Programming Language (630203)
Fall 2010/2011 – Lecture Notes # 2

Basic Elements of C++

Objectives of the Lecture

- **Basic Components of a C++ program.**
- **Preprocessor Directives.**
- **Data Types.**
- **string Data Type.**
- **cout Output Statement.**
- **cin Input Statements.**
- **Declaring and Initializing Variables.**

Basic components of a C++ program

```
#include <iostream>
using namespace std;
int main()
{
     int num;
     num = 6;
     cout << "My first C++ program." << endl;
     cout << "The sum of 2 and 3 = " << 5 << endl;
     cout << "7 + 8 = " << 7 + 8 << endl;
     cout << "Num = " << num << endl;
     return 0;
}
```

- **Function**: collection of statements; when executed, accomplishes something
  - May be predefined or standard
- **Syntax**: rules that specify which statements (instructions) are legal
  - Programming language: a set of rules, symbols, and special words
- **Semantic** rule: meaning of the instruction
- **Comments** are for the reader, not the compiler
  - Two types:
    - **Single line**

//This is a C++ program. It prints the sentence:
// Welcome to C++ Programming.

- **Multiple line**

/*
  You can include comments that can
  occupy several lines.
*/

- **Reserved words, keywords, or word symbols**
  - Include:
    - int
    - float
    - double
    - char
    - const
    - void
    - return
- **Identifiers and variables:**
  - Consist of letters, digits, and the underscore character (_)
  - Must begin with a letter or underscore
  - C++ is case sensitive
    - NUMBER is not the same as number
  - Two predefined identifiers are **cout** and **cin**
  - Unlike reserved words, predefined identifiers may be redefined, but it is not a good idea

Preprocessor Directives

- C++ has a small number of operations
- Many functions and symbols needed to run a C++ program are provided as collection of libraries
- Every library has a name and is referred to by a header file
- Preprocessor directives are commands supplied to the preprocessor
- All preprocessor commands begin with #
- No semicolon at the end of these commands
- Syntax to include a header file:

```
#include <headerFileName>
```

- For example:

```
#include <iostream>
```

Causes the preprocessor to include the header file iostream in the program

- namespace and using cin and cout in a Program
- **cin** and **cout** are declared in the header file **iostream**, but within **std** namespace
  - To use cin and cout in a program, use the following two statements:

```
#include <iostream>
using namespace std;
```

Data Types

- **Data type**: set of values together with a set of operations
- C++ data types fall into three categories:

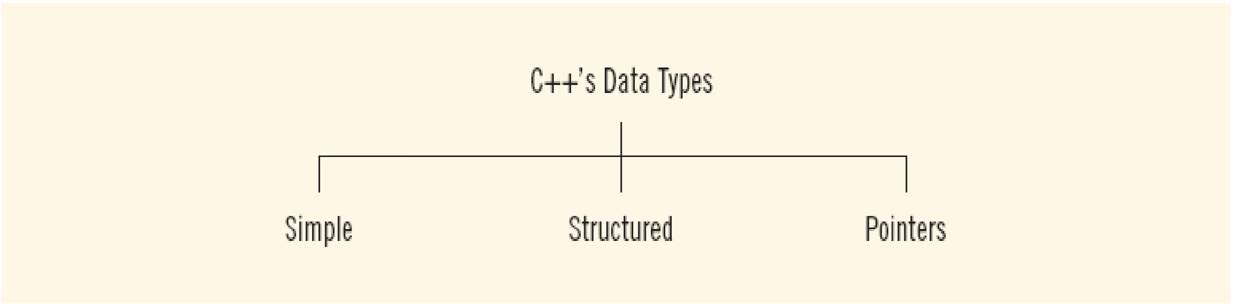


FIGURE 2-1 C++ data types

- **Simple Data Types**
- Three categories of simple data
  - **Integral**: integers (numbers without a decimal point)
  - **Floating-point**: decimal numbers
  - **Enumeration type**: user-defined data type
- **Integral data** types are further classified into nine categories:
  - **char, short, int, long, bool**
  - **unsigned char, unsigned short, unsigned int, unsigned long**
  - **bool Data Type**
- **bool type**
  - Two values: true and false.
  - Manipulate logical (Boolean) expressions: true and false.
  - Logical values: bool, true, and false are reserved words.
- **char Data Type**
  - The smallest integral data type.
  - Used for characters: letters, digits, and special symbols.
  - Each character is enclosed in single quotes.
  - 'A', 'a', '0', '*', '+', '$', '&'
  - A blank space is a character.
  - Written ' ', with a space left between the single quotes.
- **Floating-Point Data Types**
  - float: represents any real number; Range: -3.4E+38 to 3.4E+38 (four bytes)
  - double: represents any real number; Range: -1.7E+308 to 1.7E+308 (eight bytes)

string Type

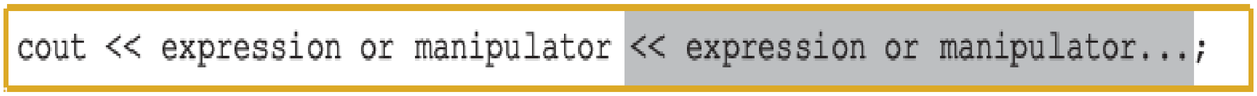
- Sequence of zero or more characters enclosed in double quotation marks.
- Using the **string** Data Type in a Program
  - To use the string type, you need to access its definition from the header file string
  - Include the following preprocessor directive:

```
#include  <string>
```

- Use **string** data types to declare a string variable.

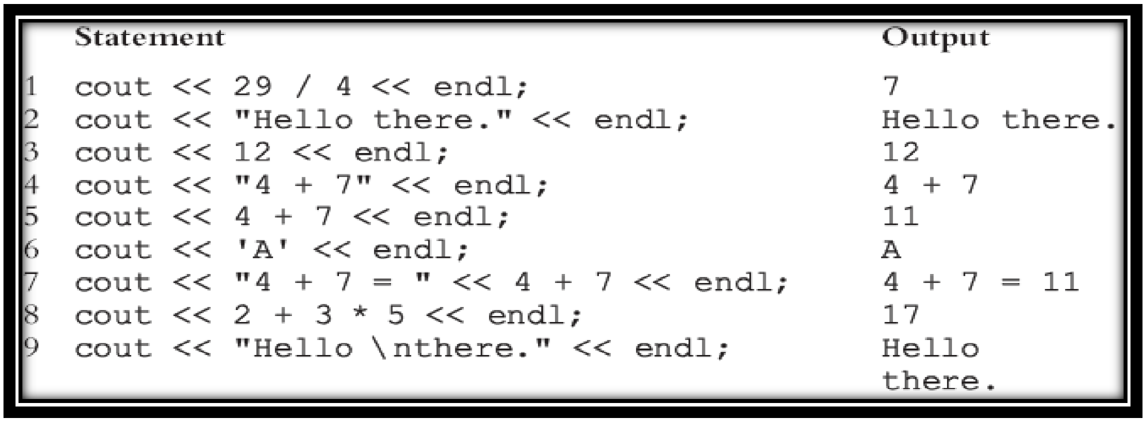
cout output statement

- The syntax of **cout** and << is:



```
cout << expression or manipulator << expression or manipulator...;
```

- The **stream insertion operator** is <<
- Expression evaluated and its value is printed at the current cursor position on the screen



	Statement	Output
1	`cout << 29 / 4 << endl;`	7
2	`cout << "Hello there." << endl;`	Hello there.
3	`cout << 12 << endl;`	12
4	`cout << "4 + 7" << endl;`	4 + 7
5	`cout << 4 + 7 << endl;`	11
6	`cout << 'A' << endl;`	A
7	`cout << "4 + 7 = " << 4 + 7 << endl;`	4 + 7 = 11
8	`cout << 2 + 3 * 5 << endl;`	17
9	`cout << "Hello \nthere." << endl;`	Hello there.

- The new line character is **'\n'**
  - May appear anywhere in the string

    Example 1

    ```
    cout << "Hello there.";
    cout << "My name is James.";
    ```

    the output is

    ```
    Hello there.My name is James.
    ```

    Example 2

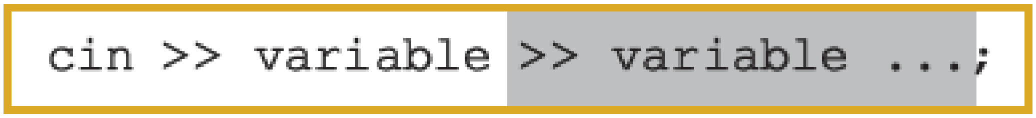
    ```
    cout << "Hello there.\n";
    cout << "My name is James.";
    ```

    the output is :

    ```
    Hello there.
    My name is James.
    ```

Input Statement

- Data must be loaded into main memory before it can be manipulated.
- **cin** is used with **>>** to gather input (Read statement) with the following syntax:



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

- The stream extraction operator is >>
  - For example, if miles is a double variable

    ```
    cin >> miles;
    ```

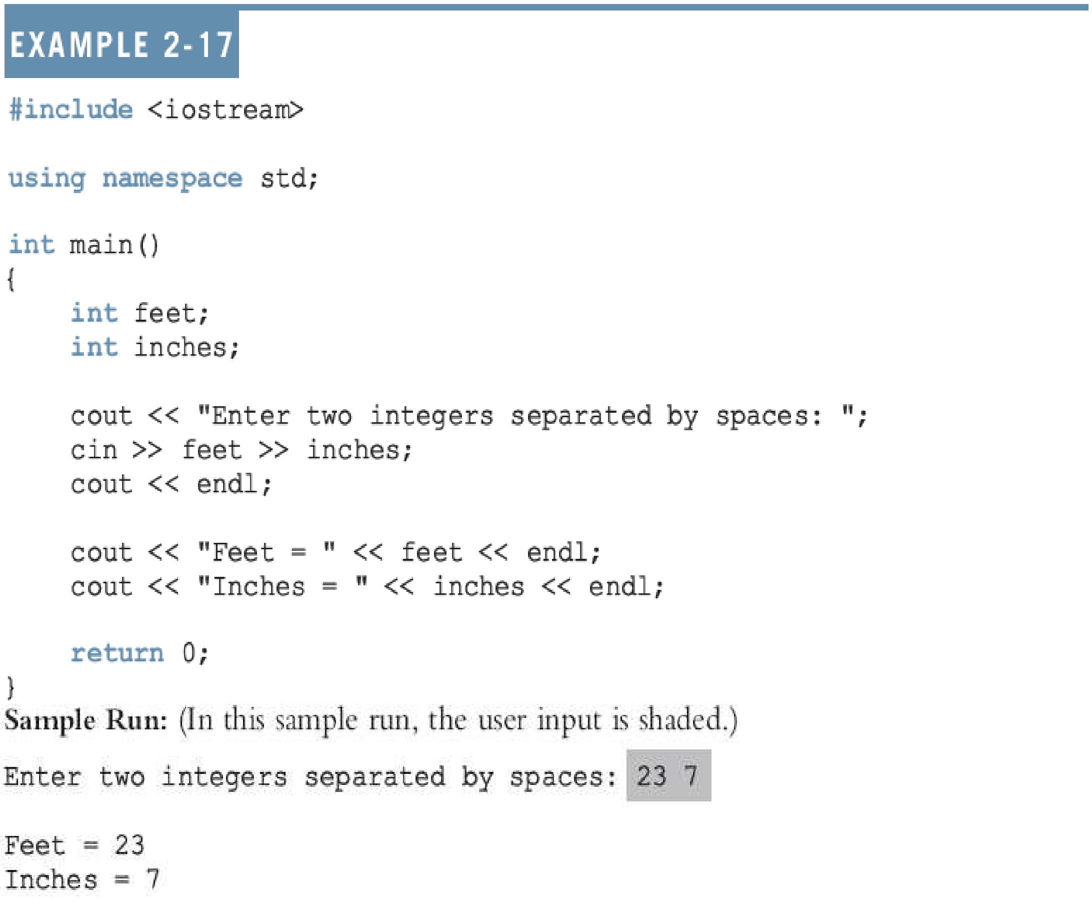
Causes computer to get a value of type double

  - Using more than one variable in cin allows more than one value to be read at a time

For example, if feet and inches are variables of type int, a statement such as:

```
cin >> feet >> inches;
```

Inputs two integers from the keyboard and places them in variables feet and inches respectively



EXAMPLE 2-17

```
#include <iostream>

using namespace std;

int main()
{
    int feet;
    int inches;

    cout << "Enter two integers separated by spaces: ";
    cin >> feet >> inches;
    cout << endl;

    cout << "Feet = " << feet << endl;
    cout << "Inches = " << inches << endl;

    return 0;
}
```

Sample Run: (In this sample run, the user input is shaded.)

```
Enter two integers separated by spaces: 23 7

Feet = 23
Inches = 7
```

Declaring & Initializing Variables

- Variables can be initialized when declared:

```
int first=13, second=10;
char ch=' ';
double x=12.6;
```

- All variables must be initialized before they are used
  - But not necessarily during declaration
- **Variable Initialization**
  - There are two ways to initialize a variable:

```
int feet;
```

    - By using the assignment statement

```
feet = 35;
```

    - By using a read statement

```
cin >> feet;
```