be passed *by reference* instead of *by value*.

When a variable is passed by reference we are not passing a copy of its value, but we are somehow passing the variable itself to the function and any modification that we do to the local variables will have an effect in their counterpart variables passed as arguments in the call to the function.

To explain it in another way, we associate `a`, `b` and `c` with the arguments passed on the function call (`x`, `y` and `z`) and any change that we do on `a` within the function will affect the value of `x` outside it. Any change that we do on `b` will affect `y`, and the same with `c` and `z`.

That is why our program's output, that shows the values stored in `x`, `y` and `z` after the call to `duplicate`, shows the values of all the three variables of `button1_Click` doubled.

If when declaring the following function:

```
void duplicate (int& a, int& b, int& c)
```

we had declared it this way:

```
void duplicate (int a, int b, int c)
```

i.e., without the ampersand signs (`&`), we would have not passed the variables by reference, but a copy of their values instead, and therefore, the output on screen of our program would have been the values of `x`, `y` and `z` without having been modified.

Passing by reference is also an effective way to allow a function to return more than one value. For example, here is a function that returns the previous and next numbers of the first parameter passed.

```
void prevnext (int x, int& prev, int& next)
{
  prev = x-1;
  next = x+1;
}
private: System::Void button1_Click(System::Object^ sender,System::EventArgs^ e)
{
  int x=100, y, z;
  prevnext (x, y, z);
  textBox1->Text = "Previous=" + y.ToString();
  textBox2->Text = "Next=" + z.ToString();
}
```

Examples of programs with functions

Example 5.1. Calculate the value of the following expression: $x = \frac{\sqrt{6}+6}{2} + \frac{\sqrt{13}+13}{2} + \frac{\sqrt{21}+21}{2}$.

Solution. All three summands in the given formula are similar to each other. Each of them can be written using the general formula: $\frac{\sqrt{a}+a}{2}$, where a – the number that in the first term is 6, in the second – 13, and in the third – 21. The better way is to create a function with parameter a, and to call this function three times for each of the values: 6, 13, 21.

The program code of the function and it's call in the main program:

```
#include <iostream>
#include <math.h>
using namespace std;
double f (double a)
{ return (sqrt(a)+a)/2;
}
int main ()
{ double x = f(6) + f(13) + f(21);
  cout<<" Result x = "<< x << endl;
  system ("pause>>void");
  return 0;
}
```

Result x = 25.32

Example 5.2. Calculate the value of expression: $x = \frac{13+\sqrt{7}}{7+\sqrt{13}} + \frac{15+\sqrt{12}}{12+\sqrt{15}} + \frac{21+\sqrt{32}}{32+\sqrt{21}}$.

Solution. The formula of this example is similar to the formula of the previous example. The only difference is that this function has two parameters.

The program code of the function and the main program:

```
include <iostream>
#include <math.h>
using namespace std;
double f(double a, double b)
{ return (a+sqrt(b))/(b+sqrt(a));
}
int main ()
{ double x = f(13,7) + f(15,12) + f(21,32);
  cout<<" Result x = "<< x << endl;
  system ("pause>>void");
  return 0;
}
```

Result x = 3.37

Example 5.3. Create a function to calculate the greatest common divisor (GCD) of two natural numbers. Input three numbers and calculate their GCD.

Solution. Calculating the GCD can be done in different ways.

Method 1: **Decomposition of numbers into simple multipliers** (see function **`GCD1()`**) is a simple bulkhead all the numbers with test for divisibility without the remainder of both numbers. GCD is the first number for which the checked condition is true. The disadvantage of this method is a low speed.

Method 2: There are some **Euclidean algorithms** for computing GCD of pair positive integers that use different recurrence relations. In the simplest case, on each iteration Euclidean algorithm generates a new pair: the smaller number and the difference between the bigger and the smaller numbers. The process is repeated until both numbers become equal. Found number is the greatest common divisor of the original pair (see function **`GCD2()`**).

Method 3: **Binary Euclidean algorithm** (see function **`GCD3()`** with operator `while` loop and **`GCD4()`** – `do-while` cycle) is faster than ordinary Euclidean algorithm. The algorithm is based on the following observations:

– If a and b are both even,
then gcd(a,b)=2gcd(a/2,b/2)gcd(a,b)=2gcd(a/2,b/2);
– If a is even and b is odd, then gcd(a,b)=gcd(a/2,b)gcd(a,b)=gcd(a/2,b);
– Otherwise both are odd,
and gcd(a,b)=gcd(|a−b|/2,b)gcd(a,b)=gcd(|a−b|/2,b).

The three conditions cover all possible cases for a and b. The algorithm systematically reduces a and b by repeatedly testing the conditions and accordingly applying the reductions. Note that the first condition, i.e., a and b both being even, applies only in the very beginning of the procedure. Thus, the algorithm first factors out the highest common power of 2 from a and b and stores it in g. In the remainder of the computation only the other two conditions are tested. The computation terminates when one of the operands becomes zero. The algorithm is given as follows.

Method 4: **Recursive Euclidean algorithm** (see function **`GCD5()`**) allows to get rid of loop statement which organized the next iteration, by recursive calls of the same function with reduced offset value for a couple of numbers.

Here is the program code for all of these methods of calculation of the GCD.

The program code of the function and the main program:

```
#include <iostream>
using namespace std;
int GCD1( int a,  int b)
{ int n=1;
  if(a > b) n = b; else n = a;
  n++;
  do  n--;
  while( a%n != 0 || b%n != 0);
  return n;
}
```

```
int GCD2( int a,  int b)
{ while( a!= b)
   if (a > b) a -= b; else b -= a;
  return a;
}
int GCD3( int a,  int b)
{ while ( a && b )
    if ( a>=b ) a %= b;
    else  b %= a;
  return a | b;
}

int GCD4( int a,  int b)
{ int n;
  do
  { n = a % b;
    a = b;
    b = n;
  } while (b > 0);
 return a;
}

int GCD5( int a,  int b)
{
  if(!b) return a; else return GCD5(b, a % b) ;
 // or a short record:   return b ? nsd5 (b, a % b) : a;
}
int main()
{
  system("color F0");
  int x, y;
  cout <<" Enter two natural (positive integer) numbers \n --> ";
  cin >> x >> y;
  if (x<=0 || y<=0) cout << "Data is incorrect!" << endl;
  else
  { cout<<" GCD:  "<< GCD1(x,y) <<endl;
    cout<<" GCD:  "<< GCD2(x,y) <<endl;
    cout<<" GCD:  "<< GCD3(x,y) <<endl;
    cout<<" GCD:  "<< GCD4(x,y) <<endl;
    cout<<" GCD:  "<< GCD5(x,y) <<endl;
  }
  system ("pause");
  return 0;
}
```

```
Enter two natural (positive integer) numbers
--> 125        80
GCD:  5
GCD:  5
GCD:  5
GCD:  5
GCD:  5
```

You can calculate the GCD of three numbers x, y and z using any of these functions. To do this, you should call the function in the following way:

```
cout << " GCD:  " << GCD1(x, GCD1(z,y)) <<endl;
```

```
Enter three natural (positive integer) numbers
--> 450        99       135
GCD:  9
```

Example 5.4. Create a function that writes digits of the number in the reverse order (e.g, 1208 → 8021). Enter a number and output its reverse number.

The program code of the function and the main program:

```
#include <iostream>
using namespace std;
int reverse(int n)
{ int a=0;
  do
  { a = a*10 + n%10;
    n = n/10 ;
  } while(n);
  return a;
}
int main()
{ int x;
  cout <<"Input an integer positive number \n --> "; cin >> x;
  cout<<"Reverse number:  "<< reverse(x) <<endl;
  system ("pause>>void");
  return 0;
}
```

Results:

```
Input an integer positive number
-->  14089
Reverse number:   98041
```

Example 5.5. Create a function that swaps the first and the last digits. Enter a number and change it using this function.

The program code of the function and the main program:

```
#include <iostream>
using namespace std;

int first_last(int n)
{ int a = n, i = 1, p = n % 10;
  while (n >= 10)
  { i *= 10;
    n /= 10;
  }
  return a - n*i - p + n + p*i;
}
```

```
int main()
{ int x;
  cout<<"Input an integer positive number \n --> ";  cin>>x;
  cout<<"The number after changing:"<< first_last(x) <<endl;
  system ("pause>>void");
  return 0;
}
```

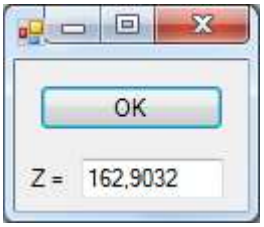
```
Input an integer positive number
-->  120569
The number after permutation:   920561
```

Example 5.6. Calculate the value of the following expression: $z = \dfrac{2 \cdot 5! + 3 \cdot 8!}{6! + 4!}$.

Solution. Factorial meets four times in the formula. Therefore, it is appropriate to calculate the value of the factorial in function and call it four times with different parameters.

The program code of the function and the main program:

```
long long fact(int n)
{
  long long c=1;
  for(int i=1; i<=n; i++) c*=i;
  return c;
}
```




```
private: System::Void button1_Click(System::Object^ sender,System::EventArgs^ e)
{   double  z = (2.0*fact(5)+3*fact(8))/(fact(6)+fact(4));
    textBox1->Text = Convert::ToString(z);
}
```

Example 5.7. Calculate three expressions in one function $y = \operatorname{arctg}^2 z$; $z = \sqrt{x^2 + a}$; $a = \lg\left|\dfrac{x+b}{e^{b+0.1}}\right|$. Input values x and b. Create a function to calculate three values, one of which can be sent via operator `return`, and the rest – the link.

Solution. The order of evaluation of expressions: the value of `a` should be calculated first, then `z` (which depends on `a`) and finally – the value of `y`, which depends on `z`.

The program code of the function and the main program:

```
#include <iostream>
#include <math.h>
using namespace std;
double f(double x, double b, double& a, double& z)
{
  a=log10(fabs((x+b)/exp(b+0.1)));
  z=sqrt(x*x+a);
  return pow(atan(z),2);
}
```

```
Input value x= 2.6
Input value b= 0.3
y=  1.46549
a=  0.28868
z=  2.65494
```

```
int main()
{
  double x, b, a, z;
  cout <<" Input value x= ";  cin >> x;
  cout <<" Input value b= ";  cin >> b;
  cout<<" y=  "<< f(x,b,a,z) <<endl;
  cout<<" a=  "<< a << endl <<" z=  "<< z <<endl;
  system ("pause>>void");
  return 0;
}
```

Control questions

1) What is the `return` statement used for?
2) What is the type of function, which returns no value?
3) What can you say about the function with the following header:
   `int fun(double x)`
4) How can we return several results from a function?

Individual task

Table 5.1 (basic level)

Problems with functions with one result

№ var.	Problems
1	Create a function to calculate $\frac{x+\sin y}{\sin x+y}$. Calculate $y=\frac{1+\sin 4}{\sin 1+4}+\frac{7+\sin 5}{\sin 7+5}+\frac{3+\sin 2}{\sin 3+2}$ with this function
2	Create a function to calculate factorial. Calculate $C_n^m=\frac{n!}{m!(n-m)!}$ with the help of this function
3	Create a function to calculate geometric mean of three numbers with the formula $\sqrt[3]{x\cdot y\cdot z}$. Input three numbers and calculate their geometric mean
4	Create a function, which defines the smaller of two numbers. Input numbers a and b and calculate $z=\min(a,2b)+\min(2a-b,b)$ with this function
5	For triangle with sides a, b and c the value of median to side a is calculated according the formula: $0.5\sqrt{2b^2+2c^2-a^2}$. Create a function to calculate median. Calculate all medians of triangle with the help of this function
6	Create a function to calculate $\frac{\sqrt{x}+x}{y+\sqrt{y}}$. Calculate $\frac{\sqrt{5}+5}{7+\sqrt{7}}+\frac{\sqrt{12}+12}{15+\sqrt{15}}+\frac{\sqrt{22}+22}{32+\sqrt{32}}$ with the help of this function
7	Create a function to calculate $\mathrm{sh}(x)=(e^x-e^{-x})/2$. Input x and calculate $\mathrm{sh}(x)\,\mathrm{tg}(x+1)-\mathrm{ctg}^2(2+\mathrm{sh}(x-1))$ with the help of this function

End of table 5.1

№ var.	Problems
8	Create a function, which defines the bigger of two numbers. Input two numbers (a and b) and calculate $y = \max(2a-b, 2b-a) + \max(a, b)$ with the help of this function
9	Create a function to calculate a sum of digits of the number. Input three numbers and define sums of digits for each number with the help of this function
10	Create a function, which determines an absolute value (in magnitude) of the number (without the use of standard mathematical functions) and calculate absolute values of three entered numbers
11	Create a function to calculate a factorial and calculate $z = \frac{3 \cdot 7! - 2 \cdot 6!}{5! + 3!}$ with this function
12	Create a function to calculate $\frac{\sqrt{x} + x}{2}$. Calculate $\frac{\sqrt{5}+5}{2} + \frac{\sqrt{11}+11}{2} + \frac{\sqrt{17}+17}{2}$ with the help of this function

Table 5.2 (basic level)

Problems with functions with three results

№ var.	Values to calculate	Arguments
1	$y = a\sin^2 b + b\cos^2 a;\ a = \sqrt[3]{\lvert b + c\rvert};\ b = \sqrt{x}$	x, c
2	$y = a^2 + b^2;\ a = \ln\lvert x\rvert;\ b = \mathrm{e}^k + a$	x, k
3	$y = \mathrm{e}^x + 5.8^c;\ c = a^2 + \sqrt{b};\ a = b^3 + \ln\lvert b\rvert$	x, b
4	$y = \sqrt[3]{\lvert a - b\rvert};\ a = \lg x;\ b = \sqrt{x^2 + t^2}$	x, t
5	$y = a^3 / b^2;\ a = \mathrm{e}^{\sqrt{\lvert x\rvert}};\ b = \left(\sin p^2 + x^3\right)$	x, p
6	$y = p^2 + t^4;\ p = x^2 - \sqrt{\lvert x\rvert};\ t = \sqrt[3]{x + a^2}$	x, a
7	$y = c^3 / \cos c;\ c = a^2 + b^2;\ a = \sqrt{\lvert x\rvert} + \mathrm{e}^{\sqrt{b}}$	x, b
8	$y = \sin^3(a + b);\ a = t^3 + \sqrt{b};\ b = \lg^2\lvert x\rvert$	x, t
9	$y = \mathrm{arctg}^3 x^2;\ x = p + k;\ k = \sqrt{p + t^2}$	t, p
10	$y = \cos^2(a + \sin b);\ a = \sqrt{\lvert x\rvert};\ b = x^4 + m^2$	m, x
11	$y = \sin^3 a + \cos^2 x;\ a = c + k^2;\ c = \mathrm{arctg}\lvert x\rvert$	k, x
12	$y = \mathrm{e}^{\sqrt{\lvert x\rvert}} + \cos x;\ x = a + c^3;\ a = \sin^5 b$	b, c

Application

Basic data types of C ++

Type	Name	Size, byte	Range	Examples of possible values	Type the numbers
`char`	character	1	-128…127	'a', '\n', '9'	integer
`unsigned char`		1	0…255	1, 233	
`short`	short integer	2	-32 768…32 767	1, 153, -349	
`unsigned short`		2	0…65 535	0, 4, 65 000	
`int`	integer	4*	-2 147 483 648… …2 147 483 647	–30 000, 0, 690	
`unsigned int`		4	0…4 294 967 295	2 348, 60 864	
`long`	long integer	4	-2 147 483 648… …2 147 483 647	-30 000, 0, 690	
`float`	single floating point	4	$3.4\cdot10^{-38}$…$3.4\cdot10^{38}$	3.23, -0.2 100.23, 12, -0.947, 0.0001, 6.34e–3, 4e5	real
`double`	double floating point	8	$1.7\cdot10^{-308}$ … …$1.7\cdot10^{308}$		
`long double`	long floating point	10	$3.4\cdot10^{-4932}$… …$1.1\cdot10^{4932}$		
`bool`	boolean	1	false or true	false(0), true(>=1)	
`enum`	enumerable	2 or 4			
`void`	valueless				

* – Size of type `int` can be 2 or 4 bytes in dependence on compiler and hardware

Recommended literature list

1. C++ // University of Cambridge: Department of Engineering: Computing Help. [Electronic source]. – Access mode: http://www-h.eng.cam.ac.uk/help/tpl/languages/C++.html.

2. C++ for C programmers // University of Cambridge: Department of Engineering. [Electronic source]. – Access mode: http://www-h.eng.cam.ac.uk/help/tpl/talks/CtoC++.html

3. C++. Основи програмування. Теорія та практика: підручник / [О. Г. Трофименко, Ю. В. Прокоп, І. Г. Швайко, Л. М. Буката та ін.] ; за ред. О. Г. Трофименко. – Одеса : Фенікс, 2010. – 544 с.

4. C++. Теорія та практика: навч. посіб. з грифом МОНУ / [О. Г. Трофименко, Ю. В. Прокоп, І. Г. Швайко, Л. М. Буката та ін.] ; за ред. О. Г. Трофименко. – Одеса : ВЦ ОНАЗ, 2011. – 587 с.

5. C++. Loops. // Wikiversity. [Electronic source]. – Access mode: https://en.wikiversity.org/wiki/C%2B%2B/Loops.

6. Control Statements // University of Cambridge: Department of Engineering: Computing Help: Languages: C++. [Electronic source]. – Access mode: http://www-h.eng.cam.ac.uk/help/languages/C++/c++_tutorial/control.html.

7. For statement (C++). [Electronic source]. – Access mode: https://msdn.microsoft.com/en-us/library/b80153d8.aspx.

8. Iteration Statements (C++). [Electronic source]. – Access mode: https://msdn.microsoft.com/ en-us/library/7ftwh93a.aspx.

9. Programing C++: Методическая система "Web-технологии". [Электронный ресурс]. – Режим доступа: http://metodweb.yomu.ru/en/oop-on-c.

10. Prokop Y. V. Basics of programming. Basic algorithms : methodical instructions for laboratory training and exercises / Prokop Y. V., Trofimenko E. G., Bukata L. N. – Part 1. – Odessa : ONAT named after A. S. Popov, 2016. – 76 p.

11. The C++ Programming Language. [Electronic source]. – Access mode: http://www.stroustrup.com/C++.html#learning

12. Visual C++ .NET: пособие для разработчиков C++ / [А. Корера, С. Фрейзер, С. Джентайл и др.]. – М. : ЛОРИ, 2003. – 398 с.

13. Дейтел Х. М. Как программировать на C++; пер с англ. / Х. М. Дейтел, П. Дж. Дейтел. – М. : ООО "Бином-Пресс", 2008. – 1456 с.

14. Довбуш Г. Ф. Visual C++ на примерах / Г.Ф. Довбуш, А.Д. Хомоненко ; под ред. проф. А.Д. Хомоненко. – СПб. : БХВ-Петербург, 2007. – 528 с.

15. Зиборов В. В. MS Visual C++ 2010 в среде .NET. Библиотека программиста / Зиборов В. В. – СПб. : Питер, 2012. – 320 с.

16. Основи програмування. Базові алгоритми : метод. вказівки для лаб. і практ. робіт / О. Г. Трофименко, Ю. В. Прокоп, І. Г. Швайко, Л. М. Буката. – Ч. 1. – Одеса: ВЦ ОНАЗ ім. О. С. Попова, 2014. – 108 с.

17. Основи програмування. Опрацювання структурованих типів : метод. вказівки для лаб. і практ. робіт / О. Г. Трофименко, Ю. В. Прокоп, І. Г. Швайко,

Л. М. Буката. – Ч. 2. – Одеса: ВЦ ОНАЗ ім. О. С. Попова, 2014. – 130 с.

18. Основи програмування. Програмне опрацювання файлів : метод. вказівки для лаб. і практ. робіт / О. Г. Трофименко, Ю. В. Прокоп, І. Г. Швайко, Л. М. Буката. – Ч. 3. – Одеса: ВЦ ОНАЗ ім. О. С. Попова, 2015. – 80 с.

19. Стивен Прата. Язык программирования C++. Лекции и упражнения : учебник ; пер. с англ. / Стивен Прата. – СПб.: ООО "ДиаСофтЮП", 2005. – 1104 с.

20. Страуструп Б. Язык программирования C++. Специальное издание; пер. с англ. / Страуструп Б. – М. : ООО "Бином-Пресс", 2006. – 1104 с.

21. Трофименко О. Г. Комп’ютерні технології та програмування. Базові алгоритми : метод. вказівки для лаб. і практ. робіт / Трофименко О. Г., Прокоп Ю. В., Буката Л. М. – Ч. 1. – Одеса: ВЦ ОНАЗ ім. О. С. Попова, 2016. – 108 с.

22. Трофименко О. Г. Комп’ютерні технології та програмування. Програмування структурованих даних : метод. вказівки для лаб. і практ. робіт / Трофименко О. Г., Прокоп Ю. В., Буката Л. М. – Ч. 2. – Одеса: ВЦ ОНАЗ ім. О. С. Попова, 2016. – 134 с.

23. Трофименко О. Г. Комп’ютерні технології та програмування. Програмне опрацювання файлів : метод. вказівки для лаб. і практ. робіт / Трофименко О. Г., Прокоп Ю. В., Буката Л. М. – Ч. 3. – Одеса: ВЦ ОНАЗ ім. О. С. Попова, 2016. – 80 с.

24. Трофименко О. Г. Створення багатомодульних програмних проектів для опрацювання даних у файлах засобами Visual C++: метод. вказівки для виконання курсової роботи з дисципліни "Основи програмування" / Трофименко О. Г., Прокоп Ю. В. – Одеса: ВЦ ОНАЗ ім. О. С. Попова, 2015. – 44 с.

25. Хортон А. Visual C++ 2010: полный курс.; пер. с англ. / Хортон А. – М. : ООО "И. Д. Вильямс", 2011. – 1216 с.

CONTENT

Methodical instructions for laboratory training and exercises

Y. V. Prokop,
L. N. Bukata,
E. G. Trofimenko

ALGORITHMIZATION AND PROGRAMMING

Part 1

BASIC ALGORITHMS