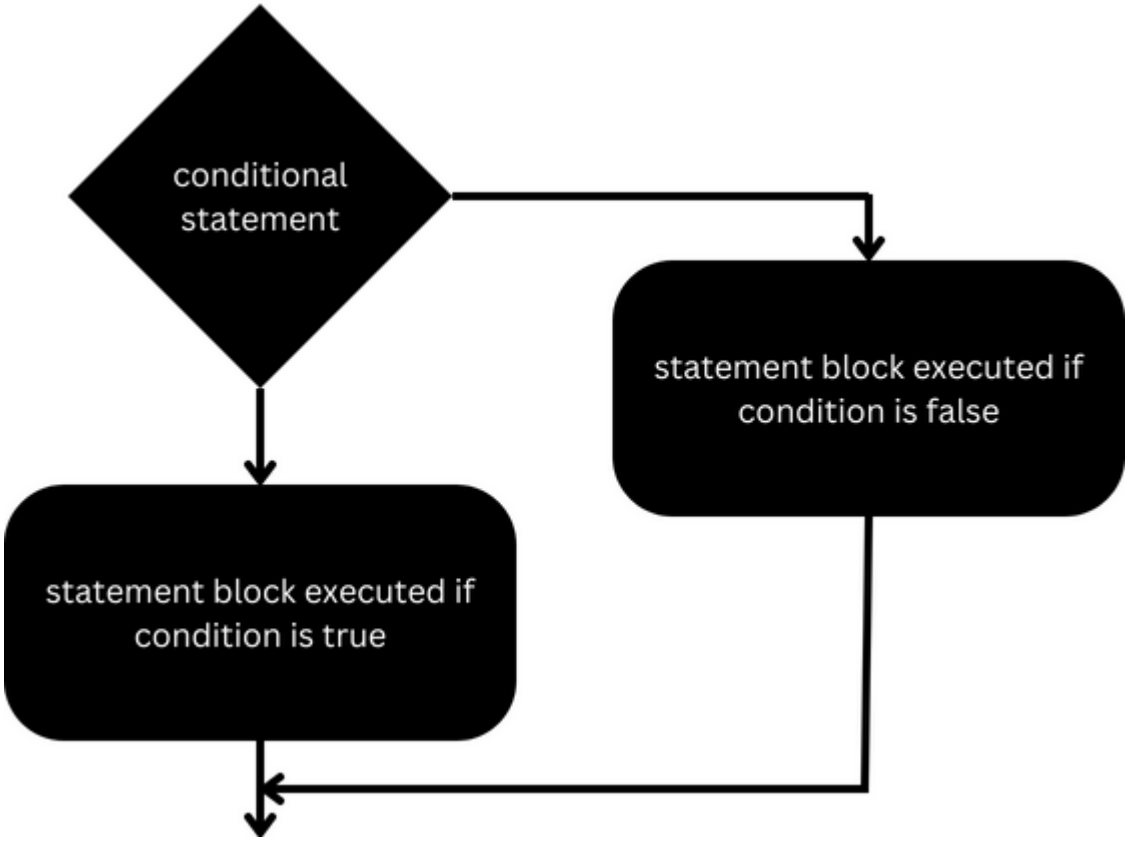
Conditionals and Loops

Conditional statements (if else)

Conditionals are used to execute a certain section of code only if some specific condition is fulfilled, and optionally execute other statements if the given condition is false. The result of given conditional expression must be either true or false.



Different variations of this conditional statements are :

Single if statement:

This is the simplest form of an if statement. It executes a block of code if the specified condition is true.

Syntax

```
if(test_expression){

      //Statements to be executed only when test_expression is true

}
```

Example :

```
#include <iostream>
using namespace std;
int main()
{
    int age = 25;
    if (age >= 18) {
        cout << "You are young" << endl;
    }
    return 0;
}
```

Output : You are young

if - else statement:

An if-else statement allows executing different blocks of code based on whether the condition is true or false.

Syntax :

```
if(test_expression)
{
      //statements to be executed when test expression is true
}
else
{
      //else part
}
```

Example :

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

int main() {
    int age = 15;
    if (age >= 18) {
        cout << "You are an adult." << endl;
    } else {
        cout << "You are a minor." << endl;
    }
    return 0;
}
```

Output : You are a minor.

if - else if statement:

This construct allows checking multiple conditions and executing corresponding blocks of code accordingly.

Syntax:

```
if(expression1)
{
    //executes when expression1 is true;
}
else if(expression2)
{
    //executes when expression2 is true;
}
else if(expression3)
{
    //executes when expression3 is true;
}
else
{
    //executes when all above expressions fails
}
```

Example :

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

int main() {
    int date = 2;
    if (day == 1) {
        cout << "It's the start of the week." << endl;
    } else if (day == 2) {
        cout << "abc" << endl;
    } else {
        cout << "It's later in the week." << endl;
    }
    return 0;
}
```

Output : abc

Nested if statement :

You can use if statements inside other if statements to create nested conditions.

Syntax:

```
if(expression1)
{
    //executes when expression1 is true;
     if(expression2)
     {
           //executes when expression2 is true;
     }
     else
     {
          //executes when expression2 is false
     }
}
```

Example

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

int main() {
    int age = 25;
    if (age >= 18) {
        if (age < 21) {
            cout << "You are not an adult" << endl;
        } else {
            cout << "You are an adult " << endl;
        }
    } else {
        cout << "You are a minor." << endl;
    }
    return 0;
}
```

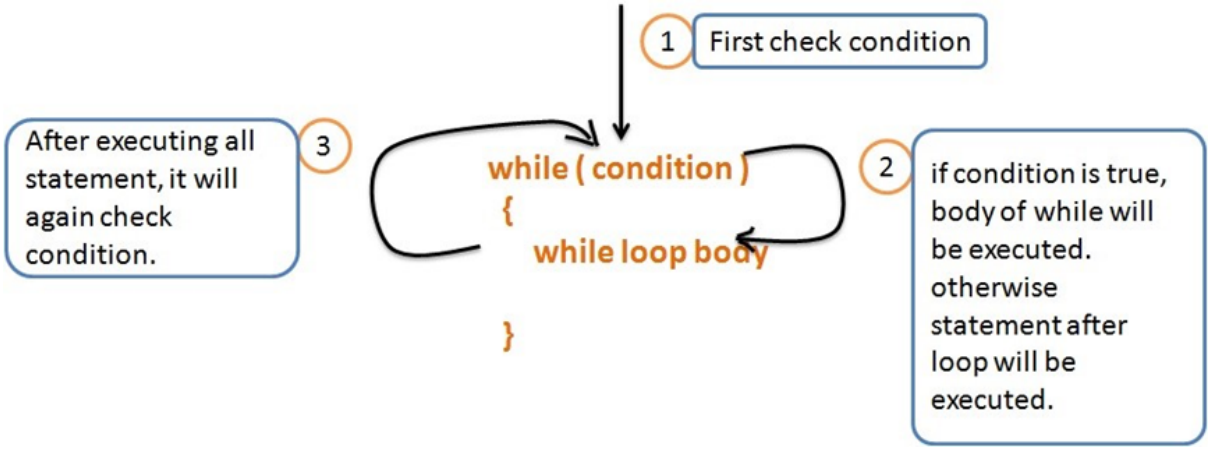
Output: You are an adult.

While loop

A while loop is a control flow statement in programming that repeatedly executes a block of code as long as a specified condition is true. The loop will continue to execute until the condition evaluates to false. The syntax of a while loop is as follows:

Syntax:

```
while (condition) {
    // Code block to be executed as long as the condition is true
}
```



Example

```
int main()
{
    int i=1;
    while(i<=5)
    {
        cout<<i<<" ";
        i++;
    }
}
```

Output : 1 2 3 4 5

Switch case block

In C++, the switch-case statement allows you to perform different actions based on the value of a given expression. **The syntax of the switch-case statement in C++ is as follows:**

```
switch (expression) {
    case value1:
        // Code to execute when expression equals value1
        break;
    case value2:
        // Code to execute when expression equals value2
        break;
    // More case statements as needed
    default:
        // Code to execute if none of the case values match the
expression
}
```

Example:

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

int main() {
    int dayOfWeek = 3;

    switch (dayOfWeek) {
        case 1:
            cout << "Monday" << endl;
            break;
        case 2:
            cout << "Tuesday" << endl;
            break;
        case 3:
            cout << "Wednesday" << endl;
            break;
        case 4:
            cout << "Thursday" << endl;
            break;
        case 5:
            cout << "Friday" << endl;
            break;
        case 6:
            cout << "Saturday" << endl;
            break;
        case 7:
            cout << "Sunday" << endl;
            break;
        default:
            cout << "Invalid day" << endl;
    }

    return 0;
}
```

Output : Wednesday

Ternary operator

The ternary operator, also known as the conditional operator, is a concise way to write simple conditional expressions in various programming languages. It allows you to evaluate a condition and choose one of two values based on whether the condition is true or false. **The syntax of the ternary operator is as follows:**

```
condition ? value_if_true : value_if_false
```

Example:

```
int main()
{
    int max = 5 > 3 ?5:3;
    cout<<max;
}
```

Output: 5

Do while loop

The do-while loop is a control flow statement in programming that is similar to the while loop. The key difference is that the do-while loop guarantees the loop body will be executed at least once before checking the loop condition. After the first iteration, the loop continues to execute as long as the specified condition is true. **The syntax of the do-while loop is as follows:**

```
do {
    // Code block to be executed
} while (condition);
```

Example :

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

int main() {
    int counter = 1;
    do {
        cout << counter << endl;
        counter++;
    } while (counter <= 5);

    return 0;
}
```

Output: 1 2 3 4 5