Lecture No.4

2-4 Comparison Operators

Comparison operators are used to compare two values (or variables). This is important in programming, because it helps us to find answers and make decisions.

The return value of a comparison is either 1 or 0, which means true (1) or false (0). These values are known as Boolean values. In the following example, we use the **greater than** operator (>) to find out if 5 is greater than 3:

```
int x = 5;
int y = 3;
cout << (x > y); // returns 1 (true) because 5 is greater than 3
```

A list of all comparison operators:

Operator	Name	Example
==	Equal to	x == y
!=	Not equal	x != y
>	Greater than	x > y
<	Less than	x < y
>=	Greater than or equal to	x >= y
<=	Less than or equal to	x <= y

2-5 Logical Operators

As with comparison operators, you can also test for true (1) or false (0) values with logical operators. Logical operators are used to determine the logic between variables or values:

Operator	Name	Description	Example
&&	Logical and	Returns true if both statements are true	x < 5 && x < 10
||	Logical or	Returns true if one of the statements is true	x < 5 || x < 4
!	Logical not	Reverse the result, returns false if the result is true	!(x < 5 && x < 10)

Exmaple:

```
#include <iostream>
using namespace std;
int main() {
  int x = 5;
  int y = 3;
  cout << (x >= y) <<endl; // returns 1 (true) because 5 is greater than 3
  cout << (x == y) <<endl;
  cout << (x != y) <<endl; // should use (...)
  cout << !(x<5 || y<10) <<endl;
 return 0;
}
```

```
"E:\0.SCIENCE\0.courses\1.programming languages\c++\0 course\
1
0
1
0

Process returned 0 (0x0)   execution time : 0.292 s
Press any key to continue.
```

2-6 Bitwise Operators

C++ supports different types of bitwise operators that can perform operations on integers at bit-level. Supported types of bitwise operators include:

- & Bitwise AND
- | Bitwise OR
- << Bitwise Left Shift
- >> Bitwise Right Shift
- ^ Bitwise XOR
- ~ Bitwise Complement

Exmaple:

```
1.  #include <iostream>
2.  using namespace std;
3.
4.  int main() {
5.   std::cout <<"Bitwise AND:" <<(4 & 5) << "\n";
6.   // Output: 100 (base-2 binary system) = 4   0100 & 0101 =0100
7.
8.   std::cout << "Bitwise OR:"<<(4 | 5) << std::endl;
9.   // Output: 101 (base-2 binary system) = 5    0100 | 0101 =0101
10.
11.  std::cout << "Bitwise Left Shift:"<<(4 << 1) << "\n";
12.  // Output: 1000 = 8  0100 << 1
13.
14.  std::cout <<"Bitwise Left Shift:"<< (4 << 2) << "\n";
15.  // Output: 10000 = 16  0100 << 2
16.
17.  std::cout << "Bitwise RIGHT Shift:"<<(4 >> 1) << "\n";
18.  // Output: 10 = 2    0100 >> 1
19.
20.  std::cout << "Bitwise RIGHT Shift:"<<(4 >> 2) << "\n";
21.  // Output: 1 = 1      0100 >> 2
22.
23.  std::cout << "Bitwise XOR:" <<(12 ^ 10) << "\n";
24.  // Output: 6  1100  XOR 1010 = 0110
25.
26. /*
27.   5 = 0000000000000101 in base-2 binary system
28.  ~5 = 1111111111111010 in base-2 binary system
29.  => -6 in base-10 binary system.
30. why -6 ???
31. because 6=0000000000000110 in base-2 binary system
32. في الحاسب تمثل الأعداد السالبة باستخدام المتمم الثنائي راجع المحاضرة 3 ch1 Number systems
    complement.pptx السلايد رقم 45 أي 1111111111111001=1+1111111111111010 هي -6
33.  unsigned(5) = 00000000000000000000000000000101 in base-2 binary system
34.  ~5 = 11111111111111111111111111111010 in base-2 binary system
35.  => 4294967290 in base-10 binary system
36.  */
```

```
37.
38.  std::cout << "Bitwise Complement:"<<(~(int)5) << "\n";
39.  // Output: 1111111111111010 = -6
40.
41.  std::cout << "Bitwise Complement:"<<(~(unsigned int)5) << "\n";
42.  // Output: 11111111111111111111111111111010 = 4294967290
43.
44.  return 0;
45. }
```

```
"E:\0.SCIENCE\0.courses\1.programming languages\c++\0 course\
Bitwise AND:4
Bitwise OR:5
Bitwise Left Shift:8
Bitwise Left Shift:16
Bitwise RIGHT Shift:2
Bitwise RIGHT Shift:1
Bitwise XOR:6
Bitwise Complement:-6
Bitwise Complement:4294967290

Process returned 0 (0x0)   execution time : 0.260 s
Press any key to continue.
```

2-7 C++ Math

C++ has many functions that allows you to perform mathematical tasks on numbers. For example: Max and min. The max(x,y) function can be used to find the highest value of x and y:

```
cout << max(5, 10);
```

C++ <cmath> Header: Other functions, such as sqrt (square root), round (rounds a number) and log (natural logarithm), can be found in the <cmath> header file:

```
// Include the cmath library
#include <cmath>
cout << sqrt(64);
cout << round(2.6);
cout << log(2);
```

Other Math Functions: A list of other popular Math functions (from the <cmath> library) can be found in the table below:

Function	Description
abs(x)	Returns the absolute value of x
acos(x)	Returns the arccosine of x
asin(x)	Returns the arcsine of x
atan(x)	Returns the arctangent of x
cbrt(x)	Returns the cube root of x
ceil(x)	Returns the value of x rounded up to its nearest integer
cos(x)	Returns the cosine of x
cosh(x)	Returns the hyperbolic cosine of x
exp(x)	Returns the value of E^x
expm1(x)	Returns e^x -1
fabs(x)	Returns the absolute value of a floating x
fdim(x, y)	Returns the positive difference between x and y
floor(x)	Returns the value of x rounded down to its nearest integer
hypot(x, y)	Returns sqrt(x^2 +y^2) without intermediate overflow or underflow
fma(x, y, z)	Returns x*y+z without losing precision
fmax(x, y)	Returns the highest value of a floating x and y
fmin(x, y)	Returns the lowest value of a floating x and y
fmod(x, y)	Returns the floating point remainder of x/y
pow(x, y)	Returns the value of x to the power of y
sin(x)	Returns the sine of x (x is in radians)
sinh(x)	Returns the hyperbolic sine of a double value
tan(x)	Returns the tangent of an angle
tanh(x)	Returns the hyperbolic tangent of a double value

Exmaple: اكتب برنامج يقوم بحساب العملية الحسابية التالية

$$z = \sqrt{\frac{x^2 + y^2}{2x}}$$

```
1. #include <iostream>
2. #include <cmath>
```

```
using namespace std;

int main()
{
    int x,y;
    float z;
    cin>>x>>y;
    z= sqrt((pow(x,2)+pow(y,2))/(2*x));
    cout<<z;
    return 0;
}
```

```
"E:\0.SCIENCE\0.courses\1.programming languages\c++\0 course\0.
4
5
2.26385
Process returned 0 (0x0)   execution time : 2.286 s
Press any key to continue.
```

Example: صب المعطيات والدقة الحسابية

```
#include <iostream>

using namespace std;

int main()

{

int a,b,c;

float a1,b1,c1;

cout <<a<<" "<<b<<" "<<c<<endl;

cout << "enter a,b (int):"<<endl;

cin>>a>>b;  //5   6


c=a/b;

cout <<endl<<"===>int c= a/b="<<c<<endl;     //0


c1=float(a/b);
```

```
15734729 135 4200782
```

```
cout <<endl<<"===>float c1= a/b="<<c1<<endl;  //0


cout << "enter a1,b1 (float):"<<endl;

cin>>a1>>b1;//5   6

c1=a1/b1;

cout <<endl<<"===>float c1= a1/b1="<<c1;  //0.833333


return 0;

}
```

```
"E:\0.SCIENCE\0.courses\1.programming languages\c++\0 course
15734729 135 4200782
enter a,b (int):
5
6

===>int c= a/b=0

===>float c1= a/b=0
enter a1,b1 (float):
5
6

===>float c1= a1/b1=0.833333
Process returned 0 (0x0)   execution time : 8.929 s
Press any key to continue.
```

Example صب المعطيات والدقة الحسابية :

```
#include <iostream>

using namespace std;

int main ( )

{

int i , j ;

float x,y,z;

i = 5/2;

x = 5/2 ;
```

```
y = (float ) (5.0/2);  //or   y = (float ) 5.0/2;

j = ( float) 5.0/2;   //or   j = ( float) (5.0/2);


z=5/2;

//z = 5. / 2 ;

cout<<i <<endl<< x<<endl << y<<endl << j<<endl << z<<endl<<"-----"<<endl;

//   2       2       2.5      2       2

cout<<3/2/3.0/4<<endl<<"-------"<<endl;  //   1/3.0/4=0.3333/4=0.083333

cout<<3/2/3/4<<endl<<"-------"<<endl;    //   1/3/4=0/4=0

cout<<1/2/3.0/4<<endl<<"-------"<<endl;  //   0/3.0/4=0

cout<<1.0/2/3.0/4<<endl<<"-------"<<endl; //   0.5/3.0/4=0.16667/4=0.04166667

return 0;

}
```

```
"E:\0.SCIENCE\0.courses\1.programming languages\c++\0 course\
2
2
2.5
2
2
-----
0.0833333
-------
0
-------
0
-------
0.0416667
-------

Process returned 0 (0x0)   execution time : 0.053 s
Press any key to continue.
```

Exmaple:

```
#include <iostream>
```

```
using namespace std;


int main()

{

   cerr << "error" << endl;

   return 0;

}
```

```
"F:\essa\mechatronic\mecha\work\c++\١٠\٢\٢٠١٢ الاطلاعطو
error

Process returned 0 (0x0)   execution time : 0.281 s
Press any key to continue.
```

Example ما هو خرج البرنامج التالي

```
#include <iostream>

using namespace std;

int main()

{

        int a=5;

        double s;

        s=3*++a;

        cout<<s<<" "<<a;

        //system("PAUSE");

        return 0;

}
```

```
"E:\0.SCIENCE\0.courses\1.programming languages\c++\0 course\
18  6
Process returned 0 (0x0)   execution time : 0.294 s
Press any key to continue.
```

Part 3- C++ Control Structures

3-1 C++ Boolean Expressions

Boolean Values: A boolean variable is declared with the bool keyword and can only take the values true or false:

```
bool isCodingFun = true;
bool isFishTasty = false;
cout << isCodingFun;  // Outputs 1 (true)
cout << isFishTasty;  // Outputs 0 (false)
```

A **Boolean expression** returns a boolean value that is either 1 (true) or 0 (false). This is useful to build logic, and find answers. You can use a comparison operator, such as the **greater than** (>) operator, to find out if an expression (or variable) is true or false:

```
int x = 10;
int y = 9;
cout << (x > y); // returns 1 (true), because 10 is higher than 9

cout << (10 > 9); // returns 1 (true), because 10 is higher than 9

int x = 10;
cout << (x == 10);  // returns 1 (true), because the value of x is equal to 10

cout << (10 == 15);  // returns 0 (false), because 10 is not equal to 15

int myAge = 25;
int votingAge = 18;
cout << (myAge >= votingAge); // returns 1 (true), meaning 25 year olds are allowed to vote!
```

An even better approach, would be to wrap the code above in an if...else statement, so we can perform different actions depending on the result:

3-2 C++ Conditions and If Statements

you already know that C++ supports the usual logical conditions from mathematics:

- Less than: a < b
- Less than or equal to: a <= b

- Greater than: a > b
- Greater than or equal to: a >= b
- Equal to a == b
- Not Equal to: a != b

You can use these conditions to perform different actions for different decisions.

C++ has the following conditional statements:

- Use if to specify a block of code to be executed, if a specified condition is true
- Use else to specify a block of code to be executed, if the same condition is false
- Use else if to specify a new condition to test, if the first condition is false
- Use switch to specify many alternative blocks of code to be executed

3-3 The if Statement

Use the if statement to specify a block of C++ code to be executed if a condition is true.

Syntax:

```
if (condition) {
  // block of code to be executed if the condition is true
}
```

Note that if is in lowercase letters. Uppercase letters (If or IF) will generate an error.

In the example below, we test two values to find out if 20 is greater than 18. If the condition is true, print some text:

Example

```
if (20 > 18) {
  cout << "20 is greater than 18";
}
```

Example

```
int x = 20;
int y = 18;
if (x > y) {
  cout << "x is greater than y";
}
```

3-4 The else Statement:

Use the else statement to specify a block of code to be executed if the condition is false.

Syntax:

```
if (condition) {
  // block of code to be executed if the condition is true
}else{
 // block of code to be executed if the condition is false
}
```

Example:

```
int time = 20;
if (time < 18) {
  cout << "Good day.";
} else {
  cout << "Good evening.";
}
// Outputs "Good evening."
```

Output "Old enough to vote!" if myAge is **greater than or equal to** 18. Otherwise output "Not old enough to vote.":

```
int myAge = 25;
int votingAge = 18;

if (myAge >= votingAge) {
  cout << "Old enough to vote!";
} else {
  cout << "Not old enough to vote.";
}

// Outputs: Old enough to vote!
```

3-5 The else if Statement

Use the else if statement to specify a new condition if the first condition is false.

Syntax:

```
if (condition1) {
  // block of code to be executed if condition1 is true
}else if (condition2){
 // block of code to be executed if the condition1 is false and
condition2 is true
```

```
}else{
 // block of code to be executed if the condition1 is false and
condition2 is false
}
```

Example

```
int time = 22;
if (time < 10) {
  cout << "Good morning.";
} else if (time < 20) {
  cout << "Good day.";
} else {
  cout << "Good evening.";
}
// Outputs "Good evening."
```

3-6 Short Hand If...Else (Ternary Operator)

There is also a short-hand if else, which is known as the **ternary operator** because it consists of three operands. It can be used to replace multiple lines of code with a single line. It is often used to replace simple if else statements:

Syntax

```
variable = (condition) ? expressionTrue : expressionFalse;
```

Instead of writing:

```
int time = 20;
if (time < 18) {
  cout << "Good day.";
} else {
  cout << "Good evening.";
}
```

You can simply write:

```
int time = 20;
string result = (time < 18) ? "Good day." : "Good evening.";
cout << result;
```

Exmaple:

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

int main() {
        int x ;
        cin>>x;
```

```
    if (x > 0)
            cout << "x is positive";
        else if (x < 0)
            cout << "x is negative";
        else
            cout << "x is 0";
    return 0;
}
```

```
"E:\0.SCIENCE\0.courses\1.programming languages\c++\0 course\0
-3
x is negative
Process returned 0 (0x0)   execution time : 2.987 s
Press any key to continue.
```

انتهت المحاضرة