Introduction to C++ Programming

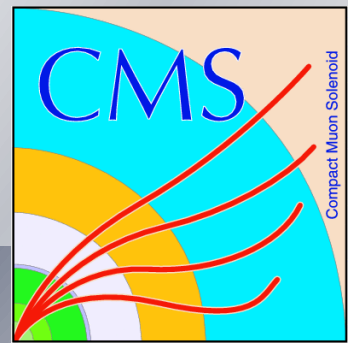
Presented by

DR. MOHAMMED ATTIA MAHMOUD

-PhD, Fayoum University, Egypt and Antwerp University, Belgium.

-Researcher in ENHEP, ASRT, Fayoum Uni, and BUE.

-FSQ Gen-Contact, CMS experiment, CERN, Geneva, Switzerland.



Outline

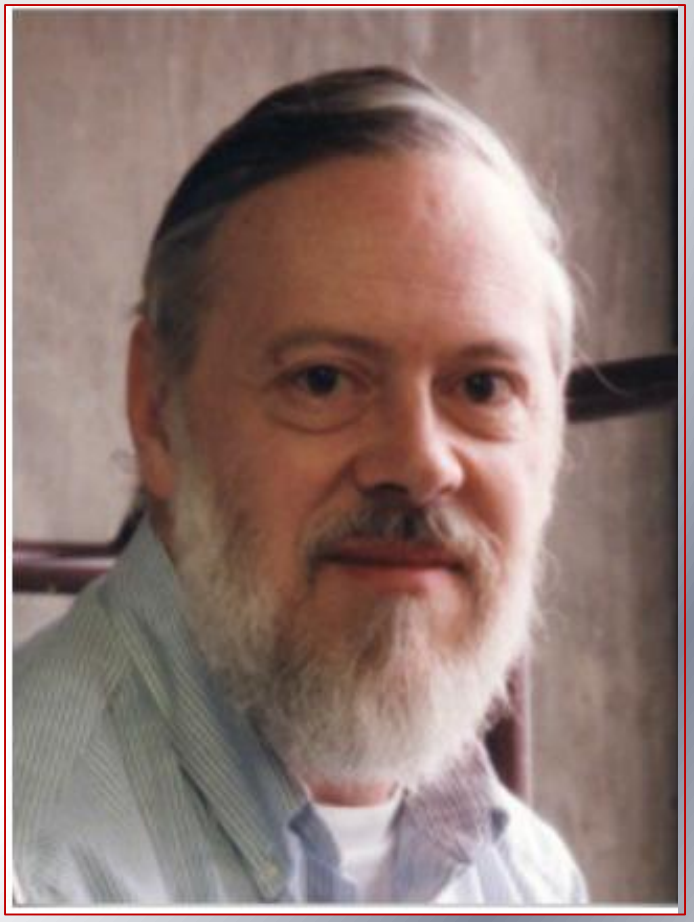
- The Task of Programming
- History of C and C++
- Basics of a Typical C++ Environment
- A Simple Program: Printing a Line of Text
- Welcome to C++!
- Another Simple Program: Adding Two Integers

The Task of Programming

- Programming a computer involves writing instructions that enable a computer to carry out a single task or a group of tasks.
- A computer programming language requires learning both vocabulary and syntax.
- Programmers use many different programming languages, including BASIC, Pascal, COBOL, RPG, python, JAVA and C++.
- The rules of any language make up its syntax.
- Machine language is the language that computers can understand; it consists of 1s and 0s

History of C and C++

- During 1970 Dennis Ritchie created C Programming language to develop the UNIX operating system at Bell Labs.
- C is a general-purpose, high-level language.
- C was originally first implemented on the PDP-11 computer in 1972



History of C and C++

- C++ Development started in 1979.
- During the creation of Ph.D. thesis, Bjarne Stroustrup worked with language called Simula.
- Simula is programming language basically useful for the simulation work.

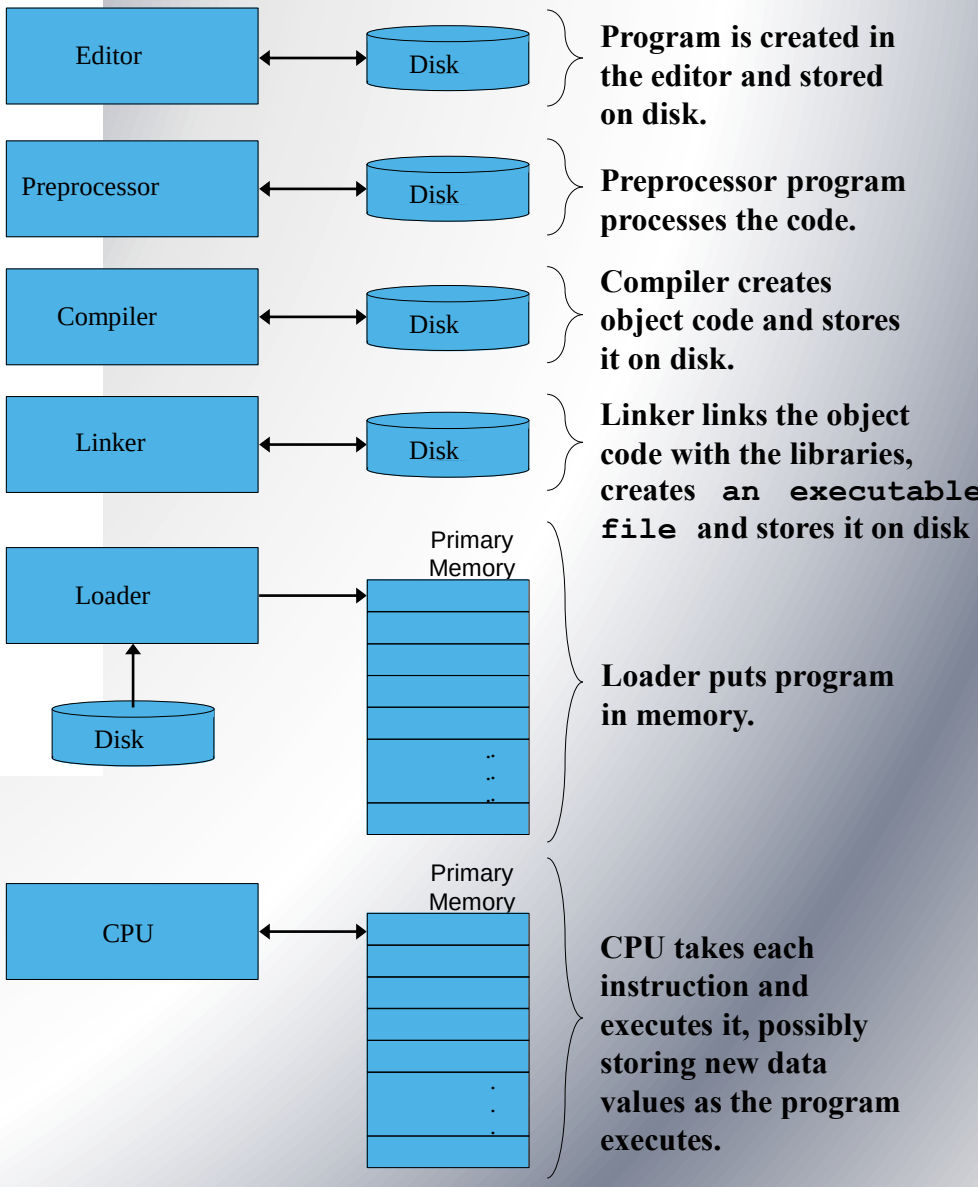
Basics of a Typical C++ Environment

- **C++ systems**
  - Program-development environment
  - Language
  - C++ Standard Library
- **C++ program names extensions**
  - .cpp
  - .cxx
  - .cc
  - .C

Basics of a Typical C++ Environment

Phases of C++ Programs:

1. Edit
2. Preprocess
3. Compile
4. Link
5. Load
6. Execute



Basics of a Typical C++ Environment

- Common Input/output functions
  - `cin`
    - Standard input stream
    - Normally keyboard
  - `cout`
    - Standard output stream
    - Normally computer screen
  - `cerr`
    - Standard error stream
    - Display error messages

A Simple Program: Printing a Line of Text

- Before writing the programs
  - Comments
    - Document programs
    - Improve program readability
    - Ignored by compiler
    - Single-line comment
      - Use C's comment /* .. */ OR Begin with **`// or`**
  - Preprocessor directives
    - Processed by preprocessor before compiling
    - Begin with **#**

Welcome to C++!

```
1  // Fig. 1.2: fig01_02.cpp
2  // A first program in C++.
3  #include <io
4
5  // function m
6  int main()
7  {
8     std::cout << "Welcome to
9
10    return 0;
11
12 } // end function m
```

Single-line comments.

Function **main** returns an integer value.

processor directive to include header file

pears exactly

Left brace { begins function body.

Statements end with a semicolon ;.

Corresponding right brace } ends function body.

successfully

Name

Stream insertion operator.

namespace **std**.

Keyword **return** is one of several means to exit function; value **0** indicates program terminated successfully.

A Simple Program: Printing a Line of Text

- **Standard output stream object**
  - `std::cout`
  - "Connected" to screen
  - `<<`
    - Stream insertion operator
    - Value to right (right operand) inserted into output stream
- **Namespace**
  - `std::` specifies using name that belongs to "namespace" `std`
  - `std::` removed through use of `using` statements
- **Escape characters**
  - \
  - Indicates "special" character output

A Simple Program: Printing a Line of Text

Escape Sequence	Description
`\n`	Newline. Position the screen cursor to the beginning of the next line.
`\t`	Horizontal tab. Move the screen cursor to the next tab stop.
`\r`	Carriage return. Position the screen cursor to the beginning of the current line; do not advance to the next line.
`\a`	Alert. Sound the system bell.
`\\`	Backslash. Used to print a backslash character.
`\"`	Double quote. Used to print a double quote character.

Another Simple Program: Adding Two Integers

- **Variables**
  - Location in memory where value can be stored
  - Common data types
    - `int` - integer numbers
    - `char` - characters
    - `double` – floating point numbers
    - float -
  - Declare variables with name and data type before use

```
int integer1;
int integer2;
int sum;
```

  - Can declare several variables of same type in one declaration
    - Comma-separated list

```
int integer1, integer2, sum;
```

Another Simple Program: Adding Two Integers

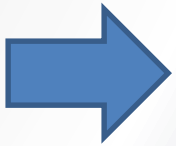
- Input stream object
  - **>>** (stream extraction operator)
    - Used with `std::cin`
    - Waits for user to input value, then press *Enter* (Return) key
    - Stores value in variable to right of operator
      - Converts value to variable data type
- **=** (assignment operator)
  - Assigns value to variable
  - Binary operator (two operands)
  - Example:

```
sum = variable1 + variable2;
```

Here is the complete list of fundamental types in C++:

Group	Type names*	Notes on size / precision
Character types	**char**	Exactly one byte in size. At least 8 bits.
	char16_t	Not smaller than `char`. At least 16 bits.
	char32_t	Not smaller than `char16_t`. At least 32 bits.
	wchar_t	Can represent the largest supported character set.
Integer types (signed)	**signed char**	Same size as `char`. At least 8 bits.
	signed **short** *int*	Not smaller than `char`. At least 16 bits.
	signed **int**	Not smaller than `short`. At least 16 bits.
	signed **long** *int*	Not smaller than `int`. At least 32 bits.
	signed **long long** *int*	Not smaller than `long`. At least 64 bits.
Integer types (unsigned)	**unsigned char**	(same size as their signed counterparts)
	unsigned short *int*	
	unsigned *int*	
	unsigned long *int*	
	unsigned long long *int*	
Floating-point types	**float**	
	double	Precision not less than `float`
	long double	Precision not less than `double`
Boolean type	**bool**	
Void type	**void**	no storage
Null pointer	**decltype(nullptr)**	

```
1  // operating with variables
2
3  #include <iostream>
4  using namespace std;
5
6  int main ()
7  {
8    // declaring variables:
9    int a, b;
10   int result;
11
12   // process:
13   a = 5;
14   b = 2;
15   a = a + 1;
16   result = a - b;
17
18   // print out the result:
19   cout << result;
20
21   // terminate the program:
22   return 0;
23 }
```




Thanks!