Programming Fundamentals

By/ Merhan Atef

Section (2) Outline

1. How to represent binary number in c++ ?
2. C++ Numeric Suffixes
3. SizeOf
4. float and double
5. Arithmetic Operators
6. Increment and Decrement
7. Examples

How to represent binary number in c++ ?

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

int main() {
    int num = 10;
    cout << "Binary representation: " << bitset<8>(num) << endl;
    return 0;
}
```

C++ Numeric Suffixes

- Integer literals can have suffixes to specify their type explicitly. These suffixes help handle large values and unsigned numbers.

Suffix	Type	Example
u / U	Unsigned int	5u, 100U
l / L	Long int	5000l, 6000L
ll / LL	Long long int	1000000ll, 9000LL
ul / UL	Unsigned long	4000ul, 7000UL
ull / ULL	Unsigned long long	9000000ull, 1200ULL
f / F	**Float**	**3.14f, 2.0F**
l / L	Long double	3.14159L, 2.5l

The sizeof operator can be used to determine to the number of bytes occupied in memory by a variable of a certain type.

```
#include<iostream>
using namespace std;
void main()
{
cout << "sizof(integer)is:" << sizeof(int) << "byte" << endl;
cout << "sizof(float)is:" << sizeof(float) << "byte" << endl;
cout << "sizof(character)is:" << sizeof(char) << "byte" <<
endl;
system("Pause");
}
```

float and double

- Float

- 32 bit (4 byte)

Sign (1 bit)	**Exponent (8 bit)**	**Mantissa (23 bit)**

- Double

- 64 bit (8 byte)

Sign (1 bit)	**Exponent (11 bit)**	**Mantissa (52 bit)**

float and double(Example)

- **Represent the Float number 95.145**

1- Convert 95 to binary → 01011111

2- Convert .145 to binary → 00100101000111101

The number is → 1011111.00100101000111101

3- Write the number in this form → $1.01111100100101000111101 * 2^6$

4- Number is positive so, first bit is 0.

5- For Float → Add 127 to exponent (in this example, 6+127= 133)

For Double → Add 1023 to exponent.

0	**10000101**	**01111100100101000111101**

Operators

There are four main classes of operators:

- Arithmetic

- Relational

- Logical

- Bitwise.

Arithmetic Operators

Operator	Action
–	Subtraction, also unary minus
+	Addition
*	Multiplication
/	Division
%	Modulus
– –	Decrement
++	Increment

Arithmetic Operators

- When you apply / to an integer any remainder will be truncated. For example, 5/2 will equal 2 in integer division.
- The modulus operator yielding the remainder of an integer division.

```
#include<iostream>
using namespace std;
void main()
{
int x, y;
x = 5;
y = 2;
cout << x << "/" << y << "="<<x/y<<endl;
cout << x << "%" << y << "=" << x%y << endl;
system("Pause");
}
```

Example

```
#include<iostream>
using namespace std;
void main()
{
int i = 5, j = 6, k = 7, n = 3;
cout << i+j*k-k%n<< endl;
system("Pause");
}
```

Example

```
#include<iostream>
using namespace std;
void main()
{
cout << 3.0 * 7.0 - 6.0 + 2.0 * 5.0 / 4.0 + 6.0 << endl;
cout << (5 + 4) * 2 - 6 / 2 << endl;
cout << 4 % 3 * 20 / 2 << endl;
cout << (8 + 6) / 2 * 3 % 4 + 6 << endl;
system("Pause");
}
```

Example

```cpp
#include<iostream>
using namespace std;
void main()
{
cout << "hello" << " world, " << "again!" << endl;
cout <<"hello,"<<endl<<"one more time."<<endl<<5<<4<<3<<" "<<2.2<<" "<<1.1<<endl;
system("Pause");
}
```

Increment and Decrement

- The operator ++ adds 1 to its operand, and - - subtracts 1.
- Both the increment and decrement operators may either **prefix** or **postfix** the operand.

x=x+1 → x++ or ++x

x=x-1 → x-- or --x

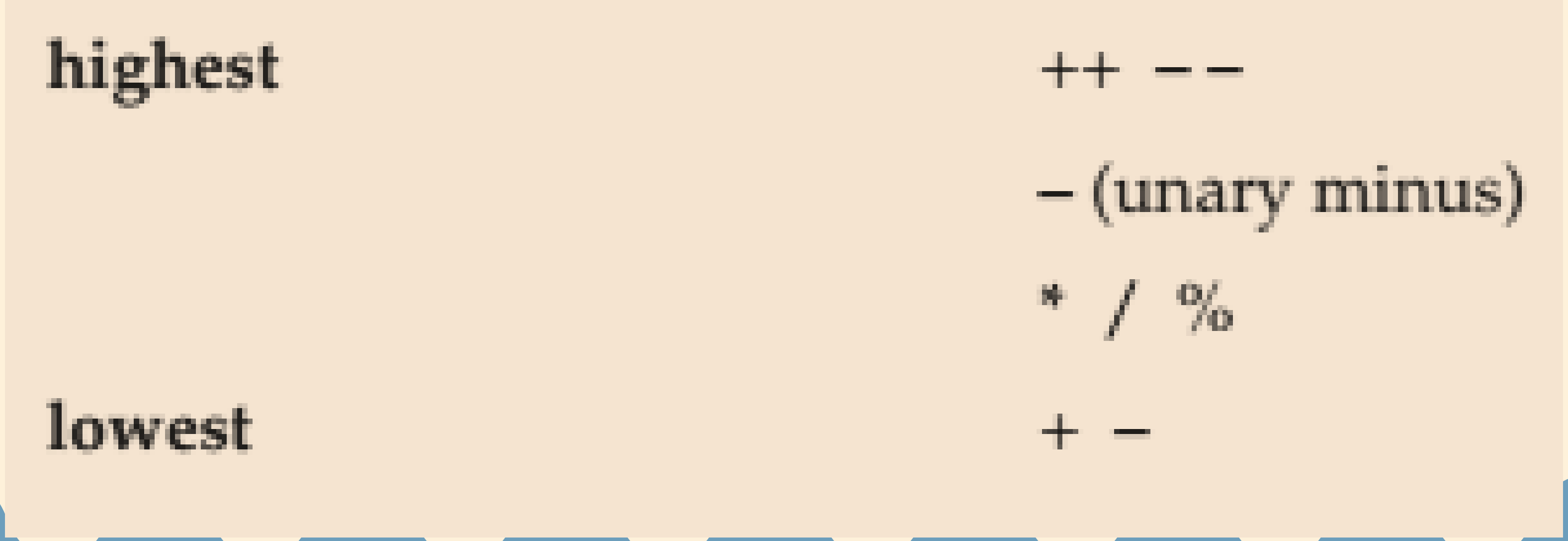
- When an increment or decrement operator precedes its operand, the increment or decrement operation is performed before obtaining the value of the operand for use in the expression. If the operator follows its operand, the value of the operand is obtained before incrementing or decrementing it

Increment and Decrement

x = 10;
y = ++x;
	x = 10;
	y = x++;
		x=10;
		y=--x;
			x=10;
			y=x--;

sets y to 11	sets y to 10	Set y to 9	Set y to 10;

The precedence of the arithmetic operators:



highest	++ – –
	– (unary minus)
	* / %
lowest	+ –

Example

```cpp
#include<iostream>
using namespace std;
void main()
{
int x = 5;
int y = 10;
int z = ++x * y-- + 10;
cout <<"z= "<< z << endl;
cout << "x= " << x << endl;
cout << "y= " << y << endl;
system("Pause");
}
```

```cpp
#include<iostream>
using namespace std;
void main()
{
int x = 5;
int y = 10;
int z = ++x * --y + 10;
cout << "z= " << z << endl;
cout << "x= " << x << endl;
cout << "y= " << y << endl;
system("Pause");
}
```

Show Output

```
#include<iostream>
using namespace std;
void main()
{
int x = 4;
int y = 3;
cout<< y++  *  x-- <<endl;
cout<< ++y  *  x-- << endl;
cout<< ++y  * --x << endl;
system("Pause");
}
```

Example

```
#include<iostream>
using namespace std;
void main()
{
int x = 30;
int y = 40;
int z = ++x * --y + x++ - y-- * 10;
cout << "z= " << z << endl;
cout << "x= " << x << endl;
cout << "y= " << y << endl;
system("Pause");
}
```

Do you have any questions ?

Thank you

piece of advice

Aim for the impossible and don’t stop until you’ve made it possible