Tishk International University
IT Department
Course Code: IT-117

# Programming I

**Lecture 4**

## Control Statement

**Fall 2023**

**Hemin Ibrahim**
**hemin.ibrahim@tiu.edu.iq**

## Outline

- Rational Operations
- Control Structures
- If Statements
- The IF Block Statement
- Nested IF
- Flags
- Logical Operators
- Switch

## Objectives

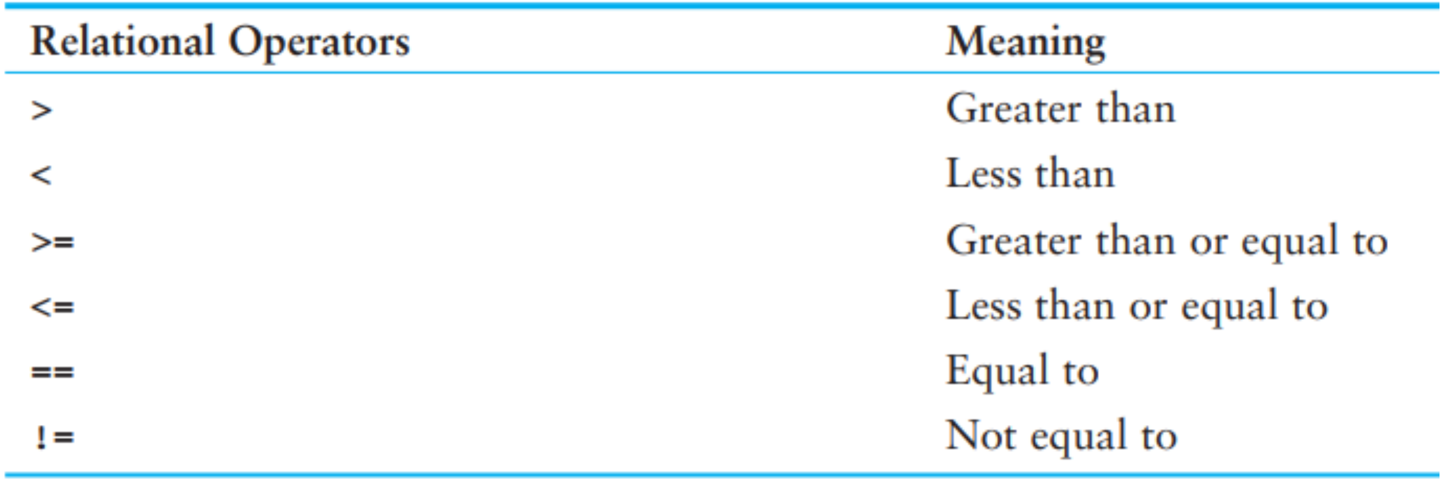
- Learn to use different control structures, like if statements, nested if, flags, logical operators, and switch statements, according to the complexity of the problem.
- Apply rational operations and control structures to solve problems logically.
- Develop versatility in choosing and implementing control flow structures such as if statements, nested if, flags, logical operators, and switch statements.
- Gain proficiency in using conditional statements for effective decision-making, handling various conditions and scenarios in programming

## Programmi

When I wrote this code, only God & I understood what it did.
Now..... only God knows.

## Relational Operators

- Relational operators compare numeric and char values to check if one is greater than, less than, equal to, or not equal to another.
- Computers excel at both calculations and value comparisons.
- Comparisons are essential for tasks like analyzing sales figures, calculating profit and loss, checking numerical ranges, and validating user input.



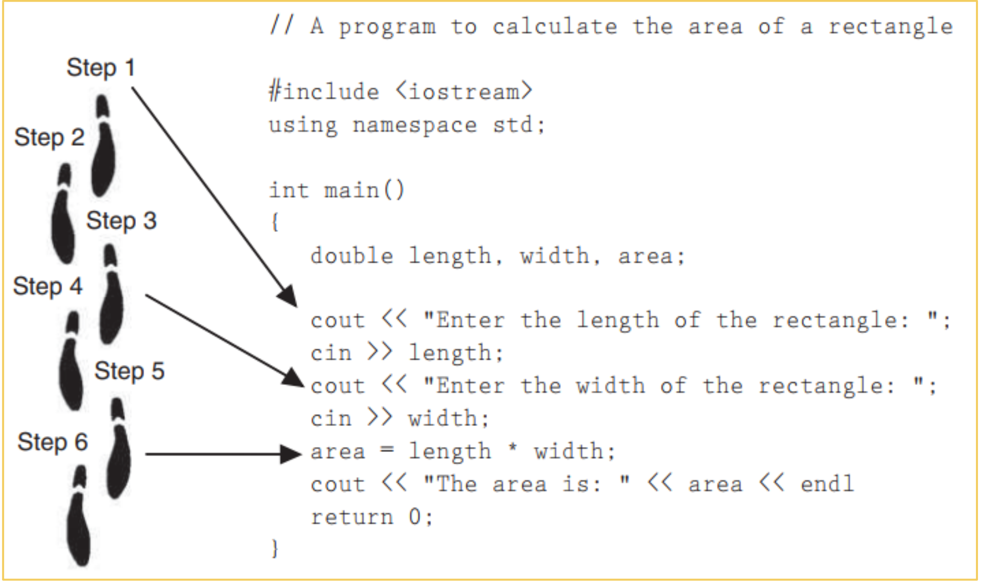
| Relational Operators | Meaning |
|---|---|
| > | Greater than |
| < | Less than |
| >= | Greater than or equal to |
| <= | Less than or equal to |
| == | Equal to |
| != | Not equal to |

## Relational Operators

| Expression | What the Expression Means |
|---|---|
| `x > y` | Is x greater than y? |
| `x < y` | Is x less than y? |
| `x >= y` | Is x greater than or equal to y? |
| `x <= y` | Is x less than or equal to y? |
| `x == y` | Is x equal to y? |
| `x != y` | Is x not equal to y? |

# Control Structures

- We know that program is executed sequentially, unless we give different instructions.
- for the program to not execute sequentially, we need to use a control structure.
- Control Structures provide two basic functions: **selection and repetition (looping)**



```
// A program to calculate the area of a rectangle

#include <iostream>
using namespace std;

int main()
{
   double length, width, area;

   cout << "Enter the length of the rectangle: ";
   cin >> length;
   cout << "Enter the width of the rectangle: ";
   cin >> width;
   area = length * width;
   cout << "The area is: " << area << endl
   return 0;
}
```

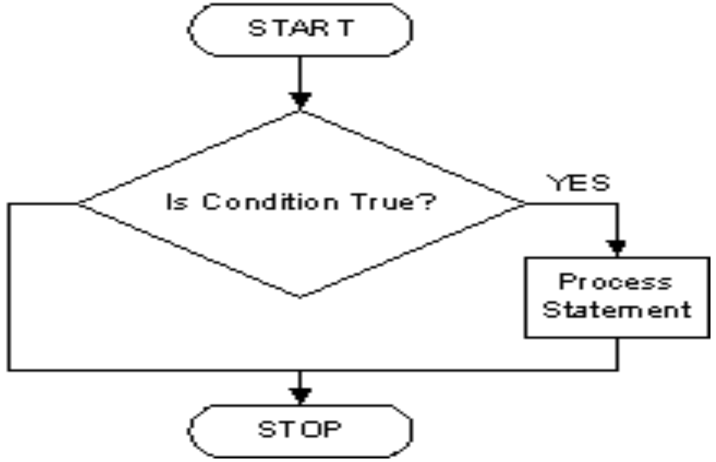
## Control Structures

- A Selection control structure is used to choose among alternative courses of action.
- There must be some *condition* that determines whether or not an action occurs.
- C++ has a number of selection control structures:
    - if
    - if…. else
    - switch

## Control Stru

## If statement

- The if statement can cause other statements to execute only under certain conditions.
- The if selection statement is a **single-selection statement**
- it selects or ignores a single statement (or block of statements) depending on the condition
- Modifies the order of the statement execution.

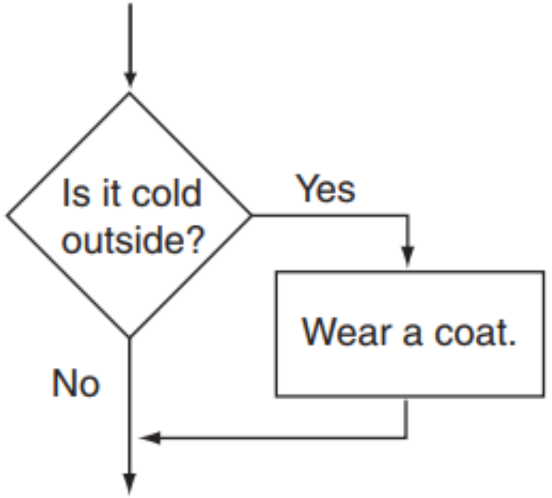


## If statement - Example

In the flowchart, the action “Wear a coat” is performed only when it is cold outside. If it is not cold outside, the action is skipped. The action is conditionally executed because it is performed only when a certain condition (cold outside) exists.

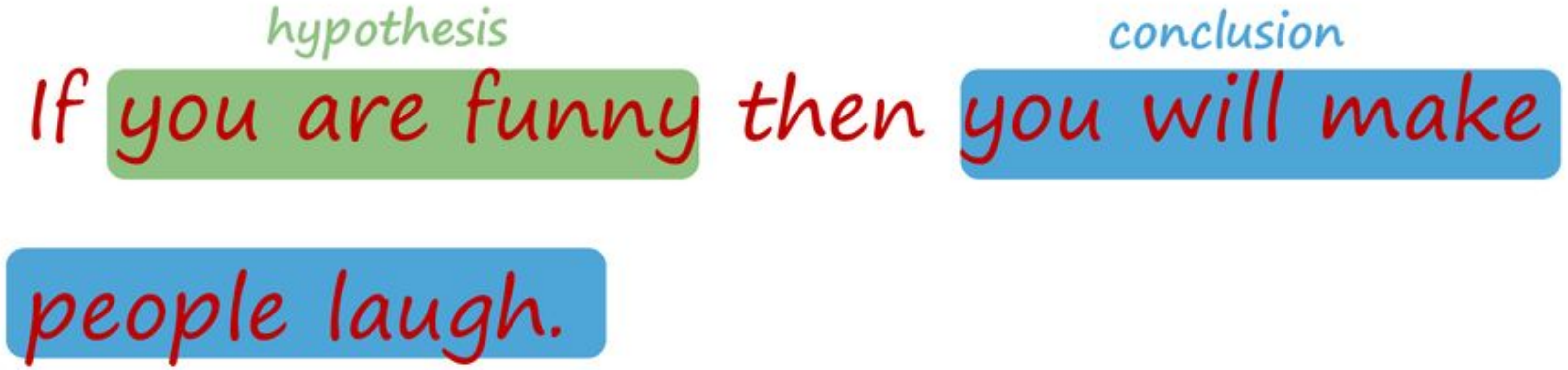
We perform mental tests like these every day. Here are some other examples:

- If the car is low on gas, stop at a service station and get gas.
- If it’s raining outside, go inside.
- If you’re hungry, get something to eat.



## If statement - Example

1. You will make people laugh if you are funny.

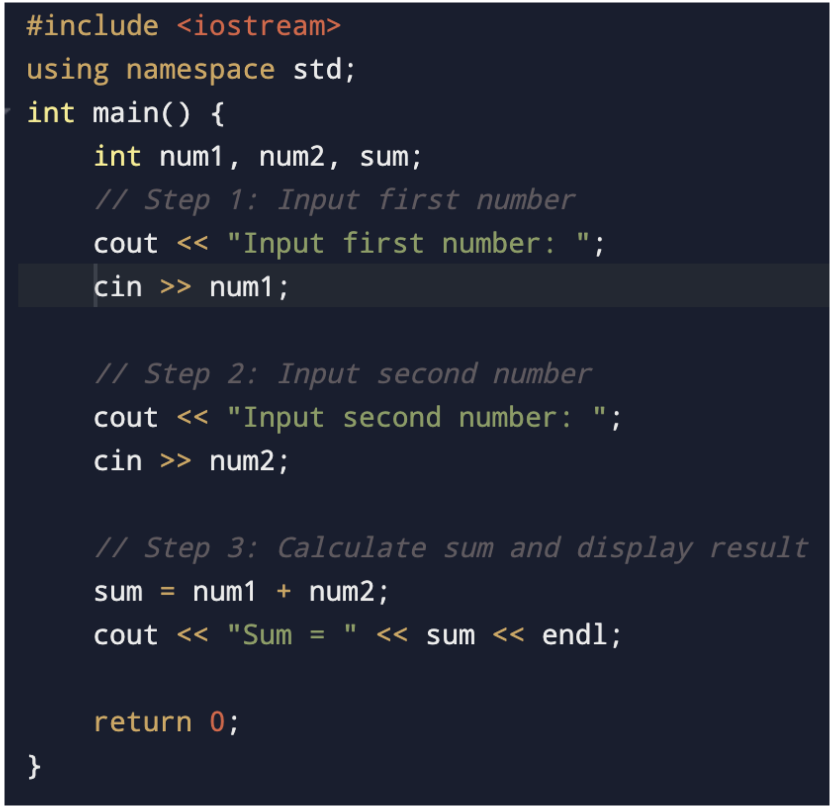

hypothesis

conclusion

If you are funny then you will make people laugh.

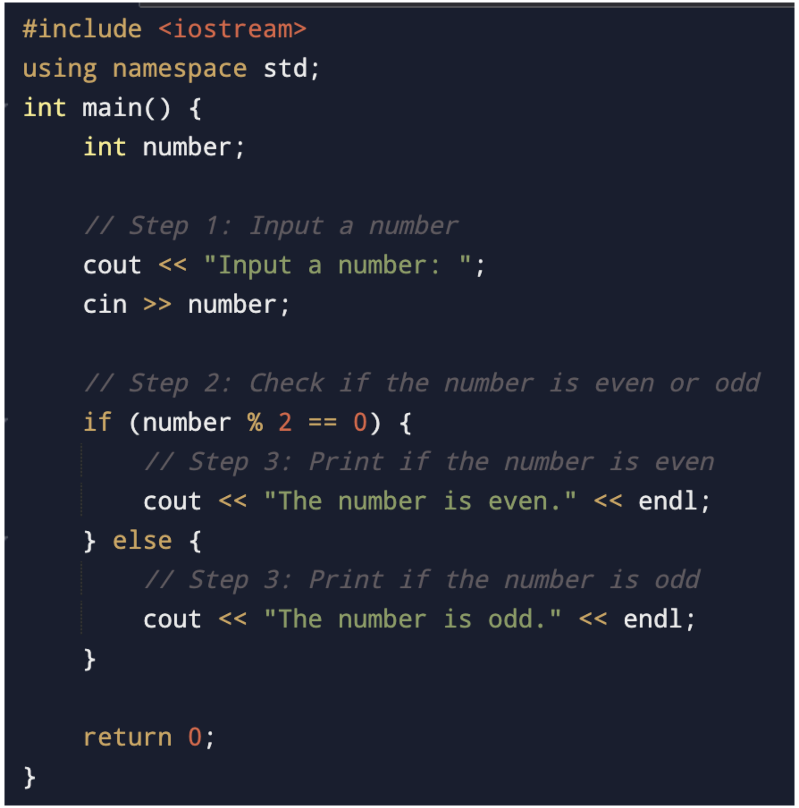
## If statement

Executed sequentially



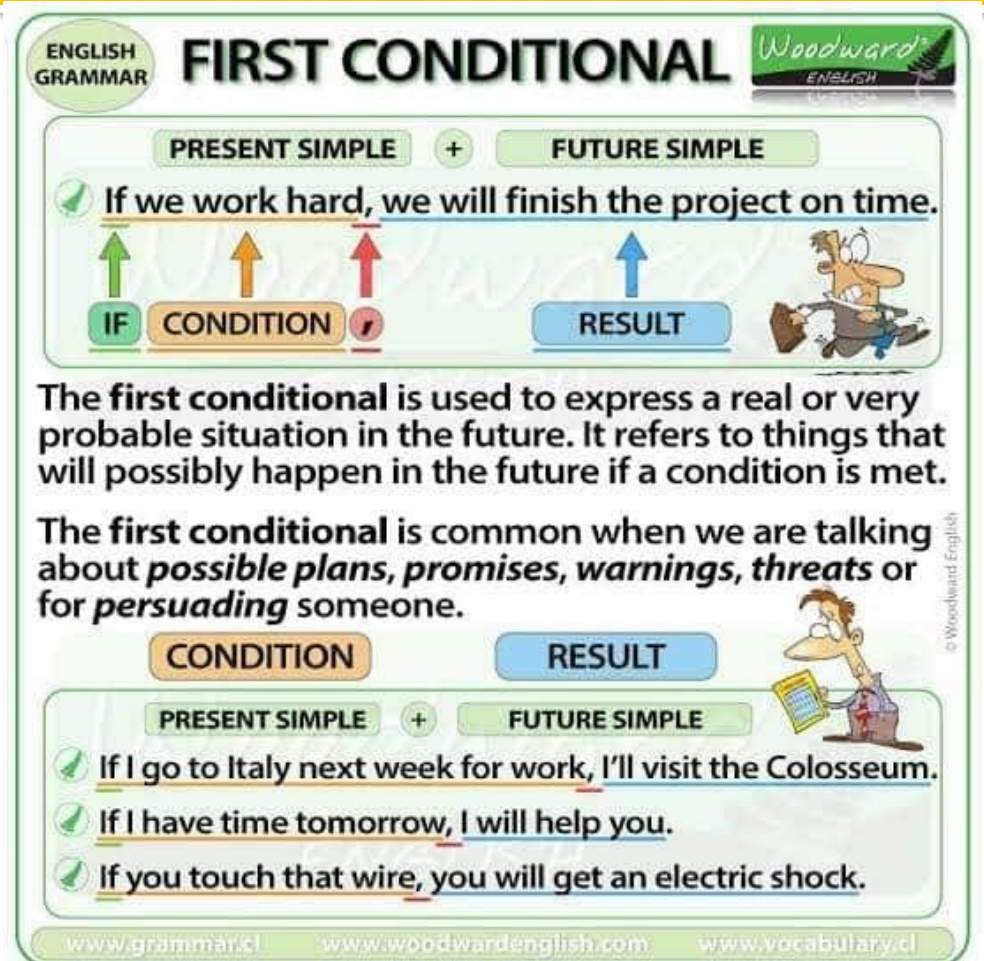
```cpp
#include <iostream>
using namespace std;
int main() {
    int num1, num2, sum;
    // Step 1: Input first number
    cout << "Input first number: ";
    cin >> num1;

    // Step 2: Input second number
    cout << "Input second number: ";
    cin >> num2;

    // Step 3: Calculate sum and display result
    sum = num1 + num2;
    cout << "Sum = " << sum << endl;

    return 0;
}
```

Using control structure



```cpp
#include <iostream>
using namespace std;
int main() {
    int number;

    // Step 1: Input a number
    cout << "Input a number: ";
    cin >> number;

    // Step 2: Check if the number is even or odd
    if (number % 2 == 0) {
        // Step 3: Print if the number is even
        cout << "The number is even." << endl;
    } else {
        // Step 3: Print if the number is odd
        cout << "The number is odd." << endl;
    }

    return 0;
}
```

# If statement - Example



ENGLISH GRAMMAR

FIRST CONDITIONAL

Woodward ENGLISH

PRESENT SIMPLE + FUTURE SIMPLE

✓ If we work hard, we will finish the project on time.

IF CONDITION , RESULT

The **first conditional** is used to express a real or very probable situation in the future. It refers to things that will possibly happen in the future if a condition is met.

The **first conditional** is common when we are talking about ***possible plans***, ***promises***, ***warnings***, ***threats*** or for ***persuading*** someone.

CONDITION RESULT

PRESENT SIMPLE + FUTURE SIMPLE

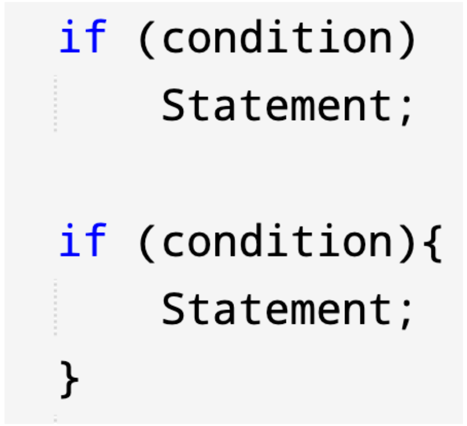
✓ If I go to Italy next week for work, I'll visit the Colosseum.

✓ If I have time tomorrow, I will help you.

✓ If you touch that wire, you will get an electric shock.

# If statement in C++

- Evaluate an expression (condition) and directs program execution depending on the result of that evaluation.
- If the expression evaluate as TRUE, statement is executed, if FALSE, statement is not executed, execution then passed to the code follows the if statement, that is the next_statement.
- So, the execution of the statement depends on the result of expression.



```
if (condition)
    Statement;

if (condition){
    Statement;
}
```

# If statement in C++

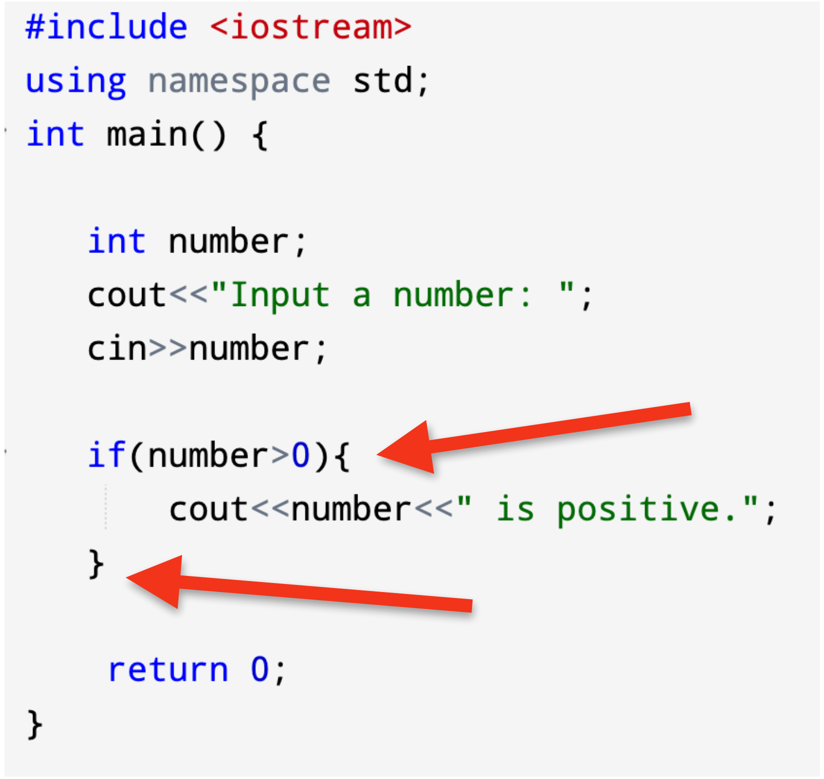
```
if (condition)
    Statement;

if (condition){
    Statement;
}
```

No semicolon goes here

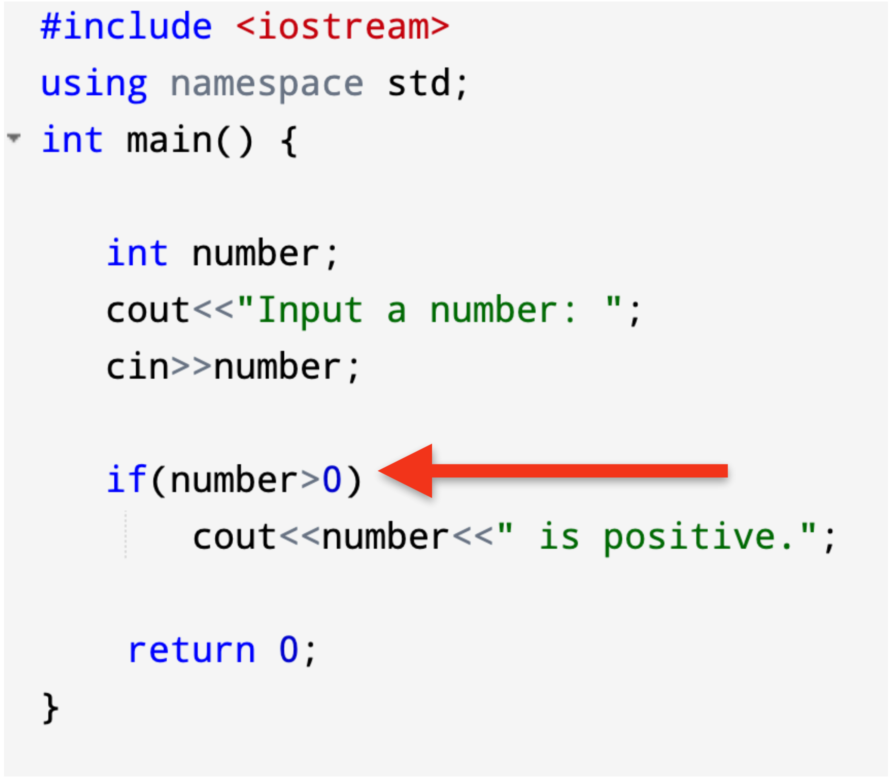
semicolon goes here

## Example #1

Write a C++ program that asks user to input a number, then check the number is positive?

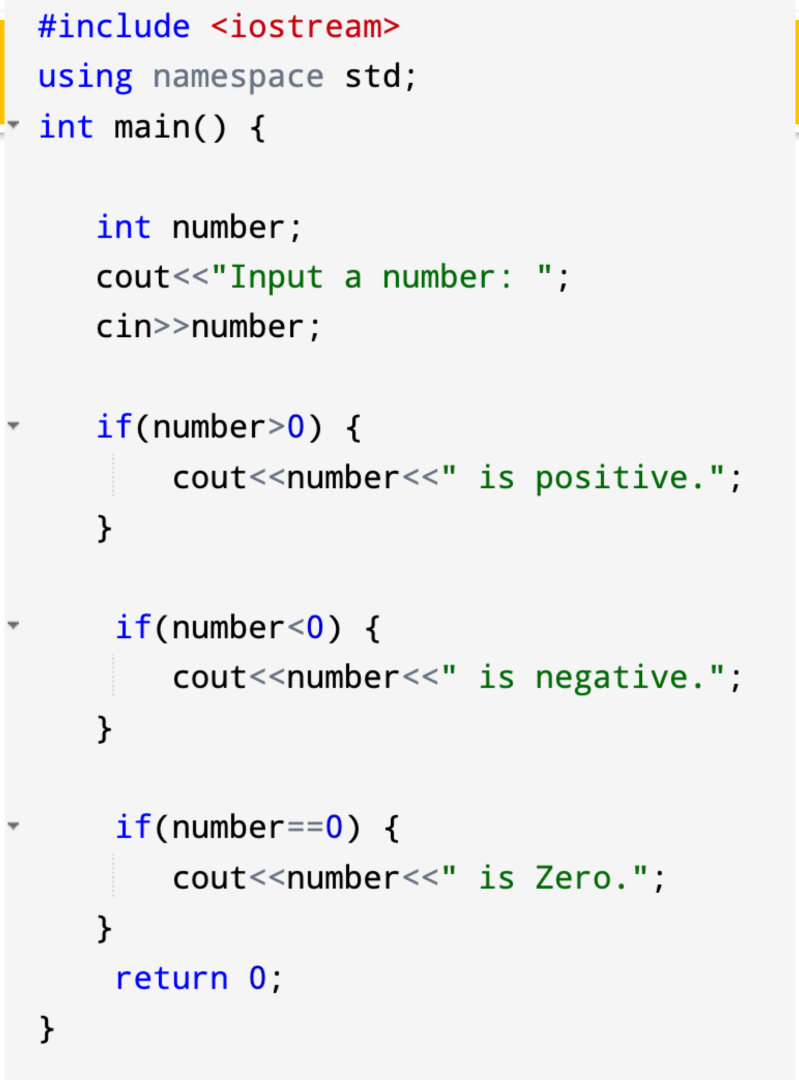


```
#include <iostream>
using namespace std;
int main() {

    int number;
    cout<<"Input a number: ";
    cin>>number;

    if(number>0){
        cout<<number<<" is positive.";
    }

    return 0;
}
```



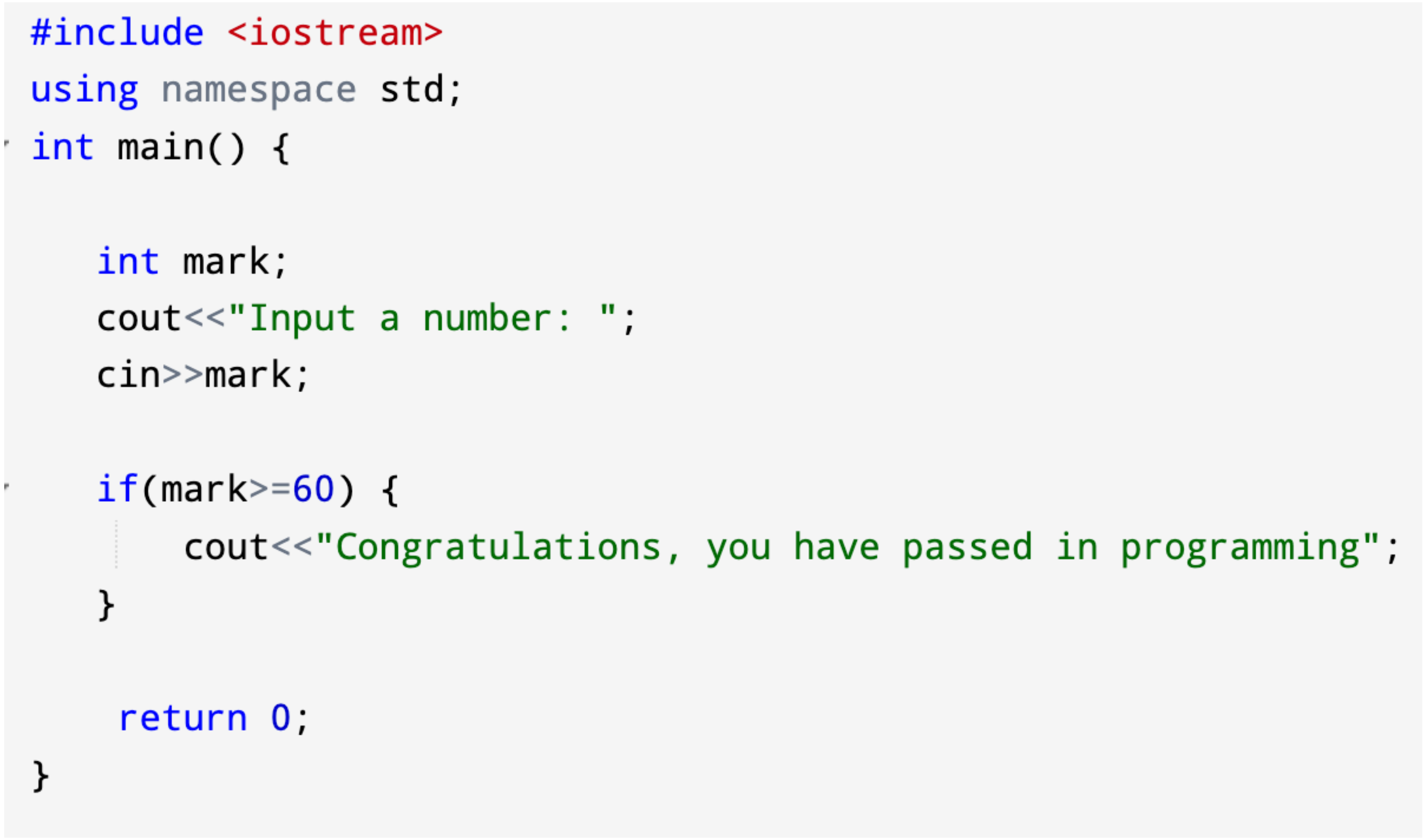
```
#include <iostream>
using namespace std;
int main() {

    int number;
    cout<<"Input a number: ";
    cin>>number;

    if(number>0)
        cout<<number<<" is positive.";

     return 0;
}
```

## Example #2



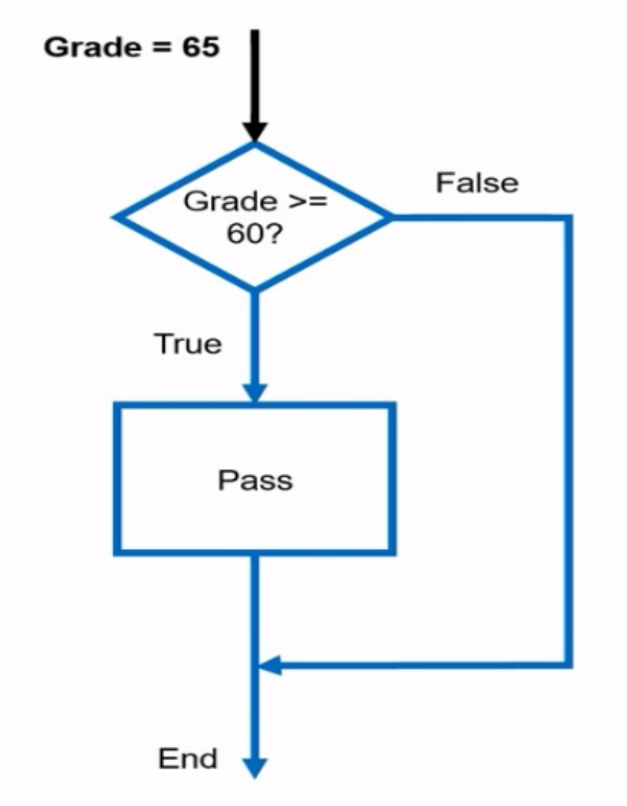
```
#include <iostream>
using namespace std;
int main() {

    int number;
    cout<<"Input a number: ";
    cin>>number;

    if(number>0) {
        cout<<number<<" is positive.";
    }

     if(number<0) {
        cout<<number<<" is negative.";
    }

     if(number==0) {
        cout<<number<<" is Zero.";
    }
     return 0;
}
```

## Example #3



```
#include <iostream>
using namespace std;
int main() {

    int mark;
    cout<<"Input a number: ";
    cin>>mark;

    if(mark>=60) {
        cout<<"Congratulations, you have passed in programming";
    }

     return 0;
}
```

# Example #3 - Flowchart



## The if Block Statement

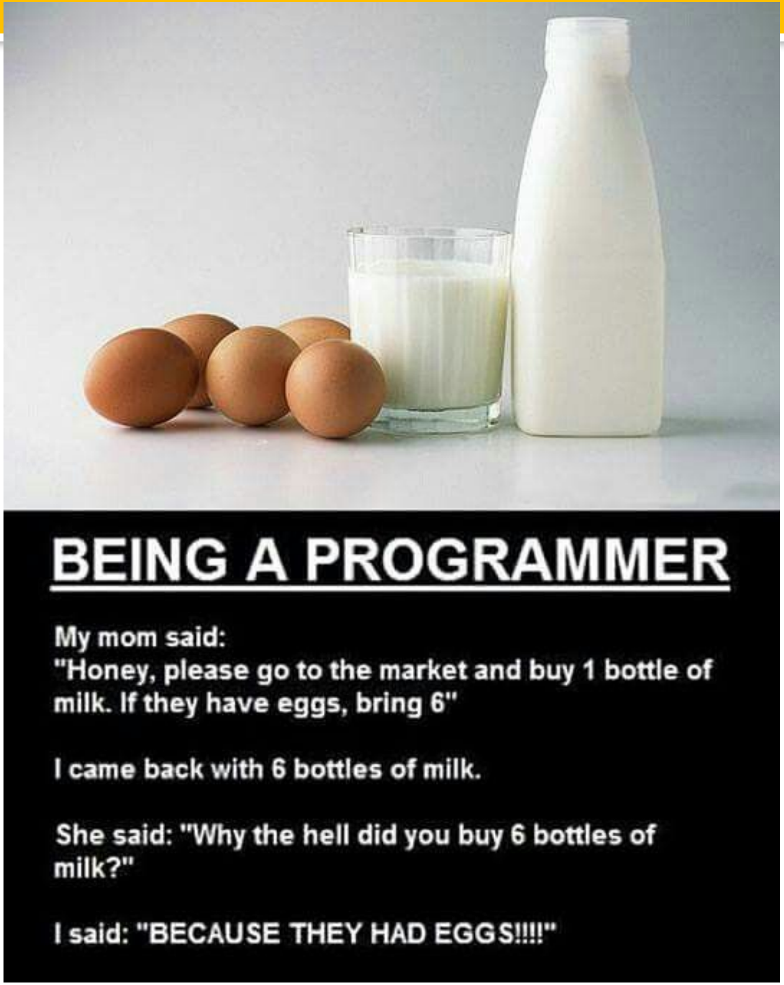
If you want an 'if' statement to execute a group of statements, use a compound statement enclosed in '{' and '}'. It allows you to control the execution of multiple statements or control structures

```
if (expression)
{
    statement;
    statement;
    // Place as many statements here as necessary.
}
```

## The if Block Statement - Example

```
#include <iostream>
using namespace std;
int main() {

   string username = "user123";

   if (username == "user123") {
        cout << "Welcome, " << username << "!" << endl;
        cout << "You have successfully logged in." << endl;
    }

    return 0;
}
```

## ??



BEING A PROGRAMMER

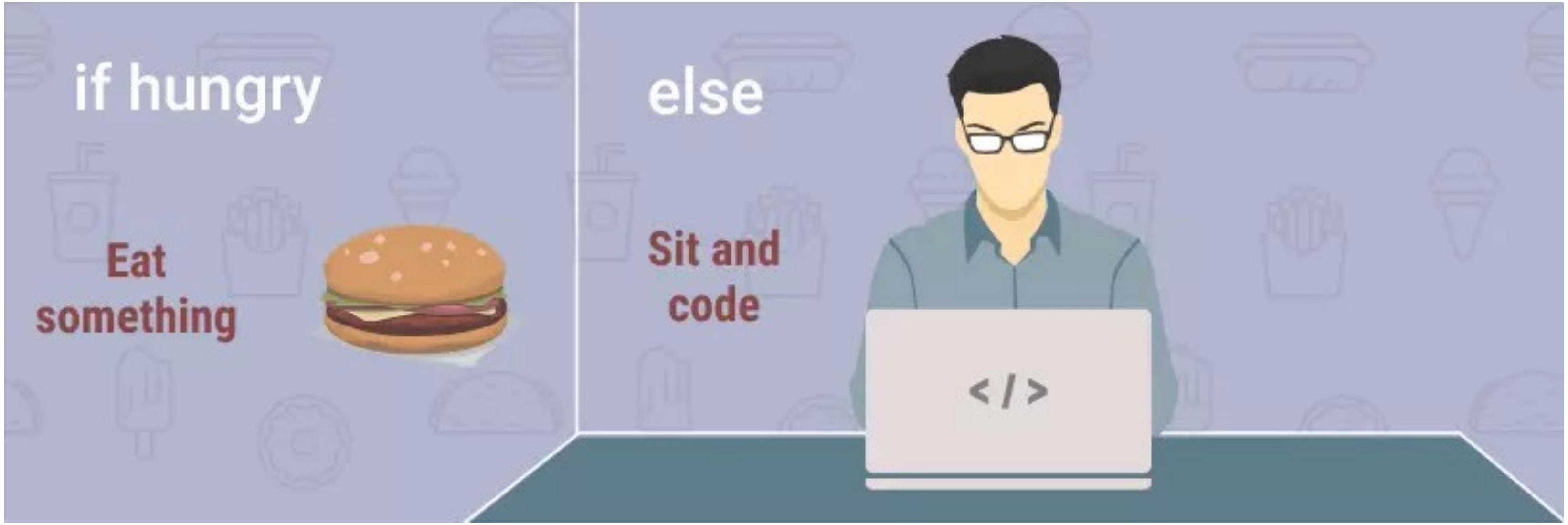
My mom said:
"Honey, please go to the market and buy 1 bottle of milk. If they have eggs, bring 6"

I came back with 6 bottles of milk.

She said: "Why the hell did you buy 6 bottles of milk?"

I said: "BECAUSE THEY HAD EGGS!!!!"

## If … else



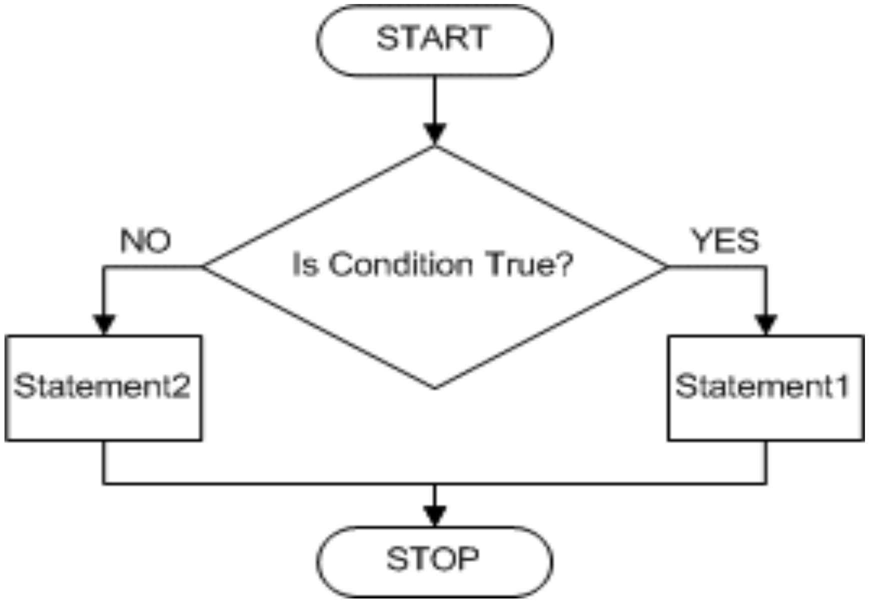
## The `if/else` Statement

The if/else statement will execute one group of statements if the expression is true, or another group of statements if the expression is false.

```
if (expression){
    statement or block;
} else {
    statement or block;
}
```

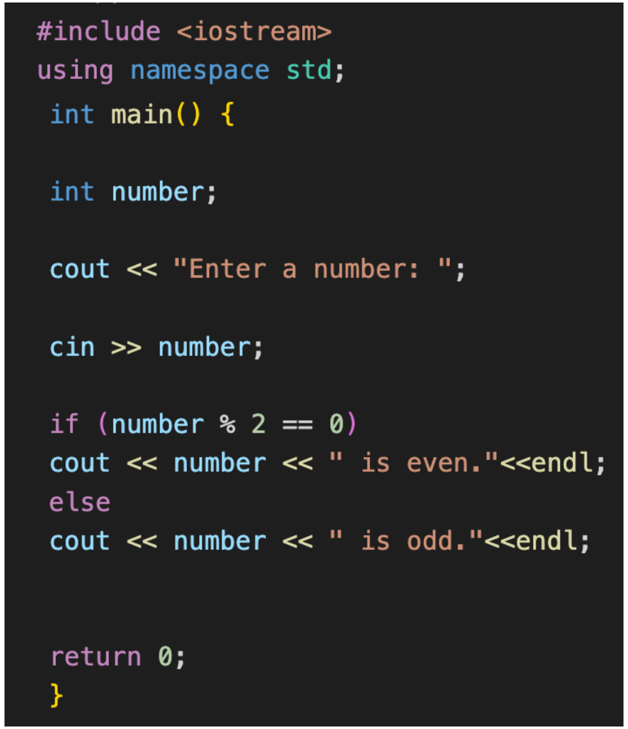
With an if statement, if the expression is true, specific statements are executed; otherwise, a different set of statements is executed

## The `if/else` Statement

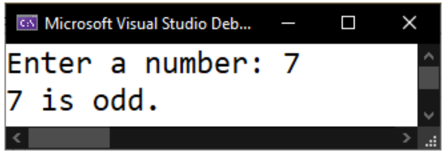


## The `if/else` Statement - Example

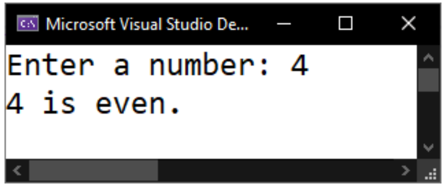
Write a C++ program to check a given integer is even or odd.



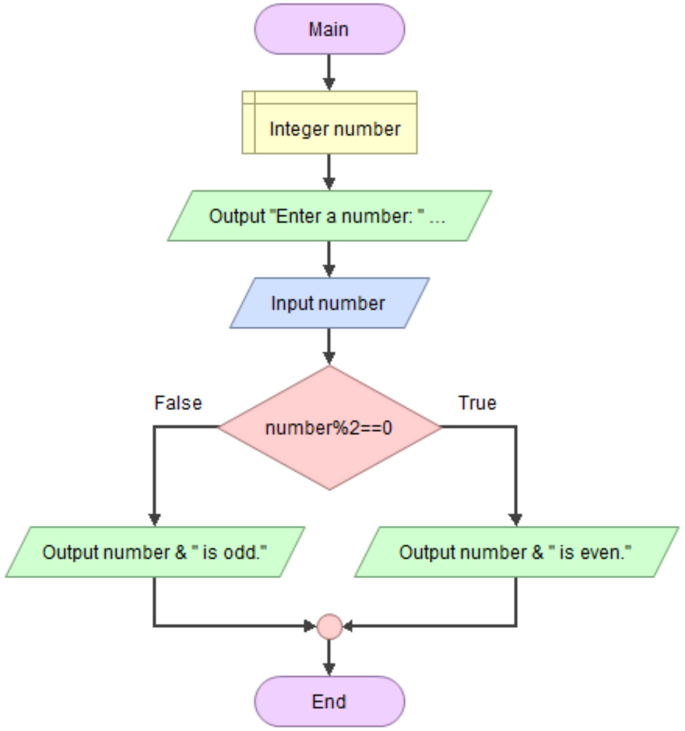
```cpp
#include <iostream>
using namespace std;
 int main() {

 int number;

 cout << "Enter a number: ";

 cin >> number;

 if (number % 2 == 0)
 cout << number << " is even."<<endl;
 else
 cout << number << " is odd."<<endl;


 return 0;
 }
```



```
Enter a number: 7
7 is odd.
```



```
Enter a number: 4
4 is even.
```

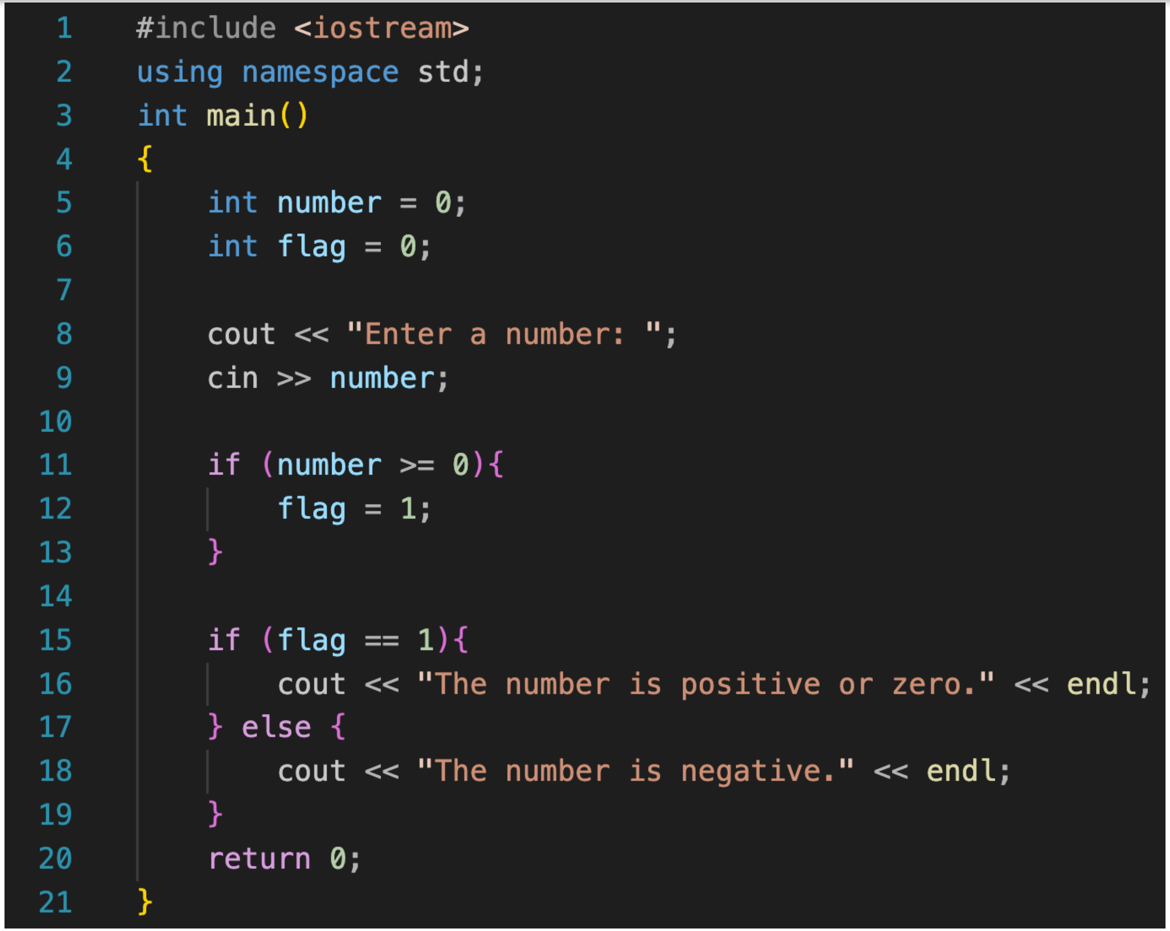


Main
Integer number
Output "Enter a number: " ...
Input number
number%2==0
False
True
Output number & " is odd."
Output number & " is even."
End

## Flags and Integer Flags

- A flag is a variable used to signal the existence of a condition in a program.
- Flags are usually Boolean or integer variables.
- When the flag is set to false, it signifies that the condition does not exist.
- Setting the flag to true indicates the presence of the specified condition.

# Flags and Integer Flags - Example 1



```cpp
#include <iostream>
using namespace std;
int main()
{
    int number = 0;
    int flag = 0;

    cout << "Enter a number: ";
    cin >> number;

    if (number >= 0){
        flag = 1;
    }

    if (flag == 1){
        cout << "The number is positive or zero." << endl;
    } else {
        cout << "The number is negative." << endl;
    }
    return 0;
}
```

# Flags and Integer Flags - Example 2

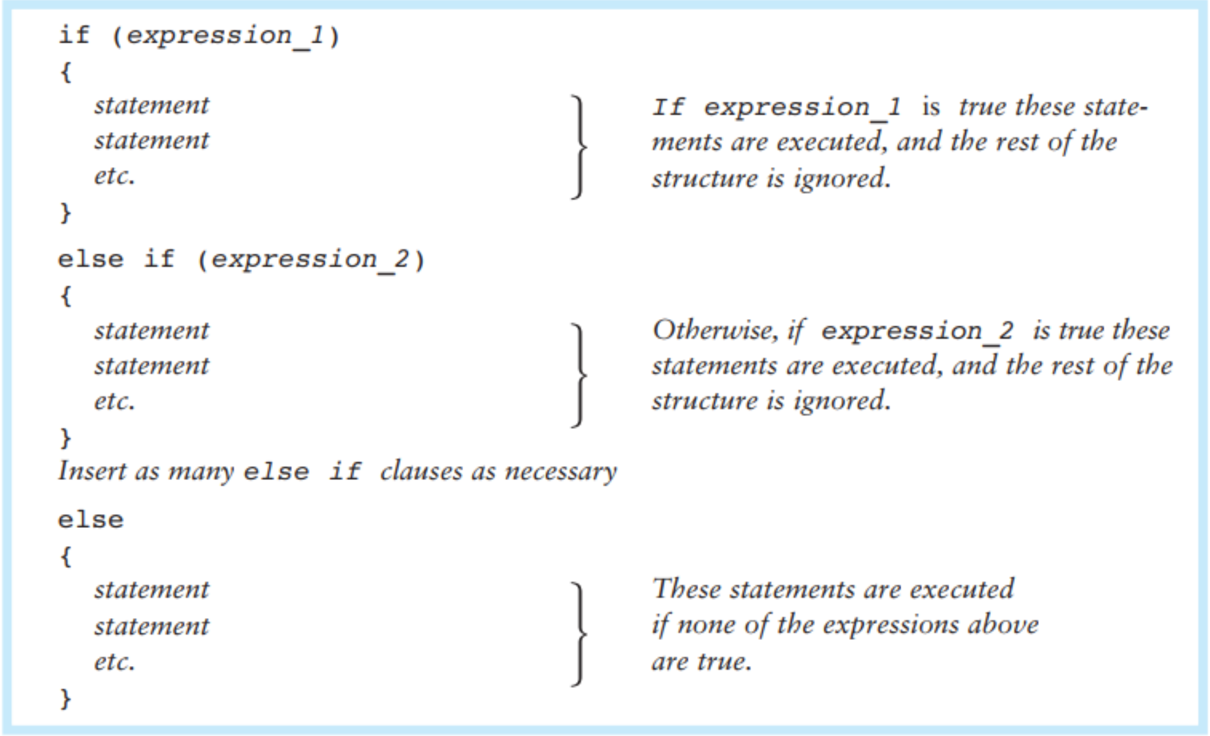
```cpp
#include <iostream>
using namespace std;
int main() {
    int number;
    bool even=false;
    cout<<"Enter an integer: ";
    cin>>number;
    if(number%2==0){
        even=true;
    }

    if(even){
        cout<<number<<" is Even."<<endl;
    } else {
        cout<<number<<" is Odd."<<endl;
    }

    return 0;
}
```

```
Enter an integer: 12
12 is Even.
```

```
Enter an integer: 11
11 is Odd.
```

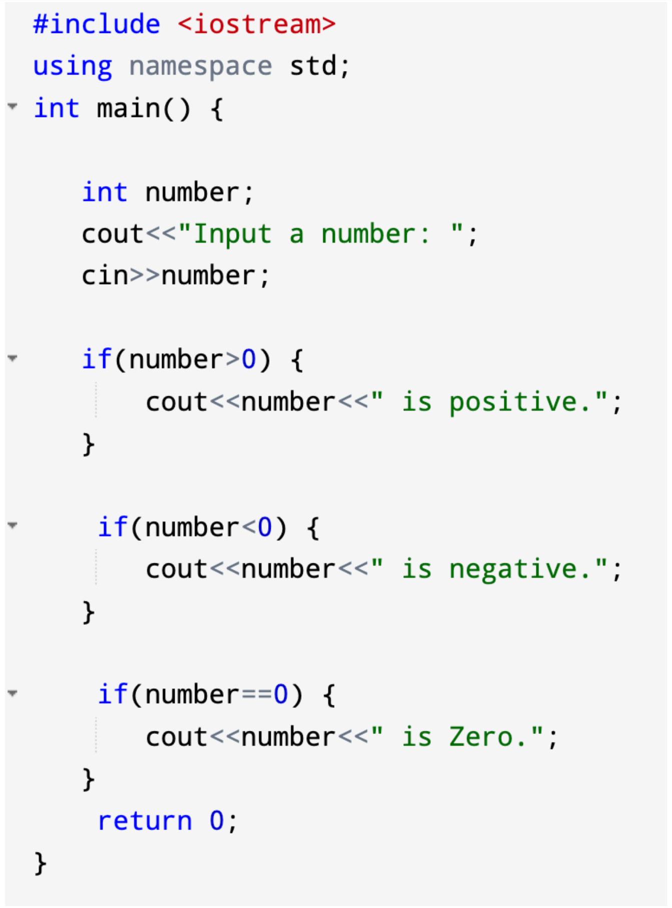
## The if/else if Statements

The if/else if statement simplifies testing multiple conditions, often done more efficiently than using nested if/else statements

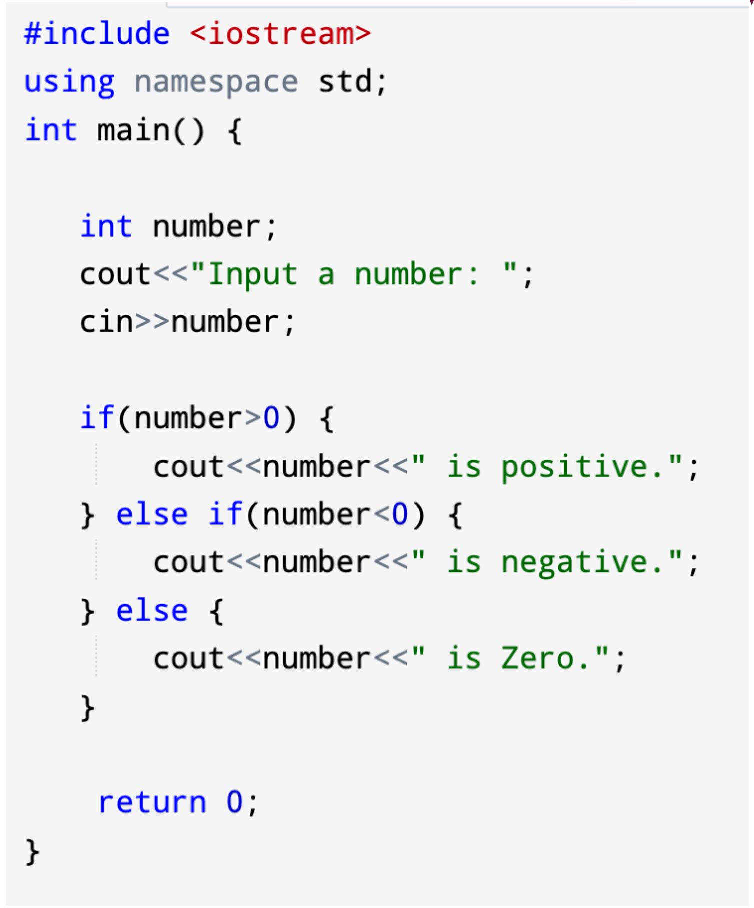


```
if (expression_1)
{
    statement                      If expression_1 is true these state-
    statement                   }  ments are executed, and the rest of the
    etc.                           structure is ignored.
}
else if (expression_2)
{
    statement                      Otherwise, if expression_2 is true these
    statement                   }  statements are executed, and the rest of the
    etc.                           structure is ignored.
}
Insert as many else if clauses as necessary
else
{
    statement                      These statements are executed
    statement                   }  if none of the expressions above
    etc.                           are true.
}
```

# The if/else if Statements - Example

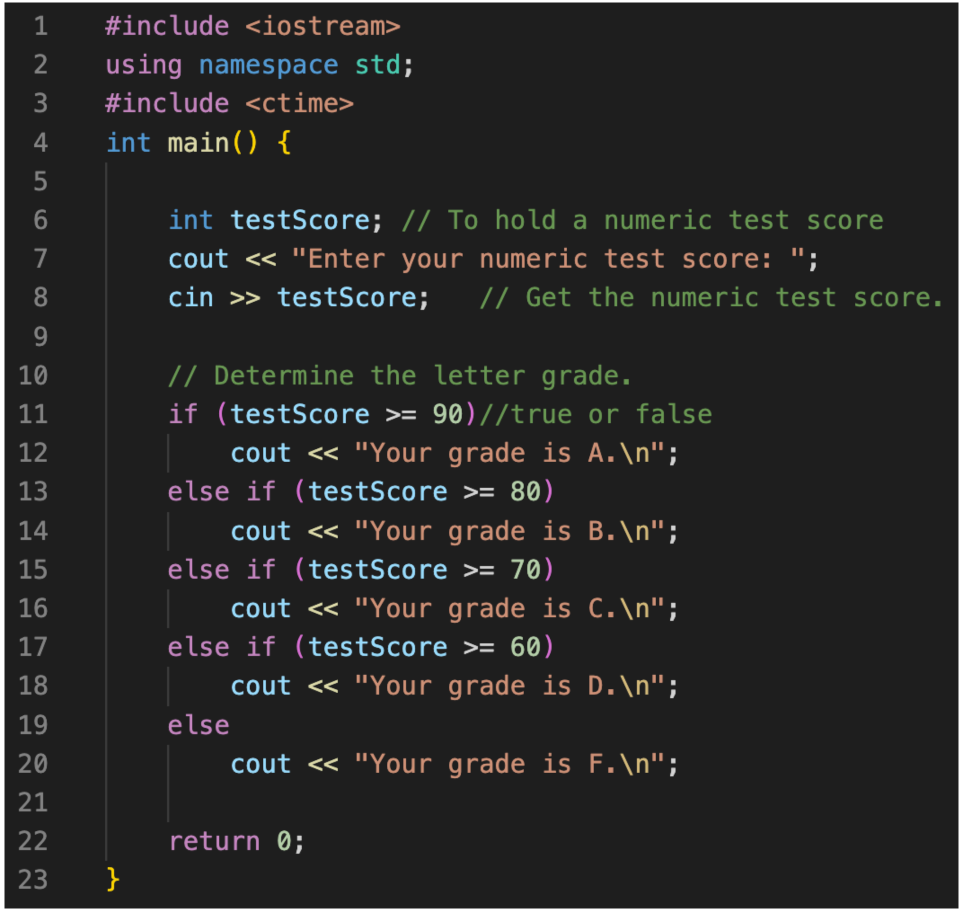


```cpp
#include <iostream>
using namespace std;
int main() {

    int number;
    cout<<"Input a number: ";
    cin>>number;

    if(number>0) {
        cout<<number<<" is positive.";
    }

     if(number<0) {
        cout<<number<<" is negative.";
    }

     if(number==0) {
        cout<<number<<" is Zero.";
    }
     return 0;
}
```



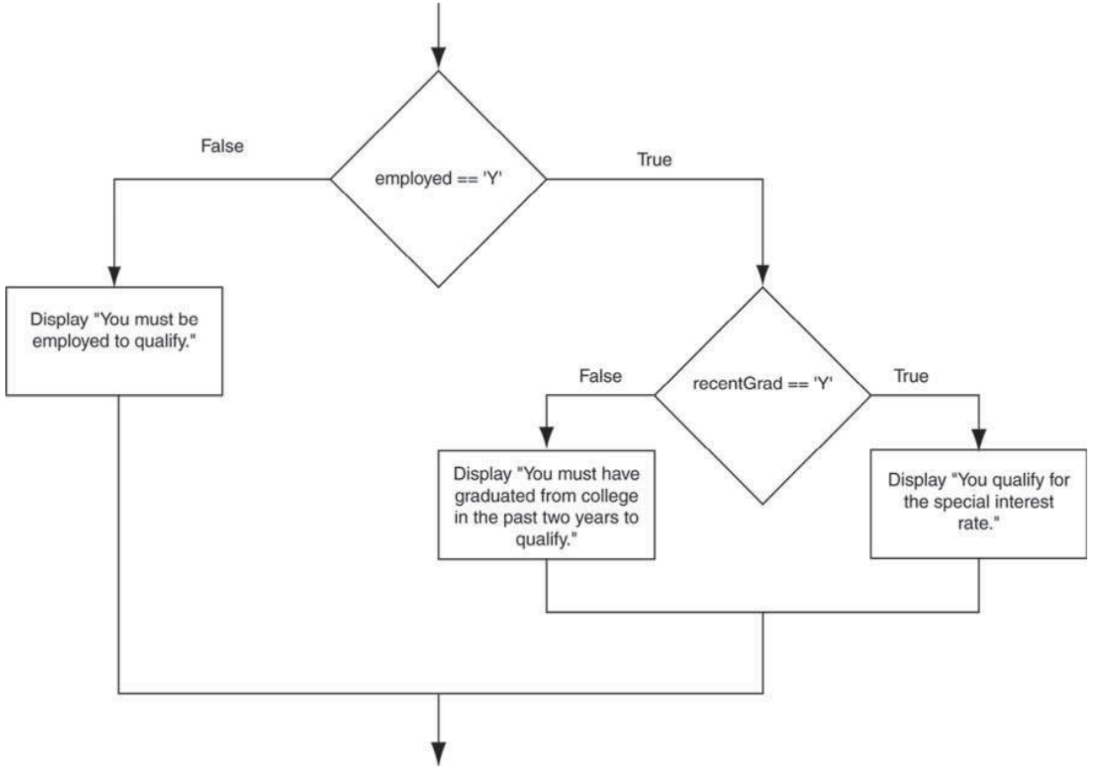
```cpp
#include <iostream>
using namespace std;
int main() {

    int number;
    cout<<"Input a number: ";
    cin>>number;

    if(number>0) {
        cout<<number<<" is positive.";
    } else if(number<0) {
        cout<<number<<" is negative.";
    } else {
        cout<<number<<" is Zero.";
    }

     return 0;
}
```

## The if/else if Statements - Example


```cpp
#include <iostream>
using namespace std;
#include <ctime>
int main() {

    int testScore; // To hold a numeric test score
    cout << "Enter your numeric test score: ";
    cin >> testScore;   // Get the numeric test score.

    // Determine the letter grade.
    if (testScore >= 90)//true or false
        cout << "Your grade is A.\n";
    else if (testScore >= 80)
        cout << "Your grade is B.\n";
    else if (testScore >= 70)
        cout << "Your grade is C.\n";
    else if (testScore >= 60)
        cout << "Your grade is D.\n";
    else
        cout << "Your grade is F.\n";

    return 0;
}
```

## Nested `if` Statements

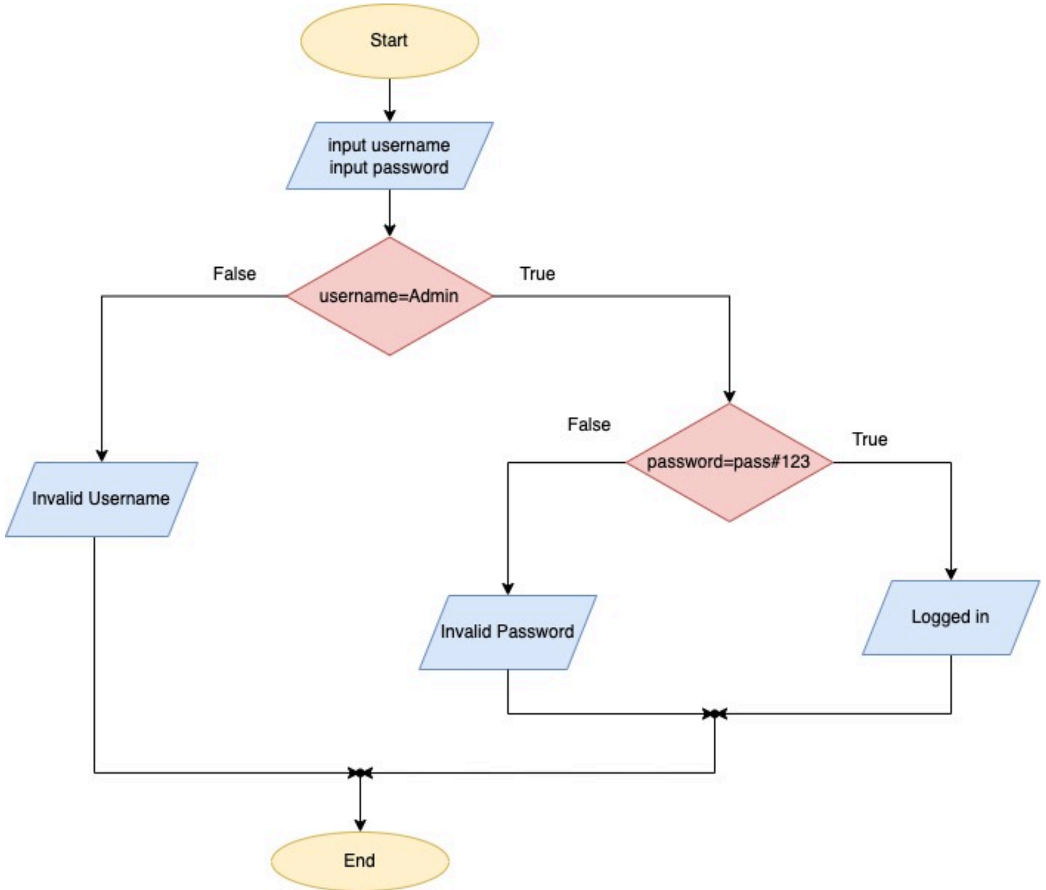
To test more than one condition, an if statement can be nested inside another if statement.



employed == 'Y'
False
True
Display "You must be employed to qualify."
recentGrad == 'Y'
False
True
Display "You must have graduated from college in the past two years to qualify."
Display "You qualify for the special interest rate."

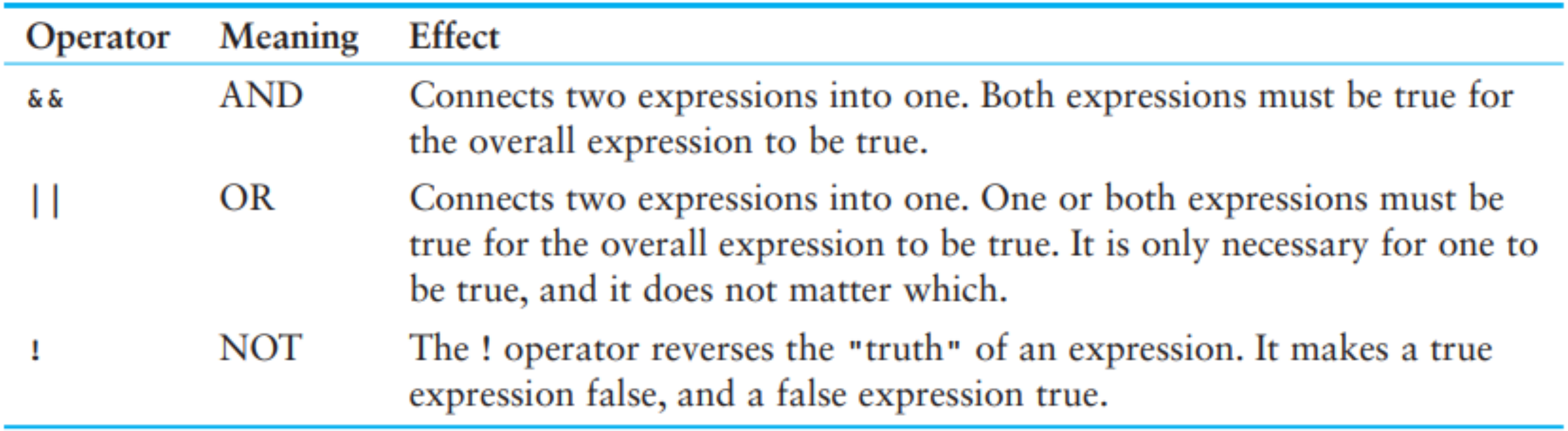
# Nested `if` Statements - Example

```cpp
#include <iostream>
using namespace std;
int main() {
    string username,password;
    cout<<"Enter your username: ";
    cin>>username;
    cout<<"Enter your password: ";
    cin>>password;

    if(username=="Admin"){
        if(password=="pass#123"){
            cout<<"Logged in successfully.";
        } else {
            cout<<"Invalid password.";
        }
    } else {
        cout<<"Invalid username.";
    }

    return 0;
}
```



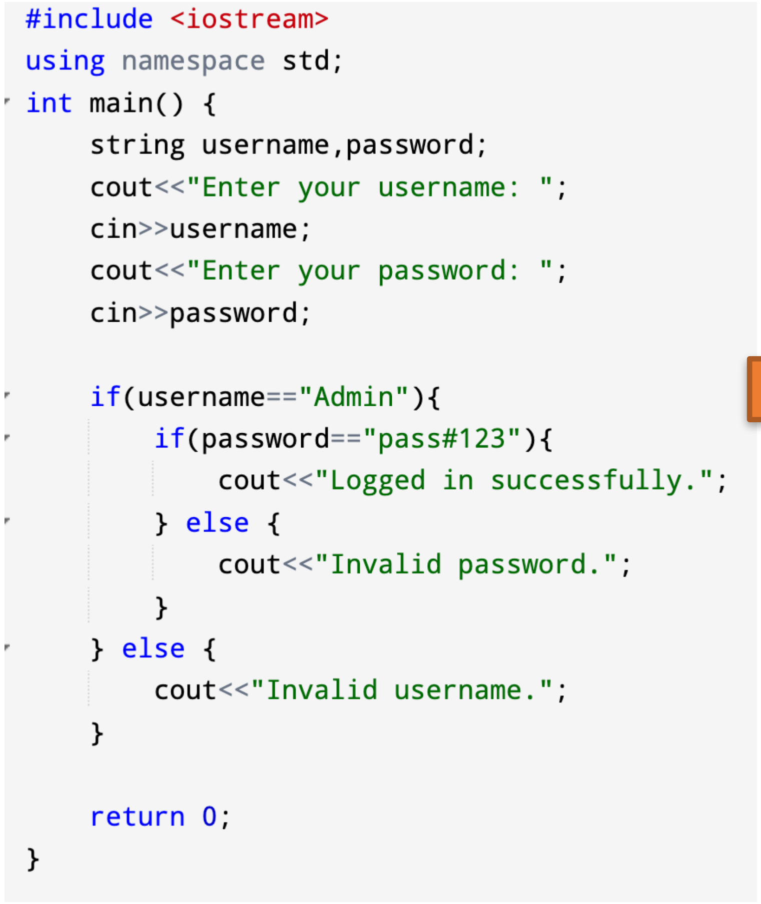
# Logical Operators

Logical operators connect two or more relational expressions into one or reverse the logic of an expression.

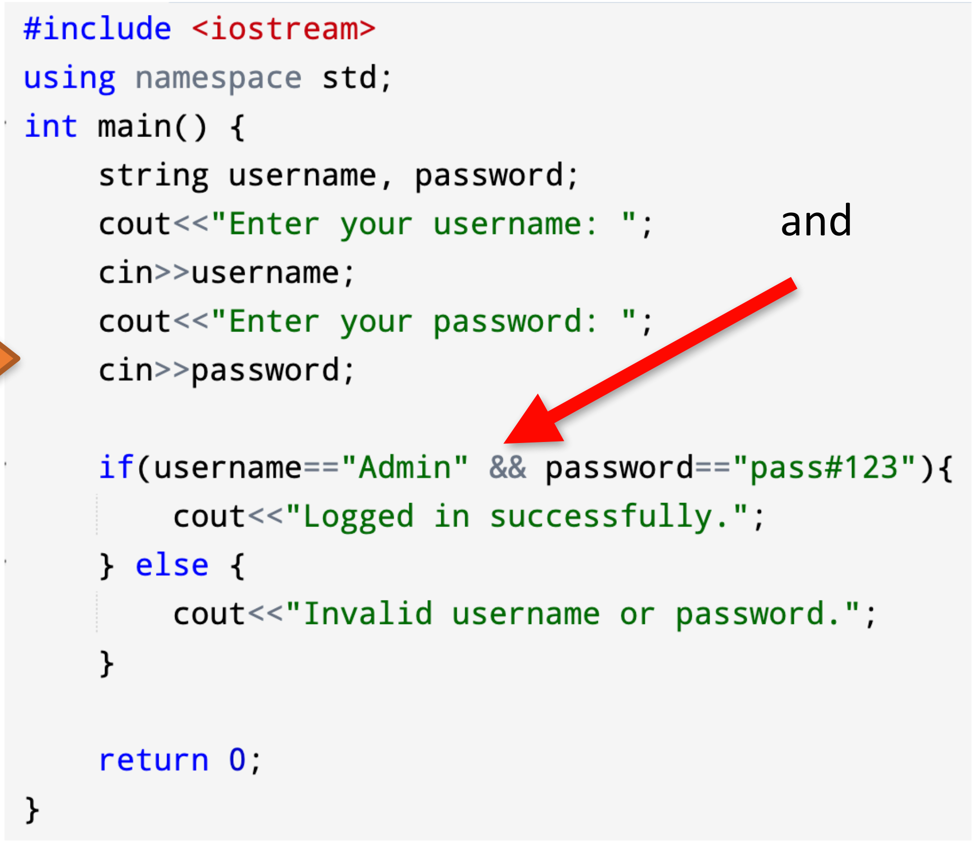


| Operator | Meaning | Effect |
|---|---|---|
| && | AND | Connects two expressions into one. Both expressions must be true for the overall expression to be true. |
| \|\| | OR | Connects two expressions into one. One or both expressions must be true for the overall expression to be true. It is only necessary for one to be true, and it does not matter which. |
| ! | NOT | The ! operator reverses the "truth" of an expression. It makes a true expression false, and a false expression true. |

# Logical Operators - Example 1 (&&)



```cpp
#include <iostream>
using namespace std;
int main() {
    string username,password;
    cout<<"Enter your username: ";
    cin>>username;
    cout<<"Enter your password: ";
    cin>>password;

    if(username=="Admin"){
        if(password=="pass#123"){
            cout<<"Logged in successfully.";
        } else {
            cout<<"Invalid password.";
        }
    } else {
        cout<<"Invalid username.";
    }

    return 0;
}
```



```cpp
#include <iostream>
using namespace std;
int main() {
    string username, password;
    cout<<"Enter your username: ";
    cin>>username;
    cout<<"Enter your password: ";
    cin>>password;

    if(username=="Admin" && password=="pass#123"){
        cout<<"Logged in successfully.";
    } else {
        cout<<"Invalid username or password.";
    }

    return 0;
}
```

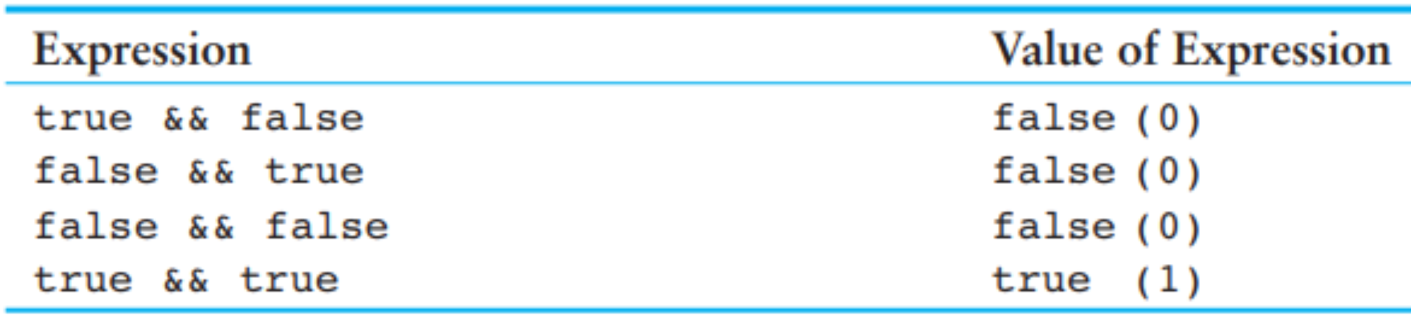
and

## Logical Operators - Example 2 (&&)

Create a C++ program that determines if a person is eligible to vote. The program should check if the person is 18 years or older and they are a citizen.

```cpp
#include <iostream>
using namespace std;
int main() {
    int age;
    char citizenship;

    cout << "Enter your age: ";
    cin >> age;

    cout << "Are you a citizen? (Y/N): ";
    cin >> citizenship;

    if (age >= 18 && citizenship == 'Y') {
        cout << "You are eligible to vote!\n";
    } else {
        cout << "Sorry, you are not eligible to vote.\n";
    }

    return 0;
}
```

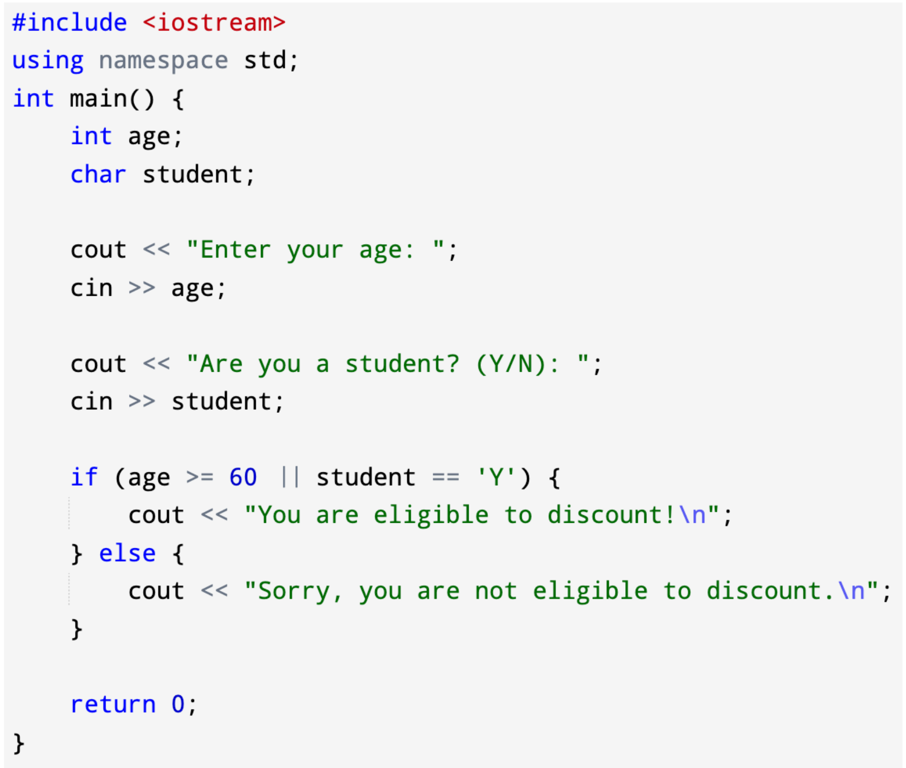
# Logical Operators - AND (&&)



| Expression | Value of Expression |
|---|---|
| `true && false` | `false (0)` |
| `false && true` | `false (0)` |
| `false && false` | `false (0)` |
| `true && true` | `true  (1)` |

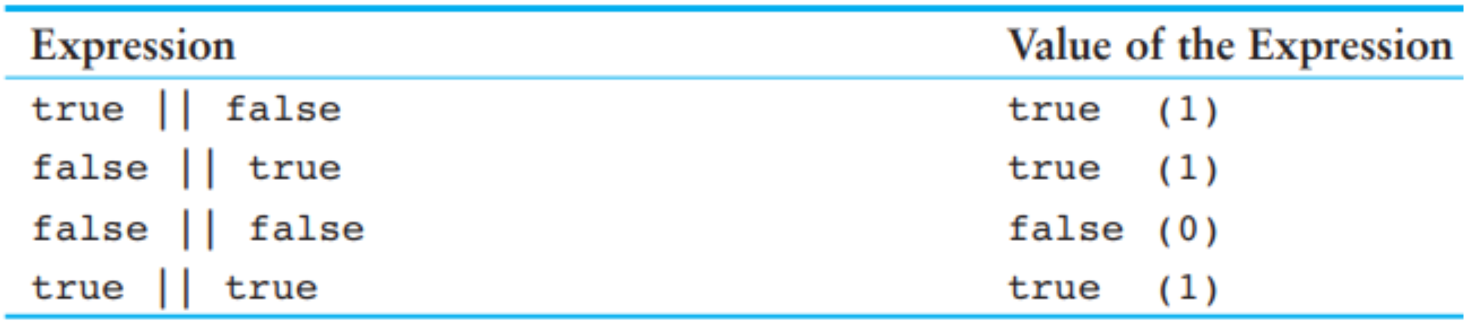
# Logical Operators - Example (||)

Develop a C++ program that determines if a person is eligible for a discount. Check if the person is a senior citizen (age 60 or above) or a student.



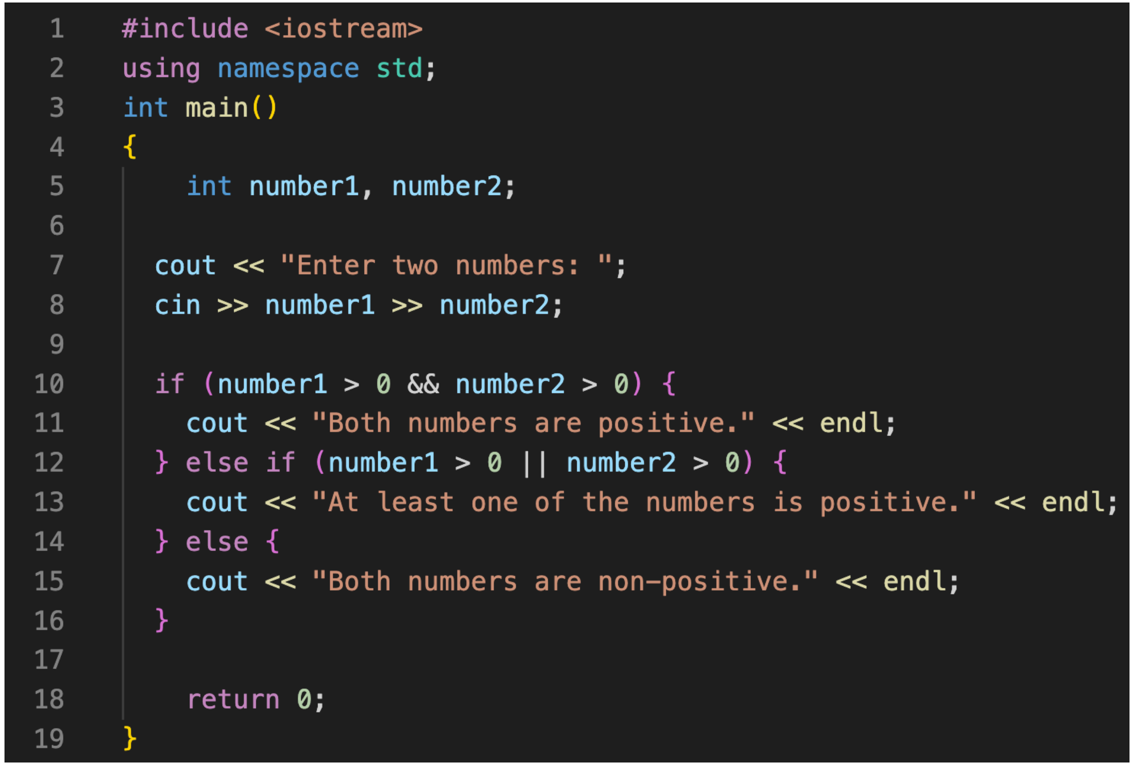
```cpp
#include <iostream>
using namespace std;
int main() {
    int age;
    char student;

    cout << "Enter your age: ";
    cin >> age;

    cout << "Are you a student? (Y/N): ";
    cin >> student;

    if (age >= 60 || student == 'Y') {
        cout << "You are eligible to discount!\n";
    } else {
        cout << "Sorry, you are not eligible to discount.\n";
    }

    return 0;
}
```

## Logical Operators - OR (||)



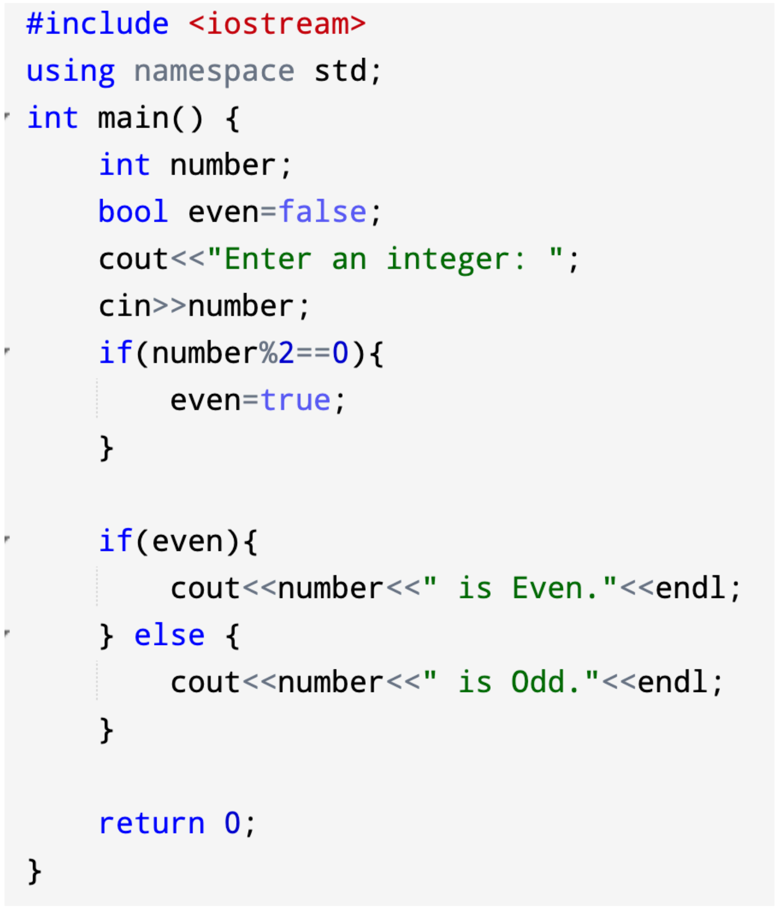
| Expression | Value of the Expression |
|---|---|
| `true \|\| false` | `true (1)` |
| `false \|\| true` | `true (1)` |
| `false \|\| false` | `false (0)` |
| `true \|\| true` | `true (1)` |

## Logical Operators - Example (&& and ||)

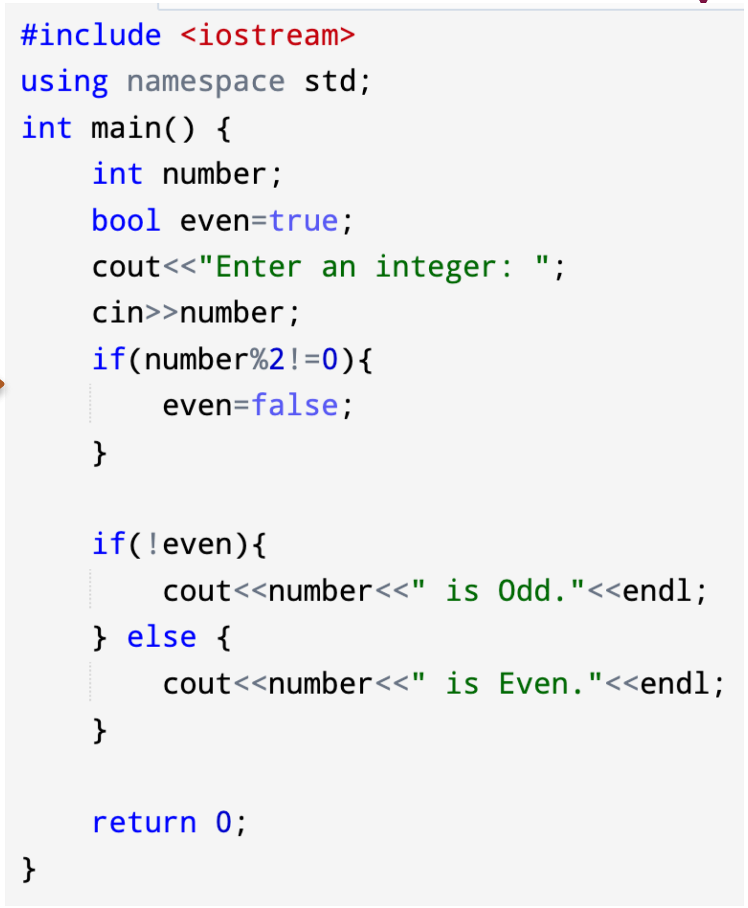


```cpp
#include <iostream>
using namespace std;
int main()
{
    int number1, number2;

  cout << "Enter two numbers: ";
  cin >> number1 >> number2;

  if (number1 > 0 && number2 > 0) {
    cout << "Both numbers are positive." << endl;
  } else if (number1 > 0 || number2 > 0) {
    cout << "At least one of the numbers is positive." << endl;
  } else {
    cout << "Both numbers are non-positive." << endl;
  }

    return 0;
}
```

# Logical Operators - NOT (!)

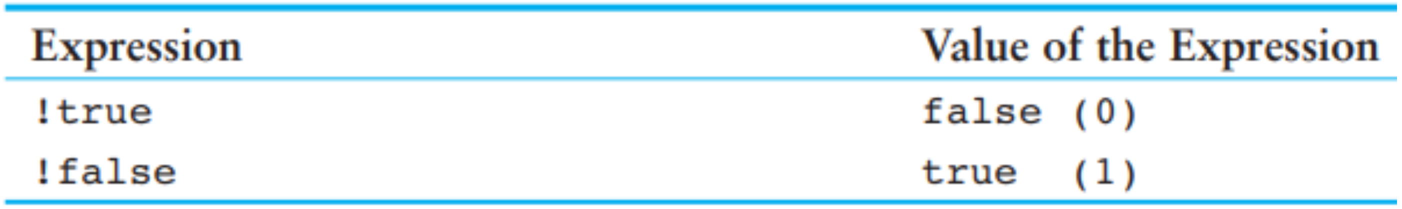


```
#include <iostream>
using namespace std;
int main() {
    int number;
    bool even=false;
    cout<<"Enter an integer: ";
    cin>>number;
    if(number%2==0){
        even=true;
    }

    if(even){
        cout<<number<<" is Even."<<endl;
    } else {
        cout<<number<<" is Odd."<<endl;
    }

    return 0;
}
```



```
#include <iostream>
using namespace std;
int main() {
    int number;
    bool even=true;
    cout<<"Enter an integer: ";
    cin>>number;
    if(number%2!=0){
        even=false;
    }

    if(!even){
        cout<<number<<" is Odd."<<endl;
    } else {
        cout<<number<<" is Even."<<endl;
    }

    return 0;
}
```

## Logical Operators - NOT (!)



| Expression | Value of the Expression |
|---|---|
| `!true` | `false (0)` |
| `!false` | `true  (1)` |

## Comparing Characters

```
#include <iostream>
using namespace std;
int main() {

    char ch;
    cout<<"Enter a digit or a letter: ";
    cin>>ch;

    if(ch >= '0' && ch <= '9'){
        cout<<"You entered a digit.\n";
    } else  if(ch >= 'A' && ch <= 'Z'){
         cout<<"You entered an uppercase letter.\n";
    } else  if(ch >= 'a' && ch <= 'z'){
         cout<<"You entered an lowercase letter.\n";
    } else {
        cout<<"That is not a letter or a digit.\n";
    }

    return 0;
}
```

| Character | ASCII Value |
|---|---|
| '0' – '9' | 48 – 57 |
| 'A' – 'Z' | 65 – 90 |
| 'a' – 'z' | 97 – 122 |
| blank | 32 |
| period | 46 |

## Blocks and Variable Scope

The scope of a variable is limited to the block in which it is defined. C++ allows you to create variables almost anywhere in a program.

```cpp
#include <iostream>
using namespace std;
int main(){

    int number;
    cout << "Enter a number greater than 0: ";
    cin >> number;

    if (number > 0){
        int number; // Another variable named number.
        cout << "Now enter another number: ";
        cin >> number;
        cout << "The second number was " << number << endl;
    }
    cout << "Your first number was " << number << endl;

    return 0;
}
```

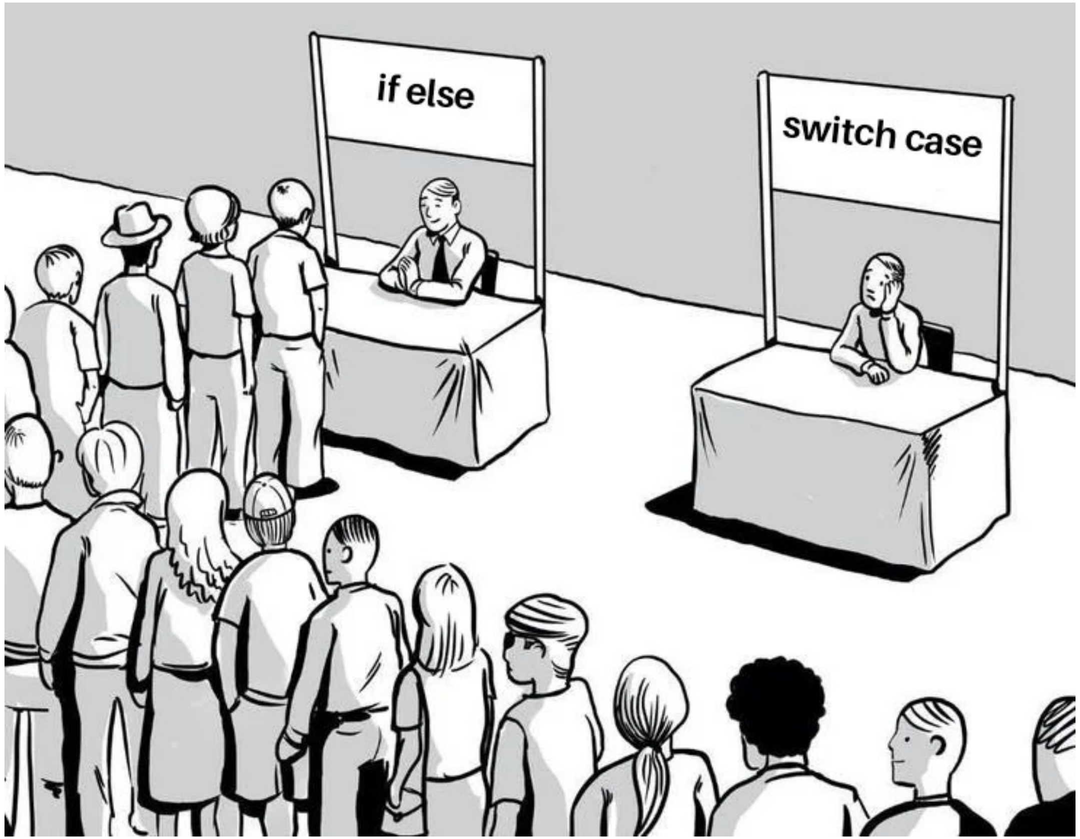
```
Enter a number greater than 0: 17
Now enter another number: 25
The second number was 25
Your first number was 17
```

## The `switch` Statement

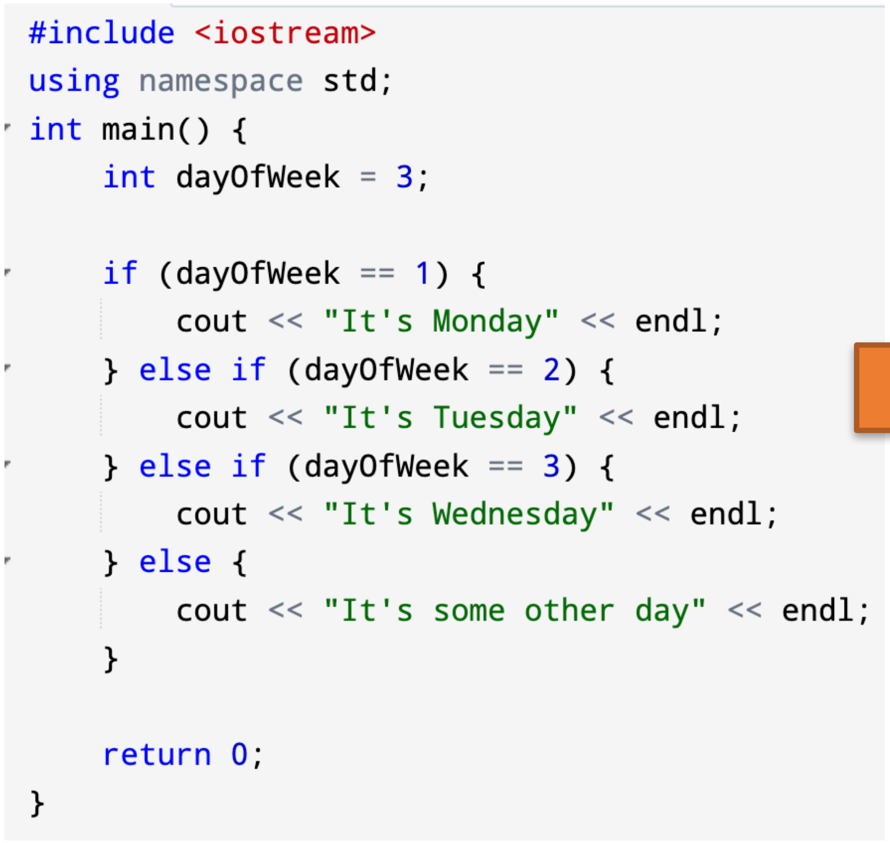
- Switch statement: Determines program branching based on the value of a variable or expression.
- Branching: Occurs when one part of the program leads to the execution of another part.
- If/else if statement: Allows branching into various paths based on true conditions in a series of tests.
- Switch vs if/else if: Switch tests integer expression values for branching, while if/else if tests relational conditions.

```
switch(value){
    case Choice1:
        Statement1;
          break;
    case Choice2:
        Statement2;
         break;
    case Choice-n:
        Statement-n;
         break;
     default:
        default statement;
}
```

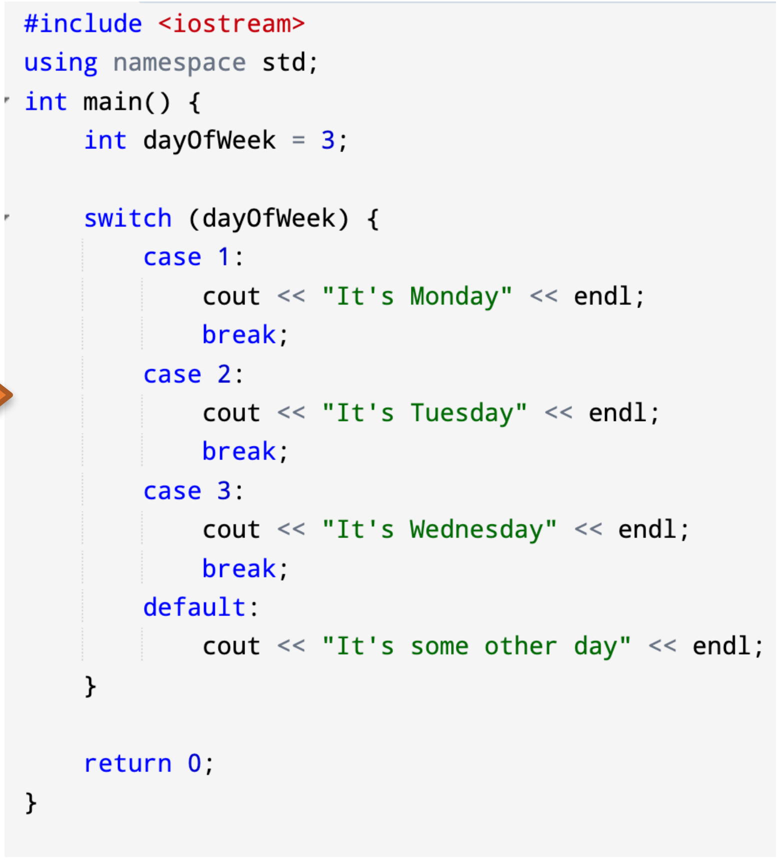
## The `switch` Statement



# The switch Statement - Example 1

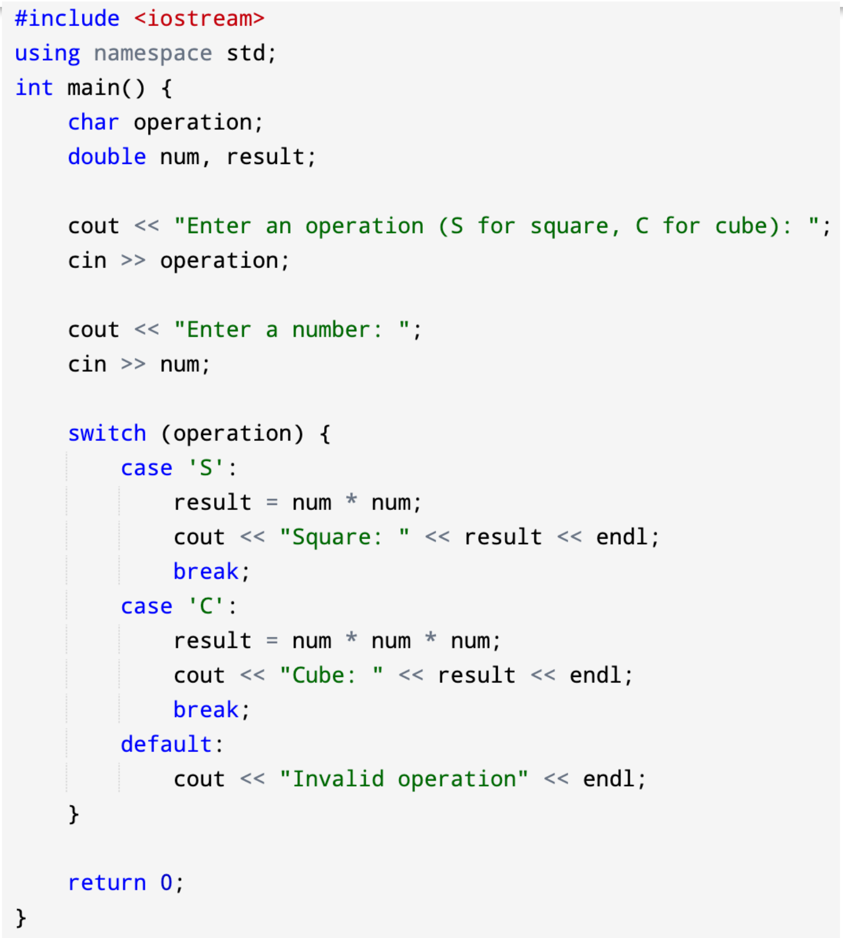


```cpp
#include <iostream>
using namespace std;
int main() {
    int dayOfWeek = 3;

    if (dayOfWeek == 1) {
        cout << "It's Monday" << endl;
    } else if (dayOfWeek == 2) {
        cout << "It's Tuesday" << endl;
    } else if (dayOfWeek == 3) {
        cout << "It's Wednesday" << endl;
    } else {
        cout << "It's some other day" << endl;
    }

    return 0;
}
```



```cpp
#include <iostream>
using namespace std;
int main() {
    int dayOfWeek = 3;

    switch (dayOfWeek) {
        case 1:
            cout << "It's Monday" << endl;
            break;
        case 2:
            cout << "It's Tuesday" << endl;
            break;
        case 3:
            cout << "It's Wednesday" << endl;
            break;
        default:
            cout << "It's some other day" << endl;
    }

    return 0;
}
```

## The switch Statement - Example 2



```cpp
#include <iostream>
using namespace std;
int main() {
    char operation;
    double num, result;

    cout << "Enter an operation (S for square, C for cube): ";
    cin >> operation;

    cout << "Enter a number: ";
    cin >> num;

    switch (operation) {
        case 'S':
            result = num * num;
            cout << "Square: " << result << endl;
            break;
        case 'C':
            result = num * num * num;
            cout << "Cube: " << result << endl;
            break;
        default:
            cout << "Invalid operation" << endl;
    }

    return 0;
}
```

Thank You

TISHK INTERNATIONAL UNIVERSITY
TIU
ERBIL 2008