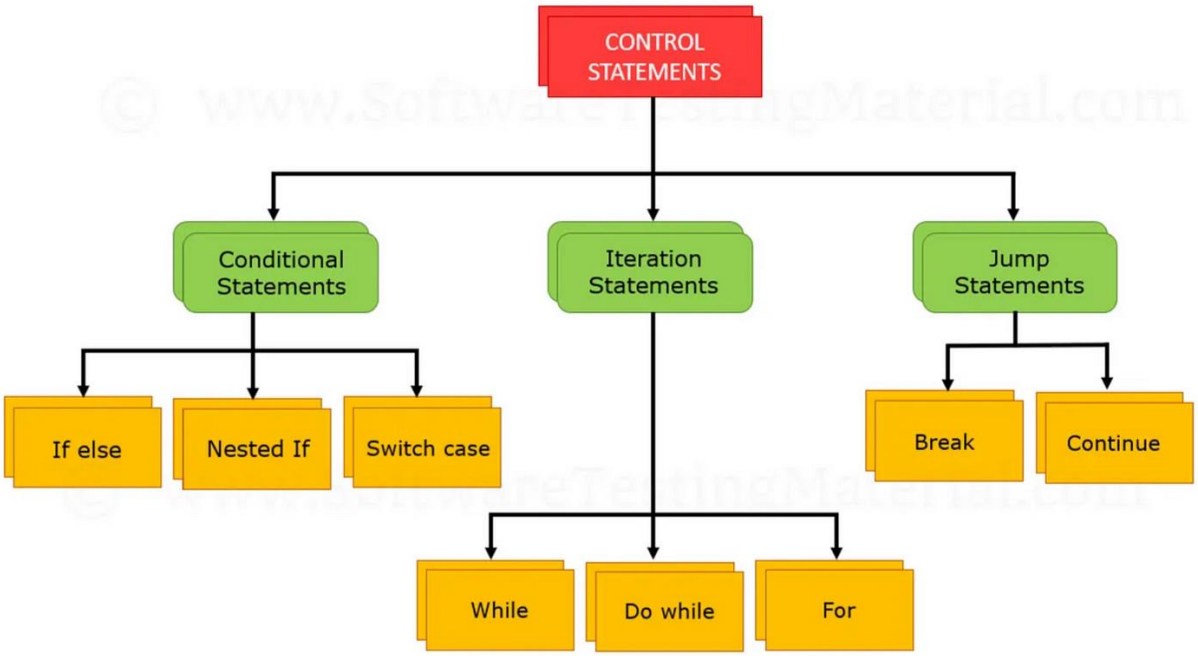
# UNIT – II

## Control Structures

**Simple sequential programs Conditional Statements (if, if-else, else if ladder, switch), Loops (for, nested for loop , while, do-while) break and continue , goto statement.**

---

**Control statements** are an essential aspect of programming languages like C, as they allow programmers to control the flow of execution of their programs. In C, there are three types of control statements: **selection statements, iteration statements, and jump statements**.

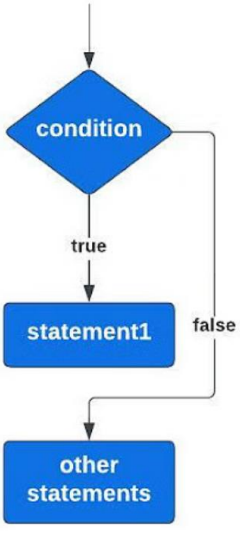



### 1. Conditional Statements or Decision-Making Control Statements:

Conditional statements help in decision-making within the program by allowing the execution of certain blocks of code depending on the result of a condition. In C programming, **conditional statements** are used to perform different actions based on whether a specified condition is true or false. The most common conditional statements in C are:

a) **Simple if:** Executes a block of code if a condition is true. If the condition of the if statement is true, then the statements under the if block is executed else the control is transferred to the statements outside the if block.

```
if (condition)
{
  Statement1;
}
```

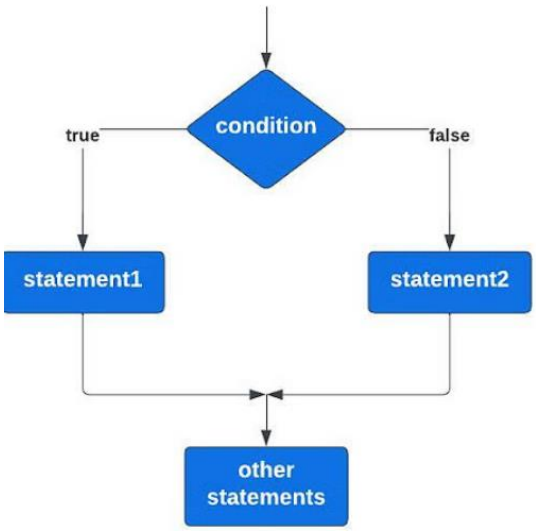


**Program2_1: Program to find the given number is positive or not.**

```
#include<stdio.h>
int main()
{
  int n;
  scanf("%d",&n);
  if(n>0)
       printf("Given number %d is a positive number\n",n);
  return 0;
}
```

**b) if else:** Executes one block of code if the condition is true, and another block if the condition is false.

```
if (condition)
  Statement1;
else
  Statement2;
```




**Program2_2: program to check if a person is eligible to vote or not based on their age.**

```
#include<stdio.h>
int main()
{
        int age;
        printf("Enter your Age:\n");
        scanf("%d",&age);
        if(age>=18)
        {
                printf("Hi! You are eligible for Voting\n");
                printf("Thank you for using my applicaiton\n");
        }
        else
        {
                printf("Sorry! You are not eligible for Voting\n");
                printf("You need to wait %d more years to get the vote\n",18-age);
        }
        return 0;
}
```

**Program2_3: Write a program that checks whether a triangle is valid or not based on the angles provided (sum of angles should be 180 degrees).**

**Input:**

45 45 90

**Output:**

Valid Triangle

**Code:**

```
#include<stdio.h>
int main()
{
	float a,b,c,sum;

	printf("Enter angles of the Triangle\n");
	scanf("%f%f%f",&a,&b,&c);
	sum=a+b+c;
	if(sum==180)
	{
		printf("It is a Valid Triangle");
	}
	else
	{
		printf("It's not a Valid Triangle");
	}
	return 0;
}
```

**Program2_4: Program to find the given character is uppercase or lowercase.**

```
#include<stdio.h>
int main()
{
	char ch;
	scanf("%c",&ch);
	//Assume the input contains only Alphabets
	if(ch>='A' && ch<='Z')
  {
	printf("%c is a Uppercase Character\n",ch);
	}
	else
	{
		printf("%c is a Lowercase Character\n",ch);
	}
	return 0;
}
```

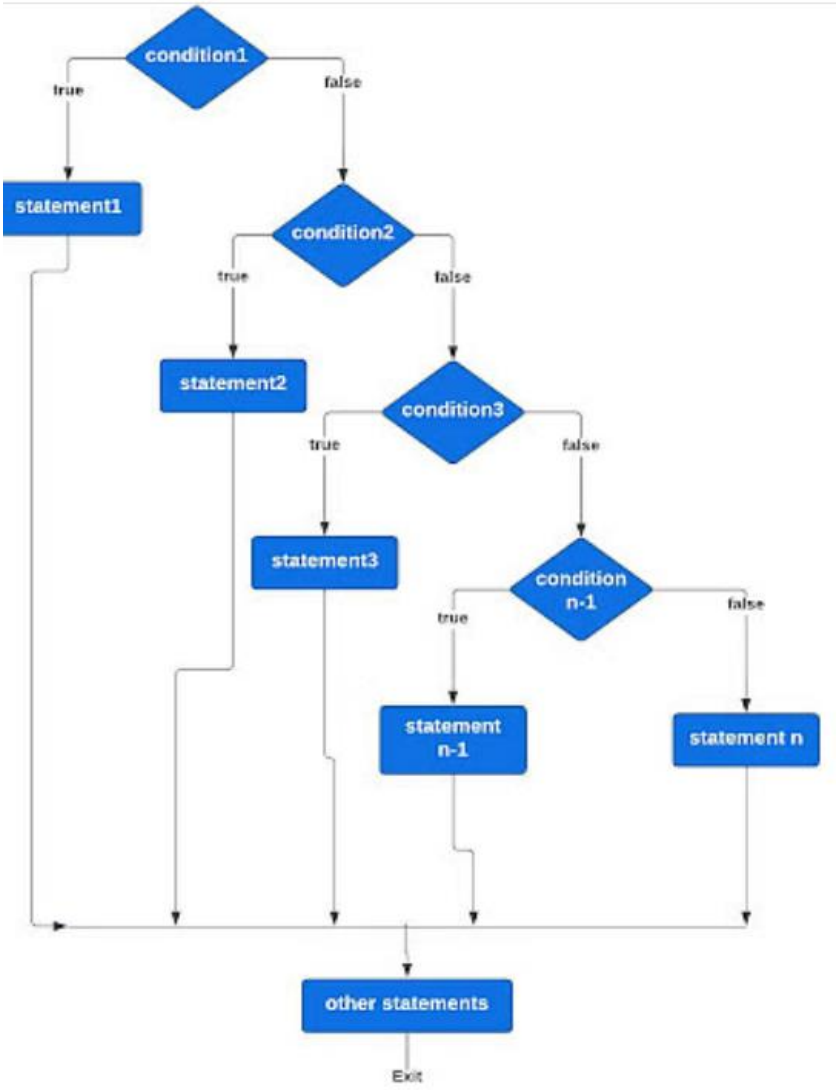
**Input:** U

**Output:** U is a Uppercase Character

**c) else if ladder:** The else-if ladder statements contain multiple else-if, when either of the condition is true the statements under that particular "if" will be executed otherwise the statements under the else block will be executed.

**Syntax:**

```
if (condition1)
{
  statement1;
}
else if (condition2)
{
  statement2;
}
else if (condition3)
{
  statement3;
}
  ------------
  ------------
else if (condition n-1)
{
  statement n-1;
}
else
{
  statement n;
}
```




**Program2_5: Program to print the color name by taking the Color code as input.**

**V -> Violet**
**I -> Indigo**
**B -> Blue**
**G -> Green**
**Y -> Yellow**
**O -> Orange**
**R -> Red**

**Input:** G
**Output:** Green

**Code:**

```
#include<stdio.h>
int main()
{
  char ch;
  scanf("%c",&ch);

  if(ch=='V' || ch=='v')
                printf("Violet\n");
  else if(ch=='I' || ch=='i')
                printf("Indigo\n");
  else if(ch=='B' || ch=='b')
                printf("Blue\n");
  else if(ch=='G' || ch=='g')
                printf("Green\n");
  else if(ch=='Y' || ch=='y')
                printf("Yellow\n");
  else if(ch=='O' || ch=='o')
                printf("Orange\n");
  else if(ch=='R' || ch=='r')
                printf("Red\n");
  else
                printf("Enter a valid color code\n");
  return 0;
}
```

**Program2_6: Program to input electricity unit charges and calculate total electricity bill according to the given condition.**

for first 50 units Rs - 0.50/unit
for next 100 units Rs - 0.75/unit
for next 100 units Rs - 1.20/unit
for units above 250 Rs - 1.50/unit
An additional surcharge of 20% is added to the bill.

Input:
150
Output:
120

**Explanation:**

150 Units => 50 X 0.50 + 100 x 0. 75 => 25 + 75 => 100
175 Units => 50 X 0.50 + 100 X 0.75 + 25 X 1.20 => 25 + 75 + 30 => 130
270 Units => 50 X 0.50 + 100 X 0.75 + 100 X 1.20 + 20 X 1.50 => ....

```
#include<stdio.h>
int main()
{
 int units;
 double bill,surcharge,tot_bill;
 scanf("%d",&units);
 if(units<=50)
 {
        bill=units*0.50;
 }
 else if(units>50 && units<=150)
 {
        bill  = 50 * 0.50 + (units-50) * 0.75;
 }
 else if(units>150 && units<=250)
 {
        bill=50*0.50 + 100 * 0.75 + (units-150) * 1.20;
 }
 else
 {
        bill=50*0.50 + 100 * 0.75 +  100*1.20 + (units-250)*1.50;
 }
 tot_bill = bill + 0.2*bill;
 printf("No of Units = %d\n",units);
 printf("Bill = %.2lf\n",bill);
 printf("Net Bill = %.2lf\n",tot_bill);
 return 0;
}
```

**Program2_7: program to find the roots of a Quadratic Equations.**

**Note:** Assume the equation is ax^2 + bx + c=0 then find the (descriminent)d =(b*b - 4*a*c)

if d = 0 => Roots are equal and print the roots.
d >0 => Roots are Real Values.
d < 0 => Roots are Imaginary.

roots = (-b +- sqrt(b^2 - 4ac))/2a

**Code:**

```
#include<stdio.h>
#include<math.h>
int main()
{
        double a,b,c;
        double d,r1,r2;
        scanf("%lf%lf%lf",&a,&b,&c);                    //Reading of input
        d=b*b-4*a*c;                                    //decriminent value
```

```
        if(d==0)
        {
                printf("Roots are Equal\n");
                r1=-b/(2*a);
                r2=-b/(2*a);
                printf("Root1 = %.2lf\n",r1);
                printf("Root2 = %.2lf\n",r2);
        }
        else if(d>0)
        {
                printf("Roots are Real Numbers\n");
                r1=(-b+sqrt(d))/(2*a);
                r2=(-b-sqrt(d))/(2*a);
                printf("Root1 = %.2lf\n",r1);
                printf("Root2 = %.2lf\n",r2);
        }
        else
        {
                printf("Roots are imaginary\n");
        }
  return 0;
}
```

**Assignment:**

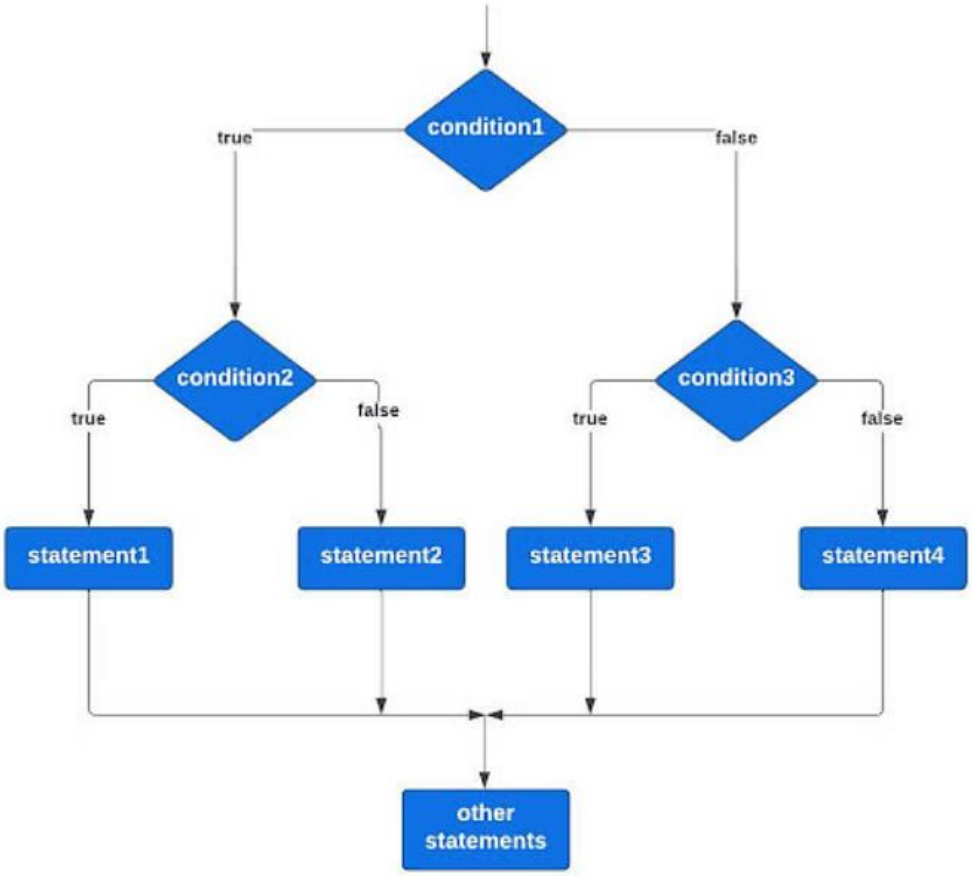
1) Write a program to read temperature in centigrade and display a suitable message according to temperature state below: [Solve]
   Temp < 0 then Freezing weather
   Temp 0-10 then Very Cold weather
   Temp 10-20 then Cold weather
   Temp 20-30 then Normal in Temp
   Temp 30-40 then Its Hot
   Temp >=40 then Its Very Hot

**Explanation:** Assume the temp=35; => Its Hot

2) Write a program to display the given digit(0 to 9) in words as follows
   Input: 9
   Output: Nine
3) Program to check whether a triangle is equilatoral, Isosceles or Scalence.
   Input: 2 3 4
   Output: Scalence
   **Hint:**
   Equilatoral -> All the sides of the Triangle are equal
   Isosceles -> Any two sides of the Triangle are equal
   Scalence -> All the sides of the Triangle are different

**d) Nested if:** A **nested if** statement in C is an **if** statement placed inside another **if** statement. It allows you to check multiple conditions in a sequence. If the outer **if** condition is true, only then will the inner **if** condition be evaluated.

```
if (condition1)
{
//if block of outer if's
  if (condition2)
  {
    statement1;
  }
  else
  {
    Statement2;
  }
}
//else block of outer if's
else
{
  if (condition3)
  {
    statement3;  //inner if
  }
  else
  {
    Statement4;  //else block of inner if
  }
}
```



**Program-2.8: Program to find the biggest of three numbers.**

```
#include<stdio.h>
int main()
{
        int a,b,c;
        scanf("%d%d%d",&a,&b,&c);
        if(a>b)
        {
                if(a>c)
                 {
                        printf("%d is bigger",a);
                 }
                else
                {
                        printf("%d is bigger",c);
                }
        }
        else
        {
                if(b>c)
                {
                        printf("%d is bigger",b);
                }
                else
                {
                        printf("%d is bigger",c);
                }
        }
        return 0;
}
```

**Program-2.9: Program to check whether a year is a leap year using nested if conditions.**

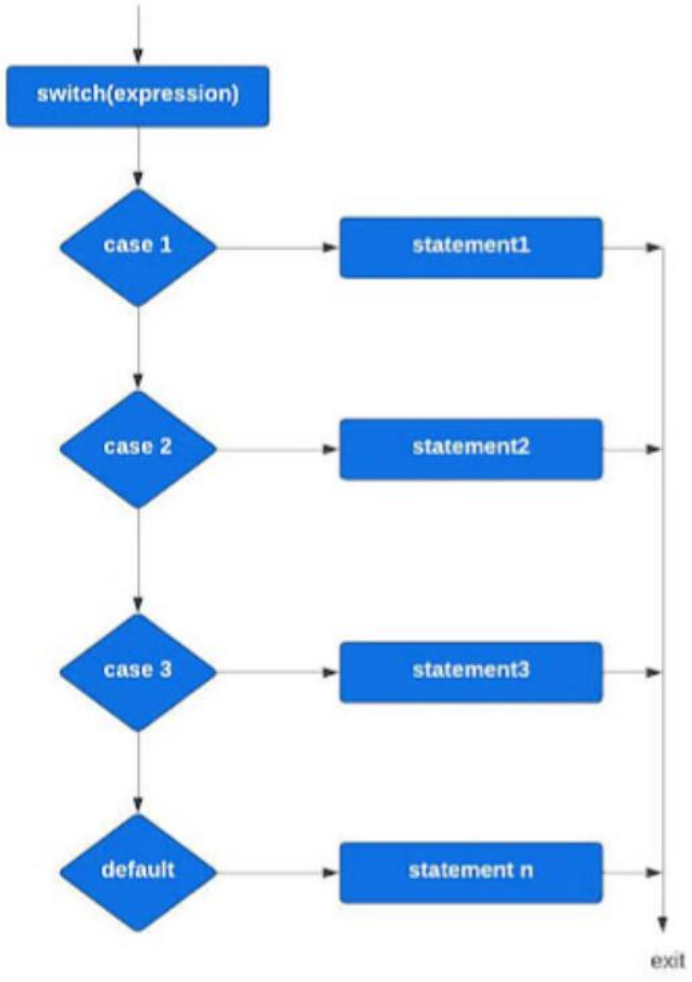
```
#include <stdio.h>
int main()
{
   int year;
   printf("Enter a year: ");
   scanf("%d", &year);
   if (year % 4 == 0)
  {
      if (year % 100 == 0)
      {
         if (year % 400 == 0)
         {
                printf("%d is a leap year.\n", year);
         }
         else
         {
       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

```
    else
    {
            printf("%d is a leap year.\n", year);
    }
  }
  else
  {
            printf("%d is not a leap year.\n", year);
  }
  return 0;
}
```

**e) switch:** The switch statement in C is used to perform different actions based on different conditions. It's an alternative to using multiple if-else statements when you have many conditions based on the value of a single variable. The switch evaluates the variable and executes the corresponding block of code that matches the value.

**Syntax:**

```
Switch (expression)
{
  Case label 1:   statement1;
                  ----------
                  break;
  Case label 2:   statement2;
                  ----------
                  break;
  Case label 3:   statement3;
                  ----------
                  break;
-------------------------------------------------
-------------------------------------------------
  Case label n:   statement n;
                  ----------
                  break;
      default:    default statement;
                  ----------
}
Other statements;
```




**How It Works:**

1) The expression inside the switch is evaluated.
2) The value of the expression is compared with the constants provided in each case.
3) If a match is found, the code block associated with that case is executed.
4) The break statement ends the switch statement. If break is not used, the execution will continue to the next case (known as fall-through behavior).
5) If no matching case is found, the default block (if present) is executed.

**Rules Followed in Switch Case:**

1) Only Integer and Character Values Allowed:
2) The expression in a switch must evaluate to an integer or character value. Floating-point and strings are not allowed.
3) Case Labels Must Be Constant and Unique:
4) Variables are not allowed in case labels.
5) **Break Statement:** The break statement is used to exit the switch once a matching case is executed. Without break, the code will continue executing the next case even if the value does not match (fall-through).
6) **Default Case:** The default case is optional and is executed when none of the other case values match. It acts like a catch-all for unhandled values.
7) **Fall-Through Behavior:** In C, if there is no break after a case, the program continues executing the next case. This is known as fall-through behavior and can be used intentionally if multiple cases share the same code block.

**Program-2.10: Program to print the week day name based on the day number.**

Assumption: 1 - sunday , 2 - Monday, .....,7- Saturday

**Input:** 2

**Output:** Monday

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

```
                break;
        default:
                printf("Enter a valid week day number(1-7)");
                break;
    }
    printf("Task Completed");
    return 0;
}
```

**Program-2.11: Traffic Light System:**

Write a program to simulate a traffic light system using if-else if. Depending on the light color input (Red, Yellow, Green), display the action for the driver (Stop, Slow down, Go).

```
#include<stdio.h>
int main()
{
    char color_Code;
    scanf("%c",&color_Code);
    switch(color_Code)
    {
        case 'R':
        case 'r':
                printf("STOP.....");
                break;
        case 'Y':
        case 'y':
                printf("SLOW DOWN.....");
                break;
        case 'G':
        case 'g':
                printf("GO.....");
                break;
        default:
                printf("Enter Valid Color Code");
                break;
    }
    return 0;
}
```

**Program-2.12: Menu-Driven Program for Temperature Conversion:**

Write a program that provides a menu to convert temperature between Celsius, Fahrenheit, and Kelvin. The user should choose from a menu (1 for Celsius to Fahrenheit, 2 for Fahrenheit to Celsius, 3 for Celsius to Kelvin, etc.), and the program should perform the appropriate conversion using switch case.

**Input:**

1. Celsius to Fahrenheit
2. Fahrenheit to Celsius
3. Celsius to Kelvin

Enter you Choice:
2
Enter the Fahrenheit Value
34

**Output:**

Print Celsius converted Value.

**Code:**

```
#include<stdio.h>
int main()
{
	double C,F,K;
	int option;
	printf("1. Celsius to Fahrenheit\n2. Fahrenheit to Celsius\n3. Celsius to Kelvin\n");
	printf("Enter your choice(1 - 3)\n");
	scanf("%d",&option);
	switch(option)
	{
		case 1:
			printf("Enter the Celsius Value\n");
			scanf("%lf",&C);
			// F = (C * 9/5)+32
			F=(double)(C*9)/5 + 32;
			printf("C = %.2lf and F = %.2lf\n",C,F);
			break;
		case 2:
			printf("Enter the Fahrenheit value\n");
			scanf("%lf",&F);
			C = (F-32) * 5.0/9;
			printf("F = %.2lf and C = %.2lf\n",F,C);
			break;
		case 3:
			printf("Enter the Celsius value\n");
			scanf("%lf",&C);
			K=C+273.15;
			printf("C = %.2lf and K = %.2lf",C,K);
			break;
		default:
			printf("Enter a valid Option\n");
			break;
	}
   return 0;
}
```

**Program-2.13: Program to implement simple calculator by using switch case.**

**Input:**

Enter any two Numbers
20 10
1) + - Addition
2) - - Subtraction
3) * - Multiplication
4) / - Division
5) % - Modulous
Enter your Choice
+

**Output:**

Sum of 20 and 10 is: 30

**Code:**

```
#include<stdio.h>
int  main()
{
        int n1,n2,res;
        char op;
        printf("+ - Addition\n- - Subtraction\n* - Multiplication\n/ - Division\n%% - Modulus\n");
        printf("Enter your Choice\n");
        scanf("%c",&op);
        printf("Enter any two numbers\n");
        scanf("%d%d",&n1,&n2);
        switch(op)
        {
                case '+':       res=n1+n2;
                                break;
                case '-':       res=n1-n2;
                                break;
                case '*':       res=n1*n2;
                                break;
                case '/':       if(n2!=0)
                                {
                                        res=n1/n2;
                                }
                                else
                                {
                                        printf("Division is not possible");
                                        res=0;
                                }
                                break;
                case '%':       if(n2!=0)
                                {
                                        res=n1%n2;
```

```
                        }
                        else
                        {
                                printf("Modulus is not possible");
                                res=0;
                        }
                        break;
                default:        printf("Enter a valid Operator\n");
                        break;
        }
        printf("Result = %d\n",res);
        return 0;
}
```

## 2) Iterative or Looping Statements:

Looping statements in C are used to repeat a block of code multiple times until a specified condition is met. These loops help reduce code redundancy and make programs more efficient.

Types of Looping Statements:

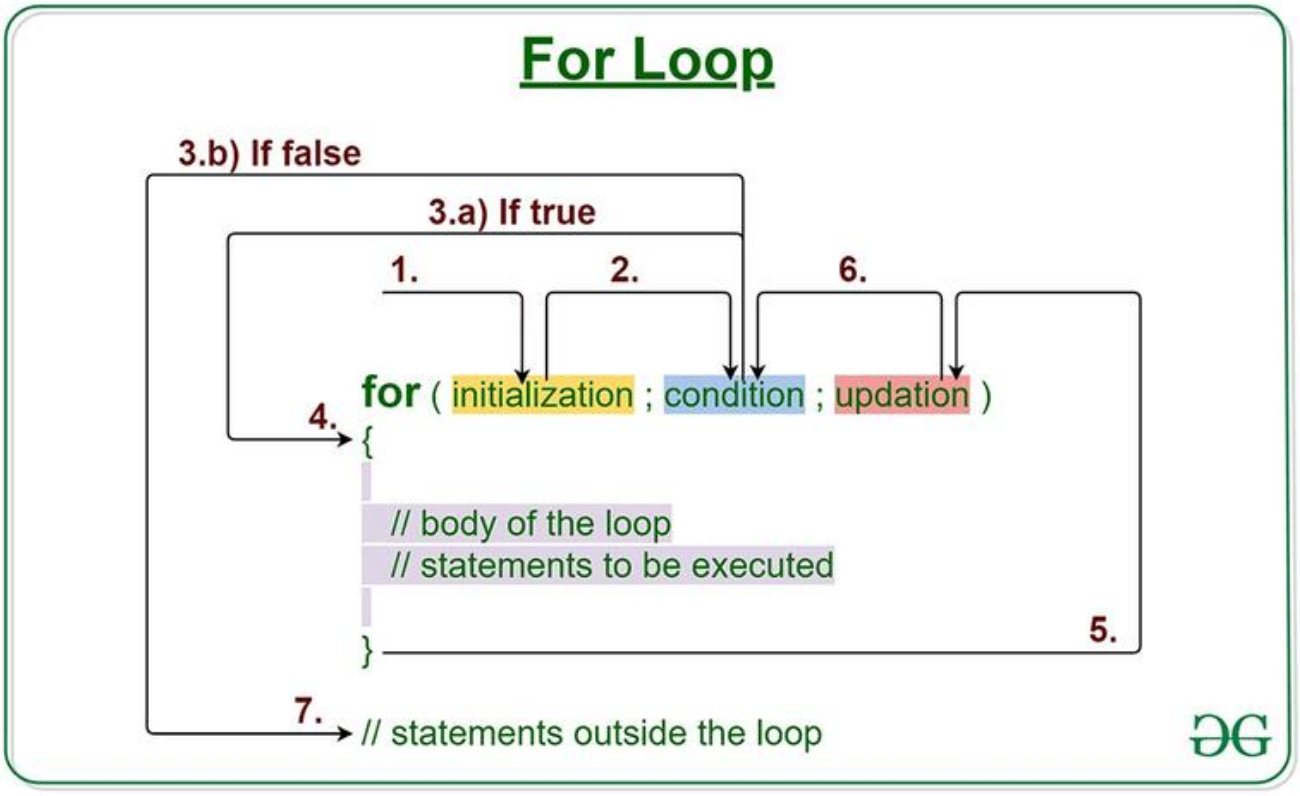
### a) for loop:

In C programming, a for loop repeats a block of code a specific number of times. It consists of three parts:

**Initialization:** Sets the loop control variable (e.g., int i = 0).

**Condition:** The loop runs as long as this condition is true (e.g., i < 5).

**Increment/Decrement:** Updates the loop control variable after each iteration (e.g., i++)




The for loop follows a very structured approach where it begins with initializing a condition then checks the condition and in the end, executes conditional statements followed by an updation of values.

**Initialization:**
This step initializes a loop control variable with an initial value that helps to progress the loop or helps in checking the condition. It acts as the index value when iterating an array or string.
**Check/Test Condition:**
This step of the for loop defines the condition that determines whether the loop should continue executing or not. The condition is checked before each iteration and if it is true then the iteration of the loop continues otherwise the loop is terminated.
**Body:**
It is the set of statements i.e. variables, functions, etc that is executed repeatedly till the condition is true. It is enclosed within curly braces { }.
**Updation:**
This specifies how the loop control variable should be updated after each iteration of the loop. Generally, it is the incrementation (variable++) or decrementation (variable--) of the loop control variable.

**Program-2.14: Program to print all the numbers from 1 to 100.**

Code:

```
#include<stdio.h>
int main()
{
	int i;
	for(i=1;i<=100;i++)
	{
		printf("%d ",i);
	}
	return 0;
}
```

**Program-2.15: Program to find sum of n natural numbers.**
**Input:** 5
**Output:** 15
**Explanation:** sum= 1 + 2 + 3 + 4 + 5 => 15
**Code:**

```
#include<stdio.h>
int main()
{
	int n,sum=0;
	scanf("%d",&n);
	int i;
	for(i=1;i<=n;i++)
	{
		sum=sum+i;
	}
	printf("Sum = %d",sum);
	return 0;
}
```

**Program-2.16: Program to find the factorial of a given number.**

Assume:

n=5 then factorial=5*4*3*2*1

**Code:**

```
#include<stdio.h>
int main()
{
	int n,prod=1;
	scanf("%d",&n);
	int i;
	for(i=n;i>=1;i--)
	{
		prod=prod*i;
	}
	printf("Factorial of %d is %d",n,prod);
	return 0;
}
```

**Program- 2.17: Program to find the value for the following expression.**

Assume:

n=5, x=2 then find the expression 1 + x^1 + x^2 + x^3 + x^4 +.....+ x^n

**Code:**

```
#include<stdio.h>
#include<math.h>
int main()
{
	int n,x;
	int sum=1;
	scanf("%d%d",&x,&n);
	int i;
	for(i=1;i<=n;i++)
	{
		sum=sum+(int)pow(x,i);
	}
	printf("%d",sum);
	return 0;
}
```

**Program-2.18: Program to find the given number is prime or not.**

**Code:**

```
#include<stdio.h>
int main()
{
	int n,count=0,i;
	scanf("%d",&n);
```

```
        for(i=1;i<=n;i++)
        {
                if(n%i==0)
                   count++;
        }
        if(count==2)
           printf("%d is a prime number",n);
        else
           printf("%d is not a prime number",n);

        return 0;
}
```

**Input:** 29
**Output:** 29 is a prime number

**Program-2.19: Program to find the prime numbers within the range of given inputs.**
**Code:**

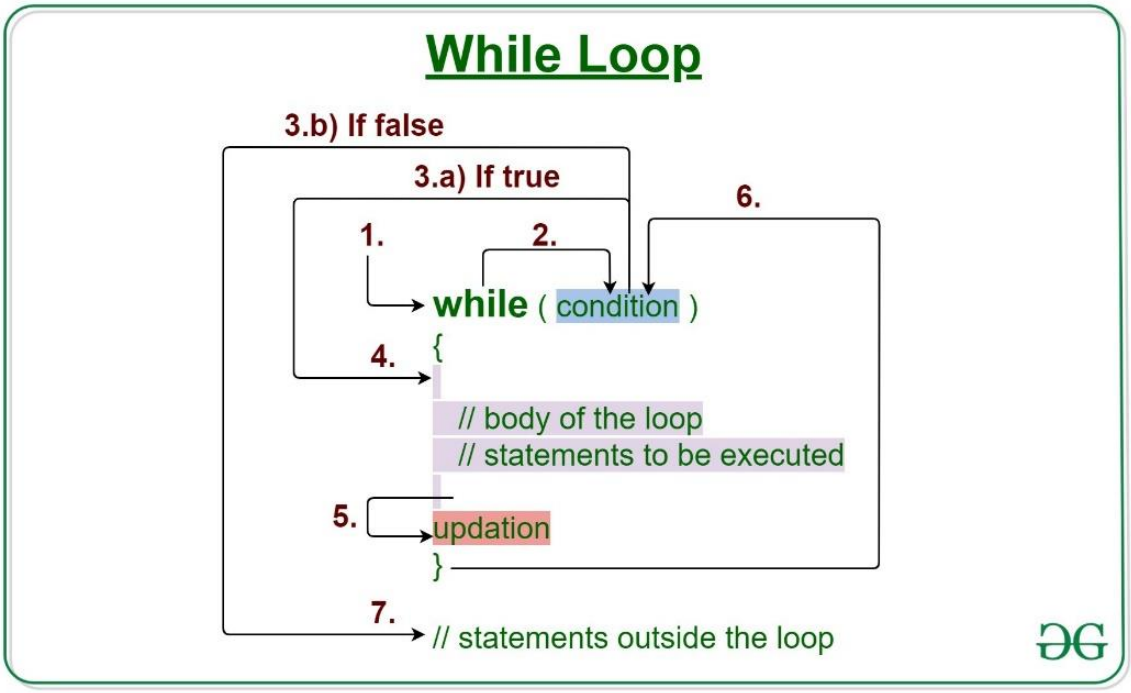
```
#include<stdio.h>
int main()
{
        int m,n,count=0,i,j;
        scanf("%d%d",&m,&n);
        for(i=m;i<=n;i++)
        {
                count=0;
                for(j=1;j<=i;j++)
                {
                        if(i%j==0)
                   count++;
                }
                if(count==2)
                  printf("%d ",i);

        }
        return 0;
}
```

**Input:** 1 100
**Output:** 2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97

## b) while Loop:

The while Loop is an entry-controlled loop in C programming language. This loop can be used to iterate a part of code while the given condition remains true.




**1) Initialization:**

In this step, we initialize the loop variable to some initial value. Initialization is not part of while loop syntax, but it is essential when we are using some variable in the test expression

**2) Conditional Statement:**

This is one of the most crucial steps as it decides whether the block in the while loop code will execute. The while loop body will be executed if and only the test condition defined in the conditional statement is true.

**3) Body:**

It is the actual set of statements that will be executed till the specified condition is true. It is generally enclosed inside { } braces.

**4) Updation:**

It is an expression that updates the value of the loop variable in each iteration. It is also not part of the syntax but we have to define it explicitly in the body of the loop.

**Program-2.20: Program to print the numbers from 1 to 10.**

```
#include<stdio.h>
int main()
{
        int i=1;                    // Initialization
        while(i<=10)                // Condition
        {
                printf("%d ",i);    // Body of the loop
                i++;                // Updation
        }
        printf("Task Completed");
        return 0;
}
```

**Program-2.21: Program to print nth table up to 12 rows.**

```
#include<stdio.h>
int main()
{
	int n;
	scanf("%d",&n);
	int i=1;
	while(i<=10)
	{
		int a=i*n;
		printf("%d x %d = %d\n",n,i,a);
		i++;
	}
	printf("Task Completed");
	return 0;
}
```

**Program-2.22: Program to find the number of digits of a give number.**

```
#include<stdio.h>
#include<math.h>
int main()
{
	int n;
	scanf("%d",&n);
	int digits= (int)log10(n)+1;
	printf("No of digits of a given Number %d is: %d",n,digits);
	return 0;
}
```

**Input:** 5342
**Output**: 4

**Program-2.23: Program to find the sum of digits of a given number**
**Input:** 234
**Output:** 9
**Explanation:** sum= 2 + 3 + 4 => 9
**Code:**

```
#include<stdio.h>
int sumOfDigits(int);   // function prototype
int main()
{
	int n;
	scanf("%d",&n);
	int sum=sumOfDigits(n);
	printf("Sum of Digits of a given number %d is: %d",n,sum);
	return 0;
}
```

```
int sumOfDigits(int n)  // function definition
{
        int sum=0;
        while(n>0)
        {
                int rem=n%10;
                sum=sum+rem;
                n=n/10;
        }
        return sum;
}
```

**Program-2.24: Program to find the reverse of a given number.**
**Input:** 123
**Output:**321
**Code:**

```
#include<stdio.h>
int reverse(int);                    // function prototype
int main()
{
        int n;
        scanf("%d",&n);
        int rev=reverse(n);
        printf("%d",rev);
        return 0;
}
int reverse(int n)                   //function  definition
{
        int rem,sum=0;
        while(n>0)
        {
                rem=n%10;
                sum=sum*10+rem;
                n=n/10;
        }
        return sum;
}
```

**Program-2.25: Program to find the given number is palindrome or not.**
**Palindrome:** Reverse of the given number is equal to given number itself is called Palindrome
Input: 323
Output: YES
Input: 123
Output: NO

**Code:**

```
#include<stdio.h>
int reverse(int);
int main()
{
	int n;
	scanf("%d",&n);
	int rev=reverse(n);
	printf("Reverse Number = %d\n",rev);
	if(n==rev)
	{
		printf("PALINDROME");
	}
	else
	{
		printf("NOT A PALINDROME");
	}
	return 0;
}
int reverse(int n)
{
	int rem,sum=0;
	while(n>0)
	{
		rem=n%10;
		sum=sum*10+rem;
		n=n/10;
	}
	return sum;
}
```

**Program-2.26: Program to check whether a given number is Armstrong number or not.**

**Armstrong Number:** A number is thought of as an Armstrong number if the sum of its own digits raised to the power number of digits gives the number itself.

Input: 153

Output: YES

**Explanation**: 1^3+5^3+3^3 = 1+125+27 = 153

Input: 1634

Output: YES

**Explanation:** 1^4 + 6^4 + 3^4 + 4^4 => 1634

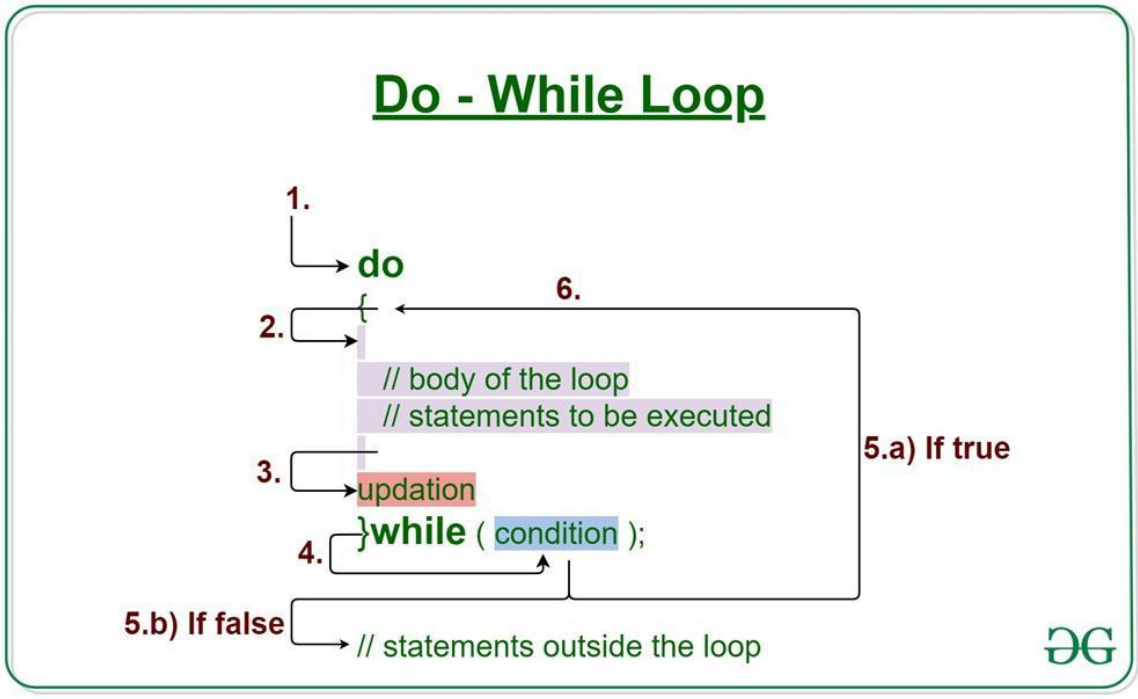
**Code:**

```
#include<stdio.h>
#include<math.h>
int findArmStrongCalculation(int);
int main()
{
```

```
        int n,res;
        scanf("%d",&n);
        res=findArmStrongCalculation(n);
        if(n==res)
                printf("Given Number %d is Armstrong Number",n);
        else
                printf("Given Number %d is Not a Armstrong Number",n);
        return 0;
}
int findArmStrongCalculation(int n)
{
        int digits=(int)log10(n)+1;
        int rem,sum=0;
        while(n>0)
        {
                rem=n%10;
                sum=sum+(int)pow(rem,digits);
                n=n/10;
        }
        return sum;
}
```

### c) do while loop

In C programming, a do-while loop is a type of loop that ensures the body of the loop is executed at least once, regardless of whether the loop condition is true or false. The condition is checked after the loop body is executed. This is also called **exit control loop**.




**Key Points:**

**Execution at least once:**

The loop body will always execute at least once, even if the condition is initially false.

**Condition check:**
After executing the loop body, the condition is checked. If it's true, the loop continues to execute; if false, the loop terminates.
**Semicolon:** The while condition in a do-while loop ends with a semicolon.
**Program-2.27: Program to print the numbers from 1 to n using do while loop.**

```
#include<stdio.h>
int main()
{
        int n;
        scanf("%d",&n);
        int i=1;                        // initialization
        do
        {                               // body of the loop
                printf("%d ",i);
                i++;                    // updation
        }while(i<=n);                   // condition
        printf("Task completed");
}
```

**Program-2.28: Program to implement the simple calculator using do while loop?**
**Expected Output:**

```
1 - Addition
2 - Subtraction
3 - Multiplication
4 - Division
5 - Modulous
Enter your choice
1
Enter any two numbers
10 20
Sum = 30
Do you want to continue...(Y/N)?
Y
1 - Addition
2 - Subtraction
3 - Multiplication
4 - Division
5 - Modulous
Enter your choice
```

**Code:**

```
#include<stdio.h>
int main()
{
        int option,num1,num2,result;
        char choice;
```

```
        do
        {
                printf("1. Addition\n2. Subtraction\n3. Mulitiplication\n4. Division\n5. Mod\n");
                printf("Enter your Option(1-5)\n");
                scanf("%d",&option);
                printf("Enter any two numbers\n");
                scanf("%d%d",&num1,&num2);
                switch(option)
                {
                        case 1:         result=num1+num2;
                                        printf("Sum = %d\n",result);
                                        break;
                        case 2:         result=num1-num2;
                                        printf("Diff = %d\n",result);
                                        break;
                        case 3:         result=num1*num2;
                                        printf("Product = %d\n",result);
                                        break;
                        case 4:         if(num2!=0)
                                        {
                                          result=num1/num2;
                                          printf("Division = %d\n",result);
                                        }
                                        else
                                          printf("Division is not Possible\n");
                                        break;
                        case 5:         if(num2!=0)
                                        {
                                                result=num1%num2;
                                                printf("Mod = %d\n",result);
                                        }
                                        else
                                          printf("Mod is not possible\n");
                                        break;
                        default:        printf("Enter a valid optioin\n");
                                        break;
                }
                fflush(stdin);
                printf("Do you want to continue...(Y/N)?\n");
                scanf("%c",&choice);
        }while(choice=='Y' || choice=='y');
        printf("Thank you for using my simple Calculator\n");
        return 0;
}
```

## Working with break and continue:

### Break:

In C programming, the break statement is used to terminate the execution of a loop or a switch statement prematurely.

The break statement can be used inside any type of loop (for, while, or do-while) to immediately exit the loop, regardless of the loop condition. Once a break is encountered, the control is transferred to the first statement following the loop.

**Program-2.29: Program to demonstrate the use of break statement**.

```
#include <stdio.h>
int main()
{
   for (int i = 1; i <= 10; i++)
   {
         if (i == 5)
         {
                  break;          // Exit the loop when i equals 5
         }
         printf("%d ", i);
   }
   return 0;
}
```

**Output:** 1 2 3 4

### Continue:

In C programming, the **continue** statement is used within loops to skip the current iteration and proceed with the next iteration of the loop. Unlike the **break statement**, which exits the loop entirely, continue allows the loop to continue but **skips the rest of the code in the current iteration**.

**Program-2.30: Program to demonstrate the use of continue statement.**

```
#include <stdio.h>
int main()
{
   for (int i = 1; i <= 5; i++)
   {
      if (i == 3)
      {
         continue;          // Skip the iteration when i is 3
      }
      printf("%d ", i);
   }
   return 0;
}
```

**Output:** 1 2 4 5

## Working with goto Statement:

The **goto** statement in C provides an unconditional jump to a labeled statement within the same function. It can disrupt the normal flow of execution by jumping directly to another part of the code, usually specified by a label.

**Syntax:**

```
goto label;
...
label:
    // Code to be executed when the goto jumps here
```

**Program-2.31: Program to demonstrate the use of goto statement.**

```
#include <stdio.h>
int main()
{
   int i = 0;
   loop:                          // label definition
      printf("i = %d\n", i);
      i++;
      if (i < 5)
     {
         goto loop;               // jump back to the label 'loop'
      }
   printf("Exited loop\n");
   return 0;
}
```

**Output:**

```
i = 0
i = 1
i = 2
i = 3
i = 4
Exited loop
```

## Nested Loops:

One loop is written inside the another loop is called nested loops. In C, this allows for repeated execution of a block of code within another loop, meaning the inner loop will be executed multiple times for each iteration of the outer loop.

**Syntax:**

```
for(initailization;condition;incr/decr)          ---> Outer Loop
{
    for(initailization;condition;incr/decr)      ---> Inner Loop
    {
        .......
       //inner block statements
    }
    //Outer block statements
}
```

**Program-2.32: Program to demonstrate the use of nested loops**

**Code:**

```
#include<stdio.h>
int main()
{
        int i,j;
        for(i=1;i<=3;i++)
        {
                for(j=1;j<=3;j++)
                {
                        printf("i -> %d and j -> %d\n",i,j);
                }
                printf("Outer Loop Iteration - %d completed\n",i);
        }
   return 0;
}
```

Output:

```
i -> 1 and j -> 1
i -> 1 and j -> 2
i -> 1 and j -> 3
Outer Loop Iteration - 1 completed
i -> 2 and j -> 1
i -> 2 and j -> 2
i -> 2 and j -> 3
Outer Loop Iteration - 2 completed
i -> 3 and j -> 1
i -> 3 and j -> 2
i -> 3 and j -> 3
Outer Loop Iteration - 3 completed
```

**Patterns:**

**Program-2.33: Program to print the following pattern. [Pattern-1]**

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

Output:

```
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
```

**Program-2.34: Program to print the following pattern. [Pattern-2]**

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

```
Output:
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
```

**Program-2.35: Program to print the following pattern. [Pattern-3]**

```
#include<stdio.h>
int main()
{
    int i,j;
    for(i=1;i<=5;i++)
    {
        for(j=1;j<=5;j++)
        {
            printf("%c ",(char)(64+j));
        }
        printf("\n");
    }
    return 0;
}
```

```
Output:
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
```

**Program-2.36: Program to print the following pattern. [Pattern-4]**

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

```
Output:
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
```

**Program-2.37: Program to print the following pattern. [Pattern-5]**

```
#include<stdio.h>
int main()
{
    int i,j,n;
    scanf("%d",&n);
    for(i=1;i<=n;i++)
    {
        for(j=n;j>=1;j--)
        {
            if(j>i)
              printf("  ");
            else
              printf("* ");
        }
    //control move to the next line
        printf("\n");
    }
    return 0;
}
```

```
Input: 5
Ouput:
        *
      * *
    * * *
  * * * *
* * * * *
```

**Program-2.38: Program to print the following pattern. [Pattern-6]**

```
#include<stdio.h>
int main()
{
    int i,j,n;
    scanf("%d",&n);
    for(i=1;i<=n;i++)
    {
        for(j=2*n-1;j>=1;j--)
        {
            if(j>2*i-1)
              printf("  ");
            else
             printf("* ");
        }
        printf("\n");
    }
    return 0;
}
```

```
Input: 5
Output:
        *
      * * *
    * * * * *
  * * * * * * *
* * * * * * * * *
```

**Program-2.39: Program to print the following pattern. [Pattern-7]**

```
#include <stdio.h>
int main()
{
    int rows, i, j, space;
    printf("Enter the number of rows: ");
    scanf("%d", &rows);
    for (i = 1; i <= rows; i++)
    {
        // Print leading spaces
        for (space = 1; space <= rows - i; space++)
        {
            printf(" ");
        }
        // Print asterisks for the pyramid
        for (j = 1; j <= (2 * i - 1); j++)
        {
            printf("*");
        }
        printf("\n");
    }
    return 0;
}
```

```
Input: 5
Output:
    *
   ***
  *****
 *******
*********
```

**Important Questions:**

1) Define Control Statements. Explain the different types of control statements with neat diagram.
2) Explain the conditional statements with syntax and an example of each.
3) Demonstrate the use of switch case with neat example. Write the rules of switch case.
4) List and explain the different types of looping statements with neat examples.
5) List the differences between entry control loop and exit control looping statements.
6) Explain break, continue and goto statements with syntax and an example of each.
7) Define nested loops. Explain the use of nested loops with an example program.
8) Compare and contrast for, while and do while loop.

**Programs to practice:**

1) Program to find the factorial of a given number.
2) Program to find the given number is prime or not.
3) Program to print the list of prime numbers between the given range.
4) Program to print the Fibonacci series up to given range.
5) Program to find the reverse of a given number.
6) Program to find the given number is palindrome or not.
7) Program to find the given number is Armstrong number or not.
8) Program to find the implement simple calculator using switch case. etc…

**Important MCQ Question and answers:**

**1) Which of the following can be used to control the flow of execution in a program?**
a) if, else, for, while b) switch, continue, break, goto **c) Both a and b** d) None of the above

**2) Which of the following is a control statement in C?**
a) printf b) scanf **c) for** d) return

**3) What is the purpose of the continue statement in a loop?**
a) Exit the loop **b) Skip the current iteration** c) Restart the loop d) Exit the program

**4) Which statement is used to exit a loop in C?**
a) exit b) continue **c) break** d) goto

**5) In which loop is the condition checked after executing the loop body?**
a) for loop b) while loop **c) do-while loop** d) if statement

**6) Which of the following is not a loop control structure in C?**
a) for b) do-while **c) switch** d) while

**7) What happens if the condition in a while loop is initially false?**
**a) The loop is skipped entirely** b) The loop runs once and stops
c) The loop runs infinitely d) The program crashes

**8) How many times is the body of a do-while loop guaranteed to execute?**
a) 0 times **b) At least once** c) Twice d) Depends on the condition

**9) What is the output of the following code?**

```
int i = 0;
while (i < 3) {
   printf("%d", i);
   i++;
}
```

**a) 012** b) 123 c) 001 d) 111

**10) Which of the following is true about the switch statement?**
a) It evaluates multiple conditions b) It supports only integers and characters
**c) It executes multiple cases if no break is present** d) It can handle floating-point numbers

**11) What is the output of the following code?**

```
int i = 0;
do {
        printf("%d ", i);
        i++;
} while (i < 3);
```

a) 0 1 **b) 0 1 2** c) 0 1 2 3 d) None of the above

**12) What is the default value of the condition in a while loop if the condition is omitted?**
a) false b) true c) Undefined **d) Syntax error**

**13) What happens if there is no break in a switch case?**
a) The program will throw an error **b) The next case will be executed**
c) Only the current case is executed d) The program crashes

**14) Which of the following is true for an infinite loop?**
**a) It has no exit condition** b) It contains a break statement
c) It always runs exactly once d) It ends when continue is used

**15) How many times will the following loop execute?**

```
int i = 10;
```

```
while (i < 5)
{
   printf("Hello");
}
```

**a) 0 times** b) Infinite times c) 1 time d) 5 times

**16) What is the purpose of the else part in an if-else statement?**

a) To execute when the if condition is true **b) To execute when the if condition is false**

c) To exit the program d) To repeat the loop

**17) Which loop structure allows you to initialize, test, and update in a single line?**

a) while b) do-while **c) for** d) goto

**18) What is the primary purpose of a break statement inside a loop?**

a) Restart the loop b) Skip the current iteration

**c) Terminate the loop** d) Return to the beginning of the program

**19) What will be the output of the following code?**

```
int i = 0;
for (; i < 3; i++) {
   if (i == 1) {
      continue;
     }
   printf("%d", i);
}
```

a) 01 b) 012 **c) 02** d) 11

**20) Which loop structure is best suited when the number of iterations is known?**

a) while b) do-while **c) for** d) goto

**21) What happens if the condition in a for loop is omitted?**

a) The loop will not execute **b) The loop will execute infinitely**

c) The loop will cause a syntax error d) The loop will execute only once

**22) Which of the following is true for a switch statement?**

**a) It allows fall-through by default.** b) It does not require break statements.

c) It evaluates floating-point numbers. d) Only the matched case is executed, even without break.

**23) Which statement can be used to jump to another section of code?**

a) continue b) break **c) goto** d) switch

**24) Which statement will immediately exit the innermost loop?**

a) exit **b) break** c) continue d) return

**25) What will be the output of the following code?**

```
int i = 1;
for (i = 1; i <= 5; i++) {
   if (i == 3) {
      break;
   }
   printf("%d ", i);
}
```

**a) 1 2** b) 1 2 3 c) 3 4 d) 1