COMSATS University
Islamabad
(Lahore Campus)

CSC103-Programming Fundamentals

MS. MAHWISH WAQAS

MAWISH.WAQAS@CUILAHORE.EDU.PK

Chapter 3: Input/Output

Objectives

- In this chapter, you will:
  - Learn what a stream is and examine input and output streams
  - Explore how to read data from the standard input device
  - Learn how to use predefined functions in a program
  - Explore how to use the input stream functions `get`

Objectives (cont’d.)

- Become familiar with input failure
- Learn how to write data to the standard output device
- Discover how to use manipulators in a program to format output
- Learn how to perform input and output operations with the `string` data type
- Learn how to debug logic errors

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 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

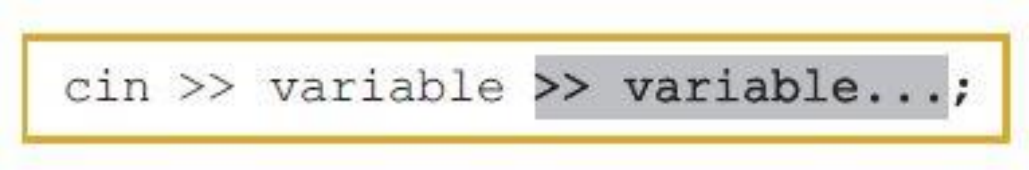
I/O Streams and Standard I/O Devices (cont'd.)

- Variable declaration is similar to:
  - `istream cin;`
  - `ostream cout;`
- To use `cin` and `cout`, the preprocessor directive

  `#include <iostream>` must be used
- Input stream variables: type `istream`
- Output stream variables: type `ostream`

cin and the Extraction Operator >>

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



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

- The extraction operator >> is binary
  - Left-side operand is an input stream variable
    - Example: cin
  - Right-side operand is a variable

cin and the Extraction Operator >> (cont'd.)

- No difference between a single cin with multiple variables and multiple cin statements with one variable
- When scanning, >> skips all whitespace
  - Blanks and certain nonprintable characters
- >> distinguishes between character 2 and number 2 by the right-side operand of >>
  - If type char then 2 is treated as a character; if type or int (or double), the 2 is treated as a number

cin and the Extraction Operator >> (cont'd.)

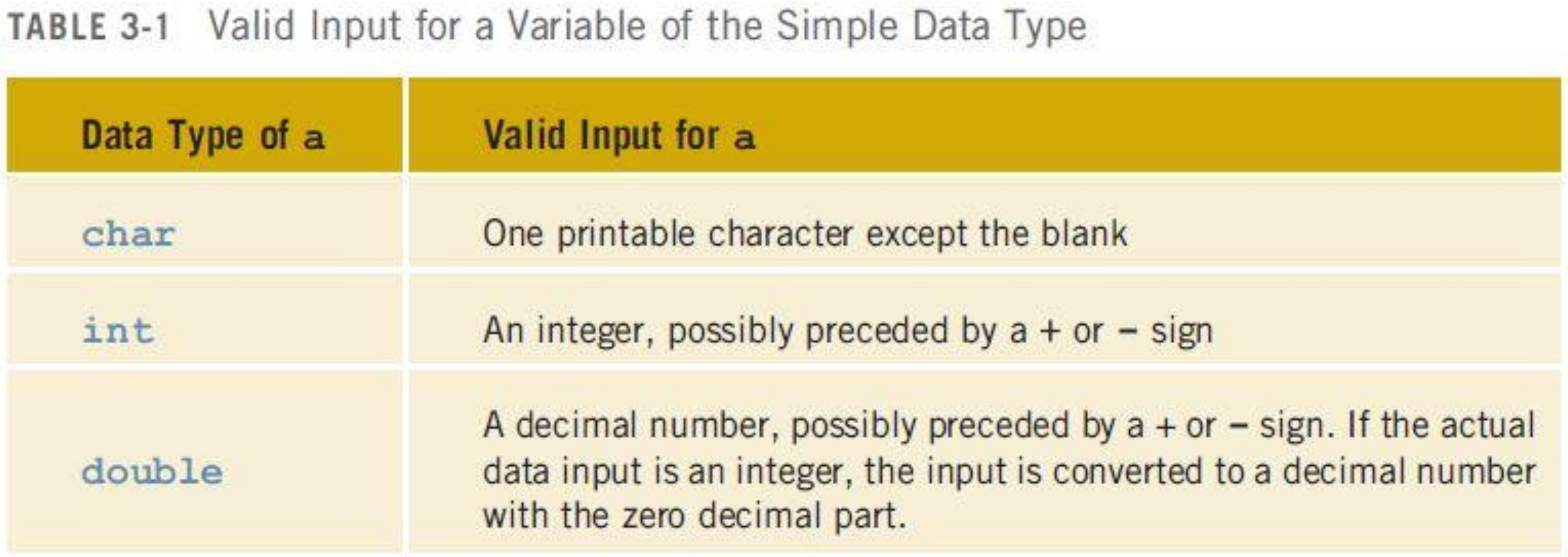


TABLE 3-1 Valid Input for a Variable of the Simple Data Type

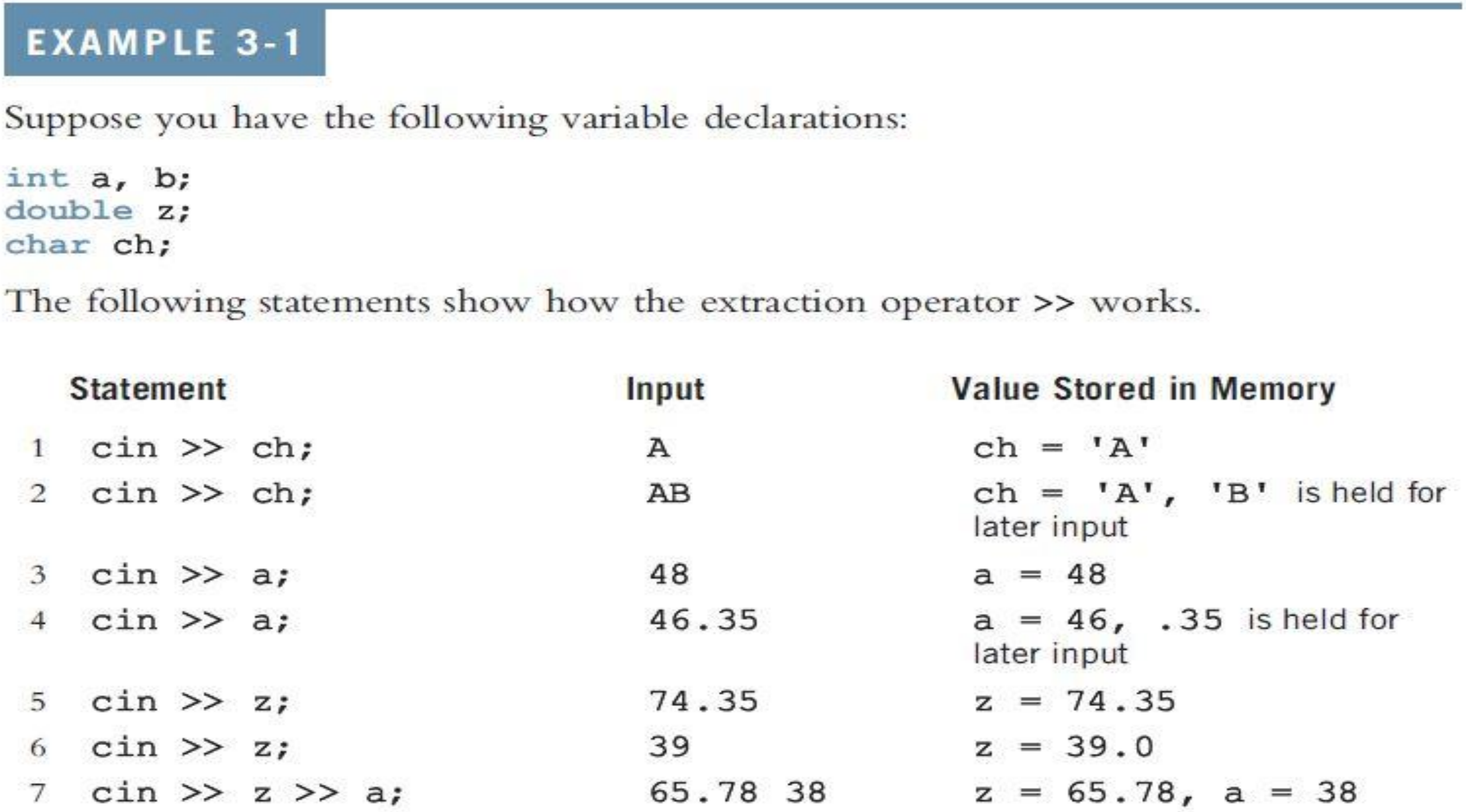
Data Type of a	Valid Input for a
char	One printable character except the blank
int	An integer, possibly preceded by a + or − sign
double	A decimal number, possibly preceded by a + or − sign. If the actual data input is an integer, the input is converted to a decimal number with the zero decimal part.

- Entering a char value into an int or double variable causes serious errors, called input failure

`cin` and the Extraction Operator >> (cont'd.)

- When reading data into a `char` variable
  - `>>` skips leading whitespace, finds and stores only the next character
  - Reading stops after a single character
- To read data into an `int` or `double` variable
  - `>>` skips leading whitespace, reads + or - sign (if any), reads the digits (including decimal)
  - Reading stops on whitespace non-digit character

`cin` and the Extraction Operator >> (cont’d.)



EXAMPLE 3-1

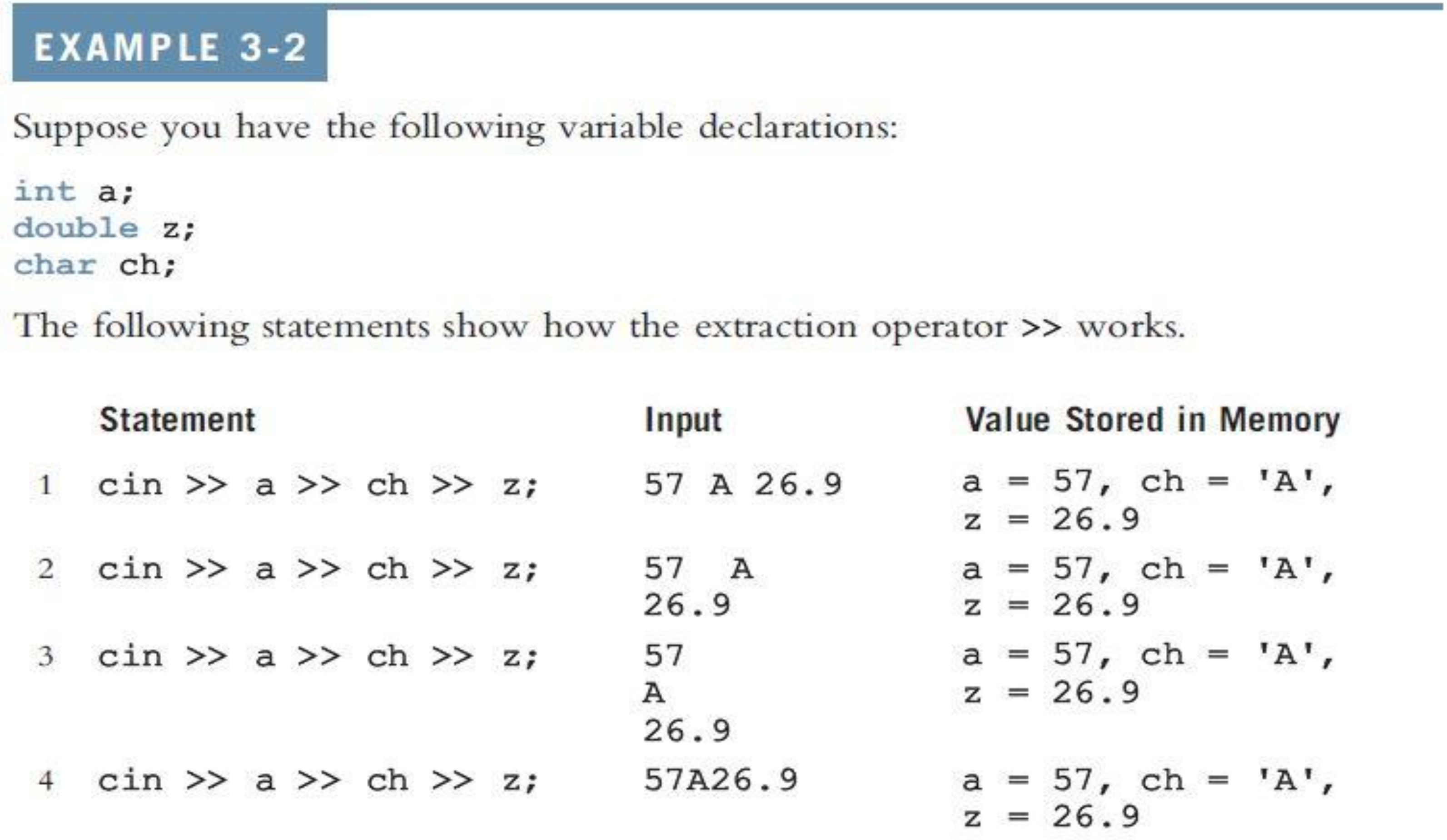
Suppose you have the following variable declarations:

```
int a, b;
double z;
char ch;
```

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

	Statement	Input	Value Stored in Memory
1	`cin >> ch;`	`A`	`ch = 'A'`
2	`cin >> ch;`	`AB`	`ch = 'A', 'B'` is held for later input
3	`cin >> a;`	`48`	`a = 48`
4	`cin >> a;`	`46.35`	`a = 46, .35` is held for later input
5	`cin >> z;`	`74.35`	`z = 74.35`
6	`cin >> z;`	`39`	`z = 39.0`
7	`cin >> z >> a;`	`65.78 38`	`z = 65.78, a = 38`

`cin` and the Extraction Operator >> (cont'd.)



EXAMPLE 3-2

Suppose you have the following variable declarations:

```
int a;
double z;
char ch;
```

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

	Statement	Input	Value Stored in Memory

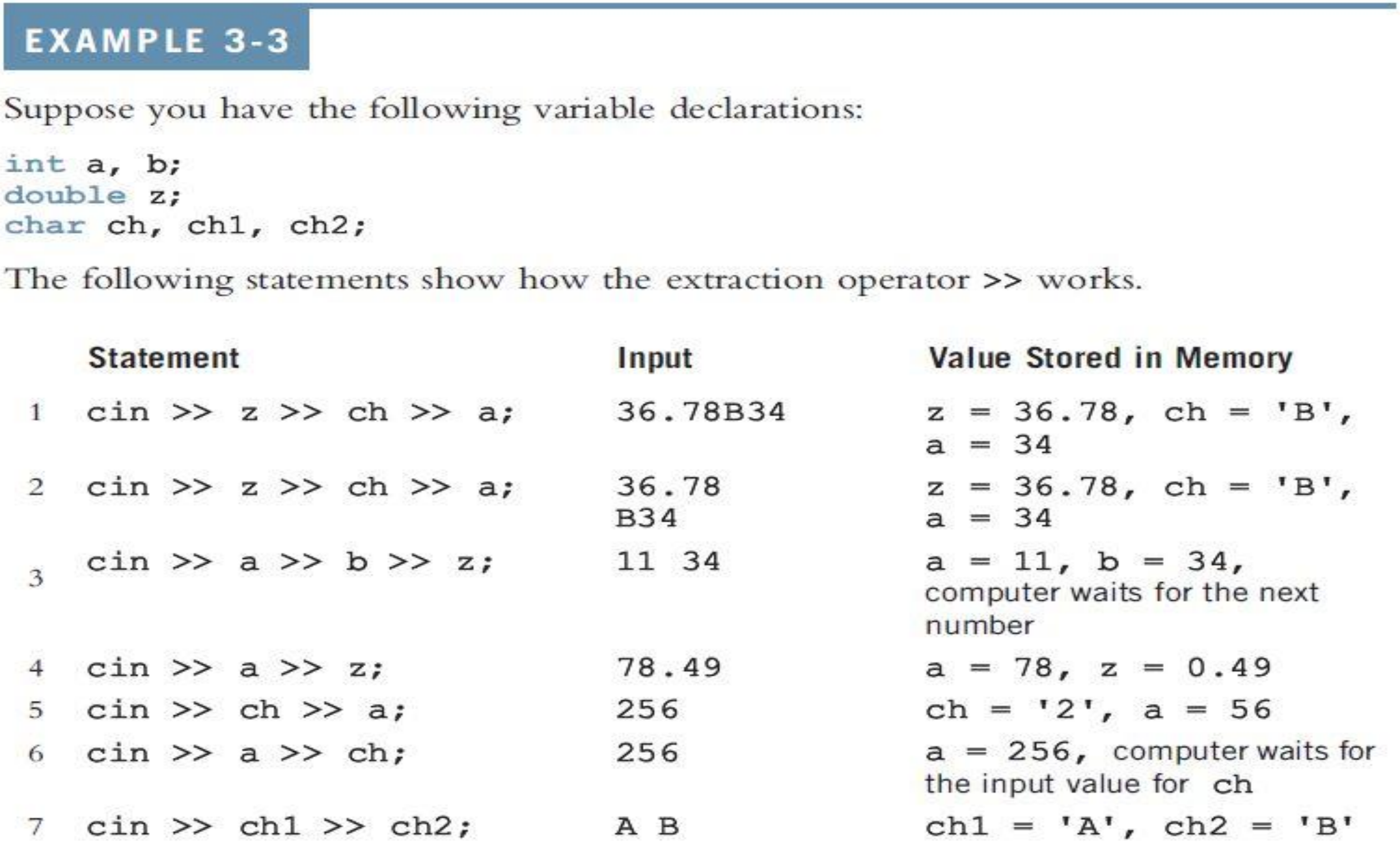
1	`cin >> a >> ch >> z;`	`57 A 26.9`	`a = 57, ch = 'A',`
			`z = 26.9`

2	`cin >> a >> ch >> z;`	`57   A`
		`26.9`
			`a = 57, ch = 'A',`
			`z = 26.9`

3	`cin >> a >> ch >> z;`	`57`
		`A`
		`26.9`
			`a = 57, ch = 'A',`
			`z = 26.9`

4	`cin >> a >> ch >> z;`	`57A26.9`	`a = 57, ch = 'A',`
			`z = 26.9`

cin and the Extraction Operator >> (cont'd.)



EXAMPLE 3-3

Suppose you have the following variable declarations:

```
int a, b;
double z;
char ch, ch1, ch2;
```

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

	Statement	Input	Value Stored in Memory

1	`cin >> z >> ch >> a;`	`36.78B34`	`z = 36.78, ch = 'B',`
			`a = 34`

2	`cin >> z >> ch >> a;`	`36.78`
		`B34`
			`z = 36.78, ch = 'B',`
			`a = 34`

3	`cin >> a >> b >> z;`	`11 34`	`a = 11, b = 34,` computer waits for the next number
4	`cin >> a >> z;`	`78.49`	`a = 78, z = 0.49`
5	`cin >> ch >> a;`	`256`	`ch = '2', a = 56`
6	`cin >> a >> ch;`	`256`	`a = 256,` computer waits for the input value for `ch`
7	`cin >> ch1 >> ch2;`	`A B`	`ch1 = 'A', ch2 = 'B'`

Using Predefined Functions in a Program

- Function (subprogram): set of instructions
  - When activated, it accomplishes a task
- `main` executes when a program is run
- Other functions execute only when called
- C++ includes a wealth of functions
  - Predefined functions are organized as a collection of libraries called header files

Using Predefined Functions in a Program (cont'd.)

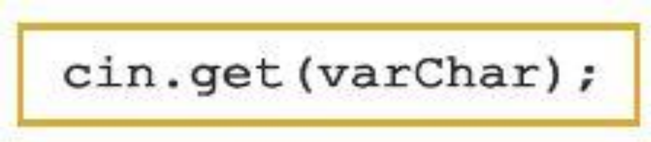
- Header file may contain several functions
- To use a predefined function, you need the name of the appropriate header file
  - You also need to know:
    - Function name
    - Number of parameters required
    - Type of each parameter
    - What the function is going to do

Using Predefined Functions in a Program (cont'd.)

- To use `pow` (power), include `cmath`
  - Two numeric parameters
  - Syntax: `pow(x,y)` = x^y
    - `x` and `y` are the arguments or parameters
  - In `pow(2,3)`, the parameters are `2` and `3`

cin and the get Function

- The get function
  - Inputs next character (including whitespace)
  - Stores in memory location indicated by its argument
- The syntax of cin and the get function:



```
cin.get(varChar);
```

- varChar
  - Is a char variable
  - Is the argument (or parameter) of the function

The Dot Notation Between I/O Stream Variables and I/O Functions

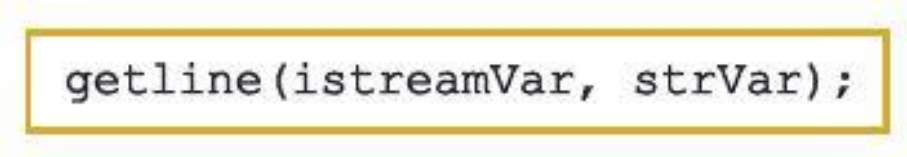
- A precaution
  - In the statement

    ```
    cin.get(ch);
    ```

    `cin` and `get` are two separate identifiers separated by a dot
  - The function get is associated with the variable cin, as it is a member of type istream.
  - Dot separates the input stream variable name from the member (or function) name
  - In C++, dot is the member access operator
    - More applications in Object Oriented Programming

Input/Output and the `string` Type

- An input stream variable (`cin`) and `>>` operator can read a string into a variable of the data type `string`
- Extraction operator
  - Skips any leading whitespace characters
  - Reading stops at a whitespace character
- The function `getline (iostream)`
  - Reads until end of the current line



```
getline(istreamVar, strVar);
```

String input program

```
#include<iostream>
using namespace std;
int main()
{
   string str;
   cout<<"Enter a string: ";
   getline(cin, str);
   cout<<"You Entered: "<<str;
   return 0;
}
```

Input/Output and the `bool` Type

- In bool type variables, C++ stores 1 for true, and 0 for false.
- In fact, any non-zero value in C++ means true, and 0 means false.
  - For example, in the following code, whatever non-zero value is stored in the Boolean variable x, it will always output 1 (true).
    - `bool x = 123;`
    - `cout<<x<<endl;`
- Similarly, For bool variable **user input**, the user must enter a non-zero value for true and 0 for false.

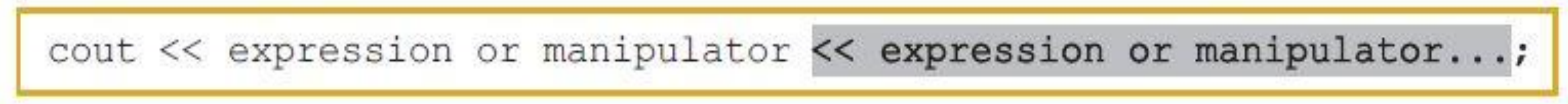
Bool input program

```
#include<iostream>
using namespace std;
int main()
{
   bool gender; // 0 means flase, non-zero (1) means true
   cout<<"Enter gender: ";
   cin>>gender;
   cout<<"You Entered: "<<gender;
   return 0;
}
```

Formatting program output

Output and Formatting Output

- Syntax of `cout` when used with `<<`



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

- `expression` is evaluated
  - It’s `value` is printed
- `manipulator` is used to format the output
  - Example: `endl`

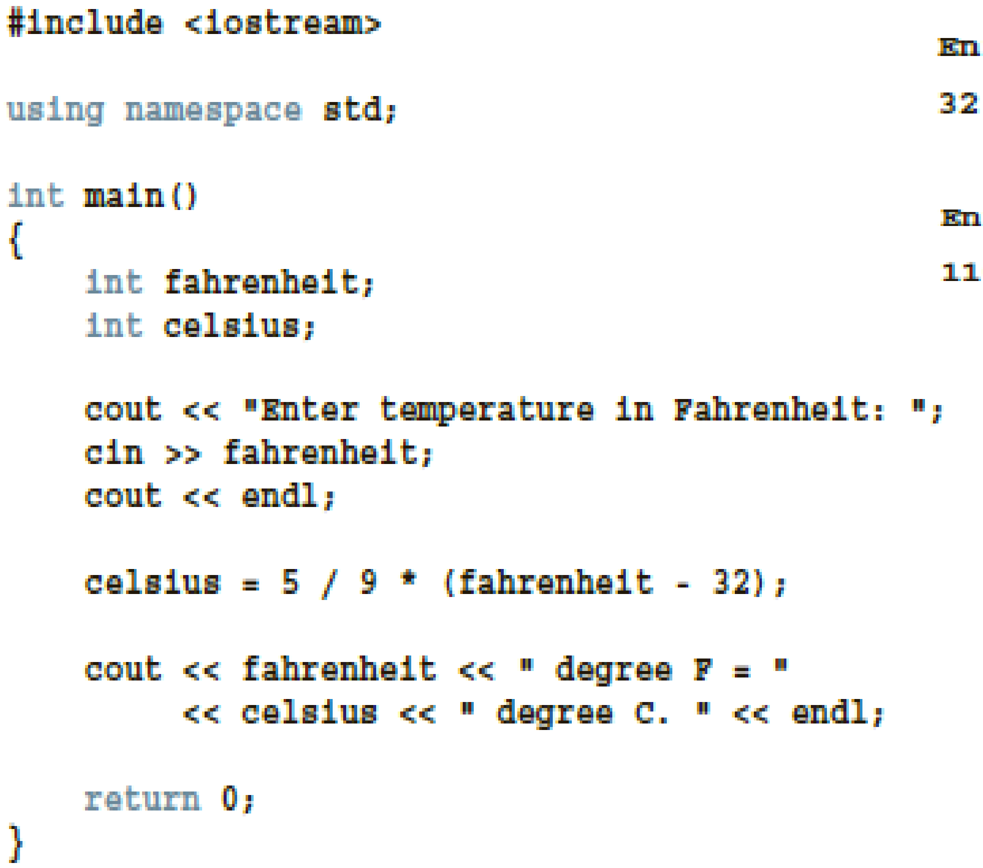
Types of Manipulators

- Two types of manipulators:
  - With parameters
  - Without parameters
- Parameterized: require `iomanip` header
  - `setprecision`, `setw`, and `setfill`
- Nonparameterized: require `iostream` header
  - `endl`, `fixed`, `showpoint`, `left`, `right`, `internal` and `flush`

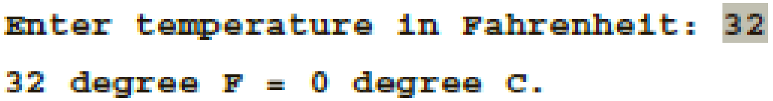
Debugging: Understanding Logic Errors and Debugging with `cout` statements

- Syntax errors
  - Reported by the compiler
- Logic errors
  - Typically not caught by the compiler
  - Spot and correct using `cout` statements
  - Temporarily insert an output statement
  - Correct problem
  - Remove output statement

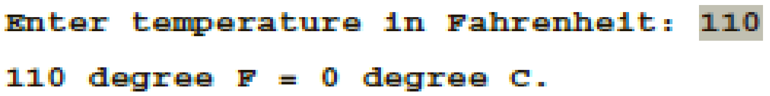
Debugging: Understanding Logic Errors and Debugging with `cout` statements



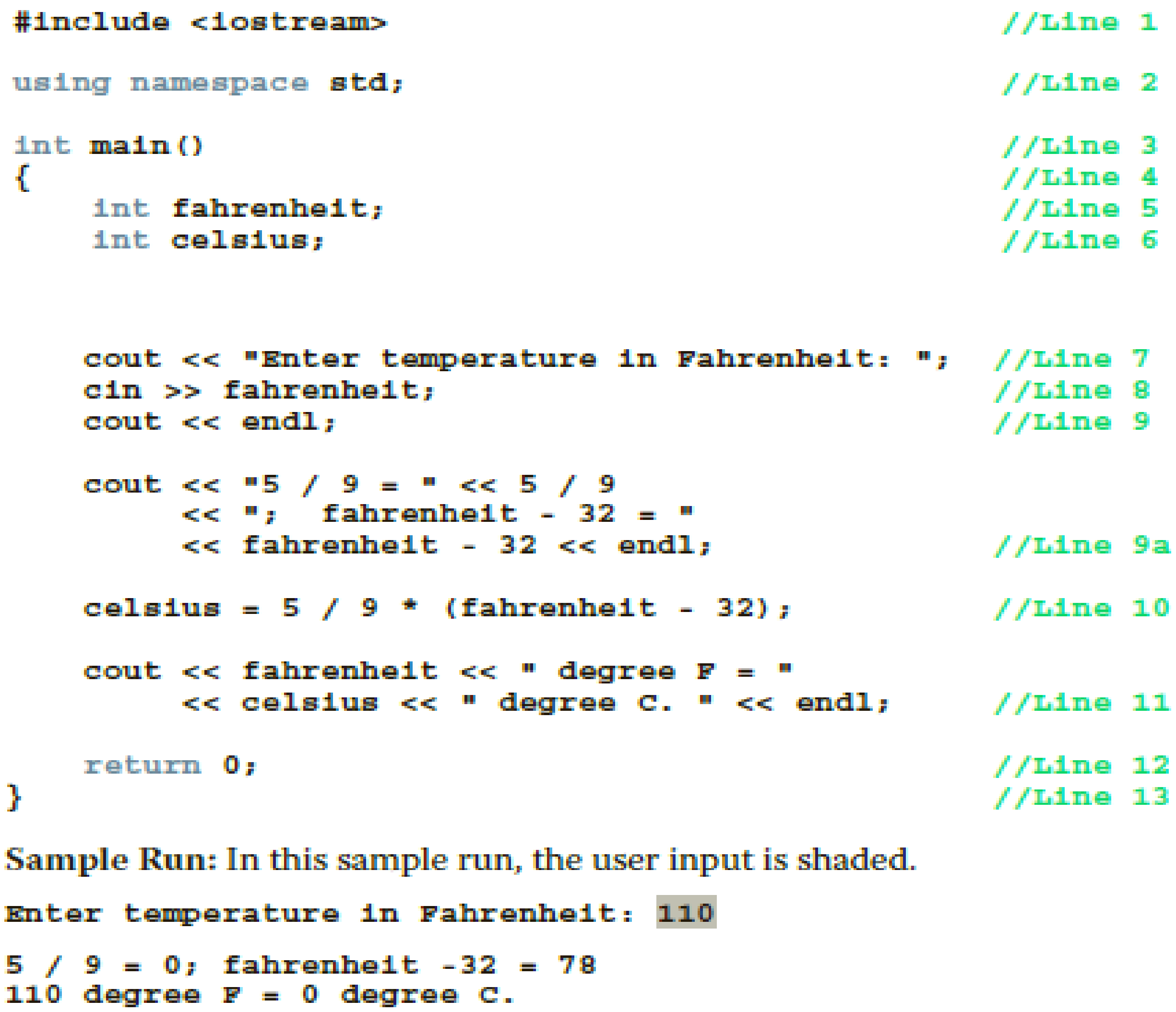
```
#include <iostream>

using namespace std;

int main()
{
    int fahrenheit;
    int celsius;

    cout << "Enter temperature in Fahrenheit: ";
    cin >> fahrenheit;
    cout << endl;

    celsius = 5 / 9 * (fahrenheit - 32);

    cout << fahrenheit << " degree F = "
         << celsius << " degree C. " << endl;

    return 0;
}
```



```
Enter temperature in Fahrenheit: 32
32 degree F = 0 degree C.
```



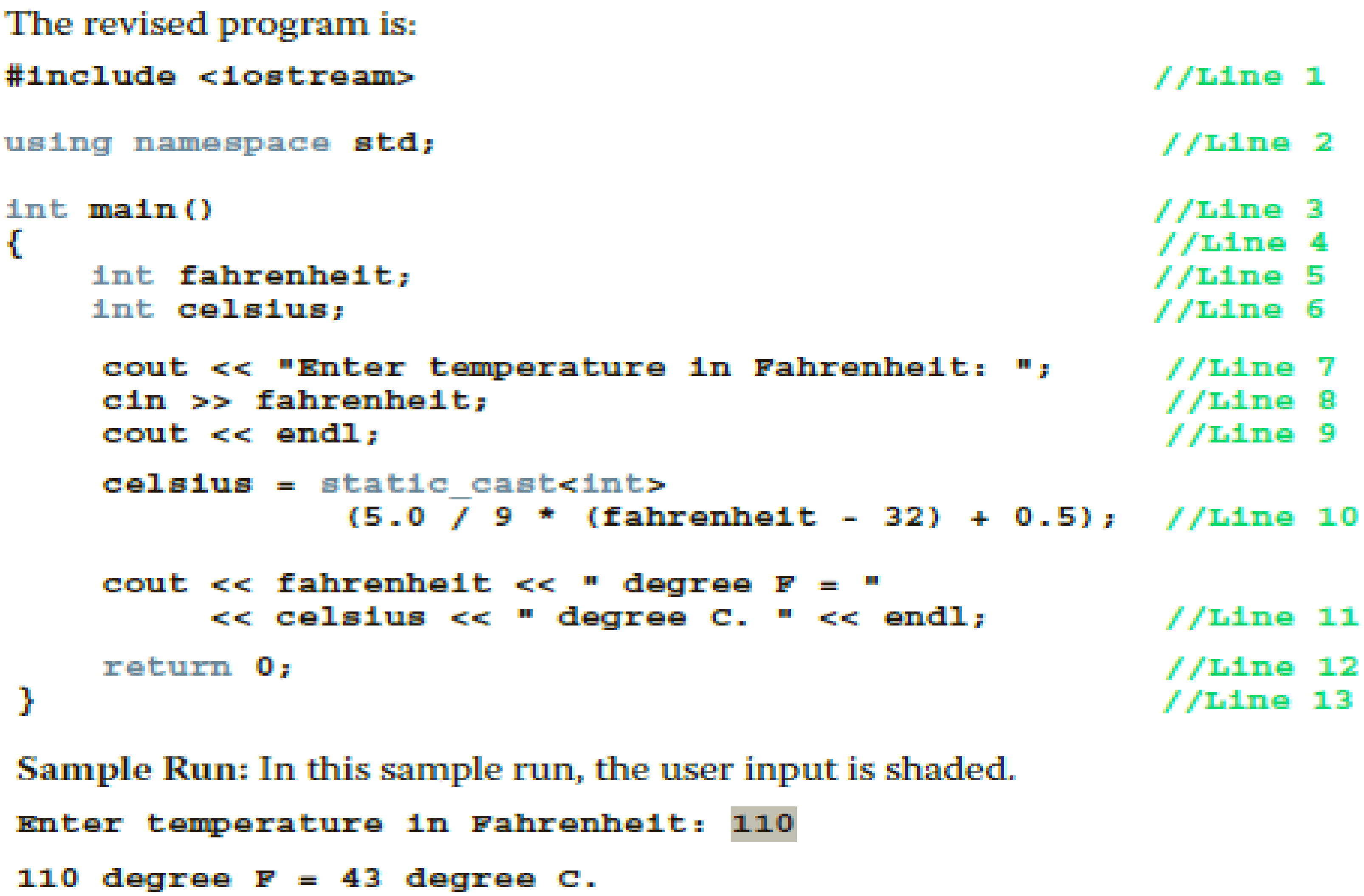
```
Enter temperature in Fahrenheit: 110
110 degree F = 0 degree C.
```



```
#include <iostream>                                       //Line 1

using namespace std;                                      //Line 2

int main()                                                //Line 3
{                                                         //Line 4
    int fahrenheit;                                       //Line 5
    int celsius;                                          //Line 6


    cout << "Enter temperature in Fahrenheit: ";         //Line 7
    cin >> fahrenheit;                                   //Line 8
    cout << endl;                                        //Line 9

    cout << "5 / 9 = " << 5 / 9
         << ";  fahrenheit - 32 = "
         << fahrenheit - 32 << endl;                     //Line 9a

    celsius = 5 / 9 * (fahrenheit - 32);                 //Line 10

    cout << fahrenheit << " degree F = "
         << celsius << " degree C. " << endl;            //Line 11

    return 0;                                            //Line 12
}                                                        //Line 13
```

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

```
Enter temperature in Fahrenheit: 110

5 / 9 = 0; fahrenheit -32 = 78
110 degree F = 0 degree C.
```



The revised program is:

```
#include <iostream>                                       //Line 1

using namespace std;                                      //Line 2

int main()                                                //Line 3
{                                                         //Line 4
    int fahrenheit;                                       //Line 5
    int celsius;                                          //Line 6

    cout << "Enter temperature in Fahrenheit: ";          //Line 7
    cin >> fahrenheit;                                    //Line 8
    cout << endl;                                         //Line 9

    celsius = static_cast<int>
                (5.0 / 9 * (fahrenheit - 32) + 0.5);      //Line 10

    cout << fahrenheit << " degree F = "
         << celsius << " degree C. " << endl;             //Line 11

    return 0;                                             //Line 12
}                                                         //Line 13
```

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

```
Enter temperature in Fahrenheit: 110

110 degree F = 43 degree C.
```

Summary

- Stream: infinite sequence of characters from a source to a destination
  - Input stream: from a source to a computer
  - Output stream: from a computer to a destination
  - `cin`: common input
  - `cout`: common output
  - To use `cin` and `cout`, include iostream header

Summary (cont'd.)

- The input entered by the user should match the type of variable to store that input.
- Attempting to read invalid data into a variable causes the input stream to enter the fail state.
  - For example, reading a character inside a double or integer variable.

Summary (cont'd.)

- The manipulators `setprecision`, `fixed`, `showpoint`, `setw`, `setfill`, `left`, and `right` can be used for formatting output
- Include `iomanip` for the manipulators `setprecision`, `setw`, and `setfill`