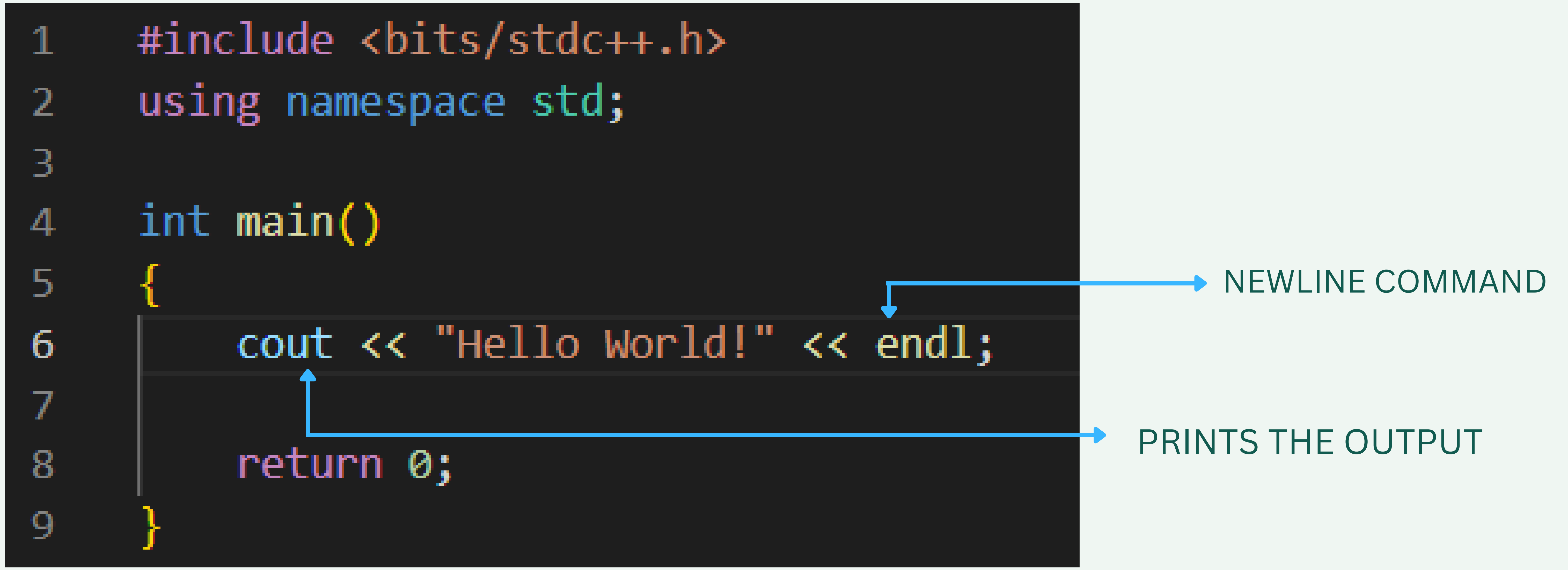
INTRO TO C++

Presenters
Ayman Akhter , Shankar Balajee

TOPICS FOR TODAY

- Basic Input and Output
- Variables and Arithmetic
- Conditional Statements(if-else)
- Loops
- Arrays
- Some Problems

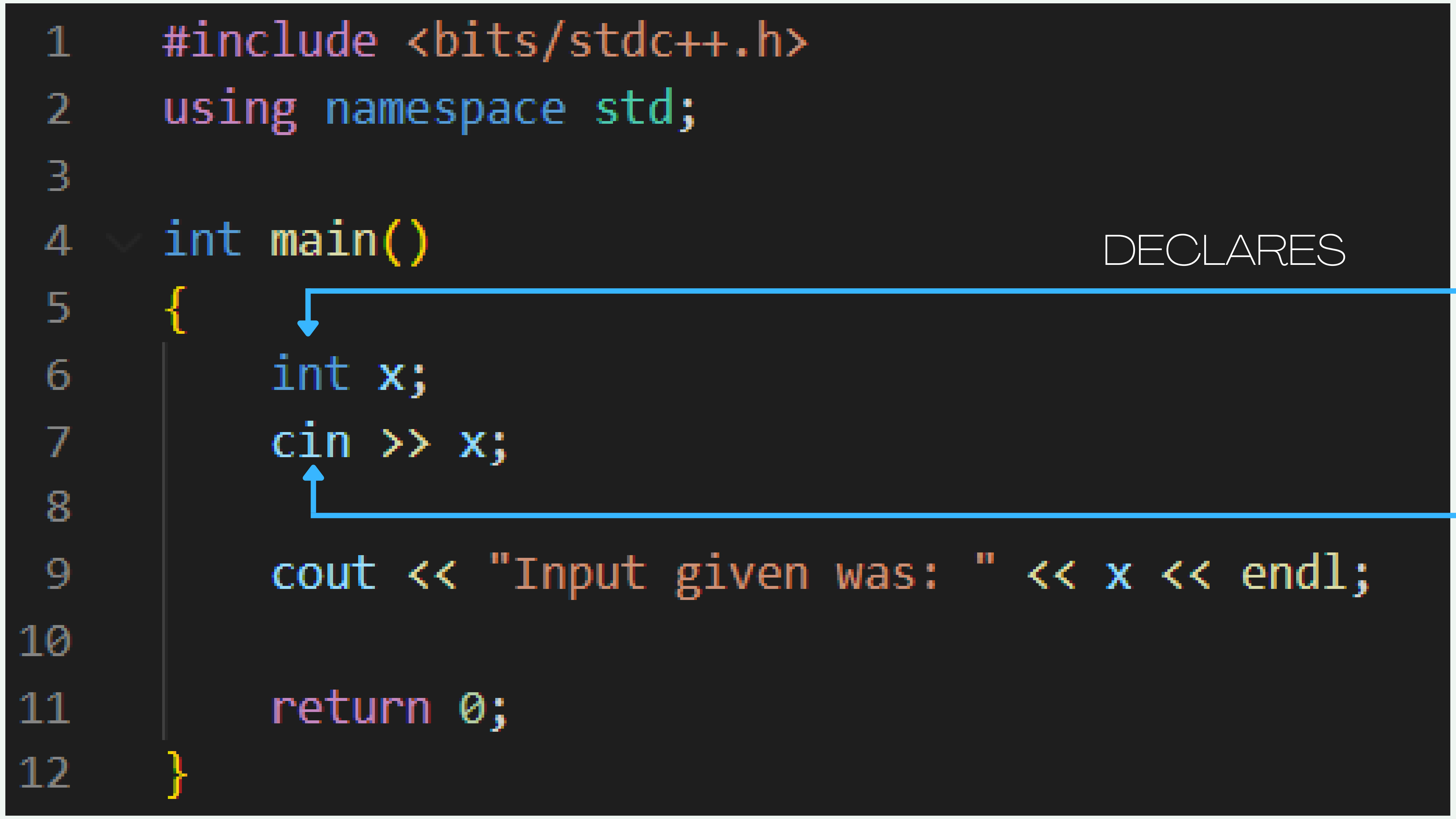
OUR FIRST PROGRAM



```
#include <bits/stdc++.h>
using namespace std;

int main()
{
    cout << "Hello World!" << endl;

    return 0;
}
```

BASIC INPUT-OUTPUT

WE'LL SEE WHAT'S A VARIABLE IN THE NEXT SLIDE



```
#include <bits/stdc++.h>
using namespace std;

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

    cout << "Input given was: " << x << endl;

    return 0;
}
```

DECLARES VARIABLE X THAT STORES INTEGER

TAKES IN AN INTEGER INPUT AND STORES IN X

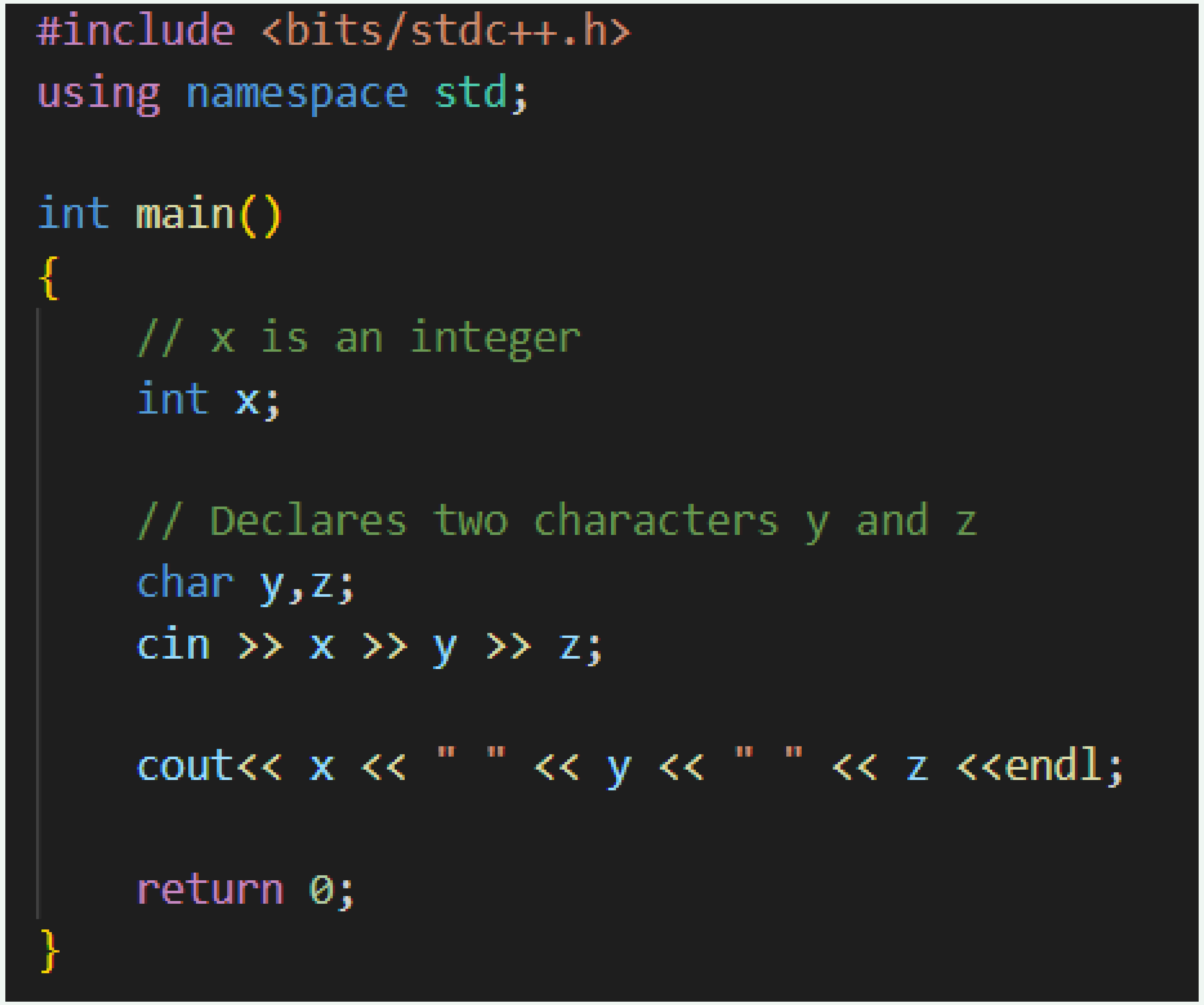
VARIABLES

Variables are containers storing data values.

Variables can store the following types of data like:-

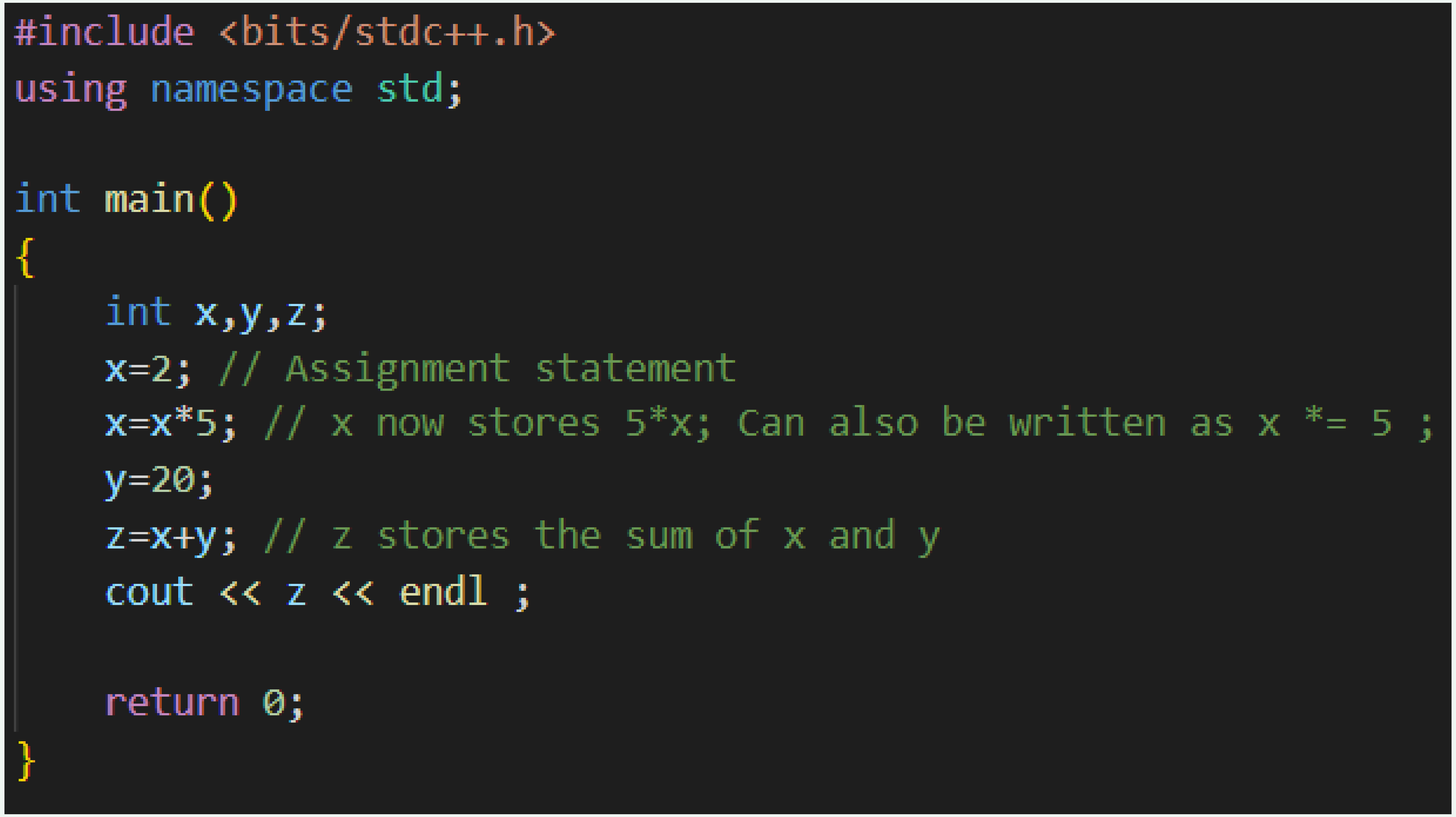
- Integer (Eg: 24, -7, 0 etc)
- Float (Ex: 3.05,-17.10, 1.00)
- Character (Ex: 'A','z','4')
- Boolean (0-false and 1-true)
- String ("Hello", "I am 12 years old")

DECLARING & OUTPUTTING VARIABLES



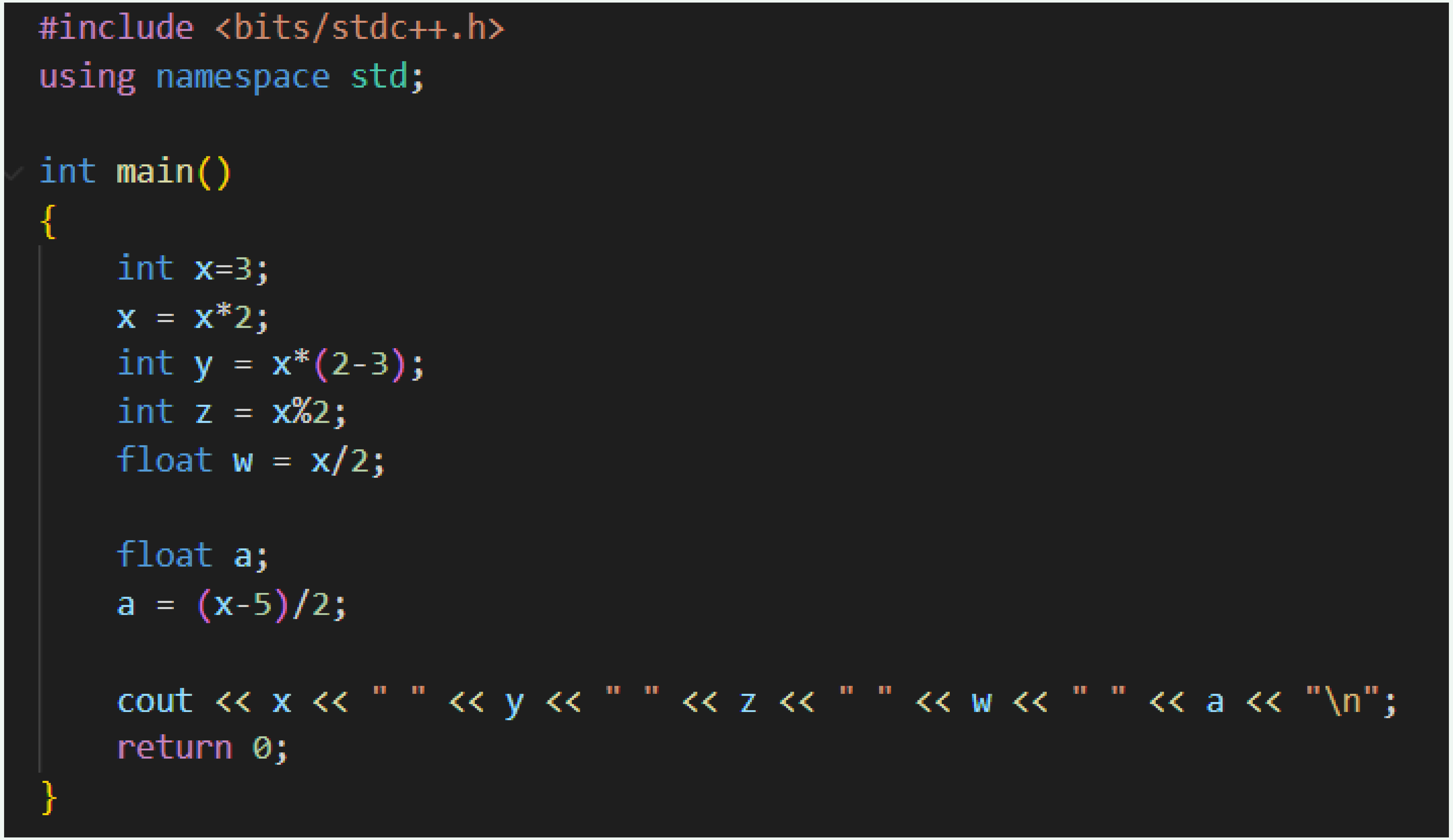
```cpp
#include <bits/stdc++.h>
using namespace std;

int main()
{
    // x is an integer
    int x;

    // Declares two characters y and z
    char y,z;
    cin >> x >> y >> z;

    cout<< x << " " << y << " " << z <<endl;

    return 0;
}
```

ASSIGNMENT AND ARITHMETIC



```
#include <bits/stdc++.h>
using namespace std;

int main()
{
    int x,y,z;
    x=2; // Assignment statement
    x=x*5; // x now stores 5*x; Can also be written as x *= 5 ;
    y=20;
    z=x+y; // z stores the sum of x and y
    cout << z << endl ;

    return 0;
}
```

EXAMPLE



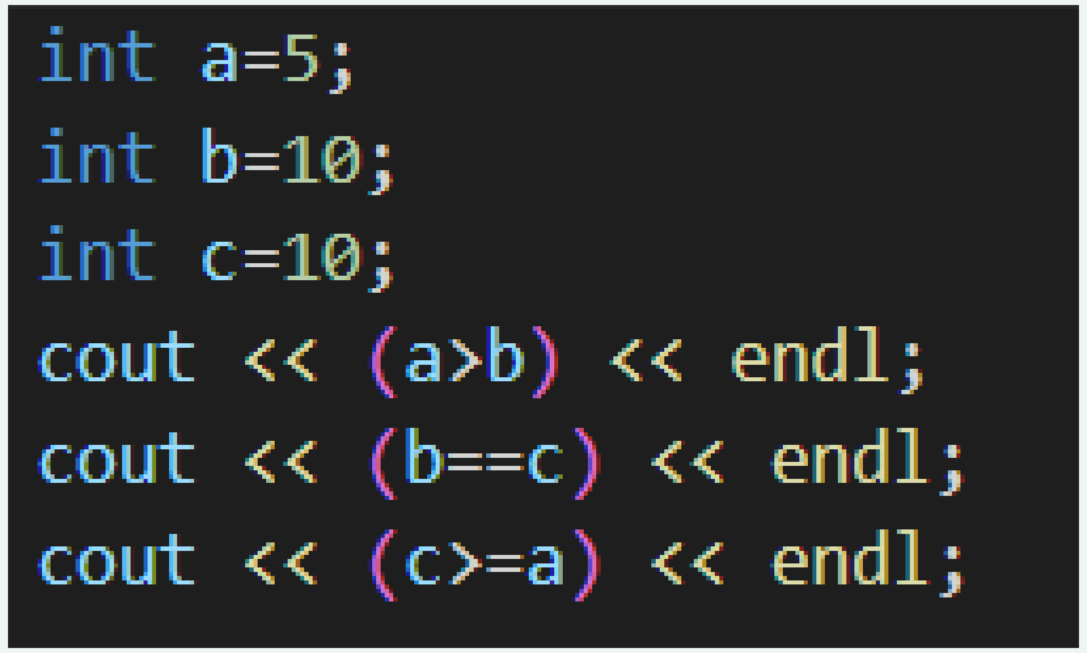
```cpp
#include <bits/stdc++.h>
using namespace std;

int main()
{
    int x=3;
    x = x*2;
    int y = x*(2-3);
    int z = x%2;
    float w = x/2;

    float a;
    a = (x-5)/2;

    cout << x << " " << y << " " << z << " " << w << " " << a << "\n";
    return 0;
}
```

RELATIONAL OPERATORS

The Relational operators are used to check the relationship between two operands.

- Equal to operator (==): This operator is used to check whether the two operands are equal or not. If they are equal, it returns true(1); otherwise, it returns false(0).
- Not Equal to operator (!=): This operator is used to check whether the two operands are equal or not. It returns true(1) if the left operand is not equal to the right operand; otherwise, it returns false(0).
- Similarly : >, <, >=, <=

Let's Try



```
int a=5;
int b=10;
int c=10;
cout << (a>b) << endl;
cout << (b==c) << endl;
cout << (c>=a) << endl;
```

LOGICAL OPERATORS

They operate on two boolean values, and return true or false as result.

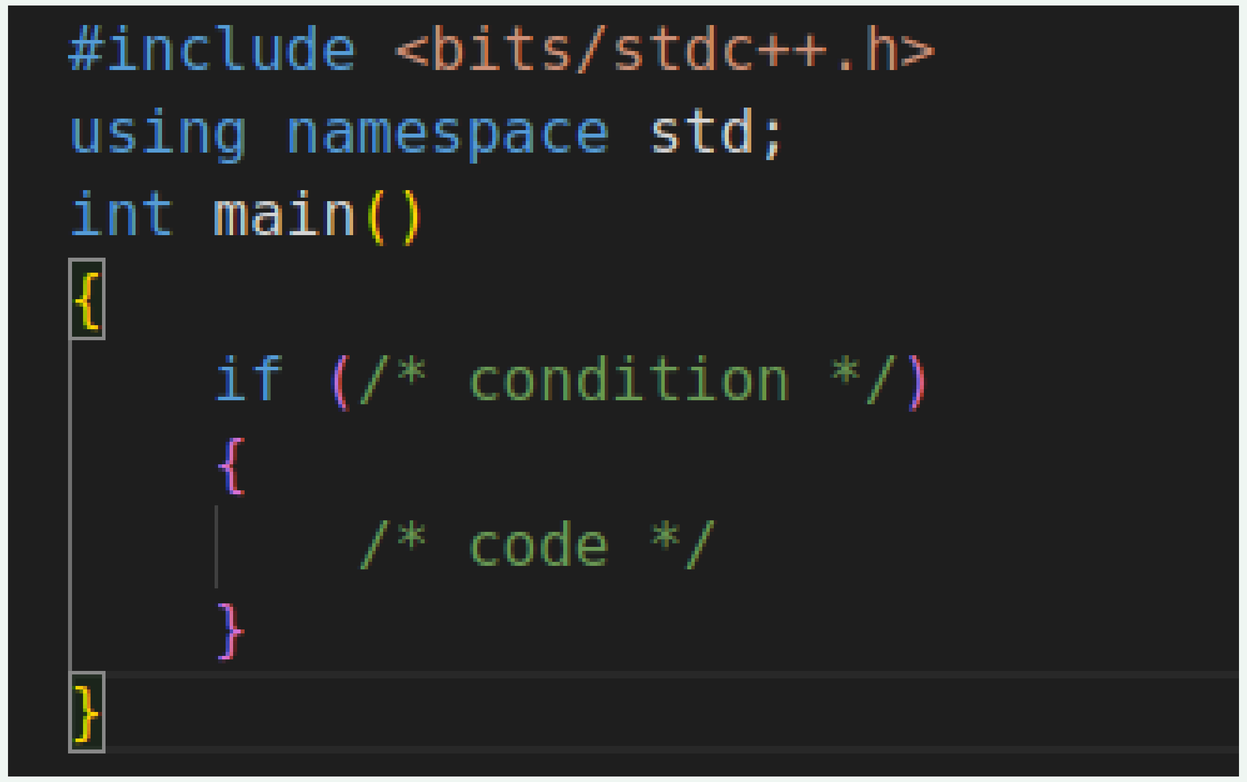
Operator	Description	Syntax
&&	Logical AND. True if both operands are true.	x && y
||	Logical OR. True if either of the operands are true	x | | y
!	Logical NOT. True if operand is false	! x

CONDITIONAL STATEMENTS

IF STATEMENT

The if keyword is used to execute a statement or block, if and only if, a condition is fulfilled.

Note: If we do not place the curly braces '{' and '}' after if(condition), then by default, the if statement will consider only the first immediate statement below it inside its block.

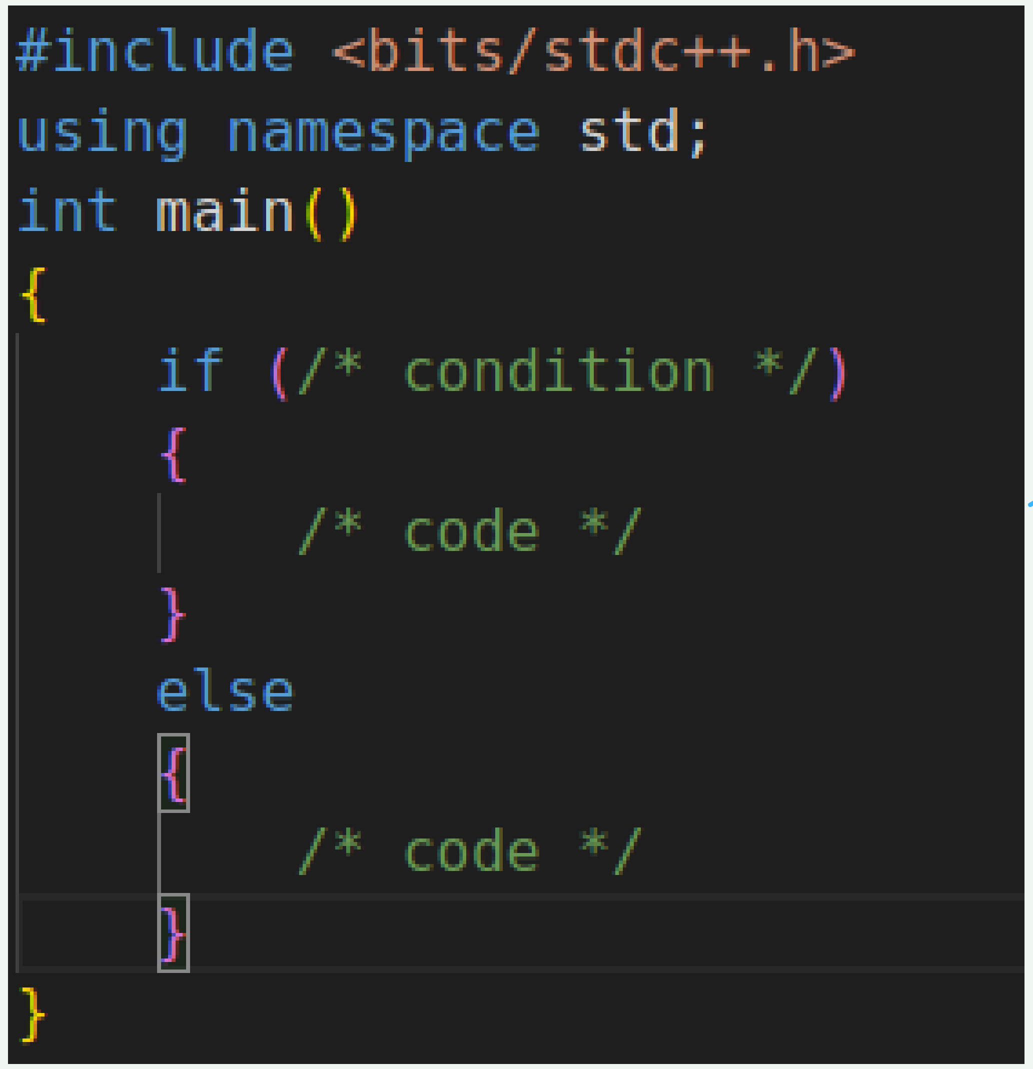


```
#include <bits/stdc++.h>
using namespace std;
int main()
{
    if (/* condition */)
    {
        /* code */
    }
}
```

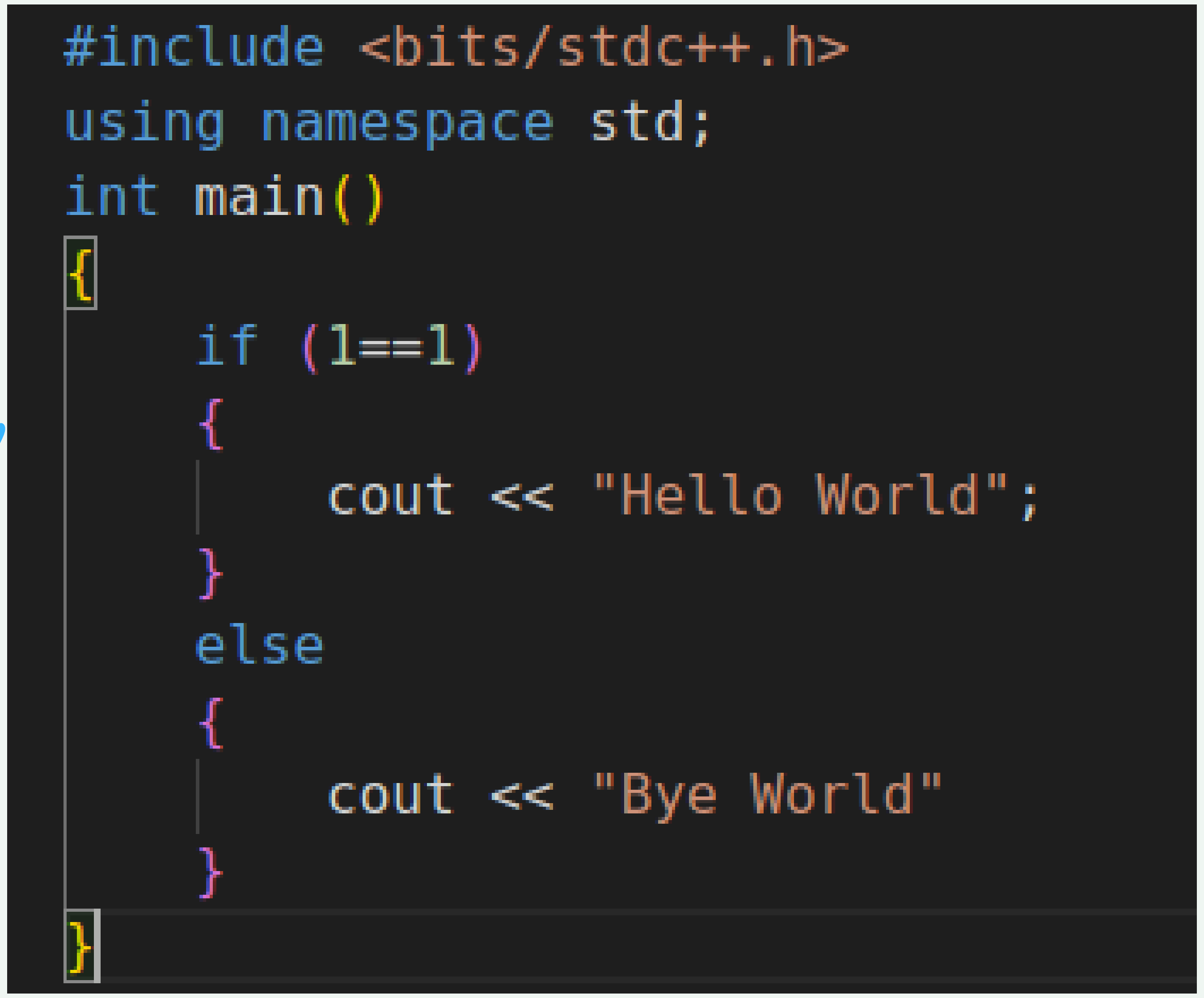
A condition is any expression which returns a boolean value as an output.

ELSE STATEMENT

The "ELSE" keyword is used to execute a statement or block, when the "if" condition is false

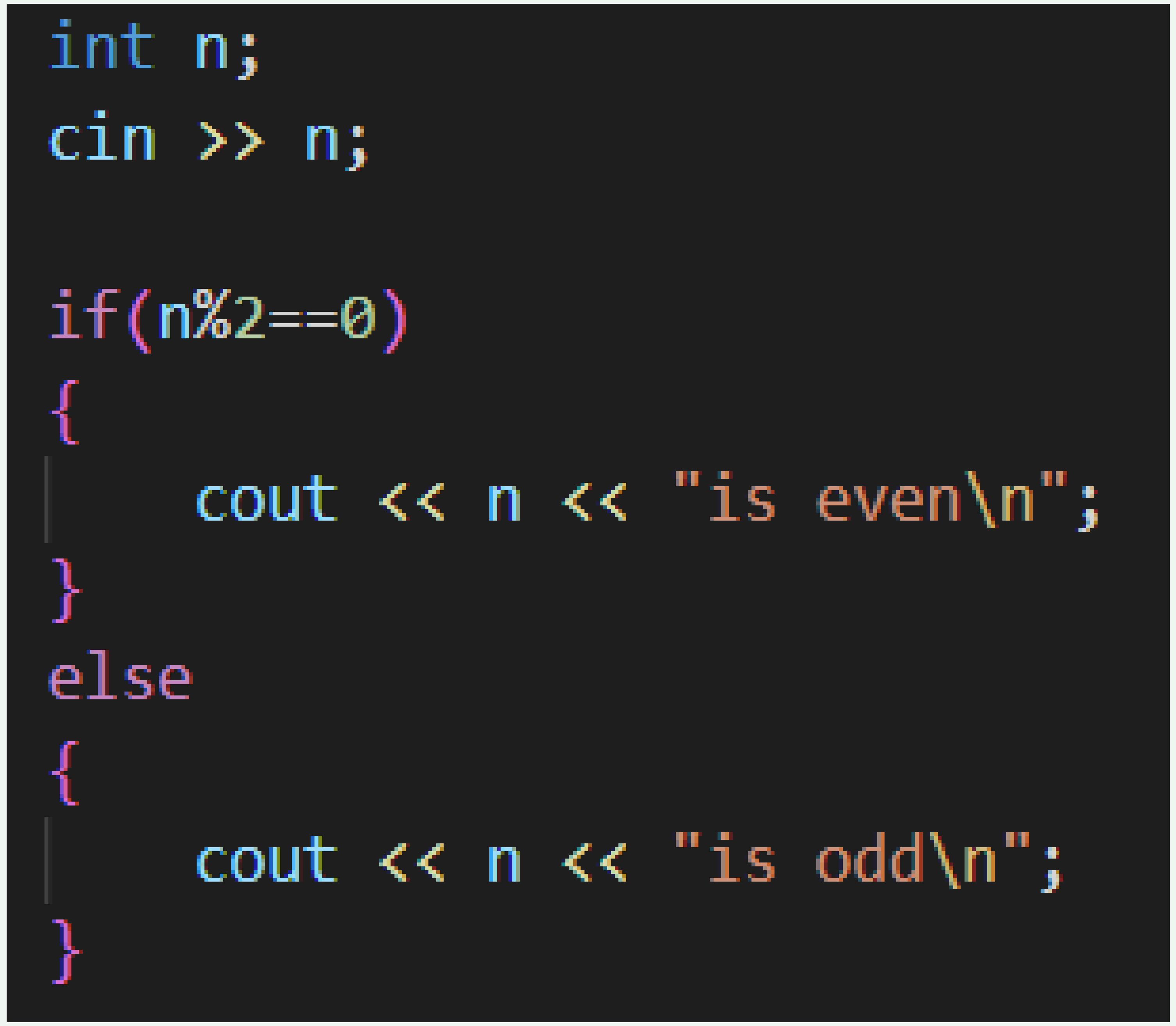


```cpp
#include <bits/stdc++.h>
using namespace std;
int main()
{
    if (/* condition */)
    {
        /* code */
    }
    else
    {
        /* code */
    }
}
```



```cpp
#include <bits/stdc++.h>
using namespace std;
int main()
{
    if (1==1)
    {
        cout << "Hello World";
    }
    else
    {
        cout << "Bye World"
    }
}
```

DETERMINING IF AN INTEGER IS ODD OR EVEN.

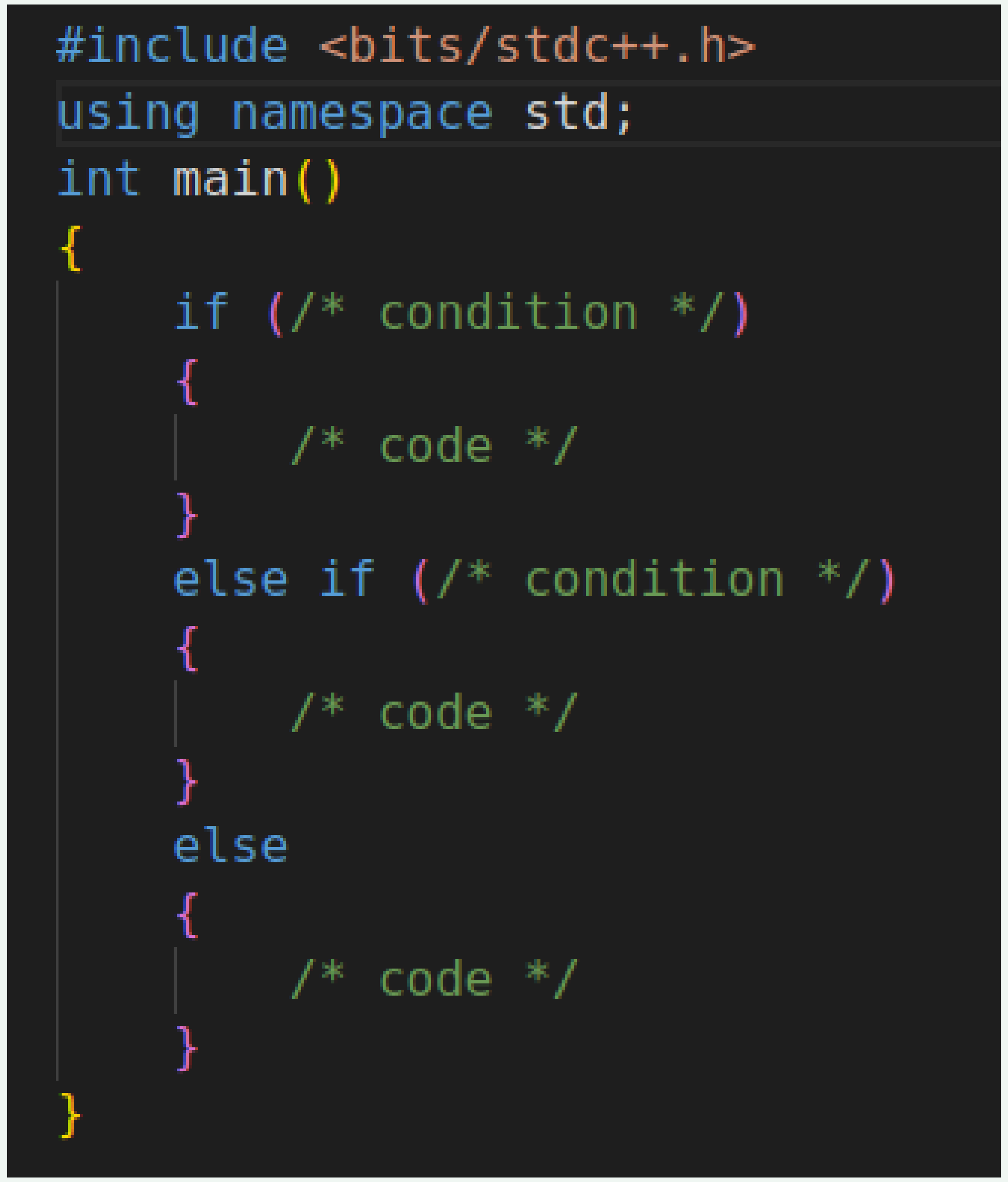


```
int n;
cin >> n;

if(n%2==0)
{
    cout << n << "is even\n";
}
else
{
    cout << n << "is odd\n";
}
```

LET'S SOLVE A PROBLEM

Watermelon

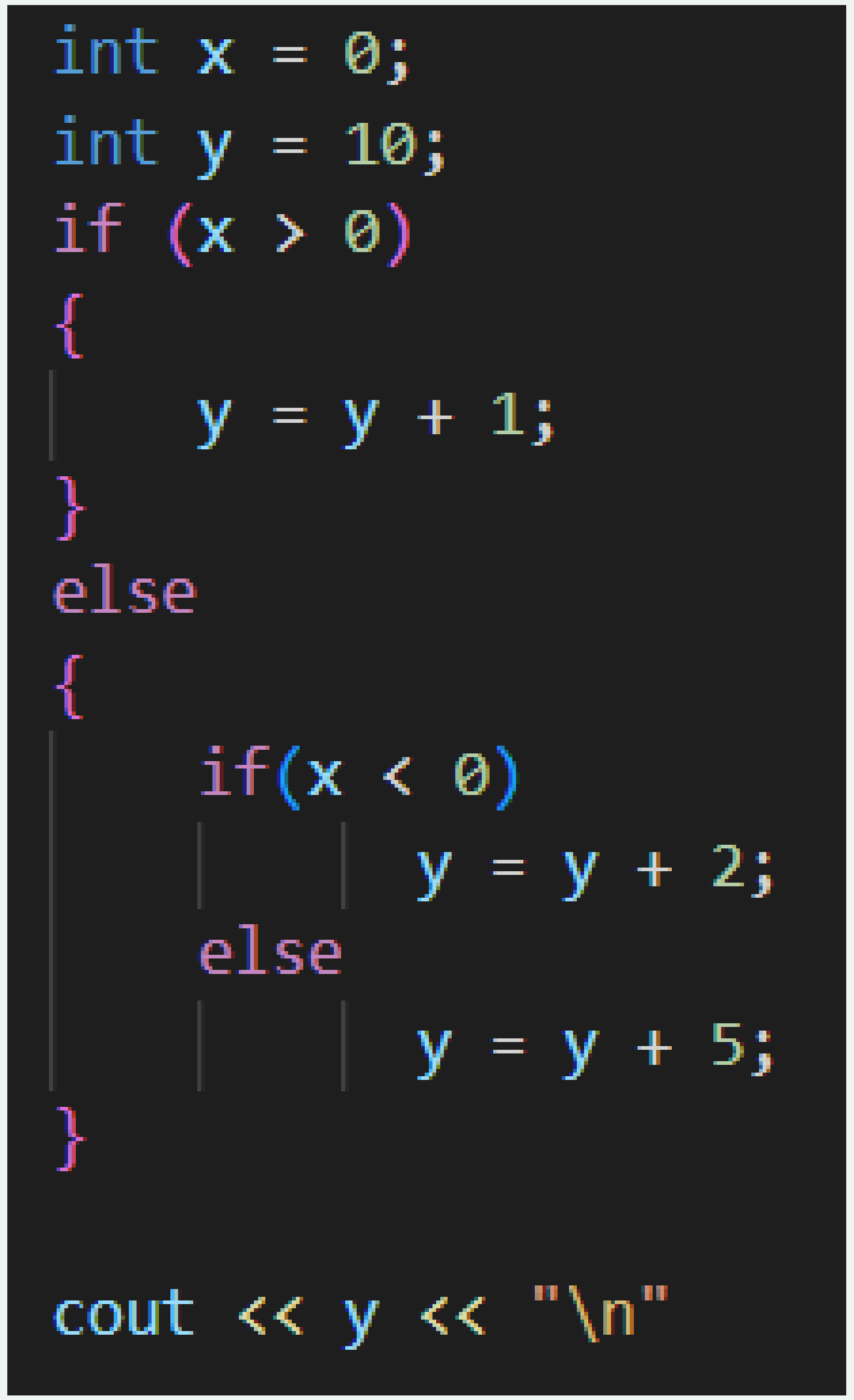
MULTIPLE CONDITIONS?



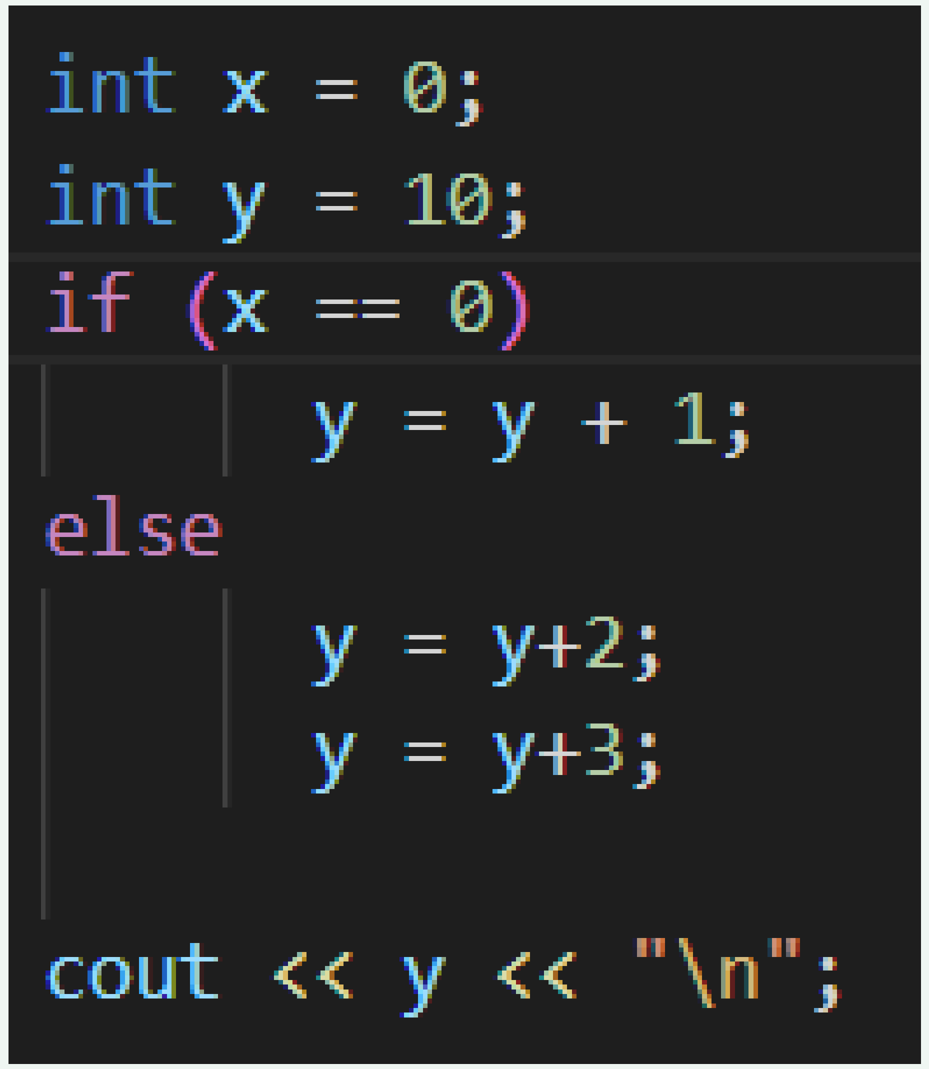
```
#include <bits/stdc++.h>
using namespace std;
int main()
{
    if (/* condition */)
    {
        /* code */
    }
    else if (/* condition */)
    {
        /* code */
    }
    else
    {
        /* code */
    }
}
```

- If the condition under "if" is true, the first block gets executed
- Say one wants to add multiple conditions, we do this by adding an "else if" block
- an "else if" block checks for the condition if and only if the previous "if" and "else if" blocks are false

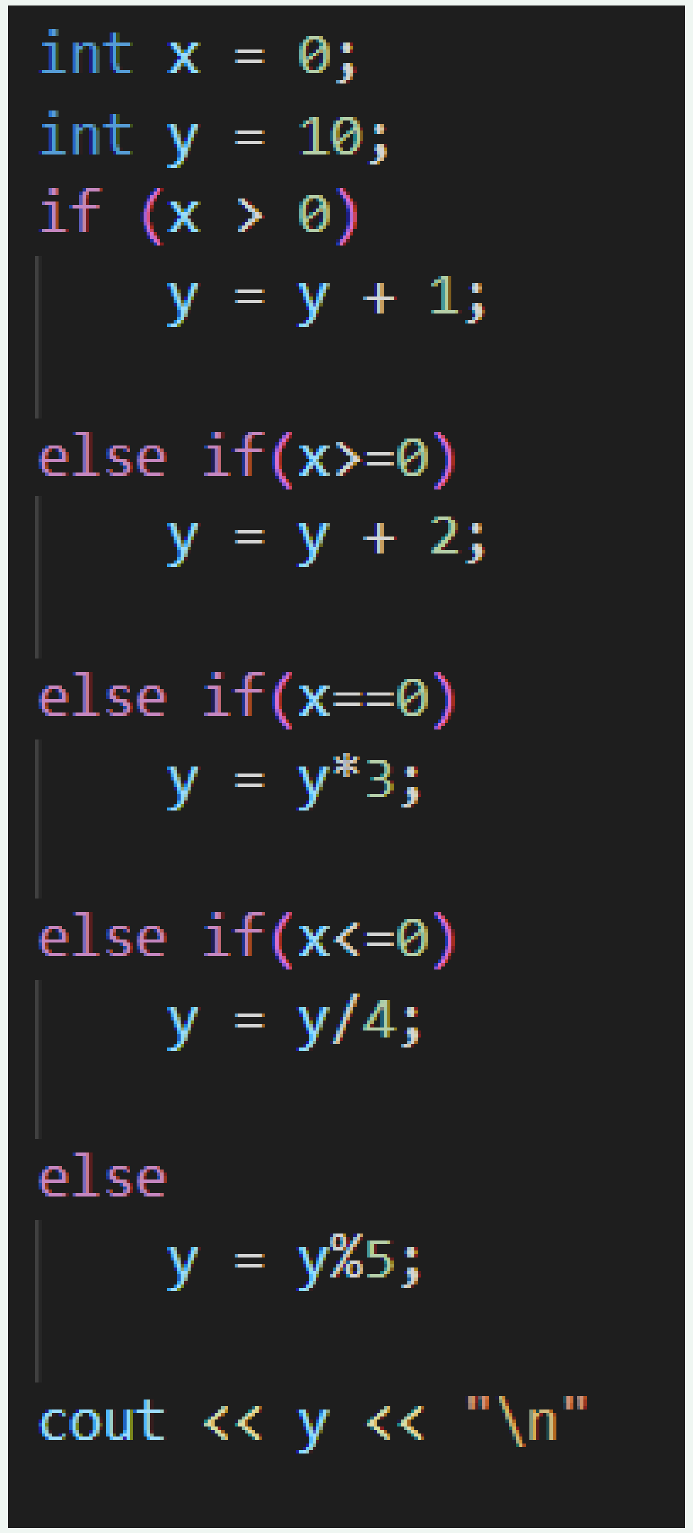
WHAT WILL BE THE OUTPUT?



```cpp
int x = 0;
int y = 10;
if (x > 0)
{
    y = y + 1;
}
else
{
    if(x < 0)
        y = y + 2;
    else
        y = y + 5;
}

cout << y << "\n"
```

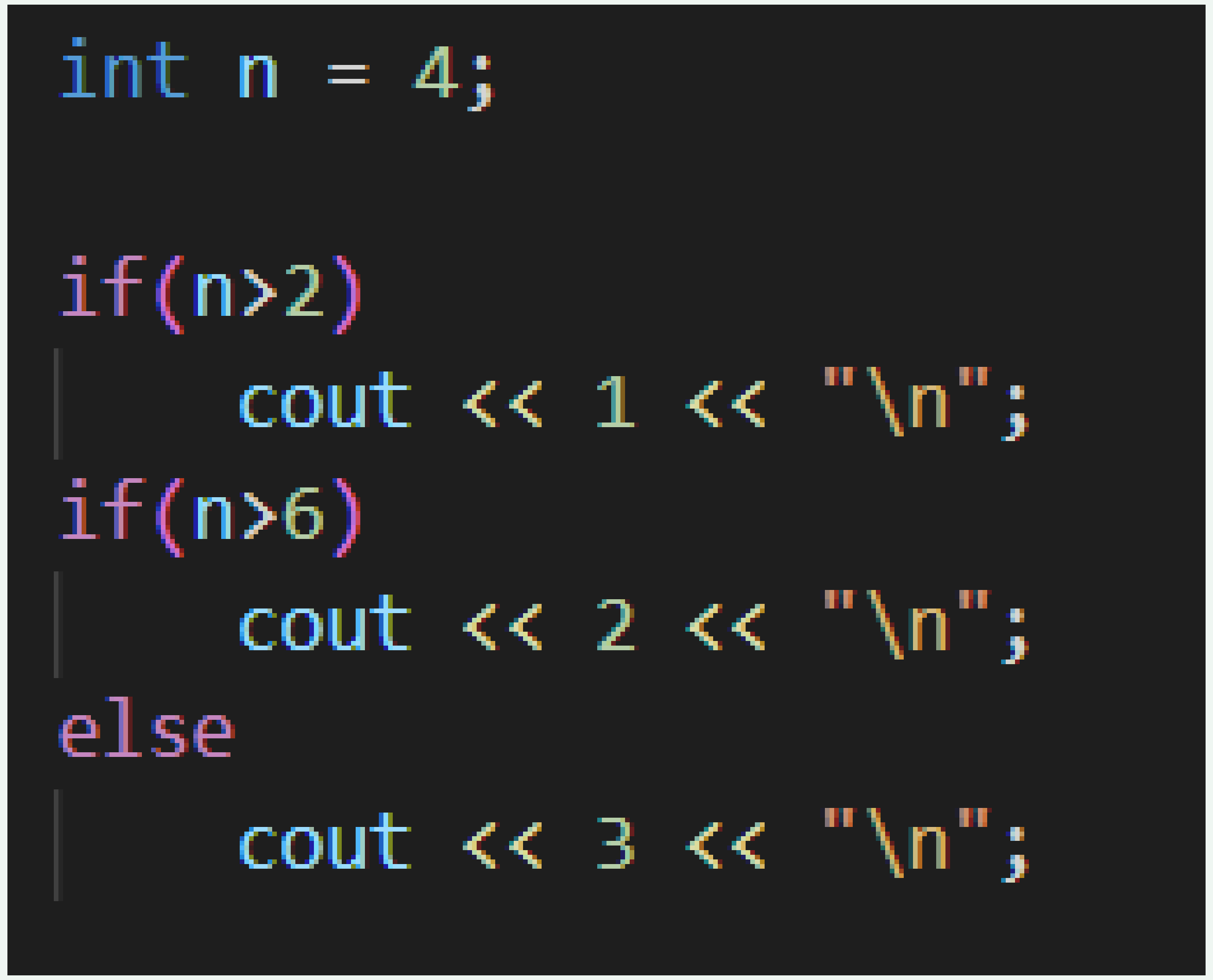


```cpp
int x = 0;
int y = 10;
if (x == 0)
        y = y + 1;
else
        y = y+2;
        y = y+3;

cout << y << "\n";
```

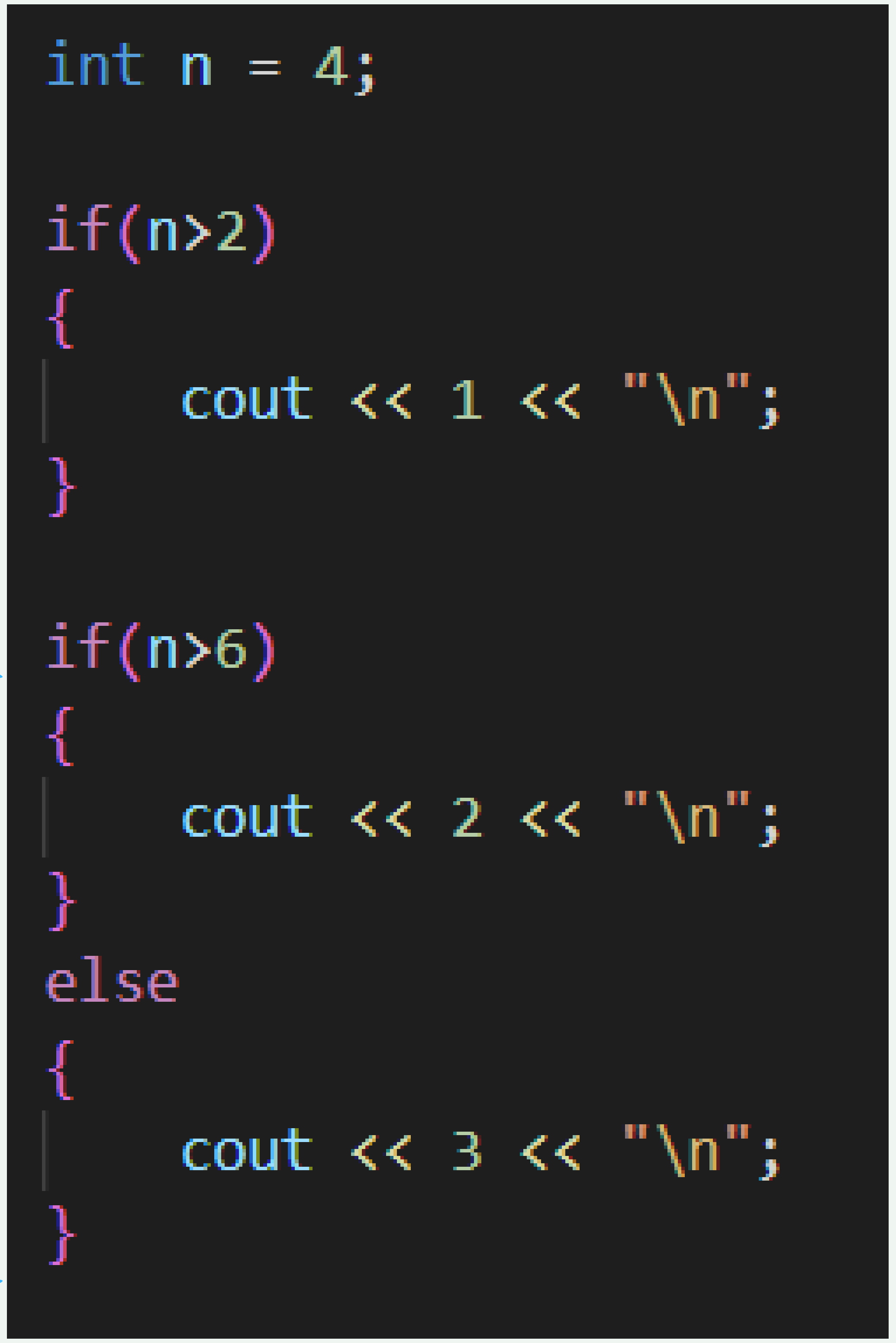


```cpp
int x = 0;
int y = 10;
if (x > 0)
    y = y + 1;

else if(x>=0)
    y = y + 2;

else if(x==0)
    y = y*3;

else if(x<=0)
    y = y/4;

else
    y = y%5;

cout << y << "\n"
```

DANGLING IF-ELSE



```cpp
int n = 4;

if(n>2)
    cout << 1 << "\n";
if(n>6)
    cout << 2 << "\n";
else
    cout << 3 << "\n";
```

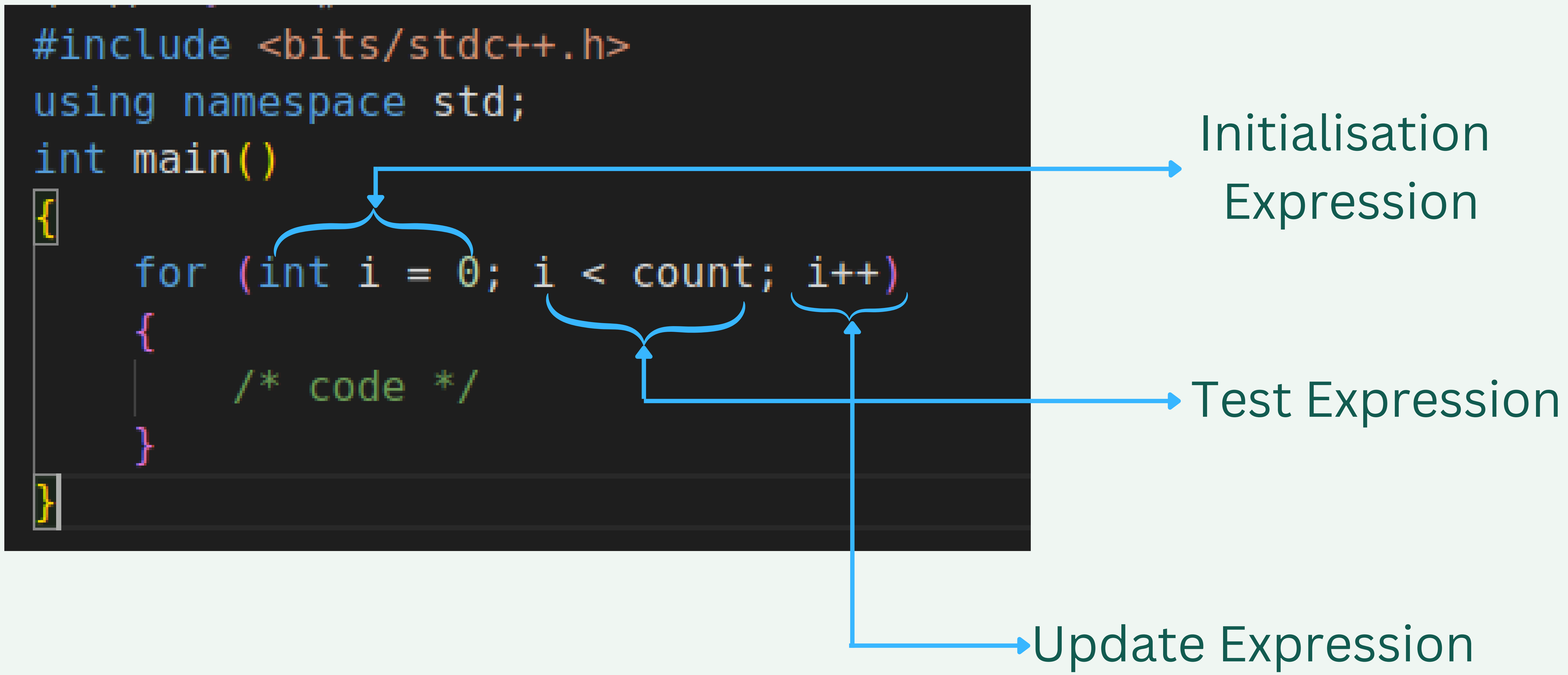


```cpp
int n = 4;

if(n>2)
{
    cout << 1 << "\n";
}

if(n>6)
{
    cout << 2 << "\n";
}
else
{
    cout << 3 << "\n";
}
```

The else statement here corresponds to the latest unresolved if statement i.e. the else here belongs to the if(n>6).

LOOPS

THE "FOR" LOOP



```
#include <bits/stdc++.h>
using namespace std;
int main()
{
    for (int i = 0; i < count; i++)
    {
        /* code */
    }
}
```

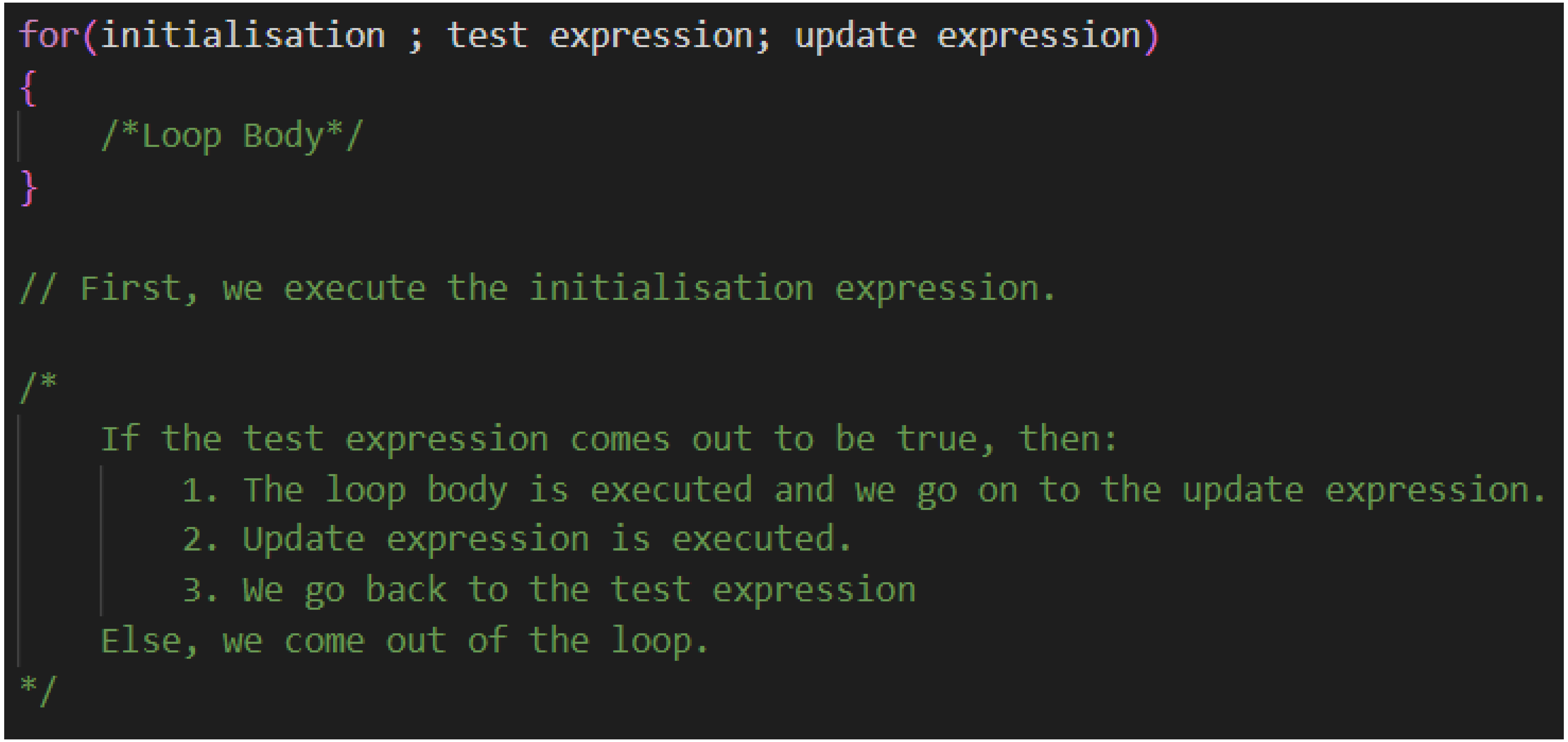
THE "FOR" LOOP

- **Initialisation** - This statement is executed first. It is also done only once, where the loop variables are initialized to some value.

- **Test Expression** - This expression involves testing the condition.. If the condition evaluates to true then the body of the loop is executed.. If the condition evaluates to false then we exit from the loop, and the control jumps to the statements following the loop.

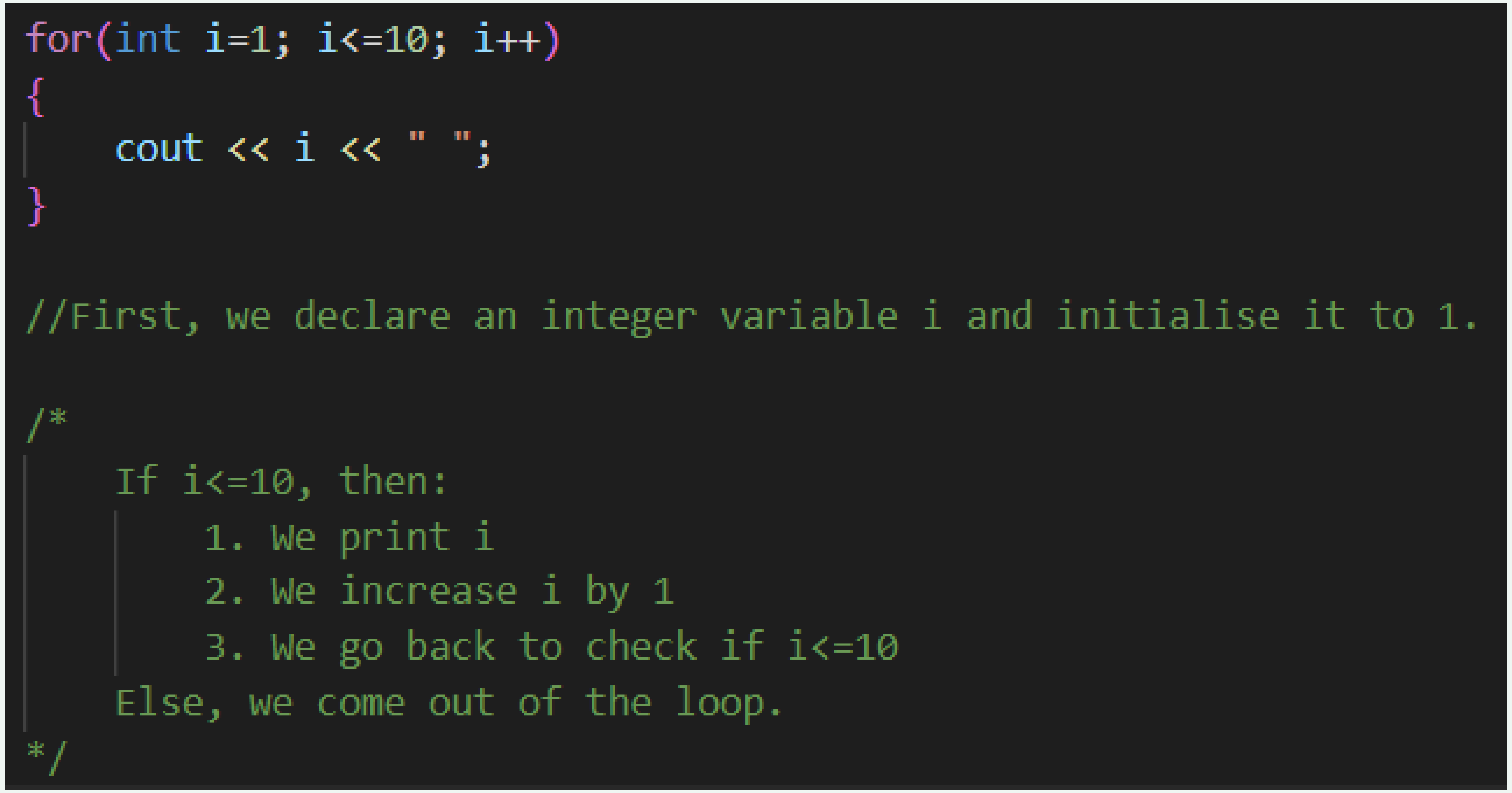
- **Update** - After the execution of the body of the loop, the control jumps to the update expression where the loop variables are updated by some value. After update statement, we check the test expression.

THE "FOR" LOOP



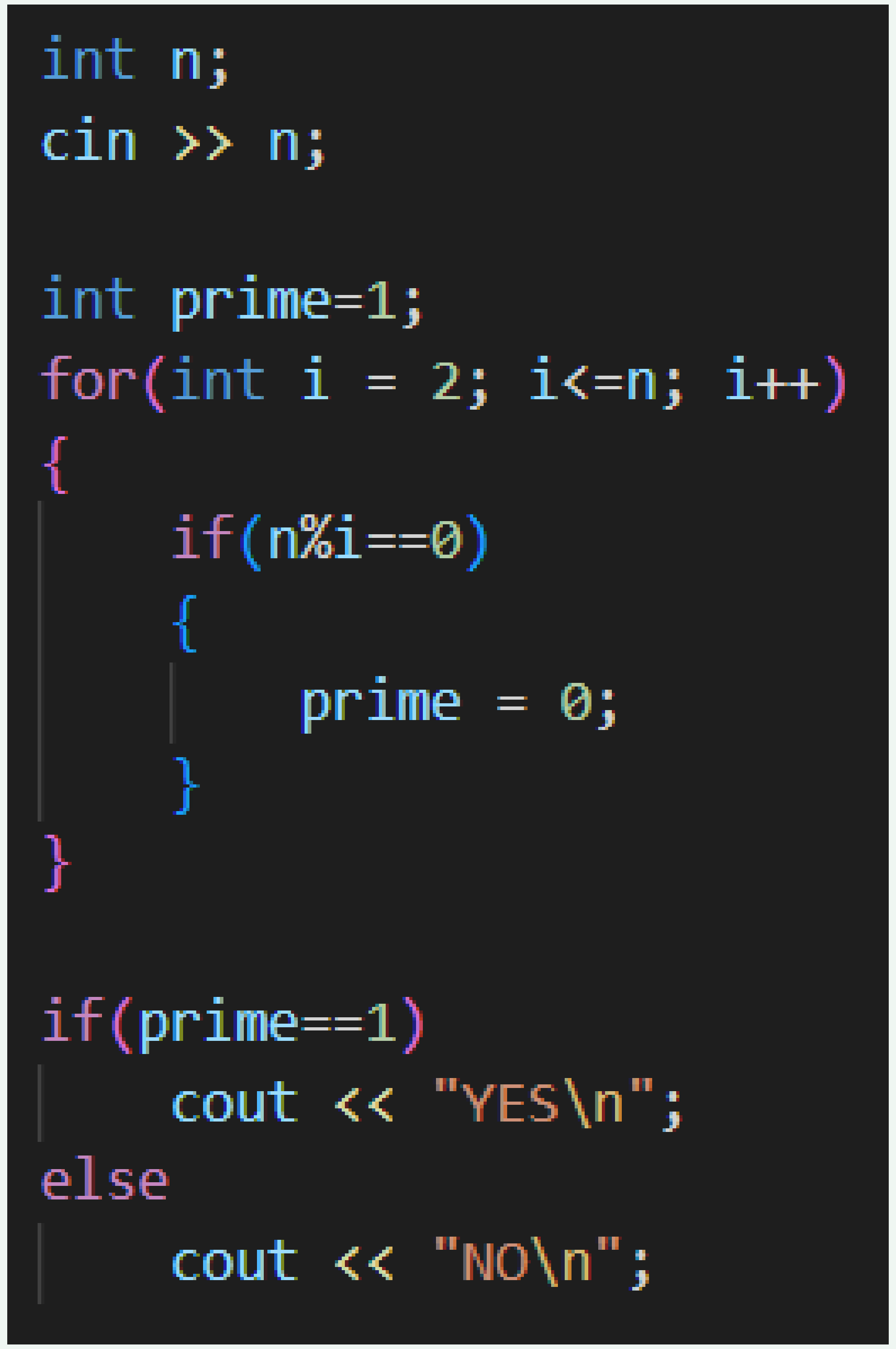
```
for(initialisation ; test expression; update expression)
{
    /*Loop Body*/
}

// First, we execute the initialisation expression.

/*
    If the test expression comes out to be true, then:
        1. The loop body is executed and we go on to the update expression.
        2. Update expression is executed.
        3. We go back to the test expression
    Else, we come out of the loop.
*/
```

PRINTING NUMBERS FROM 1 TO 10



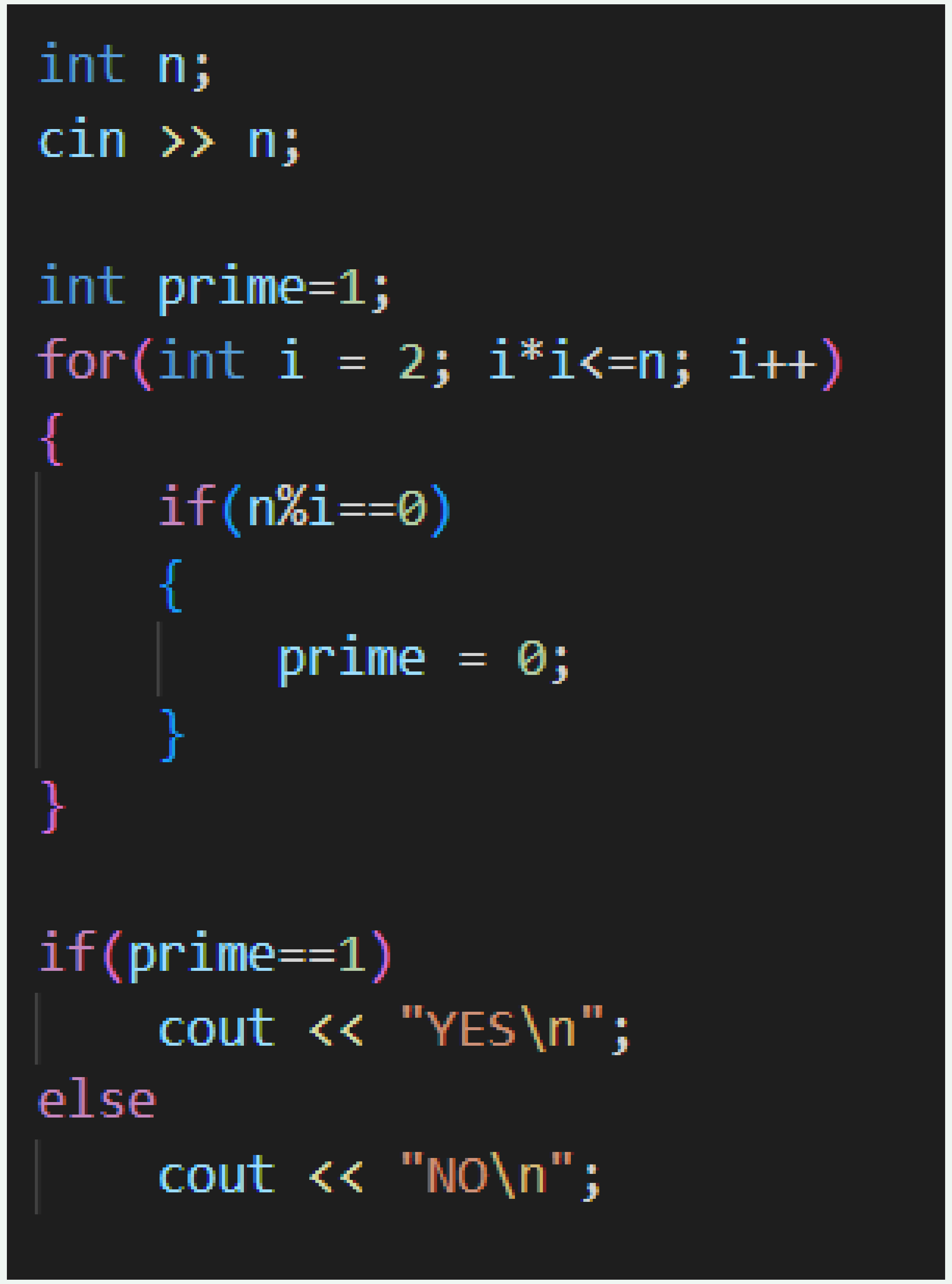
```
for(int i=1; i<=10; i++)
{
    cout << i << " ";
}

//First, we declare an integer variable i and initialise it to 1.

/*
    If i<=10, then:
        1. We print i
        2. We increase i by 1
        3. We go back to check if i<=10
    Else, we come out of the loop.
*/
```

ARE YOU PRIME?



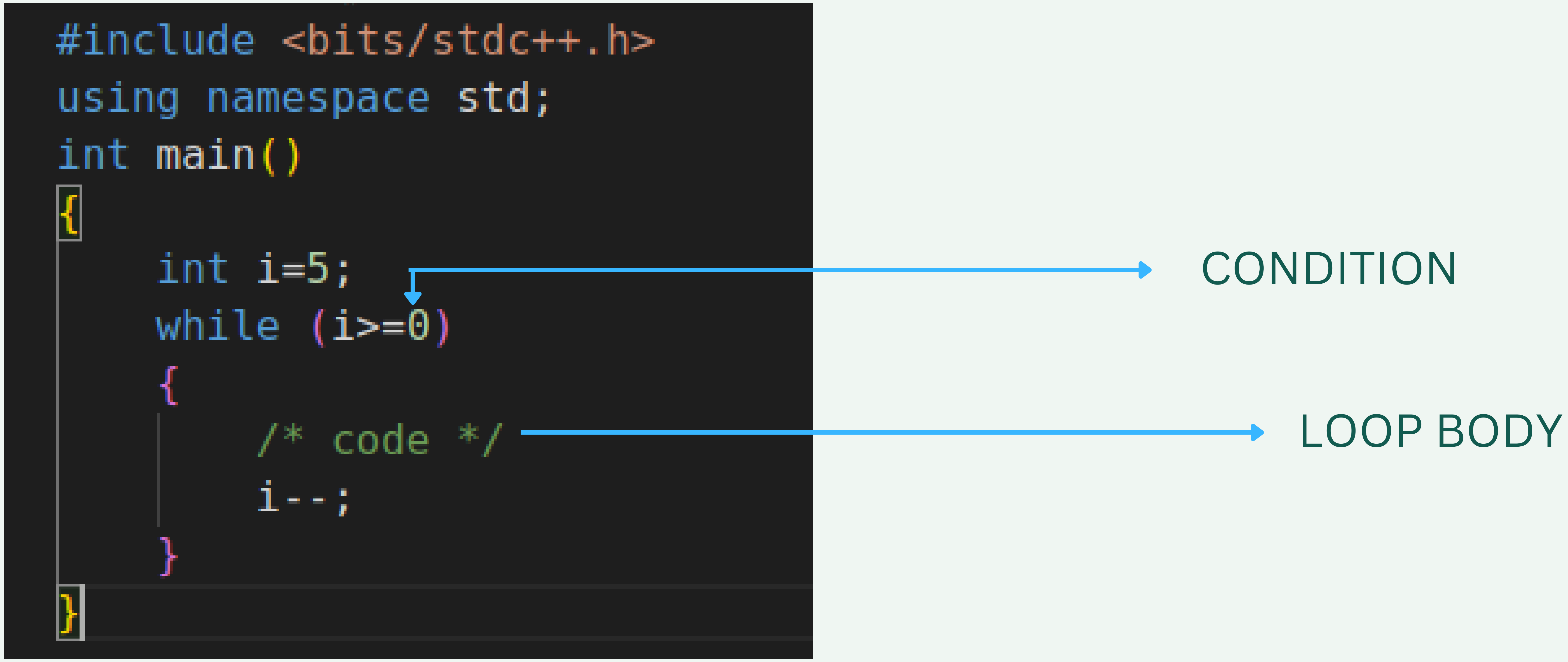
```
int n;
cin >> n;

int prime=1;
for(int i = 2; i<=n; i++)
{
    if(n%i==0)
    {
        prime = 0;
    }
}

if(prime==1)
    cout << "YES\n";
else
    cout << "NO\n";
```

ARE YOU PRIME?(Optimised)



```
int n;
cin >> n;

int prime=1;
for(int i = 2; i*i<=n; i++)
{
    if(n%i==0)
    {
        prime = 0;
    }
}

if(prime==1)
    cout << "YES\n";
else
    cout << "NO\n";
```

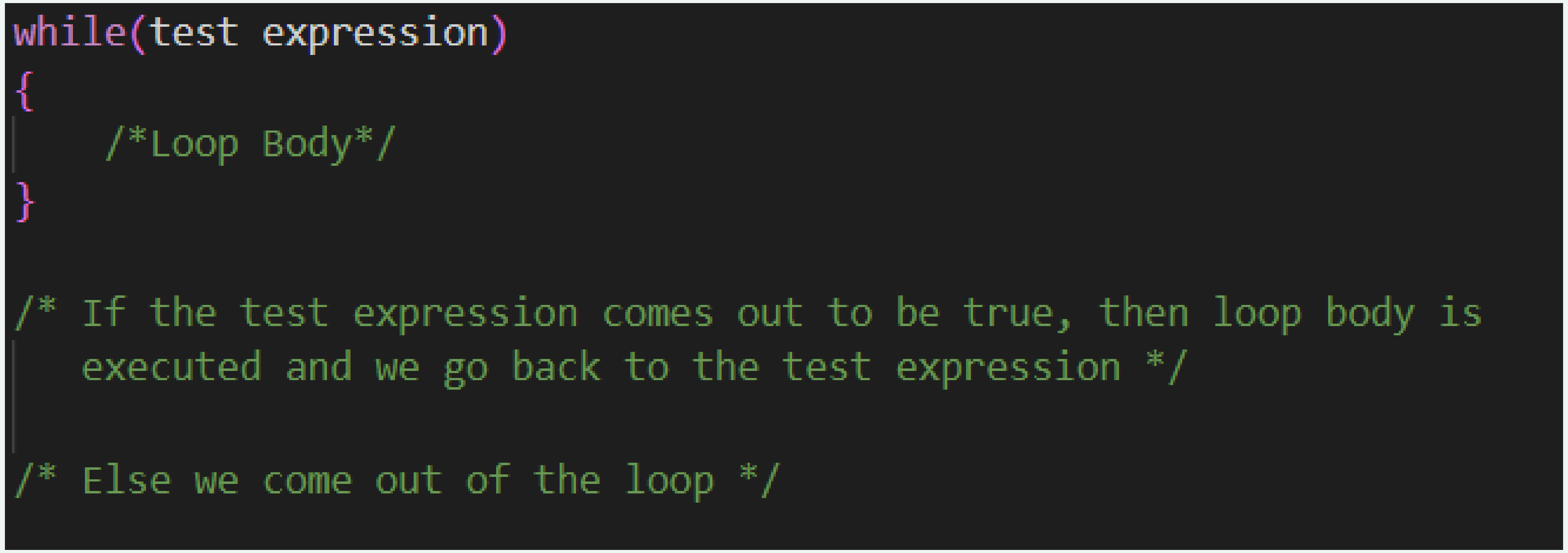
THE "WHILE" LOOP



```
#include <bits/stdc++.h>
using namespace std;
int main()
{
    int i=5;
    while (i>=0)
    {
        /* code */
        i--;
    }
}
```

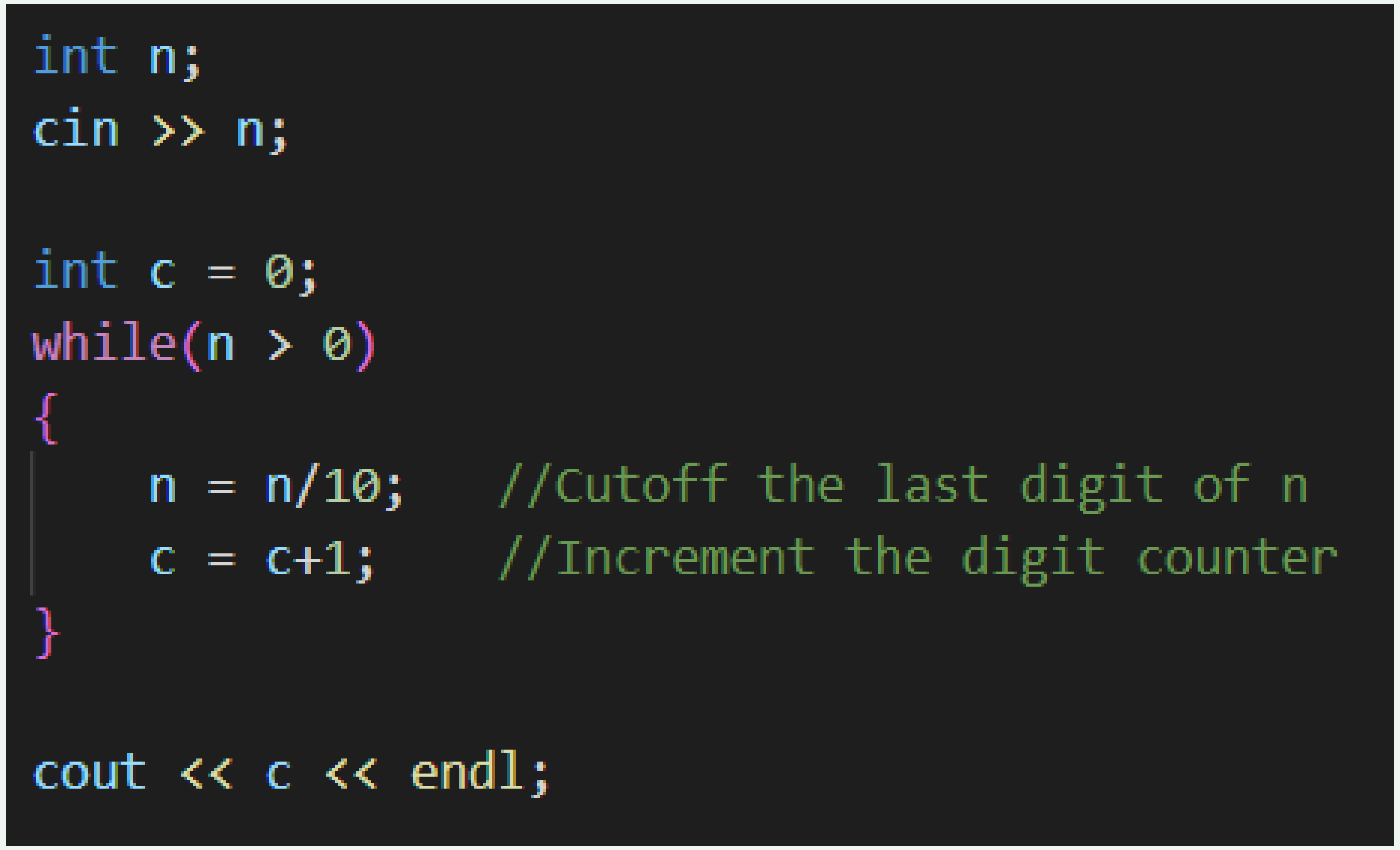
THE "WHILE" LOOP

The while loop helps us execute a set of statements as long as a given condition is true



```
while(test expression)
{
    /*Loop Body*/
}

/* If the test expression comes out to be true, then loop body is
   executed and we go back to the test expression */

/* Else we come out of the loop */
```

COUNTING THE NUMBER OF DIGITS



```
int n;
cin >> n;

int c = 0;
while(n > 0)
{
    n = n/10;   //Cutoff the last digit of n
    c = c+1;    //Increment the digit counter
}

cout << c << endl;
```

CHOCOLATE PROBLEM

Vinit likes chocolates, and he has recently won 100 Rupees in a CP contest. He decides to spend it all on chocolates. He goes to the nearby store to buy chocolates. 1 chocolate costs 5 Rupees here.
The owner of the shop supports the Swachh Bharat Mission, so for every 5 chocolate wrappers that Vinit gives him, the owner gives him a chocolate for free.

Assumptions:
1. Vinit eats the chocolate as soon as he buys it.
2. One chocolate gives 1 wrapper.

What is the maximum number of chocolates that Vinit can eat?
B1. If you have solved the first part, can you solve the same question if Vinit has 125 Rupees?
B2. Try the problem for 90 Rupees as well.
C. If you have done B as well, can you provide an algorithm if Vinit has 5N Rupees?

Solving the chocolate problem

100 Rupees - 24 chocolates.

- Vineet has 100 Rupees, so with that, he can buy 20 chocolates initially.
- Now with this, he gets 20 wrappers. Using these 20 wrappers, he can get 4 chocolates from the owner.
- Vinit can’t get any more chocolates as he has only 4 wrappers with him.

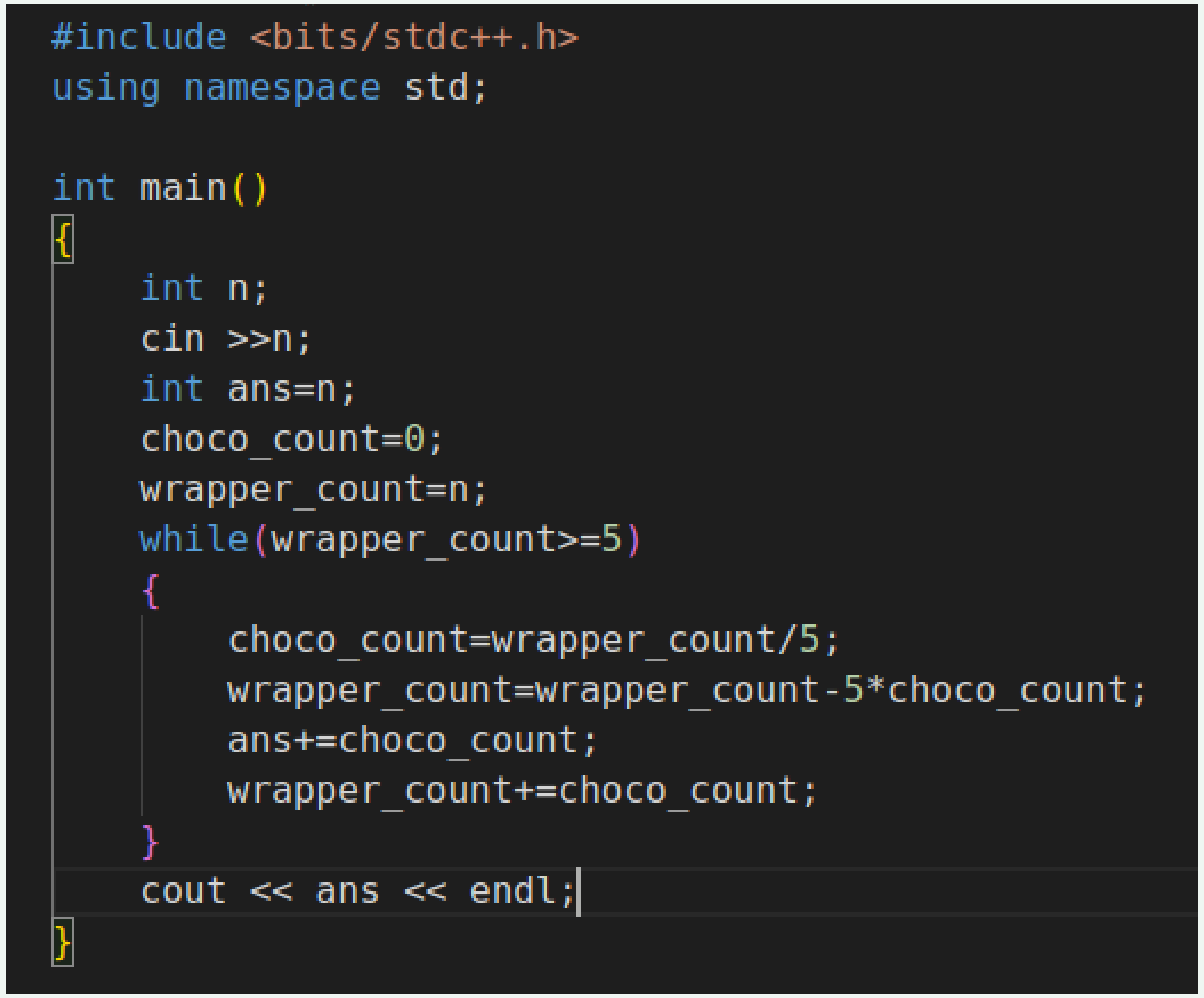
125 Rupees - 31 chocolates.

- Vineet has 125 Rupees, so with that, he can buy 25 chocolates initially.
- With this, he gets 25 wrappers. Using these 25 wrappers, he can get 5 chocolates from the
- owner.
- With this, he gets 5 wrappers. He can get 1 more chocolate from the owner.

90 Rupees - 22 chocolates.

- He can buy 18 chocolates initially.
- Now with this, he gets 18 wrappers. Using these 18 wrappers, he can get 3 chocolates from the owner (and he'll keep 3 wrappers with him).
- Now he has 6 wrappers with him. Using 5 of them, he can get one chocolate.
- In the end, he is left with 2 wrappers.

CHOCOLATE PROBLEM



```
#include <bits/stdc++.h>
using namespace std;

int main()
{
    int n;
    cin >>n;
    int ans=n;
    choco_count=0;
    wrapper_count=n;
    while(wrapper_count>=5)
    {
        choco_count=wrapper_count/5;
        wrapper_count=wrapper_count-5*choco_count;
        ans+=choco_count;
        wrapper_count+=choco_count;
    }
    cout << ans << endl;
}
```

Print these patterns!

1)
```
*
* *
* * *
* * * *
* * * * *
```

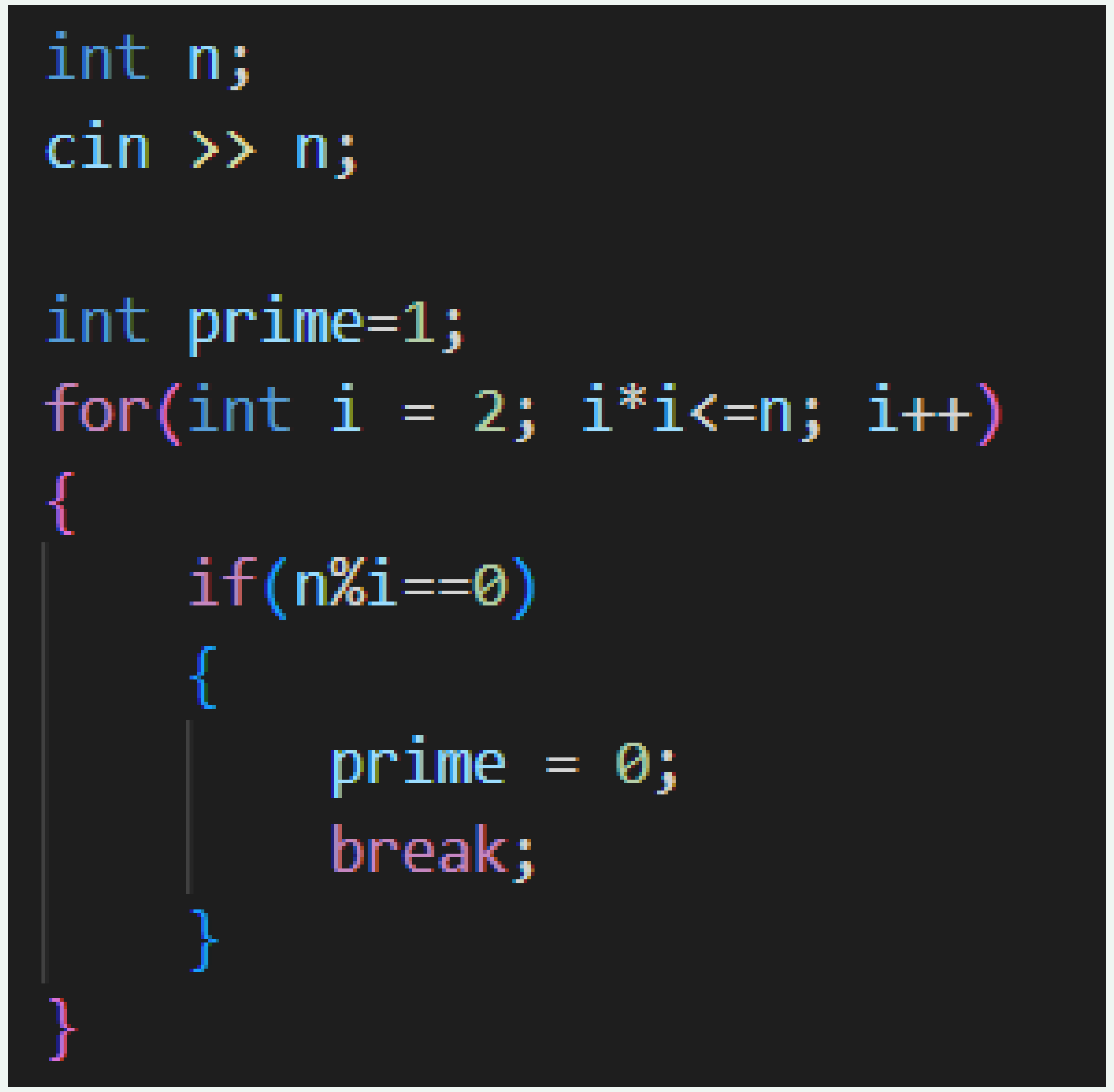
2)
```
* * * * *
* * * *
* * *
* *
*
```

3)
```
*   *   *
  *   *
*   *   *
  *   *
*   *   *
```

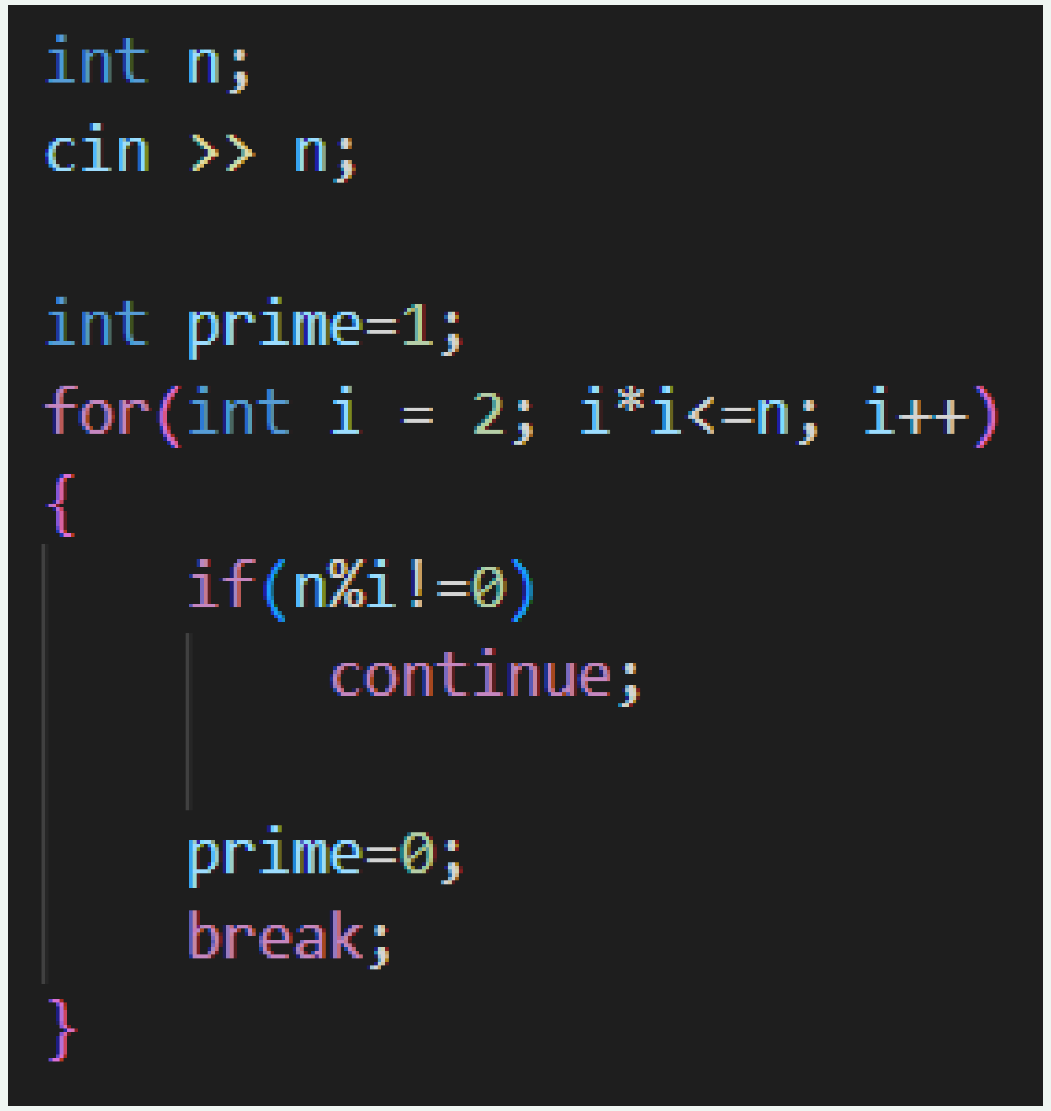
BREAK AND CONTINUE STATEMENTS

- Break and Continue are used to modify the flow of the loop
- The Break statement breaks out of the loop.
- Continue statement goes to the next iteration of the loop.

PRIMALITY TEST WITH BREAK AND CONTINUE



```cpp
int n;
cin >> n;

int prime=1;
for(int i = 2; i*i<=n; i++)
{
    if(n%i==0)
    {
        prime = 0;
        break;
    }
}
```



```cpp
int n;
cin >> n;

int prime=1;
for(int i = 2; i*i<=n; i++)
{
    if(n%i!=0)
        continue;

    prime=0;
    break;
}
```

ARRAYS

MOTIVATION

Let’s say that you want to store marks of all students in your class(say 50) and then perform some operations on it (like finding maximum, finding minimum, converting into percentage, sorting in decreasing order, etc).

Would you make all 50 separate variables as int mark1,mark2,mark3,...,mark50;?
and then do operation on each of them manually?

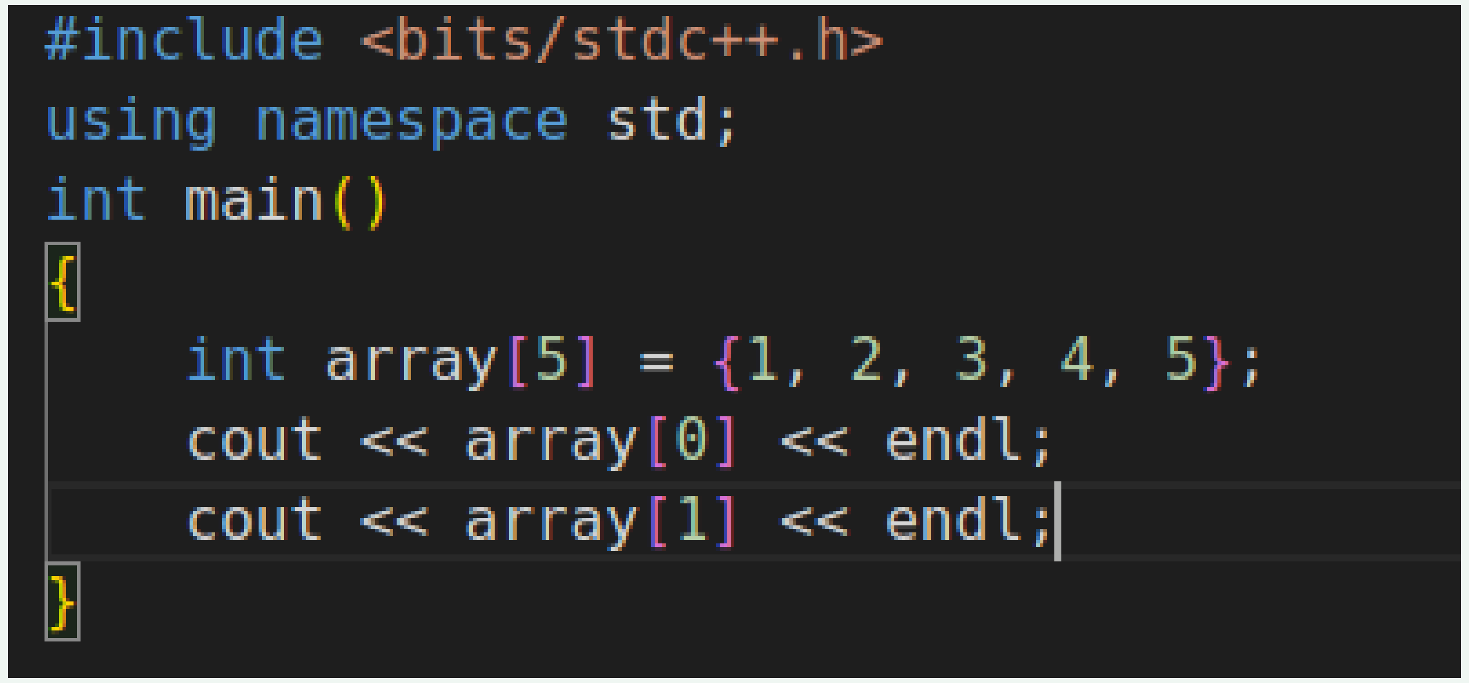
This is a very lengthy and tiring process. What if there was some way that we can assign all?

Arrays store some type of data(only one type in one array) in contiguous memory locations so that we can access any element easily.

We can access elements as a[i] where i is the index of the element you want to use.
See that indexing starts from 0.

ACCESSING ELEMENTS OF AN ARRAY

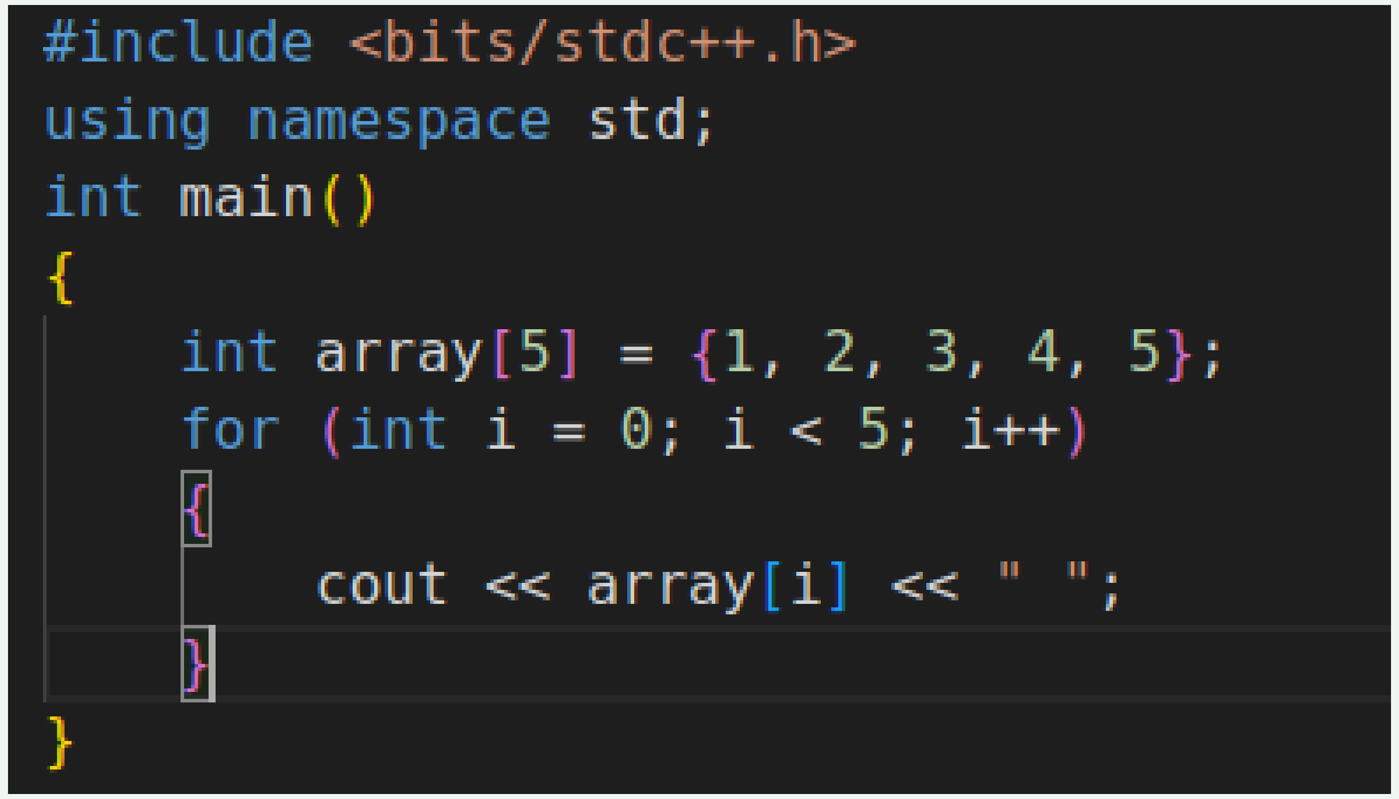
Arrays support random access, which means you can access any element of an array at any instance randomly.



```
#include <bits/stdc++.h>
using namespace std;
int main()
{
    int array[5] = {1, 2, 3, 4, 5};
    cout << array[0] << endl;
    cout << array[1] << endl;
}
```

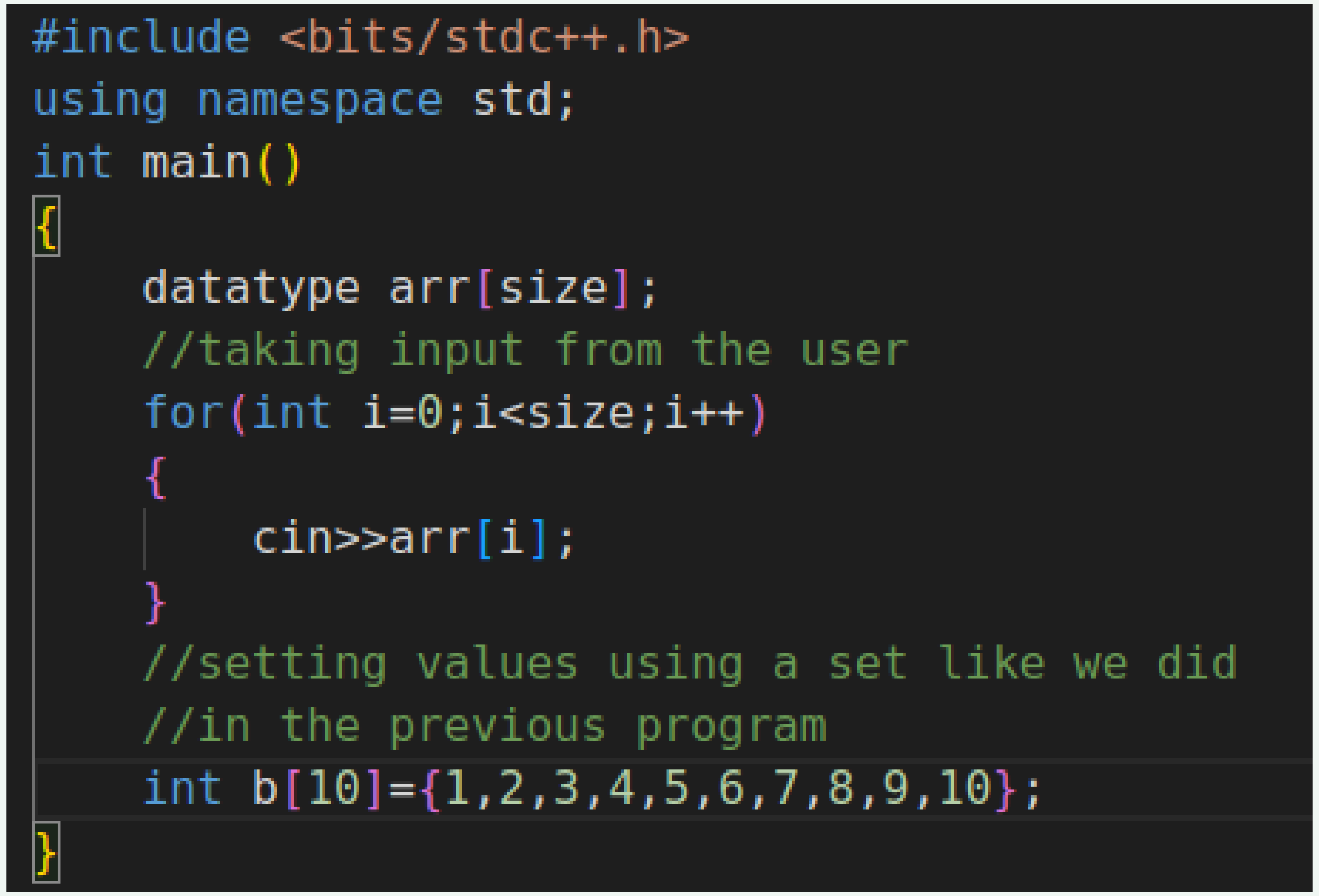
Indexing starts from 0

To traverse or move through the array, one can use a loop quite conveniently



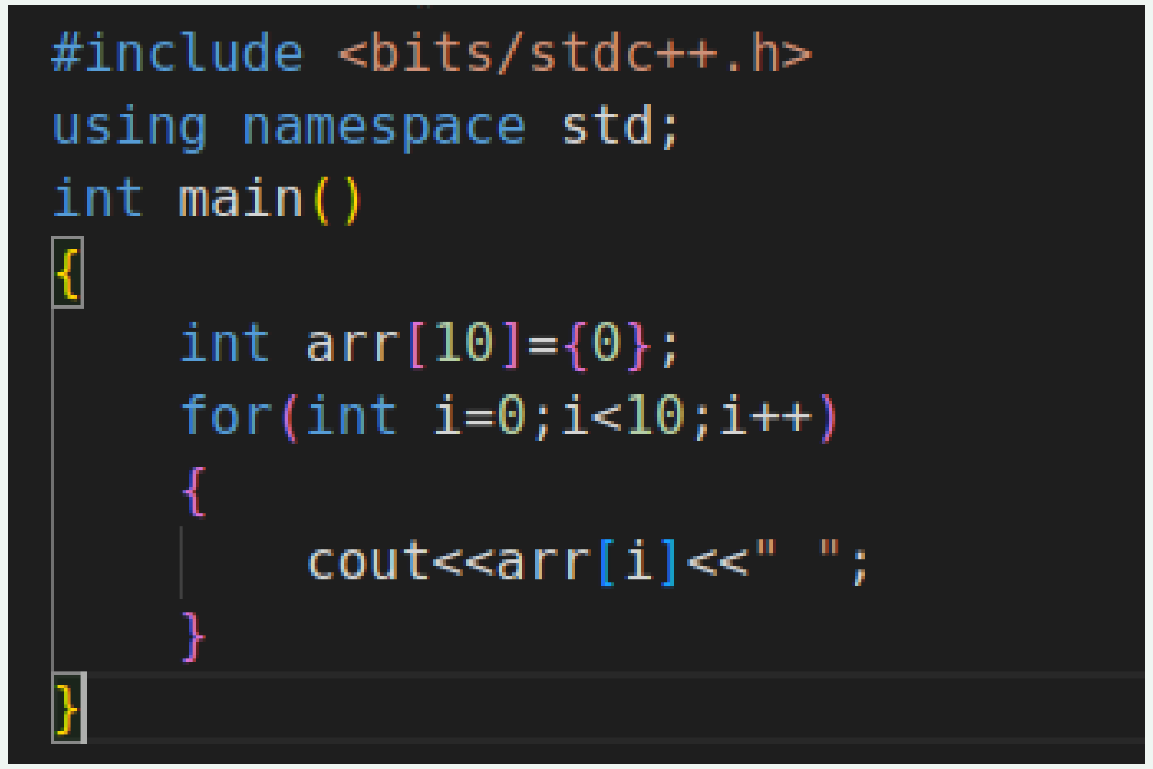
```
#include <bits/stdc++.h>
using namespace std;
int main()
{
    int array[5] = {1, 2, 3, 4, 5};
    for (int i = 0; i < 5; i++)
    {
        cout << array[i] << " ";
    }
}
```

DECLARATION AND USAGE



```
#include <bits/stdc++.h>
using namespace std;
int main()
{
    datatype arr[size];
    //taking input from the user
    for(int i=0;i<size;i++)
    {
        cin>>arr[i];
    }
    //setting values using a set like we did
    //in the previous program
    int b[10]={1,2,3,4,5,6,7,8,9,10};
}
```

Arrays can be declared as shown in the snippet, mention the datatype, name of the array and the required size within the square brackets.
Do note that for an array with size N, the indexing will be from 0...N-1



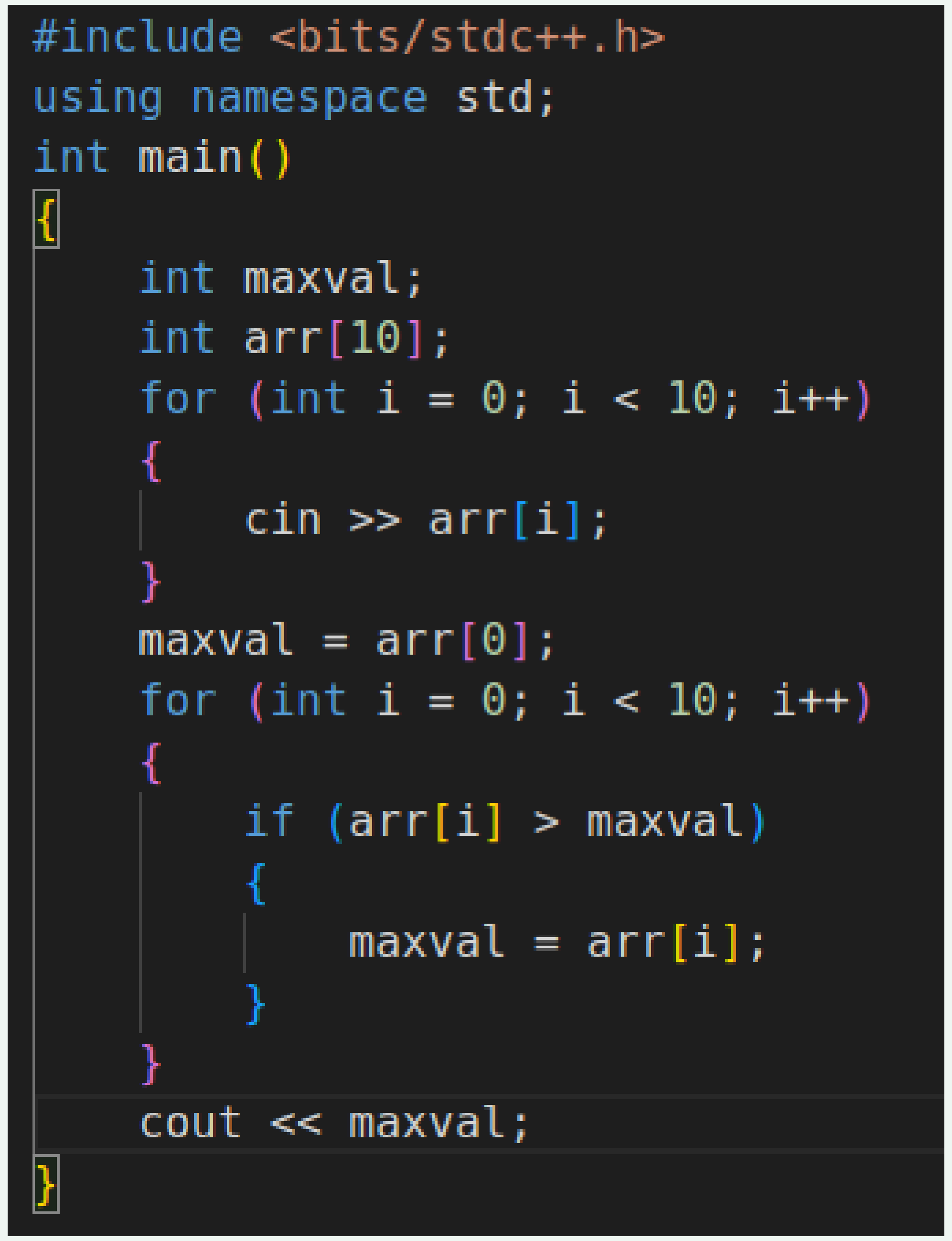
```
#include <bits/stdc++.h>
using namespace std;
int main()
{
    int arr[10]={0};
    for(int i=0;i<10;i++)
    {
        cout<<arr[i]<<" ";
    }
}
```

Values in an array are random initially, they can be set to suit our requirement by either taking in input using a for-loop or using a set of values. (given in curly braces)

The above code sets all values to 0.

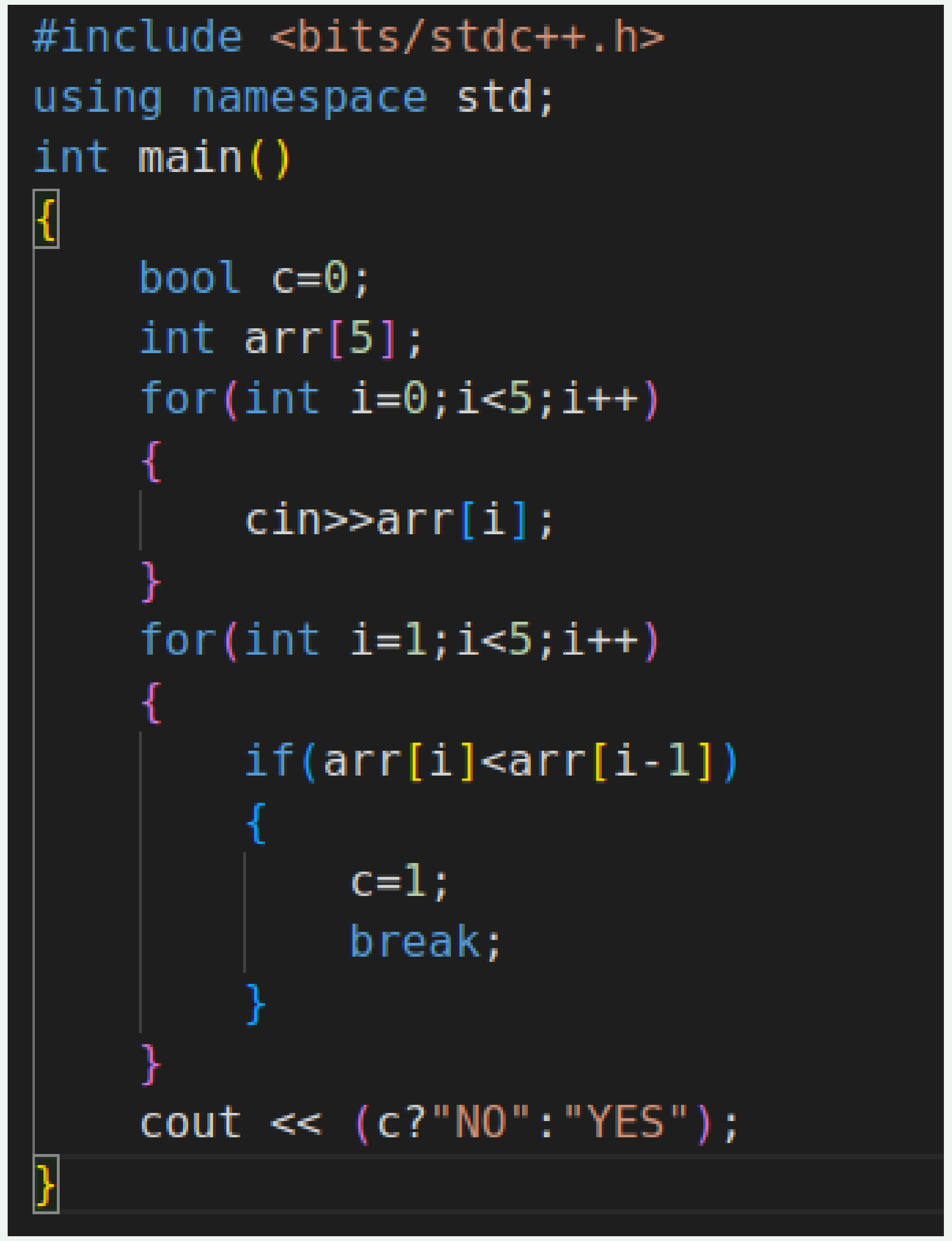
USAGE

Find the maximum value in an array
 of integers and print it out



```cpp
#include <bits/stdc++.h>
using namespace std;
int main()
{
    int maxval;
    int arr[10];
    for (int i = 0; i < 10; i++)
    {
        cin >> arr[i];
    }
    maxval = arr[0];
    for (int i = 0; i < 10; i++)
    {
        if (arr[i] > maxval)
        {
            maxval = arr[i];
        }
    }
    cout << maxval;
}
```

Check if the array is sorted in nature



```cpp
#include <bits/stdc++.h>
using namespace std;
int main()
{
    bool c=0;
    int arr[5];
    for(int i=0;i<5;i++)
    {
        cin>>arr[i];
    }
    for(int i=1;i<5;i++)
    {
        if(arr[i]<arr[i-1])
        {
            c=1;
            break;
        }
    }
    cout << (c?"NO":"YES");
}
```

PROBLEMS

1) Equal Candies

2) Odd Divisor

3) Vasya and Coins

THANK YOU!