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
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.
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
Phases of C++ Programs:

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

- Program is created in the editor and stored on disk.
- Preprocessor program processes the code.
- Compiler creates object code and stores it on disk.
- Linker links the object code with the libraries, creates an executable file and stores it on disk.
- Loader puts program in memory.
- CPU takes each instruction and executes it, possibly storing new data values as the program executes.
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
• 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++!

- // Fig. 1.2: fig01_02.cpp
- // A first program in C++.
- #include <iostream>
- // function main begins program execution
- int main()
- {
-     std::cout << "Welcome to C++!
"
-     return 0;
- } // end function main

**Single-line comments.**

**Preprocessor directive to include input/output stream header file <iostream>.**

**Function main** returns an integer value.

**Left brace { begins function body.**

**Statements end with a semicolon ;.**

**Corresponding right brace } ends function body.**

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

**Name of Stream insertion operator.**

**namespace std.**

**Left brace { begins function body.**
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
**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++:

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

#include <iostream>
using namespace std;

int main ()
{
    // declaring variables:
    int a, b;
    int result;

    // process:
    a = 5;
    b = 2;
    a = a + 1;
    result = a - b;

    // print out the result:
    cout << result;

    // terminate the program:
    return 0;
}
Thanks!