Control Statements & Decision-Making In C

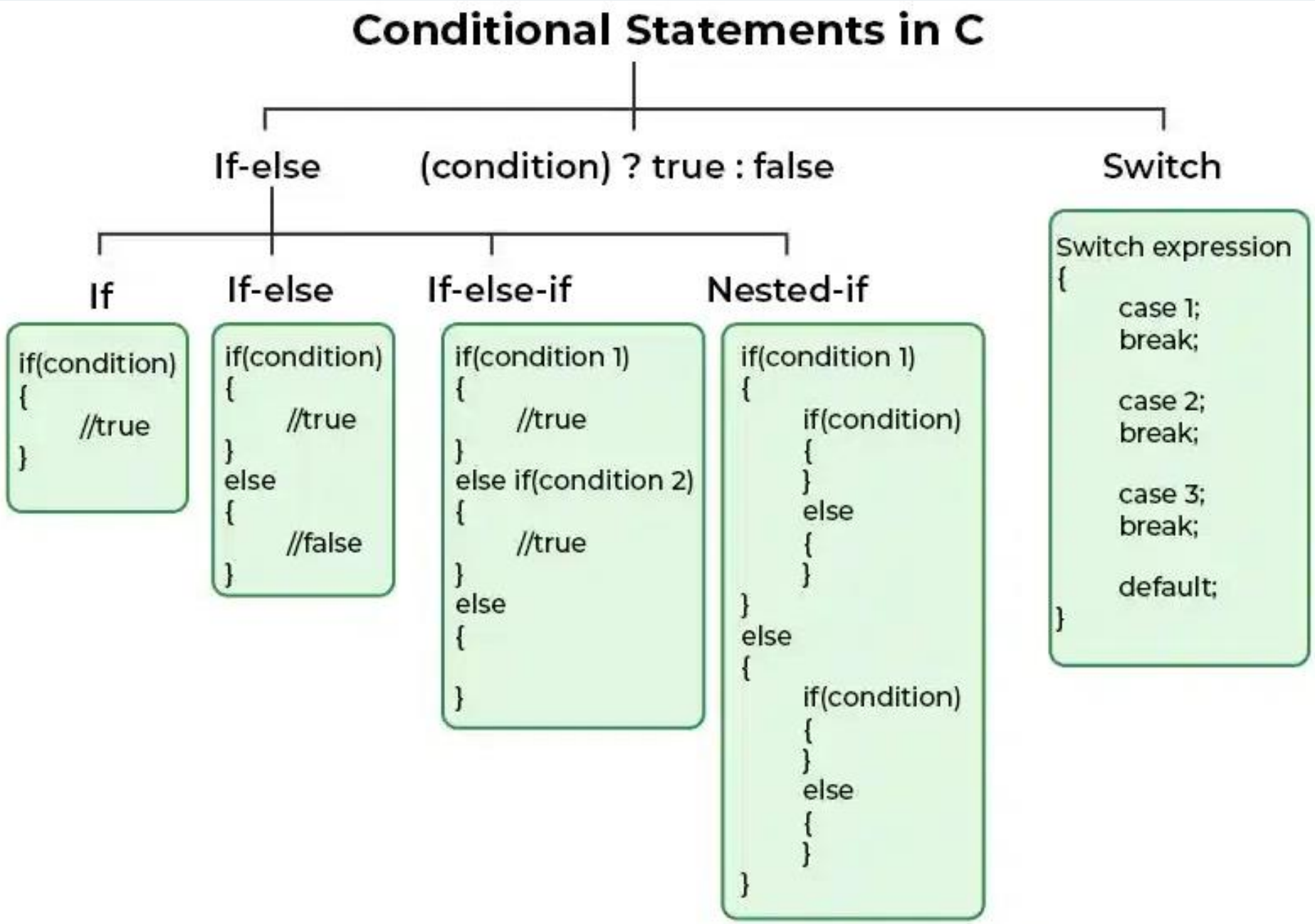
Course Title :- Structured Programming Language Sessional

Course Code :- CSE-122 [SECTION-B]

Level Term: 1-II-A(G1) & 1-II-B(G3,G4)

Khandaker Jannatul Ritu, Lecturer(CSE), BAIUST

Decision Making in C (if , if..else, Nested if, if-else-if)

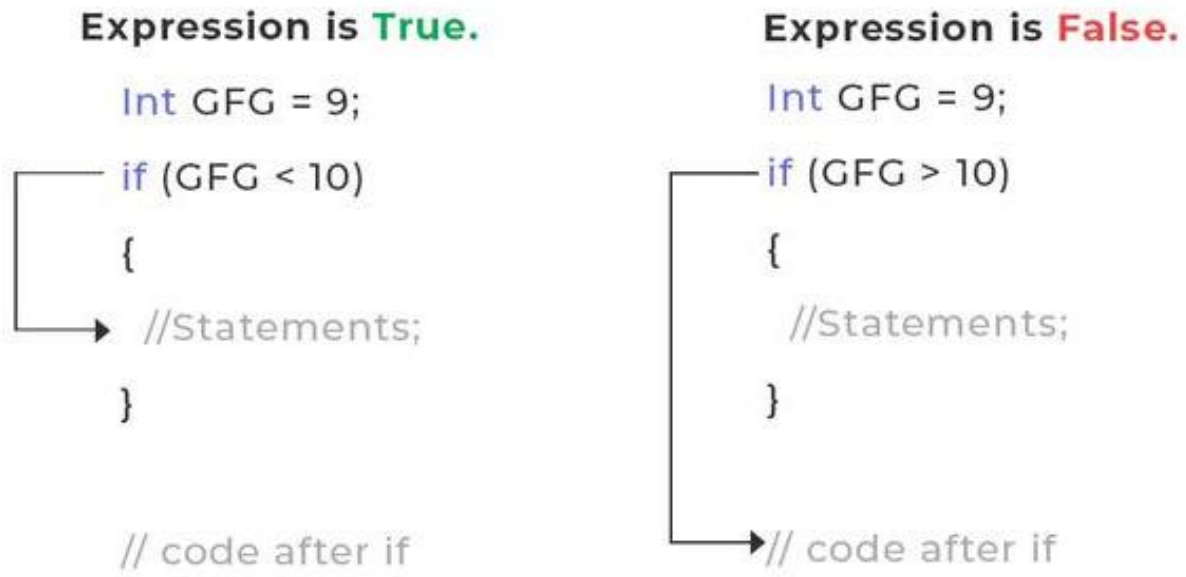


C – if Statement

Syntax of if Statement in C

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

How if in C works?




```
// The syntax of if statement
#include <stdio.h>
int main()
{
      int gfg = 9;
      // if statement with true condition
      if (gfg < 10)
      {
            printf("%d is less than 10", gfg);
      }
      // if statement with false condition
      if (gfg > 20) {
            printf("%d is greater than 20", gfg);
      }
}
```

Output: 9 is less than 10

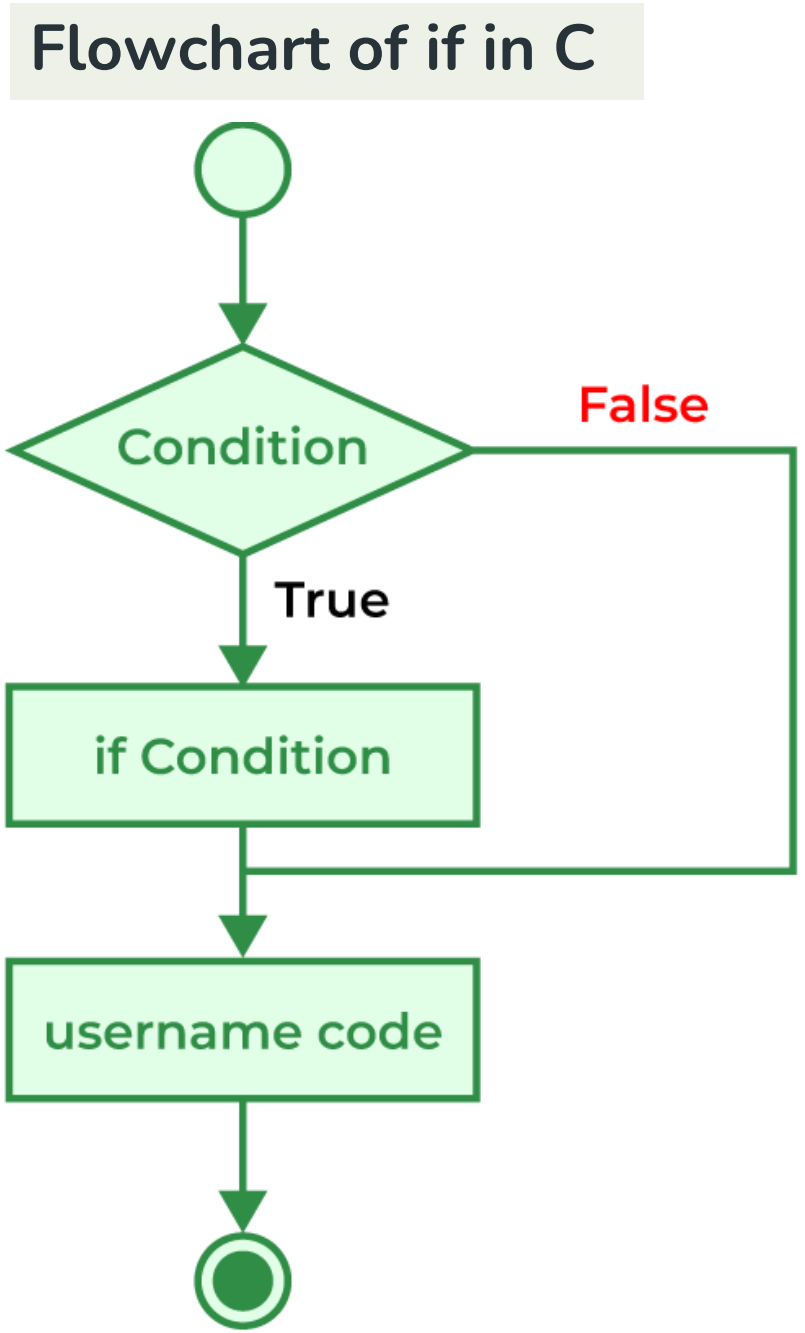
The working of the if statement in C is as follows:

1.STEP 1: When the program control comes to the if statement, the test expression is evaluated.

2.STEP 2A: If the condition is true, the statements inside the if block are executed.

3.STEP 2B: If the expression is false, the statements inside the if body are not executed.

4.STEP 3: Program control moves out of the if block and the code after the if block is executed.



Can we specify multiple conditions in if statement?

We can specify multiple conditions in the if statement but not separately. We have to join these multiple conditions using logical operators making them into a single expression. We can then use this expression in the if statement.

Valid Expressions

```
if (a < b && a < c);
if (a == 25 || a < 25);
```

Invalid Expressions

```
if (a < b, a < c);
```

C if-else Statement

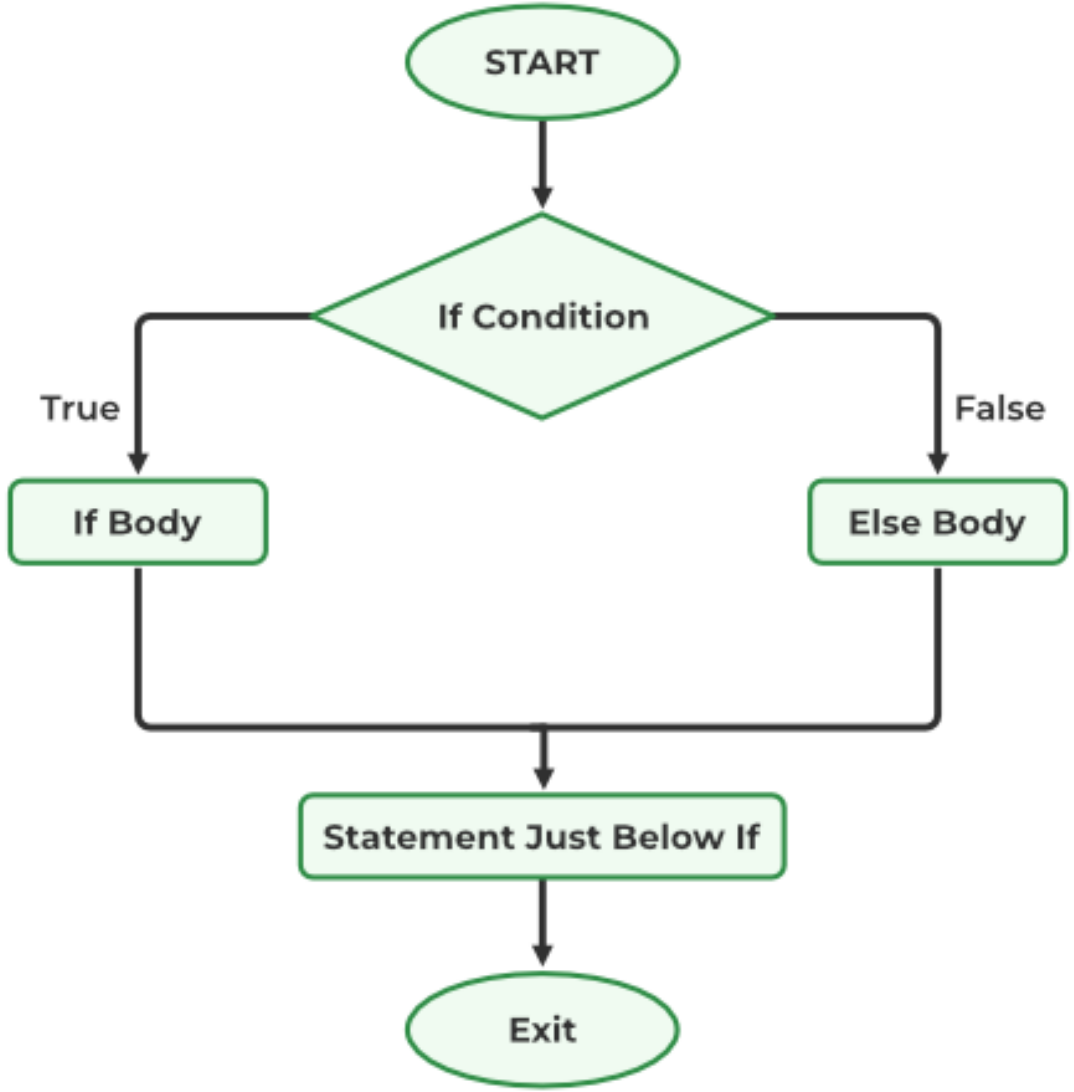
Syntax of if-else

```
if (condition)
 {
   // code executed when the condition is
true
}
else
{
   // code executed when the condition is
false
}
```

Conditions and If Statements

Less than: a < b
Less than or equal to: a <= b
Greater than: a > b
Greater than or equal to: a >= b
Equal to a == b
Not Equal to: a != b

Flowchart of the if-else statement



Q-1: How if-else Statement works?

Working of the if-else statement in C is explained below:

1. When the program control first comes to the if-else block, the test condition is checked.
2. If the test condition is **true**:
    1. The if block is executed.
3. If the test condition is **false**:
    1. The else block is executed
4. After that, the program control continues to the statements below the if-else statement.

Q-2: Can we skip second braces{} around the body of the if-else block in C?

Answer:

We can skip the braces of the body of the if or else block as long as there is only a single statement inside their body. We will get an error if there is more than one statement in the body without braces.

Expression is true.

```
int test = 5;

if (test < 10)
{
   // body of if
}
else
{
   // body of else
}
```

Expression is false.

```
int test = 5;

if (test > 10)
{
   // body of if
}
else
{
   // body of else
}
```

We can also test variables:

Example

```
int x = 20;
int y = 18;
if (x > y) {
  printf("x is greater than y");
}
```

Example

```
int time = 20;
if (time < 18) {
  printf("Good day.");
} else {
  printf("Good evening.");
}
// Outputs "Good evening."
```

```
// C Program to demonstrate the use of if-else statement
#include <stdio.h>
int main(){
    // if block with condition at the start
    if (5 < 10) {
        // will be executed if the condition is true
        printf("5 is less than 10.");
    }
    // else block after the if block
    else {

        // will be executed if the condition is false
        printf("5 is greater that 10.");
    }
}
```

Output: 5 is less than 10.

C Short Hand If Else / Conditional Operator

There is also a short-hand if else, which is known as the **ternary operator** because it consists of three operands. It can be used to replace multiple lines of code with a single line. It is often used to replace simple if else statements:

Syntax

variable = (*condition*) ? *expressionTrue* : *expressionFalse*;

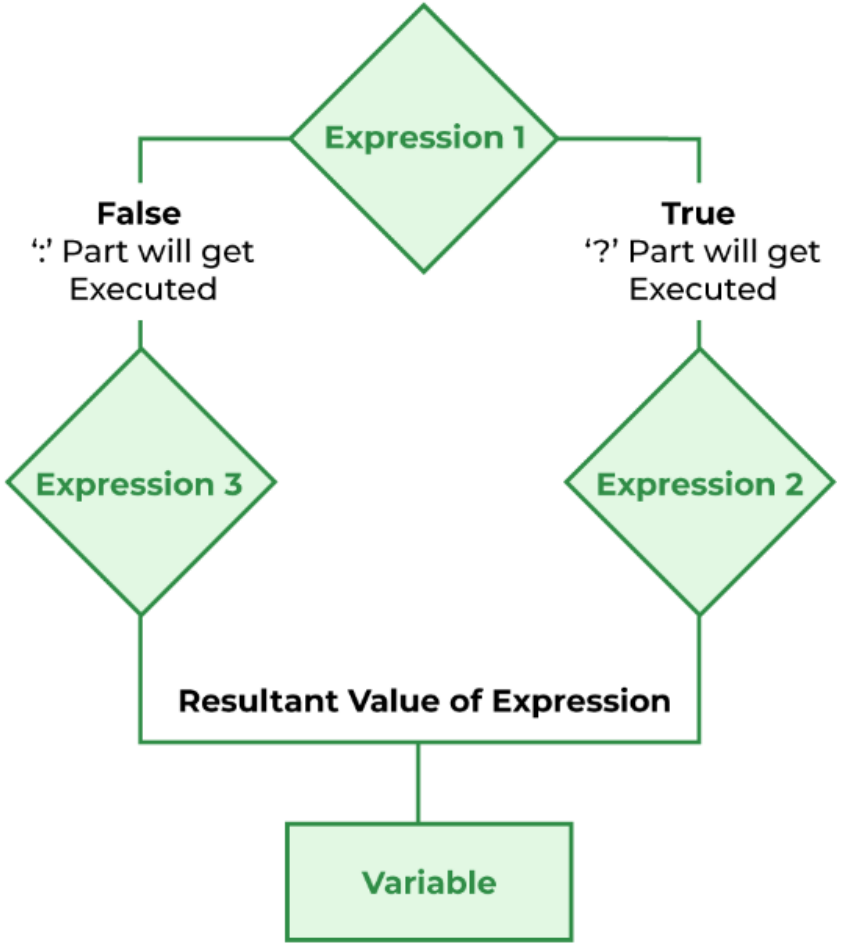
Example

```
int time = 20;
if (time < 18) {
  printf("Good day.");
}
else {
  printf("Good evening.");
}
```

Example

```
int time = 20;
(time < 18) ? printf("Good day.") : printf("Good evening.");
```

Flowchart of Conditional Operator



```
///check voting eligibility
#include <stdio.h>
int main() {
  int age;
  printf("Enter your age: ");
  scanf("%d", &age);
  (age >= 18) ? printf("You can vote") : printf("You cannot vote");
}
```

```
///check even odd number
#include <stdio.h>
int main() {
  int number = 3;
  (number % 2 == 0) ? printf("Even Number") : printf("Odd Number");
}
```

```
// C program to find largest among two numbers using ternary operator
#include <stdio.h>
int main() {
	int m = 5, n = 4;
	(m > n) ? printf("m is greater than n”) : printf("n is greater than m");
}
```

The else if Statement

Use the else if statement to specify a new condition if the first condition is false.

Syntax

```
if (condition1) {
  // block of code to be executed if condition1 is true
}
else if (condition2) {
  // block of code to be executed if the condition1 is false and condition2 is true
}
else {
  // block of code to be executed if the condition1 is false and condition2 is false
}
```

Example

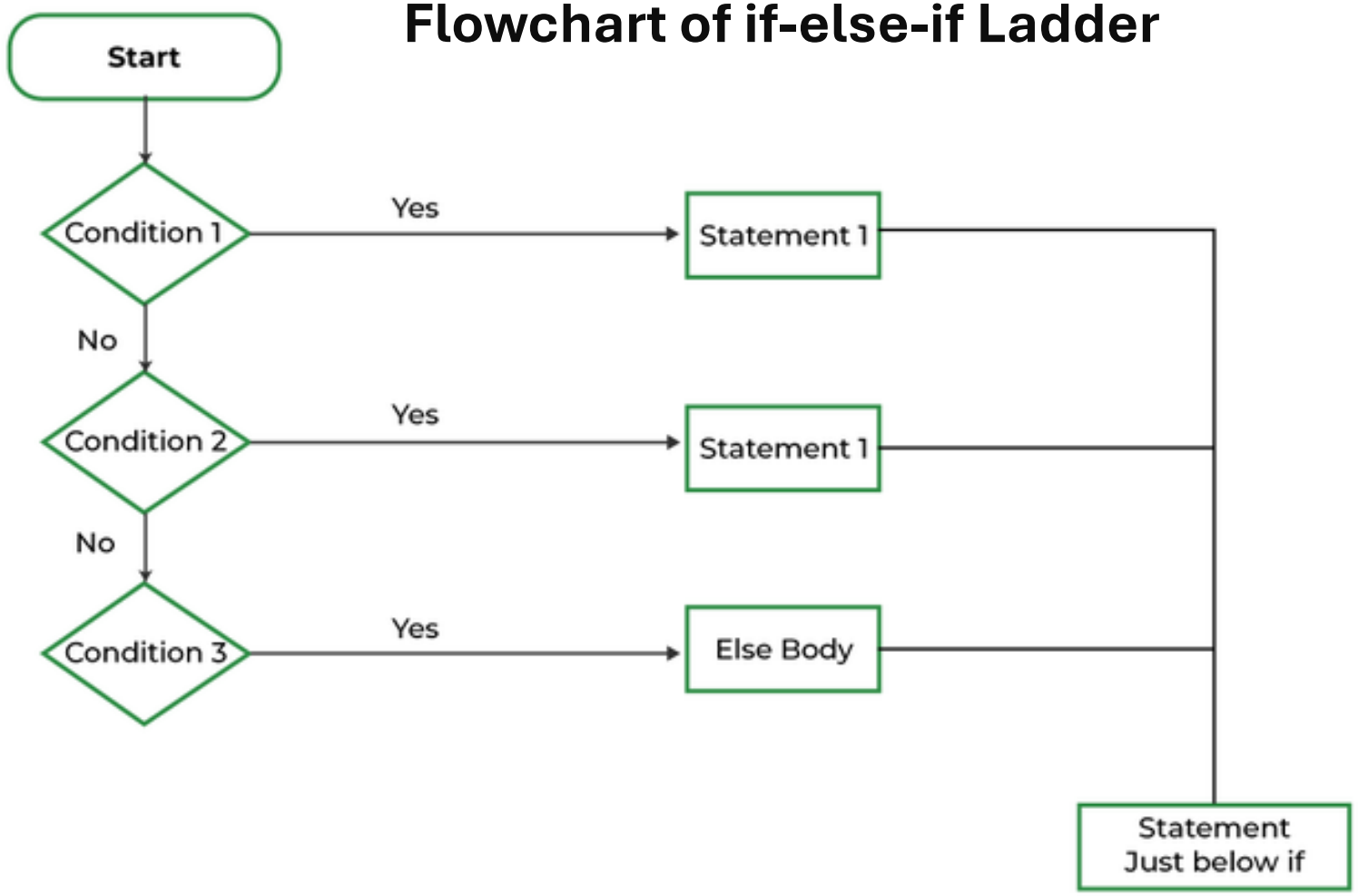
```
int time = 22;
if (time < 10) {
  printf("Good morning.");
}
else if (time < 20) {
  printf("Good day.");
}
else {
  printf("Good evening.");
}

// Outputs "Good evening."
```

// C program to illustrate if-else-if statement

```
#include <stdio.h>
int main() {
    int i = 20;
    if (i == 10)
        printf("i is 10");
    else if (i == 15)
        printf("i is 15");
    else if (i == 20)
        printf("i is 20");
    else
        printf("i is not present");
}
```

Output: i is 20

Flowchart of if-else-if Ladder

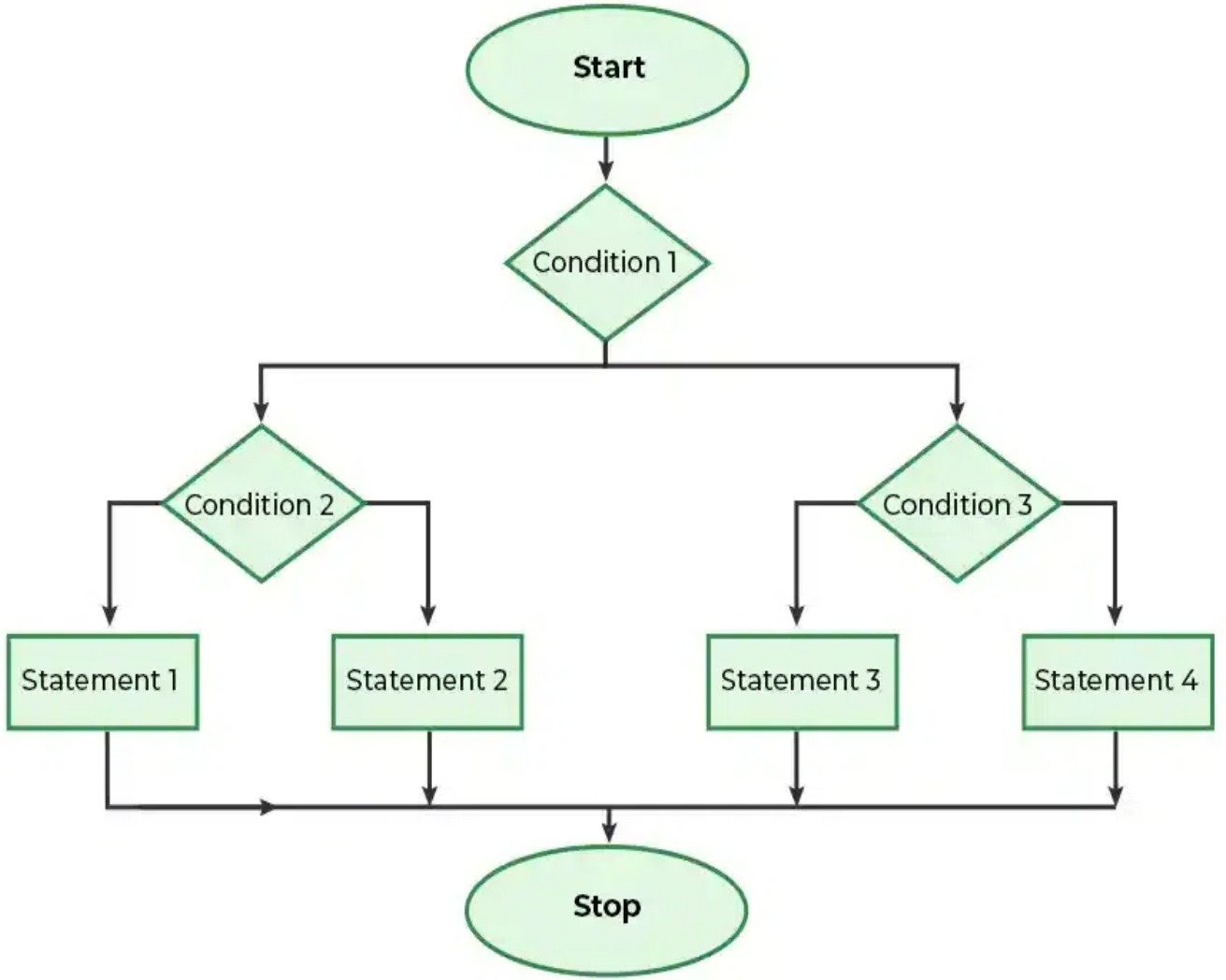


Nested if-else in C

Syntax of Nested if-else

```
if (condition1) {
  // Executes when condition1 is true
  if (condition2) {
    // Executes when condition2 is true
  }
  else{
      // Executes when condition2 is false
  }
}
else{
  //execute when condition-1 is not true
}
```

Flowchart of Nested if-else



```
// C program to illustrate nested-if statement
#include <stdio.h>
int main() {
    int i = 10;
    if (i == 10)  {
        if (i < 15)
            printf("i is smaller than 15\n");
        if (i < 12)
            printf("i is smaller than 12 too\n");
        else
            printf("i is greater than 15");
    }
}
```

Output

```
i is smaller than 15
i is smaller than 12 too
```

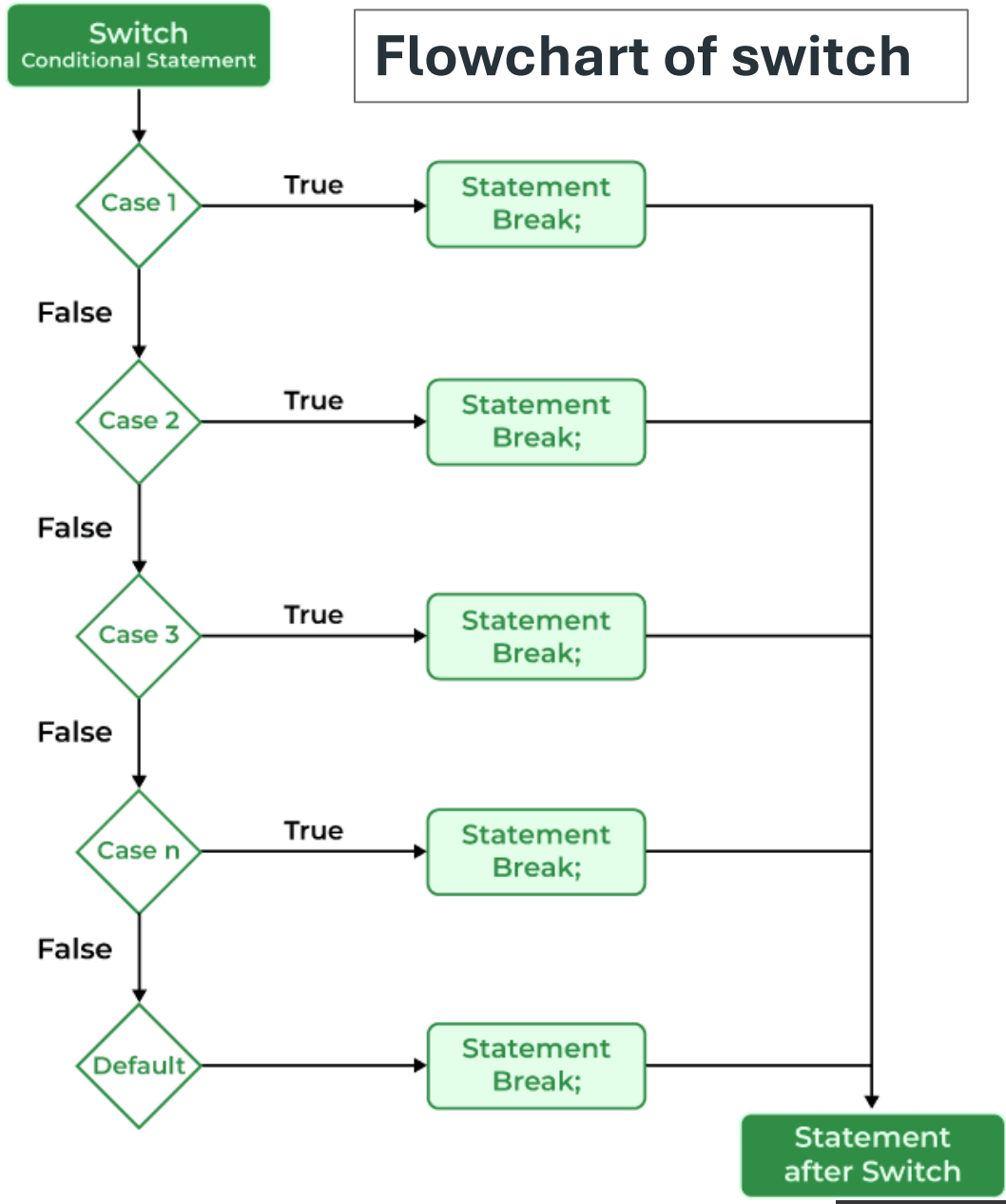
switch Statement in C

The switch case statement is an alternative to the if else if ladder that can be used to execute the conditional code based on the value of the variable specified in the switch statement.

Syntax of switch Statement in C

```
switch(expression){
case value1:
        statement_1;
        break;
case value2:
        statement_2;
        break;
.
.
case value_n:
        statement_n;
        break;
default: default_statement;
}
```

Note: The switch expression should evaluate to either integer or character. It cannot evaluate any other data type.



```
// An example of switch case
#include <stdio.h>
int main(){
      int var = 1;
      switch (var) {
      case 1:
         printf("Case 1 is Matched.");
         break;
      case 2:
         printf("Case 2 is Matched.");
         break;
      case 3:
         printf("Case 3 is Matched.");
         break;
      default:
         printf("Default case is Matched.");
         break;
   }
}
```

Output: Case 1 is Matched.

```
// switch case without break keyword
#include <stdio.h>
int main(){
   int var = 2;

   switch (var) {
    case 1:
       printf("Case 1 is executed.\n");
    case 2:
       printf("Case 2 is executed.\n");
    case 3:
       printf("Case 3 is executed.");
    case 4:
       printf("Case 4 is executed.");
   }
}
```

Output

Case 2 is executed.
Case 3 is executed.Case 4 is executed.

Q-1: Use of Break in switch case

This keyword is used to stop the execution inside a switch block. It helps to terminate the switch block and break out of it. When a break statement is reached, the switch terminates, and the flow of control jumps to the next line following the switch statement.

The **break statement is optional**. If omitted, execution will continue on into the next case. The flow of control will fall through to subsequent cases until a break is reached.

Q-2: Use of Default in switch case

The default keyword is used to specify the set of **statements to execute if there is no case match**.

It is **optional** to use the default keyword in a switch case. Even if the switch case statement does not have a default statement, it would run without any problem.

Q-3: Important Characteristics About Switch Case Statements

1. Switch expression should result in a constant value

If the expression provided in the switch statement does not result in a constant value, it would not be valid. Some valid expressions for switch case will be,

// Constant expressions allowed

switch(1+2+23)

switch(1*2+3%4)

// Variable expression are allowed provided

// they are assigned with fixed values

switch(a*b+c*d)

switch(a+b+c)

2. Expression value should be only of int or char type.

The switch statement can only evaluate the integer or character value. So the switch expression should return the values of type int or char only.

3. Case Values must be Unique

In the C switch statement, duplicate case values are not allowed.

4. Nesting of switch Statements

Nesting of switch statements is allowed, which means you can have switch statements inside another switch. However nested switch statements should be avoided as it makes the program more complex and less readable.

5. The default block can be placed anywhere

Regardless of its placement, the default case only gets executed if none of the other case conditions are met. So, putting it at the beginning, middle, or end doesn’t change the core logic.

What are the differences between switch and if else if ladder in C?

switch	if - else if
It executes the different cases on the basis of the value of the switch variable.	It executes the different blocks based on the condition specified.
It can only evaluate the int or char type expressions.	It can evaluate any type of expression.
Faster and easier to read for the large number of conditions.	It can get messy when there are lots of conditions.

Important programs in control statements [if - else]

if-else:

---

1. basic [if] and [if-else] and [if-else-if]
2. only use if ladder for multiple statements
3. C Program to check whether the number is even or odd.
4. C program for voting system
5. print weeks day
6. calculator using
7. Program to compare two integers using =, > , <
8. C Program to find out if a number is positive or negative or 0
9. check alphabet or not
10. find vowel or consonant
11. Grade According to marks
12. largest among three numbers
13. smallest among three numbers
14. check leap year

switch:

---

1. switch case basic example
2. switch case without break keyword
3. C Program to check whether the number is even or odd.
4. C program for voting system
5. print weeks day
6. calculator using

Example–1-a. basic [if] and [if-else] and [if-else-if]

```
int main()
{  int x;
   printf("Enter a number : ");
   scanf("%d", &x);
   if(x >= 6){
      printf("Number >= 6\n");
   }
   else if(x == 5){
      printf("Number = 5\n");
   }
   else if(x == 4){
      printf("Number = 4\n");
   }
}
```

```
else if(x == 3){
    printf("Number = 3\n");
 }
 else if(x == 2){
    printf("Number = 2\n");
 }
 else {
    printf("Number <= 1\n");
 }
```

Example-1-b. switch case basic example

```
int main(){
  int x; ///basic switch case
   printf("Enter a number : ");
   scanf("%d", &x);
   switch(x){
      case 6:
         printf("number = 6\n");
         break;
      case 5:
         printf("number = 5\n");
         break;
      case 4:
         printf("number = 4\n");
         break;
      case 3:
         printf("number = 3\n");
         break;
      case 2:
         printf("number = 2\n");
         break;
      default:
         printf("number <= 1\n");
         break;
   }
}
```

Example-2-a. only use if ladder for multiple statements

```
#include<stdio.h>
int main()
{
   int x;
   printf("Enter a number : ");
   scanf("%d", &x);
   if(x >= 6){
      printf("Number >= 6\n");
   }
   if(x >= 5){
      printf("Number = 5\n");
   }
   if(x >= 4){
      printf("Number = 4\n");
   }
   if(x >= 3){
      printf("Number = 3\n");
   }
   if(x >= 2){
      printf("Number = 2\n");
   }
   else {
      printf("Number <= 1\n");
   }
}
```

2. switch case without break keyword

```
#include<stdio.h>
int main(){
   int x;
   printf("Enter a number : ");
   scanf("%d", &x);
   switch(x){
      case 6:
         printf("number = 6\n");
      case 5:
         printf("number = 5\n");
      case 4:
         printf("number = 4\n");
      case 3:
         printf("number = 3\n");
      case 2:
         printf("number = 2\n");
      default:
         printf("value not matched\n");
   }
}
```

3. C Program to check whether the number is even or odd.

```
#include<stdio.h>
int main()
{
  int number;
   scanf("%d", &number);
   number = abs(number);
   int result = number % 2 ;

   if(result == 0)
   {
      printf("Even\n");
   }
   else{
      printf("Odd\n");
   }
}
```

```
#include<stdio.h>
int main(){
int number;
   scanf("%d", &number);
   number = abs(number);
   int result = number % 2 ;
   switch(result){
      case 0:
         printf("Even");
         break;
      case 1:
         printf("Odd");
         break;
      default:
         printf("Invalid statements");
   }
}
```

4. C program to check voting eligibility

```
#include<stdio.h>
int main(){
   int age;
   scanf("%d", &age);
   if(age >= 18){
      printf("You can vote\n");
   }
   else{
      printf("You can't vote\n");
   }
}
```

```
How old are you?
= 12
Sorry! You are below 18.
```

```
#include<stdio.h>
int main(){
int age;
   scanf("%d", &age);
   switch(age >= 18)
   {
      case 0:
         printf("You can't vote\n");
         break;
      case 1:
         printf("You can vote\n");
         break;
   }
}
```

5-a. print day names of weeks

```
#include<stdio.h>
int main(){
  int day;
   scanf("%d", &day);
   if(day == 1){
      printf("Sunday\n");
   }
   else if(day == 2){
      printf("Monday\n");
   }
   else if(day == 3){
      printf("Tuesday\n");
   }
   else if(day == 4){
      printf("Wednesday\n");
   }
   else if(day == 5){
      printf("Thursday\n");
   }
   else if(day == 6){
      printf("Friday\n");
   }
   else if(day == 7){
      printf("Saturday\n");
   }
   else{
      printf("Invalid input\n");
   }
}
```

5-b. print day names of weeks

```
#include<stdio.h>
int main(){
  int day;
   scanf("%d", &day);
   switch(day)
   {
   case 1:
      printf("Sunday\n");
      break;
   case 2:
      printf("Monday\n");
      break;
   case 3:
      printf("Tuesday\n");
      break;
   case 4:
      printf("Wednesday\n");
      break;
   case 5:
      printf("Thursday\n");
      break;
   case 6:
      printf("Friday\n");
      break;
   case 7:
      printf("Saturday\n");
   default:
      printf("Invalid");
   }
}
```

6-a.Simple calculator using if else

```
#include<stdio.h>
int main(){
  int a,b;
   char sign;
   scanf("%d %c %d", &a, &sign, &b);
   if(sign == '+'){
      printf("%d %c %d = %d\n", a, sign, b, a+b);
   }
   else if(sign == '-'){
      printf("%d %c %d = %d\n", a, sign, b, a-b);
   }
 else if(sign == '*'){
      printf("%d %c %d = %d\n", a, sign, b, a*b);
   }
   else if(sign == '/'){
      printf("%d %c %d = %0.2f\n", a, sign, b, (float)a/b);
   }
  else if(sign == '%'){
      printf("%d %c %d = %d\n", a, sign, b, a%b);
   }
   else{
      printf("Invalid Sign");
   }
}
```

6-b.Simple calculator using switch case

```
#include<stdio.h>
int main(){
   int a,b;
   char sign;
   scanf("%d %c %d", &a, &sign, &b);
   switch(sign) {
   case '+':
      printf("%d %c %d = %d\n", a, sign, b, a+b);
      break;
   case '-':
      printf("%d %c %d = %d\n", a, sign, b, a-b);
      break;
   case '*':
      printf("%d %c %d = %d\n", a, sign, b, a*b);
      break;
   case '/':
      printf("%d %c %d = %0.2f\n", a, sign, b, (float)a/b);
      break;
   case '%':
      printf("%d %c %d = %d\n", a, sign, b, a%b);
      break;
   default:
      printf("Invalid Sign");
   }
}
```

7. Program to compare two integers using =, > , <

```
#include<stdio.h>
int main(){
   ///num1 > num2  or  num1 < num2   or    num1 == num2
   int num1, num2;
   scanf("%d %d", &num1, &num2);
   if(num1 > num2){
      printf("%d > %d\n", num1, num2);
   }
   else if(num1 < num2){
      printf("%d < %d\n", num1, num2);
   }
   else{
      printf("%d == %d\n", num1, num2);
   }
}
```

```
Enter two integers: 2 2
Result: 2 = 2

Enter two integers: 12 11
Result: 12 > 11

Enter two integers: 12 44
Result: 12 < 44
```

8. C Program to find out if a number is positive or negative or 0

```
#include<stdio.h>
int main(){
   int n;
   scanf("%d", &n);
   if(n > 0) {
      printf("Positive\n");
   }
   else if(n < 0){
      printf("Negative\n");
   }
   else{
      printf("Zero");
   }
}
```

```
Enter an integer: -155
You entered -155.
The if statement is easy.

Enter an integer: 200
The if statement is easy.
```

9. check alphabet(a-z , A-Z) or not

```
#include<stdio.h>
int main(){
char input;
   scanf("%c", &input);
   ///method -1
   if( input>='a' && input<='z'){
      printf("alphabet[smaller]\n");
   }
   else if(input>='A' && input<='Z'){
      printf("alphabet[upper]\n");
   }
   else{
      printf("Not an alphabet\n");
   }
```

```
///method-2
   if (isalpha(input) == 0)
      printf("not an alphabet.");
   else
      printf("alphabet");
}
```

10. find vowel[a, e, I, o, u] or consonant

```
#include <stdio.h>
int main() {
   char vowel;
   printf("Enter an alphabet: ");
   scanf("%c", &vowel);

   if(vowel == 'a' || vowel == 'A'){
      printf("Vowel\n");
   }
   else if(vowel == 'e' || vowel == 'E'){
      printf("Vowel\n");
   }
   else if(vowel == 'i' || vowel == 'I'){
      printf("Vowel\n");
   }
   else if(vowel == 'o' || vowel == 'O'){
      printf("Vowel\n");
   }
   else if(vowel == 'u' || vowel == 'U'){
      printf("Vowel\n");
   }
   else{
      printf("Consonant\n");
   }
}
```

11.Grade According to marks

```
#include<stdio.h>
int main()
{
   int marks;
   scanf("%d", &marks);
   if(marks>=80 && marks<=100){
      printf("You got \"A+\" \n");
   }
   else if(marks>=75 && marks<=79){
      printf("You got \"A\" \n");
   }
   else if(marks>=70 && marks<=74){
      printf("You got \"A-\" \n");
   }
   else if(marks>=65 && marks<=69){
      printf("You got \"B+\" \n");
   }
   else if(marks>=60 && marks<=64){
      printf("You got \"B\" \n");
   }
   else{
      printf("You failed\n");
   }
}
```

12. largest among three numbers

```
#include <stdio.h>
int main() {
  int n1, n2, n3;

  ///type-1
  printf("Enter three different numbers: ");
  scanf("%d %d %d", &n1, &n2, &n3);
  if (n1 >= n2 && n1 >= n3)
    printf("%d is the Largest Number\n", n1);
  if (n2 >= n1 && n2 >= n3)
    printf("%d is the Largest Number\n", n2);
  if (n3 >= n1 && n3 >= n2)
    printf("%d is the Largest Number\n", n3);
```

```
  ///type-2
  if (n1 >= n2 && n1 >= n3)
    printf("%d is the largest number\n", n1);
  else if (n2 >= n1 && n2 >= n3)
    printf("%d is the largest number\n", n2);
  else
    printf("%d is the largest number\n", n3);
}
```

13. smallest among three numbers

```
#include <stdio.h>
int main() {
int n1, n2, n3;
///type-1
printf("Enter three different numbers: ");
scanf("%d %d %d", &n1, &n2, &n3);
if (n1 <= n2 && n1 <= n3)
  printf("%d is the Smallest Number\n", n1);
if (n2 <= n1 && n2 <= n3)
  printf("%d is the Smallest Number\n", n2);
if (n3 <= n1 && n3 <= n2)
  printf("%d is the Smallest Number\n", n3);
  ///type-2
  if (n1 <= n2 && n1 <= n3)
    printf("%d is the Smallest number\n", n1);
  else if (n2 <= n1 && n2 <= n3)
    printf("%d is the Smallest number\n", n2);
  else
    printf("%d is the Smallest number\n", n3);
}
```

14. check leap year

```
#include<stdio.h>
int main(){
   int year;
   printf("Enter a year: ");
   scanf("%d", &year);
   if(year % 400 == 0){
      printf("%d is a leap year\n", year);
   }
   else if (((year % 4 == 0) && (year % 100!= 0))){
      printf("%d is a leap year\n", year);
   }
   else{
      printf("%d is not a leap year\n", year);
   }
}
```

What is Loop?

Loop

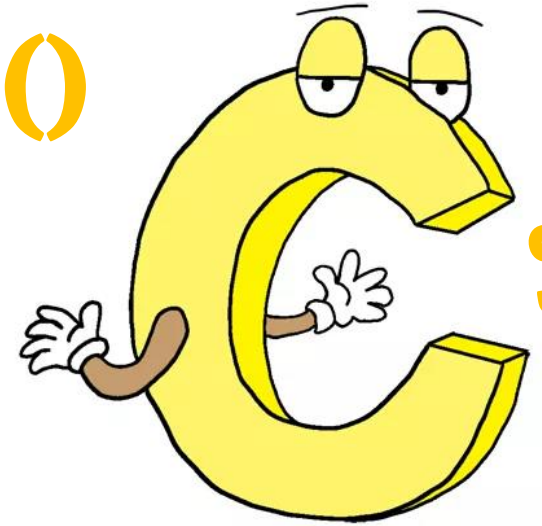
Loop

while()

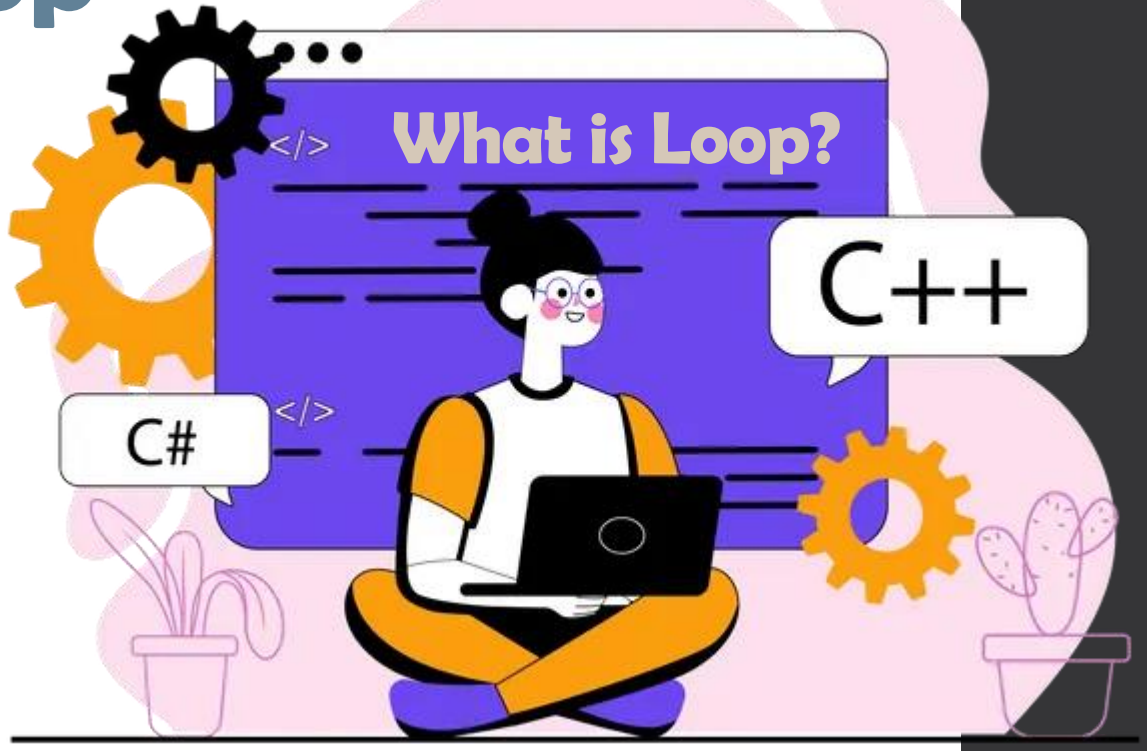
for()

Loop

Oh! I



do while()



What is Loop?

Loop

while()

Loop

do while()

Loop

Loop

Loop

for()

Why C – Loops?

Loops in programming are used to repeat a block of code until the specified condition is met. A loop statement allows programmers to execute a statement or group of statements multiple times without repetition of code.

```
#include <stdio.h>
int main()
{
    printf( "Hello World\n");
    printf( "Hello World\n");
    printf( "Hello World\n");
    printf( "Hello World\n");
    printf( "Hello World\n");
    printf( "Hello World\n");
    printf( "Hello World\n");
    printf( "Hello World\n");
    printf( "Hello World\n");
    printf( "Hello World\n");
}
```

Output

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

Loop

for(start ; condition ; increment/decrement)

for(i = 0; i <= 5; i++)

i = 0 ; 0 <= 5 → printf(“%d”, i); i = 0+1=1 ;

1 <= 5 → printf(“%d”, i); i = 1+1=2 ;

2 <= 5 → printf(“%d”, i); i = 2+1=3 ;

3 <= 5 → printf(“%d”, i); i = 3+1=4 ;

4 <= 5 → printf(“%d”, i); i = 4+1=5 ;

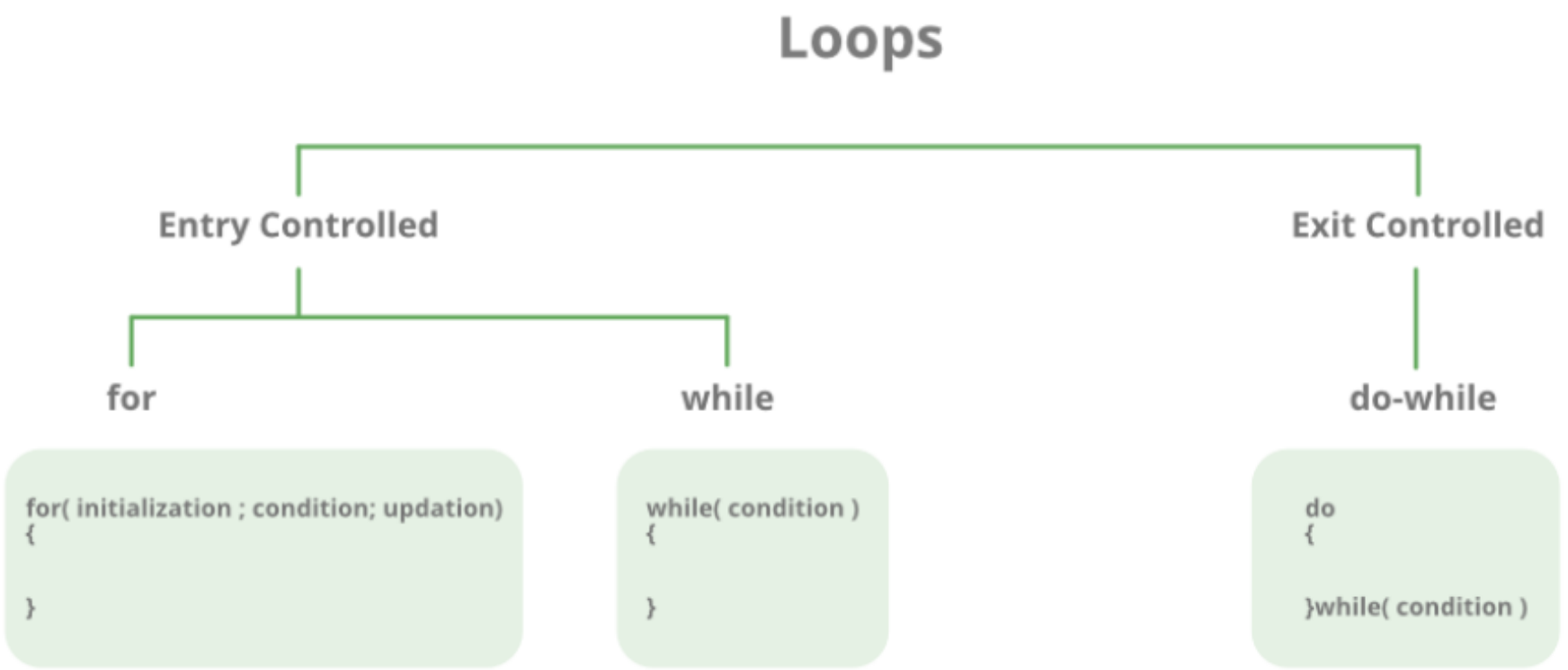
5 <= 5 → printf(“%d”, i); i = 5+1=6 ;

6 <= 5 → condition not true, so exit

There are mainly two types of loops in C Programming:

1.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.

2.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 Exit Controlled loop.



Loop Type	Description
for loop	first Initializes, then condition check, then executes the body and at last, the update is done.
while loop	first Initializes, then condition checks, and then executes the body, and updating can be inside the body.
do-while loop	do-while first executes the body and then the condition check is done.

for Loop

for loop in C programming is a repetition control structure that allows programmers to write a loop that will be executed a specific number of times. for loop enables programmers to perform n number of steps together in a single line.

Syntax:

```
for (initialize expression; test expression; update expression)
{
  // body of for loop
}
```

Example:

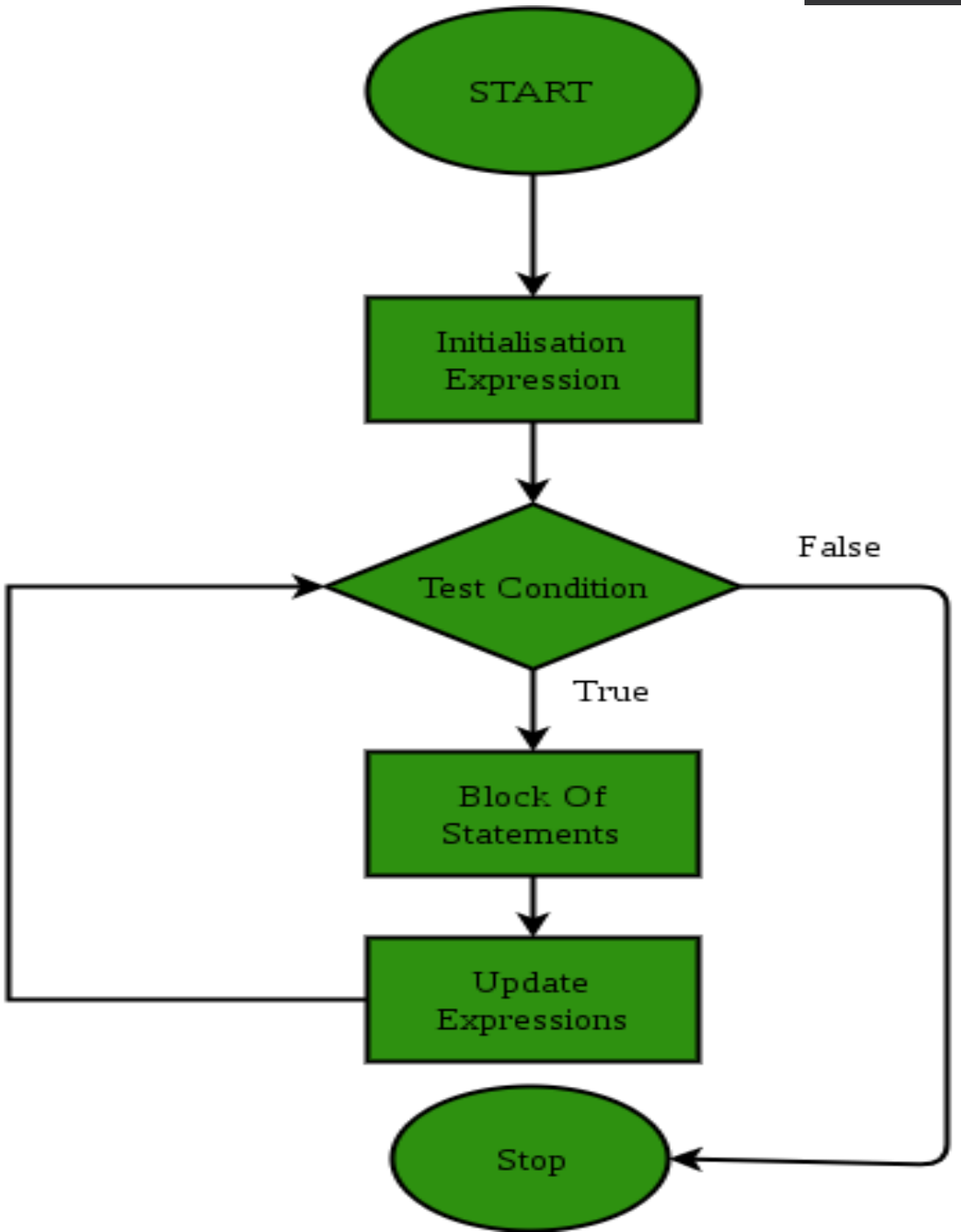
```
for(int i = 0; i < n; ++i)
{
   printf("Body of for loop which will execute till n");
}
```

If the test condition will be false then it will stop.

•**Initialization Expression:** In this expression, we assign a loop variable or loop counter to some value. for example: int i=1;

•**Test Expression:** In this expression, test conditions are performed. If the condition evaluates to true then the loop body will be executed and then an update of the loop variable is done. If the test expression becomes false then the control will exit from the loop. for example, i<=9;

•**Update Expression:** After execution of the loop body loop variable is updated by some value it could be incremented, decremented, multiplied, or divided by any value.



```c
#include <stdio.h>
// Driver code
int main(){
    int i = 0;
    for (i = 1; i <= 10; i++){
        printf( "Hello World\n");
    }
}
```

Output

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

```c
#include <stdio.h>
int main(){
    int i;
    // for loop without curly braces
    for (i = 1; i <= 10; i++)
        printf("%d ", i);
    printf("\nThis statement executes after for loop end!!!!");
}
```

Output

```
1 2 3 4 5 6 7 8 9 10
This statement executes after for loop end!!!!
```

```
#include <stdio.h>
int main() {
  int i, j;
  for (i = 1; i <= 2; ++i)
  {
    printf("Outer: %d\n", i);
    for (j = 1; j <= 3; ++j)
    {
      printf(" Inner: %d\n", j);
    }
  }
}
```

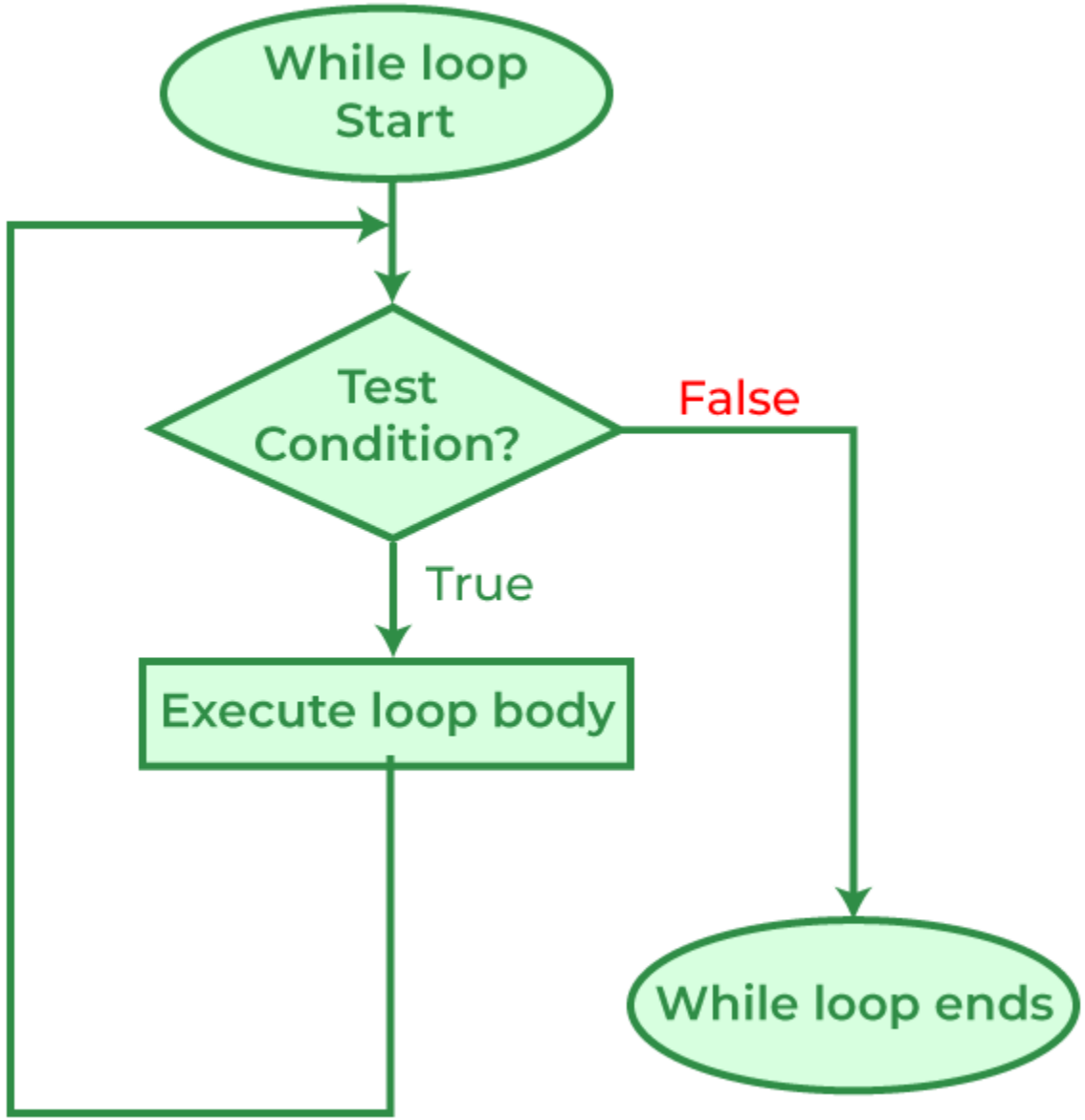
```
Outer: 1
 Inner: 1
 Inner: 2
 Inner: 3
Outer: 2
 Inner: 1
 Inner: 2
 Inner: 3
```

While Loop

While loop does not depend upon the number of iterations. In for loop the number of iterations was previously known to us but in the While loop, the execution is terminated on the basis of the test condition. If the test condition will become false then it will break from the while loop else body will be executed.

Syntax:

```
initialization_expression;
while (test_expression)
{
   // body of the while loop

   update_expression;
}
```



```c
#include <stdio.h>
int main()
{
    int i = 2;
    while(i < 10)
    {
        printf( "Hello World\n");
        i++;
    }
}
```

Output

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

Traverse a while() loop

```
i=0;
while(i <= 5)
{
    printf("%d\n", i);
    i++;
}
```

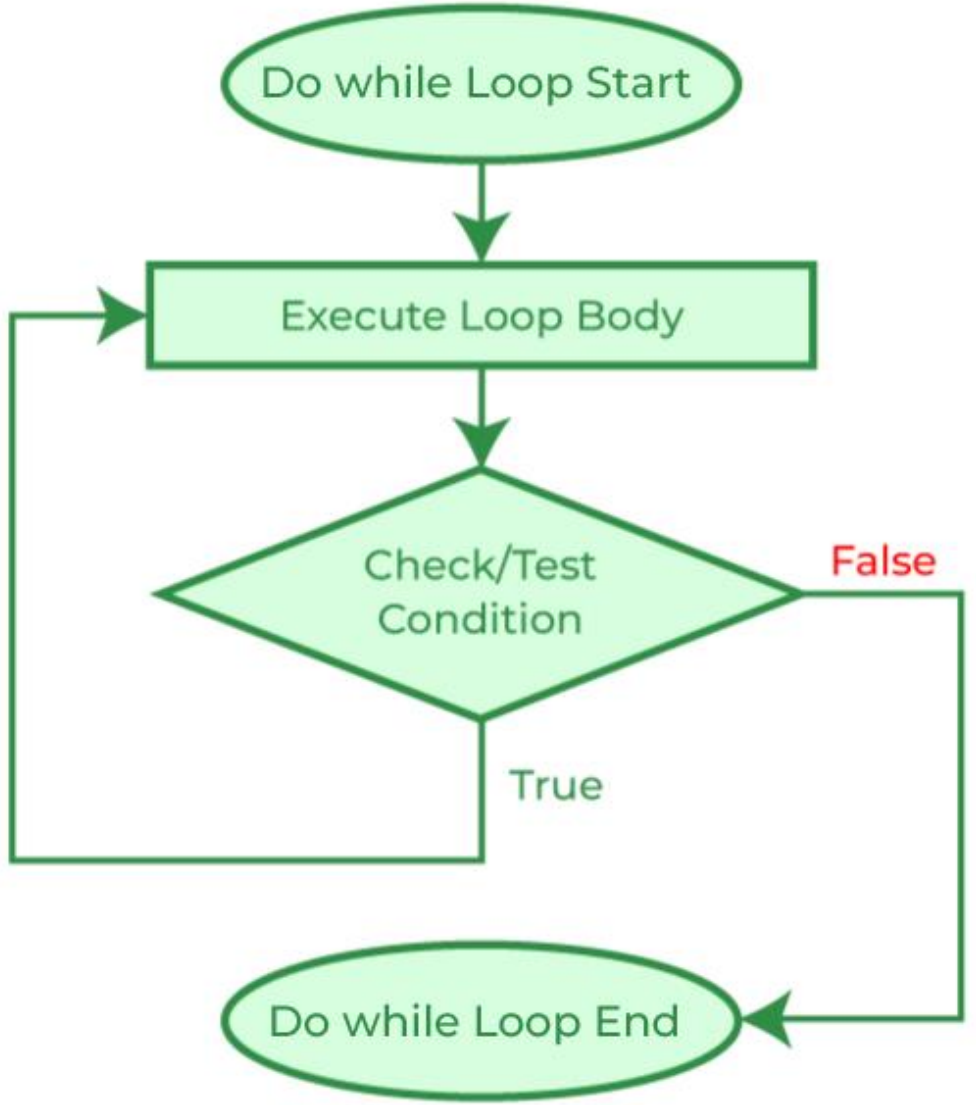
```
i = 0   while(0 <= 5)    printf("%d\n", i);   i = i+1 = 0+1=1
        while(1 <= 5)    printf("%d\n", i);   i = i+1 = 1+1=2
        while(2 <= 5)    printf("%d\n", i);   i = i+1 = 2+1=3
        while(3 <= 5)    printf("%d\n", i);   i = i+1 = 3+1=4
        while(4 <= 5)    printf("%d\n", i);   i = i+1 = 4+1=5
        while(5 <= 5)    printf("%d\n", i);   i = i+1 = 5+1=6
        while(6 <= 5)  → condition overflow, so exit
```

do-while Loop

The do-while loop is similar to a while loop b
the only difference lies in the do-while loop te
condition which is tested at the end of the boc
In the do-while loop, the loop body will execu
at least once irrespective of the test condition.

Syntax:

```
initialization_expression;
do
{
   // body of do-while loop
   update_expression;

} while (test_expression);
```



Traverse a do while() loop

Example-1:

```
i = 0;
    do{
        printf("%d\n", i);
        i++;
    }while(i <= 5);
```

```
i = 0   printf("%d\n", i);   i = i + 1 = 0 + 1 = 1     while (1 <= 5)
        printf("%d\n", i);   i = i + 1 = 1 + 1 = 2     while (2 <= 5)
        printf("%d\n", i);   i = i + 1 = 2 + 1 = 3     while (3 <= 5)
        printf("%d\n", i);   i = i + 1 = 3 + 1 = 4     while (4 <= 5)
        printf("%d\n", i);   i = i + 1 = 4 + 1 = 5     while (5 <= 5)
        printf("%d\n", i);   i = i + 1 = 5 + 1 = 6     while (6 <= 5)
        → now condition overflow, so exit the loop
```

Ex-2:

```
#include <stdio.h>
int main()
{
// Initialization expression
int i = 2;
do{
      // loop body
      printf( "Hello World\n");

      // Update expression
      i++;

      // Test expression
} while (i < 1);

}
```

Output

```
Hello World
```

Ex-3:

```
#include <stdio.h>
int main(){
      int i = 0;
      // do while loop
      do {
            printf("Geeks\n");
            i++;
      } while (i < 3);
}
```

Output

```
Geeks
Geeks
Geeks
```

Loop Control Statements

Name	Description
break statement	the break statement is used to terminate the switch and loop statement. It transfers the execution to the statement immediately following the loop or switch.
continue statement	continue statement skips the remainder body and immediately resets its condition before reiterating it.
goto statement	goto statement transfers the control to the labeled statement.

Infinite Loop

```
#include <stdio.h>
int main (){
	int i;
	for ( ; ; )
	{
		printf("This loop will run forever.\n");
	}
}
```

Output

```
This loop will run forever.
This loop will run forever.
This loop will run forever.
...
```

```
#include <stdio.h>
int main() {
	while (1)
		printf("This loop will run forever.\n");
}
```

Output

```
This loop will run forever.
This loop will run forever.
This loop will run forever.
...
```

```
#include <stdio.h>
int main(){
	do
	{
		printf("This loop will run
	forever.\n");
	} while (1);
}
```

Output

```
This loop will run forever.
This loop will run forever.
This loop will run forever.
...
```

Nested for loop in C

C provides the feature of a nested loop where we can place a loop inside another loop.

Syntax:

```
for ( initialization; condition; increment ) {
  for ( initialization; condition; increment ) {
    // statement of inside loop
  }
  // statement of outer loop
}
```

Nested while loop in C

C provides the feature of a nested loop where we can place a loop inside another loop.

Syntax:

```
while (condition) {
 while (condition) {
    // statement of inside loop
  }
  // statement of outer loop
}
```

Nested do while loop in C

C provides the feature of a nested loop where we can place a loop inside another loop.

Syntax:

```
do{
        do{
           // statement of inside loop
         } while(condition);

  // statement of outer loop

} while(condition);
```

Nested loop in C

C provides the feature of a nested loop where we can place a loop inside another loop.

Syntax:

```
while (condition) {
        for  ( initialization; condition; increment ) {
            // statement of inside loop
          }
  // statement of outer loop
}
```

for Versus while

```
#include <stdio.h>
int main(){
    int sum=0, i;
    for(i=1;i<=5;i++){
        sum=sum+i;
    }
    printf("SUM = %d" , sum);
}
```

Output

SUM = 15

```
#include<stdio.h>
int main(){
    int no=1, sum=0;
    while(no<=5){
        sum=sum+no;
        no++;
    }

    printf("SUM = %d" , sum);
}
```

Output

SUM = 15

Part-1 : loop Practise problems

1. a. Traverse a for() loop → print numbers 1 to n & print numbers n to 1
   b. Traverse a while() loop → print numbers 1 to n & print numbers n to 1
   c. Traverse a do while() loop → print numbers 1 to n & print numbers n to 1

2. Print all even/odd numbers for 1 to n
3. Sum of all numbers 1 to n

4. a. C Program to Display Characters from a to z Using Loop
   b. C Program to Display Characters from A to Z Using Loop

5. Multiplication Table Up to 10
6. C Program to Find Factorial of a Number
7. C Program to Check Whether a Number is Prime or Not

1 a. Traverse a for() loop → print numbers 1 to n & print numbers n to 1

///traverse a for loop: 0 to n

```
#include<stdio.h>
int main()
{
   int n,i;
   printf("Enter n : ");
   scanf("%d", &n);

   printf("-----print 0 to n----\n");
   /// print 0 to n
   for(i = 0 ; i<=n; i++)
   {
      printf("hello %d\n", i);
   }
}
```

///traverse a for loop: n to 0

```
#include<stdio.h>
int main()
{
   int n,i;
   printf("Enter n : ");
   scanf("%d", &n);

printf("-----print n to 0-----\n");
   for(i = n ; i>= 0 ; i--)
   {
      printf("hello %d\n", i);
   }

}
```

1 b. Traverse a while() loop → print numbers 1 to n & print numbers n to 1

///traverse a while loop: 0 to n

```
#include<stdio.h>
int main(){
   int n,i;
   printf("Enter n : ");
   scanf("%d", &n);
   printf("-----print 0 to n----\n");
   /// print 0 to n
   i=0;
   while(i <= n)
   {
      printf("%d\n", i);
      i++;
   }
}
```

///traverse a while loop: n to 0

```
#include<stdio.h>
int main(){
   int n,i;
   printf("Enter n : ");
   scanf("%d", &n);
   /// print n to 0
   printf("-----print n to 0-----\n");
   i = n;
   while( i >= 0)
   {
      printf("%d\n", i);
      i--;
   }
}
```

1 c. Traverse a do while() loop → print numbers 1 to n & print numbers n to 1

///traverse a do while() loop: 0 to n

```
#include<stdio.h>
int main(){
   int n,i;
   printf("Enter n : ");
   scanf("%d", &n);

   printf("-----print 0 to n----\n");
   i = 0;
   do{
      printf("%d\n", i);
      i++;
   }while(i <= n);
}
```

///traverse a do while() loop: n to 0

```
#include<stdio.h>
int main(){
   int n,i;
   printf("Enter n : ");
   scanf("%d", &n);

   printf("-----print n to 0-----\n");
   i = n;
   do{
      printf("%d\n", i);
      i--;
   }while(i>=0);
}
```

2. Print all even/odd numbers for 1 to n

```
#include<stdio.h>
int main(){
   int number,i;

   printf("Enter a number: ");
   scanf("%d", &number);

   for(i = 0; i <= number; i++){
      if( i % 2 == 0){
         printf("%d : Even\n", i);
      }
      else{
         printf("%d : Odd\n", i);
      }
   }
}
```

3. sum of all numbers for 1 to n

```
#include<stdio.h>
int main()
{
   int number,i, sum;
   printf("Enter a number: ");
   scanf("%d", &number);

   sum = 0;
   for(i = 1; i <= number; i++)
   {
      sum = sum + i;
   }
   printf("sum [1 to n] : %d\n", sum);
}
```

4 a. Display Characters from a to z Using Loop

```
#include<stdio.h>
int main()
{
   char i;
   printf("---Print small letters [a to z] ---\n");
   for(i = 'a'; i <= 'z'; i++)
   {
      printf("%c ", i);
   }
}
```

4 b. Display Characters from A to Z Using Loop

```
#include<stdio.h>
int main()
{
   char i;
   printf("\n---Print capital letters [A to Z]---\n");
   for(i = 'A'; i <= 'Z'; i++)
   {
      printf("%c ", i);
   }
}
```

5. Multiplication Table Up to 10

```
#include<stdio.h>
int main(){
   int n,i;
   printf("Enter a number : ");
   scanf("%d", &n);

   for(i = 1 ; i <= 10 ; i++){
      printf("%d * %d = %d\n",i, n, (i*n));
   }
}
```

```
Enter a number : 10
1 * 10 = 10
2 * 10 = 20
3 * 10 = 30
4 * 10 = 40
5 * 10 = 50
6 * 10 = 60
7 * 10 = 70
8 * 10 = 80
9 * 10 = 90
10 * 10 = 100
```

6. C Program to Find Factorial of a Number

```
#include<stdio.h>
int main()
{
   int factorial, result, i;
   printf("Enter a number : ");
   scanf("%d", &factorial);

   result = 1;
   for(i = 2; i<=factorial; i++)
   {
      result = result * i ;
   }
   printf("Factorial : %d\n", result);
}
```

5! = 1 x 2 x 3 x 4 x 5

1 * 2 = 2

2 * 3 = 6

6 * 4 = 24

24 * 5 = 120

7. C Program to check whether a number is prime or not.

```
#include<stdio.h>
#include<stdbool.h>
int main(){
   int n;
   scanf("%d", &n);
   printf("\n");
   if(n<=1){ //n =  0,1
      printf("Not prime");
   }
   else if(n == 2){
      printf("Prime!!");
   }
   else if(n!=2 && n % 2 == 0){ //all even numbers
      printf("Not prime");
   }
   else{
      //eikhane shob odd numbers ashbe
      bool check = true;
      for(int i = 2 ; i<= n - 1 ; i++){
         if(n % i == 0){
            printf("Not prime");
            check = false;
            break;
         }
      }
      /// check == false or check == true
      if(check == true){
         printf("Prime!!");
      }
   }
   printf("\n\n\n");
}
```

Part-2 : Pattern printing - nested loop examples

Pattern Type - 1				
N = 5 1 0 0 1 1 1 0 0 0 0 1 1 1 1 1	N = 5 1 1 0 1 0 1 1 0 1 0 1 0 1 0 1	N = 5 1 2 2 3 3 3 4 4 4 4 5 5 5 5 5	N = 5 1 1 2 1 2 3 1 2 3 4 1 2 3 4 5	N = 5 * * * * * * * * * * * * * * *
N = 5 a b b c c c d d d d e e e e e	N = 5 a a b a b c a b c d a b c d e	N = 5 A B B C C C D D D D E E E E E	N = 5 A A B A B C A B C D A B C D E	N = 5 # # # # # # # # # # # # # # #

```c
#include<stdio.h>
int main(){
    int n, row, col;
    printf("Enter n : ");
    scanf("%d", &n);

    for(row = 1; row<= n; row++)
    {
        for(col = 1; col <= row; col++)
        {
            printf("* ");
        }
        printf("\n");
    }
}
```

```c
/*
printf("%d ", row % 2);
printf("%d ", col % 2);
printf("%d ", row);
printf("%d ", col);
printf("* ");
printf("%c ", 96+row);
printf("%c ", 96+col);
printf("%c ", 64+row);
printf("%c ", 64+col);
printf("# ");
*/
```

Output:

```
Enter n : 7
*
* *
* * *
* * * *
* * * * *
* * * * * *
* * * * * * *
```

Pattern Type - 2				
N = 5 1 1 1 1 1 0 0 0 0 1 1 1 0 0 1	N = 5 1 0 1 0 1 1 0 1 0 1 0 1 1 0 1	N = 5 5 5 5 5 5 4 4 4 4 3 3 3 2 2 1	N = 5 1 2 3 4 5 1 2 3 4 1 2 3 1 2 1	N = 5 * * * * * * * * * * * * * * *
N = 5 e e e e e d d d d c c c b b a	N = 5 a b c d e a b c d a b c a b a	N = 5 E E E E E D D D D C C C B B A	N = 5 A B C D E A B C D A B C A B A	N = 5 # # # # # # # # # # # # # # #

```
#include<stdio.h>
int main()
{
   int n, row, col;
   printf("Enter n : ");
   scanf("%d", &n);

   for(row = n; row >= 1; row--)
   {
      for(col = 1; col<= row; col++)
      {
         printf("%c ", 64+col);
      }
      printf("\n");
   }
}
```

```
/*
printf("%d ", row % 2);
printf("%d ", col % 2);
printf("%d ", row);
printf("%d ", col);
printf("* ");
printf("%c ", 96+row);
printf("%c ", 96+col);
printf("%c ", 64+row);
printf("%c ", 64+col);
printf("# ");
*/
```

Output:

```
A B C D E
A B C D
A B C
A B
A
```

Pattern Type - 3

N = 5 1 0 0 1 1 1 0 0 0 0 1 1 1 1 1 0 0 0 0 1 1 1 0 0 1	N = 5 1 1 0 1 0 1 1 0 1 0 1 0 1 0 1 1 0 1 0 1 0 1 1 0 1	N = 5 1 2 2 3 3 3 4 4 4 4 5 5 5 5 5 4 4 4 4 3 3 3 2 2 1	N = 5 1 1 2 1 2 3 1 2 3 4 1 2 3 4 5 1 2 3 4 1 2 3 1 2 1	N = 5 *
N = 5 a b b c c c d d d d e e e e e d d d d c c c b b a	N = 5 a a b a b c a b c d a b c d e a b c d a b c a b a	N = 5 A B B C C C D D D D E E E E E D D D D C C C B B A	N = 5 A A B A B C A B C D A B C D E A B C D A B C A B A	N = 5 # ## ### #### ##### #### ### ## #

```c
#include<stdio.h>
int main(){
    int n, row, col;
    printf("Enter n : ");
    scanf("%d", &n);
    for(row = 1; row<= n; row++){
        for(col = 1; col <= row; col++) {
            printf("# ");
        }
        printf("\n");
    }
    for(row = n-1; row >= 1; row--){
        for(col = 1; col<= row; col++){
            printf("# ");
        }
        printf("\n");
    }
}
```

```c
/*
printf("%d ", row % 2);
printf("%d ", col % 2);
printf("%d ", row);
printf("%d ", col);
printf("* ");
printf("%c ", 96+row);
printf("%c ", 96+col);
printf("%c ", 64+row);
printf("%c ", 64+col);
printf("# ");
*/
```

Output:

```
#
# #
# # #
# # # #
# # # # #
# # # #
# # #
# #
#
```

Pattern Type - 4				
N = 5 1 00 111 0000 11111	N = 5 1 10 101 1010 10101	N = 5 1 22 333 4444 55555	N = 5 1 12 123 1234 12345	N = 5 * ** *** **** *****
N = 5 a bb ccc dddd eeeee	N = 5 a ab abc abcd abcde	N = 5 A BB CCC DDDD EEEEE	N = 5 A AB ABC ABCD ABCDE	N = 5 # ## ### #### #####

```
#include<stdio.h>
int main(){
   int n, row, col;
   printf("Enter n : ");
   scanf("%d", &n);

   for(row = 1; row<= n; row++){
      for(col = 1; col <= n-row; col++){
         printf(" ");
      }
      for(col = 1; col<= row; col++){
         printf("#");
      }
      printf("\n");
   }
}
```

```
/*
printf("%d", row % 2);
printf("%d", col % 2);
printf("%d", row);
printf("%d", col);
printf("*");
printf("%c", 96+row);
printf("%c", 96+col);
printf("%c", 64+row);
printf("%c", 64+col);
printf("#");
*/
```

Output:

```
     #
    ##
   ###
  ####
 #####
```

Pattern Type - 5				
N = 5 11111 0000 111 00 1	N = 5 10101 1010 101 10 1	N = 5 55555 4444 333 22 1	N = 5 12345 1234 123 12 1	N = 5 ***** **** *** ** *
N = 5 eeeee dddd ccc bb a	N = 5 abcde abcd abc ab a	N = 5 EEEEE DDDD CCC BB A	N = 5 ABCDE ABCD ABC AB A	N = 5 ##### #### ### ## #

```
#include<stdio.h>
int main(){
    int n, row, col;
    printf("Enter n : ");
    scanf("%d", &n);

    for(row = n; row >= 1; row--){
        for(col = 1; col <= n-row; col++) {
            printf(" ");
        }
        for(col = 1; col<= row; col++) {
            printf("%c", 64+col);
        }
        printf("\n");
    }
}
```

```
/*
printf("%d", row % 2);
printf("%d", col % 2);
printf("%d", row);
printf("%d", col);
printf("*");
printf("%c", 96+row);
printf("%c", 96+col);
printf("%c", 64+row);
printf("%c", 64+col);
printf("#");
*/
```

Output:

```
ABCDE
 ABCD
  ABC
   AB
    A
```

Pattern Type - 6

N = 5 1 0 0 1 1 1 0 0 0 0 1 1 1 1 1 0 0 0 0 1 1 1 0 0 1	N = 5 1 1 0 1 0 1 1 0 1 0 1 0 1 0 1 1 0 1 0 1 0 1 1 0 1	N = 5 1 2 2 3 3 3 4 4 4 4 5 5 5 5 5 4 4 4 4 3 3 3 2 2 1	N = 5 1 1 2 1 2 3 1 2 3 4 1 2 3 4 5 1 2 3 4 1 2 3 1 2 1	N = 5 *
N = 5 a b b c c c d d d d e e e e e d d d d c c c b b a	N = 5 a a b a b c a b c d a b c d e a b c d a b c a b a	N = 5 A B B C C C D D D D E E E E E D D D D C C C B B A	N = 5 A A B A B C A B C D A B C D E A B C D A B C A B A	N = 5 # ## ### #### ##### #### ### ## #

```
#include<stdio.h>
int main(){
   int n, row, col;
   printf("Enter n : ");
   scanf("%d", &n);
   for(row = 1; row<= n; row++){
      for(col = 1; col <= n-row; col++){
         printf(" ");
      }
      for(col = 1; col<= row; col++){
         printf("%d", row % 2);
      }
      printf("\n");
   }
   for(row = n-1; row >= 1; row--){
      for(col = 1; col <= n-row; col++){
         printf(" ");
      }
      for(col = 1; col<= row; col++){
         printf("%d", row % 2);
      }
      printf("\n");
   }
}
```

```
/*
printf("%d", row % 2);
printf("%d", col % 2);
printf("%d", row);
printf("%d", col);
printf("*");
printf("%c", 96+row);
printf("%c", 96+col);
printf("%c", 64+row);
printf("%c", 64+col);
printf("#");
*/
```

Output:

```
    1
  2 2
 3 3 3
4 4 4 4
5 5 5 5 5
4 4 4 4
 3 3 3
  2 2
    1
```

Pattern Type - 7				
N = 5 1 1 1 1 1 0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 1 1 1 1 1	N = 5 1 0 1 0 1 1 0 1 0 1 1 0 1 0 1 1 0 1 0 1 1 0 1 0 1	N = 5 1 1 1 1 1 2 2 2 2 2 3 3 3 3 3 4 4 4 4 4 5 5 5 5 5	N = 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5	N = 5 *
N = 5 a a a a a b b b b b c c c c c d d d d d e e e e e	N = 5 a b c d e a b c d e a b c d e a b c d e a b c d e	N = 5 A A A A A B B B B B C C C C C D D D D D E E E E E	N = 5 A B C D E A B C D E A B C D E A B C D E A B C D E	N = 5 #

```c
#include<stdio.h>
int main(){
   int n, row, col;
   printf("Enter n : ");
   scanf("%d", &n);

   for(row = 1; row<= n; row++) {
      for(col = 1; col<= n; col++){
         printf("# ");
      }
      printf("\n");
   }
}
```

```c
/*
printf("%d ", row % 2);
printf("%d ", col % 2);
printf("%d ", row);
printf("%d ", col);
printf("* ");
printf("%c ", 96+row);
printf("%c ", 96+col);
printf("%c ", 64+row);
printf("%c ", 64+col);
printf("# ");
*/
```

```
# # # # #
# # # # #
# # # # #
# # # # #
# # # # #
```

Pattern Type - 8				
N = 5 1 0 0 1 1 1 0 0 0 0 1 1 1 1 1	N = 5 1 1 0 1 0 1 1 0 1 0 1 0 1 0 1	N = 5 1 2 2 3 3 3 4 4 4 4 5 5 5 5 5	N = 5 1 1 2 1 2 3 1 2 3 4 1 2 3 4 5	N = 5 * * * * * * * * * * * * * * *
N = 5 a b b c c c d d d d e e e e e	N = 5 a a b a b c a b c d a b c d e	N = 5 A B B C C C D D D D E E E E E	N = 5 A A B A B C A B C D A B C D E	N = 5 # ## ### #### #####

```
#include<stdio.h>
int main(){
   int n, row, col;
   printf("Enter n : ");
   scanf("%d", &n);

   for(row = 1; row<= n; row++) {
      for(col = 1; col <= n-row; col++){
         printf(" ");
      }
      for(col = 1; col<= row; col++){
         printf("# ");
      }
      printf("\n");
   }

}
```

```
/*
printf("%d ", row % 2);
printf("%d ", col % 2);
printf("%d ", row);
printf("%d ", col);
printf("* ");
printf("%c ", 96+row);
printf("%c ", 96+col);
printf("%c ", 64+row);
printf("%c ", 64+col);
printf("# ");
*/
```

```
    *
   * *
  * * *
 * * * *
* * * * *
```

Pattern Type - 9				
N = 5 1 1 1 1 1 0 0 0 0 1 1 1 0 0 1	N = 5 1 0 1 0 1 1 0 1 0 1 0 1 1 0 1	N = 5 5 5 5 5 5 4 4 4 4 3 3 3 2 2 1	N = 5 1 2 3 4 5 1 2 3 4 1 2 3 1 2 1	N = 5 * * * * * * * * * * * * * * *
N = 5 e e e e e d d d d c c c b b a	N = 5 a b c d e a b c d a b c a b a	N = 5 E E E E E D D D D C C C B B A	N = 5 A B C D E A B C D A B C A B A	N = 5 # # # # # # # # # # # # # # #

```
#include<stdio.h>
int main(){
   int n, row, col;
   printf("Enter n : ");
   scanf("%d", &n);

   for(row = n; row >= 1; row--){
      for(col = 1; col <= n-row; col++) {
         printf(" ");
      }
      for(col = 1; col<= row; col++) {
         printf("%c ", 64+col);
      }
      printf("\n");
   }
}
```

```
/*
printf("%d ", row % 2);
printf("%d ", col % 2);
printf("%d ", row);
printf("%d ", col);
printf("* ");
printf("%c ", 96+row);
printf("%c ", 96+col);
printf("%c ", 64+row);
printf("%c ", 64+col);
printf("# ");
*/
```

N = 5	N = 5
e e e e e	a b c d e
d d d d	a b c d
c c c	a b c
b b	a b
a	a

Pattern Type - 10				
N = 5 1 0 0 1 1 1 0 0 0 0 1 1 1 1 1 0 0 0 0 1 1 1 0 0 1	N = 5 1 1 0 1 0 1 1 0 1 0 1 0 1 0 1 1 0 1 0 1 0 1 1 0 1	N = 5 1 2 2 3 3 3 4 4 4 4 5 5 5 5 5 4 4 4 4 3 3 3 2 2 1	N = 5 1 1 2 1 2 3 1 2 3 4 1 2 3 4 5 1 2 3 4 1 2 3 1 2 1	N = 5 *
N = 5 a b b c c c d d d d e e e e e d d d d c c c b b a	N = 5 a a b a b c a b c d a b c d e a b c d a b c a b a	N = 5 A B B C C C D D D D E E E E E D D D D C C C B B A	N = 5 A A B A B C A B C D A B C D E A B C D A B C A B A	N = 5 # ## ### #### ##### #### ### ## #

```c
#include<stdio.h>
int main(){
   int n, row, col;
   printf("Enter n : ");
   scanf("%d", &n);
   for(row = 1; row<= n; row++){
      for(col = 1; col <= n-row; col++){
         printf(" ");
      }
      for(col = 1; col<= row; col++){
         printf("%c ", 64+col);
      }
      printf("\n");
   }
   for(row = n-1; row >= 1; row--){
      for(col = 1; col <= n-row; col++){
         printf(" ");
      }
      for(col = 1; col<= row; col++){
         printf("%c ", 64+col);
      }
      printf("\n");
   }
}
```

```c
/*
printf("%d ", row % 2);
printf("%d ", col % 2);
printf("%d ", row);
printf("%d ", col);
printf("* ");
printf("%c ", 96+row);
printf("%c ", 96+col);
printf("%c ", 64+row);
printf("%c ", 64+col);
printf("# ");
*/
```

N = 5	N = 5
1	1
1 0	2 2
1 0 1	3 3 3
1 0 1 0	4 4 4 4
1 0 1 0 1	5 5 5 5 5
1 0 1 0	4 4 4 4
1 0 1	3 3 3
1 0	2 2
1	1

Floyd's triangle

	C=1	C=2	C=3	C=4	C=5
R=1	1				
R=2	2	3			
R=3	4	5	6		
R=4	7	8	9	10	
R=5	11	12	13	14	15

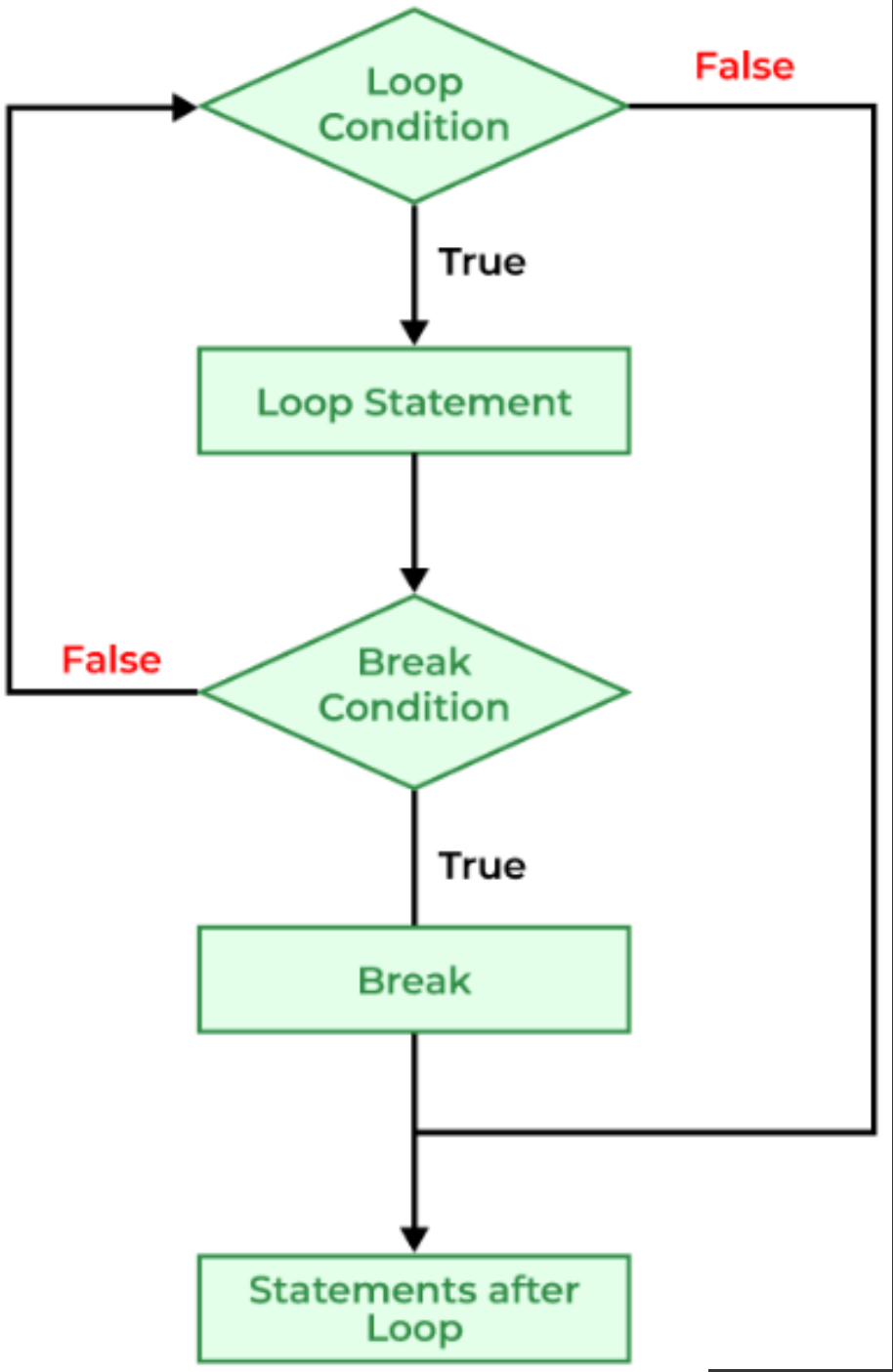
```
#include<stdio.h>
int main(){
    int n, row, col;
    printf("Enter n : ");
    scanf("%d", &n);

    int count = 1;
    for(row = 1; row<= n; row++){
        for(col = 1; col<= row; col++){
            printf("%d ", count);
            count++;
        }
        printf("\n");
    }
}
```

Jump Statements in C

These statements are used in C for the unconditional flow of control throughout the functions in a program. They support four types of jump statements:

A) break

This loop control statement is used to terminate the loop. As soon as the break statement is encountered from within a loop, the loop iterations stop there, and control returns from the loop immediately to the first statement after the loop.



Use of break in C

- Simple Loops
- Nested Loops
- Infinite Loops
- Switch case

Break in C switch case

Syntax of break in switch case

```
switch(expression)
{
case value1:
   statement_1;
   break;

case value2:
   statement_2;
   break;
.....
.....

case value_n:
   statement_n;
   break;

default:
   default statement;
}
```

// C Program to demonstrate infinite loop without using break statement

```
#include <stdio.h>
int main(){
      int i = 0;
      // while loop which will always be true
      while (1) {
            printf("%d ", i);
            i++;
            if (i == 5) {
                  break;
            }
      }
}
```

Output: 0 1 2 3 4

B) continue

This loop control statement is just like the break statement. The continue statement is opposite to that of the break statement, instead of terminating the loop, it forces to execute the next iteration of the loop. As the name suggests the continue statement forces the loop to continue or execute the next iteration. When the continue statement is executed in the loop, the code inside the loop following the continue statement will be skipped and the next iteration of the loop will begin.

```
#include <stdio.h>
int main() {
    // loop from 1 to 10
    for (int i = 1; i <= 10; i++) {
        if (i == 6)
            continue;
        else
            printf("%d ", i);
    }
}
```

Output

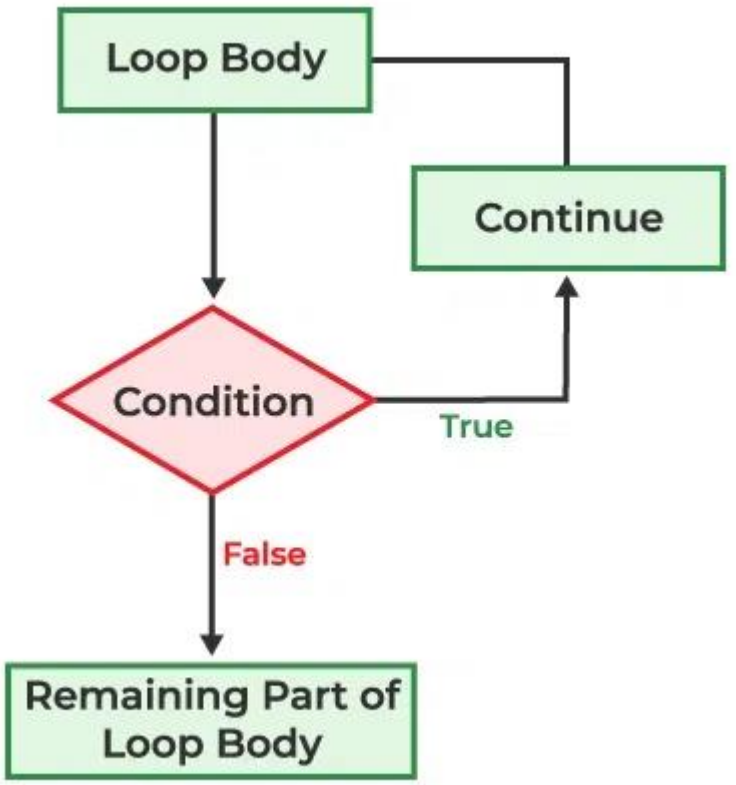
1 2 3 4 5 7 8 9 10

```
#include <stdio.h>
int main(){
    int i = 0;
    while (i < 8) {
        i++;
        if (i == 4) {
            continue;
        }
        printf("%d ", i);
    }
}
```

Output

1 2 3 5 6 7 8

Flowchart of Continue



Use of **continue** in C

1. Single Loops
2. Nested Loops

What is the difference between break and continue?

break	**continue**
The break statement terminates the loop and brings the program control out of the loop.	The continue statement terminates only the current iteration and continues with the next iterations.
The syntax is: **break;**	The syntax is: **continue;**
The break can also be used in switch case.	Continue can only be used in loops.

What is the use of continue statement in C?

The continue statement in C is used in loops to skip the current iteration and move on to the next iteration without executing the statements below the continue in the loop body.

Example: C Program to demonstrate the difference between the working of break and continue statement in C.

```
#include <stdio.h>
int main(){
    printf("The loop with break produces output as: \n");
    for (int i = 1; i <= 7; i++) {
        if (i == 3)
            break;
        else
            printf("%d ", i);
    }
    printf("\nThe loop with continue produces output as: \n");
    for (int i = 1; i <= 7; i++) {
        if (i == 3)
            continue;
        printf("%d ", i);
    }
}
```