# Chapter 3. The structure of the program is in C++. Operation

## 3.1 Program Entry Point

The main part of the program is the main() function. When you run a console application written in the C++ programming language, the computer's operating system passes control to a function named main. The main() function cannot be called from other program functions, it is the entry point into the program.

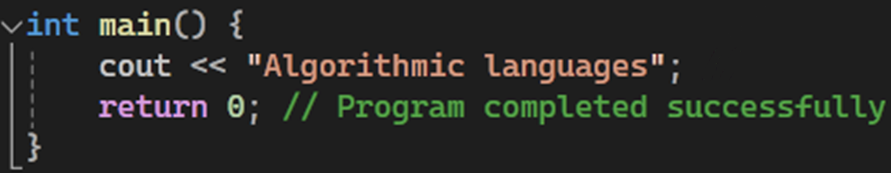
The main function is declared as follows: "int main()" or "void main()".

Parentheses after the function name are used to pass parameters. In this case, the operating system does not pass any parameters to the main() function, so there is nothing in parentheses.

The return type is specified before the function name. When the main() function is accessed, an integer value is returned to the operating system. Depending on the return value, the operating system determines whether the program terminated correctly.

If this value is not returned, the operating system understands that the program has crashed. To return an integer value, a string is added before the function completes: "return 0;". If the program quits and returns 0, it means that no errors have occurred.

After declaring a function, curly braces are written in which the body of the program is located. An example of a program that displays text on the screen is shown in Picture 3.1.

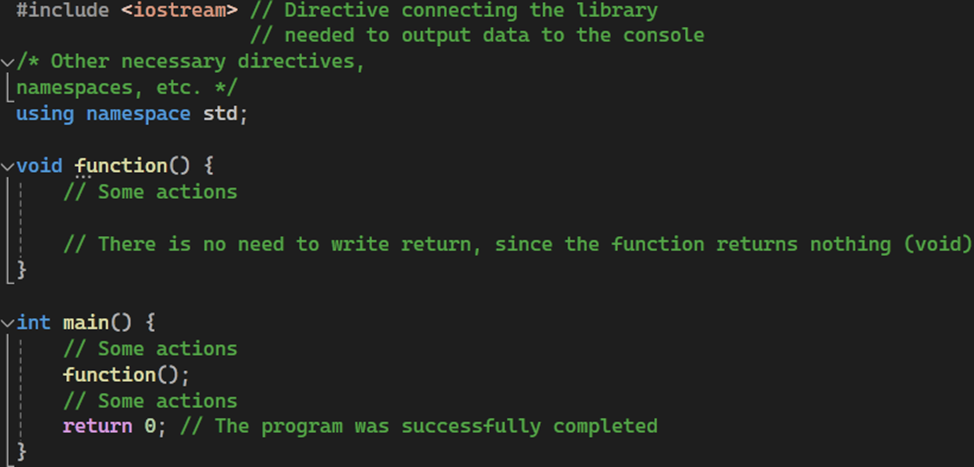


Picture 3.1 – Example of C++ Text Output Program Code

## 3.2 Preprocessor and namespace directives

In addition to the main() function, any program contains other important components. A C++ program starts with preprocessor directives, such as "#include", which tell the compiler which libraries or files it should include in the program [1]. Directives help developers use out-of-the-box language features and capabilities. A general view of the program in the algorithmic language C++ is shown in Picture 3.2.

The string "using namespace std;" indicates that the standard namespace "std" is being used. With the help of namespaces, the program makes sure that variable names are not repeated.



Picture 3.2 – Structure of the general view of a C++ program

For example, with the "<iostream>" library attached, but without "using namespace std;", you need to write "std::cout<<"some data";" to print messages to the console, and with "using namespace std;" you use the cout output operator.

Namespaces are important because they allow you to isolate pieces of code from each other. Without a namespace declaration, name conflicts can arise, especially if you are using third-party libraries or have large projects with many files. Namespaces help prevent such conflicts by dividing your code into logical groups and providing a unique space for each.

Next, we will consider the functions from the example (Picture 3.2). The most important of these is the main() function, which starts the execution of the program. The rest of the functions can be called from main(), in Picture 3.2 such a function is function(), which does not return anything.

## 3.3 Binary and unary arithmetic operations

In the C++ language, there are several arithmetic operations that can be used. These are addition, subtraction, multiplication, division and the remainder of division are binary operations. These operations allow you to perform simple mathematical calculations in a program: "+" is addition; "-" – subtraction; "\*" – multiplication; "/" – division; "%" is the remainder of the integer division.

Binary arithmetic operations can be used in conjunction with the assignment operation. The following operators are analogous to the operators given in the comments (//):

1. object += expression; object = object + expression

2. object -= expression; object = object – expression

3. object \*= expression; object = object \* expression

4. object /= expression; object = object/expression

5. object %= expression; object = object % expression

Unary operations are performed on a single operand. In the C++ language, it is possible to use unary operations such as increment, decrement, negation, and logical negation. These operations allow variable values to change to 1 or perform Boolean operations on them.

1. "++" is an increment, i.e. an increase of 1

2. "--"is a decrease, that is, a decrease of 1

3. "-" is a change of sign

The result of evaluating an expression that contains an increase or decrease operation depends on whether the action sign is located before or after the variable.

If an operation is placed before a variable or expression, the value of the variable is first changed to 1, and then the value is used to perform the next operations. If one of these operations is located after a variable, the operation is performed first, and then the value of the variable is changed to 1 [23].

int one = 5;

int two = 6;

int three = 0;

three = one\*++two; three = 35, because in the multiplication operation, two=7.

three = one \* two++; three = 30, because in the multiplication operation, two=6.

After operations, two = 7.

## 3.4 The process of building a program

The whole process consists of several stages: pre-processing, compilation, linking, and execution.

*Preprocessing Conversion:*

*Preprocessor directives:* Header files are included in the program text using the preprocessor directive #include. All preprocessor directives begin with the # sign, which must be the very first character of the string.

*Preprocessor:* In this step, the preprocessor's directives are processed. The preprocessor performs tasks such as including header files, macro definitions, and conditional compilation.

*Compile:*

*Compiler:* In this step, the compiler converts the preprocessor-processed source code of the program into machine code. This process includes parsing, semantic analysis, and optimization. The result of the compilation is an object file (or several object files) that contains machine code, but is not yet an executable file.

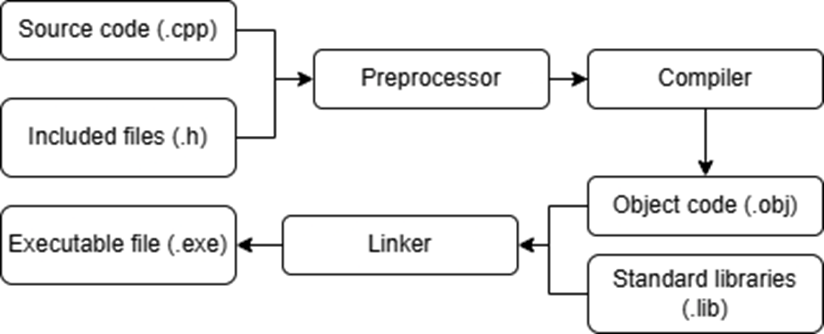
*Linking (layout):*

*Linker:* In this step, a single executable is created from multiple object files. The linker links the object files, resolves external references (such as function calls from other files or libraries), and adds the necessary system libraries.

*Execution:*

*Running a program:* An executable file is loaded into memory by the operating system and its execution begins. The execution process is controlled by the main() function, which is the entry point into the program.

A scheme for building a program in the C++ algorithmic language is shown in picture 3.3.

Picture 3.3 – Scheme of building a program in C++

## 3.5 Creating a project in Visual Studio and debugging the program

In a development environment, the process of building a program is simplified. The development environment will automatically perform all the necessary steps, including compilation and linking, and show the result.

It all starts with the opening of the project. On the main page, you need to select "Create project". From the proposed options, you should select "Empty project".

The name of the project is entered in the "name" column.

After creating a project, you need to create a file to store the program code. To do this, right-click on the project name, select "Add"/"Create Item". In the window that appears, select "C++ File" and give it a name. In this open window, the program code is written.

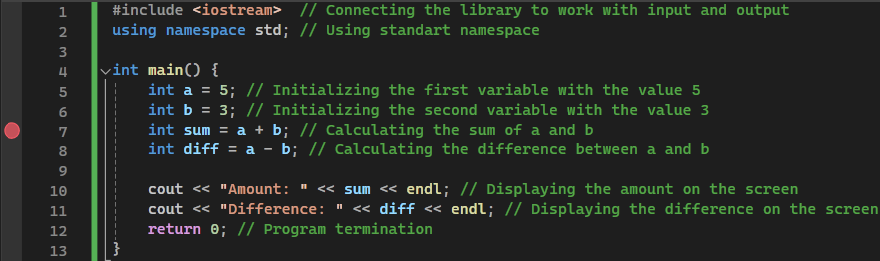
If you click "Local Debugger" or the keyboard shortcut after typing CTRL+F5, the program will automatically go through all three stages of debugging and run if there are no errors. There is also step-by-step debugging, which allows you to go through the program step by step, finding and correcting errors and inaccuracies. The developer can set breakpoints on the desired lines of code, and then monitor changes in the values of variables, execute the code step by step, and analyze its state. It is a powerful tool for debugging and fixing errors.

To start step-by-step debugging, you need to press the F10 button. The console will open in which the data from the program will be displayed. Debugging starts at the –main() entry point.

The yellow cursor on the left indicates the line that should be executed. Now, when you press the F10 button – step with the input of callable functions – or F11 – step without the input of callable functions, the cursor will go down, thereby executing the program gradually, line by line.

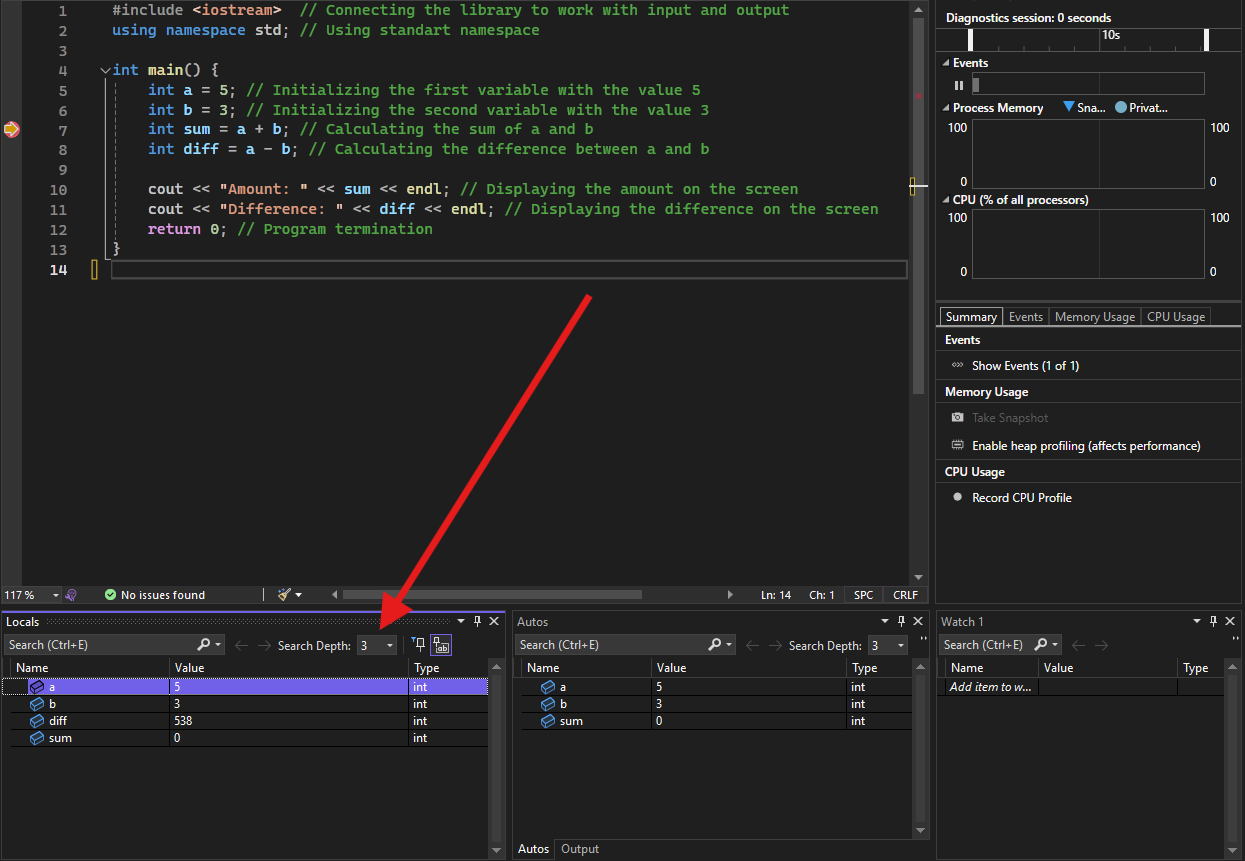
At the bottom of the window, there are three tabs for tracking variables: visible variables, local values, and control values.

An example of tracking the values of variables in a program: Calculating the sum of two numbers and the difference between them (Picture 3.4).



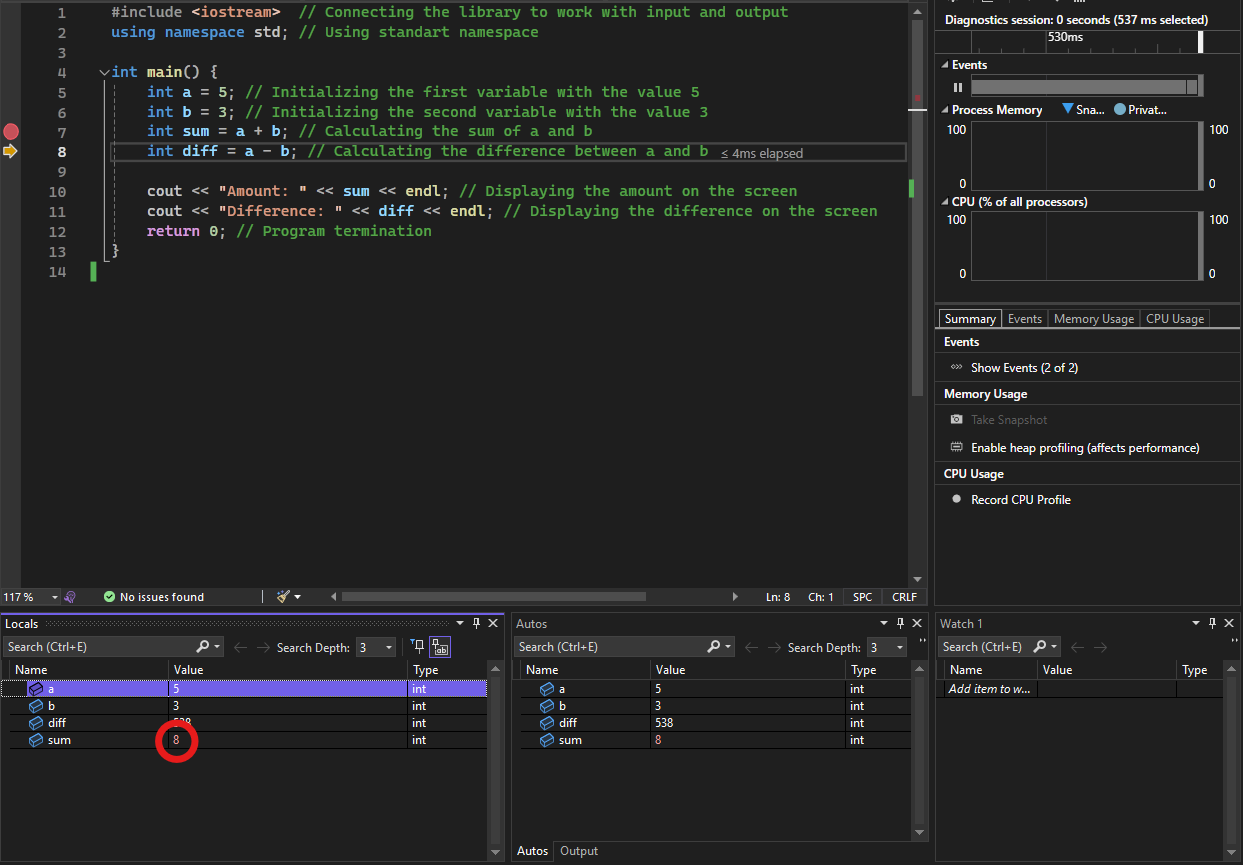
Picture 3.4 – Code of the variable value tracking program

A breakpoint is set on line 7. After starting the program in debug mode, the program will be stopped at it. Picture 3.5 shows tabs for tracking variables, variables a and b are initialized, but variables sum and diff are not, so they have random values.



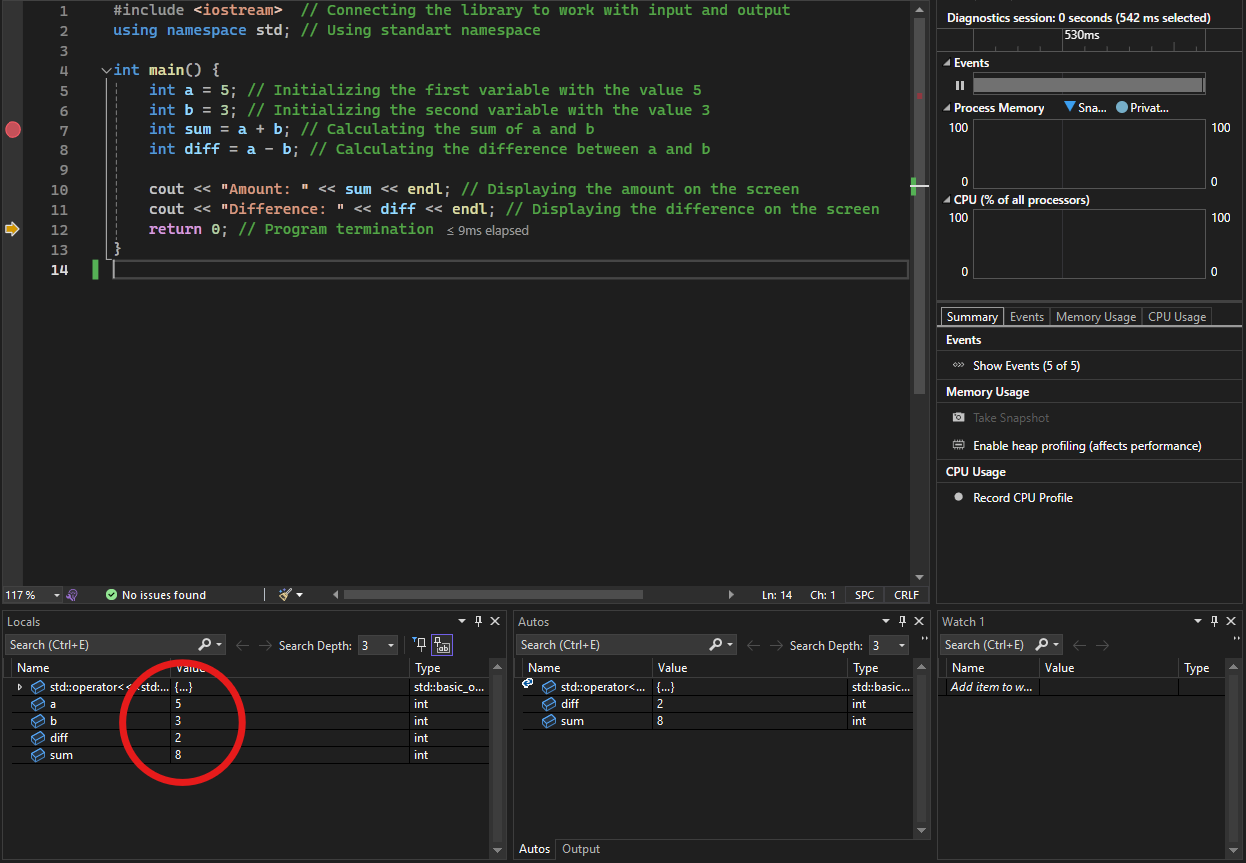
Picture 3.5 – Variable Tracking tabs

After pressing the F10 or F11 button, the sum variable will be initialized and will receive the corresponding value (picture 3.6).



Picture 3.6 – Screenshot of the program after completing one step

After executing the program (but before executing the return 0 line), all variables are initialized. The value of any variable can be changed and this will be displayed on the tabs for tracking variables (picture 3.7).



Picture 3.7 – Screenshot of the program after its execution

To consolidate the material, it is necessary to solve practical problems:

1. Develop a C++ program that outputs the message "Algorithmic languages" to the console.
2. Determine what the expression

i = ++ i \* i++;

will be equal when the initial value is i = 1.

1. Find the error in the line

int a=2.8; int b=12;

1. Develop a C++ program that reads two integers from the keyboard, adds them up and displays the result on the screen.
2. Find the result of executing the following code in C++:

int x = 5;

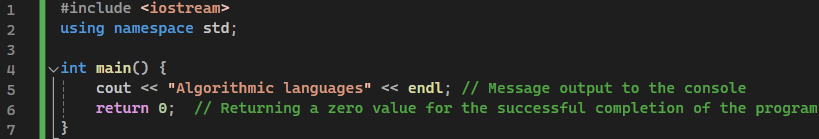
int y = 10;

int z = x++ + ++y;

Answers to practical tasks with explanations:

Task 1: Develop a C++ program that outputs the message "Algorithmic languages" to the console.

Solution to problem 1: The program starts by connecting the <iostream> library, which is necessary to work with the cout function that outputs data to the console. The main() function is the entry point to the program, and it is from there that code execution begins. The message is output using the << operator, and endl is used for line wrapping (Picture 3.8).



Picture 3.8 – Solution of problem 1

Task 2: Determine what the expression

i = ++ i \* i++

will be equal to when the initial value is i = 1.

To understand the result, you need to consider how increment works in C++:

++i is a prefix increment: first, the value of a variable is incremented by 1, then it is used in an expression.

i++ is a postfix increment: first, the value of a variable is used in an expression, and then incremented by 1.

Solution of problem 2:

Initial value i = 1.

++i increments i to 2, then this value is used in the expression.

After that, multiplication is performed: 2 \* 2 = 4.

After using i++, the value of i increases by 1 and becomes 3.

Thus, the result of executing the expression will be i = 4.

Task 3: Find the error in the line

int a=2.8;

int b=12;

The error lies in the fact that the int type is used for the variable a, which is initialized with the value 2.8. The int type is intended for integers, and when assigning a fractional number, it will be truncated to an integer part, i.e. a will become 2, not 2.8. If you want to save the fractional part, you must use the float or double type. Otherwise, the fractional part will be lost when casting the type.

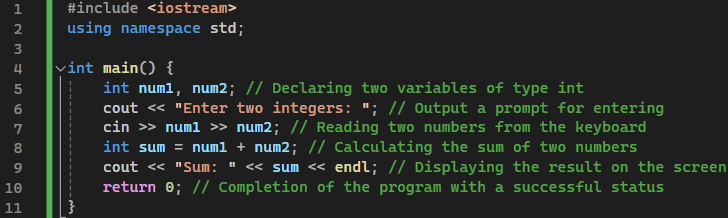
Solving problem 3:

float a = 2.8; // Or double a = 2.8;

int b = 12;

Task 4: Develop a C++ program that reads two integers from the keyboard, adds them up and displays the result on the screen.

The program starts by declaring two variables num1 and num2, which will be used to store the numbers entered by the user. cin is used to receive data from the user. After entering the numbers, the program calculates their sum and displays the result on the screen using cout.



Picture 3.9 – Solution of problem 4

Task 5: Find the result of executing the following code in C++:

int x = 5;

int y = 10;

int z = x++ + ++y;

Solving problem 5:

Initial values: x = 5, y = 10.

x++ is a postfix increment: first, the value of x is used in the expression, then it is increased by 1. The value x = 5 is involved in the expression, but after that x becomes equal to 6.

++y is a prefix increment: first y is increased by 1, and then used in the expression. The y value increases to 11, and it is this value that is involved in the addition operation.

In the expression z = x++ + ++y, the addition is performed: 5 + 11 = 16.

Thus, the result of the execution:

x will be equal to 6,

y will be equal to 11,

z will be equal to 16.

Answer: After executing the code

x = 6, y = 11, and z = 16.