

Lecture 2

February 23, 2021

Today's Topics

Operator

Basic Input/Output

Control Structures

Operator

Once we know of the existence of variables and constants, we can begin to operate with them.

Assignment (=)

The assignment operator assigns a value to a variable. `a = 5;`

```
[1]: #include <iostream>

using namespace std;
```

```
[2]: int main ()
{
    int a, b;    // a:?, b:?
    a = 10;      // a:10, b:?
    b = 4;       // a:10, b:4
    a = b;       // a:4, b:4
    b = 7;       // a:4, b:7
    cout << "a:";
    cout << a;
    cout << " b:";
    cout << b;
    return 0;
}
```

```
[3]: main();
```

a:4 b:7

```
[ ]: int b;
int a = 2 + (b = 5);
```

```
[4]: b = 5;
      a = 2 + b;
      int c;
```

input_line_11:2:2: **error:** use of undeclared identifier

'b'

```
  b = 5;
  ^
```

input_line_11:3:1: **error:** use of undeclared

identifier 'a'

```
a = 2 + b;
^
```

input_line_11:3:9: **error:** use of undeclared

identifier 'b'

```
a = 2 + b;
      ^
```

Interpreter Error:

```
[6]: int a, b, c;
      a = b = c = 5;
```

Arithmetic operators (+, -, *, /, %)

```
[ ]: int sum1 = 100 + 50;      // 150 (100 + 50)
      int sum2 = sum1 + 250;   // 400 (150 + 250)
      int sum3 = sum2 + sum2;  // 800 (400 + 400)
```

```
[ ]: int sum1 = 100 - 50;      // 50 (100 - 50)
      int sum2 = sum1 - 250;   // -200 (50 - 250)
      int sum3 = sum2 - sum2;  // -400 (-200 - 200)
```

```
[8]: // Modulus
      int a = 11 % 3;
      cout<<a;
```

2

```
[9]: #include <iostream>
      using namespace std;
```

```
[10]: int main()
{
    int num = 5;
    cout << num << endl;
    num = num + 2;    cout << num << endl;
    num = num / 3 * 6;    cout << num << endl;
    cout << 7 + 15 % 4 << endl;
    num = 24 / 3 / 4;    cout << num << endl;
    num = 24 / (num / 4);    cout << num << endl;
}
```

```
[ ]: main();
```

5
7
12
10

Compound assignment (+=, -=, *=, /=, %=, »=, «=, &=, ^=, |=)

```
[12]: //bitwise operator
```

```
int x = 5;

x >>= 2;

cout<<x;
```

1

```
[ ]: #include <iostream>
using namespace std;
```

```
[ ]: int main() {
    int x = 5;
    x %= 3;
    cout << x;
    return 0;
}

// bitwise operator
// x |=3 = 7 // or
// x &=3 = 1 // 1 or 0 - 101 & 11 // and
// x >>= 3 -- 0 shift right
// x <<=3 --40 left shift
// x ^=3 // 6 xor
```

Logical operators (!, &&, ||)

```
[2]: // ! NOT
!(5 == 5); // evaluates to false because the expression at its right (5 == 5)
    ↳ is true.
!(6 <= 4); // evaluates to true because (6 <= 4) would be false.
!true; // evaluates to false
!false;
```

```
[ ]: // && AND
true && true; // true
true && false; // false
false && true; // false
false && false; // false
```

```
[ ]: // || OR
true || true; // true
true || false; // true
false || true; // true
false || false; // false
```

```
[ ]: // example
((5 == 5) && (3 > 6)); // evaluates to false ( true && false ).
((5 == 5) || (3 > 6)); // evaluates to true ( true || false ).
```

Conditional operator

The conditional operator evaluates an expression returning a value if that expression is true and a different one if the expression is evaluated as false. Its format is: condition ? result1 : result2

```
[4]: int a, b;
7==5 ? 4 : 3; // returns 3, since 7 is not equal to 5.
7==5+2 ? 4 : 3; // returns 4, since 7 is equal to 5+2.
5>3 ? a : b; // returns the value of a, since 5 is greater than 3.
a>b ? a : b; // returns whichever is greater, a or b.
```

```
[4]: 0
```

```
[13]: // conditional operator
#include <iostream>
using namespace std;
```

```
[14]: int main ()
{
    int a,b,c;
    a=13;
    b=10;
    c = (a>b) ? a : b;
    cout << c;
    return 0;
```

```
}  
main();
```

13

Explicit type casting operator

Type casting operators allow you to convert a datum of a given type to another

```
[8]: int i;  
     float f = 3.14;  
     i = (int) f;
```

Precedence of operators

```
[9]: a = 5 + (7 % 2); // with a result of 6, or  
     a = (5 + 7) % 2; // with a result of 0
```

[9]: 0

```
[ ]: // exercise  
     //a = 5 + 7 % 2;  
     //a = 5 + (7 % 2);  
     //a = (5 + 7) % 2;
```

Basic Input/Output

Until now, the example programs of previous sections provided very little interaction with the user, if any at all. Using the standard input and output library, we will be able to interact with the user by printing messages on the screen and getting the user's input from the keyboard.

C++ uses a convenient abstraction called streams to perform input and output operations in sequential media such as the screen or the keyboard. A stream is an object where a program can either insert or extract characters to/from it. We do not really need to care about many specifications about the physical media associated with the stream - we only need to know it will accept or provide characters sequentially.

The standard C++ library includes the header file `iostream`, where the standard input and output stream objects are declared.

```
[14]: //Standard Output (cout)  
  
cout << "Output sentence"<< endl; // prints Output sentence on screen  
cout << 120<<endl; // prints number 120 on screen  
cout << 'x'; // prints the content of x on screen
```

Output sentence

120

x

[14]: @0x7fabcb1b6eb60

```
[15]: cout << "Hello, " << "I am " << "a C++ statement";
```

Hello, I am a C++ statement

```
[16]: // Standard Input (cin)
```

```
int age;  
cin>> age;
```

3

```
[17]: #include <iostream>  
using namespace std;
```

```
[18]: int main(){  
  
    float area, pi, r;  
  
    pi = 3.14;  
  
    cout<<"input radius : ";  
    cin>>r;  
  
    area = pi*r*r;  
    cout<<"area of circle is: "<<area;  
    return 0;  
}
```

```
[21]: main();
```

input radius : 3
area of circle is: 28.26

```
[19]: #include <iostream>  
using namespace std;
```

```
[20]: int main ()  
{  
    int i;  
    cout << "Please enter an integer value: ";  
    cin >> i;  
    cout << "The value you entered is " << i;  
    cout << " and its double is " << i*2 << ".\n";  
    return 0;  
}
```

```
}
```

```
[21]: main()
```

Please enter an integer value: 3
The value you entered is 3 and its double is 6.

```
[21]: 0
```

```
[22]: // cin with strings
#include <iostream>
#include <string>
using namespace std;
```

```
[23]: int main ()
{
    string mystr;
    cout << "What's your name? ";
    getline (cin, mystr);
    cout << "Hello " << mystr << ".\n";
    cout << "What is your favorite team? ";
    getline (cin, mystr);
    cout << "I like " << mystr << " too!\n";
    return 0;
}
```

```
[24]: main();
```

What's your name? biruk
Hello biruk.
What is your favorite team? Man utd
I like Man utd too!

stringstream

The standard header file `sstream` defines a class called `stringstream` that allows a string-based object to be treated as a stream. This way we can perform extraction or insertion operations from/to strings, which is especially useful to convert strings to numerical values and vice versa.

```
[26]: // stringstream
#include <iostream>
#include <string>
#include <sstream>
using namespace std;
```

```
[27]: int main ()
{
    string mystr;
```

```

float price=0;
int quantity=0;

cout << "Enter price: ";
getline (cin,mystr);
stringstream(mystr) >> price;

cout << "Enter quantity: ";
getline (cin,mystr);
stringstream(mystr) >> quantity;

cout << "Total price: " << price*quantity <<endl;
return 0;
}

```

[28]: main();

```

Enter price: 20
Enter quantity: 6
Total price: 120

```

[]:

[]:

Write a program in C++ to find the sum of the series $[x - x^3 + x^5 + \dots]$.

[29]:

```
#include <iostream>
#include <math.h>
using namespace std;
```

[30]:

```
int main()
{
    float x, sum, ctr;
    int i, n, m, mm, nn = 0;
    cout << "\n\n Display the sum of the series [ x - x^3 + x^5 + .....]\n";
    cout << "-----\n";
    cout << " Input the value of x: ";
    cin >> x;
    cout << " Input number of terms: ";
    cin >> n;
    sum = x;
    m = -1;
    cout << "The values of series:" << endl;
    cout << sum << endl;
    for (i = 1; i < n; i++) {
        ctr = (2 * i + 1);
```

```

        mm = pow(x, ctr);
        nn = mm * m;
        cout << nn << endl;
        sum = sum + nn;
        m = m * (-1);
    }
    cout << "\n The sum of the series upto " << n << " term is: " << sum << "\n";
}

```

[31]: `main();`

Display the sum of the series [$x - x^3 + x^5 + \dots$]

```

-----
Input the value of x: 6
Input number of terms: 3
The values of series:
6
-216
7776

```

The sum of the series upto 3 term is: 7566

Control Structures

A program is usually not limited to a linear sequence of instructions. During its process it may

Conditional structure: if and else

The if keyword is used to execute a statement or block only if a condition is fulfilled. Its form is:
if (condition) statement

[5]:

```
int x = 10;
if (x == 100)
    cout << "x is 100";
```

We can additionally specify what we want to happen if the condition is not fulfilled by using the keyword else. Its form used in conjunction with if is: if (condition) statement1 else statement2

```
[6]: if (x == 100)
      cout << "x is 100";
      else
      cout << "x is not 100";
```

x is not 100

```
[7]: if (x > 0)
      cout << "x is positive";
      else if (x < 0)
      cout << "x is negative";
      else
      cout << "x is 0";
```

x is positive

Iteration structures (loops)

Loops have as purpose to repeat a statement a certain number of times or while a condition is fulfilled.

```
[15]: // the while loop
      /*
      its format is:
      while (expression) statement
      */

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

```
[16]: int main ()
      {
      int n;
      cout << "Enter the starting number > ";
      cin >> n;
      while (n>0) {
      cout << n << ", ";
      --n;
      }
      cout << "FIRE!\n";
      return 0;
      }
```

```
[17]: main();
```

Enter the starting number >

23

23, 22, 21, 20, 19, 18, 17, 16, 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2,
1, FIRE!

```
[18]: // The Do-While loop
/*
  its format is:

  do statement while (condition);
*/
#include <iostream>
using namespace std;
```

```
[19]: int main ()
{
    unsigned long n;
    do {
        cout << "Enter number (0 to end): ";
        cin >> n;
        cout << "You entered: " << n << "\n";
    } while (n != 0);
    return 0;
}
```

```
[20]: main();
```

Enter number (0 to end): 0
You entered: 0

```
[21]: // The for loop

/*
  Its format is:
  for (initialization; condition; increase) statement;
*/

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

```
[22]: int main ()
{
    for (int n=10; n>0; n--) {
        cout << n << ", ";
    }
    cout << "FIRE!\n";
    return 0;
}
```

```
[23]: main();
```

10, 9, 8, 7, 6, 5, 4, 3, 2, 1, FIRE!

Jump statements.

The break statements

The Continue statement

The goto statement

```
[ ]: // break loop example
#include <iostream>
using namespace std;
```

```
[24]: int main (){

    int n;
    for (n=10; n>0; n--){

        cout << n << ", ";
        if (n==3){
            cout << "countdown aborted!";
            break;
        }
    }

    return 0;

}
```

```
[25]: main();
```

10, 9, 8, 7, 6, 5, 4, 3, countdown aborted!

```
[26]: // continue loop example
#include <iostream>
using namespace std;
```

```
[27]: int main ()
{

    for (int n=10; n>0; n--) {

        if (n==5) continue;
        cout << n << ", ";
    }

    cout << "FIRE!\n";
    return 0;
}
```

goto allows to make an absolute jump to another point in the program.

```
[28]: // goto loop example
#include <iostream>
using namespace std;
```

```
[29]: int main ()
{
    int n=10;
    loop:
    cout << n << ", ";
    n--;
    if (n>0) goto loop;

    cout << "FIRE!\n";
    return 0;
}
```

```
[30]: main();
```

10, 9, 8, 7, 6, 5, 4, 3, 2, 1, FIRE!

The selective structure: switch

```
[31]: switch (x) {
    case 1:
    cout << "x is 1";
    break;
    case 2:
    cout << "x is 2";
    break;
    default:
    cout << "value of x unknown";
}
```

value of x unknown

```
[32]: if (x == 1) {
    cout << "x is 1";
}
else if (x == 2) {
    cout << "x is 2";
}
else {
    cout << "value of x unknown";
}
```

value of x unknown

```
[33]: switch (x) {  
      case 1:  
      case 2:  
      case 3:  
        cout << "x is 1, 2 or 3";  
        break;  
      default:  
        cout << "x is not 1, 2 nor 3";  
    }
```

x is not 1, 2 nor 3

[]:

[]:

[]: