



In this lecture you will learn:

- **Escape sequences.**
- **Using Directives.**
- **cin and Extraction operator >>**

### Escape sequences

The backslash ( \ ) is called an **escape character**. It indicates that a “**special**” character is to be output. When a backslash is encountered in a string of characters, the next character is combined with the backslash to form an escape sequence. The escape sequence `\n` means newline. It causes the cursor (i.e., the current screen-position indicator) to move to the beginning of the next line on the screen. Some common escape sequences are listed below:

Escape sequence	Description
<code>\n</code>	Newline. Position the screen cursor to the beginning of the next line.
<code>\t</code>	Horizontal tab. Move the screen cursor to the next tab stop.
<code>\r</code>	Carriage return. Position the screen cursor to the beginning of the current line; do not advance to the next line.
<code>\a</code>	Alert. Sound the system bell.
<code>\\</code>	Backslash. Used to print a backslash character.
<code>\'</code>	Single quote. Use to print a single quote character.
<code>\"</code>	Double quote. Used to print a double quote character.

### Example 1:

```
// Text-printing program.  
  
#include <iostream> // allows program to output data to the screen  
  
// function main begins program execution  
  
using std::cout; // program uses cout  
  
using std::cin; // program uses cin  
  
using std::endl; // program uses endl  
  
int main()  
  
{  
  
cout << "Welcome to C++!\n"; // display message  
  
return 0; // indicate that program ended successfully  
  
} // end function main
```

### Using Directives

The **using directives** that eliminate the need to repeat the `std::` prefix as we did in earlier programs. We can now write `cout` instead of **`std::cout`**, `cin` instead of **`std::cin`** and `endl` instead of **`std::endl`**, respectively, in the remainder of the program. Many programmers prefer to use the directive

```
using namespace std;
```

Which enables a program to use all the names in any standard C++ header (such as `<iostream>`) that a program might include. From this point forward, we'll use the preceding directive in our programs.

## **cin and the Extraction Operator >>**

In this lecture, you will learn how to input data from the standard input device by using cin and the extraction operator >>. Suppose **payRate** is a double variable. Consider the following C++ statement:

```
cin >> payRate;
```

When the computer executes this statement, it inputs the next number typed on the keyboard and stores this number in **payRate**. Therefore, if the user types 15.50, the value stored in payRate is 15.50.

The extraction operator >> is binary and thus takes two operands. The left-side operand must be an input stream variable, such as cin. Because the purpose of an input statement is to read and store values in a memory location and because only variables refer to memory locations, the right-side operand is a variable.

The syntax of an input statement using cin and the extraction operator >> is:

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

**Suppose you have the following variable declarations:**

```
int a, b;
```

```
double z;
```

```
char ch;
```

The following statements show how the extraction operator >> works.

### **Example 2:**

```
// Addition program that displays the sum of two integers.
```

```
#include <iostream> // allows program to perform input and output  
using namespace std; // allows using cin, cout, and endl without std::  
// function main begins program execution  
int main()  
{  
    // variable declarations  
    int number1; // first integer to add  
    int number2; // second integer to add  
    int sum; // sum of number1 and number2  
    cout << "Enter first integer: "; // prompt user for data  
    cin >> number1; // read first integer from user into number1  
    cout << "Enter second integer: "; // prompt user for data  
    cin >> number2; // read second integer from user into number2  
    sum = number1 + number2; // add the numbers; store result in sum  
    cout << "Sum is " << sum << endl; // display sum; end line  
} // end function main
```

## **Programming Exercises**

- Write a single C++ statement to accomplish each of the following (assume that using directives have not been used):
  - a) Declare the variables c, thisIsAVariable, q76354 and number to be of type int.

- b) Prompt the user to enter an integer. End your prompting message with a colon (:) followed by a space and leave the cursor positioned after the space.
- c) Read an integer from the user at the keyboard and store it in integer variable age.
- d) Print the message "This is a C++ program" on one line.
- e) Print the message "This is a C++ program" on two lines. End the first line with C++.
- f) Print the message "This is a C++ program" with each word on a separate line.
- g) Print the message "This is a C++ program". Separate each word from the next by a tab.
- What, if anything, prints when each of the following C++ statements is performed? If nothing prints, then answer "nothing." Assume  $x = 2$  and  $y = 3$ .
  - a) `cout << x;`
  - b) `cout << x + x;`
  - c) `cout << "x=";`
  - d) `cout << "x = " << x;`
  - e) `cout << x + y << " = " << y + x;`
  - f) `z = x + y;`
  - g) `cin >> x >> y;`
  - h) `// cout << "x + y = " << x + y;`

i) `cout << "\n";`

- Write a program that asks the user to enter two numbers, obtains the two numbers from the user and prints the sum, product, difference, and quotient of the two numbers.
- Write a program that prints the numbers 1 to 4 on the same line with each pair of adjacent numbers separated by one space. Do this several ways:
  - a) Using one statement with one stream insertion operator.
  - b) Using one statement with four stream insertion operators.
  - c) Using four statements.
- Write a program that accepts as input the mass, in grams, and density, in grams per cubic centimeters, and outputs the volume of the object using the formula: **density = mass / volume**. Format your output to two decimal places.
- Write a program that prompts the user to enter the weight of a person in kilograms and outputs the equivalent weight in pounds. Output both the weights rounded to two decimal places. (Note that 1 kilogram = 2.2 pounds.) Format your output with two decimal places.

### **References:**

- [1] Deitel, P. & Deitel, R., "C++ How to Program", Eighth Edition, Pearson Education Inc., 2012.
- [2] Stefan B., " C++ Primer Plus ", Sixth Edition, Pearson Education Inc., 2012.