CHAPTER 1.2
INTRODUCTION TO C++ PROGRAMMING

Dr. Shady Yehia Elmashad

Outline

1. Introduction to C++ Programming

- C++ language
  - Facilitates a structured and disciplined approach to computer program design

- Following are several examples
  - The examples illustrate many important features of C++
  - Each example is analyzed one statement at a time.

```
// Fig. 1.2: fig01_02.cpp
// A first program in C++
#include <iostream>

int main()
{
   cout << "Welcome to C++!

   return 0;      // indica
}
```

```
Welcome to C++!
```

Outline

ments

ream>

main

come
n"

n 0)

utput

Comments
Written between `/*` and `*/` or following a `//`.
Improve program readability and do not cause the computer to perform any action.

preprocessor directive
Message to the C++ preprocessor.
Lines beginning with `#` are preprocessor directives.
`#include <iostream>` tells the preprocessor to include the contents of the file `<iostream>`, which

C++ programs contain one or more functions, one of which must be `main`
Parenthesis are used to indicate a function
`int` means that `main` "returns" an integer value.

Prints the *string* of characters contained between the

ncluding `std::cout`, the `<<`
ing `"Welcome to C++!\n"` and
), is called a *statement*.

All statements must end with a semicolon.

very function

`return` is a way to exit a function from a function.
`return 0`, in this case, means that the program terminated normally.

2. Comment

- Message to everyone who reads source program and is used to document source code.
- Makes the program more readable and eye catching.
- Non executable statement in the C++.
- Always neglected by compiler.
- Can be written anywhere and any number of times.
- Use as many comments as possible in C++ program.

2. Comment

Types of comment

1. Single Line Comment

•starts with “**//**” symbol.

•Remaining line after “**//**” symbol is ignored by browser.

•End of Line is considered as End of the comment.

2. Multiple Line Comment (Block Comment)

•starts with “**/***” symbol.

•ends with “***/**” symbol.

2. Comment

Example

```
/* this program calculate the sum of
   two numbers  */
#include<iostream>        // header file
using namespace std;
int main( )               // الدالة الرئيسية
{
     int    x, y , sum ;          // declaration part
/* read  the two numbers */
     cin >> x >> y ;
// calculate the sum
     sum = x + y ;
// print the result
     cout << sum ;
return 0;
}
```

contoso

3. Variables and Constants

Variables

- Variables are memory location in computer's memory to store data.

- Each variable should be given a unique name called identifier, to indicate the memory location in addition to a data type.

- Variable names are just the symbolic representation of a memory location.

- Variable value can be changed during program execution

3. Variables and Constants

Variables Declaration

variable_type variable_name;

Example: int a; - Declares a variable named **`a`** of type **`int`**

int a, b, c; - Declares three variables, each of type **`int`**

int a; float b;

3. Variables and Constants

Constants

- Constant is the term that has a unique value and can't be changed during the program execution.

- Declaration:

1.
```
#define    constant_name    constant_value
```

Example:
```
#define   PI    3.14
```

2.
```
const    constant_type    constant_name = constant_value ;
```

Example:
```
const    float    PI  = 3.14;
```

3. Variables and Constants

Variables and Constants Names

- Can be composed of letters (both uppercase and lowercase letters), digits and underscore '_' only.
- Must begin with a letter or underscore ‘_’.
- Don’t contain space or special character:

(#, *, ?, -, @, !, $, %,&, space,……)
- Can’t be one of the reserved words (they are used by the compiler so they are not available for re-definition or overloading.)

3. Variables and Constants

Reserved Words Examples

int	float	double	char
string	short	long	signed
for	while	if	switch
break	default	do	else
case	return	sizeof	static
continue	goto	true	false
const	void	private	struct
class	cin	cout	new

3. Variables and Constants

Reserved Words Examples

- Which of the following variable names are valid/not valid and why if not?

Name	Valid or not	Name	Valid or not
area		10rate	
shoubra_faculty		Shoubra faculty	
w234		W#d	
Ahmed		1233	
A3		Cin	
A_3		Shoubra-faculty	
temp		int	

```cpp
// Fig. 4.7: fig04_07.cpp
// A const object must be initialized

int main()
{
   const int x;   // Error: x must be

   x = 7;         /

   return 0;
}
```

Notice that **const** variables must be initialized because they cannot be modified later.

Outline

1. Initialize const variable

2. Attempt to modify variable

Program Output

```
Fig04_07.cpp:
Error E2304 Fig04_07.cpp 6: Constant variable 'x' must be
   initialized in function main()
Error E2024 Fig04_07.cpp 8: Cannot modify a const object in
   function main()
*** 2 errors in Compile ***
```

4. Basic C++ Data Types

Type	Keyword
Integer	short - int - long
Real	float - double - long double
Character	char
String	string
Boolean	bool

4. Basic C++ Data Types

- **Real:** hold numbers that have fractional part with different levels of precision, depending on which of the three floating-point types is used.
  Example: float PI = 3.14;

- **Character:** hold a single character such as ‘a’, ‘A’ and ‘$’.
  Example: char ch = ‘a’;

- **String:** store sequences of characters, such as words or sentences.
  Example: string mystring = "This is a string";

- **Boolean:** hold a Boolean value. It may be assigned an integer value **1** (**true**) or a value **0** (**false**).
  Example: bool status;

4. Basic C++ Data Types

typedef Declarations

- You can rename an existing type using **typedef**.

```
typedef   type   freshname;
```

- For example, this tells the compiler that number is another name for int:

```
typedef    int      number;
```

- Therefore, the following declaration is perfectly legal and creates an integer variable called distance:

```
number   distance;
```

5. a Simple Program:

Printing a Line of Text

- **`std::cout`**
  - ➢ Standard output stream object
  - ➢ “Connected” to the screen
  - ➢ **`std::`** specifies the "namespace" which **`cout`** belongs to
    - - **`std::`** can be removed through the use of **`using`** statements
- **`<<`**
  - ➢ Stream insertion operator
  - ➢ Value to the right of the operator (right operand) inserted into output stream (which is connected to the screen)
  - ➢ **`std::cout << “Welcome to C++!\n”;`**
- **`\`**
  - ➢ Escape character
  - ➢ Indicates that a “special” character is to be output

5. 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.

```cpp
// Fig. 1.4: fig01_04.cpp
// Printing a line with multiple statements
#include <iostream>

int main()
{
   cout << "Welcome ";
   cout << "to C++!\n";

   return 0;   // indicate that program ended
}
```

```
Welcome to C++!
```

Unless new line '\n' is specified, the text continues on the same line.

Outline

1. Load <iostream>

2. main

2.1 Print "Welcome"

2.2 Print "to C++!"

2.3 newline

2.4 exit (return 0)

Program Output

```cpp
// Fig. 1.5: fig01_05.cpp
// Printing multiple lines with a single
#include <iostream>

int main()
{
    cout << "Welcome\nto\n\nC++!\n";

    return 0;    // indicate that program ended
}
```

```
Welcome
to

C++!
```

Multiple lines can be printed with one statement.

Outline

1. Load `<iostream>`

2. `main`

2.1 Print `"Welcome"`

2.2 newline

2.3 Print `"to"`

2.4 newline

2.5 newline

2.6 Print `"C++!"`

2.7 newline

2.8 exit (`return 0`)

Program Output

6. a Simple Program:

Adding Two Integers

- (stream extraction operator)
  - When used with **`std::cin`**, waits for the user to input a value and stores the value in the variable to the right of the operator
  - The user types a value, then presses the *Enter* (Return) key to send the data to the computer
  - Example:

```
int myVariable;
std::cin >> myVariable;
```

    - Waits for user input, then stores input in **`myVariable`**

- **=** (assignment operator)
  - Assigns value to a variable
  - Binary operator (has two operands)
  - Example:

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

```
// Fig. 1.6: fig01_06.cpp
// Addition program
#include <iostream>

int main()
{
   int integer1, integer2, sum;              //

   cout << "Enter first integer
   cin >> integer1;
   cout << "Enter second integer\n";  // prompt
   cin >> integer2;                   // read
   sum = integer1 + integer2;              //
   cout << "Sum is " << sum << std::endl; //

   return 0;   // indicate that prog
}
```

Notice how **std::cin** is used to get user input.

std::endl flushes the buffer and prints a newline.

Variables can be output using **std::cout << variableName**.

```
Enter first integer
45
Enter second integer
72
Sum is 117
```

Outline

Load <iostream>.1

2. main

2.1 Initialize variables integer1,

first integer"

2.2.1 Get input

2.3 Print "Enter second integer"

2.4 Add variables and put result into sum

2.5 Print "Sum is"

2.5.1 Output sum

2.6 exit (return 0)

Program Output

7. a Simple Program:

Calculating the area of a Circle

```
# include <iostream>
# define   PI    3.14
using namespace std;
int main ( )
{   /* This program asks the user to enter a radius then calculate the
area */
float  radius, Area;
cout<< “ Please enter a radius: “ ;
cin>> radius;
Area    =   PI  *  radius  *   radius  ;
cout<<  “ The area of the circle is ” << Area ;
return 0;
}
```

- Write a program to calculate the volume of a sphere.