CS1002 – Programming Fundamentals

Lecture # 11
Monday, October 03, 2022
FALL 2022
FAST – NUCES, Faisalabad Campus

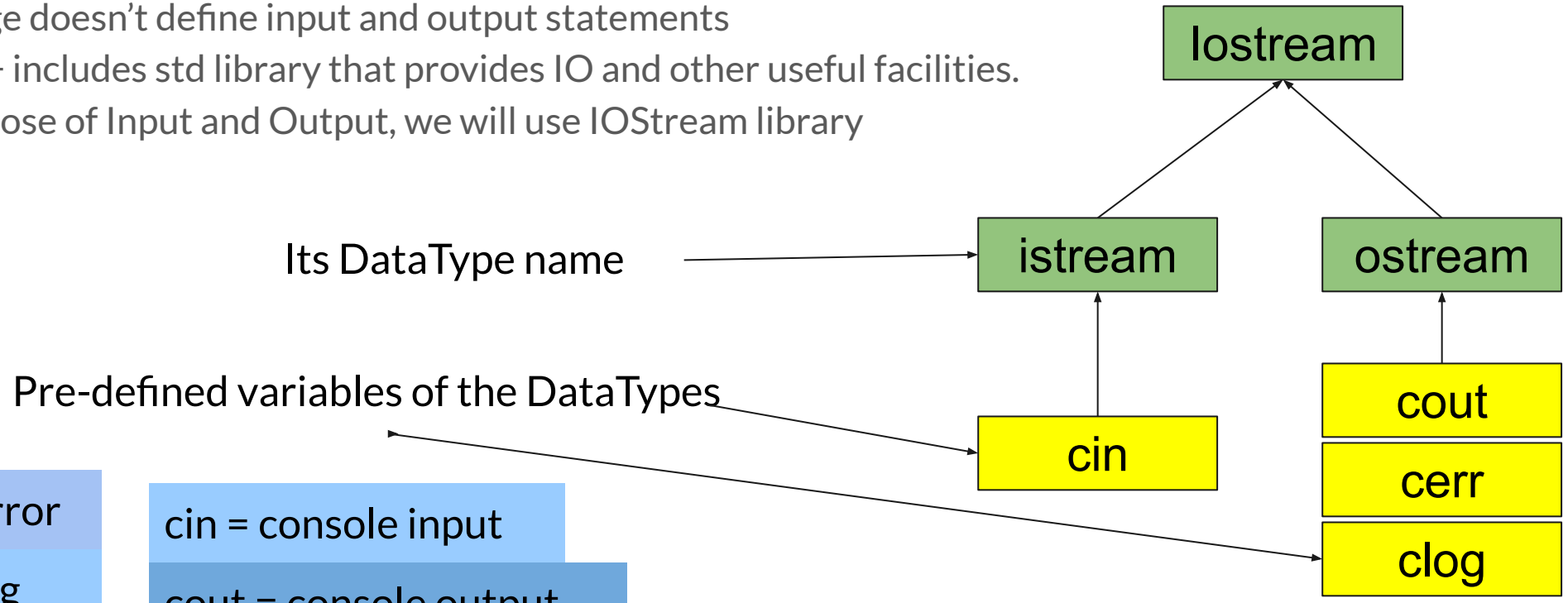
Muhammad Yousaf

Outline

- IOStream Header File
- Stream Objects
- Stream Operators
- Variables & Constants
- C++ Example Programs

IOStream Header File ; Input & Output in C++

- C++ language doesn't define input and output statements
- Instead, C++ includes std library that provides IO and other useful facilities.
- For the purpose of Input and Output, we will use IOStream library



cerr = console error

clog = console log

cin = console input

cout = console output

Streams

- Stream: sequence of characters i.e. stream of water, bikes etc.

- Input Stream: sequence of characters reading from input device.
  - when user enters/inputs data, it lies in the input stream i.e.

- Output Stream: sequence of characters writing to output device
  - when user outputs data, it goes to the output stream

input stream

2 4 a b @ * 88 { P } .....

output stream

2 4 a b .....

Streams Operators >> , <<

- Stream: sequence of characters i.e. stream of water, bikes etc.
- Input Stream: sequence of characters reading from input device.
- Output Stream: sequence of characters writing to output device

```
int num=0; char ch = '';

std::cin >> num >> ch ;

std::cout << num <<" and " << ch  << std::endl;
```

input stream

23 Y a 33 #……

output stream

23 and Y

Basics of a Typical C++ Program Environment ...

- Standard input stream object
  - `std::cin`
  - “Connected” to screen
  - `>>`
    - Stream extraction operator
    - Value from input stream inserted into right operand
- **Namespace**
  - **std::** specifies that entity belongs to “namespace” **using binary scope resolution operator(::)**
  - **std::** removed through use of **using** statements

The Corresponding C++ Program

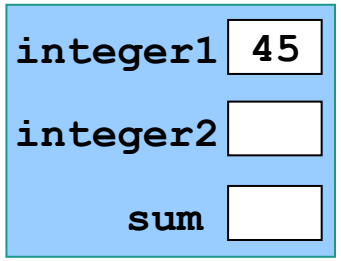
```
#include <iostream>
int main()
{
    int first, second, sum;
    std::cout << "Peter: Hey Frank, I just learned how to add”
        << " two  numbers together."<< std::endl;
    std::cout << "Frank: Cool!" << endl;
    std::cout << "Peter: Give me the first number." << std::endl;
    std::cout <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

Memory Concepts

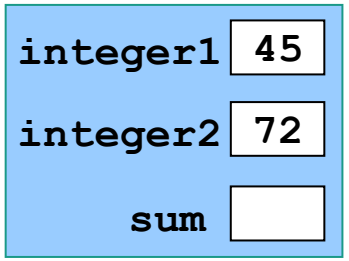
Variable names

- 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

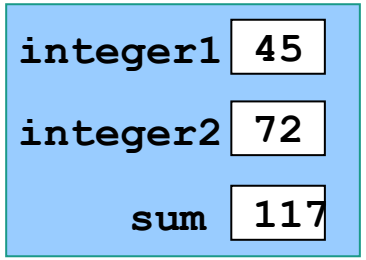
- std::cin >> integer1;
- Assume user entered 45



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



- sum = integer1 + integer2;



Adding Two Integers

= (assignment operator)

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

```
1   // C++ Program
2   // Addition program.
3   #include <iostream>
4
5   // function main begins program
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
23
24
```

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;

Example

- Write a program that gets a length in inches from user, then determine and output the equivalent length in feet and inches. After that converts inches into centimeters
- (Hint: 12 inches in a feet , therefore 100 inches equal to 8 feet and 4 inches; 1 inch=2.54 centimeters)

Allocating Memory with Constants **and** Variables

- Storing data in the computer's memory is a two-step process:
  1. Instruct the computer to allocate memory
  2. Include statements in the program to put data into the allocated memory

Contd..

- **Named constant:** A memory location whose content is not allowed to change during program execution.
- To allocate memory, we use C++'s declaration statements. The syntax to declare a named constant is:

```
const dataType identifier = value;
```

Consider the following C++ statements:

```
const double CONVERSION = 2.54;
const int NO_OF_STUDENTS = 20;
const char BLANK = ' ';
```

Assignment Statement

- The assignment statement takes the form:

```
variable = expression;
```

Should match datatype of variable

- Expression is evaluated and its value is assigned to the variable on the left side
- In C++, '=' is called the assignment operator

- Value can be assigned by taking input from user

Assignment Statement (cont'd.)

Example 1:

```
int num1, num2;
double sales;
char ch;
float average;
string str;

num1 = 4;
num2 = 4 * 5 - 10 ;
sales = 0.03 * 50000;
ch = 'A';
str = "It is sunny day";
```

==========

Example 2:

int num1, num2, num3;

1. num1 = 18 ;
2. num1 = num1 + 27 ;
3. num2 = num1 ;
4. num3 = num2 / 5 ;
5. num3 = num3 / 4 ;

Variable and memory

1. num1 = 18 ;
2. num1 = num1 + 27 ;
3. num2 = num1 ;
4. num3 = num2 / 5 ;
5. num3 = num3 / 4 ;

	Values of the Variables			Explanation
Before Statement 1	? num1	? num2	? num3	
After Statement 1	18 num1	? num2	? num3	
After Statement 2	45 num1	? num2	? num3	`num1 + 27 = 18 + 27 = 45.` This value is assigned to `num1`, which replaces the old value of `num1`.
After Statement 3	45 num1	45 num2	? num3	Copy the value of `num1` into `num2`.
After Statement 4	45 num1	45 num2	9 num3	`num2 / 5 = 45 / 5 = 9.` This value is assigned to `num3`. So `num3 = 9`.
After Statement 5	45 num1	45 num2	2 num3	`num3 / 4 = 9 / 4 = 2.` This value is assigned to `num3`, which replaces the old value of `num3`.

Saving and Using the Value of an Expression

- To save the value of an expression:
  - Declare a variable of the appropriate data type
  - Assign the value of the expression to the variable that was declared
    - Use the assignment statement
- Wherever the value of the expression is needed, use the variable holding the value

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

Input Statement

Putting data into variables from the standard input device is accomplished via the use of `cin` and the operator `>>`. The syntax of `cin` together with `>>` is:

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

This is called an **input (read)** statement. In C++, `>>` is called the **stream extraction operator**.

Suppose that `miles` is a variable of type `double`. Further suppose that the input is `73.65`. Consider the following statements:

```
cin >> miles;
```

This statement causes the computer to get the input, which is `73.65`, from the standard input device and stores it in the variable `miles`. That is, after this statement executes, the value of the variable `miles` is `73.65`.

To calculate the equivalent change, the program performs calculations using the values of a half-dollar, which is 50; a quarter, which is 25; a dime, which is 10; and a nickel, which is 5. Because these data are special and the program uses these values more than once, it makes sense to declare them as named constants. Using named constants also simplifies later modification of the program:

```
const int HALF_DOLLAR = 50;
const int QUARTER  = 25;
const int DIME = 10;
const int NICKEL = 5;
```

1. Prompt the user for input.
2. Get input.
3. Echo the input by displaying the entered change on the screen.
4. Compute and print the number of half-dollars.
5. Calculate the remaining change.
6. Compute and print the number of quarters.
7. Calculate the remaining change.
8. Compute and print the number of dimes.
9. Calculate the remaining change.
10. Compute and print the number of nickels.
11. Calculate the remaining change.
12. Print the remaining change.

```
    //Declare variable
int change;

    //Statements: Step 1 - Step 12
cout << "Enter change in cents: ";                    //Step 1
cin >> change;                                        //Step 2
cout << endl;

cout << "The change you entered is " << change
     << endl;                                         //Step 3

cout << "The number of half-dollars to be returned "
     << "is " << change / HALF_DOLLAR
     << endl;                                         //Step 4

change = change % HALF_DOLLAR;                        //Step 5

cout << "The number of quarters to be returned is "
     << change / QUARTER << endl;                     //Step 6

change = change % QUARTER;                            //Step 7

cout << "The number of dimes to be returned is "
     << change / DIME << endl;                        //Step 8

change = change % DIME;                               //Step 9

cout << "The number of nickels to be returned is "
     << change / NICKEL << endl;                      //Step 10

change = change % NICKEL;                             //Step 11

cout << "The number of pennies to be returned is "
     << change << endl;                               //Step 12

return 0;
```

Input Failure Program

```
//Input Failure program

#include <iostream>

using namespace std;

int main()
{
    int a = 10;                                          //Line 1
    int b = 20;                                          //Line 2
    int c = 30;                                          //Line 3
    int d = 40;                                          //Line 4

    cout << "Line 5: Enter four integers: ";             //Line 5
    cin >> a >> b >> c >> d;                             //Line 6
    cout << endl;                                        //Line 7
    cout << "Line 8: The numbers you entered are:"
         << endl;                                        //Line 8
    cout << "Line 9: a = " << a << ", b = " << b
         << ", c = " << c << ", d = " << d << endl;      //Line 9

    return 0;
}
```

Output

Sample Run 1

Line 5: Enter four integers: 34 K 67 28

Line 8: The numbers you entered are:

Line 9: a = 34, b = 20, c = 30, d = 40

Sample Run 2

Line 5: Enter four integers: 43 225.56 39 61

Line 8: The numbers you entered are:

Line 9: a = 43, b = 225, c = 30, d = 40

More on Assignment Statements

- C++ has special assignment statements called compound assignments

  +=, -=, *=, /=, and %=

- Example:

  x = x * y ;

  as

  x *= y ;

Example

EXAMPLE 2-31

This example shows several compound assignment statements that are equivalent to simple assignment statements.

Simple Assignment Statement	Compound Assignment Statement
`i = i + 5;`	`i += 5;`
`counter = counter + 1;`	`counter += 1;`
`sum = sum + number;`	`sum += number;`
`amount = amount * (interest + 1);`	`amount *= interest + 1;`
`x = x / ( y + 5);`	`x /= y + 5;`

Formatting Output

Manipulators

- A manipulator functions format the output to present it in more readable fashion
- **setbase(...)** -- set base field to octal, decimal and hexadecimal of the variables.
  - `std::cout << setbase(16) << “Base 16 is  ”<< 255 << std::endl ;`
- **setw(...)** -- set width of output fields
  - `std::cout << setw(10) << “Hello”<<“\t |1o characters width” << std::endl ;`
- **setfill(...)** -- specifies fill character
  - `std::cout << setw(10) << setfill(‘*’) << “Hello”<<“\t |find difference in format of output” << std::endl ;`
- **setprecision(...)** -- specifies number of decimals for floating point : e.g. `setprecision(2)` ....

setbase(int base); base will be zero other than 8,10,16

```
#include <iostream>
#include <iomanip> // std::setbase
int main()
{
   std::cout << std::setbase(16);
   std::cout << 255 << std::endl;
   std::cout << std::setbase(8);
   std::cout << 255 << std::endl;
   return 0;
}
```

setwidth(int n); n is no. of characters time width

```
#include <iostream>
#include <iomanip> // std::setbase
int main()
{
   std::cout << std::setw(10);
   std::cout << 100 << std::endl;
   std::string str = "GFG";
   std::cout << std::setw(12);
   std::cout << str << std::endl;
   return 0;
}
```

setwidth(int n); n is no. of characters time width

```
#include <iostream>
#include <iomanip> // std::setwidth
#include<string>
using std::cout;
using std::string;
int main()
{
     string temp="Hello setw";
     cout<<std::setw(5)<<temp<< std::endl;
  return 0;
}
```

setfill(char c); c is the stream's fill character

```
#include <iostream>
#include <iomanip> // std::setwidth
#include<string>
int main()
{
    std::cout << std::setfill('x') << std::setw(10);
    std::cout << 77 << std::endl;
    std::string str = "Geeks";
    // setfill is G and width is set as 10
    // And std::left is used set str to left side
    std::cout << std::left << std::setfill('G') << std::setw(10);
    std::cout << str << std::endl;
    return 0;
}
```

```
xxxxxxxx77
GeeksGGGGG
```

Programming Example: Variables and Constants

- **Variables**

```
int feet;                    //variable to hold given feet
int inches;                  //variable to hold given inches
int totalInches;             //variable to hold total inches
double centimeters;          //variable to hold length in
                             //centimeters
```

- **Named Constant**

```
const double CENTIMETERS_PER_INCH = 2.54;
const int INCHES_PER_FOOT = 12;
```

Writing

```
using namespace std;

    //Named constants
const double CENTIMETERS_PER_INCH = 2.54;
const int INCHES_PER_FOOT = 12;

int main ()
{
        //Declare variables
    int feet, inches;
    int totalInches;
    double centimeter;

        //Statements: Step 1 - Step 7
    cout << "Enter two integers, one for feet and "
         << "one for inches: ";                          //Step 1
    cin >> feet >> inches;                               //Step 2
    cout << endl;
    cout << "The numbers you entered are " << feet
         << " for feet and " << inches
         << " for inches. " << endl;                     //Step 3

    totalInches = INCHES_PER_FOOT * feet + inches;       //Step 4

    cout << "The total number of inches = "
         << totalInches << endl;                         //Step 5

    centimeter = CENTIMETERS_PER_INCH * totalInches;     //Step 6

    cout << "The number of centimeters = "
         << centimeter << endl;                          //Step 7

    return 0;
}
```

Programming Example: Sample Run

```
Enter two integers, one for feet, one for inches: 15 7

The numbers you entered are 15 for feet and 7 for inches.
The total number of inches = 187
The number of centimeters = 474.98
```

```
//Example setfile
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
     int x = 15;                    //Line 1
     int y = 7643 ;            //Line 2

     cout << "12345678901234567890" << endl;        //Line 3
     cout << setw(5) << x << setw(7) << y
          << setw(8) <<"Warm" << endl ;             //Line 4

     cout << setfill('*');                          //Line 5
     cout << setw(5) << x << setw(7) << y
          << setw(8) << "Warm" << endl ;            //Line 6

     cout << setw(5) << x << setw(7) << setfill('#')
          << y << setw(8) << "Warm" << endl ;       //Line 7

     cout << setw(5) << setfill('@') << x
           << setw(7) << setfill('#') << y
           << setw(8) << setfill('^') << "Warm"
           << endl ;                                //Line 8

     cout << setfill(' ') ;                         //Line 9
     cout << setw(5) << x << setw(7) << y
```

Sample Run:

```
12345678901234567890
   15   7634    Warm
***15***7634****Warm
***15###7634####Warm
@@@15###7634^^^^Warm
   15   7634    Warm
```

Questions