1.1 INTRODUCTION IN C++

Programming languages are means of communication between the programmer and the computer. They are languages similar to human language. They contain words (usually quite few), punctuation marks, mathematical operations, and writing rules. Programs that run on any computer are written in a programming language. There are many programming languages, such as **C, C++, Pascal, Java, Python, PHP, JavaScript**, etc.

The program written in a programming language is called the **source program** and needs to be translated into a language understood by the processor, called machine code or an executable program. For some programming languages, the translation operation is called **compilation** (in the case of C, C++, Pascal, etc.), while for others (PHP, Python, JavaScript, etc.), the translation operation is called **interpretation**. The translation is performed by a specialized program called a compiler or an interpreter.

The key words in defining a programming language are:

- **Syntax** - represents the totality of rules based on which the elements of the language are formed
- **Semantics** - defines the meaning of syntactically correct constructions (of language elements)
- **Pragmatics** - defines the way of using language elements

Notice that these three aspects are also studied in the case of a natural language, for example English. But unlike the English language, where, if we make a grammatical mistake or use an expression incorrectly in a context, there are quite high chances that the interlocutor will understand us, when we communicate with a computer, we must strictly follow the rules of communication. The computer doesn't make assumptions, it doesn't guess what you wanted to tell. If you do not express yourself correctly, you will only get error messages from the computer.

The C++ language is a **compiled language**. The description of an algorithm in a programming language is called a **program**.

The **stages** of writing a program in C++ are:

- **editing the C++ program**; the source file is obtained, with the extension ".cpp"
- **compiling the source file**; here the syntactic correctness of the program is checked (the correctness of the words used, the presence of punctuation marks, etc.); if the program is syntactically correct, the object file will be obtained, with the extension ".o" or ".obj"

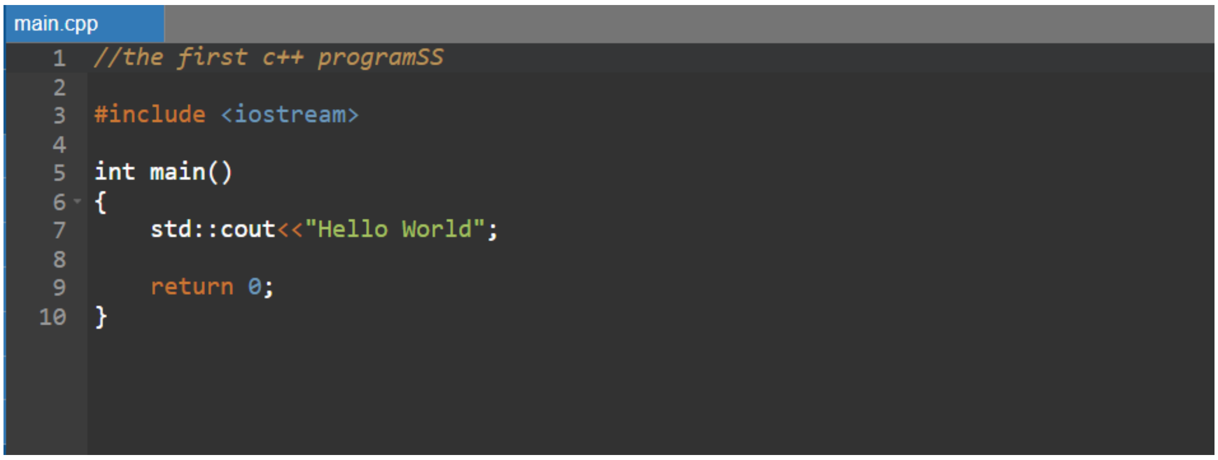
- **link editing**; links are established between the current object file and other object files, of the programmer or included in the compiler; after this stage the executable program is obtained. In Windows, executable files have the extension ".exe"

At the end, the executable program can be **launched** (run).

The first C++ program

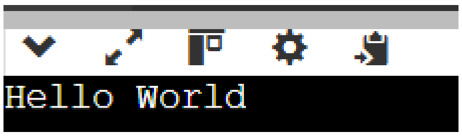
How do we write a C++ program? We need at least a **text editor** for writing the source and a **C++ compiler**. Although the source file can be made with any text editor, most of the time we use an **IDE**. A widely used C/C++ IDE is Code::Blocks.

Consider a first C++ program:



```
//the first c++ programSS

#include <iostream>

int main()
{
    std::cout<<"Hello World";

    return 0;
}
```

If we compile and run this program, the screen will show:



```
Hello World
```

Let's analyze this program. It consists of several lines:

- // first C++ program

  This line represents a **comment**. Comments are explanatory texts that do not influence the behavior of the program. They are for programmers to understand the meaning of the program more quickly. This comment starts at the two slash characters // and ends at the end of the line.

- #include <iostream>

  Lines starting with # are called preprocessor directives. They are interpreted before the actual compilation, by a program called a preprocessor. In our case, the #include directive asks the preprocessor to include in the source a section of the standard C++ code, the iostream **header**, which allows reading and display operations – in our case, displaying the "*Hello world*" message on the screen.

- int main()

  This line represents the **declaration of a function**. Essentially, a function is a group of instructions that has a given name; in this case, the function is called main and consists of all the statements that follow. We will discuss functions in detail later.

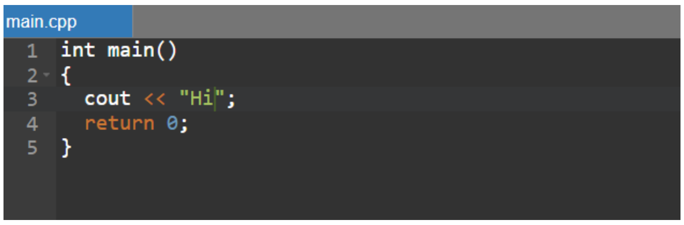
  The function called **main** is special in all C++ programs; this function is called when the program is launched and must appear in any C++ program only once.

- {
  The **braces** on lines 4 and 10 delimit the **statements** that are part of the main function.

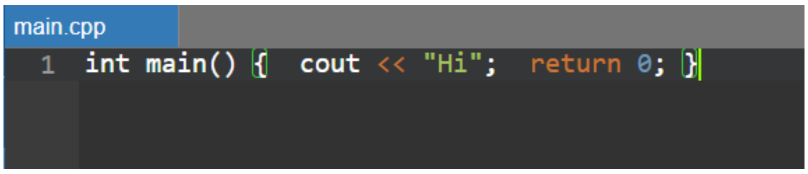
- std :: cout << "Hello world";
  This is a C++ **statement**. A statement is a construct (expression, command) that does something. Instructions are the "core" of programs, they determine their behavior. The instructions in a program are executed in order, one after the other.
  This statement produces the text *"Hello world"* to be displayed on the screen. It consists of three parts. std::cout means the standard output device (**st**andar**d** **c**haracter **out**put) – most often the computer screen. The second part is the insertion operator <<, which indicates that what follows is inserted into std::cout (sent to the screen). The third part is the text, "*Hello world*", enclosed in quotes, which will be inserted into std::cout.
  Let's observe the presence of the character; at the end of the instruction. Any C++ statement must end with **;**, just like any sentence in English ends with the character **.** (point).

- return 0;
  This statement marks **the end** of the execution of the main function and our program. The value 0 means that the program **finished successfully**!
  If there were other statements in our program **afte**r the *"return 0;"* statement, they **would not be executed**.

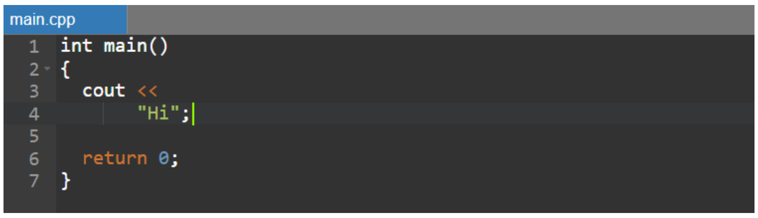
Note that not all program lines produce effects when the program is executed. Some lines (comments) are written only to make the program easier for the reader/writer to understand. Furthermore, it is not mandatory that every statement be written on a single line. The following three examples of the main function have the same effect:


main.cpp

```
int main()
{
  cout << "Hi";
  return 0;
}
```

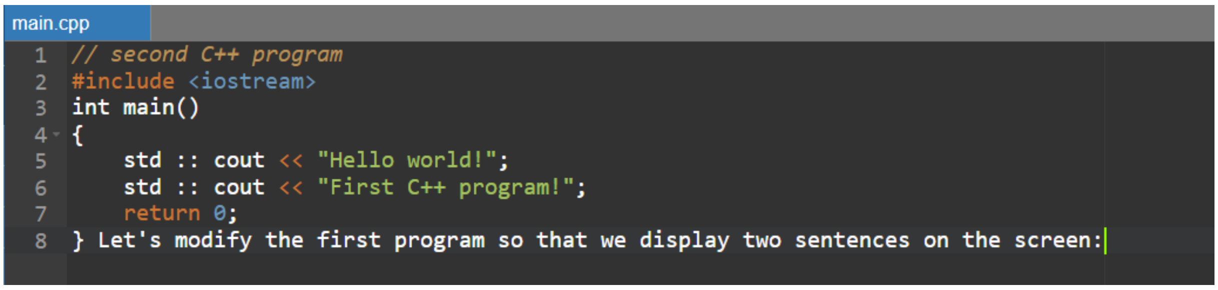

main.cpp

```
int main() {  cout << "Hi";  return 0; }
```



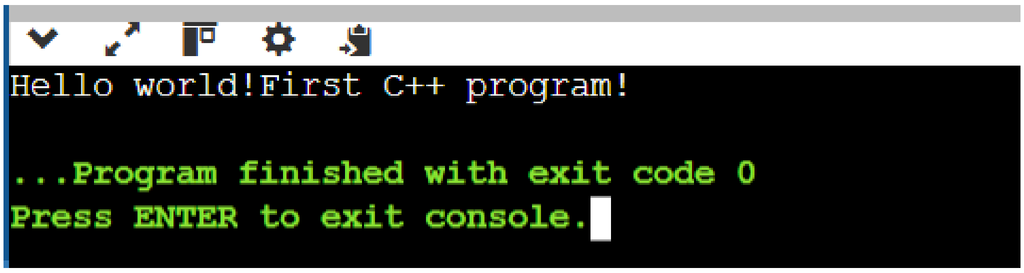
```cpp
int main()
{
  cout <<
        "Hi";

  return 0;
}
```

Let's modify the first program so that we display two sentences on the screen:



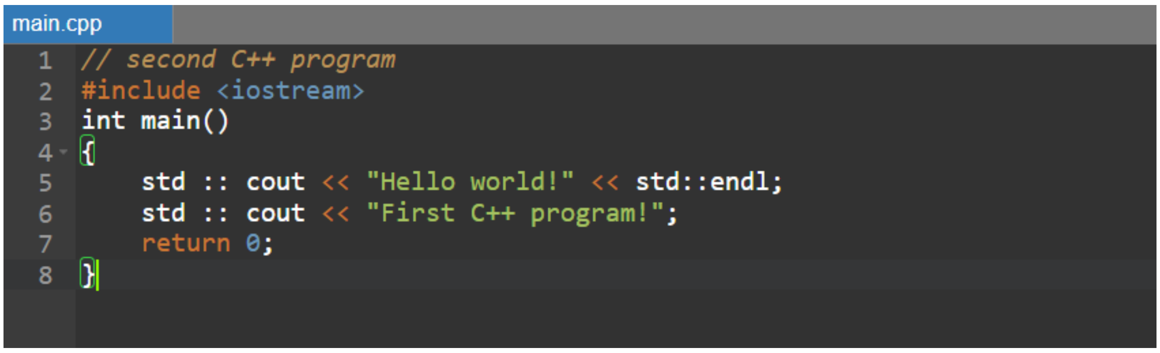
```cpp
// second C++ program
#include <iostream>
int main()
{
    std :: cout << "Hello world!";
    std :: cout << "First C++ program!";
    return 0;
} Let's modify the first program so that we display two sentences on the screen:
```

When running, the program will display:



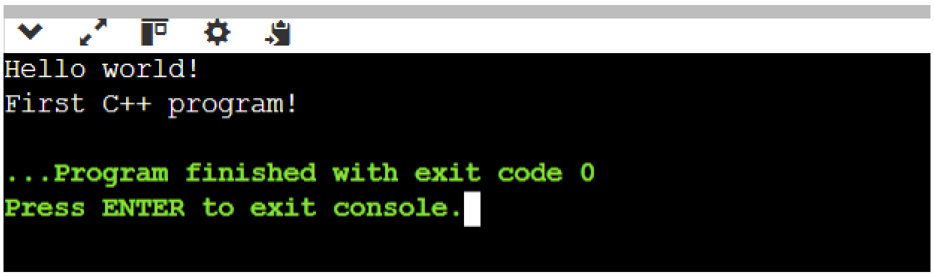
```
Hello world!First C++ program!

...Program finished with exit code 0
Press ENTER to exit console.
```

Notice that the two sentences are written on the same line of the screen, even though they were written with two separate instructions. If we want the two sentences to be displayed on different lines of the screen, we use std::endl.



```cpp
// second C++ program
#include <iostream>
int main()
{
    std :: cout << "Hello world!" << std::endl;
    std :: cout << "First C++ program!";
    return 0;
}
```

Now the program will display:

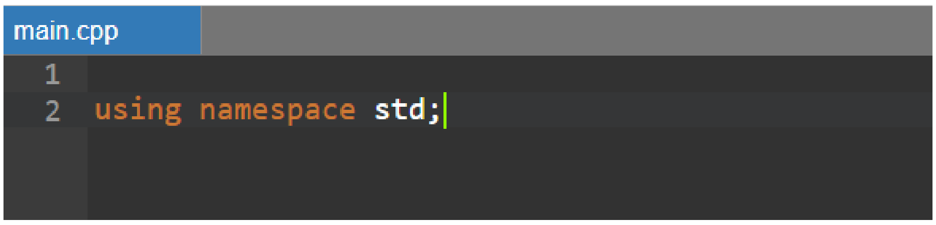


```
Hello world!
First C++ program!

...Program finished with exit code 0
Press ENTER to exit console.
```

The statement using namespace std;

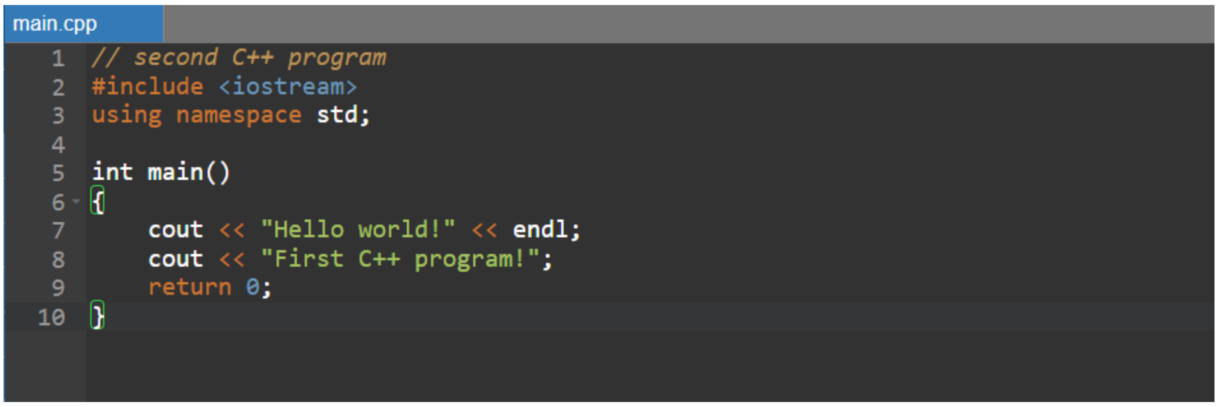
In C++, **identifiers** are grouped into namespaces. There is a predefined namespace named std that contains all identifiers from the standard C++ library.

cout, like endl, is an identifier in the std namespace, and to use it the std::cout expression **must** be used. To refer more simply to the identifiers in the std namespace, the instruction can be used:


main.cpp

```cpp

using namespace std;
```

Thus, the previous program can be rewritten:


main.cpp

```cpp
// second C++ program
#include <iostream>
using namespace std;

int main()
{
    cout << "Hello world!" << endl;
    cout << "First C++ program!";
    return 0;
}
```

Comments

Comments are text that may appear in the source program and are not considered when compiling. They are read only by humans to explain certain more important sections of the program. As we saw above, there are two types of comments in C++:

- // comment on a line
- /* block comment */

A **line comment** starts at the // characters and ends at the end of the line. The **block comment** starts at "/*", ends at "*/", and can span multiple lines.

Pay attention to!

- One of the most common syntax errors is forgetting to write ; at the end of an instruction.
- Comments are important! We need to learn to write code that we can understand a day or a year from now, and the presence of comments is a step forward.