جامعة المستقبل
AL MUSTAQBAL UNIVERSITY

كلية العلوم
قسم الانظمة الطبية الذكية

Lecture: (3)

C++ PROGRAMMING LANGUAGE

Subject: Computer Programming (I)
Level: First
Lecturer: Dr. Maytham N. Meqdad

Lecture 3

1	C++ programming language
2	Structure of C++ program
3	Reserved words and Header files
4	Character set and Identifiers
5	Variable and Constant , Data type (int , float , char , void)
6	Cin,cout

1. C++ programming language:

What is C++?

C++ is a cross-platform language that can be used to create high-performance applications.

C++ was developed by Bjarne Stroustrup, as an extension to the C language.

C++ gives programmers a high level of control over system resources and memory.

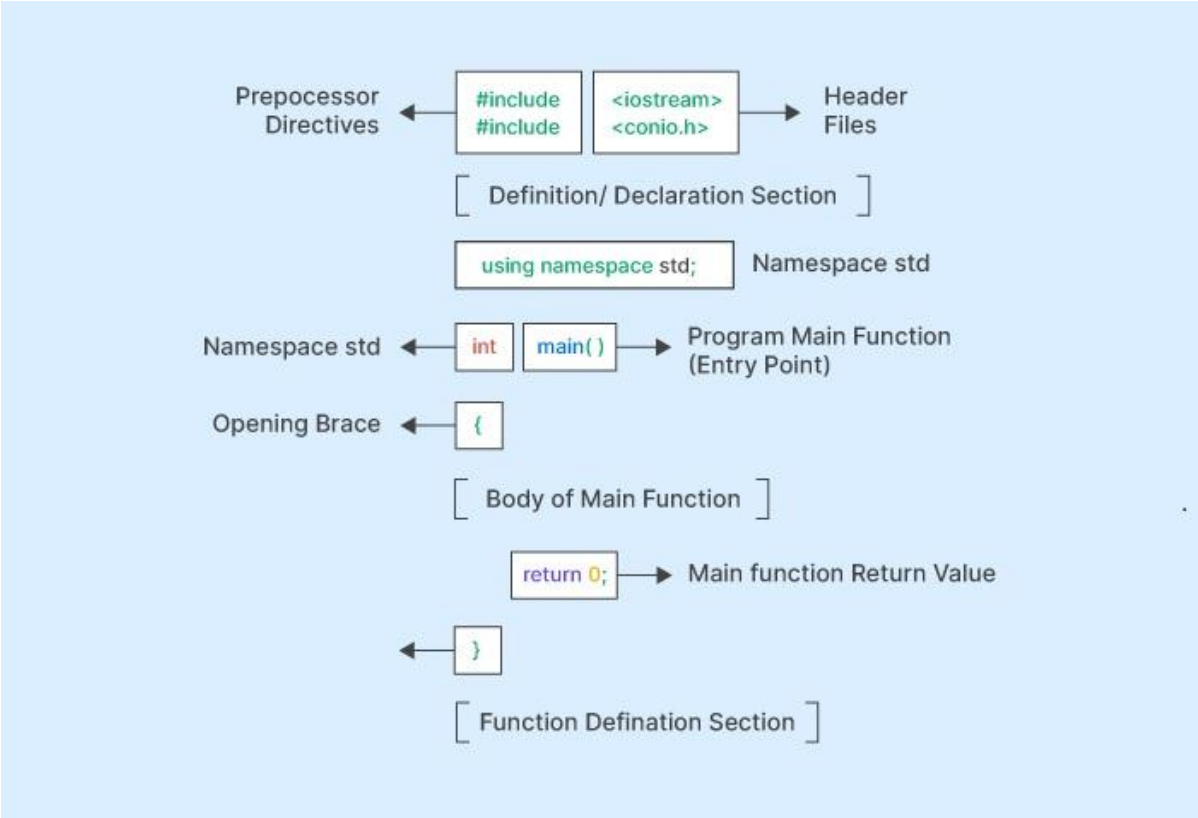
Why Use C++

C++ is one of the world's most popular programming languages. C++ can be found in today's operating systems, Graphical User Interfaces, and embedded systems. C++ is an object-oriented programming language which gives a clear structure to programs and allows code to be reused, lowering development costs. C++ is portable and can be used to develop applications that can be adapted to multiple platforms. C++ is fun and easy to learn! .As C++ is close to C, C# and Java, it makes it easy for programmers to switch to C++ or vice versa.

==========

2. Structure of C++ program

The structure of a C++ program is made up of multiple source code files that cater to different components such as main function, member functions, class definition, headers/ standard headers, comments, variables, data types, namespaces, input/ output statements, etc.



A **C++ program** is structured in a specific and particular manner. In C++, a program is divided into the following three sections:

1. Standard Libraries Section
2. Main Function Section
3. Function Body Section

For example, let's look at the implementation of the `Hello World` program:

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

int main() {
  cout << "Hello World!" << endl;
  return 0;
}
```

1. Standard libraries section

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

- `#include` is a specific preprocessor command that effectively copies and pastes the entire text of the file, specified between the angle brackets, into the source code.
- The file `<iostream>`, which is a standard file that should come with the C++ compiler, is short for **input-output streams**. This command contains code for displaying and getting an input from the user.
- `namespace` is a prefix that is applied to all the names in a certain set. `iostream` file defines two *names* used in this program - **cout** and **endl**.
- This code is saying: Use the cout and endl tools from the std toolbox.

2. Main function section

```cpp
int main() {}
```

- The starting point of all C++ programs is the `main` function.
- This function is called by the operating system when your program is executed by the computer.
- `{` signifies the start of a block of code, and `}` signifies the end.

3. Function body section

```
cout << "Hello World" << endl;
return 0;
```

- The name `cout` is short for **character output** and displays whatever is between the << brackets.
- Symbols such as << can also behave like functions and are used with the keyword `cout`.
- The `return` keyword tells the program to return a value to the function `int main`
- After the return statement, execution control returns to the operating system component that launched this program.
- Execution of the code terminates here.

3. Reserved words and Header files

Header files: there are libraries contain groups of instructions. For first program shown above, #include <iostram> is most important library for each program written in C++ language. Other libraries we will use if need it like string library and file libraries. Using namespace std; is a phrase must be written under the libraries. Usefulness of this phrase to translate any instruction in the program and return it to the library.

Beginning of program must be done by write int main ( ) function. This function tells the C++ language that the main program will begin from this point. The statements of program will written between the two brackets { }. Ending of program will puts before end bracket}. The ending statement represented by return 0; statement. Most important rules, for programming in C++ language, must be studied. These rules is:

1. Writing in C++ language is contain characters, numbers, and symbols only.
2. Identifiers that is a sequence of one or more letters, digits or underscore characters (_). Neither spaces nor punctuation marks or symbols can be part of an identifier.
3. Any identifier must not begin with numbers.
4. They cannot match any keyword of the C++ language nor your compiler's specific ones, which are reserved keywords. The standard reserved keywords are:

struct	switch	template	this	thread_local	throw
TRUE	try	typedef	typeid	typename	union
unsigned	using	virtual	void	volatile	wchar_t
while	–	–	–	–	–

alignas	alignof	asm	auto	bool	break
case	catch	char	char16_t	char32_t	class
const	constexpr	const_cast	continue	decltype	default
delete	double	do	dynamic_cast	else	enum
explicit	export	extern	FALSE	float	for
friend	goto	if	inline	int	long
mutable	namespace	new	noexcept	nullptr	operator
private	protected	public	register	reinterpret_cast	return
short	signed	sizeof	static	static_assert	static_cast

- Additionally, alternative representations for some operators cannot be used as identifiers since they are reserved words under some circumstances:

and, and_eq, bitand, bitor, compl, not, not_eq, or, or_eq, xor, xor_eq

5.Comments: are phrases that do nothing. Their purpose is only to allow the programmer to insert notes or descriptions embedded within the source code. C++ . supports two ways to insert comments:

Ex:

//line comment

/block comment/

The first of them, known as line comment, discards everything from where the pair of slash signs (//) is found up to the end of that same line. The second one, known as block comment, discards everything between the /* characters and the first appearance of the */ characters, with the possibility of including more than one line.

4. Character set and Identifiers

The C++ character set consists of 3 main elements. They are:

1. **Letters:** These are alphabets ranging from A-Z and a-z (both uppercase and lowercase characters convey different meanings)

Uppercase:	A, B, C, . . ., Z
Lowercase:	a, b, c, . . ., z
Digits:	0, 1, 2, . . .,9

2. **Digits:** All the digits from 0 – 9 are valid in C++.
3. **Special symbols:** There are a variety of special symbols available in C++ like mathematical, logical and relational operators like **+,-, *, /, \, ^, %, !, @, #, ^, &, (,), [,], ;** and many more.

+	-	*	/	^
(	[	{	}	]
)	<	=	>	, (Comma)
" (Double Conations)	. (Dot)	: (Colon)	; (Semicolon)	␣ (Blank Space)

C++ Identifiers

All C++ **variables** must be **identified** with **unique names**. These unique names are called **identifiers**. Identifiers can be short names (like x and y) or more descriptive names (age, sum, totalVolume).

Note: It is recommended to use descriptive names in order to create understandable and maintainable code:

```
// Good
int minutesPerHour = 60;

// OK, but not so easy to understand what m actually is
int m = 60;
```

The general rules for naming variables are:

- Names can contain letters, digits and underscores
- Names must begin with a letter or an underscore (_)
- Names are case sensitive (`myVar` and `myvar` are different variables)
- Names cannot contain whitespaces or special characters like !, #, %, etc.
- Reserved words (like C++ keywords, such as `int`) cannot be used as names

5. Variable and Constant

When programming, we store the variables in our computer's memory, but the computer has to know what kind of data we want to store in them. Since it is not going to occupy the same amount of memory to store a simple number than to store a single letter or a large number. They are not going to be interpreted the same way. You have a summary of the basic fundamental data types in C++, as well as the range of values that can be represented with each one:

Name	Description	Size	Range
char	Character or small integer.	1byte	signed: -128 to 127 unsigned: 0 to 255
short int (short)	Short Integer.	2bytes	signed: -32768 to 32767 unsigned: 0 to 65535
int	Integer.	4bytes	signed: -2147483648 to 2147483647 unsigned: 0 to 4294967295
long int (long)	Long integer.	4bytes	signed: -2147483648 to 2147483647 unsigned: 0 to 4294967295
bool	Boolean. It can take one of two values: true or false.	1byte	true or false
float	Floating point number.	4bytes	+/- 3.4e +/- 38 (~7 digits)
double	Double precision floating point number.	8bytes	+/- 1.7e +/- 308 (~15 digits)
Long double	Long double precision floating point number.	8bytes	+/- 1.7e +/- 308 (~15 digits)
Wchar	Wide character.	4bytes	1 wide character
string	String definition	8bytes	Group of characters

The values of the columns Size and Range depend on the system the program is compiled for. The values shown above are those found on most 32-bit systems. But for other systems, the general specification is that int has the natural size suggested by the system architecture (one "word") and the four integer types char, short, int and long must each one be at least as large as the one preceding it, with char being always 1 byte in size. The same applies to the floating point types float, double and long double, where each one must provide at least as much precision as the preceding one.

Declaration of variables

In order to use a variable in C++, we must first declare it specifying which data type we want it to be. The syntax to declare a new variable is to write the specifier of the desired data type (like int, bool, float...) followed by a valid variable identifier.

Ex-:

```
int a;
float mynumber;
```

These are two valid declarations of variables. The first one declares a variable of type int with the identifier a. The second one declares a variable of type float with the identifier mynumber. Once declared, the variables a and mynumber can be used within the rest of their scope in the program. If you are going to declare more than one variable of the same type, you can declare all of them in a single statement by separating their identifiers with commas.

Ex-:

```
int a, b, c;
```

This declares three variables (a, b and c), all of them of type int, and has exactly

the same meaning as:

```
int a;
int b;
int c;
```

The integer data types char, short, long and int can be either signed or unsigned depending on the range of numbers needed to be represented. Signed types can represent both positive and

negative values, whereas unsigned types can only represent positive values (and zero). This can be specified by using either the specifier signed or the specifier unsigned before the type name.

Ex-:

```
Int main () {
Unsigned int NumberOfSisters;
Signed int MyAccountBalance;
 }
```

By default, if we do not specify either signed or unsigned most compiler settings will assume the type to be signed, therefore instead of the second declaration above we could have written:

```
Int main () {
int MyAccountBalance;
}
```

-Variables can be divided to(.Integer Numbers, Floating-Point Numbrs,Characters,Boolean Values,Strings)

1. Integer Numerals

75//Decimal

0113// octal

0x4b // hexadecimal

5//int

75u // unsigned int

75l // long int

75ul // unsigned long int

---

2. Floating Point Numbers

3.14159L // long double

6.02e23f // float

3. Character and string literals

'z'

'p'

"Hello world"

"How do you do?"

\n	newline
\t	Tab
\v	Vertical Tab
\b	Back space
\a	Alert (beep)
\'	Single quote
\"	Double quote
\?	Question mark
\\	Back slash
\	Octal number
\x	Hexadecimal number

4. Boolean literals: There are only two valid Boolean values: **true** and **false**. These can be expressed in C++ as values of type bool by using the Boolean literals true and false.

5. strings: Variables that can store non-numerical values that are longer than one single character are known as strings.

```
// my first string
#include <iostream>
#include <string>
using namespace std;
int main ()
{
string mystring = "This is a string";
cout << mystring;
        return 0;
}
```

```
// my first string
#include <iostream>
#include <string>
using namespace std;
int main ()
{
string mystring;
mystring = "This is the initial string content";
 cout << mystring << endl;
mystring = "This is a different string content";
 cout << mystring << endl;
return 0;
}
```

```
o/p
This is the initial string content
This is a different string content
```

6. Inputs and outputs in C++ (cin , cout)

In C+ +, there are two instructions for input the variables and output the results.These functions are: **cin and cout**

for input variables, we will use cin instruction. The instruction will write as:

cin >>---;

* After the two-comparator mark (>>) we will inter the variables.

Ex:- in this example, we will input variable (a) as integer number.

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

```
int a;
cin >> a ;
return 0;
}
```

-The above example remain uncompleted, we need a function for output that write variables, constants, characters, and strings on the output screen. We will use cout function as:

cout<< -----;

```
#include <iostream>
using namespace std;
int main ()
{
int a;
cin >> a ;
cout << "a="<<a;
return 0;
}
```

```
o/p
a = 25
```

-Any phrase (letters or numbers) putted between two quote (" ") in cout function will print as written.

==========

```
// defined constants: calculate circumference

#include <iostream>
using namespace std;

#define PI 3.14159
#define NEWLINE '\n'

int main ()
{
  double r=5.0;               //    radius
  double circle;

  circle = 2 * PI * r;
  cout << circle;
  cout << NEWLINE;

  return 0;
}
```

Q1\Write a program to read three letters and print them in reverse order.

```
#include <iostream>

int main() {
    char letter1, letter2, letter3;

    // Read three letters
    std::cout << "Enter the first letter: ";
    std::cin >> letter1;

    std::cout << "Enter the second letter: ";
    std::cin >> letter2;

    std::cout << "Enter the third letter: ";
    std::cin >> letter3;

    // Print in reverse order
    std::cout << "Letters in reverse order: " <<
letter3 << letter2 << letter1 << std::endl;

    return 0;
}
```

==========

Q2\ Write a program to reads three floating-point values, then compute and print their arithmetic sum, products and their average

```
#include <iostream>

int main() {
    float num1, num2, num3;

    // Read three floating-point values
    std::cout << "Enter the first floating-point number: ";
    std::cin >> num1;

    std::cout << "Enter the second floating-point number: ";
    std::cin >> num2;

    std::cout << "Enter the third floating-point number: ";
    std::cin >> num3;

    // Compute and print arithmetic sum
```

```
    float sum = num1 + num2 + num3;
    std::cout << "Arithmetic Sum: " << sum << std::endl;

    // Compute and print product
    float product = num1 * num2 * num3;
    std::cout << "Product: " << product << std::endl;

    // Compute and print average
    float average = sum / 3.0;
    std::cout << "Average: " << average << std::endl;

    return 0;
}
```

Q3\Write a program that read your age in years and prints it in days.

```
#include <iostream>

int main() {
    int ageInYears;

    // Read age in years
    std::cout << "Enter your age in years: ";
    std::cin >> ageInYears;

    // Calculate age in days (assuming 365 days in a year)
    int ageInDays = ageInYears * 365;

    // Print age in days
    std::cout << "Your age in days is approximately: " << ageInDays << "
days" << std::endl;

    return 0;
}
```

Q4\ Write a program to read a time in (hour: minute: second) and prints it in second.

```
#include <iostream>

int main() {
    int hours, minutes, seconds;

    // Read time in (hour:minute:second) format
    std::cout << "Enter the time (hour minute second): ";
    std::cin >> hours >> minutes >> seconds;

    // Convert time to seconds
```

```
    int totalTimeInSeconds = hours * 3600 + minutes * 60 + seconds;

    // Print time in seconds
    std::cout << "The time in seconds is: " << totalTimeInSeconds << "
seconds" << std::endl;

    return 0;
}
```

Q5\Write a program to read unsigned decimal number then print it in all three notation (decimal, octal and hexadecimal).

```
#include <iostream>
#include <iomanip>

int main() {
    unsigned int decimalNumber;

    // Read unsigned decimal number
    std::cout << "Enter an unsigned decimal number: ";
    std::cin >> decimalNumber;

    // Print in decimal notation
    std::cout << "Decimal Notation: " << decimalNumber << std::endl;

    // Print in octal notation
    std::cout << "Octal Notation: " << std::oct << decimalNumber <<
std::endl;

    // Reset stream flags to default
    std::cout << std::dec;

    // Print in hexadecimal notation
    std::cout << "Hexadecimal Notation: " << std::hex << std::uppercase <<
decimalNumber << std::endl;

    return 0;
}
```

Q6\ Write a program that reads the radius of a circle (as double value), and computes and prints the area, diameter and circumference.

```
#include <iostream>
#include <cmath>

int main() {
    double radius;

    // Read the radius of the circle
    std::cout << "Enter the radius of the circle: ";
```

```
    std::cin >> radius;

    // Calculate area, diameter, and circumference
    double area = M_PI * radius * radius;
    double diameter = 2 * radius;
    double circumference = 2 * M_PI * radius;

    // Print the results
    std::cout << "Area of the circle: " << area << std::endl;
    std::cout << "Diameter of the circle: " << diameter << std::endl;
    std::cout << "Circumference of the circle: " << circumference <<
std::endl;

    return 0;
}
```