Program Control Functions

Module 2

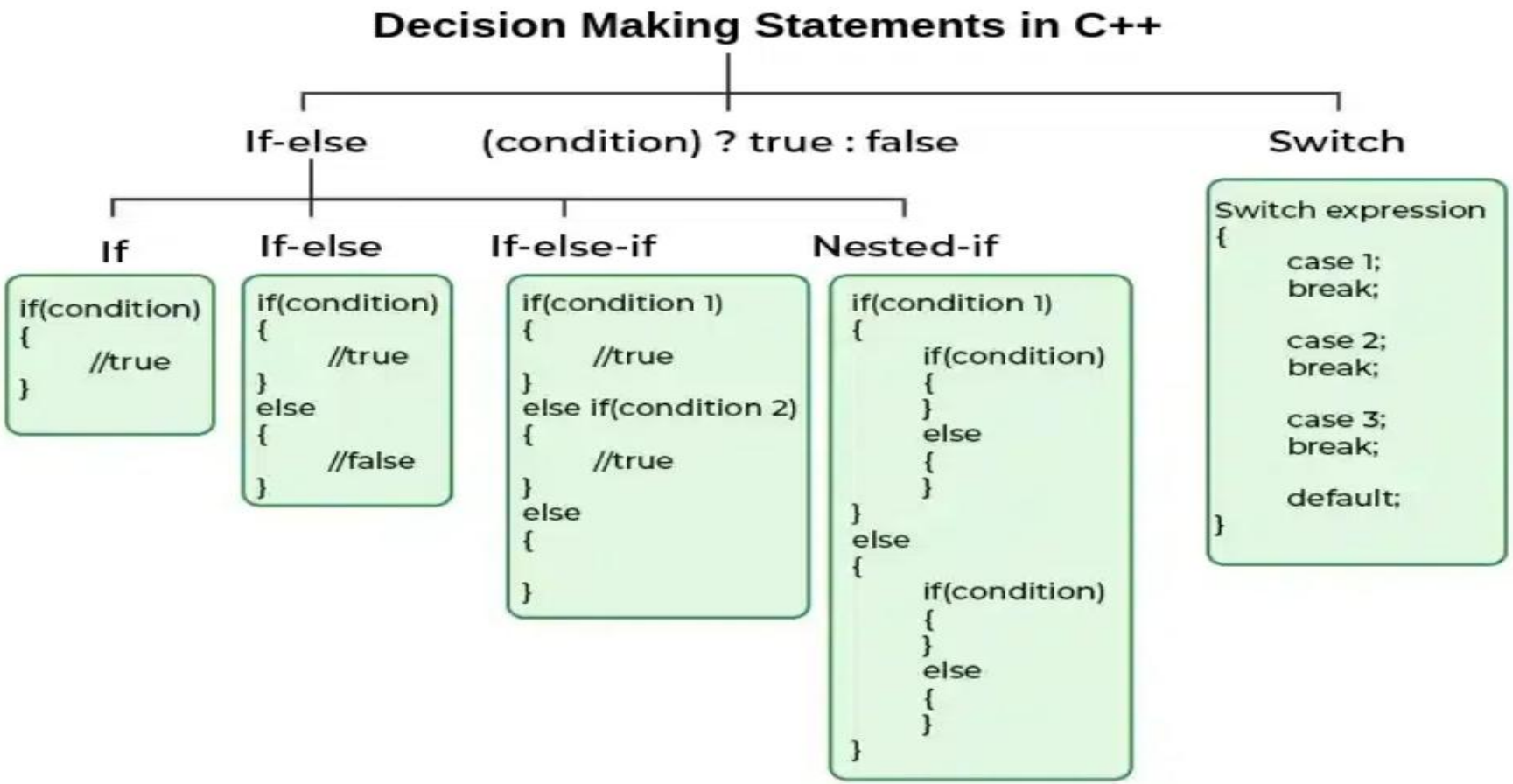
Outline

- Decision Making and Branching Control Structures:
  - Two Way
  - Selection,
  - Multiway Selection
- Looping Control Structures,
- Flag Concept,
- Counting Loops
- Documentation and Making Source Code Readable

Decision Making in C++

- **Decision-making**: Choosing which part of code to execute based on a condition.
- Uses conditional statements (decision control statements) to run specific code blocks depending on the situation and result.
- Widely used in real-world applications like billing, grading, authentication, etc.
- Essential for making programs interactive and intelligent.

Types of Decision-Making Statements in C++



if Statement

- if statement - simplest decision-making statement in C++.
- Executes a block of code only if condition is true.
- Body of the if runs when the condition evaluates to true.

The general form of a simple if statement is:

```
if (test_condition)
 {
        statement-block;
 }
statement x;
```

Operators used to specify the test condition for selection or iteration statements

Relational Operators

Operator	Meaning	Example (a=5 , b=7)	Result
==	Equal to	a == b	false (0)
!=	Not equal to	a != b	true (1)
>	Greater than	a > b	false (0)
<	Less than	a < b	true (1)
>=	Greater than or equal to	a >= 5	true (1)
<=	Less than or equal to	b <= 5	false (0)

To specify compound condition

- **Logical operators**

Operator	Meaning	Example (a=5, b=7)	Result
&&	Logical AND (true if both are true)	(a < b) && (a > 0)	true (1)
||	Logical OR (true if atleast is true)	(a == 5 || b == 10)	True(1)
!	Logical NOT (inverts the condition)	!(a < b)	false (0)

Specifying the test condition

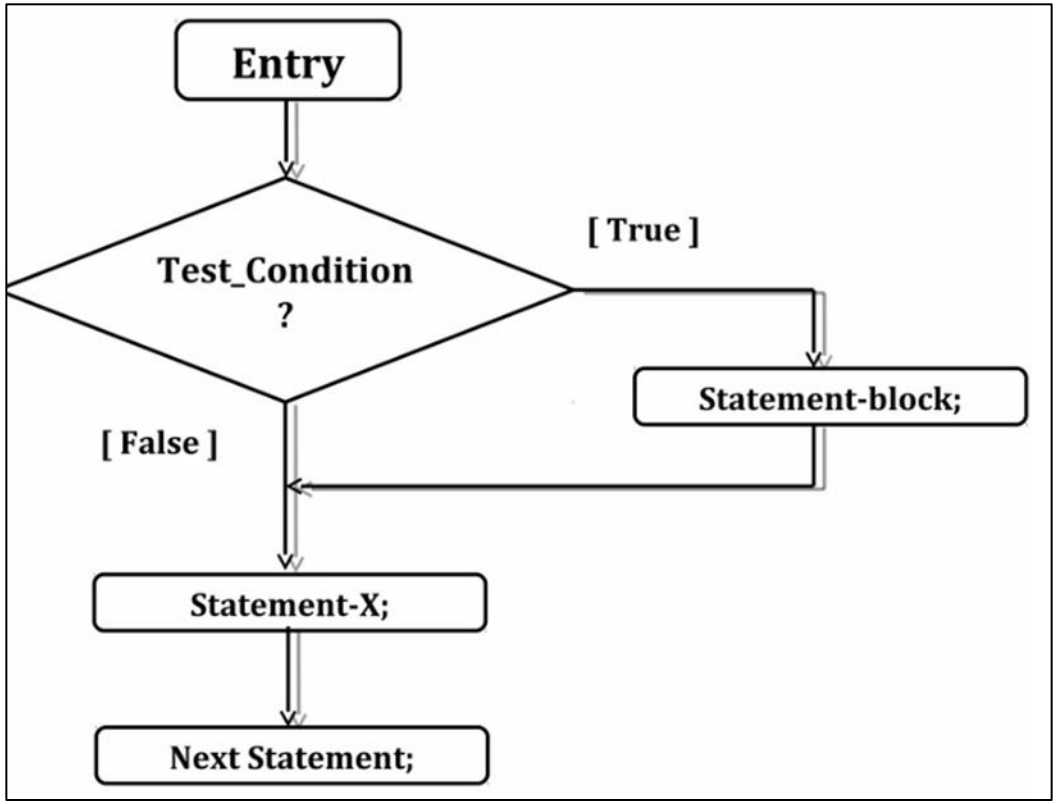
Basic Test condition

variable | expression | constant relational_operator variable | expression | constant

Compound Test condition

(condition1) logical_operator (condition2) [logical_operator (condition3) ...]

Flow Chart


Entry
Test_Condition
?
[True]
Statement-block;
[False]
Statement-X;
Next Statement;

Understanding True and False in c++

Rules for Truth/False in C++

- **False** - if the value of the expression is **0**.
- **True** - if the value of the expression is **non-zero** (positive or negative).

Precedence & Associativity Table

Precedence Level	Operator(s)	Description	Associativity
1 (Highest)	!	Logical NOT	Right → Left
2	<, <=, >, >=	Relational (less, less equal, greater, greater equal)	Left → Right
3	==, !=	Equality (equal to, not equal to)	Left → Right
4	&&	Logical AND	Left → Right
5 (Lowest)	||	Logical OR	Left → Right

if Statement example

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

int main() {
    int age = 19;

    if (age > 18) {
        cout << "allowed to vote";
    }
    return 0;
}
```

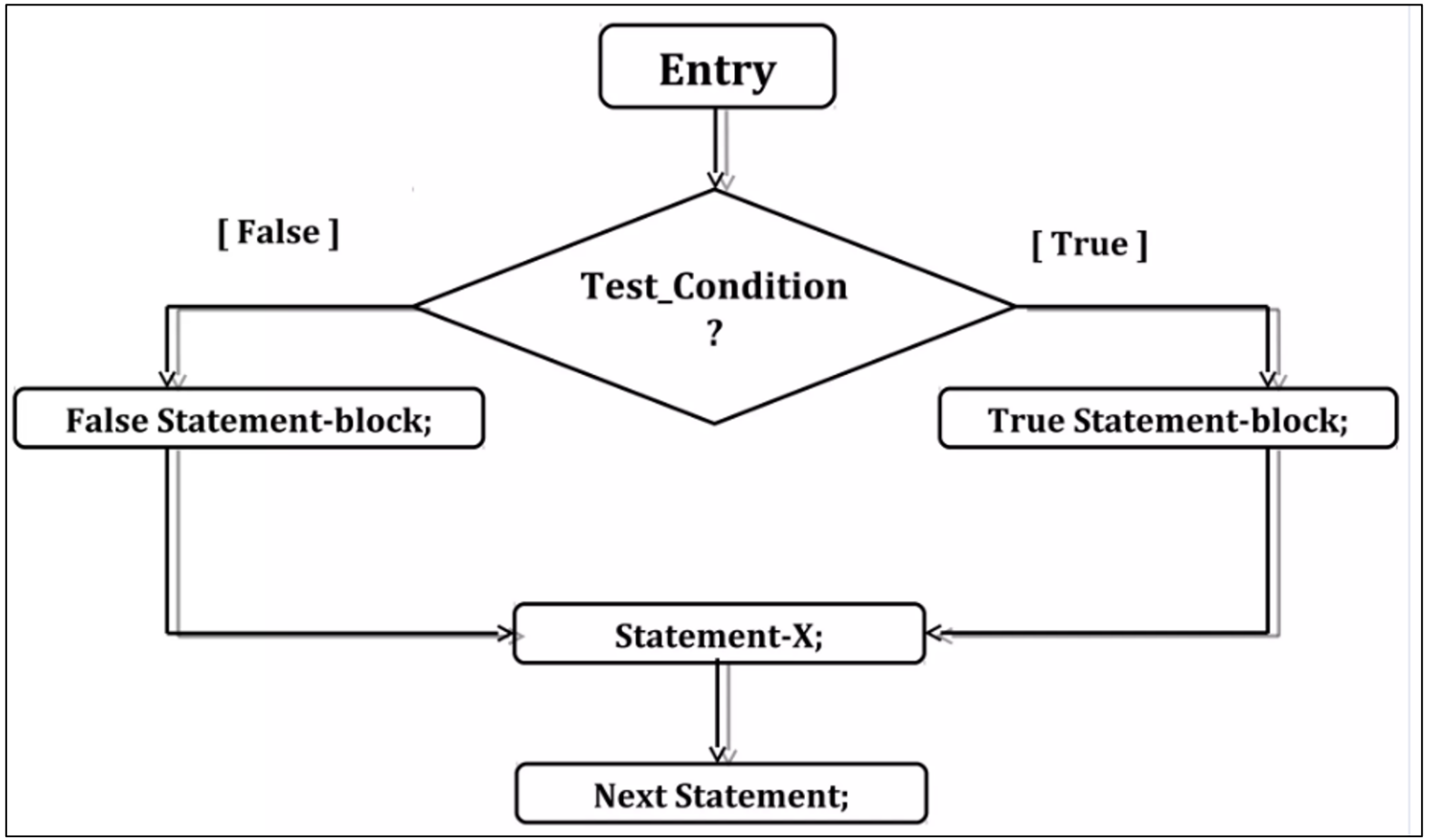
```
allowed to vote
```

if-else Statement

- If condition is true - execute code in if block.
- If condition is false - execute code in else block.
- General syntax:

```
if (test_condition)
      {
      True block statements;
      }
else
      {
      False block statements;
      }
statement-x;
```

Flow chart


Entry
[False]
Test_Condition
?
[True]
False Statement-block;
True Statement-block;
Statement-X;
Next Statement;

if-else Statement example

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

int main() {
    int n = -5;
    // Using if-else to determine if the number is positive
    // or non positive
    if (n > 0) {
        cout << "number is positive.";
    }
    else {
        cout << "number is non-positive.";
    }
    return 0;
}
```

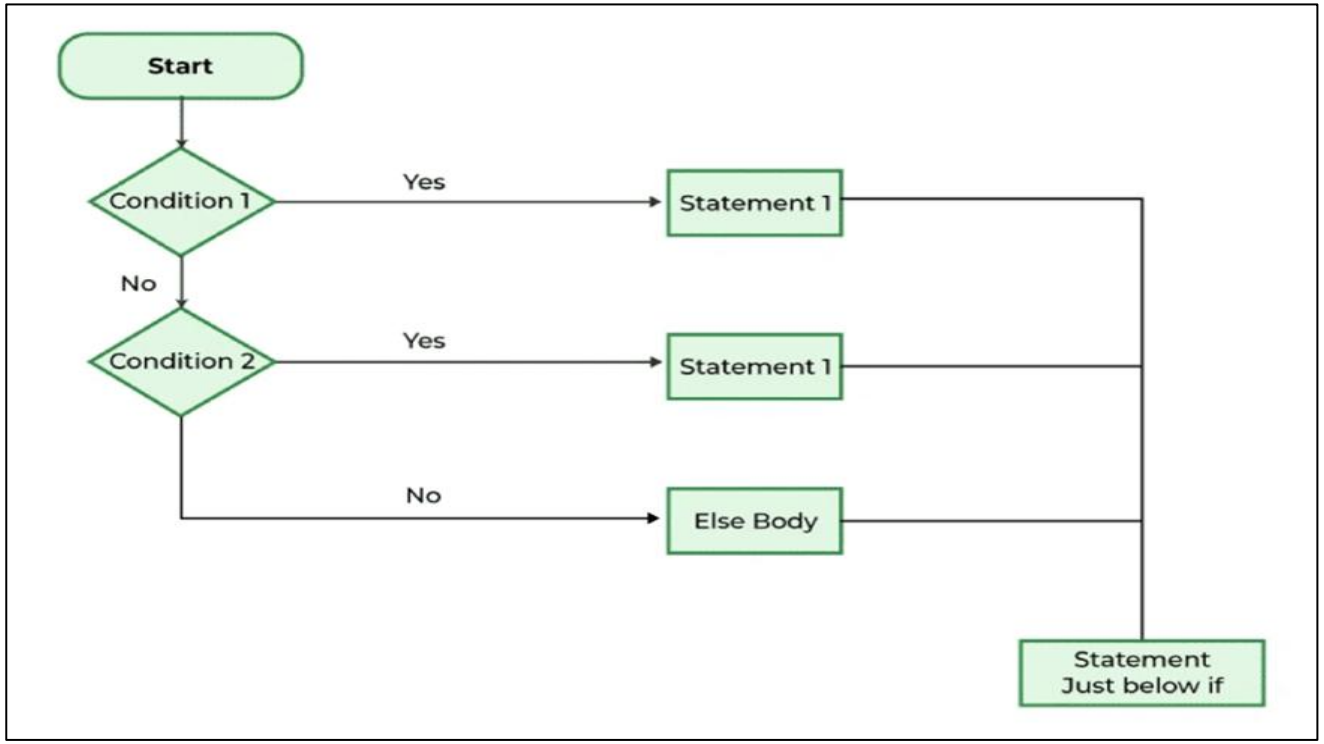
```
number is non-positive.
```

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

int main() {
    int a = 5, b = 10, c = 5;

    if (a == c || b > a && !(b == 5)) {
        cout << "Condition is true" << endl;
    } else {
        cout << "Condition is false" << endl;
    }

    return 0;
}
```

if else if Ladder

- **if-else if ladder** handles multiple conditions in sequence.
- **First condition true** execute its block, skip others.
- **Else if** checked only if previous conditions are false.
- **Else** - executes if none of the conditions are true.

Flow Chart


Start
Condition 1
Yes
Statement 1
No
Condition 2
Yes
Statement 1
No
Else Body
Statement
Just below if

Example

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

int main() {
    int number;
    cout << "Enter a number: ";
    cin >> number;

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

    return 0;
}
```

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

int main() {
    int Mark;
    cout << "Enter your marks: ";
    cin >> Mark;

    if (Mark >= 50 && Mark < 60) {
        cout << "Your grade is D" << endl;
    }
    else if (Mark >= 60 && Mark < 70) {
        cout << "Your grade is C" << endl;
    }
    else if (Mark >= 70 && Mark < 80) {
        cout << "Your grade is B" << endl;
    }
    else if (Mark >= 80 && Mark < 90) {
        cout << "Your grade is A" << endl;
    }
    else if (Mark >= 90) {
        cout << "Your grade is A+" << endl;
    }
    else {
        cout << "You have failed" << endl;
    }

    return 0;
}
```

```
Enter your marks: 45
You have failed
```

Nested if else

- Nested if-else - an if inside another if.
- Used for complex decision-making with multiple conditions.
- Inner if executes only when the outer condition is true.
- Helps check multiple conditions step by step.

Example

```
int main() {
    int marks;
    cout << "Enter your marks: ";
    cin >> marks;

    if (marks >= 50) {
        if (marks >= 90) {
            cout << "Grade: A+" << endl;
        }
        else if (marks >= 80) {
            cout << "Grade: A" << endl;
        }
        else if (marks >= 70) {
            cout << "Grade: B" << endl;
        }
        else if (marks >= 60) {
            cout << "Grade: C" << endl;
        }
        else {
            cout << "Grade: D" << endl;
        }
    }
    else {
        cout << "Fail" << endl;
    }

    return 0;
}
```

Programs

- WAP to check whether the entered character is vowel,consonant, digit or special character.

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

int main() {
    char ch;
    cout << "Enter a character: ";
    cin >> ch;

    if ((ch >= 'a' && ch <= 'z') || (ch >= 'A' && ch <= 'Z')) {
        // Check for vowel
        if (ch=='a' || ch=='e' || ch=='i' || ch=='o' || ch=='u' ||
            ch=='A' || ch=='E' || ch=='I' || ch=='O' || ch=='U') {
            cout << "The character is a Vowel." << endl;
        }
        else {
            cout << "The character is a Consonant." << endl;
        }
    }
    else if (ch >= '0' && ch <= '9') {
        cout << "The character is a Digit." << endl;
    }
    else {
        cout << "The character is a Special Character." << endl;
    }

    return 0;
}
```

```
Enter a character: @
The character is a Special Character.
```

```
Enter a character: 4
The character is a Digit.
```

WAP to calculate the Electricity Bill based on the units used.
Slab rates are given below:

First 100 units - ₹5 per unit
Next 200 units (101–300) - ₹7 per unit
Above 300 units - ₹10 per unit

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

int main() {
    int units;
    float bill = 0;

    cout << "Enter electricity units consumed: ";
    cin >> units;

    if (units <= 100) {
        bill = units * 5;
    }
    else if (units <= 300) {
        bill = (100 * 5) + (units - 100) * 7;
    }
    else {
        bill = (100 * 5) + (200 * 7) + (units - 300) * 10;
    }

    cout << "Total Electricity Bill = Rs " << bill << endl;

    return 0;
}
```

```
Enter electricity units consumed: 500
Total Electricity Bill = Rs 3900
```

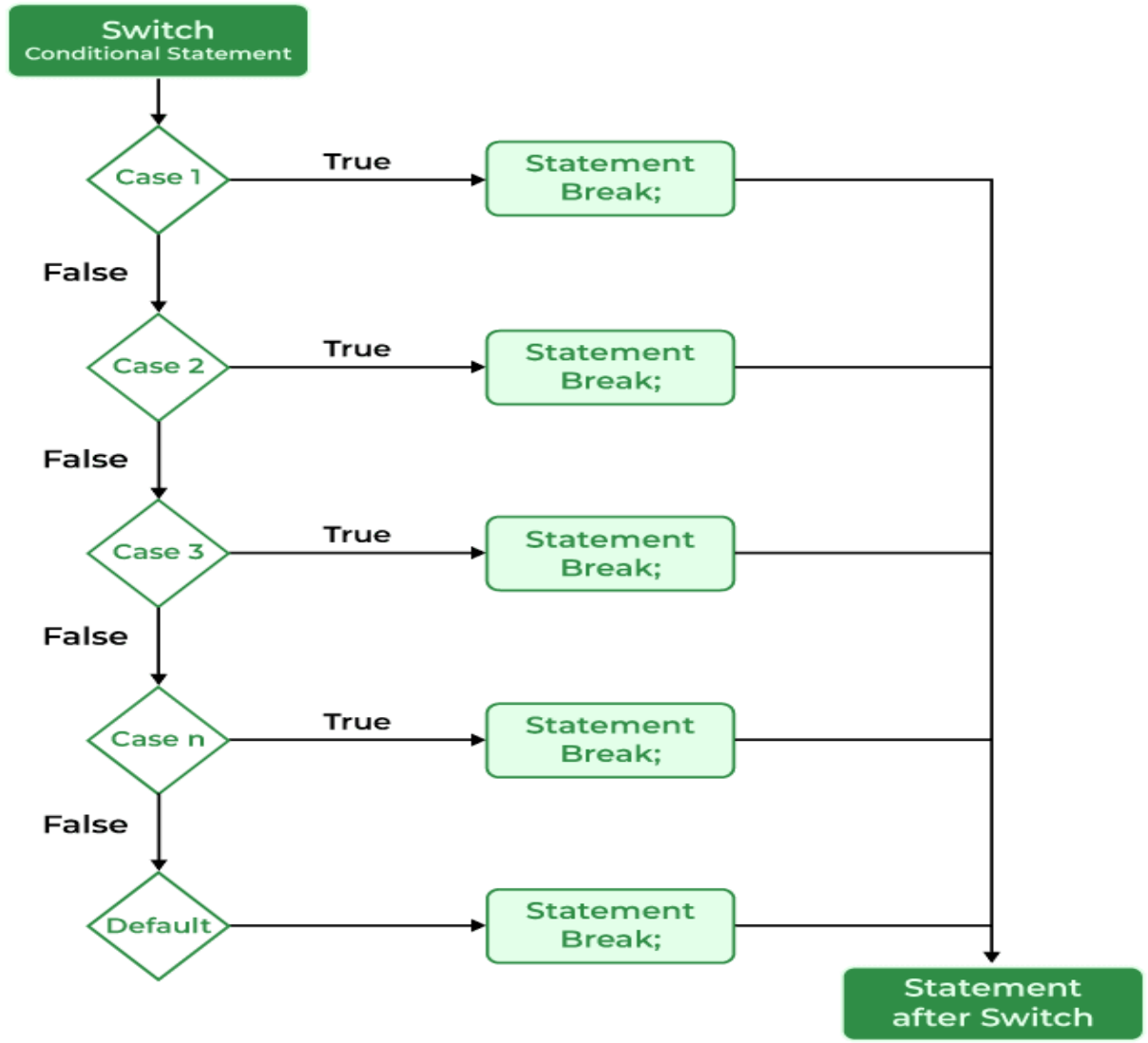
- WAP to check if the year is leap year or not

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

int main() {
    int year;
    cout << "Enter a year: ";
    cin >> year;

    if ((year % 400 == 0) || (year % 4 == 0 && year % 100 != 0)) {
        cout << year << " is a Leap Year." << endl;
    }
    else {
        cout << year << " is NOT a Leap Year." << endl;
    }

    return 0;
}
```

```
Enter a year: 2024
2024 is a Leap Year.
```

Switch Statement

- In C++, the switch statement is used when multiple situations need to be evaluated primarily based on the value of a variable or an expression.
- The switch statement acts as an alternative to multiple if statements or an if-else ladder.
- It provides a cleaner structure compared to multiple conditional statements.
- It is easy to handle multiple conditions using a switch statement.


Switch
Conditional Statement
Case 1
True
Statement
Break;
False
Case 2
True
Statement
Break;
False
Case 3
True
Statement
Break;
False
Case n
True
Statement
Break;
False
Default
Statement
Break;
Statement
after Switch

Syntax

```
switch (expression) {
    case constant1:
        // statements
        break;

    case constant2:
        // statements
        break;

    case constant3:
        // statements
        break;

    // you can add more cases as needed

    default:
        // statements (executes if no case matches)
}
```

- **expression** must be of integral type (int, char etc.).
- Each **case label** must be a **constant value**.
- **break**; is used to exit the switch block (otherwise execution falls through to the next case).
- **default** is optional but useful as a catch-all if no case matches.

```cpp
int main() {
    int num1, num2;
    char op;

    cout << "Enter first number: ";
    cin >> num1;
    cout << "Enter operator (+, -, *, /): ";
    cin >> op;
    cout << "Enter second number: ";
    cin >> num2;

    switch(op) {
        case '+':
            cout << "Result = " << num1 + num2 << endl;
            break;
        case '-':
            cout << "Result = " << num1 - num2 << endl;
            break;
        case '*':
            cout << "Result = " << num1 * num2 << endl;
            break;
        case '/':
            if (num2 != 0)
                cout << "Result = " << num1 / num2 << endl;
            else
                cout << "Error! Division by zero." << endl;
            break;
        default:
            cout << "Invalid operator!" << endl;
    }

    return 0;
}
```

```
Enter first number: 45
Enter operator (+, -, *, /): *
Enter second number: 5
Result = 225
```

- Write a C++ program using switch case to check if a number is even or odd.

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

int main() {
    int n;
    cout << "Enter a number: ";
    cin >> n;

    switch(n % 2) {
        case 0: cout << n << " is Even."; break;
        case 1: cout << n << " is Odd."; break;
    }
    return 0;
}
```

- Write a program in C++ to simulate a simple ATM menu using switch case. The program should allow the user to:
    1. Check balance
    2. Deposit money
    3. Withdraw money
    4. Exit

```cpp
int main() {
    int choice, balance = 5000, amount;

    cout << "ATM Menu:\n1. Check Balance\n2. Deposit\n3. Withdraw\n4. Exit\n";
    cout << "Enter choice: ";
    cin >> choice;

    switch(choice) {
        case 1:
            cout << "Balance: " << balance;
            break;
        case 2:
            cout << "Enter amount to deposit: ";
            cin >> amount;
            balance += amount;
            cout << "Updated Balance: " << balance;
            break;
        case 3:
            cout << "Enter amount to withdraw: ";
            cin >> amount;
            if(amount <= balance) {
                balance -= amount;
                cout << "Remaining Balance: " << balance;
            } else {
                cout << "Insufficient Balance!";
            }
            break;
        case 4:
            cout << "Thank you! Exiting...";
            break;
        default:
            cout << "Invalid Choice!";
    }
    return 0;
}
```

```
ATM Menu:
1. Check Balance
2. Deposit
3. Withdraw
4. Exit
Enter choice: 3
Enter amount to withdraw: 7000
```

- Write a C++ program that takes marks (0–100) as input and prints the grade using switch case:
  - Marks 90–100 → Grade A
  - Marks 80–89 → Grade B
  - Marks 70–79 → Grade C
  - Marks 60–69 → Grade D
  - Marks 50–59 → Grade E
  - Marks below 50 → Fail

```cpp
int main() {
    int choice, balance = 5000, amount;

    cout << "ATM Menu:\n1. Check Balance\n2. Deposit\n3. Withdraw\n4. Exit\n";
    cout << "Enter choice: ";
    cin >> choice;

    switch(choice) {
        case 1:
            cout << "Balance: " << balance;
            break;
        case 2:
            cout << "Enter amount to deposit: ";
            cin >> amount;
            balance += amount;
            cout << "Updated Balance: " << balance;
            break;
        case 3:
            cout << "Enter amount to withdraw: ";
            cin >> amount;
            if(amount <= balance) {
                balance -= amount;
                cout << "Remaining Balance: " << balance;
            } else {
                cout << "Insufficient Balance!";
            }
            break;
        case 4:
            cout << "Thank you! Exiting...";
            break;
        default:
            cout << "Invalid Choice!";
    }
    return 0;
}
```

- Write a C++ program to calculate the area of different shapes (circle, rectangle, triangle, square) using switch case.

```cpp
int main() {
    int choice;
    double area, r, l, b, h;

    cout << "Choose Shape:\n1. Circle\n2. Rectangle\n3. Triangle\n4. Square\n";
    cin >> choice;

    switch(choice) {
        case 1:
            cout << "Enter radius: ";
            cin >> r;
            area = M_PI * r * r;
            cout << "Area of Circle = " << area;
            break;
        case 2:
            cout << "Enter length and breadth: ";
            cin >> l >> b;
            area = l * b;
            cout << "Area of Rectangle = " << area;
            break;
        case 3:
            cout << "Enter base and height: ";
            cin >> b >> h;
            area = 0.5 * b * h;
            cout << "Area of Triangle = " << area;
            break;
        case 4:
            cout << "Enter side: ";
            cin >> l;
            area = l * l;
            cout << "Area of Square = " << area;
            break;
        default:
            cout << "Invalid Choice!";
    }
```

- WAP a program to check whether the entered character is vowel, consonants, digit or special character using **switch statement**

```cpp
int main() {
    char ch;
    cout << "Enter a character: ";
    cin >> ch;

    switch(ch) {
        // Vowels
        case 'a': case 'e': case 'i': case 'o': case 'u':
        case 'A': case 'E': case 'I': case 'O': case 'U':
            cout << "The character is a Vowel." << endl;
            break;

        // Digits
        case '0': case '1': case '2': case '3': case '4':
        case '5': case '6': case '7': case '8': case '9':
            cout << "The character is a Digit." << endl;
            break;

        // Consonants (check using default logic)
        default:
            if ((ch >= 'a' && ch <= 'z') || (ch >= 'A' && ch <= 'Z'))
                cout << "The character is a Consonant." << endl;
            else
                cout << "The character is a Special Character." << endl;
    }

    return 0;
}
```

Quiz

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

int main() {
    int x = 2;
    switch(x) {
        case 1: cout << "One ";
        case 2: cout << "Two ";
        case 3: cout << "Three ";
        default: cout << "Default";
    }
    return 0;
}
```

What is the output?

a) Two

b) Two Three Default

c) One Two Three Default

d) Error

Quiz - Q1

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

int main() {
    int x = 2;
    switch(x) {
        case 1: cout << "One ";
        case 2: cout << "Two ";
        case 3: cout << "Three ";
        default: cout << "Default";
    }
    return 0;
}
```

What is the output?

a) Two

b) Two Three Default

c) One Two Three Default

d) Error

Answer: b) Two Three Default

Q2

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

int main() {
    int n = 5;
    switch(n) {
        case 5: cout << "Five"; break;
        case 5: cout << "Duplicate"; break;
        default: cout << "Default";
    }
}
```

What happens?

a) Output: Five

b) Output: Duplicate

c) Output: Five Duplicate

d) Compilation Error

Q2

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

int main() {
    int n = 5;
    switch(n) {
        case 5: cout << "Five"; break;
        case 5: cout << "Duplicate"; break;
        default: cout << "Default";
    }
}
```

What happens?

a) Output: Five

b) Output: Duplicate

c) Output: Five Duplicate

d) Compilation Error

d) Compilation Error (duplicate case not allowed)

Q3

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

int main() {
    char ch = 'b';
    switch(ch) {
        case 'a': cout << "Apple"; break;
        case 'b': cout << "Ball"; break;
        default: cout << "Other";
    }
}
```

Output:

a) Apple

b) Ball

c) Other

d) Error

Q3

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

int main() {
    char ch = 'b';
    switch(ch) {
        case 'a': cout << "Apple"; break;
        case 'b': cout << "Ball"; break;
        default: cout << "Other";
    }
}
```

Output:

a) Apple

b) Ball

c) Other

d) Error

Answer: b) Ball

Q4

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

int main() {
    int n = 4;
    switch(n) {
        case 1: cout << "One ";
        case 2: cout << "Two ";
        case 3: cout << "Three ";
        case 4: cout << "Four ";
        case 5: cout << "Five ";
    }
}
```

What is the output?

Q5

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

int main() {
    int n = 5;
    switch(n) {
        default: cout << "Default ";
        case 1: cout << "One "; break;
        case 2: cout << "Two "; break;
        case 3: cout << "Three ";
    }
}
```

What is the output?

Q6

```
    char inchar = 'A';
switch (inchar) {
    case 'A':
        printf("choice A\n");
    case 'B':
        printf("choice B");
    case 'C':
    case 'D':
    case 'E':
    default:
        printf("No Choice");
}
```

What is the output?

Options:

a) No choice

b) Choice A

c) Choice A Choice B No choice

d) Program gives no output as it is erroneous

Q6

```
    char inchar = 'A';
switch (inchar) {
    case 'A':
        printf("choice A\n");
    case 'B':
        printf("choice B");
    case 'C':
    case 'D':
    case 'E':
    default:
        printf("No Choice");
}
```

What is the output?

Options:

a) No choice

b) Choice A

c) Choice A Choice B No choice

d) Program gives no output as it is erroneous

Answer: c) Choice A Choice B No choice

Q7

```c
int x = 15, y = 25, z;
z = (x < y) ? x : y;
printf("%d", z);
```

Output?

a) 15

b) 25

c) 0

d) Compilation error

Q8

```c
int a = 0, b;
b = a-- ? a++ : (a ? a++ : a--);
printf("%d", b);
```

Output?

a) 0

b) -1

c) 1

d) Undefined behavior

Module 2.2

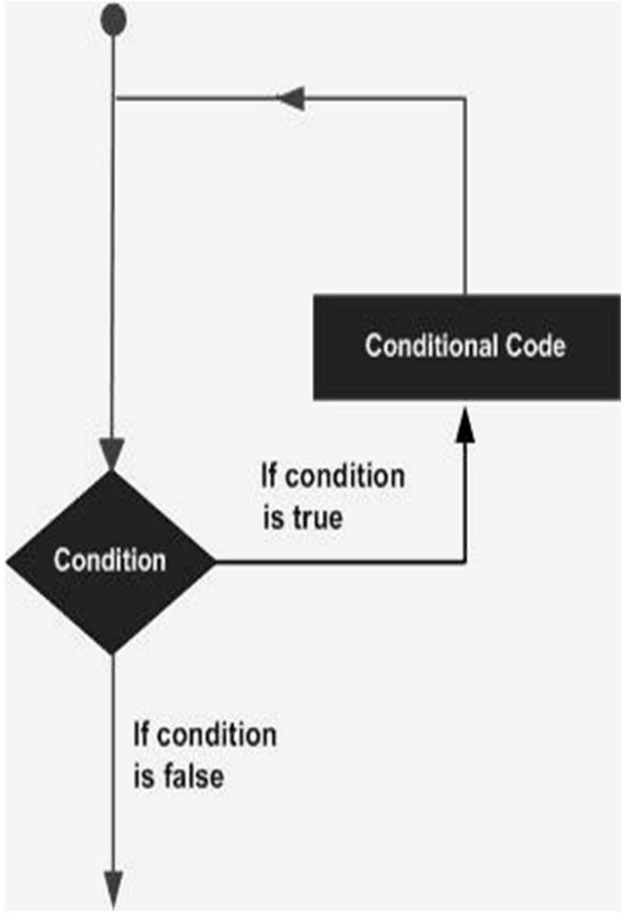
- Looping Control Structures
- Flag Concept
- Counting Loops

Loops in Programming

- Loops (iterative statements) execute a block of code repetitively
- Run as long as a specified condition is true.
- Enable iteration in programming.
- Improve code efficiency and readability.
- Promote reuse of code logic.

PARTS OF A LOOP

- **Initialization Expression(s)** initialize(s) the loop variables in the beginning of the loop.
- **Test Expression** decides whether the loop will be executed (if test expression is true) or not (if test expression is false).
- **Update Expression(s)** update(s) the values of loop variables after every iteration of the loop.
- **The Body-of-the-Loop** contains statements to be executed repeatedly.



TYPES OF LOOPS

C++ programming language provides following types of loop to handle looping requirements:

Loop Type	Description
while loop	Repeats a statement or group of statements until a given condition is true. It tests the condition before executing the loop body.
for loop	Execute a sequence of statements multiple times and abbreviates the code that manages the loop variable.
do...while loop	Like a while statement, except that it tests the condition at the end of the loop body
nested loops	You can use one or more loop inside any another while, for or do..while loop.

Types of Loops in Programming:

- In programming, loops are categorized into two main types based on the control mechanism:
  - **entry-controlled loops (Pre-test loop)**
  - **exit-controlled loops (post –test loop)**

Entry-Controlled loops:

- In Entry controlled loops the test condition is checked before entering the main body of the loop.
- **For Loop** and **While Loop** is Entry-controlled loops

Example

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

int main()
{

    // Entry-controlled for loop
    int i;
    for (i = 0; i < 5; i++) {
        cout << i << " ";
    }
    cout << endl;

    // Entry-controlled while loop
    i = 0;
    while (i < 5) {
        cout << i << " ";
        i++;
    }

    return 0;
}
```

```
0 1 2 3 4
0 1 2 3 4
```

Exit-Controlled loops:

- In Exit controlled loops the test condition is evaluated at the end of the loop body.
- The loop body will execute at least once, irrespective of whether the condition is true or false.
- **Do-while Loop** is an example of Exit Controlled loop.

Example

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

int main()
{
    // Exit controlled do-while loop
    int i = 0;
    do {
        cout << i << " ";
        i++;
    } while (i < 5);
    return 0;
}
```

```
0 1 2 3 4
```

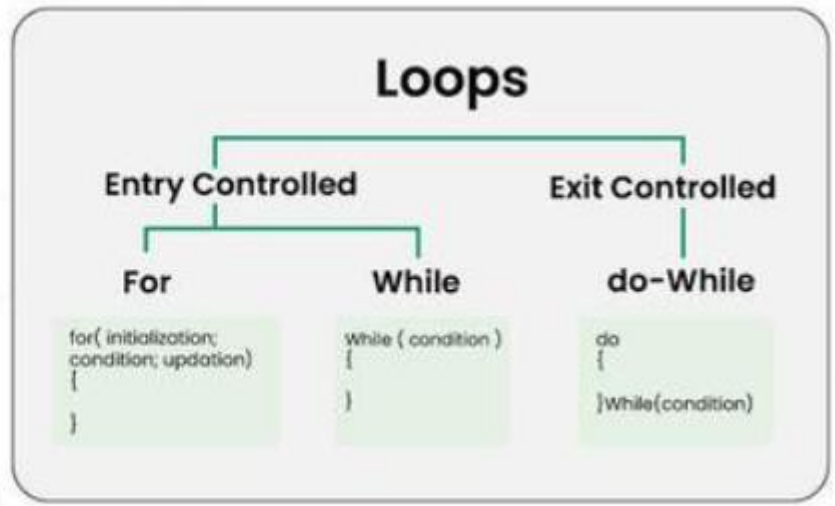
Need for loops

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

int main() {

    cout << "Hello World\n";
    cout << "Hello World\n";
    cout << "Hello World\n";
    cout << "Hello World\n";
    cout << "Hello World";
    return 0;
}
```

```
Hello World
Hello World
Hello World
Hello World
Hello World
```

Loops

in Programming



Types of loop (another characterization)

- Event-controlled loop
- Counter controlled loop

All the loops used in C++ programming are either event or counter-controlled loops.

Counter controlled loop

- The number of iterations is known in advance.
- A loop variable (counter) is incremented/decremented until it reaches a limit.
- for loop is usually counter-controlled.
- Example: Printing numbers from 1 to 10 using a for loop.

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

Event-controlled loop

- The number of iterations is not known beforehand.
- Loop execution depends on an event or condition.
- while / do...while - usually event-controlled.
- Example: Reading input until the user enters -1.

```
int num;
cout << "Enter numbers (-1 to stop): ";
cin >> num;

while (num != -1) {
    cout << "You entered: " << num << endl;
    cin >> num;
}
```

Comparison between entry controlled and exit controlled loop

	Entry-controlled loop	Exit controlled loop
Initialization	Once	Once
Number of tests	N+1	N
Action executed	N	N
Updating executed	N	N
Minimum Iteration	Not even once	At least once

While loop

- While loop is an entry-controlled loop.
- Executes a block of code repeatedly while the condition is true.
- Terminates when the condition becomes false.

Syntax:

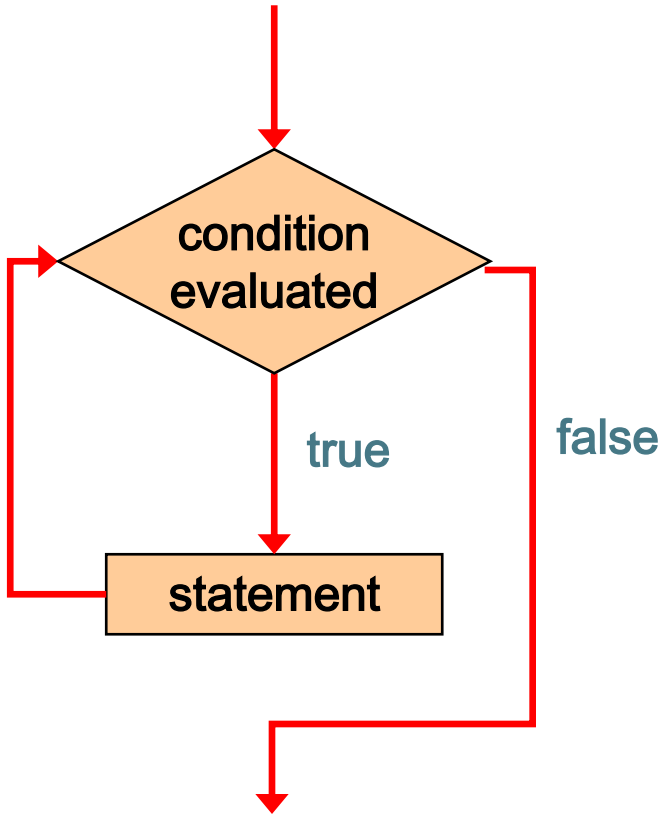
```
while (TestExpression) {
    // Statement(s);

}
```

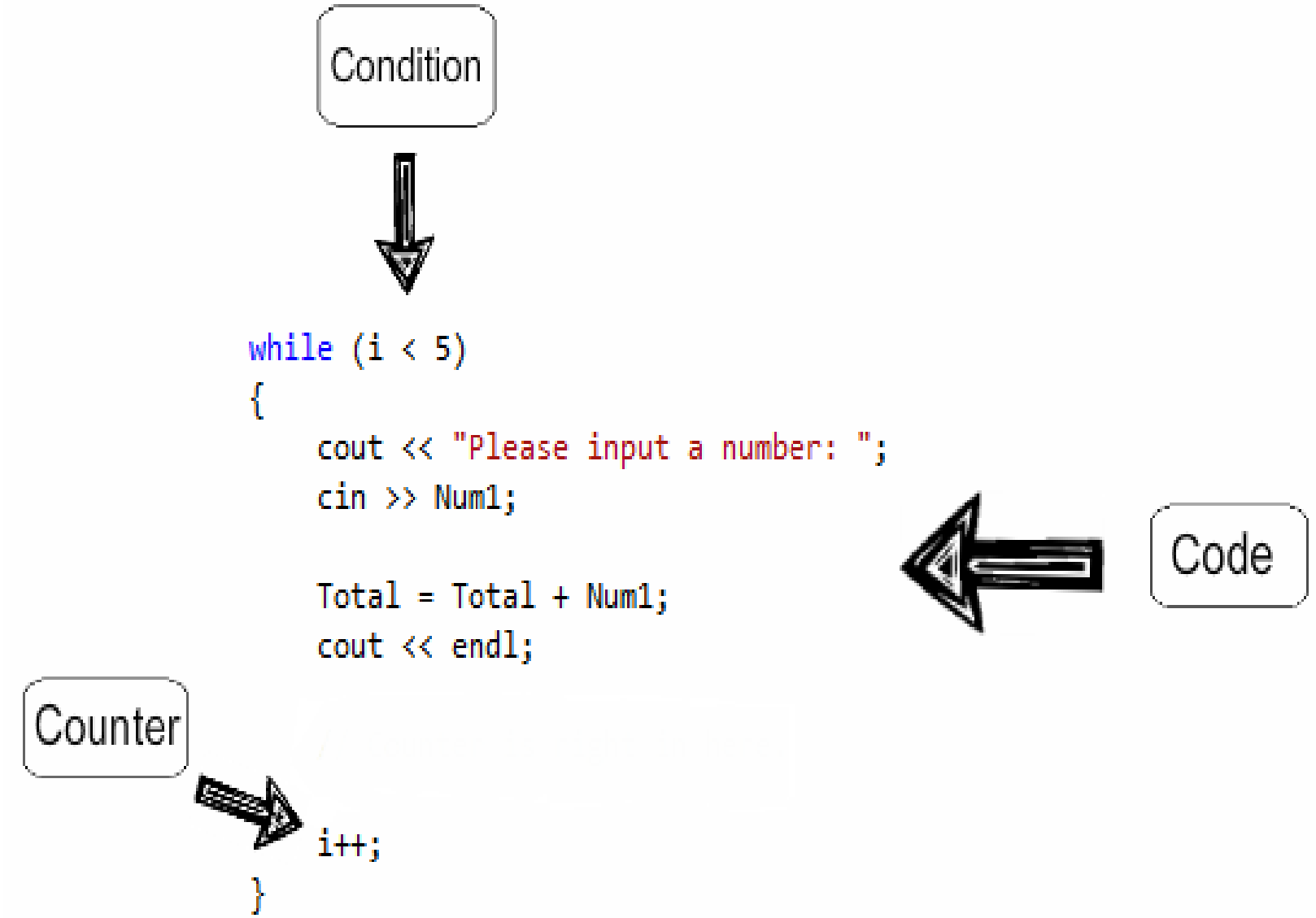
While loop

- statement(s) may be a single statement or a block of statements.
- TestExpression (condition) may be any expression, and true is any nonzero value.
- loop iterates while the condition is true.
- When the condition becomes false, the program control passes to the line immediately following the loop.

- TestExpression should contain the loop control variable
- Intialization of loop control variable should be done before the loop
- Updating it should be within the body of loop.

Flow chart of a while Loop

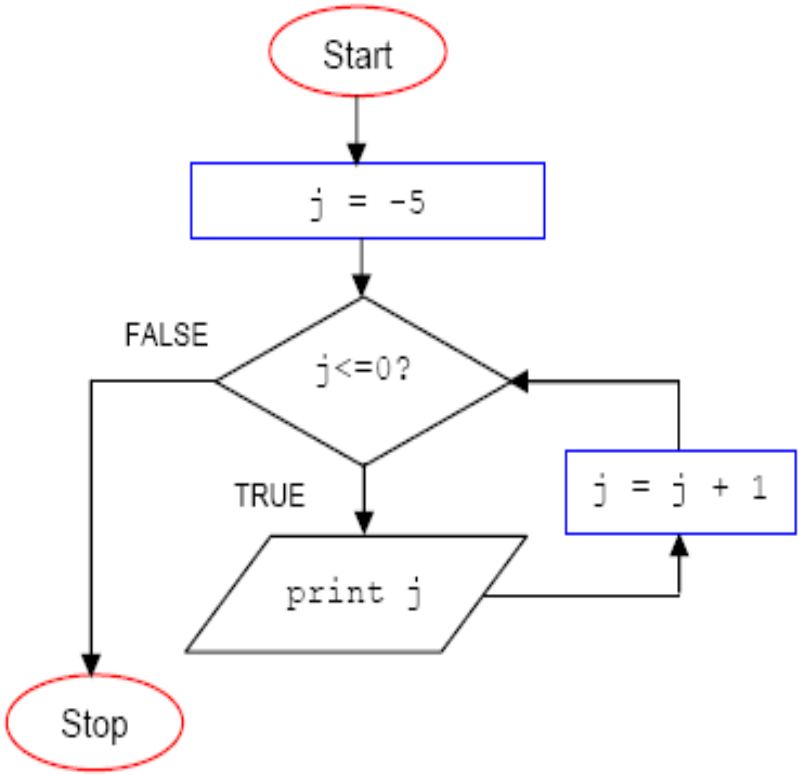



Condition
while (i < 5)
{
cout << "Please input a number: ";
cin >> Num1;
Total = Total + Num1;
cout << endl;
Code
Counter
i++;
}

EXAMPLE:

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

int main()
{
    int j;
    j = -5;

    // while loop
    while (j <= 0)
    {
        cout << j << endl;
        j = j + 1;
    }

    return 0;
}
```

```
-5 -4 -3 -2 -1 0
Press any key to continue . . .
```



- Write a program to print a row of 50 asterisks (*) using while loop

Output:

****

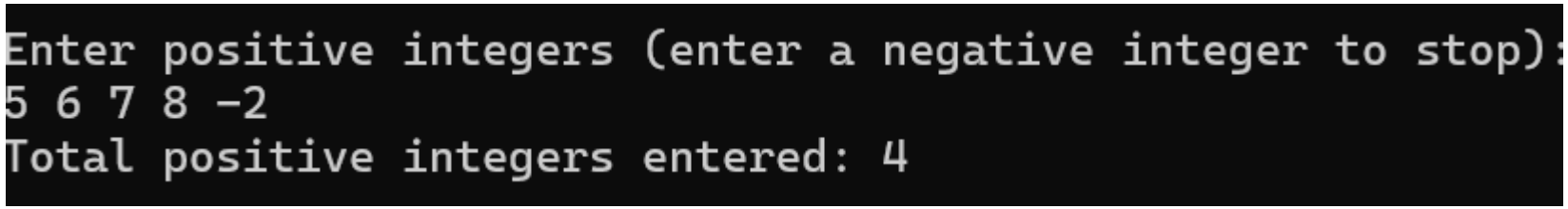
Solution

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

int main() {
    int i = 1;

    while (i <= 50) {
        cout << "*";
        i++;
    }

    cout << endl;   //
    return 0;
}
```

Changes in code

- i=0
- i=50

- Program to count the no of positive integers until negative integer is entered (while loop)



Solution

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

int main() {
    int num, count = 0;

    cout << "Enter positive integers (enter a negative integer to stop): " << endl;
    cin >> num;

    while (num >= 0) {
        count++;
        cin >> num;
    }

    cout << "Total positive integers entered: " << count << endl;
    return 0;
}
```

- WAP that asks the user to accept some numbers and than find their average
- Output:

```
Enter how many numbers you want to average: 5
Enter number 1: 23
Enter number 2: 67
Enter number 3: 34
Enter number 4: 66
Enter number 5: 44
Average = 46.8
```

```cpp
using namespace std;

int main() {
    int n, i = 0;
    int num, sum = 0;
    float avg;

    cout << "Enter how many numbers you want to average: ";
    cin >> n;

    while (i < n) {
        cout << "Enter number " << (i + 1) << ": ";
        cin >> num;
        sum += num;
        i++;
    }

   avg = (float) sum / n;
   cout << "Average = " << avg << endl;

    return 0;
}
```

Infinite loop using while loop

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

int main() {
    int c=5;
    while(c)
    {

        cout<<c<<" ";
        c--;
    }

    return 0;
}
```

What is the output?

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

int main() {
    int c=5;
    while(c)
    {

        cout<<c<<" ";
        c=c-2;
    }

    return 0;
}
```

What is the output?

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

int main() {
    int c=5;
    while(c)
    {

        cout<<c<<" ";
        c=c-2;
    }

    return 0;
}
```

Infinite loop.

After print 1 value of c becomes -1, non zero values are treated as true

Another Infinite loop

```
while (1) {
    // statements to be executed repeatedly
}
```

- While(1) loop iterates forever because the while loop will exit only if the expression 1 becomes 0.
- Only way to exit is through the break statement

Using break statement to exit while(1) loop

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

int main() {
    int num;

    while (1) {    // infinite loop
        cout << "Enter a number (enter -1 to exit): ";
        cin >> num;

        if (num == -1) {
            cout << "Exiting loop..." << endl;
            break;    // exits the loop when -1 is entered
        }

        cout << "You entered: " << num << endl;
    }

    return 0;
}
```

```
Enter a number (enter -1 to exit): 2
You entered: 2
Enter a number (enter -1 to exit): 3
You entered: 3
Enter a number (enter -1 to exit): 6
You entered: 6
Enter a number (enter -1 to exit): 7
You entered: 7
Enter a number (enter -1 to exit): -1
Exiting loop...
```

Quiz- **Q1**

What happens if you execute this pseudocode?

```
while (1)
    cout<< "Hello“;
```

A) Syntax error

B) "Hello" is printed zero times

C) "Hello" is printed countless times continuously

D) "Hello" is printed only once

Q2

```
int i = 0;
while (i < 3)
    i++;
printf("In while loop\n");
```

How many times is the condition i < 3 evaluated?

A) 4

B) 3

C) 2

D) 1

Q3

```c
int i = 8;
while (i = 8) {
    printf("Getting out");
    i++;
}
```

What is the output?

A) "Getting out" is printed infinite times

B) Nothing is printed

C) "Getting out" is printed once

D) "Getting out" is printed 7 times

Q4

What is the output?

```cpp
#include <iostream>
using namespace std;
int main() {
    int i = 1;
    while (i <= 3) {
        cout << i << " ";
        // Forgot to update i
    }
    return 0;
}
```

Q5

```cpp
#include <iostream>
using namespace std;
int main() {
    int x = 0;
    while (x = 5) {
        cout << x << " ";
        x++;
    }
    return 0;
}
```

Q6

```cpp
#include <iostream>
using namespace std;
int main() {
    int i = 1;
    while (i <= 10) {
        cout << i << " ";
        i = i - 1;
    }
    return 0;
}
```

- WAP to find the sum of digits of the number(using while loop)

- Output:

```
Enter a number: 5678
Sum of digits = 26
```

Solution

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

int main() {
    int num, sum = 0;

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

    while (num > 0) {
        int digit = num % 10;    // extract last digit
        sum += digit;            // add digit to sum
        num = num / 10;          // remove last digit
    }

    cout << "Sum of digits = " << sum << endl;
    return 0;
}
```

- WAP reverse the digits of the number

```
Enter a number: 12345
Reversed number = 54321
```

Solution

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

int main() {
    int num, reverse = 0;

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

    while (num > 0) {
        int digit = num % 10;            // extract last digit
        reverse = reverse * 10 + digit; // build reversed number
        num = num / 10;                  // remove last digit
    }

    cout << "Reversed number = " << reverse << endl;
    return 0;
}
```

For loop

- For loop is a control flow structure for iteration.
- Iterates over sequences (numbers, lists, strings, etc.).
- Entry-controlled loop – checks condition before execution.
- Determines number of iterations in advance.

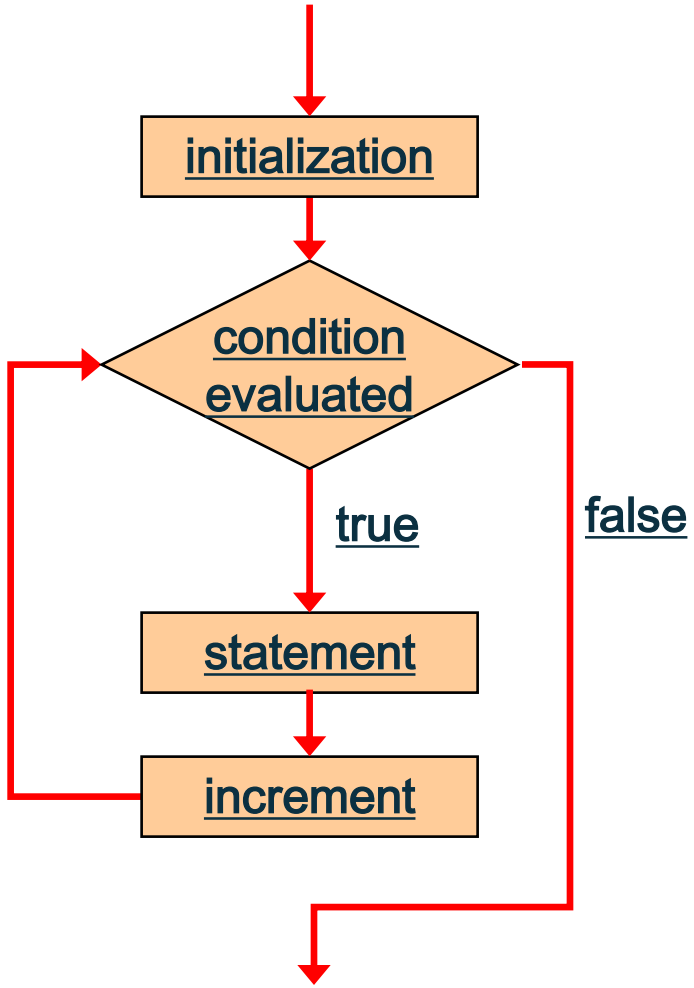
Synatax:

```
for (initialization; condition; updation) {
    // body of for loop
}
```

Parts of for Loop

- **Initialization –** The statements here are executed only once. Involves setting the loop control variable to a starting value.
- **Test Condition/Test Expression–** Evaluates a condition; if true, the loop body executes, if false, the loop ends.
- **Update Expression –** Stmts here executed every time through the loop before the test condition is checked. Involves a loop control variable that is usually incremented or decremented.

Flowchart of a for loop



How many times this for loop executes

```
for(i = m; i <= n; i++) {
// body
}
```

Starts with m, m+1, … , n

How many times this for loop executes

```
for(i = m; i <= n; i++) {
// body
}
```

Starts with m, m+1, … , n

n-m+1 times

How many times this for loop executes

```
for(i = m; i < n; i++) {
// body
}
```

Starts with m, m+1, … , n-1

n-m times

How many times this for loop executes

```
for(i = m; i < n; i+=x) {
// body
}
```

Starts with m, m+1, … , n-1

n-m times

Program to calculate the factorial of the number

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

int main() {
    int n;
     int fact = 1;
    cout << "Enter a number: ";
    cin >> n;

        for (int i = 1; i <= n; i++) {
             fact *= i;    // multiply fact by i
         }
         cout << "Factorial of " << n << " = " << fact << endl;

    return 0;
}
```

```
Enter a number: 5
Factorial of 5 = 120
```

Other code: Computing factorial

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

int main() {
    int n;
     int fact = 1,m;
    cout << "Enter a number: ";
    cin >> n;
    m=n;
       for (; n>0; n--) {
            fact *= n;    // multiply fact by i
        }
        cout << "Factorial of " << m << " = " << fact << endl;

    return 0;
}
```

Effect of Semi-colons after for statement

```
int main() {
    int n;
     int fact = 1,m;
    cout << "Enter a number: ";
    cin >> n;
    m=n;
       for (; n>0; n--); {
            fact *= n;    // multiply fact by i
        }
        cout << "Factorial of " << m << " = " << fact << endl;

    return 0;
}
```

Output will be : fact=0

Some variations of for loop

- All the 3 expressions (initialization, test expression, updating) need for be present in **for** loop but semicolon need to be present.

Program to find the sum of series of upto ‘n’ terms (Program 1)

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

int main() {
    int n, sum = 0;

    cout << "Enter number of terms: ";
    cin >> n;

    for (int i = 1; i <= n; i++) {
        sum += i;    // add each term
    }

    cout << "Sum of series 1 + 2 + ... + " << n << " = " << sum << endl;
    return 0;
}
```

```
Enter number of terms: 5
Sum of series 1 + 2 + ... + 5 = 15
```

Program to find the sum of series of upto ‘n’ terms (Program 2)

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

int main() {
    int n, sum, i=1 ;

    cout << "Enter number of terms: ";
    cin >> n;

    for (sum=0; i <= n; i++) {
        sum += i;    // add each term
    }

    cout << "Sum of series 1 + 2 + ... + " << n << " = " << sum << endl;
    return 0;
}
```

```
Enter number of terms: 5
Sum of series 1 + 2 + ... + 5 = 15
```

Program to find the sum of series of upto ‘n’ terms (Program 3)

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

int main() {
    int n, sum, i;

    cout << "Enter number of terms: ";
    cin >> n;

    for ( i=1,sum=0; i <= n; i++) {
        sum += i;    // add each term
    }

    cout << "Sum of series 1 + 2 + ... + " << n << " = " << sum << endl;
    return 0;
}
```

```
Enter number of terms: 5
Sum of series 1 + 2 + ... + 5 = 15
```

Program to find the sum of series of upto ‘n’ terms (Program 4)

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

int main() {
    int n, sum=0, i=1;

    cout << "Enter number of terms: ";
    cin >> n;

    for ( ; i <= n; i++) {
        sum += i;    // add each term
    }

    cout << "Sum of series 1 + 2 + ... + " << n << " = " << sum << endl;
    return 0;
}
```

Empty initialization statment. Semicolon has to be specified

Multiple condition in test expression

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

int main() {
    int n, sum = 0;

    cout << "Enter the value of n: ";
    cin >> n;

    // multiple conditions in test expression
    for (int i = 1; i <= n && i % 2 == 0 || i <= n && i % 2 != 0; i++) {
        if (i % 2 == 0)    // ensure only even numbers are added
            sum += i;
    }

    cout << "Sum of even numbers between 1 and " << n << " = " << sum << endl;
    return 0;
}
```

```
Enter the value of n: 5
Sum of even numbers between 1 and 5 = 6
```

Third expression of for statement

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

int main() {
    int num, sum = 0;

    cout << "Enter a number: ";
    cin >> num;
    // for loop: keep extracting digits until n
    for ( ; num > 0; num = num / 10) {
        int digit = num % 10;  // extract last
        sum += digit;          // add digit to
    }

    cout << "Sum of digits = " << sum << endl;
    return 0;
}
```

```
Enter a number: 1234
Sum of digits = 10
```

Empty third expression

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

int main() {
    int num, sum = 0;

    cout << "Enter a number: ";
    cin >> num;
    // for loop: keep extracting digits until number becomes 0
    for ( ; num > 0; ) {
        int digit = num % 10;  // extract last digit
        sum += digit;
        num/=10;            // add digit to sum
    }

    cout << "Sum of digits = " << sum << endl;
    return 0;
}
```

Multiple statements in third expression

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

int main() {
    int num, sum = 0,digit;

    cout << "Enter a number: ";
    cin >> num;
    // for loop: keep extracting digits until num
    for ( ; num > 0; sum+=digit, num/=10) {
        digit = num % 10;  // extract last digit

    }

    cout << "Sum of digits = " << sum << endl;
    return 0;
}
```

Comma operator associates from left to right

```
for(s=0,i=1;i<=n;++i)
    s+=i
```

Can be written as

```
for(s=0,i=1;i<=n;s+=i,++i);
```

But not as

```
for(s=0,i=1;i<=n;++i,s+=i);
```

Pre-increment and post-increment same effect

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

int main() {
    cout << "Using post-increment (i++):" << endl;
    for (int i = 1; i <= 5; i++) {
        cout << i << " ";
    }

    cout << "\n\nUsing pre-increment (++i):" << endl;
    for (int i = 1; i <= 5; ++i) {
        cout << i << " ";
    }

    return 0;
}
```

- WAP to find the sum of n integers using for loop

```
Enter how many numbers you want to add: 5
Enter 5 integers:
23
56
34
78
-90
Sum of 5 numbers = 101
```

Solution

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

int main() {
    int n, num, sum = 0;

    cout << "Enter how many numbers you want to add: ";
    cin >> n;

    cout << "Enter " << n << " integers: " << endl;
    for (int i = 0; i < n; i++) {
        cin >> num;
        sum += num;
    }

    cout << "Sum of " << n << " numbers = " << sum << endl;
    return 0;
}
```

Program to generate Multiplication table

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

int main() {
    int num;
    cout << "Enter a number: ";
    cin >> num;

    cout << "Multiplication Table of " << num << ":\n";
    for (int i = 1; i <= 10; i++) {
        cout << num << " x " << i << " = " << num * i << endl;
    }

    return 0;
}
```

```
Enter a number: 5
Multiplication Table of 5:
5 x 1 = 5
5 x 2 = 10
5 x 3 = 15
5 x 4 = 20
5 x 5 = 25
5 x 6 = 30
5 x 7 = 35
5 x 8 = 40
5 x 9 = 45
5 x 10 = 50
```

Quiz- What is the output?

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

int main() {

    for (int i = 0; i++; i++) {
        cout << i << " ";
    }

    return 0;
}
```

What is the output?

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

int main() {

    for (int i = 0; ++i; i++) {
        cout << i << " ";
    }

    return 0;
}
```

What is the output?

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

int main() {

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

    return 0;
}
```

Quiz- What is the output?

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

int main() {

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

    return 0;
}
```

Quiz- What is the output?

What would be the output from the given program?

```
int main()
{
  int i=9;
  for(i--; i--; i--)
    printf("%d", i);
    return 0;
}
```

Answer

What would be the output from the given program?

```
int main()
{
  int i=9;
  for(i--; i--; i--)
    printf("%d", i);
    return 0;
}
```

Output: 7 5 3 1

Output?

What would be the output from the given program?

```
int main()
{
  int i;
    for(i=5; ++i; i-=3)
      printf("%d", i);
    return 0;
}
```

Answer

What would be the output from the given program?

```
int main()
{
  int i;
    for(i=5; ++i; i-=3)
      printf("%d", i);
    return 0;
}
```

Output: 6 4 2

What output is obtained from the given program

```
int main()
{
  int i=3;
  for(i--; i<7; i=7)
      printf("%d",i++);
      return 0;
  }
```

What output is obtained from the given program?

```
int main()
{
  int i=3;
  for(i--; i<7; i=7)
      printf("%d",i++);
      return 0;
  }
```

Output: 2

Quiz- What is the output?

```
int i;
for(i = 0; i < 5; i++);
cout << i;
```

Options:

a) 4

b) 5

c) 0

d) Infinite loop

Quiz- What is the output?

```
#include <iostream>
using namespace std;
int main() {
    int i;
    for(i = 0; i < 10; i += 2)
        cout << i << " ";
    return 0;
}
```

Options:

a) 0 1 2 3 4 5 6 7 8 9

b) 0 2 4 6 8

c) 2 4 6 8 10

d) Infinite loop

Quiz- What is the output?

- What is the output?

```
#include <iostream>
using namespace std;
int main() {
    int i = 0, j = 10;
    for(; i < j; i++, j--) {
        cout << i + j << " ";
    }
    return 0;
}
```

Options:

a) 10 10 10 10 10

b) 10 10 10 10 11

c) 10 10 10 10

d) None of the above

Quiz

Which of the following is valid in C++ for writing an infinite loop?

Options:

a) for(; ;) { }
b) for(0; 1; 0) { }
c) for(int i=0; i>=0; i++) { }
d) All of the above

Quiz- What is the output?

```cpp
#include <iostream>
using namespace std;
int main() {
    for (int i = 0; i < 3; i++) {
        cout << i << " ";
    }
    cout << i;
    return 0;
}
```

Options:

a) 0 1 2 3

b) 0 1 2

c) Error: ‘i’ was not declared in this scope

d) None of the above

- WAP to display the quotient of the number without using operator ‘/’

Hint: Use repeated subtraction

Do-while loop

- **Exit-controlled** loop (condition is checked after execution).
- Ensures the loop body executes at least once.
- Syntax:

```
do {
   // Statements
   // Update expression
} while (TestExpression);
```

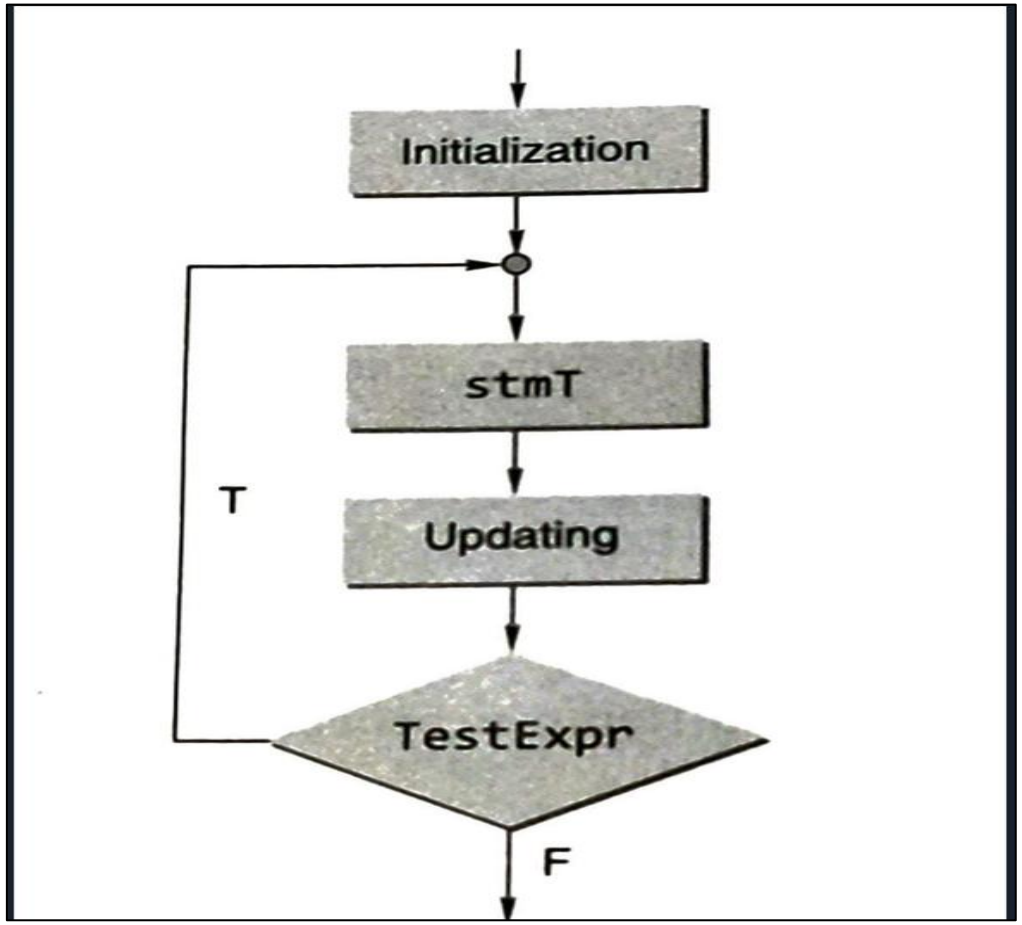
- Continues execution as long as the TestExpression is true.
- Different from while loop - do-while guarantees one execution even if TestExpression is false.

Do while loop

- **Test Expression:** Condition Checked **after** the loop body.
- **Update Expression:** Modifies loop variable towards termination.
- **Statments:** Executes **at least once**, then repeats while TestExprepssion is true.

- Commonly used when:
  - You want code to run once before checking condition.
  - Input-validation loops (e.g., menu-driven programs).

Flow chart of Do while loop



Print numbers from 1 to n

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

int main() {
    int n, i = 1;
    cout << "Enter a number: ";
    cin >> n;

    do {
        cout << i << " ";
        i++;
    } while (i <= n);

    return 0;
}
```

```
Enter a number: 5
1 2 3 4 5
```

Menu driven calculator

```cpp
int main() {
    int choice;
    double a, b;

    do {
        cout << "\n--- Calculator Menu ---\n";
        cout << "1. Add\n2. Subtract\n3. Multiply\n4. Divide\n5. Exit\n";
        cout << "Enter your choice: ";
        cin >> choice;

        if (choice >= 1 && choice <= 4) {
            cout << "Enter two numbers: ";
            cin >> a >> b;
        }

        switch (choice) {
            case 1: cout << "Result = " << a + b << endl; break;
            case 2: cout << "Result = " << a - b << endl; break;
            case 3: cout << "Result = " << a * b << endl; break;
            case 4:
                if (b != 0) cout << "Result = " << a / b << endl;
                else cout << "Error! Division by zero." << endl;
                break;
            case 5: cout << "Exiting...\n"; break;
            default: cout << "Invalid choice!\n";
        }
    } while (choice != 5);

    return 0;
}
```

```
--- Calculator Menu ---
1. Add
2. Subtract
3. Multiply
4. Divide
5. Exit
Enter your choice: 1
Enter two numbers: 25 67
Result = 92

--- Calculator Menu ---
1. Add
2. Subtract
3. Multiply
4. Divide
5. Exit
Enter your choice:
```

WAP Sum of digits of a number using do while loop

Guess the output

1. How many times will the following loop repeat i.e how many x's are printed?

```
int main() {
    int i=5;

    while (i-- > 0) cout<<"x";
    return 0;
}
```

Guess the output

```c
int main() {
    int i = 5;
    while (i-- > 0) {
        printf("%d ", i);
    }
    return 0;
}
```

Guess the output

```
int i = 0;
do {
    cout << i << " ";
    i++;
} while (i < 0);
```

Guess the output

```
int i = 0;
while (i++ < 5)
    cout << i << " ";
```

- Write a C++ program that displays the n terms of the square natural numbers and their sum using loops.
  1, 4, 9, 16 … n Terms

Break statement

Purpose:

The break statement is used to terminate the innermost enclosing loop or switch statement immediately. Control then passes to the statement following the terminated loop or switch.

Usage:

It is typically used within a conditional statement (if) inside a loop to exit the loop prematurely when a certain condition is met.

Break statement example

```
for (int i = 0; i < 10; i++) {
  if (i == 4) {
    break;
  }
  cout << i << "\n";
}
```

Output

```
0
1
2
3
```

Flag concept

- A flag is a variable used as an indicator (True/False).
- Commonly implemented using data type **bool** .
- Signals whether a specific condition/state is present or absent.
  - true - condition met / state active.
  - false - condition not met / state inactive

  Eg: IsPrime=false
- Helps in guiding program flow and decision-making.

WAP to check whether the number is prime no

A **prime number** is a **natural number greater than 1** that has **exactly two factors**:

- **1**
- **Itself**

2 - divisible only by 1 and 2
3 - divisible only by 1 and 3
5 - divisible only by 1 and 5
7 - divisible only by 1 and 7

11, 13, 17, 19,.....
2 is only even prime no.

Program

```cpp
int main() {
    int n;
    bool isPrime = true;

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

    if (n <= 1) {
        isPrime = false;    // 0, 1, and negatives are not prime
    } else {
        for (int i = 2; i < n; i++) {    // check divisors up to n-1
            if (n % i == 0) {
                isPrime = false;
                break;    // no need to check further
            }
        }
    }

    if (isPrime)
        cout << n << " is a Prime number." << endl;
    else
        cout << n << " is NOT a Prime number." << endl;

    return 0;
}
```

```
Enter a number: 45
45 is NOT a Prime number.
```

```
Enter a number: 29
29 is a Prime number.
```

WAP to check whether a number is power of 2

```
Enter a number: 18
It is NOT a power of 2.
```

```
Enter a number: 32
It is a power of 2.
```

Power of 2 = 2, 4, 8, 16, 32, 64 , 128, 256, 512, ...

Example:

- **Input = 16**
- 16 % 2 == 0 – quotient -> 8
- 8 % 2 == 0 - quotient → 4
- 4 % 2 == 0 - quotient → 2
- 2 % 2 == 0 - quotient → 1
- Loop ends at 1 → **Power of 2**

Example:

Input = 18

- 18 % 2 == 0 - quotient → 9
- 9 % 2 != 0 - **NOT a power of 2**

Program

```
int n;
cout << "Enter a number: ";
cin >> n;

bool isPower = true;

        while (n > 1) {
         if (n % 2 != 0) {    // not divisible by
              isPower = false;
              break;
         }
         n /= 2;   // keep dividing by 2
     }
if (isPower)
     cout << "It is a power of 2." << endl;
else
     cout << "It is NOT a power of 2." << endl;
```

Continue Statement

- The continue statement breaks one iteration (in the loop), if a specified condition occurs, and continues with the next iteration in the loop.

```
for (int i = 0; i < 10; i++) {
  if (i == 4) {
    continue;
  }
  cout << i << "\n";
}
```

Output

```
0
1
2
3
5
6
7
8
9
```

Guess the Output?

```
int i;
cout << "The loop with break produces output as: \n";

for (i = 1; i <= 5; i++) {

    // Program comes out of loop when
    // i becomes multiple of 3.
    if ((i % 3) == 0)
        break;
    else
        cout << i << " ";
}

cout << "\nThe loop with continue produces output as: \n";
for (i = 1; i <= 5; i++) {

    // The loop prints all values except
    // those that are multiple of 3.
    if ((i % 3) == 0)
        continue;
    cout << i << " ";
}
```

Guess the Output?

```
int main() {
    int sum = 0;
    for (int i = 1; i <= 5; i++) {
        if (i % 2 == 0)
            continue;
        if (i == 5)
            break;
        sum += i;
    }
    cout << sum;
    return 0;
}
```

Fibonacci numbers

- **Definition**: A sequence where each term is the sum of the two preceding ones.

$$F(n)=F(n-1)+F(n-2)$$

Starting Values:

$$F(0) = 0$$
$$F(1) = 1$$

- **Sequence**:
  0, 1, 1, 2, 3, 5, 8, 13, 21, …

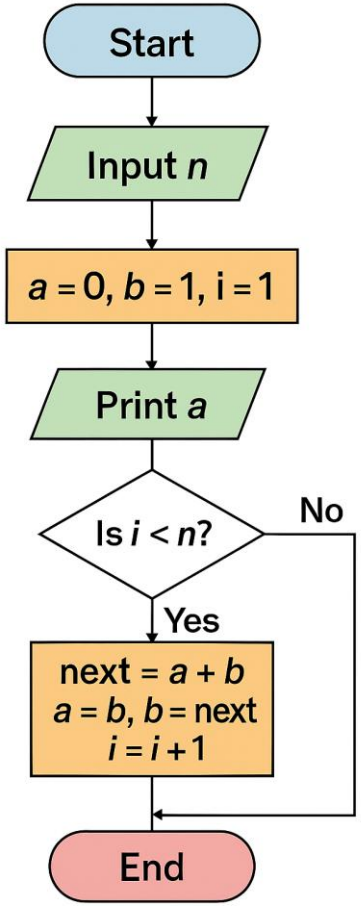
WAP to print all Fibonacci numbers up to a given number n

Program

```
int n;
cout << "Enter the limit : ";
cin >> n;

int a = 0, b = 1;

cout << "Fibonacci numbers up to " << n << ": ";

if (n >= 0) cout << a << " ";    // printing  0
if (n >= 1) cout << b << " ";    // printing  1

int c = a + b;
while (c <= n) {
    cout << c << " ";
    a = b;
    b = c;
    c = a + b;
}
```

```
Enter the limit : 25
Fibonacci numbers up to 25: 0 1 1 2 3 5 8 13 21
```

Flow chart


Start
Input n
a = 0, b = 1, i = 1
Print a
Is i < n?
No
Yes
next = a + b
a = b, b = next
i = i + 1
End

Armstrong Number

- A number is called an **Armstrong number** if the **sum of its digits each raised to the power of the number of digits** is equal to the number itself.

- Example:
- **3-digit Armstrong number**
- $153 \rightarrow 1^3 + 5^3 + 3^3 = 1 + 125 + 27 = 153$
- $370 \rightarrow 3^3 + 7^3 + 0^3 = 27 + 343 + 0 = 370$
- $371 \rightarrow 3^3 + 7^3 + 1^3 = 27 + 343 + 1 = 371$

Use pow function

- pow() is a mathematical function in the <cmath> library.
- It is used to calculate power of number:
- pow(base, exponent)=base $^{\text{exponent}}$

Syntax:

double pow(double base, double exponent);

Nested Loops

- A nested loop is a loop inside another loop.
- The inner loop runs completely for each iteration of the outer loop.
- Commonly used for:
  - Patterns (stars, numbers)
  - Matrices and 2D arrays
  - Complex iterations

Example Nested loop

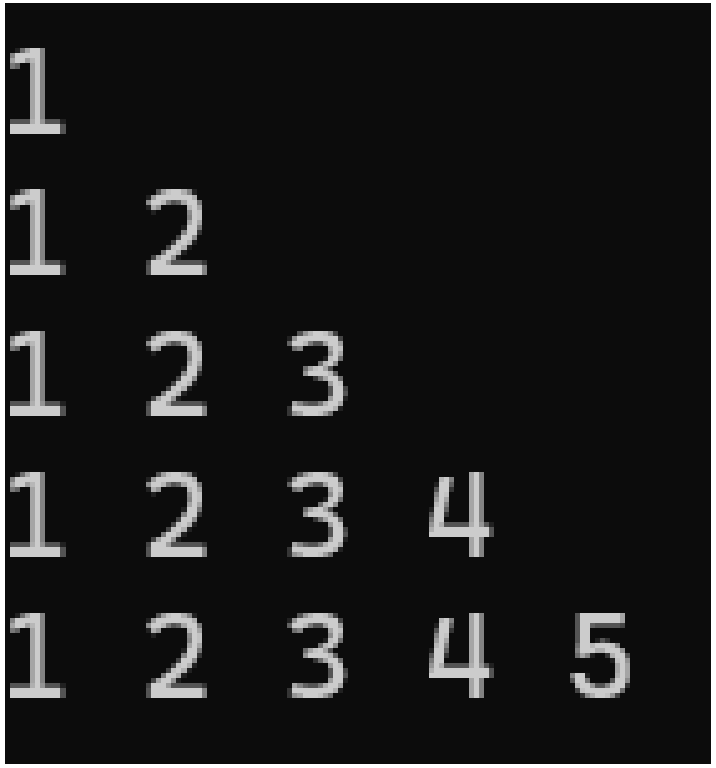
```
for (int i = 1; i <= 3; i++)
{
    for (int j = 1; j <= 3; j++)
    {
        cout << i * j << " ";
    }
    cout << endl; // new line after each row
}
```

```
1 2 3
2 4 6
3 6 9
```

Example Nested Loop

```
int main()
{
    int n = 5;
    for (int i = 1; i <= n; i++)
    {
        for (int j = 1; j <= i; j++)
        {
            cout << "* ";
        }
        cout << endl;
    }
return 0;
}
```

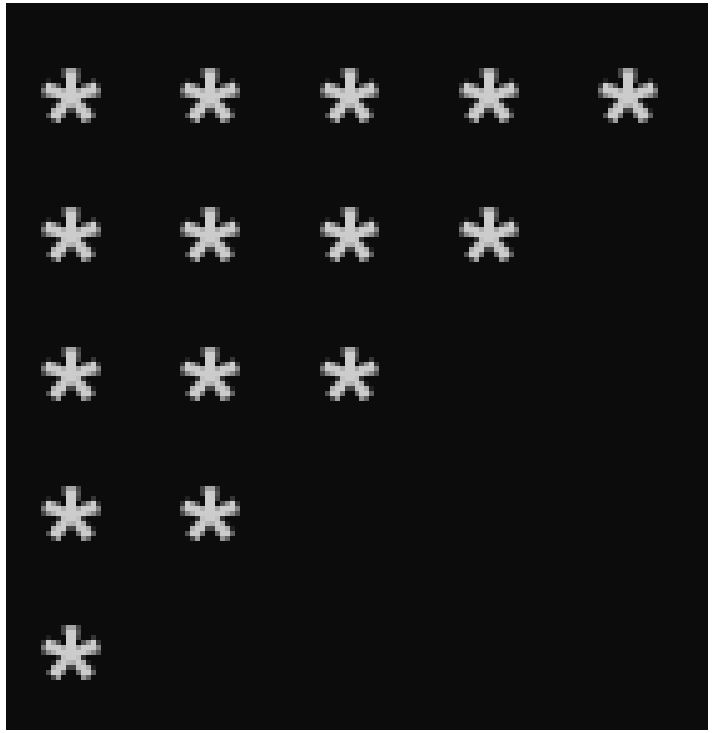
Generate the following Number Pyramid pattern



Solution:

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

int main() {
    int n = 5;

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

    return 0;
}
```

Generate the following Inverted Right Traingle Pattern



Solution:

```
int n = 5;

for (int i = n; i >= 1; i--) {
    for (int j = 1; j <= i; j++) {
        cout << "* ";
    }
    cout << endl;
}
```

Exercise: WAP

- To find the GCD of 2 numbers
- To find LCM of 2 numbers
- To display prime numbers in the given range
- To display prime factors of the given number
- Write a program in C++ to display the **n terms of the geometric series**:

$$1 + \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \ldots \text{ up to n terms}$$

and also compute their **sum**.

Geometric series solution

```cpp
int main() {
    int n;
    double term = 1.0, sum = 0.0;

    cout << "Enter the number of terms: ";
    cin >> n;

    cout << "Geometric series up to " << n << " terms:" << endl;

    for (int i = 1; i <= n; i++) {
        cout <<  term << " ";
        sum += term;
        term = term / 2;  // next term is half of previous
    }

    cout << "\nSum of series = " << sum << endl;

    return 0;
}
```

```
Enter the number of terms: 5
Geometric series up to 5 terms:
1 0.5 0.25 0.125 0.0625
Sum of series = 1.9375
```

- Write a menu-driven program in C++ for the following options:
  a. Print the multiplication table of a given number using a for loop.
  b. Calculate the sum of digits of a number using a do...while loop.

Solution:

```
int choice;

do {
    cout << "\n--- Menu ---\n";
    cout << "1. Print multiplication table (for loop)\n";
    cout << "2. Sum of digits (do...while loop)\n";
    cout << "3. Exit\n";
    cout << "Enter your choice: ";
    cin >> choice;

    switch (choice) {
        case 1:
            int num;
            cout << "Enter a number: ";
            cin >> num;
            cout << "Multiplication Table of " << num << ":\n";
            for (int i = 1; i <= 10; i++) {
                cout << num << " x " << i << " = " << num * i << endl;
            }
            break;
```

```
        case 2:{
            int n, sum = 0, digit;
            cout << "Enter a number: ";
            cin >> n;
            do {
                digit = n % 10;
                sum += digit;
                num /= 10;
            } while (n > 0);

            cout << "Sum of digits = " << sum << endl;
            break;
    }
        case 3:
            cout << "Exiting program..." << endl;
            break;

        default:
            cout << "Invalid choice! Try again." << endl;
    }
} while (choice != 3);

return 0;
```