CS118 – Programming Fundamentals

Lecture # 06
Tuesday, September 03, 2019
FALL 2019
FAST – NUCES, Faisalabad Campus

Zain Iqbal

Basic Components of C++ Program

Some common include statements

- Basic I/O: iostream
  - Provides functionality of input and output
- I/O manipulation: iomanip
  - Format's the input and output
- Standard Library: stdlib.h
  - Functions for memory allocation, process control, conversion etc.
- Time and Date support: time.h
  - Functionality of time manipulation
- Mathematics support: math.h
  - Functionality of basic mathematical functions

Basics of a Typical C++ Program 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

I/O Streams and Standard I/O Devices

- **I/O:** sequence of bytes (stream of bytes) from source to destination
  - Bytes are usually characters, unless program requires other types of information
- **Stream:** Sequence of characters from source to destination
- **Input stream:** Sequence of characters from an input device to the computer
- **Output stream:** Sequence of characters from the computer to an output device

I/O Streams and Standard I/O Devices (cont'd.)

- Use iostream header file to extract (receive) data from keyboard and send output to the screen
- Contains definitions of two data types:
  - **istream**: input stream
  - **ostream**: output stream
- Has two variables:
  - **cin**: stands for common input
  - **cout**: stands for common output

I/O Streams and Standard I/O Devices (cont'd.)

- To use **cin** and **cout**, the preprocessor directive #include <iostream> must be used
- Variable declaration is similar to:
  - istream cin;
  - ostream cout;
- **Input stream variables:** type istream
- **Output stream variables:** type ostream

Basics of a Typical C++ Program Environment

- Insertion operator & extraction
- 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
- Use **Using namespace std;** to reduce typing work

```
cin >> variable >> variable ...;
```

Basics of a Typical C++ Program Environment …

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

Hello World++


```
// Hello World program

#include <iostream>

int main() {

  std::cout << "Hello World\n";

  return 0;
}
```

comment

Allows access to an I/O library

Starts definition of special function main()

output (print) a string

Program returns a status code (0 means OK)

A Simple Program: Printing a Line of Text

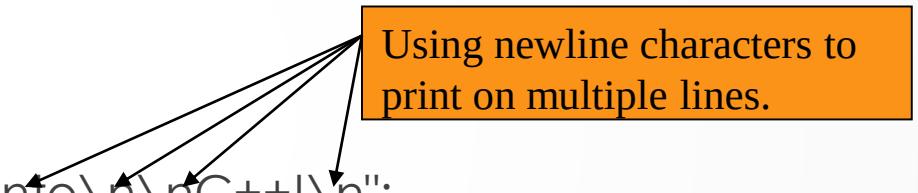

```
1   // C++ Prgram
2   // Printing a line with multiple statements.
3   #include <iostream>
4
5   // function main begins program execution
6   int main()
7   {
8      std::cout << "Welcome ";
9      std::cout << "to C++!\n";
10
11     return 0;   // indicate that program ended successfully
12
13  } // end function main
```

Multiple stream insertion statements produce one line of output.

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

A Simple Program: Printing a text in multiple lines


```
1   // C++ Program
2   // Printing multiple lines with a single statement
3   #include <iostream>
4
5   // function main begins program execution
6   int main()
7   {
8      std::cout << "Welcome\nto\n\nC++!\n";
9
10     return 0;   // indicate that program ended successfully
11
12  } // end function main
```




```
Welcome
to

C++!
```

Escape Sequences

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 (8-spaces).
\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.

C++ comments

- Comments appear in green in Visual C++.
- Comments are explanatory notes; they are ignored by the compiler.
- There are two ways to include comments in a program:

```
// A double slash marks the start of a
//single line comment

/* A slash followed by an asterisk marks the start of a multiple line comment. It ends with an asterisk followed by a slash. */
```

Comments

- Comments contain text that is not converted to machine language (it's just there for humans).
- Everything after "//" is ignored by the compiler.
- Everything between "/*" and "*/" is ignored.

- Document programs
- Improve program readability
- Ignored by compiler
- Single-line comment

Example – adding 2 numbers

Peter: Hey Frank, I just learned how to add two numbers together.

Frank: Cool!

Peter : Give me the first number.

Frank: 2.

Peter : Ok, and give me the second number.

Frank: 5.

Peter : Ok, here's the answer: 2 + 5 = 7.

Frank: Wow! You are amazing!

Ask computer to solve the same problem?
Problem: **Add two Numbers**
Programmer: You

after Frank says "2", Peter has to keep this number in his mind.

after Frank says "5", Peter also needs to keep this number in his mind.

First number: 2 **Second number:** 5 **Sum:** 7

The Corresponding C++ Program


```
#include <iostream>
using namespace std;
int main()
{
        int first, second, sum;
        cout << "Peter: Hey Frank, I just learned how to add"
                << " two  numbers together."<< endl;
        cout << "Frank: Cool!" <<endl;
        cout << "Peter: Give me the first number."<< endl;
        cout << "Frank: ";
        cin >> first;
        cout << "Peter: Give me the second number."<< endl;
        cout << "Frank: ";
        cin >> second;
        sum = first + second;
        cout << "Peter: OK, here is the answer:";
        cout << sum << endl;
        cout << "Frank: Wow! You are amazing!" << endl;
        return 0;
}
```

C++ identifiers

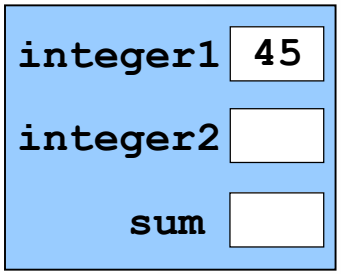
Identifiers appear in **black** in Visual C++.

- An **identifier** is a name for a variable, constant, function, etc.
- It consists of a letter followed by any **sequence of letters**, **digits**, and **underscores**
- Must Begin with a **Letter** Or **Underscore**
- Examples of valid identifiers: First_name, age, y2000, y2k
- Examples of **invalid identifiers**: 2000y
- Identifiers cannot have special characters in them. For example: X=Y, J-20, ~Ricky, *Michael are invalid identifiers
- Identifiers are case-sensitive. For example: Hello, hello, WHOAMI, WhoAmI, whoami are unique identifiers

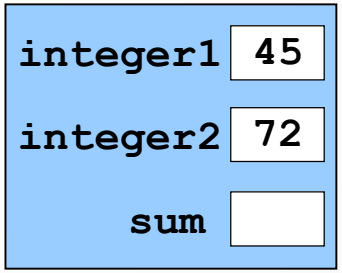
Memory Concepts

Variable names

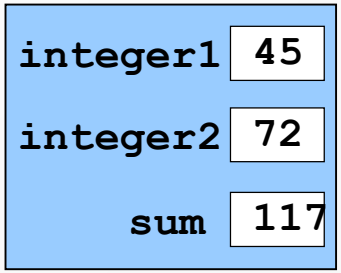
- Correspond to **actual locations** in computer's memory
- Every variable has **name, type, size** and **value**
- When new value placed into variable, **overwrites** previous value



- cin >> integer1;
- Assume user entered 45



- cin >> integer2;
- Assume user entered 72



- sum = integer1 + integer2;

Memory Concepts

Variables

- Location in memory where value can be stored
- Common data types
- **int** - integer numbers like 34, 79, 23167…
- **char** – characters, like ‘a’, ‘$’,…
- **double** -floating point numbers like 3.1416, 5.675, …
- 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;
```

C++ identifiers

Variables

- Variable names
- Valid identifier
  - Series of characters (letters, digits, underscores)
  - Cannot begin with digit
  - Case sensitive

Adding Two Integers

= (assignment operator)

- Assigns value to variable
- Binary operator (two operands)
- Example:
  - sum = variable1 + variable2;

Another C++ Program


```
// C++ Addition of integers
#include <iostream>
Using namespace std;

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

      cout << "Enter first integer\n";
      cin >> integer1;
      cout << "Enter second integer\n";
      cin >> integer2;
      sum = integer1 + integer2;
      cout << "Sum is " << sum << endl;
      return 0;
}
```

```
1       // C++ Program
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 which sum will be stored
11
12         std::cout << "Enter first integer\n";  // prompt
13         std::cin >> integer1;                  // read an integer
14
15         std::cout << "Enter second integer\n"; // prompt
16         std::cin >> integer2;                  // read an integer
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.

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

Concatenating, chaining or cascading stream insertion operations.

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

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

Program Output


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

Constants

- Constants are data values that can not be changed during program execution
- Constants have type like integer, floating-point, character, string and boolean
  - const double pi=3.1415926536;

Questions