Badji-Mokhtar University Annaba
Faculty of Technology
Department of Electronics
3rd year License in Automatique
Academic Year: 2025-2026

Chapter 1. Introduction to the C++ language

1- Introduction

C++ is a programming language that was developed by Bjarne Stroustrup in 1983, with the aim of enhancing the capabilities of the C language. The name "C++" symbolizes this intention to "increment" C, with "++" being an operator in C that adds 1 to a variable.

One of the main innovations of C++ is the introduction of object-oriented programming (OOP), which allows for more modular and reusable code structuring. Thanks to this approach, developers can create "classes," which are templates for objects, encapsulating both data and associated behaviors.

In addition to classes, C++ introduces advanced concepts such as "templates," which allow for the creation of generic functions and classes that can work with different types of data. This flexibility makes C++ particularly well-suited for the development of libraries and complex applications.

Other notable features of C++ include memory management using pointers, operator overloading, and the ability to integrate low-level functions, which allows developers to work both at an abstract level and close to the hardware. In summary, C++ represents a significant advancement over C, offering programmers powerful tools to develop robust and efficient software.

2- History of C++

The C++ language is one of the most popular programming languages in the world. Particularly prized in the field of video games for its performance and features, C++ has become essential for developers. Designed by Bjarne Stroustrup in 1982 at Bell Laboratories of AT&T, C++ was developed to integrate object-oriented programming concepts while addressing the limitations of the C language. Thus, it combines the features of a classical language like C with those of an object-oriented language, similar to Java. Stroustrup sought to enrich an existing structured language (C) with features that allow for object-oriented programming, thereby facilitating the transition for programmers familiar with C. Since its creation until its standardization, C++ has undergone some evolution. Although it is derived from C, C++ distinguishes itself with its new features, notably object-oriented programming, thus offering a different and powerful approach compared to its predecessor.
The C++ language can be considered an evolution of the C language, enriched with several new features. It incorporates advanced programming concepts, such as object-oriented programming.

Programs form the foundation of computer science, allowing your computer to perform actions.

• To create programs, we use a programming language, and there are hundreds of them.
• C++ is one of the most widely used programming languages in the world.
• C++ is a descendant of C, which it enriches by adding several features.
• C++ is classified as a low-level language because it is closely related to machine language (binary), which can sometimes make it complex to use.
• C++ is classified as a low-level language because it is closely related to machine language (binary), which can sometimes make it complex to use. This language stands out for its exceptional speed, making it the preferred choice for creating many video games that require high performance.

3- The Software Required for Programming

To program in C++, the installation of certain specific software is essential.

3-1- ESSENTIAL TOOLS FOR THE PROGRAMMER

The compiler is a key element because it transforms your C++ code into binary language. Various compilers are available for the C++ language.

- **Visual C++(Windows seulement)**
- **Visual C++ Express**
- **Xcode (Mac OS seulement)**
- **Turbo C++**
- **Code::blocks (Windows, Mac OS, Linux)**

3-2- The Bare Minimum for a Programmer:

- **A text editor** is necessary to write the source code in C++. In theory, software like Notepad in Windows or "vi" in Linux can work. However, the best thing to do is to use a smart editor that automatically colors the code, making it easier to navigate. That's why no sensible programmer turns to Notepad.
- **A compiler** is necessary to convert your source code into binary.
- **A debugger** is useful to help you identify errors in your program, even though, unfortunately, the "corrector" that would automatically fix our errors does not yet exist.

3-2- General Structure of a C++ Program

A C++ program is generally composed of several source files, which fall into two categories:

1. The files containing instructions, with the extension .cpp.
2. The files that contain only declarations, with the extension .h (for "header").

C++ allows for modular programming, which means that a program can be structured into several modules (files).

Context: The instructions must absolutely be encapsulated in functions, and there is a special function called main, which serves as the starting point for every program.

The minimal structure of a C++ program is as follows:

```
#include < iostream>
using namespace std ;
int main()
{
 .........; // body of the program

}
```

An example: The following example shows a simple program that displays 'Hello' on the screen:

```
 /* exemple:
Un simple programme en C++*/
#include < iostream>
using namespace std ;
int main()
{
cout<<"Bonjour!"<<endl; // display
 return 0 ;
}
```

To wait for user action

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

int main() {
    cout << "Bonjour!" << endl; // display
    cin.ignore();
    return 0;
}
```

Or, by removing the line `using namespace std;`

```
#include <iostream>

int main() {
    std::cout << "Bonjour!" << std::endl; // affichage
    std::cin.ignore();
    return 0;
}
```

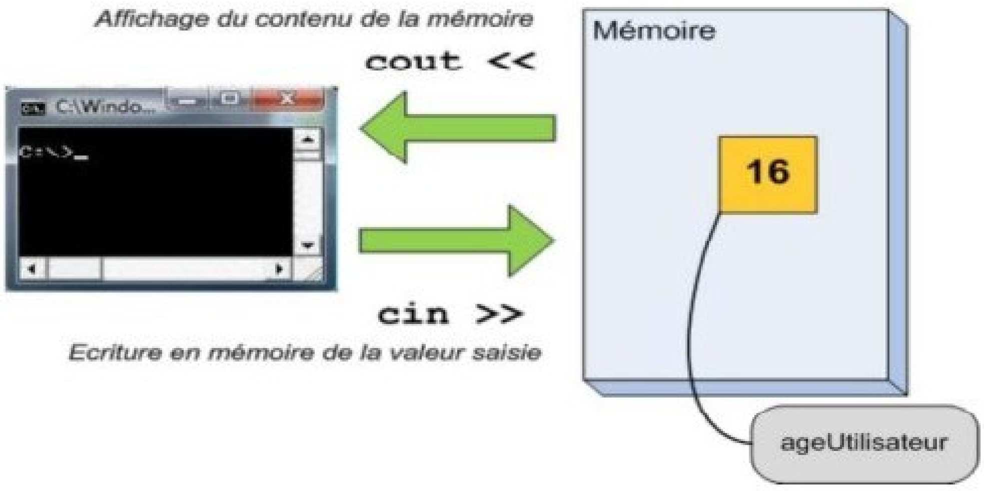
• The **#include** directive

Generally, we place a certain number of instructions starting with #include at the beginning of the program. Context: In general, we place a certain number of instructions starting with #include at the beginning of the program. This directive allows us to include in the program the definition of certain objects, types, or functions. Context: The file name **can be specified either between angle brackets < >, or between quotation marks:**

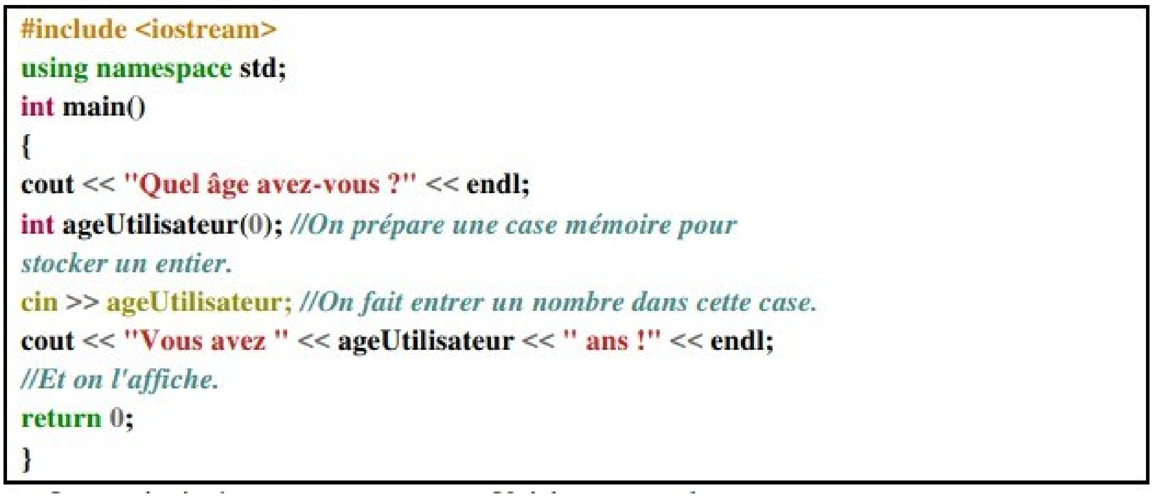
- #include : includes the file filename by first searching for it in the configured paths, then in the same directory as the source file.
  #include "filename": includes the file filename by first searching for it in the same directory as the source file, then in the configured paths.

- **using namespace std;**

This line may seem a bit more complex to understand: it indicates the use of the std namespace. A namespace is a collection of classes that includes cout. Since we want to use the cout object, we specify that we want to use, by default, the std namespace. In summary, it is important to remember that, as soon as we want to use cin or cout, this directive must be included.


Affichage du contenu de la mémoire
cout <<
C:\Windo...
C:\>_
cin >>
Ecriture en mémoire de la valeur saisie
Mémoire
16
ageUtilisateur

Example : Code : C++ First example of cin and cout



```
#include <iostream>
using namespace std;
int main()
{
cout << "Quel âge avez-vous ?" << endl;
int ageUtilisateur(0); //On prépare une case mémoire pour
stocker un entier.
cin >> ageUtilisateur; //On fait entrer un nombre dans cette case.
cout << "Vous avez " << ageUtilisateur << " ans !" << endl;
//Et on l'affiche.
return 0;
}
```

Voici ce que ça donne :

Code : Console - Quel âge avez-vous ?

```
Quel âge avez-vous ?
22
Vous avez 22 ans !
```

It is important to note that standard header files are no longer named with the .h extension (like iostream.h). If these files are included without the using namespace std; statement, the program may not work correctly. In some versions of C++, if you include a standard header file with the .h extension (e.g., iostream.h) during compilation, the compiler will use a backward-compatible version and issue a warning.

- **The file iostream**

  The iostream file contains several object definitions used for managing the program's inputs and outputs, such as displaying text on the screen or writing to files. For example, the cout object, which allows you to display text, is defined in this file. Context: For example, the cout object, which allows you to display text, is defined in this file. To use cout in a program, it is necessary to include the following directive at the beginning of the code:
  #include <iostream>
  This file is provided by the compiler and is part of the standards of the C++ language.

- **The function** `main()`

Our program contains a function called main, which marks the starting point of the program's execution. Launching a C++ program is equivalent to executing the main function. Context: Launching a C++ program is equivalent to executing the main function. Thus, every C++ program must imperatively include a main function.
Context: Thus, every C++ program must imperatively include a main function.
The main function takes the following form:

```
int main()
{
on place ici une liste d'instructions en C++
}
```

The instructions located between the curly braces are executed sequentially: each instruction is executed one after the other, in the order they appear.

- **cout**
This is the output stream of the program, called "Console Output" (output to the console). By default, this stream is directed to the screen. It allows you to display messages on the screen using the << operator. This operator looks like an arrow, symbolizing the direction of data transfer to the console.

Example : cout<<"BONJOUR";

This instruction displays BONJOUR on the screen.

-Another example:
cout<<endl;
When you send endl (End of Line) to the display, you move to the next line.
You also need to know how to write more condensed statements. Instead of writing them in three instructions:

```
cout << "BONJOUR";
cout << endl;
cout << "AU REVOIR";
```

We can write in a single instruction:

```
cout << "BONJOUR" << endl << "AU REVOIR";
```

However, on some implementations, this shorthand instruction may not compile correctly, because the implementation of the endl symbol does not allow the << operator to be used after it.

```
cout << "BONJOUR" << endl;
cout << "AU REVOIR";
```

• **Return of the function**
The last instruction of our program, return 0;, simply indicates that the main function has completed successfully, without any errors.

• **Program execution**
When we edit our source file, then compile and run the program, the message "BONJOUR" appears on the screen.

Note:

• The addition of the system instruction (system("PAUSE");) may be necessary to prevent the program from closing immediately after it opens. Context: • The addition of the system instruction (system("PAUSE");) may be necessary to prevent the program from closing immediately after opening. This instruction must be placed before return 0;.
•
Context: This instruction must be placed before return 0;.
• It is important to note that: It is important to note that:

Unlike the Pascal language, C++ distinguishes between lowercase and uppercase letters; for example, x and X represent two different variables.
• Instructions generally end with a semicolon, which marks the end of an instruction.
• Spaces (including tabs and line breaks) can be used freely before or after a separator (e.g., #, <, >, (,), {, }, ;, etc.). This means that C++ is a free-format language.