Programming in c++
Lecture_2

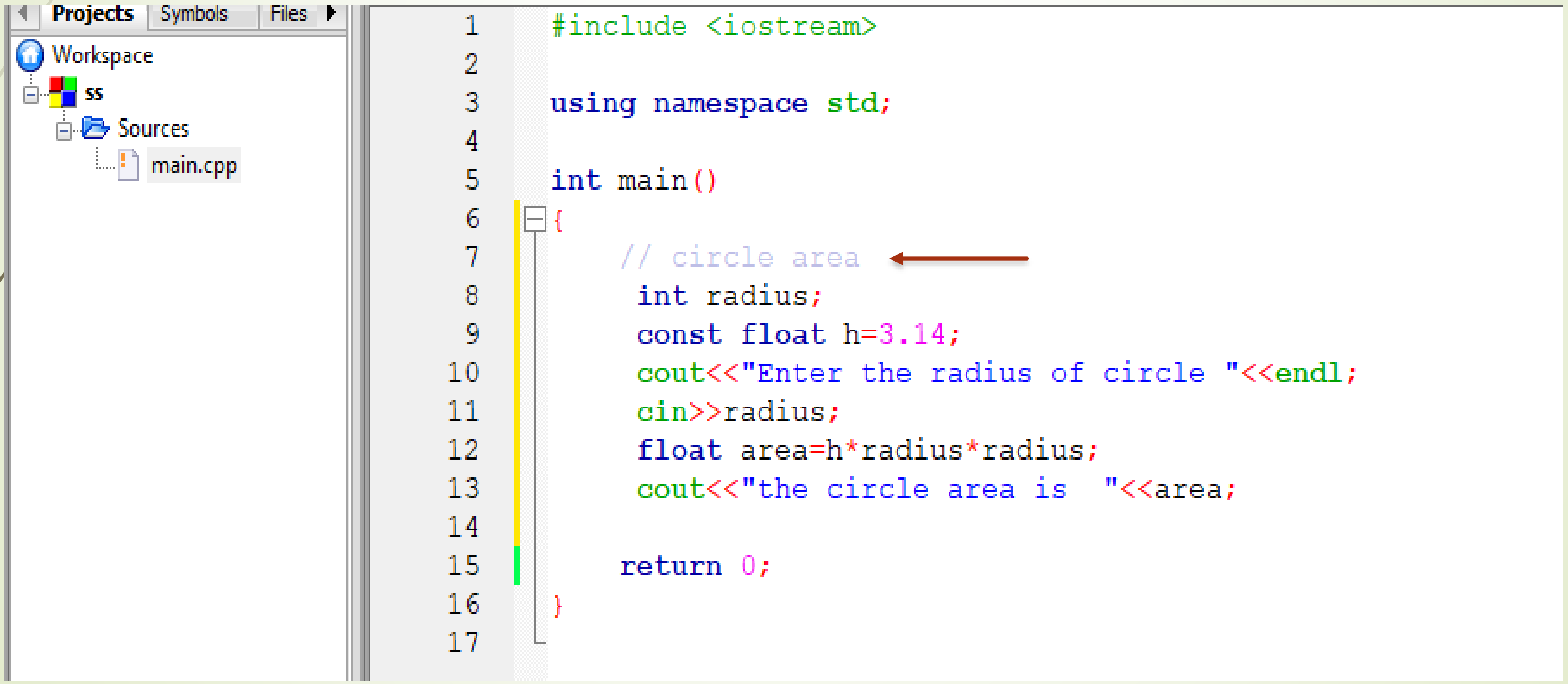
Dr. Sara Mohamed

C++ Comments

- **Comments** can be used to explain C++ code
- to make it more readable. It can also be used to prevent execution when testing alternative code.
- Comments can be **singled-lined or multi-lined.**

Single-line comments

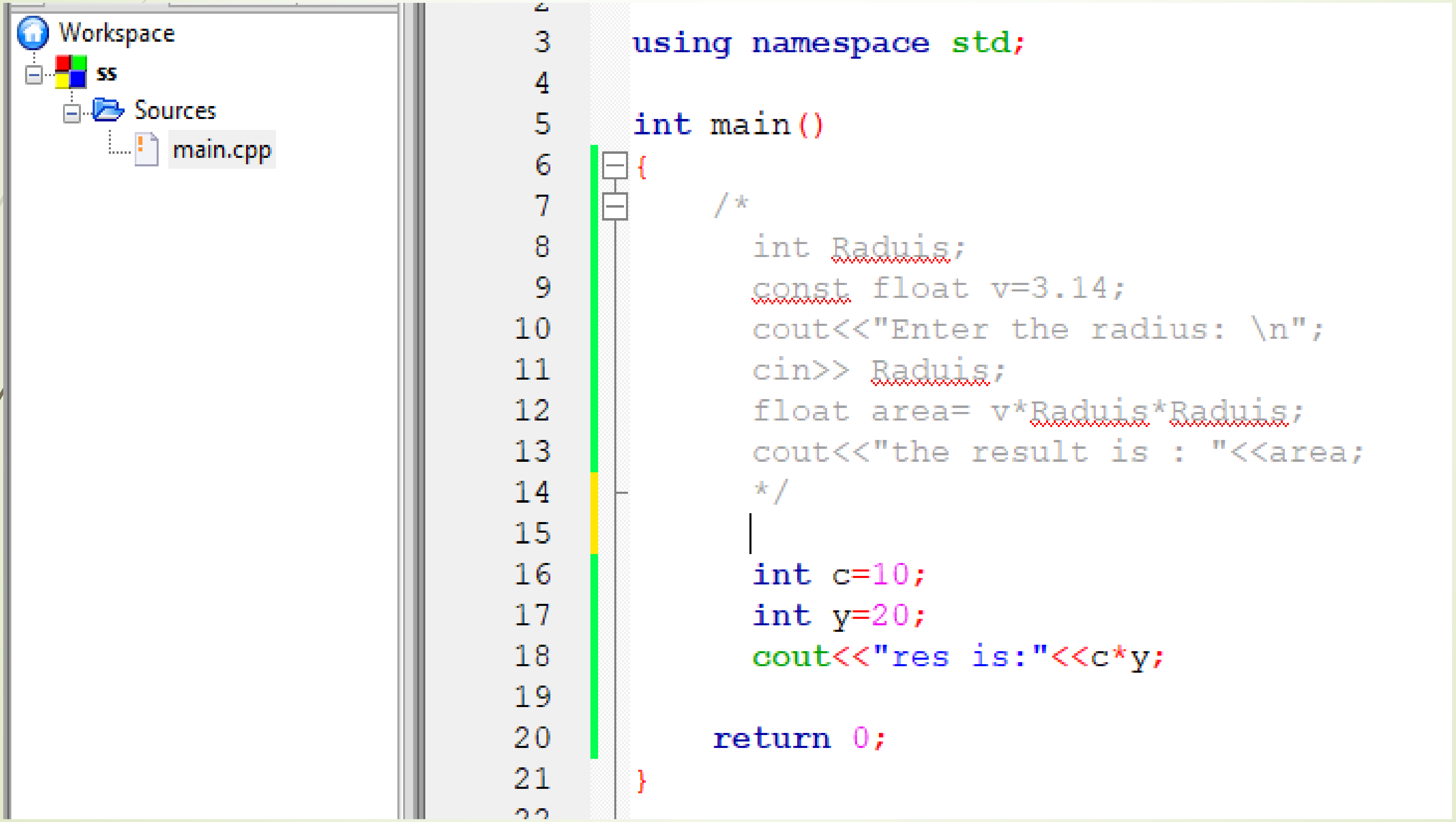
- **Single-line** comments start with two forward slashes (//).
- Any text between // and the end of the line is ignored by the compiler (will not be executed)



```cpp
#include <iostream>

using namespace std;

int main()
{
    // circle area
     int radius;
     const float h=3.14;
     cout<<"Enter the radius of circle "<<endl;
     cin>>radius;
     float area=h*radius*radius;
     cout<<"the circle area is  "<<area;

    return 0;
}

```

C++ Multi-line Comments

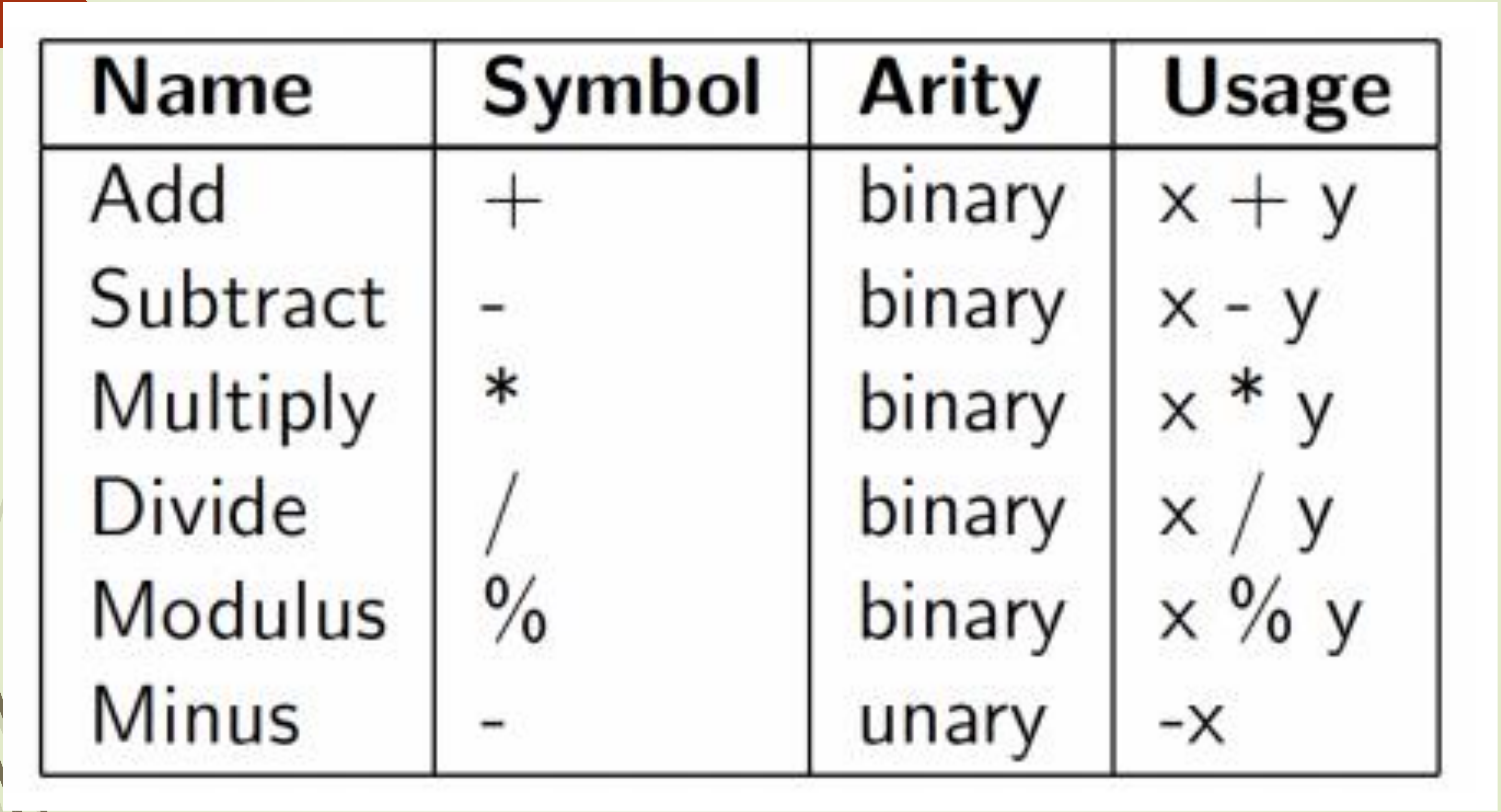
Multi-line comments start with /* and ends with */.
Any text between /* and */ will be ignored by the compiler.



```
using namespace std;

int main()
{
    /*
      int Raduis;
      const float v=3.14;
      cout<<"Enter the radius: \n";
      cin>> Raduis;
      float area= v*Raduis*Raduis;
      cout<<"the result is : "<<area;
      */

      int c=10;
      int y=20;
      cout<<"res is:"<<c*y;

    return 0;
}
```

C++ Operators

- Operators are used to perform operations on variables and values.
- C++ divides the operators into the following groups:
  - **Arithmetic operators**
  - **Assignment operators**
  - **Comparison operators**
  - **Logical operators**



Name	Symbol	Arity	Usage
Add	+	binary	x + y
Subtract	-	binary	x - y
Multiply	*	binary	x * y
Divide	/	binary	x / y
Modulus	%	binary	x % y
Minus	-	unary	-x

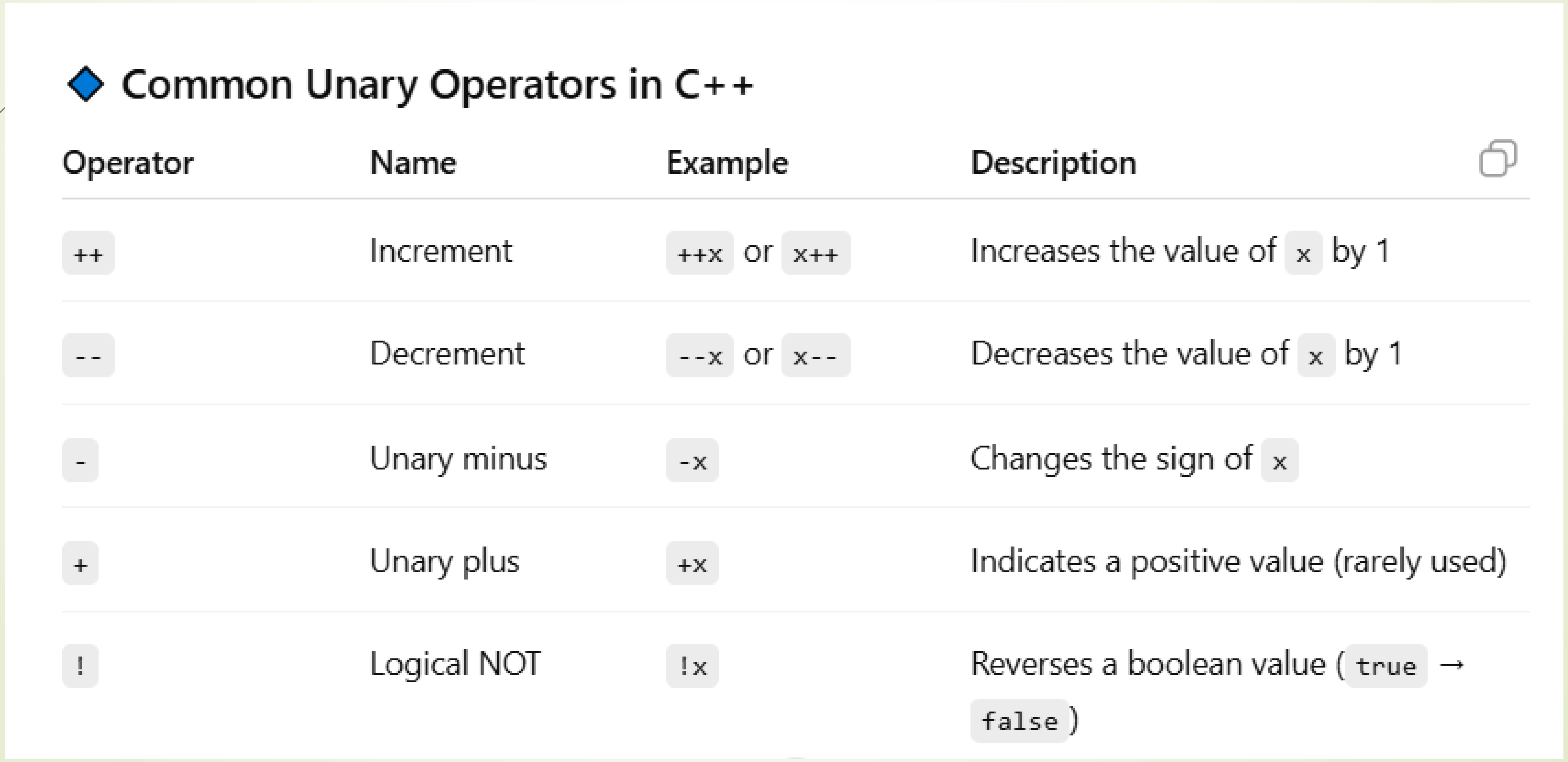
Operators

- Special built-in symbols that have functionality, and work on operands
- **operand** - an input to an operator
- **Arity** - how many operands an operator takes
  - *unary operator* - has one operand
  - *binary operator* - has two operands
  - *ternary operator* - has three operands
- Examples:

```
int x, y = 5, z;
z = 10; // assignment operator (binary)
x = y + z; // addition (binary operator)
x = -y; // -y is a unary operation (negation)
x++; // unary (increment)
```

Unary operators perform an operation using **a single variable or value**.

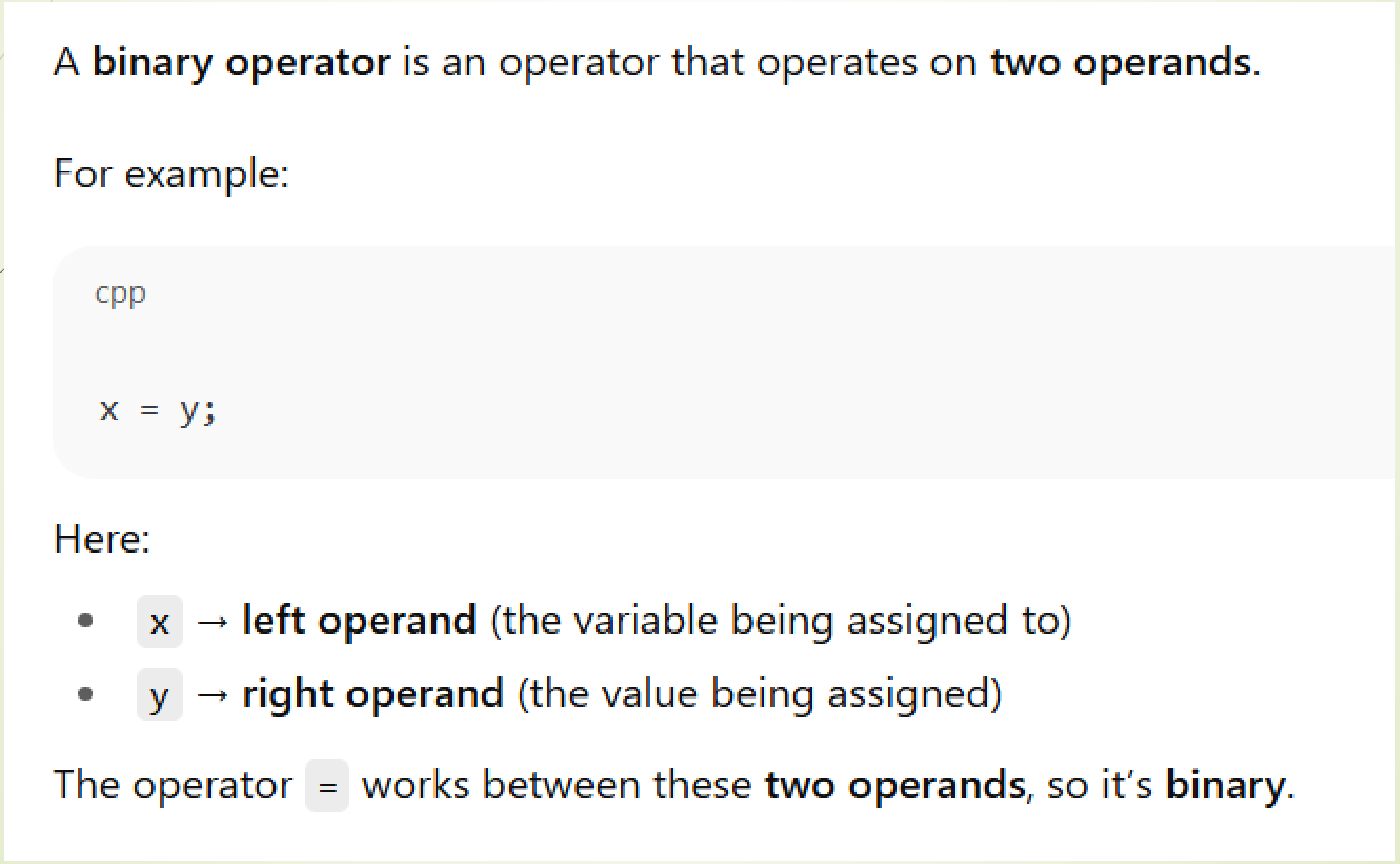
- A **unary operator** in C++ is an operator that works on **only one operand**.



🔷 Common Unary Operators in C++

Operator	Name	Example	Description
`++`	Increment	`++x` or `x++`	Increases the value of `x` by 1
`--`	Decrement	`--x` or `x--`	Decreases the value of `x` by 1
`-`	Unary minus	`-x`	Changes the sign of `x`
`+`	Unary plus	`+x`	Indicates a positive value (rarely used)
`!`	Logical NOT	`!x`	Reverses a boolean value (`true` → `false`)

The operator = works between these **two operands**, so it's **binary**.



A **binary operator** is an operator that operates on **two operands**.

For example:

```cpp
x = y;
```

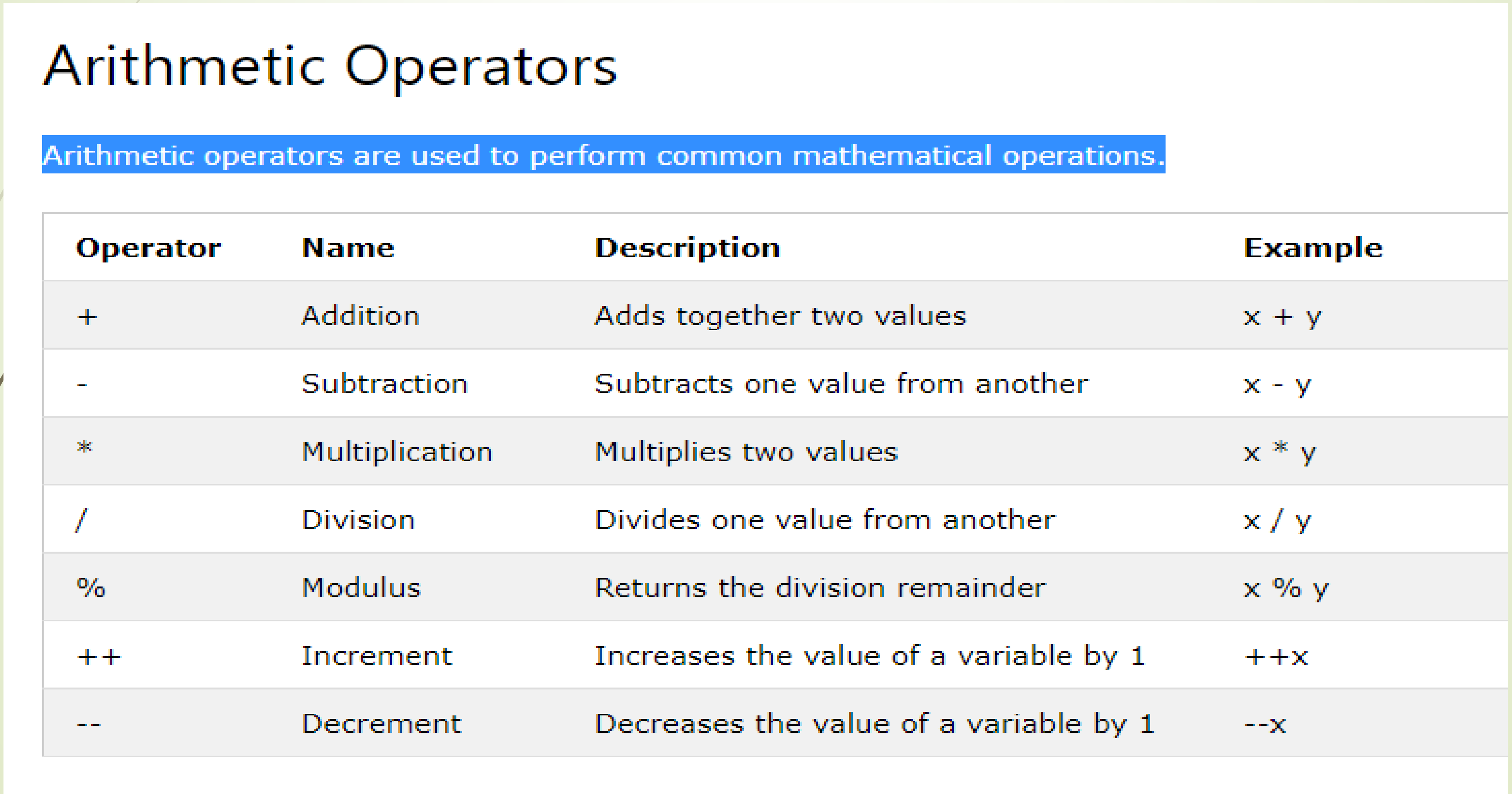
Here:

- `x` → **left operand** (the variable being assigned to)
- `y` → **right operand** (the value being assigned)

The operator `=` works between these **two operands**, so it's **binary**.

Arithmetic Operators

- Arithmetic operators are used to perform common mathematical operations.



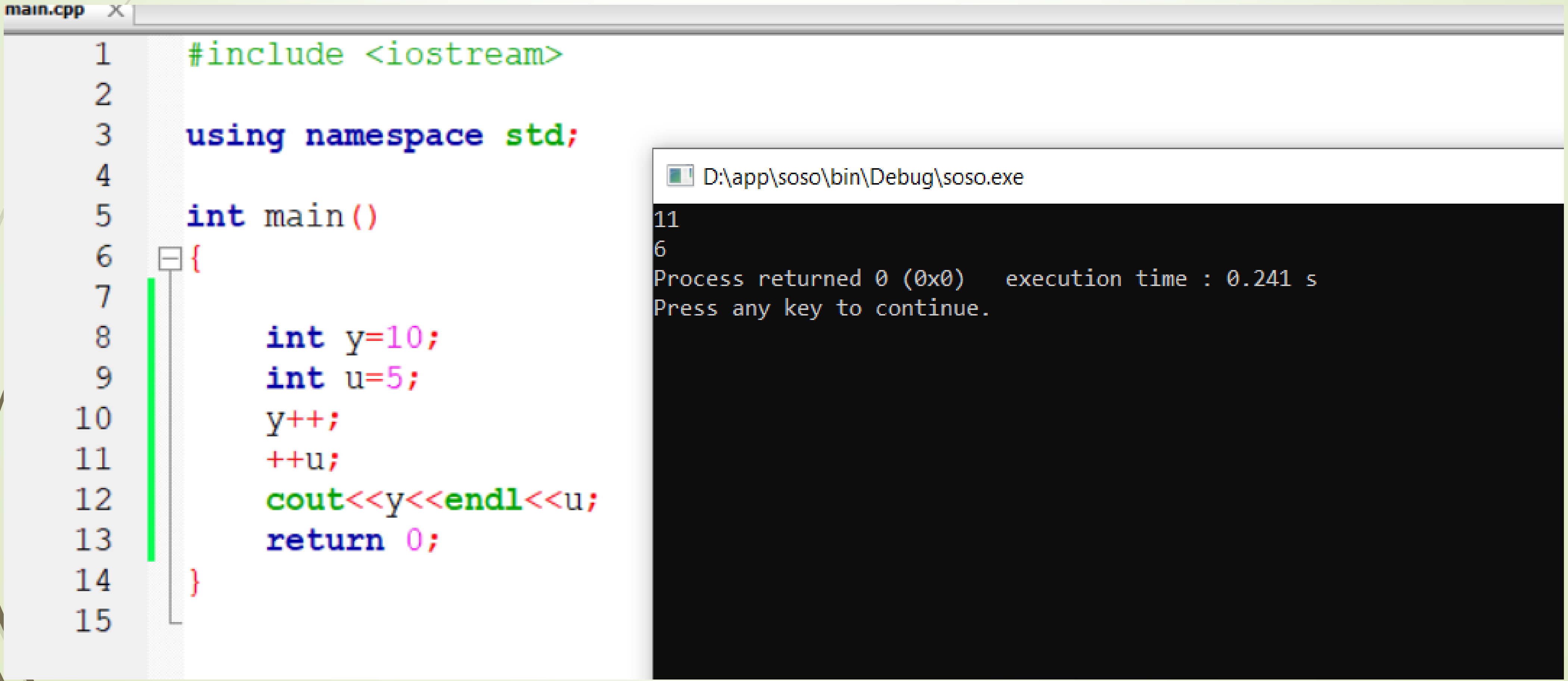
Arithmetic Operators

Arithmetic operators are used to perform common mathematical operations.

Operator	Name	Description	Example
+	Addition	Adds together two values	x + y
-	Subtraction	Subtracts one value from another	x - y
*	Multiplication	Multiplies two values	x * y
/	Division	Divides one value from another	x / y
%	Modulus	Returns the division remainder	x % y
++	Increment	Increases the value of a variable by 1	++x
--	Decrement	Decreases the value of a variable by 1	--x

Increment and decrement

- **Increment** : increase the variable value by 1



main.cpp

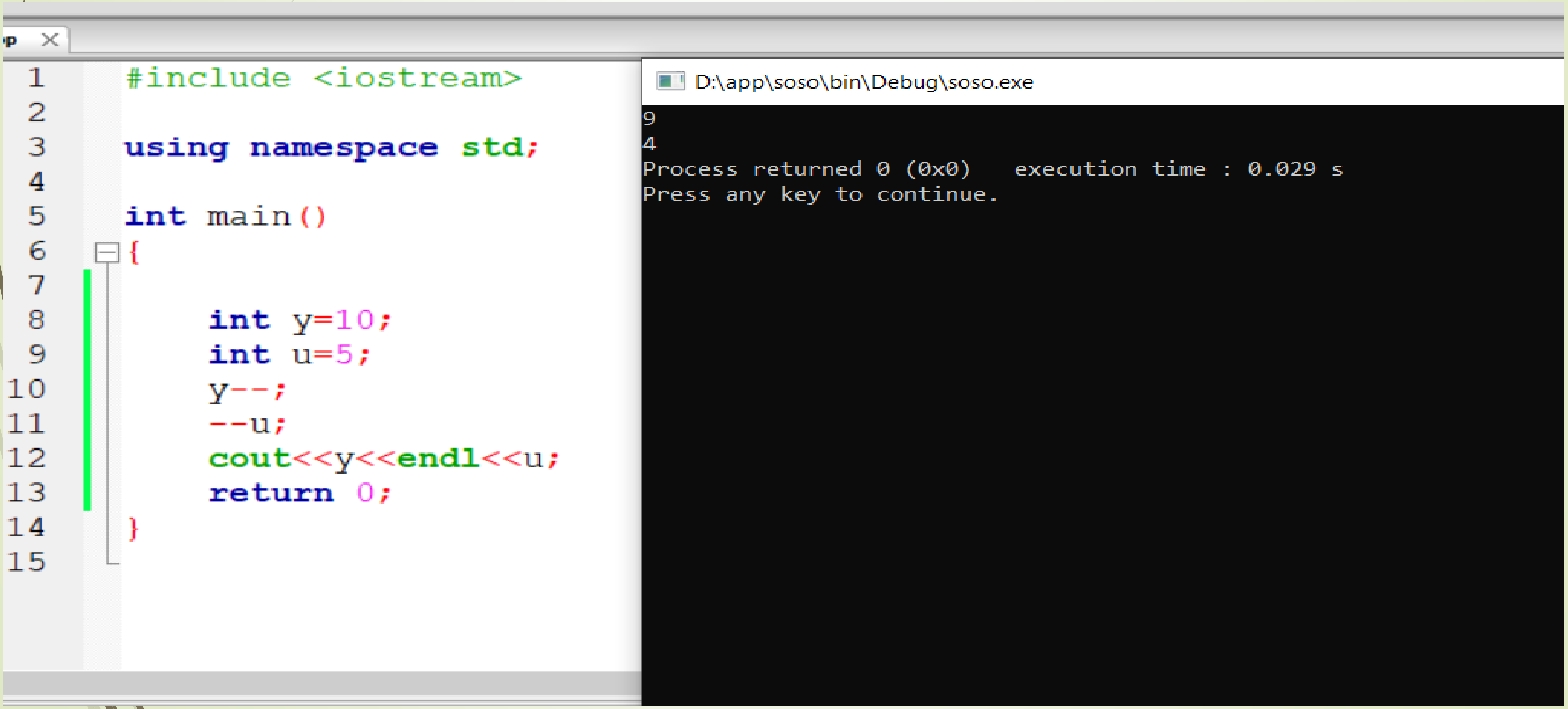
```cpp
#include <iostream>

using namespace std;

int main()
{

    int y=10;
    int u=5;
    y++;
    ++u;
    cout<<y<<endl<<u;
    return 0;
}
```

D:\app\soso\bin\Debug\soso.exe

```
11
6
Process returned 0 (0x0)   execution time : 0.241 s
Press any key to continue.
```

Decrement

- **Decrement** : decrease the variable value by 1



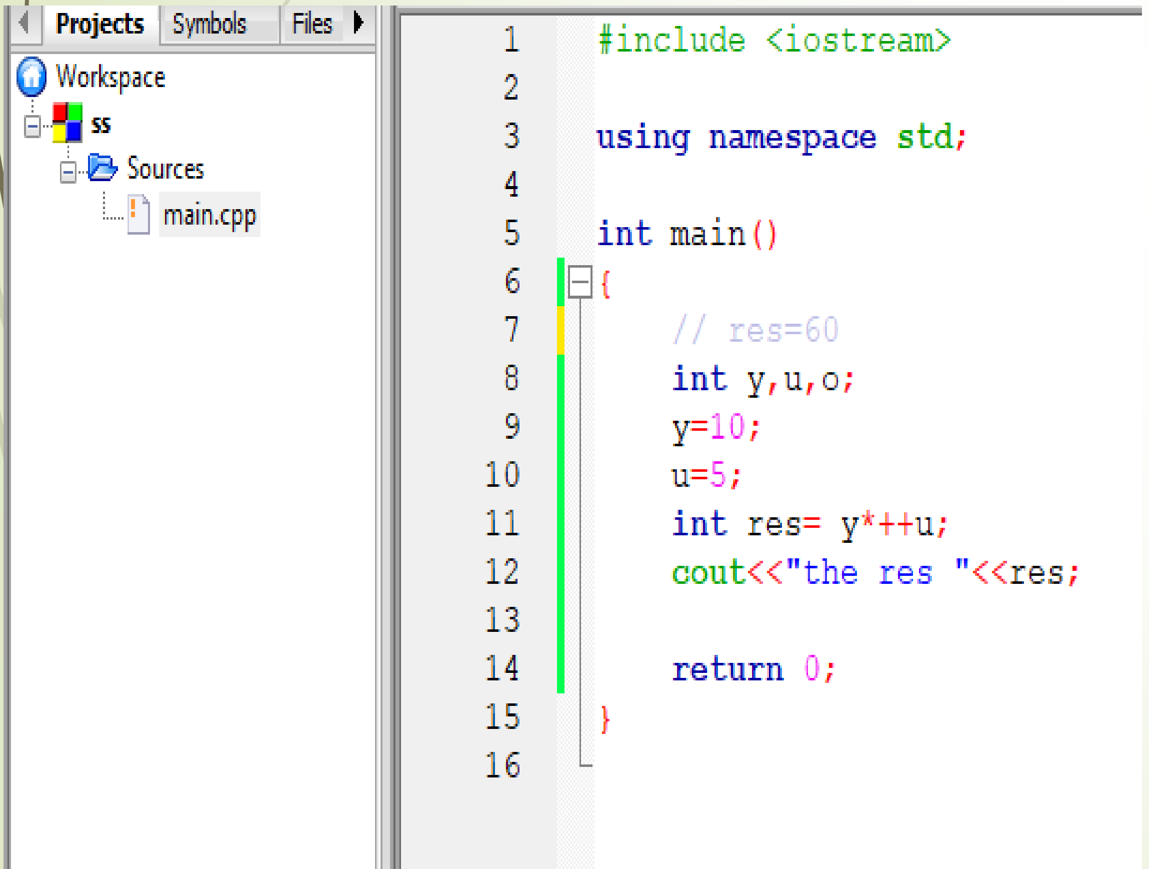
```cpp
#include <iostream>

using namespace std;

int main()
{

    int y=10;
    int u=5;
    y--;
    --u;
    cout<<y<<endl<<u;
    return 0;
}

```

D:\app\soso\bin\Debug\soso.exe

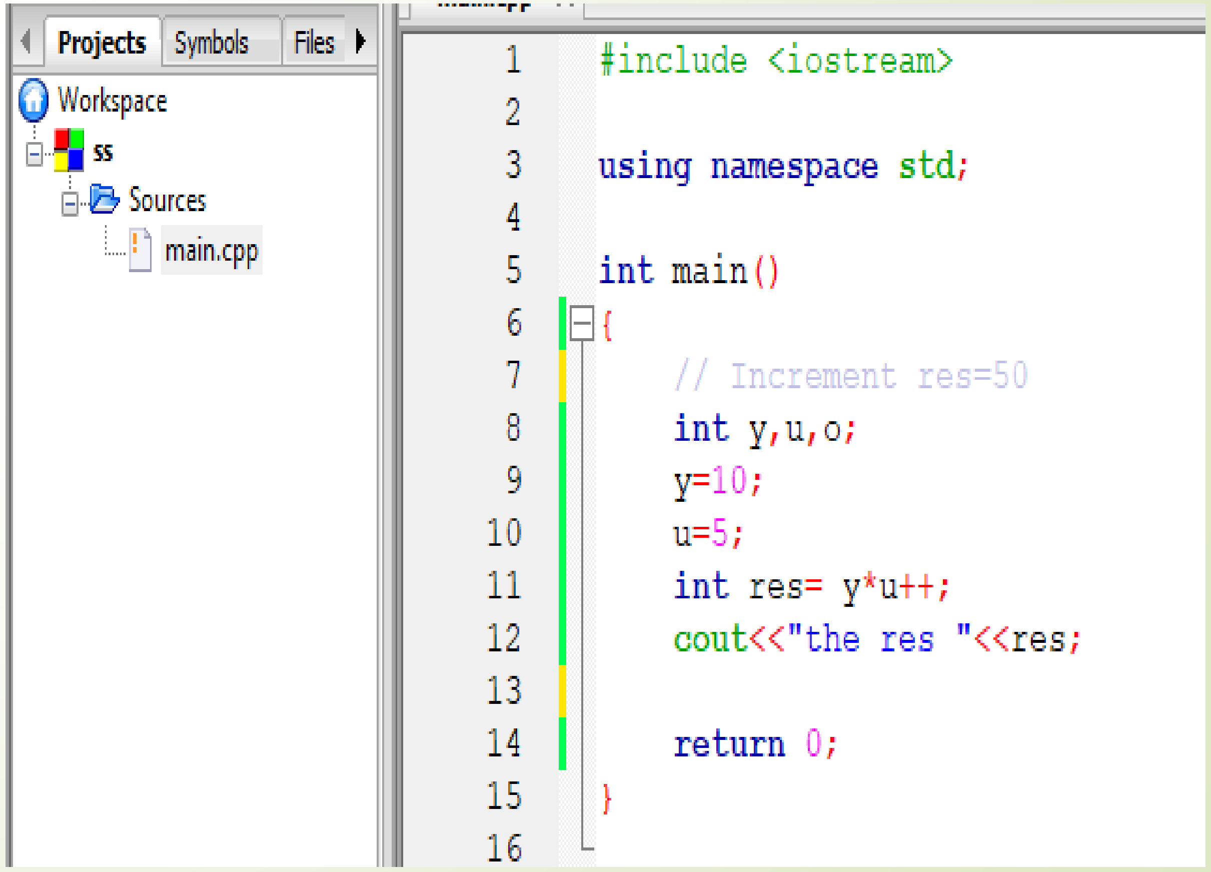
```
9
4
Process returned 0 (0x0)   execution time : 0.029 s
Press any key to continue.
```

Difference between ++x and x++

both ++x and x++ are used to increment variable x by 1.. the prime difference is that: ++x i.e. **pre-increment** operator uses the principle 'change-then-use' while, x++ i.e. **post-increment** operator uses the principle 'use-then-change'.

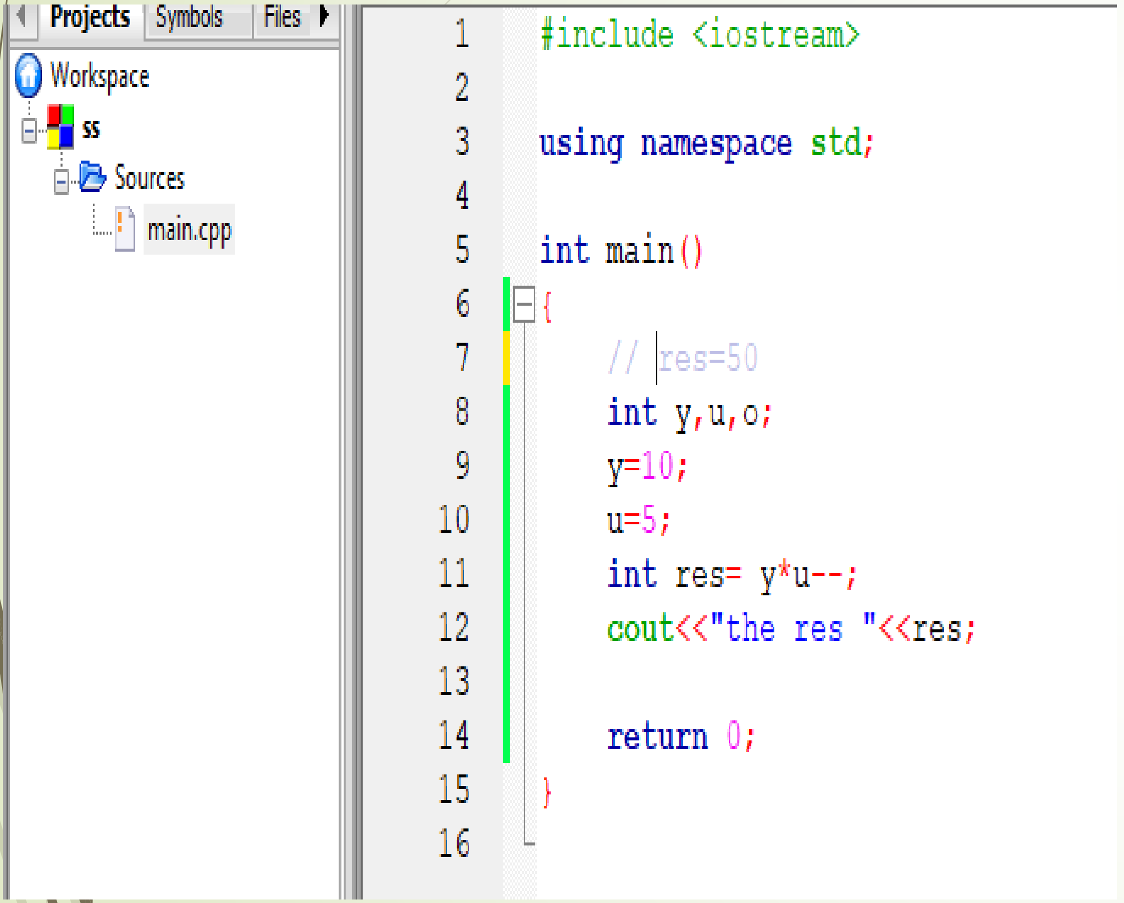


```
#include <iostream>

using namespace std;

int main()
{
    // res=60
    int y,u,o;
    y=10;
    u=5;
    int res= y*++u;
    cout<<"the res "<<res;

    return 0;
}
```

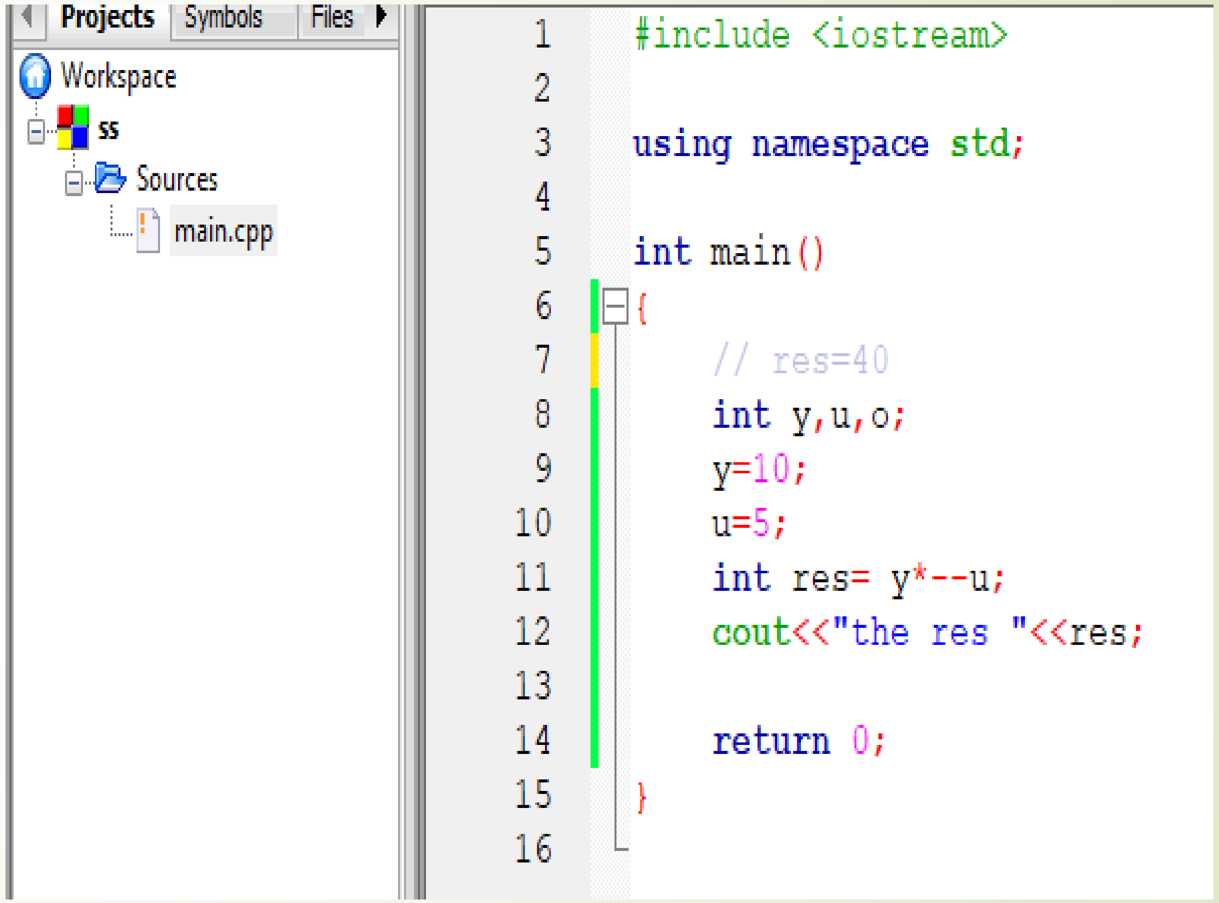


```
#include <iostream>

using namespace std;

int main()
{
    // Increment res=50
    int y,u,o;
    y=10;
    u=5;
    int res= y*u++;
    cout<<"the res "<<res;

    return 0;
}
```

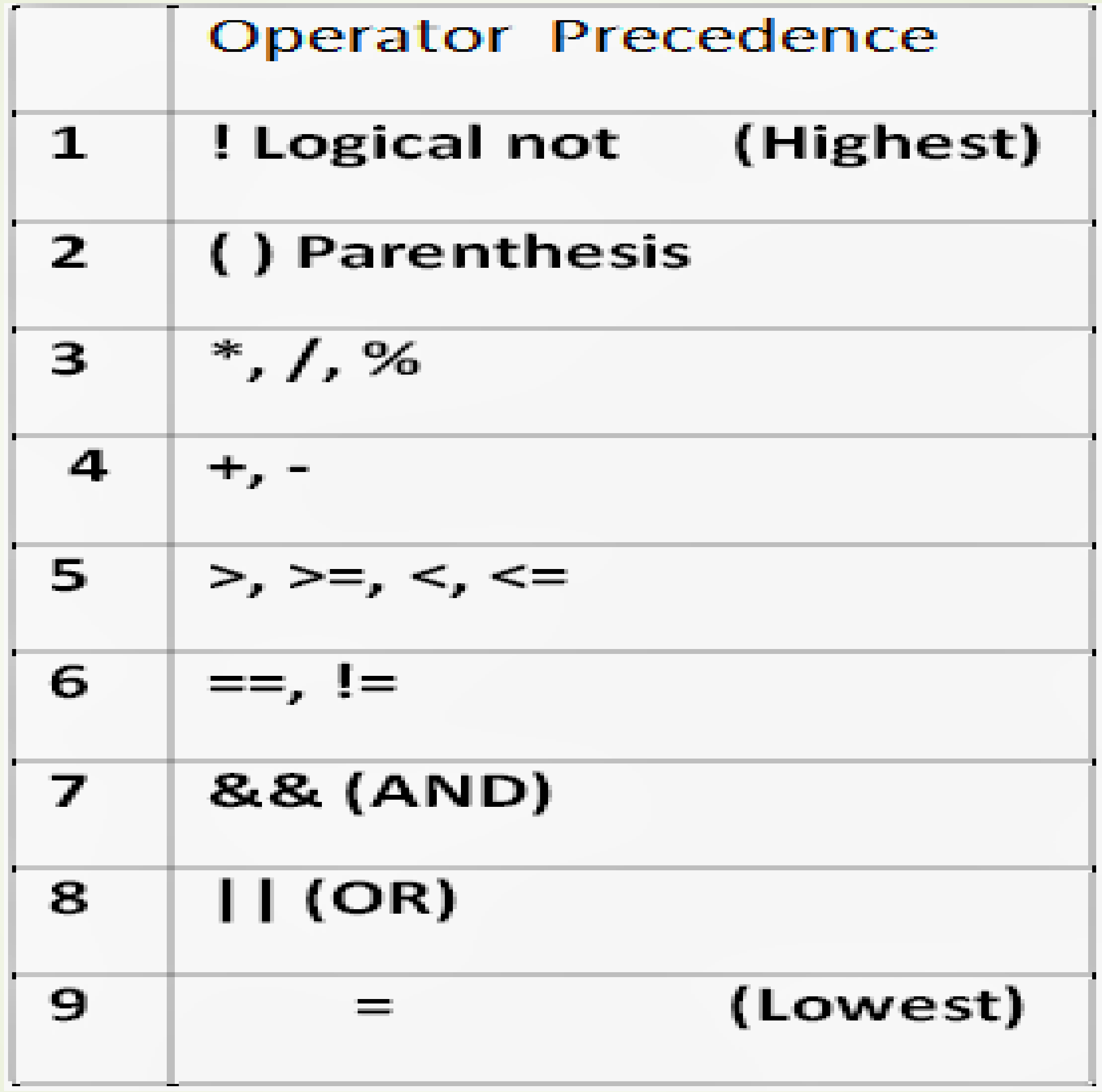
Difference between --x and x--



```
#include <iostream>

using namespace std;

int main()
{
    // res=50
    int y,u,o;
    y=10;
    u=5;
    int res= y*u--;
    cout<<"the res "<<res;

    return 0;
}
```



```
#include <iostream>

using namespace std;

int main()
{
    // res=40
    int y,u,o;
    y=10;
    u=5;
    int res= y*--u;
    cout<<"the res "<<res;

    return 0;
}
```



	Operator Precedence
1	! Logical not (Highest)
2	( ) Parenthesis
3	*, /, %
4	+, -
5	>, >=, <, <=
6	==, !=
7	&& (AND)
8	|| (OR)
9	= (Lowest)

Arithmetic Expressions

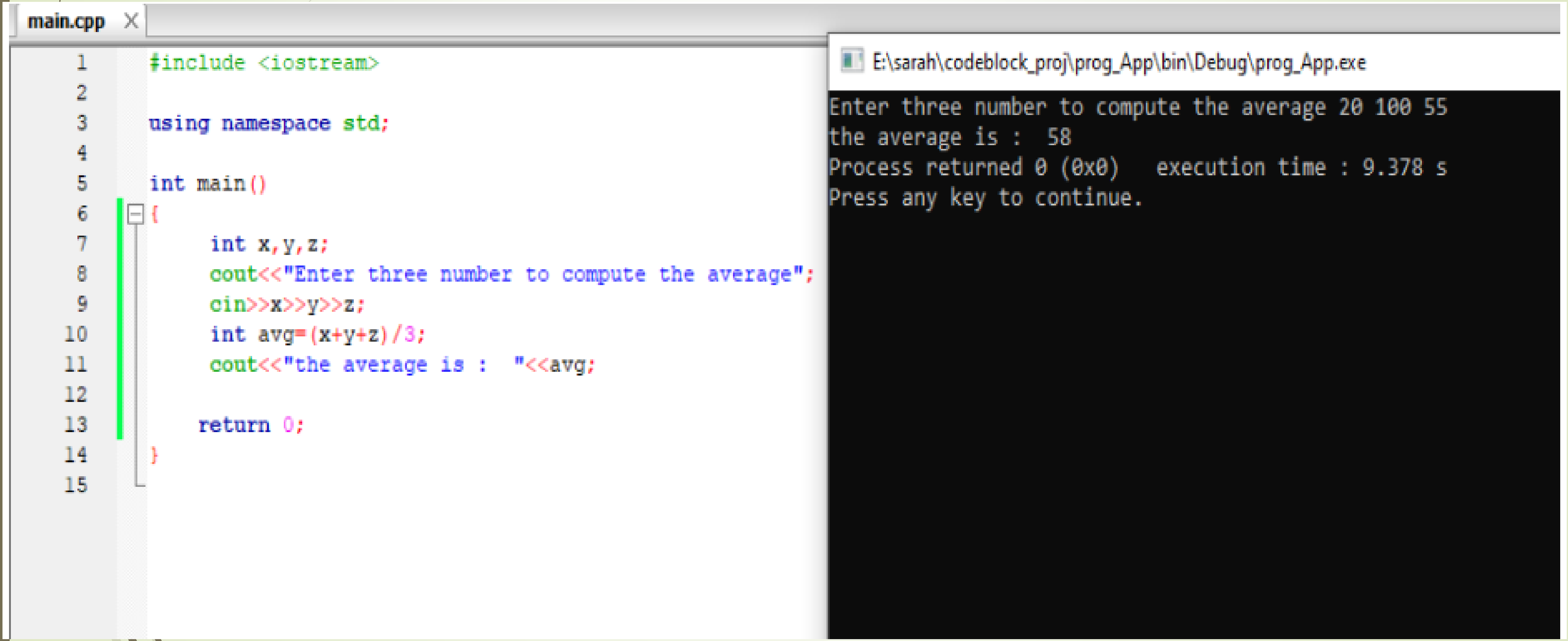
- Use spacing to make expressions readable
  - Which is easier to read?

```
x+y*z        or    x + y * z
```

- Precedence rules for operators are the same as used in your algebra classes
- Use parentheses to alter the order of operations

```
 x + y * z     ( y is multiplied by z first)
(x + y) * z   ( x and y are added first)
```

Write c++ program that ask the user to enter 3 numbers and then program compute the average of these number and print the result on screen?



main.cpp

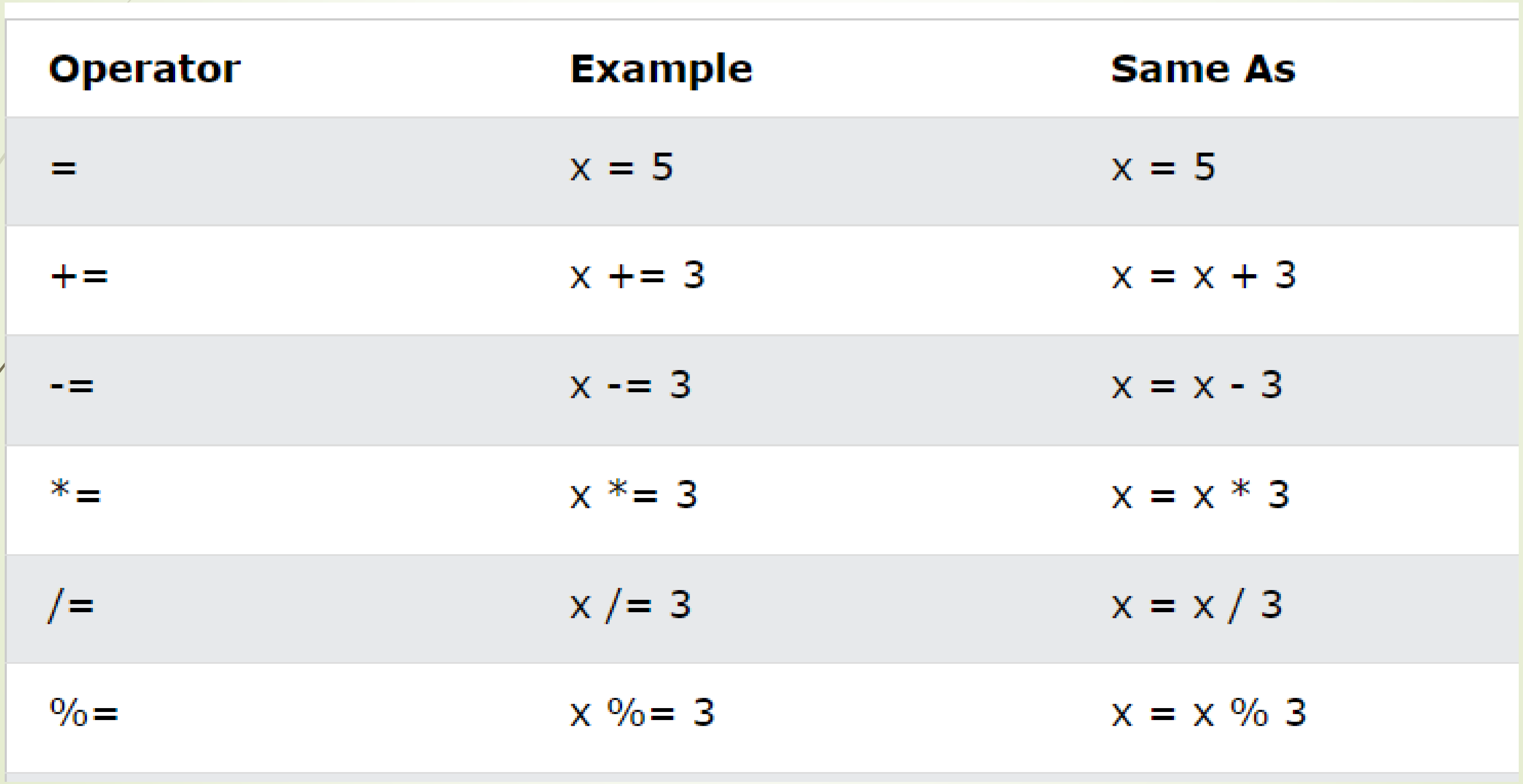
```cpp
#include <iostream>

using namespace std;

int main()
{
      int x,y,z;
      cout<<"Enter three number to compute the average";
      cin>>x>>y>>z;
      int avg=(x+y+z)/3;
      cout<<"the average is :  "<<avg;

     return 0;
}
```

E:\sarah\codeblock_proj\prog_App\bin\Debug\prog_App.exe

```
Enter three number to compute the average 20 100 55
the average is :  58
Process returned 0 (0x0)   execution time : 9.378 s
Press any key to continue.
```

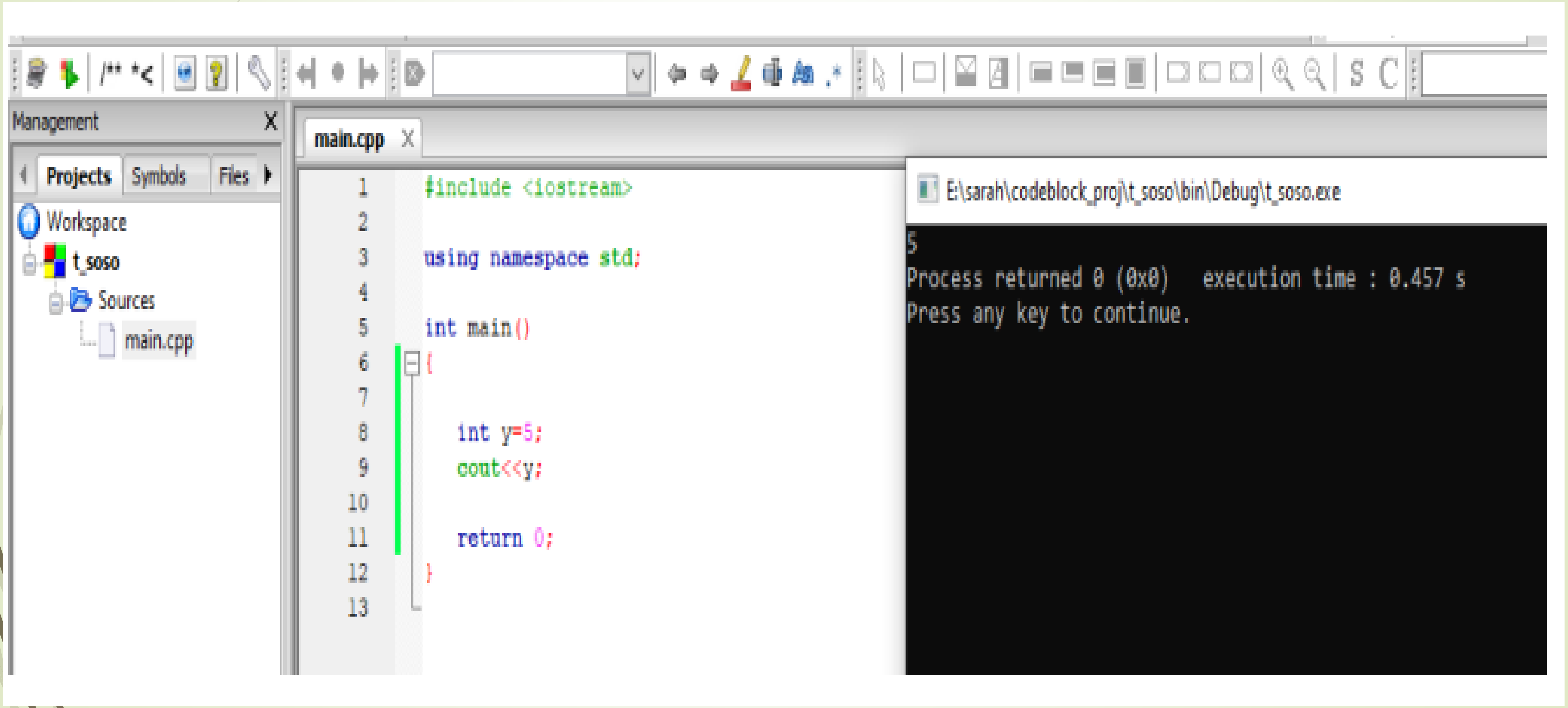
Operator Shorthand(Assignment)

- **Assignment operators are used to assign values to variables.**



Operator	Example	Same As
=	x = 5	x = 5
+=	x += 3	x = x + 3
-=	x -= 3	x = x - 3
*=	x *= 3	x = x * 3
/=	x /= 3	x = x / 3
%=	x %= 3	x = x % 3

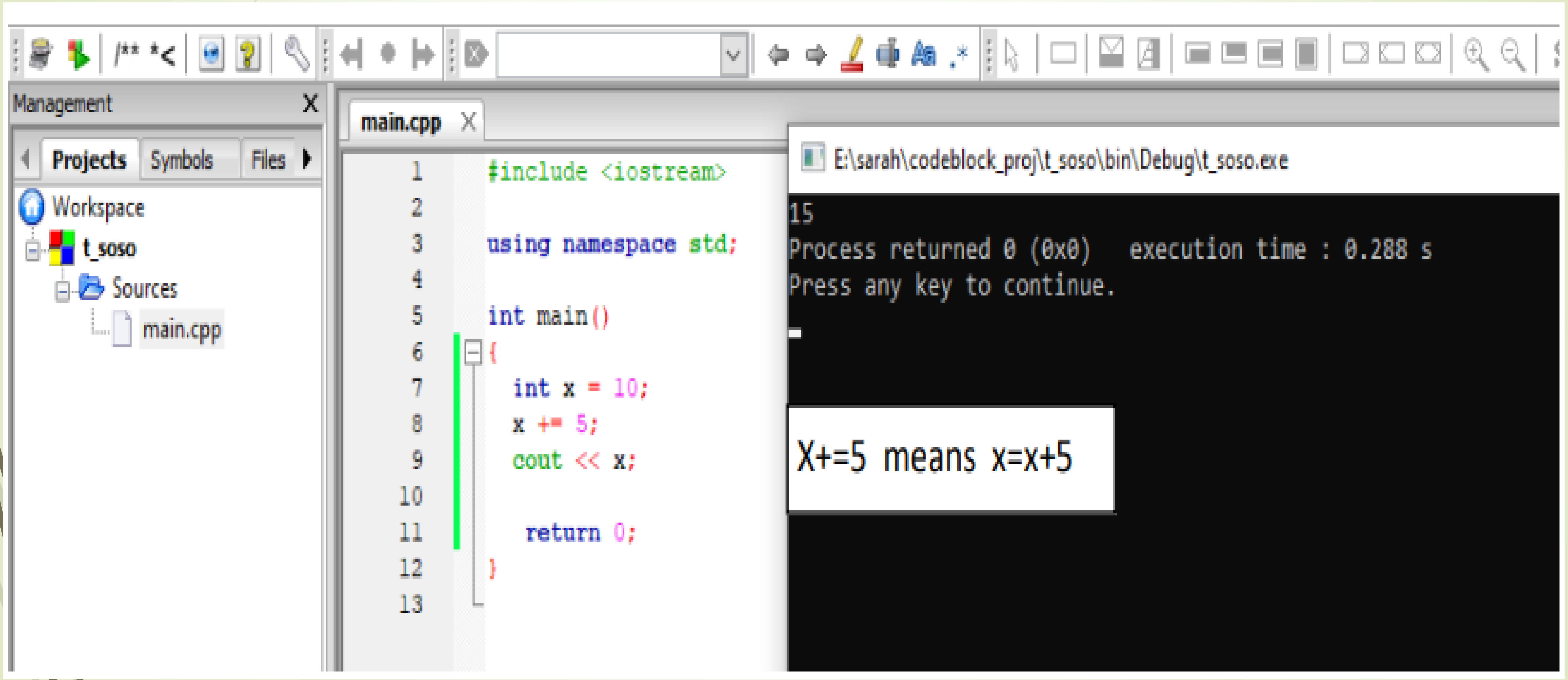
In the example below, we use the assignment operator (=) to assign the value 5 to a variable called x:



```cpp
#include <iostream>

using namespace std;

int main()
{

    int y=5;
    cout<<y;

    return 0;
}

```

```
5
Process returned 0 (0x0)   execution time : 0.457 s
Press any key to continue.
```

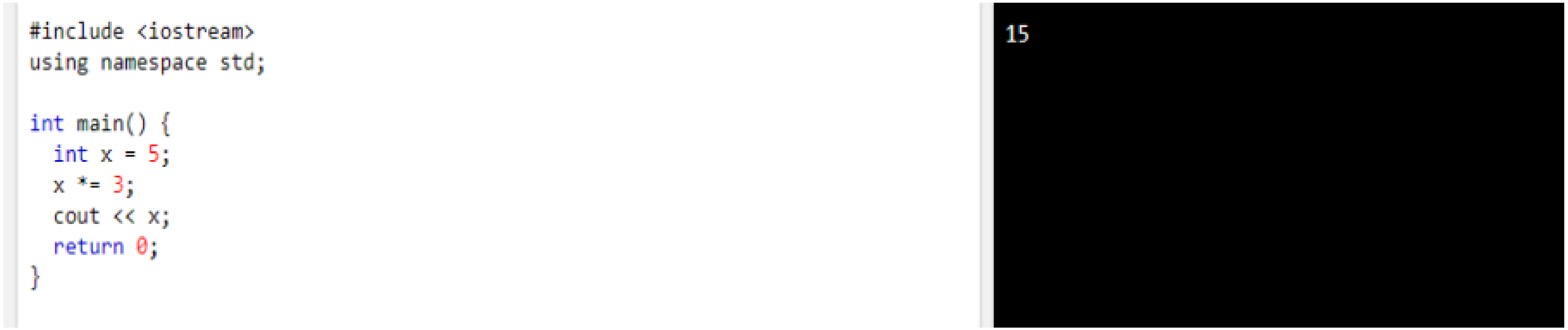
The addition assignment operator (+=) adds a value to a variable:



```cpp
#include <iostream>

using namespace std;

int main()
{
   int x = 10;
   x += 5;
   cout << x;

    return 0;
}
```

```
15
Process returned 0 (0x0)   execution time : 0.288 s
Press any key to continue.
```

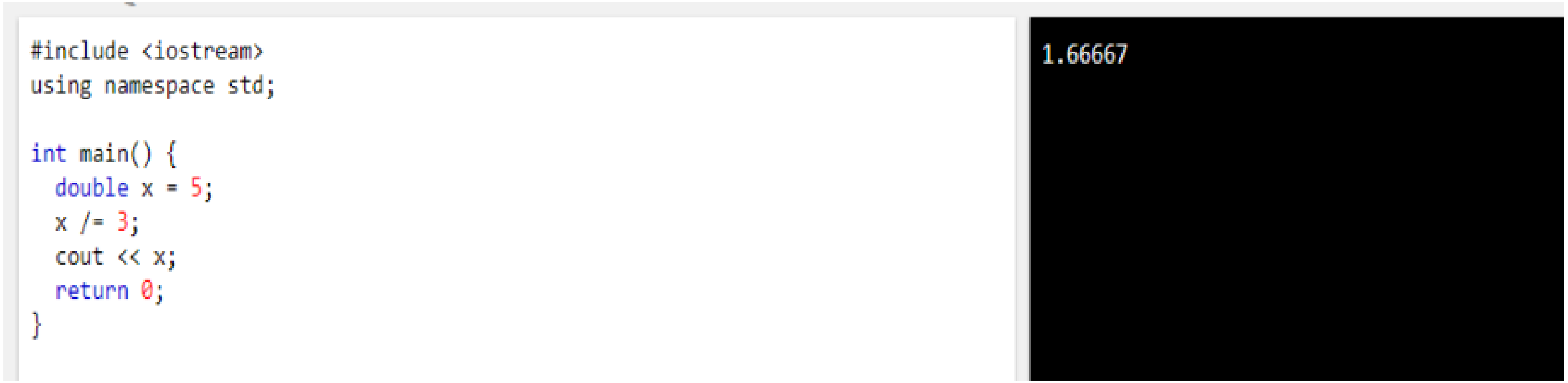
X+=5 means x=x+5



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

int main() {
  int x = 5;
  x *= 3;
  cout << x;
  return 0;
}
```

```
15
```

Example 3:

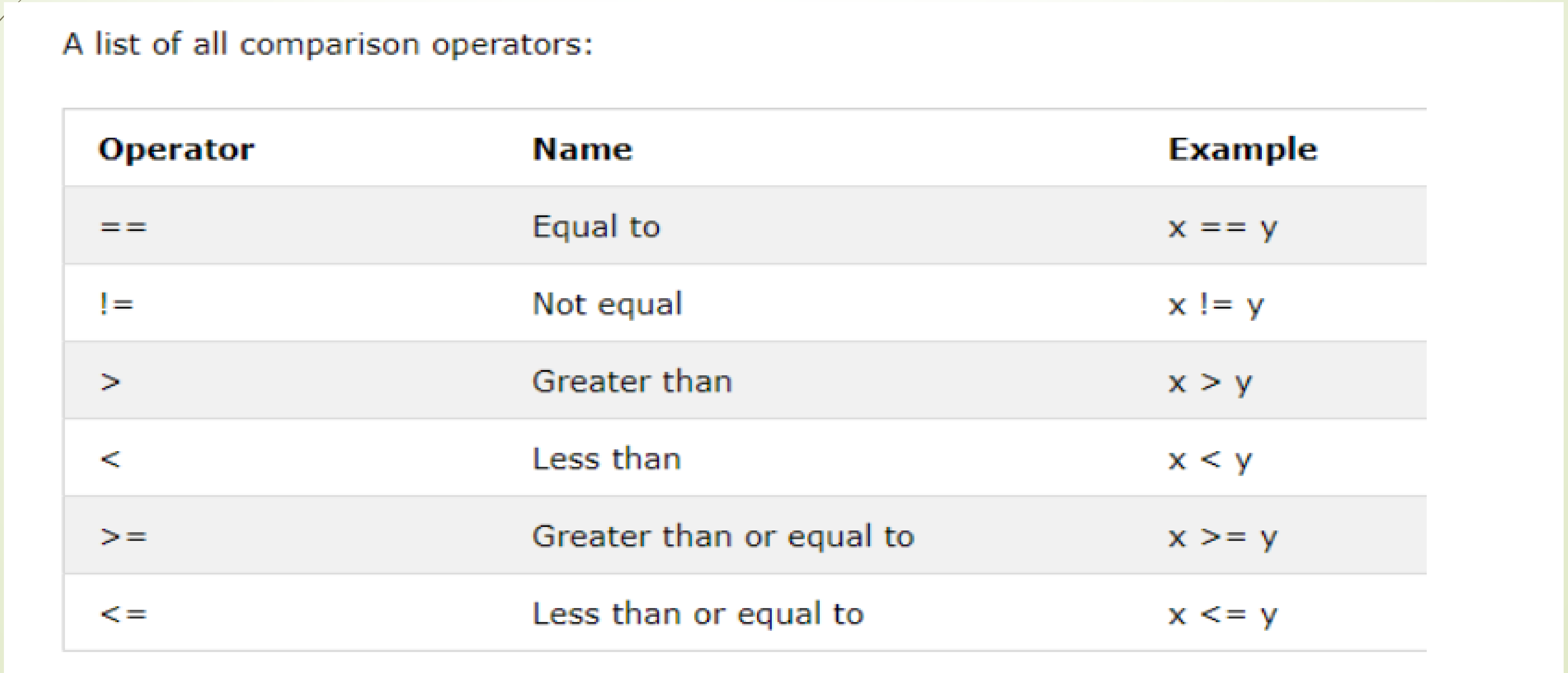


```
#include <iostream>
using namespace std;

int main() {
  double x = 5;
  x /= 3;
  cout << x;
  return 0;
}
```

```
1.66667
```

Comparison Operators

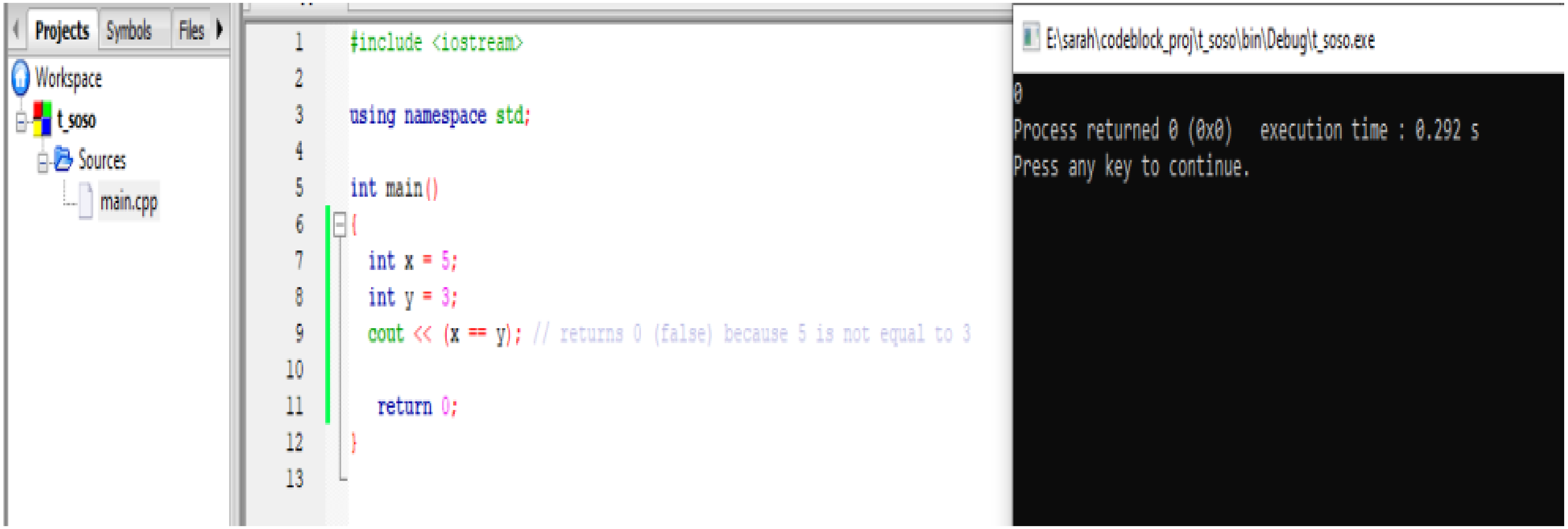
- In order to evaluate a comparison between two expressions we can use the relational and equality operators. The result of a relational operation is a Boolean value that can only be true or false, according to its Boolean result.
- Comparison operators are used to compare two values. Note: The return value of a comparison is either true (1) or false (0).



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

Example1:

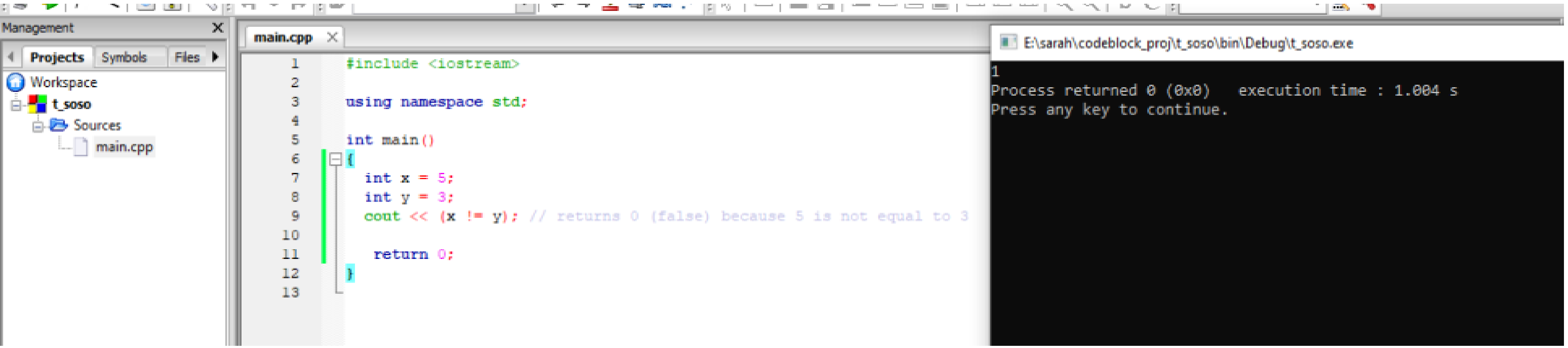


```cpp
#include <iostream>

using namespace std;

int main()
{
  int x = 5;
  int y = 3;
  cout << (x == y); // returns 0 (false) because 5 is not equal to 3

   return 0;
}

```

E:\sarah\codeblock_proj\t_soso\bin\Debug\t_soso.exe

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

Example2:



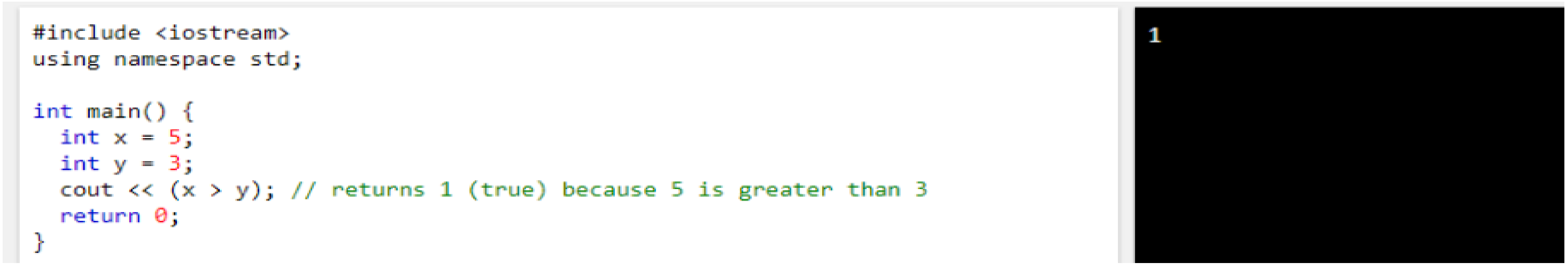
main.cpp

```
#include <iostream>

using namespace std;

int main()
{
  int x = 5;
  int y = 3;
  cout << (x != y); // returns 0 (false) because 5 is not equal to 3

   return 0;
}

```

E:\sarah\codeblock_proj\t_soso\bin\Debug\t_soso.exe

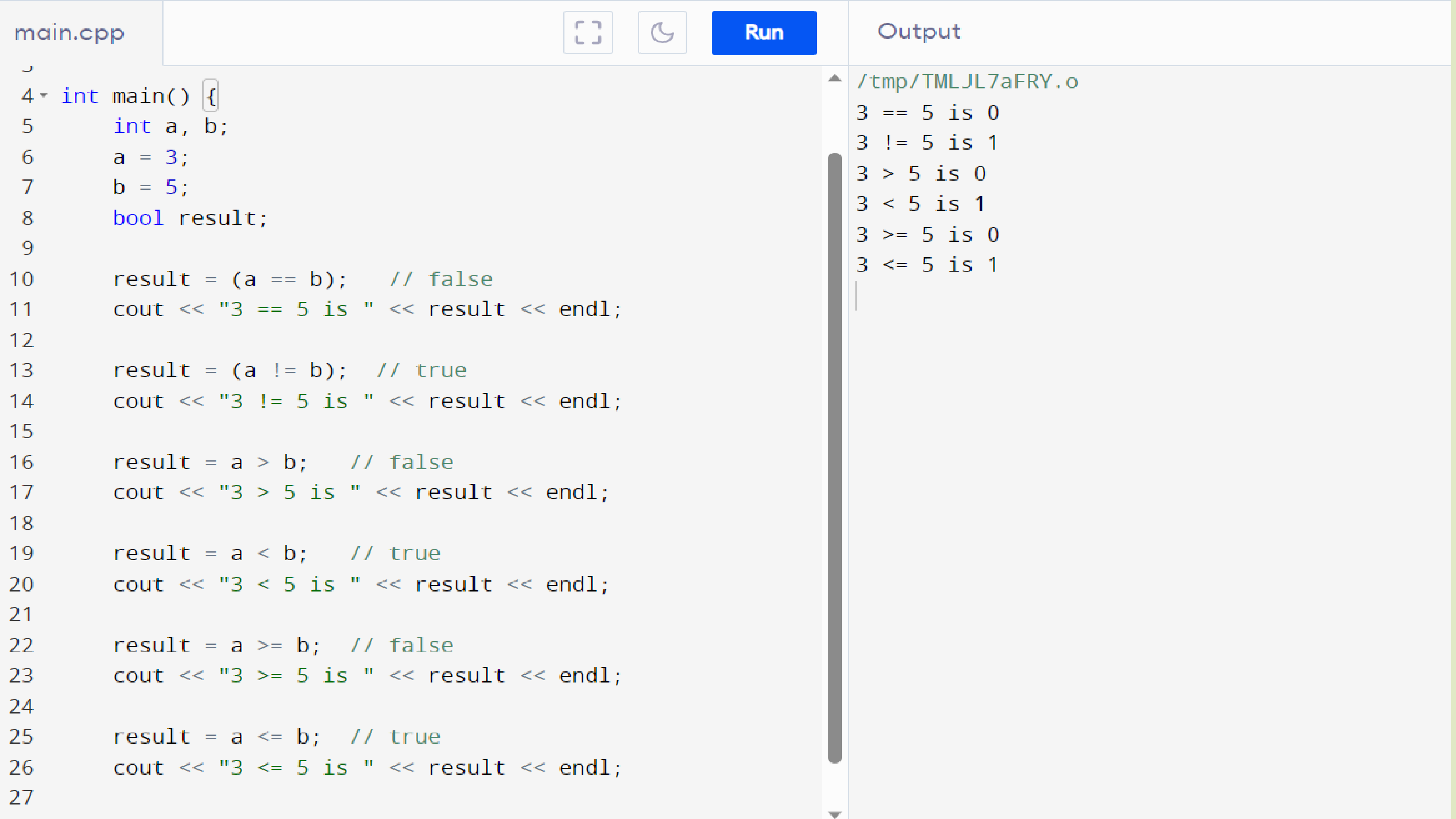
```
1
Process returned 0 (0x0)   execution time : 1.004 s
Press any key to continue.
```

Example 3



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

int main() {
  int x = 5;
  int y = 3;
  cout << (x > y); // returns 1 (true) because 5 is greater than 3
  return 0;
}
```

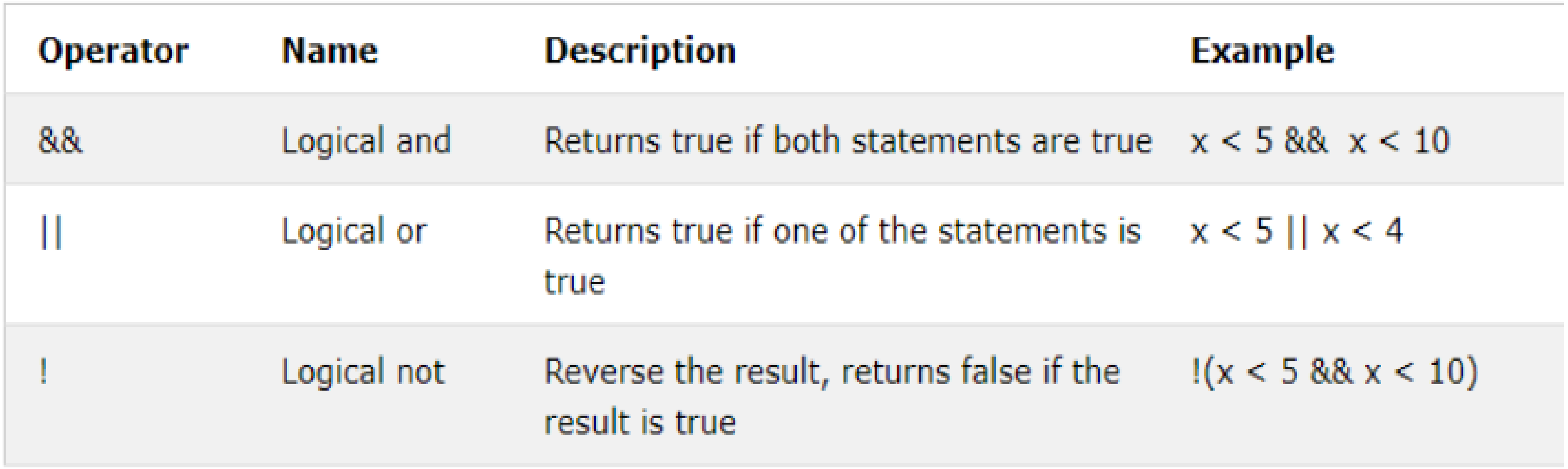
```
1
```



```cpp
int main() {
    int a, b;
    a = 3;
    b = 5;
    bool result;

    result = (a == b);   // false
    cout << "3 == 5 is " << result << endl;

    result = (a != b);  // true
    cout << "3 != 5 is " << result << endl;

    result = a > b;   // false
    cout << "3 > 5 is " << result << endl;

    result = a < b;   // true
    cout << "3 < 5 is " << result << endl;

    result = a >= b;  // false
    cout << "3 >= 5 is " << result << endl;

    result = a <= b;  // true
    cout << "3 <= 5 is " << result << endl;

```

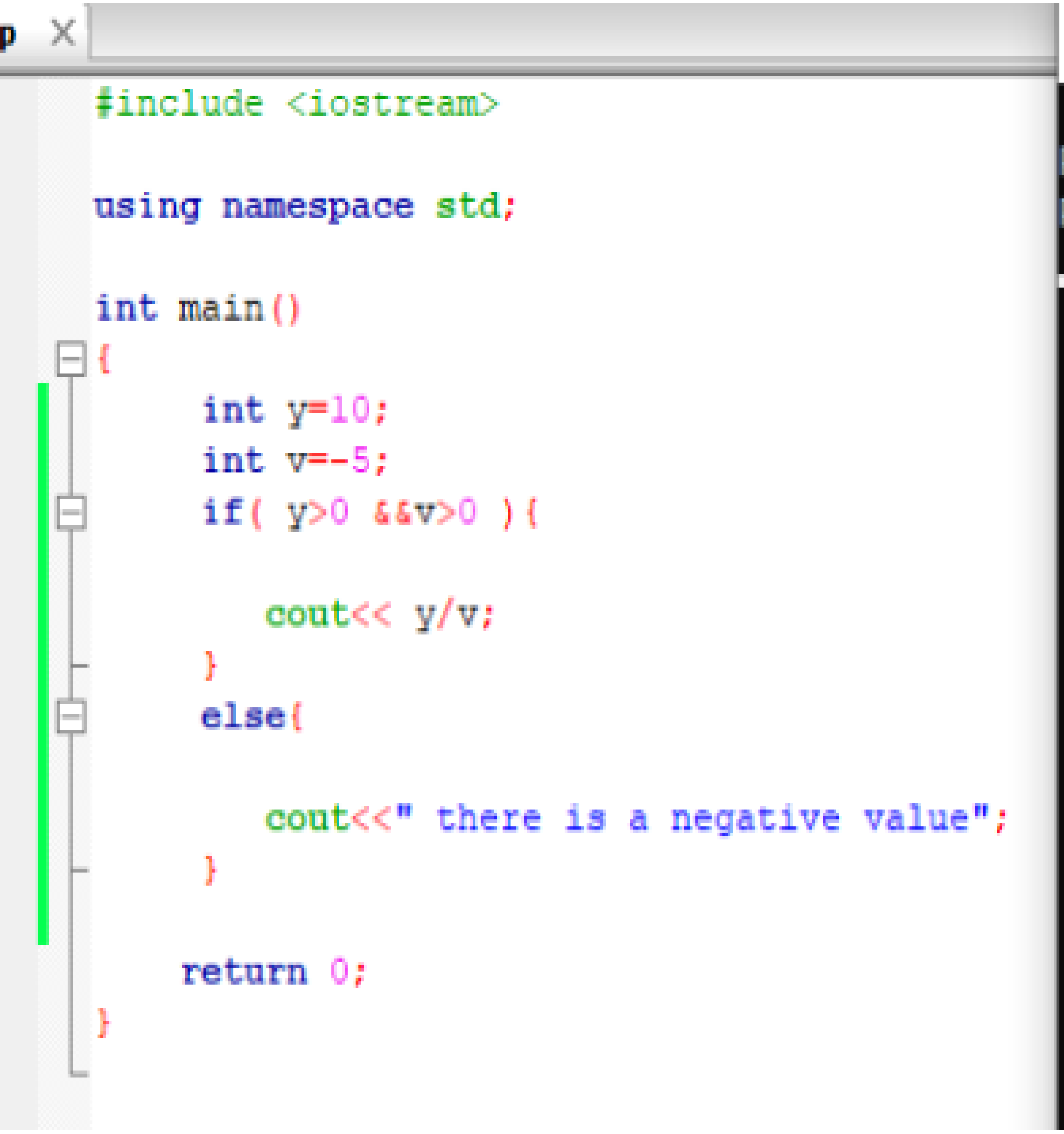
```
/tmp/TMLJL7aFRY.o
3 == 5 is 0
3 != 5 is 1
3 > 5 is 0
3 < 5 is 1
3 >= 5 is 0
3 <= 5 is 1
```

Logical Operators

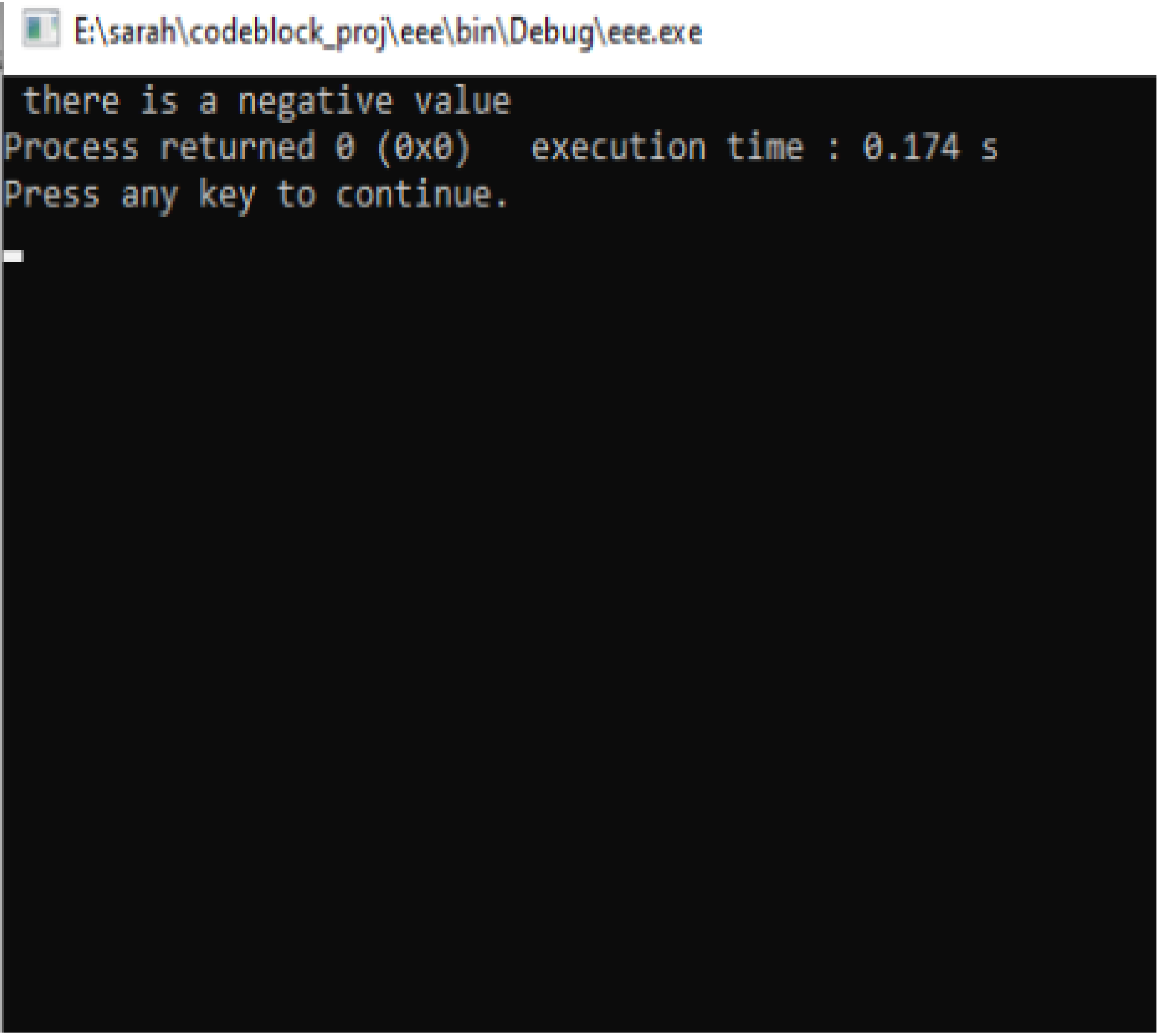


Operator	Name	Description	Example
&&	Logical and	Returns true if both statements are true	x < 5 && x < 10
||	Logical or	Returns true if one of the statements is true	x < 5 || x < 4
!	Logical not	Reverse the result, returns false if the result is true	!(x < 5 && x < 10)

Example for And &&



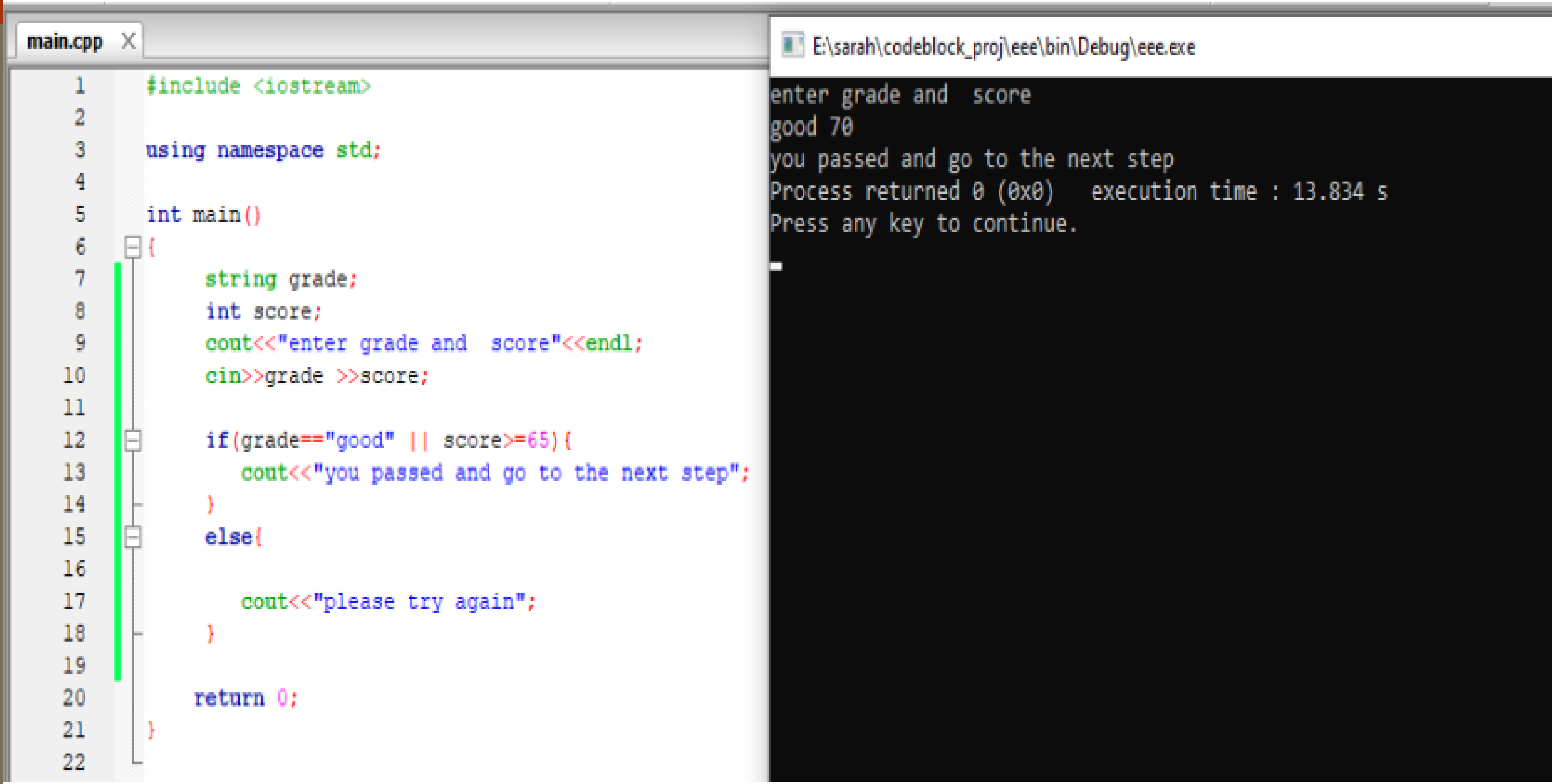
```
#include <iostream>

using namespace std;

int main()
{
      int y=10;
      int v=-5;
      if( y>0 &&v>0 ){

         cout<< y/v;
      }
      else{

         cout<<" there is a negative value";
      }

    return 0;
}
```



E:\sarah\codeblock_proj\eee\bin\Debug\eee.exe

```
 there is a negative value
Process returned 0 (0x0)   execution time : 0.174 s
Press any key to continue.
```

Example :Or ||

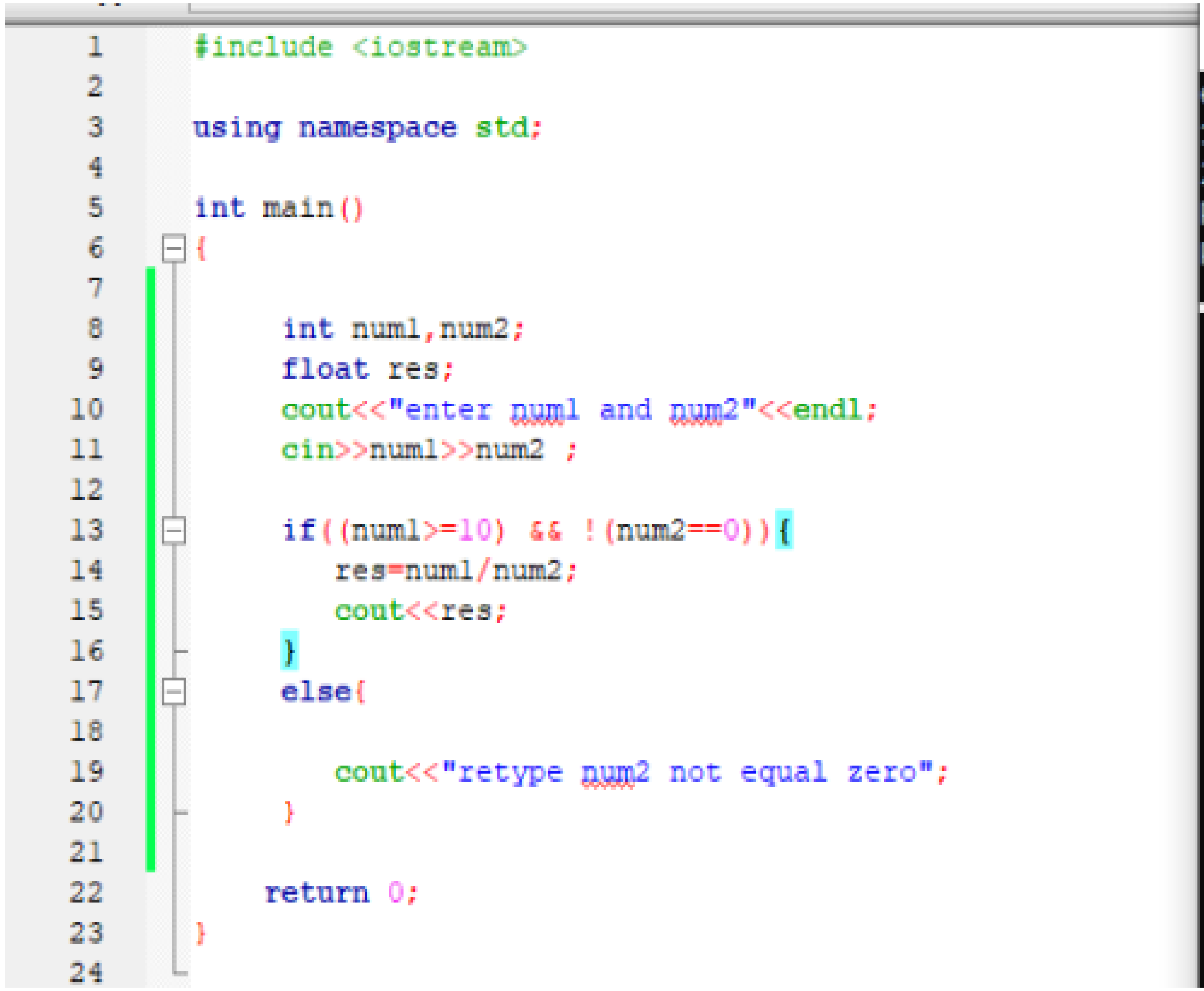

main.cpp

```cpp
#include <iostream>

using namespace std;

int main()
{
     string grade;
     int score;
     cout<<"enter grade and  score"<<endl;
     cin>>grade >>score;

     if(grade=="good" || score>=65){
        cout<<"you passed and go to the next step";
     }
     else{

        cout<<"please try again";
     }

    return 0;
}
```

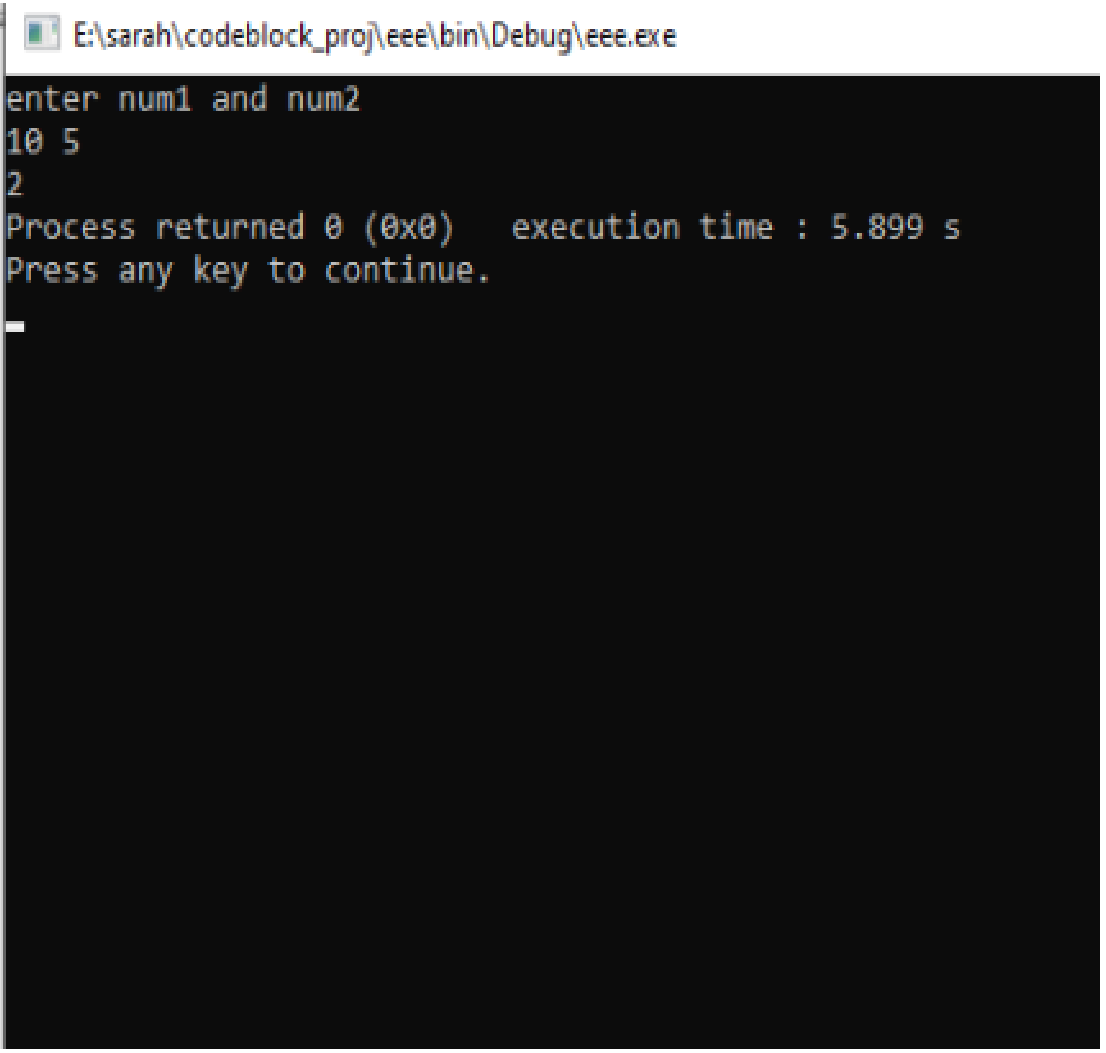
E:\sarah\codeblock_proj\eee\bin\Debug\eee.exe

```
enter grade and  score
good 70
you passed and go to the next step
Process returned 0 (0x0)   execution time : 13.834 s
Press any key to continue.
```

Example for Not



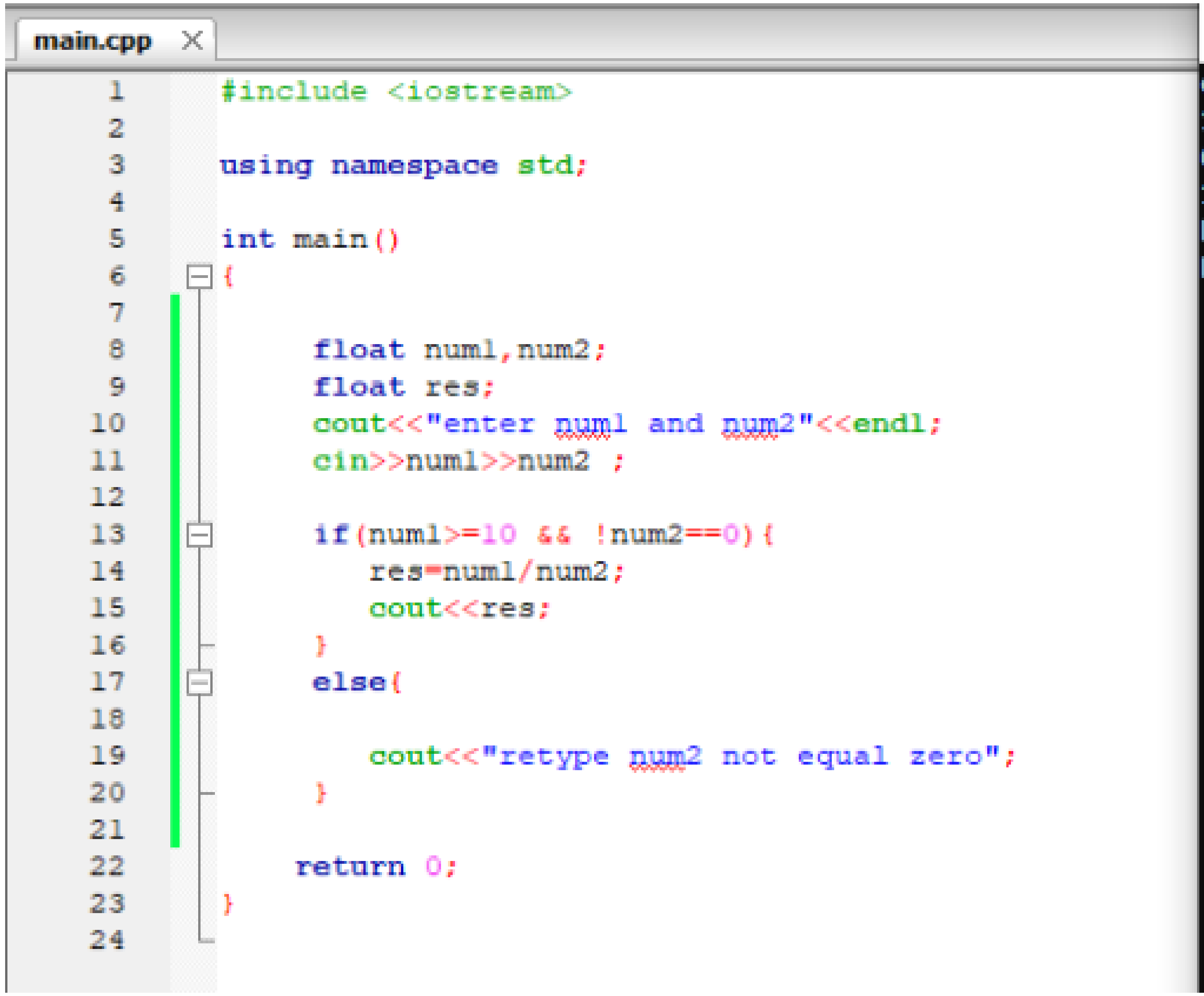
```cpp
#include <iostream>

using namespace std;

int main()
{

     int num1,num2;
     float res;
     cout<<"enter num1 and num2"<<endl;
     cin>>num1>>num2 ;

     if((num1>=10) && !(num2==0)){
        res=num1/num2;
        cout<<res;
     }
     else{

        cout<<"retype num2 not equal zero";
     }

    return 0;
}
```



E:\sarah\codeblock_proj\eee\bin\Debug\eee.exe

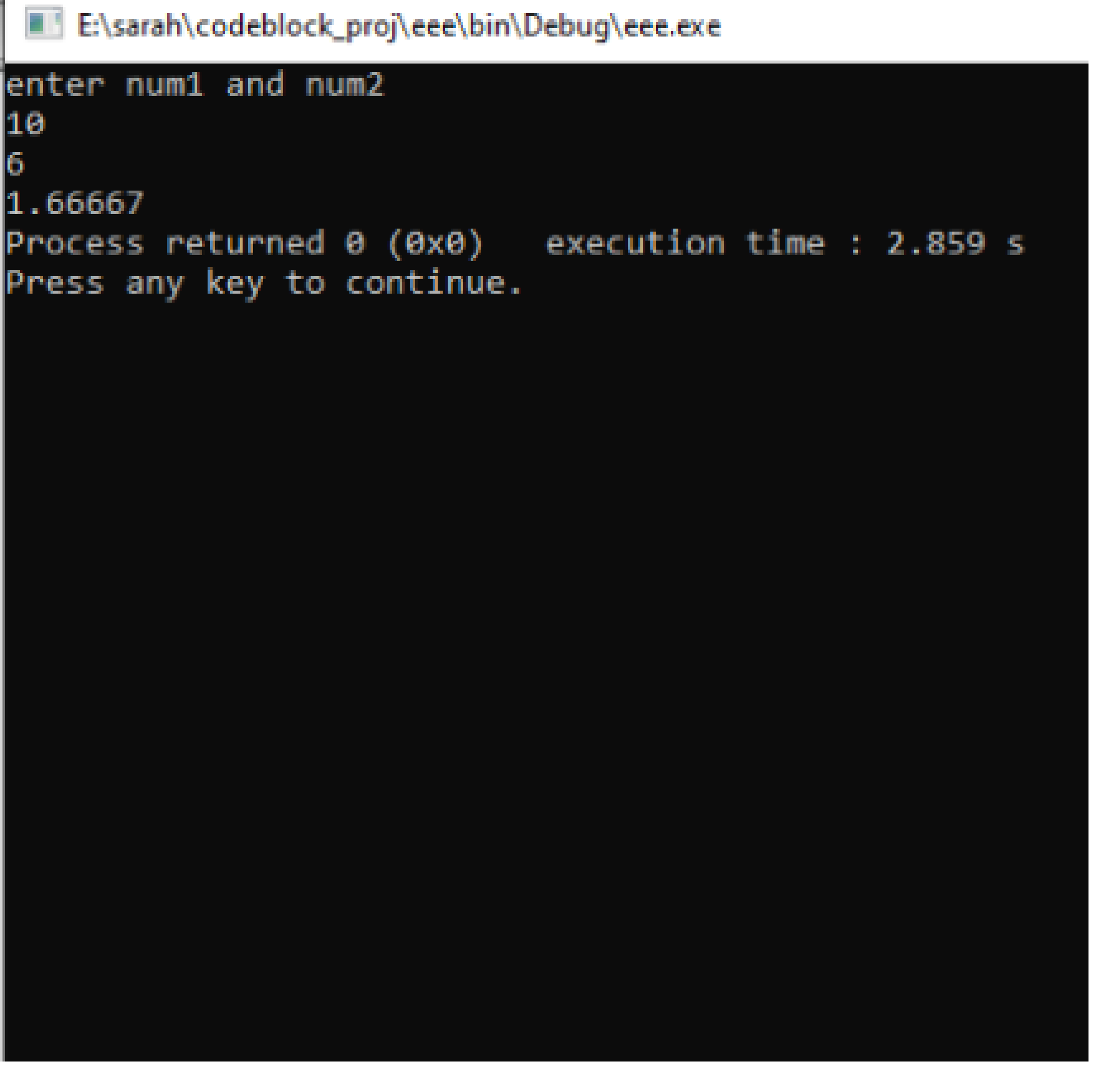
```
enter num1 and num2
10 5
2
Process returned 0 (0x0)   execution time : 5.899 s
Press any key to continue.
```

This also right



main.cpp

```cpp
#include <iostream>

using namespace std;

int main()
{

      float num1,num2;
      float res;
      cout<<"enter num1 and num2"<<endl;
      cin>>num1>>num2 ;

      if(num1>=10 && !num2==0){
         res=num1/num2;
         cout<<res;
      }
      else{

         cout<<"retype num2 not equal zero";
      }

     return 0;
}
```



E:\sarah\codeblock_proj\eee\bin\Debug\eee.exe

```
enter num1 and num2
10
6
1.66667
Process returned 0 (0x0)   execution time : 2.859 s
Press any key to continue.
```

AND

- Boolean expressions can be combined into more complex expressions with
  - && -- The AND operator
    - True if both expressions are true
  - Syntax: (*Comparison_1*) && (*Comparison_2*)
- Example: if ((2 < x) && (x < 7))
  - True only if x is between 2 and 7
  - Inside parentheses are optional but enhance meaning

OR

- | | :The OR operator (no space!)
  - True if either or both expressions are true

- Syntax: (Comparison_1) | | (Comparison_2)

- Example: if ((x = = 1) | | (x = = y))
  - True if x contains 1
  - True if x contains the same value as y
  - True if both comparisons are true

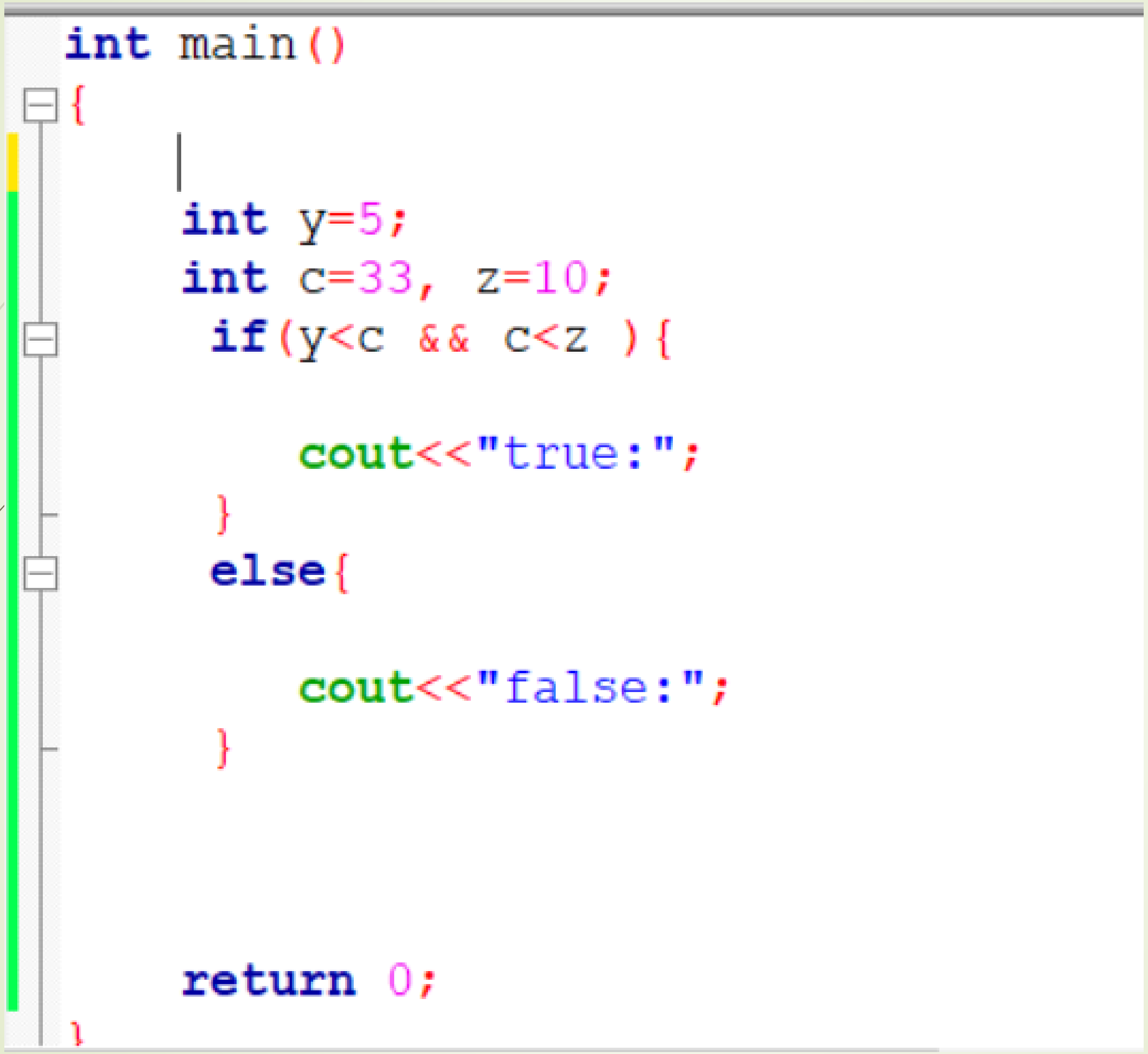
NOT

- ! -- negates any Boolean expression
  - !(x < y)
    - True if x is NOT less than y
  - !(x = = y)
    - True if x is NOT equal to y
- ! Operator can make expressions difficult to understand…use only when appropriate

Inequalities

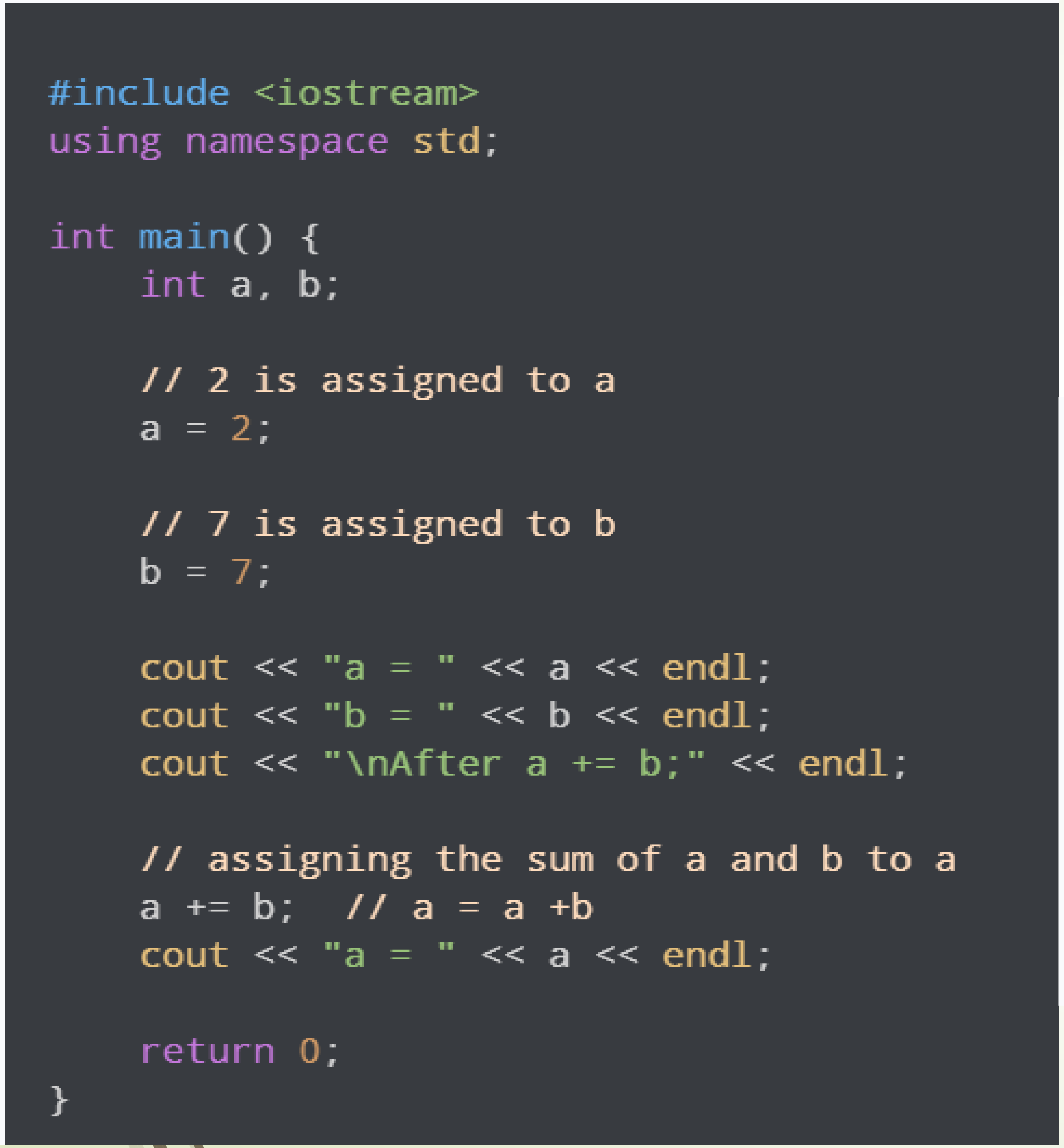
- Be careful translating inequalities to C++
- if x < y < z   translates as

```
if ( ( x < y )  && ( y < z ) )

     NOT

if ( x < y < z )
```

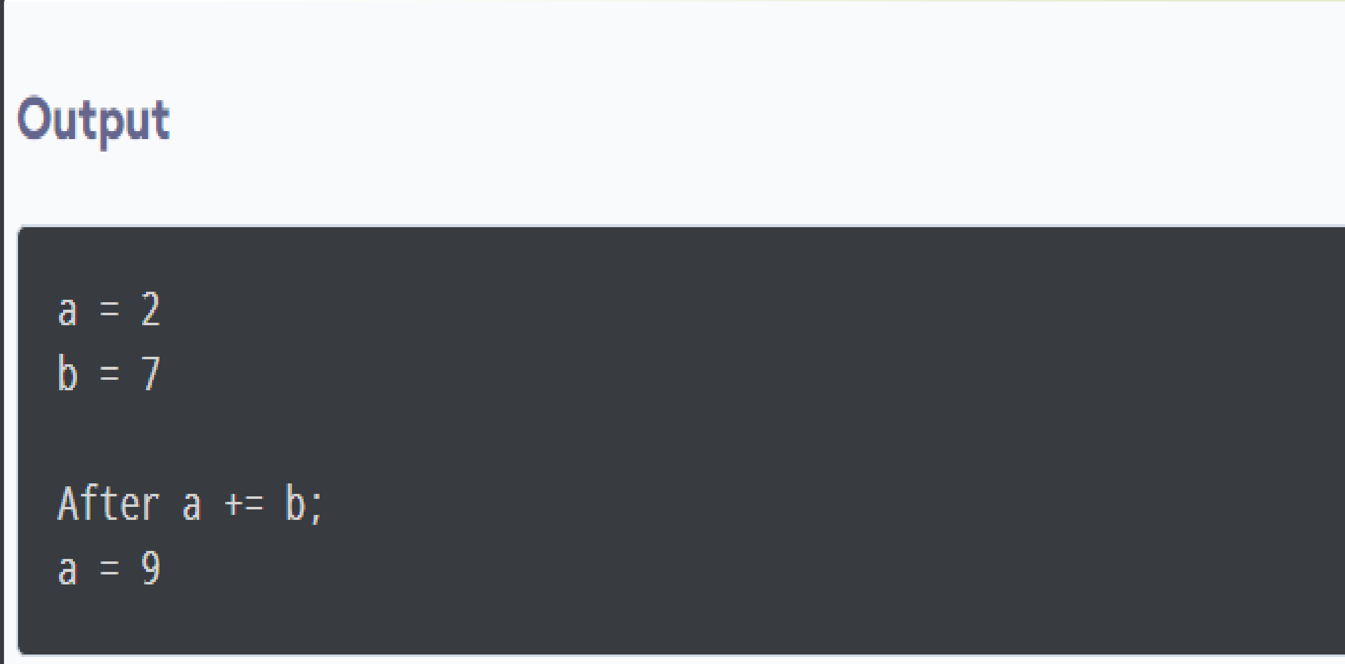


```
int main()
{

      int y=5;
      int c=33, z=10;
       if(y<c && c<z ){

          cout<<"true:";
       }
       else{

          cout<<"false:";
       }



      return 0;
}
```

Example



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

int main() {
    int a, b;

    // 2 is assigned to a
    a = 2;

    // 7 is assigned to b
    b = 7;

    cout << "a = " << a << endl;
    cout << "b = " << b << endl;
    cout << "\nAfter a += b;" << endl;

    // assigning the sum of a and b to a
    a += b;  // a = a +b
    cout << "a = " << a << endl;

    return 0;
}
```

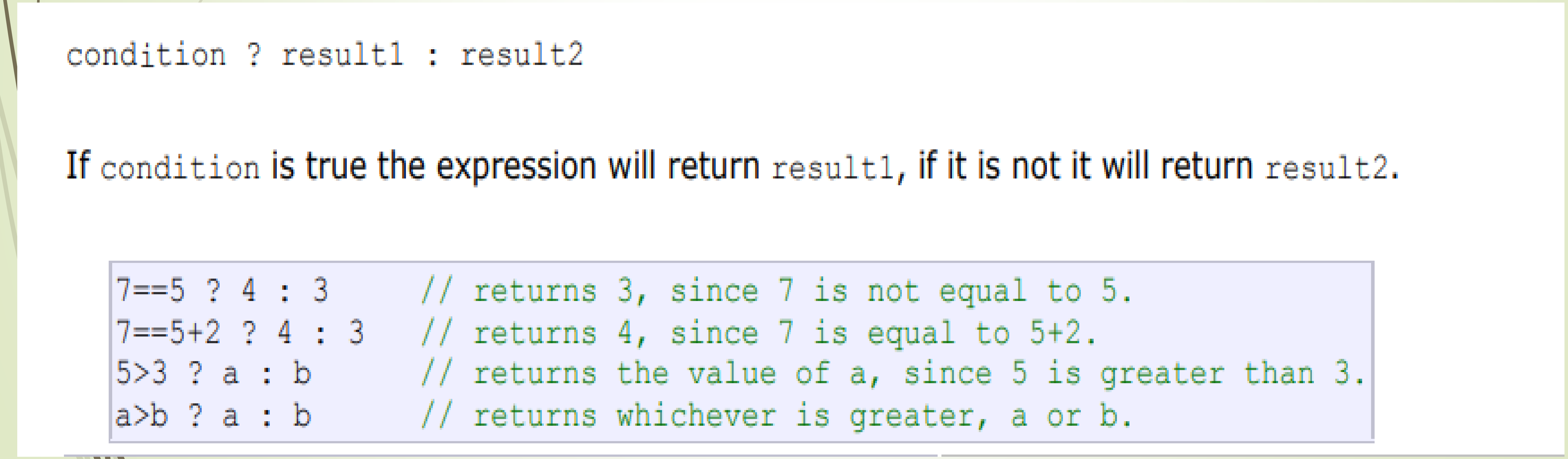


Output

```
a = 2
b = 7

After a += b;
a = 9
```

Conditional operator (?)

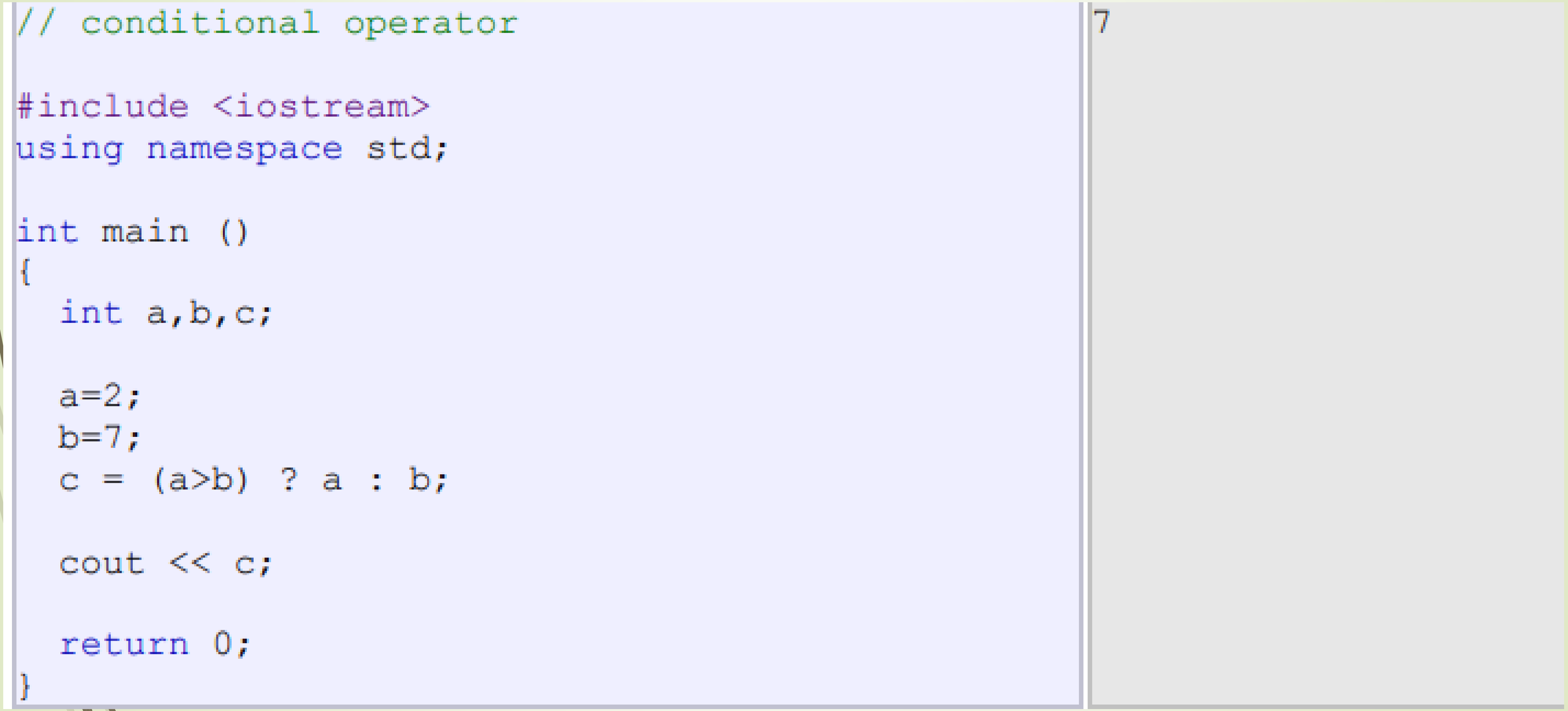
- **The conditional operator evaluates an expression returning a value if that expression is true and a different one if the expression is evaluated as false. Its format is:**



```
condition ? result1 : result2
```

If `condition` is true the expression will return `result1`, if it is not it will return `result2`.

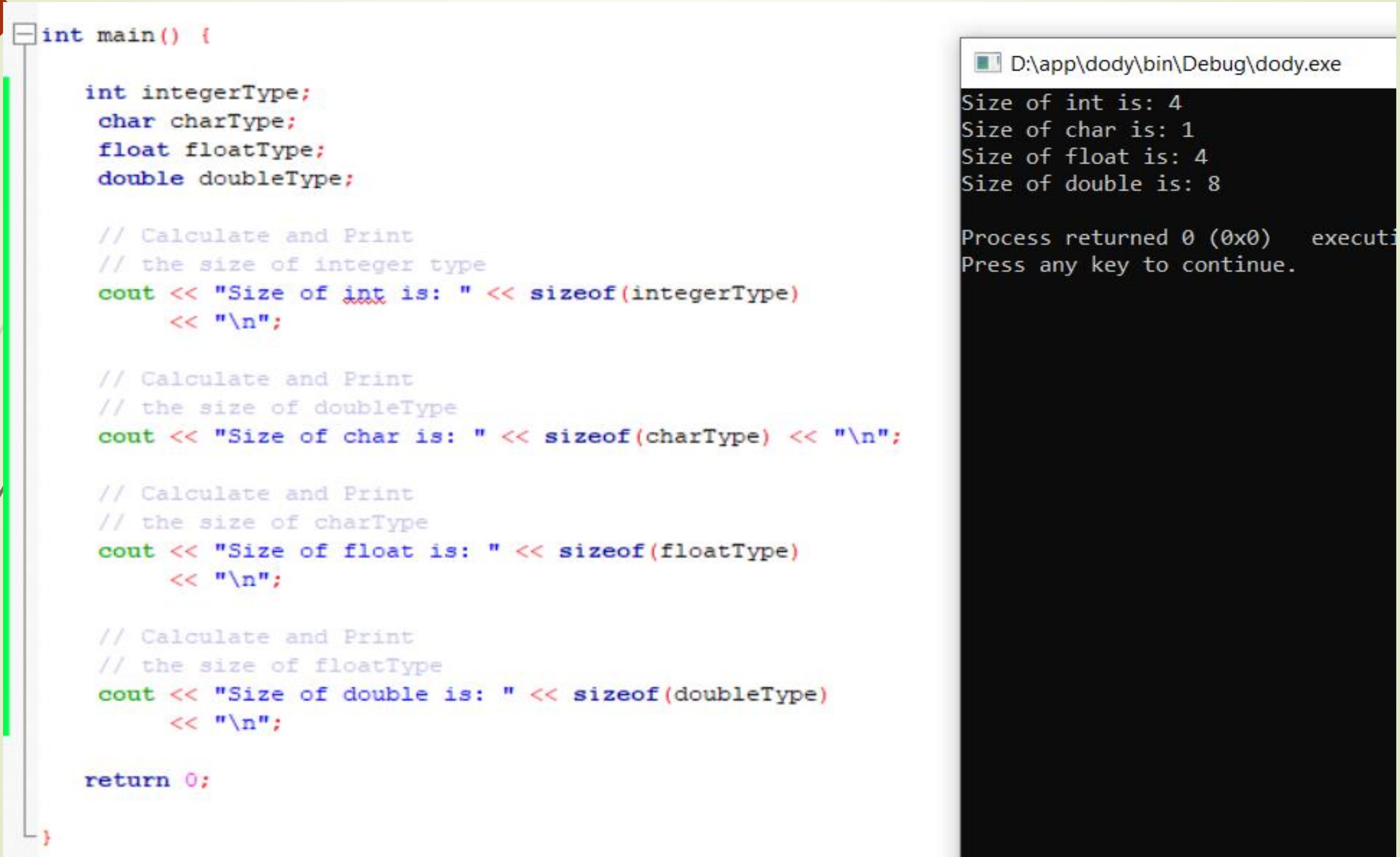
```
7==5 ? 4 : 3     // returns 3, since 7 is not equal to 5.
7==5+2 ? 4 : 3   // returns 4, since 7 is equal to 5+2.
5>3 ? a : b      // returns the value of a, since 5 is greater than 3.
a>b ? a : b      // returns whichever is greater, a or b.
```



```
// conditional operator

#include <iostream>
using namespace std;

int main ()
{
  int a,b,c;

  a=2;
  b=7;
  c = (a>b) ? a : b;

  cout << c;

  return 0;
}
```

```
7
```

C++ Program to Find the Size of a Data Types



```
int main() {

    int integerType;
     char charType;
     float floatType;
     double doubleType;

     // Calculate and Print
     // the size of integer type
     cout << "Size of int is: " << sizeof(integerType)
          << "\n";

     // Calculate and Print
     // the size of doubleType
     cout << "Size of char is: " << sizeof(charType) << "\n";

     // Calculate and Print
     // the size of charType
     cout << "Size of float is: " << sizeof(floatType)
          << "\n";

     // Calculate and Print
     // the size of floatType
     cout << "Size of double is: " << sizeof(doubleType)
          << "\n";

    return 0;
}
```

D:\app\dody\bin\Debug\dody.exe

```
Size of int is: 4
Size of char is: 1
Size of float is: 4
Size of double is: 8

Process returned 0 (0x0)   executi
Press any key to continue.
```

References

- Sharanya Jayaraman, introduction to C++ , Department of Computer Science Florida State University 2024.

- Juan Soulié,C++ Language Tutorial 2008, Available online at : http://www.cplusplus.com/doc/tutorial/

Thank you