When evaluating expressions, post-increment (x++) and post-decrement (x--) operators return their current value and then apply the operators. However, when using pre-increment (++x) and pre-decrement (--x) operators, the operator is applied to the variable prior to returning the final value.

In Listing 2-2, the unary variable is initialized to zero. When the pre-increment (++x) operator is used, unary is incremented to 1 and the value 1 is assigned to the preIncrement variable. The pre-decrement (--x) operator turns unary back to a 0 and then assigns the value to the preDecrement variable.

When the post-decrement (x--) operator is used, the value of unary, 0, is placed into the postDecrement variable and then unary is decremented to -1. Next the post-increment (x++) operator moves the current value of unary, -1, to the postIncrement variable and then increments unary to 0.

While the (~) operator works by flipping bits, the logical negation operator (!) is a logical operator that works on bool values, changing true to false or false to true. In the case of the logNot variable in Listing 2-2, the value is initialized to false, and the next line applies the logical negation operator, (!), which returns true and reassigns the new value, true, to logNot. Essentially, it is toggling the value of the bool variable, logNot.

        int unary = 0;  
        int preIncrement;  
        int preDecrement;  
        int postIncrement;  
        int postDecrement;        preIncrement = ++unary;  
        Console.WriteLine("pre-Increment: {0}", preIncrement);        preDecrement = --unary;  
        Console.WriteLine("pre-Decrement: {0}", preDecrement);  
  
        postDecrement = unary--;  
        Console.WriteLine("Post-Decrement: {0}", postDecrement);  
  
        postIncrement = unary++;  
        Console.WriteLine("Post-Increment: {0}", postIncrement);  
  
        Console.WriteLine("Final Value of Unary: {0}", unary);

pre-Increment: 1  
 pre-Decrement 0  
 Post-Decrement: 0  
 Post-Increment: -1  
 Final Value of Unary: 0

int t = 5;

Console.WriteLine(t++);

Console.WriteLine(t);

t = 5;

Console.WriteLine(++t);

Console.WriteLine(t);

Console.WriteLine("----");

t = 5;

Console.WriteLine(t--);

Console.WriteLine(t);

t = 5;

Console.WriteLine(--t);

Console.WriteLine(t);



To find a sum of primary numbers from 1 to 100.

int S = 6; bool z = false;

for (int k = 4; k <= 100; k++)

{

int i = 2;

while (i <= k / 2)

{

if (k % i == 0)

{

z = true;

break;

}

i++;

}

if (z == false)

{

S += k;

}

z = false;

}

Console.WriteLine(S);

**Continue statement**

>a

double a = int.Parse(Console.ReadLine());

double n, sum = 0, denominator = 0; int i=0;

while (sum<a)

{

If(i%2==1)

* denominator += Math.Sin(i);
* else denominator -= Math.Sin(i);

if(denominator ==0) continue;

else

{

sum +=1/ denominator;

}

i++;

}

Console.WriteLine("i="+i);

Remainder – остаток от деления

целая часть integer part