Control Structures in C++

Control structures in C++ are the fundamental building blocks that determine the flow of a program. They enable programmers to specify the sequence in which statements are executed, allowing for decision-making, looping, and branching. Mastering these structures is crucial for writing efficient and logical programs.

Types of Control Structures

1. **Decision-Making Control Structure**
2. **Looping Control Structure**
3. **Jump Control Structure**

---

1. Decision-Making Control Structure

These structures allow the program to make decisions based on conditions.

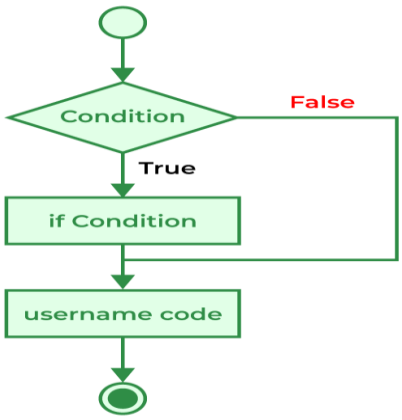
a. if Statement

Definition: Executes a block of code if a specified condition is true.

Syntax:

```
if (condition) {
   // Code to execute if condition is true
}
```

Flowchart:



Example:

```
int age = 18;
if (age >= 18) {
```

```
    cout << "You are eligible to vote.";
  }
```

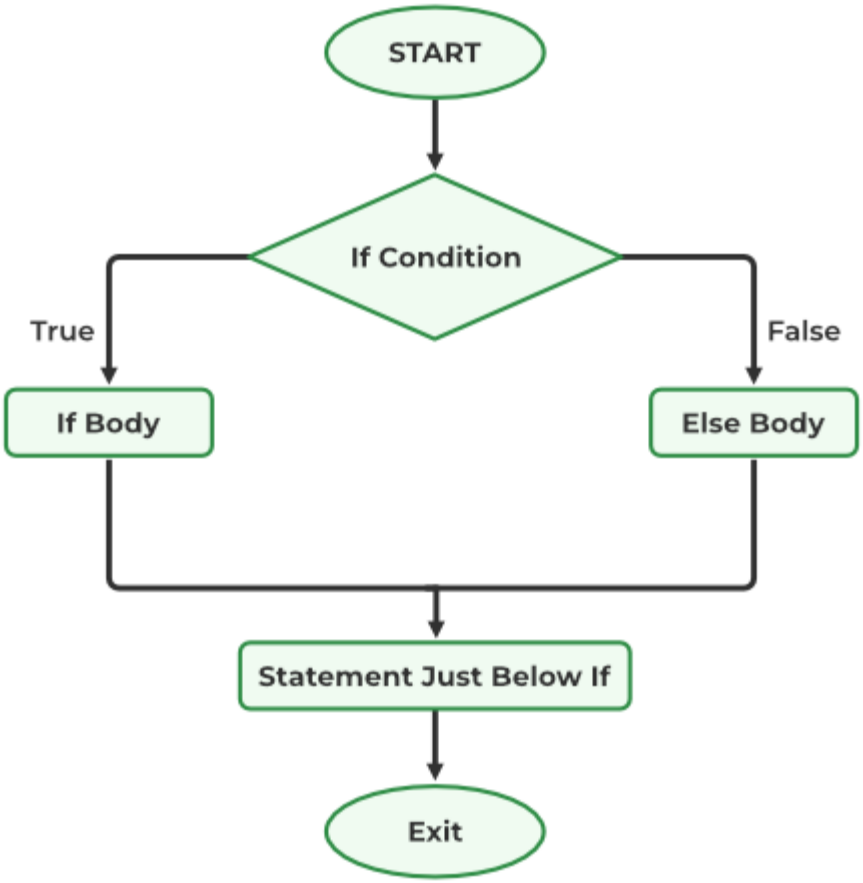
b. if-else Statement

Definition: Executes one block of code if the condition is true and another if it is false.

Syntax:

```
if (condition) {
  // Code if condition is true
} else {
  // Code if condition is false
}
```

Flowchart:



Example:

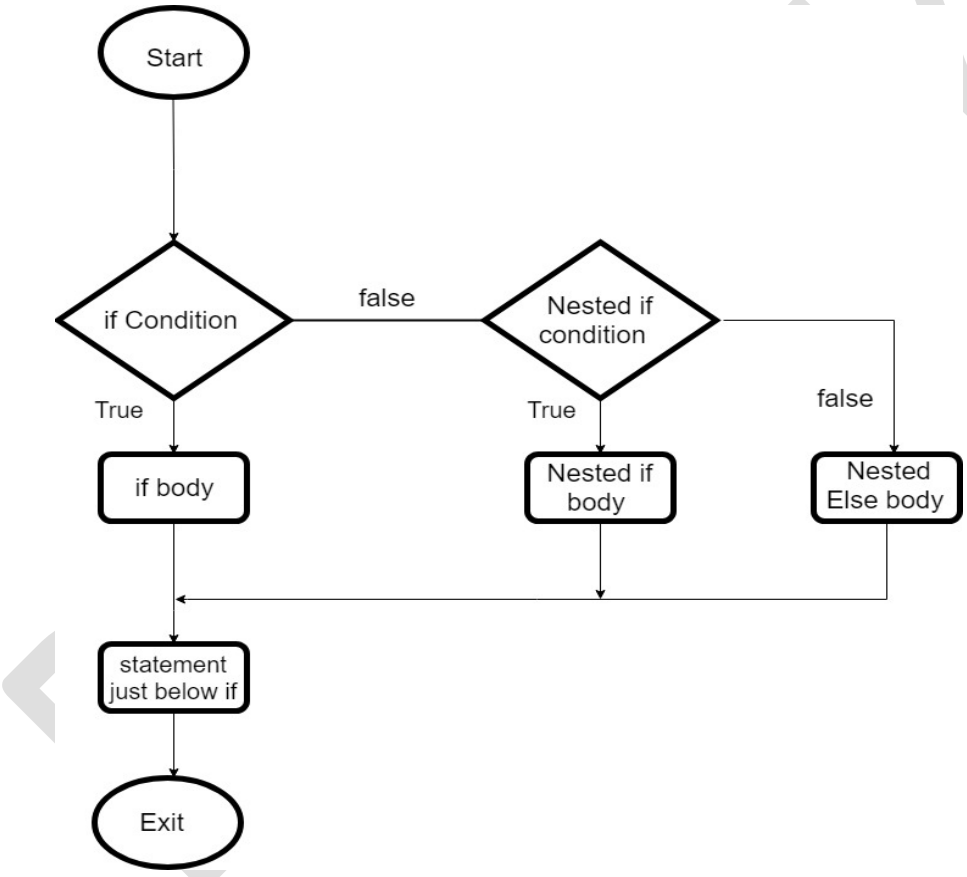
```
int num = 10;
```

```
if (num % 2 == 0) {
    cout << "Even number.";
} else {
    cout << "Odd number.";
}
```

c. Nested if

Definition: An if statement inside another if statement.

Flowchart:



Example:

```
int marks = 85;
if (marks >= 50) {
   if (marks >= 75) {
      cout << "Distinction";
   } else {
      cout << "Pass";
   }
} else {
   cout << "Fail";
}
```

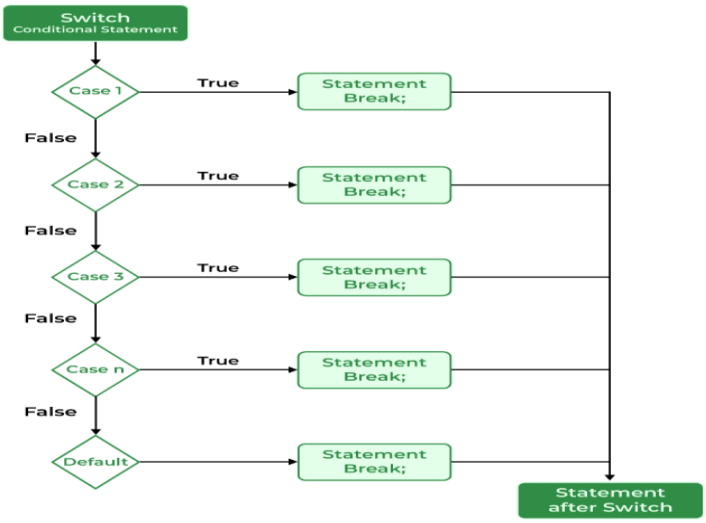
d. switch Statement

Definition: Used to execute one code block from multiple options.

Syntax:

```
switch (variable) {
   case value1:
      // Code for value1
      break;
   case value2:
      // Code for value2
      break;
   default:
      // Code if none of the cases match
}
```

Flowchart:



Example:

```
char grade = 'A';
switch (grade) {
   case 'A':
      cout << "Excellent!";
      break;
   case 'B':
      cout << "Good!";
      break;
   default:
      cout << "Invalid grade.";
}
```

Looping Control Structure

These structures are used to repeat a block of code as long as a condition is true.

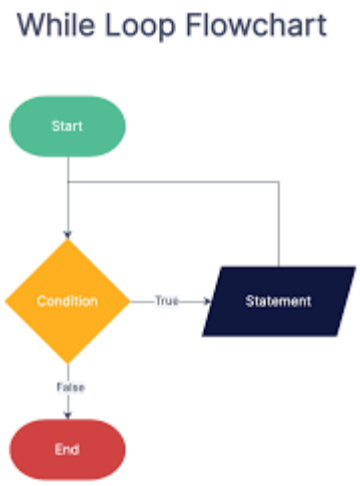
a.while Loop

Definition: Repeats a block of code while a condition is true.

Syntax:

```
while (condition) {
   // Code to execute
}
```

Flowchart:



Example:

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

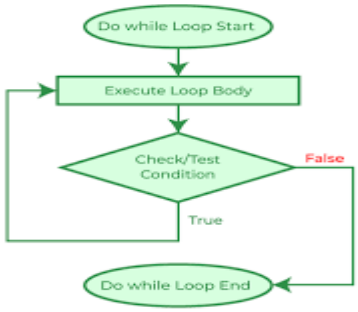
b.do-while Loop

Definition: Executes a block of code at least once, then repeats it while the condition is true.

Syntax:

```
do {
    // Code to execute
} while (condition);
```

Flowchart:



Example:

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

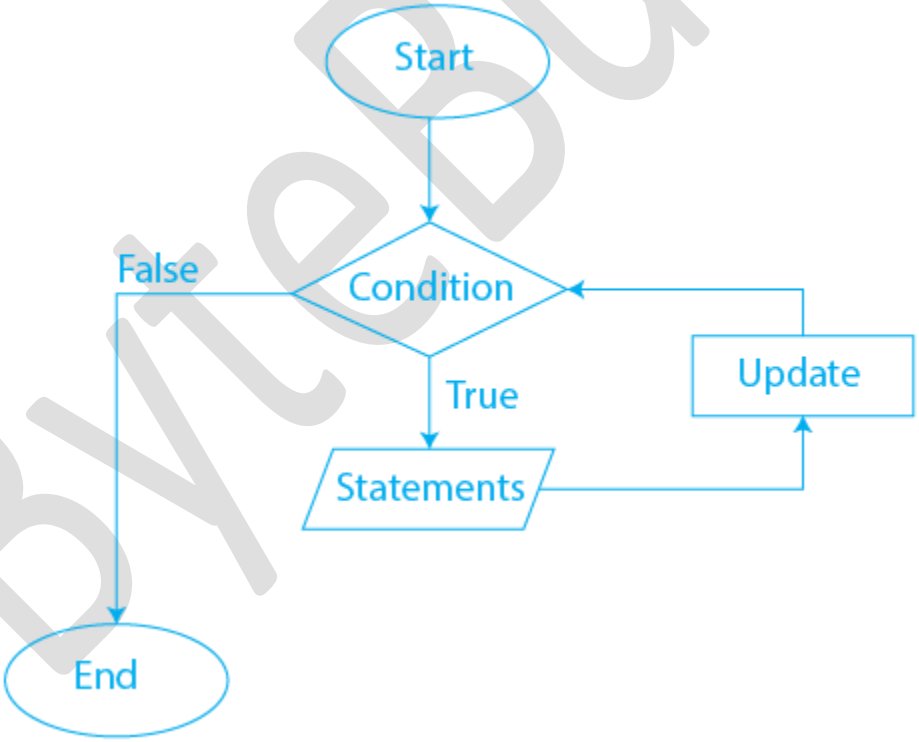
c.for Loop

Definition: Executes a block of code a specific number of times.

Syntax:

```
for (initialization; condition; increment/decrement) {
    // Code to execute
}
```

Flowchart:



Example:

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

d.Nested Loops

Definition: A loop inside another loop.

Example:

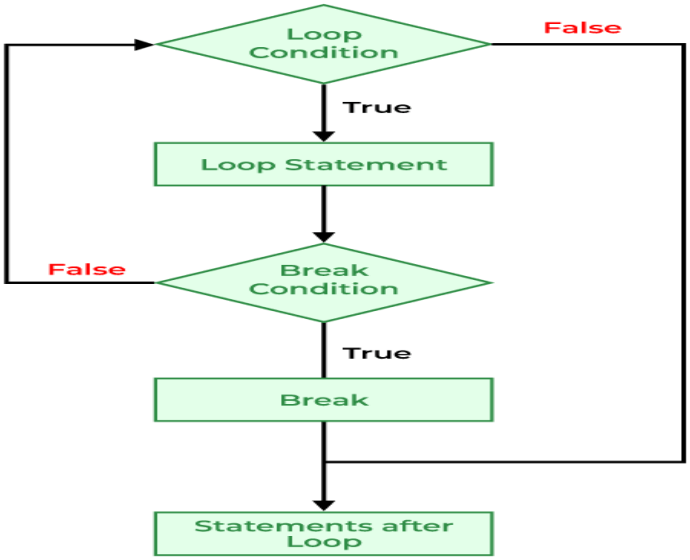
```
for (int i = 1; i <= 3; i++) {
   for (int j = 1; j <= 2; j++) {
      cout << "i=" << i << " j=" << j << endl;
   }
}
```

Jump Control Structure

These structures alter the flow of the program by jumping to another part of the code.

a.break Statement

Definition: Exits a loop or a switch statement prematurely.

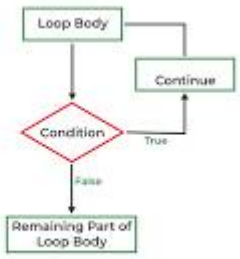


Example:

```
for (int i = 1; i <= 5; i++) {
   if (i == 3) {
      break;
   }
   cout << i << " ";
}
```

b.continue Statement

Definition: Skips the current iteration of a loop and moves to the next iteration.

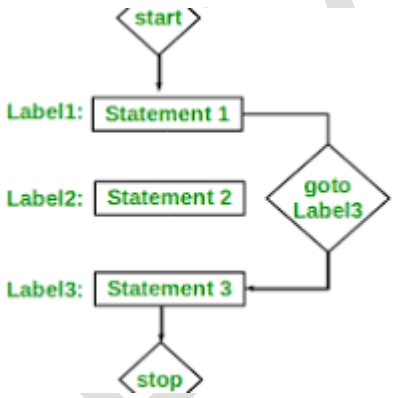


Example:

```
for (int i = 1; i <= 5; i++) {
   if (i == 3) {
      continue;
   }
   cout << i << " ";
}
```

c.goto Statement

Definition: Transfers control to a labeled statement. Its use is generally discouraged as it can make code harder to read and maintain.



Example:

```
int x = 10;
if (x > 0) {
   goto positive;
}
negative:
   cout << "Negative number";
   return 0;
```

```
    positive:
        cout << "Positive number";
```

Comprehensive Example

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

int main() {
    // Sequential Control
    cout << "Sequential Control Example" << endl;

    // Decision-Making Control
    int num;
    cout << "Enter a number: ";
    cin >> num;
    if (num > 0) {
        cout << "Positive number" << endl;
    } else if (num < 0) {
        cout << "Negative number" << endl;
    } else {
        cout << "Zero" << endl;
    }

    // Looping Control
    for (int i = 1; i <= 5; i++) {
        cout << i << " ";
    }
    cout << endl;

    // Jump Control
    for (int i = 1; i <= 5; i++) {
```

```
        if (i == 3) {
            continue;
        }
        cout << i << " ";
    }

    return 0;
}
```