Introduction to C++ Programming: Lecture 2

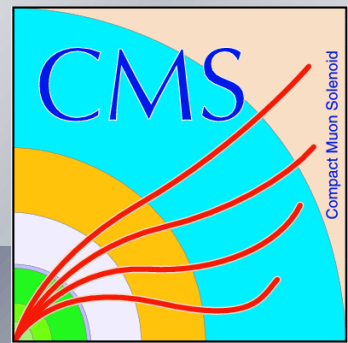
Presented by

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.



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

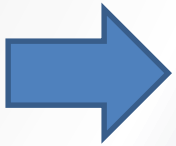
Stream insertion operator.

namespace **std**.

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

Simples examples

```
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 }
```



Simples examples

```
2  // Addition program.
3  #include <iostream>
4
5  // function main begins program execution
6  int main()
7  {
8     int integer1;  // first number to be input by user
9     int integer2;  // second number to be input by user
10    int sum;       // variable in wh
11
12    std::cout << "Enter first inte
13    std::cin >> integer1;
14
15    std::cout << "Enter second in
16    std::cin >> integer2;
17
18    sum = integer1 + integer2;  // assign result to sum
19
20    std::cout << "Sum is " << sum << std::endl; // print sum
21
22    return 0;   // indicate that program ended successfully
23
24 } // end function main
```

Declare integer variables.

Use stream extraction operator with standard input stream to obtain user input.

Calculations can be performed in output statements: alternative for lines 18 and 20:

```
std::cout << "Sum is " << integer1 + integer2 << std::endl;
```

`std::endl` outputs a newline, then "flushes output buffer."

Concatenating, chaining or cascading stream insertion operations.

Simples examples

```
1  // i/o example
2
3  #include <iostream>
4  using namespace std;
5
6  int main ()
7  {
8    int i;
9    cout << "Please enter an integer value: ";
10   cin >> i;
11   cout << "The value you entered is " << i;
12   cout << " and its double is " << i*2 << ".\n";
13   return 0;
14 }
```

```
Please enter an integer value: 702
The value you entered is 702 and its double is 1404.
```

Simples examples

```
1  // cin with strings
2  #include <iostream>
3  #include <string>
4  using namespace std;
5
6  int main ()
7  {
8    string mystr;
9    cout << "What's your name? ";
10   getline (cin, mystr);
11   cout << "Hello " << mystr << ".\n";
12   cout << "What is your favorite team? ";
13   getline (cin, mystr);
14   cout << "I like " << mystr << " too!\n";
15   return 0;
16 }
```

Getline() To get an entire line from cin, there exists a function, that takes the stream (cin) as first argument, and the string variable as second

```
What's your name? Homer Simpson
Hello Homer Simpson.
What is your favorite team? The Isotopes
I like The Isotopes too!
```

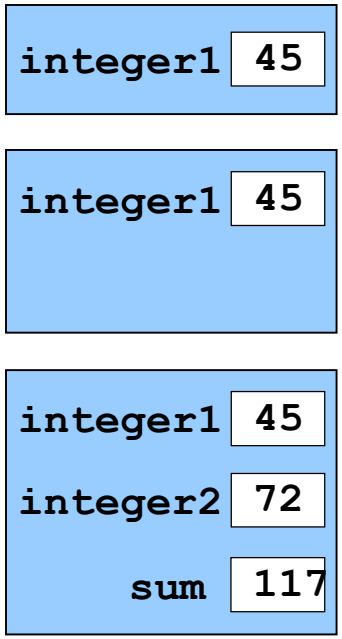
Memory Concepts

- **Variable names**
  - Correspond to actual locations in computer's memory
  - Every variable has name, type, size and value
  - When new value placed into variable, overwrites previous value

  - std::cin >> integer1;
  - Assume user entered 45

  - std::cin >> integer2;
  - Assume user entered 72

  - sum = integer1 + integer2;



Memory Concepts

- Arithmetic calculations
  - `*` : Multiplication
  - `/` : Division
    - Integer division truncates remainder
      - `7 / 5` evaluates to 1
  - `%` : Modulus operator returns remainder
    - `7 % 5` evaluates to 2

Operator(s)	Operation(s)	Order of evaluation (precedence)
`()`	Parentheses	Evaluated first. If the parentheses are nested, the expression in the innermost pair is evaluated first. If there are several pairs of parentheses "on the same level" (i.e., not nested), they are evaluated left to right.
`*`, `/`, or `%`	Multiplication Division Modulus	Evaluated second. If there are several, they re evaluated left to right.
`+` or `-`	Addition Subtraction	Evaluated last. If there are several, they are evaluated left to right.

Decision Making: Equality and Relational Operators

- `if` structure
  - Make decision based on truth or falsity of condition
    - If condition met, body executed
    - Else, body not executed
- Equality and relational operators
  - Equality operators
    - Same level of precedence
  - Relational operators
    - Same level of precedence
  - Associate left to right
- `using` statements
  - Eliminate use of `std::` prefix
  - Write `cout` instead of `std::cout`

Decision Making: Equality and Relational Operators

Standard algebraic equality operator or relational operator	C++ equality or relational operator	Example of C++ condition	Meaning of C++ condition
Relational operators			
>	`>`	`x > y`	`x` is greater than `y`
<	`<`	`x < y`	`x` is less than `y`
≥	`>=`	`x >= y`	`x` is greater than or equal to `y`
≤	`<=`	`x <= y`	`x` is less than or equal to `y`
Equality operators			
=	`==`	`x == y`	`x` is equal to `y`
≠	`!=`	`x != y`	`x` is not equal to `y`

Logical operators (!, &&, ||)

&& OPERATOR (and)		
a	b	a && b
true	true	True
true	false	false
False	true	false
False	false	False

|| OPERATOR (or)		
a	b	a || b
true	true	true
true	false	true
false	true	true
false	false	false

```
1  ( (5 == 5) && (3 > 6) ) // evaluates to false ( true && false )
2   ( (5 == 5) || (3 > 6) ) // evaluates to true ( true || false )
```


Thanks!