UNIT - 5
Control Structures

1. Control Structure

Control structure enables us to specify the order in which the various instructions in a program are to be executed by the computer. In other words, the control instructions determine the "flow of control" in a program. There are 3 types of control instructions in C. They are:

1.1 Sequence Control Instruction
1.2 Selection or Decision Control Instruction
1.3 Repetition or Loop Control Instruction

1.1 Sequential Control Instruction:

The sequential structure consists of a sequence of program statements that are executed one after another in order. In this structure, each statement is executed exactly once. In this, the input is received, processed and output is generated. No any condition is evaluated.

1.2 Selection or Branching or Decision Control Instruction(Condition Expression)

The statements that make selection of statements to be executed are called selection statement or branching statement. The selective structure consists of a test for a condition followed by alternative path that the program can follow. The conditional expression is mainly used for decision making. There are various structures of the control statements. They are:

1.2.1 if statement:

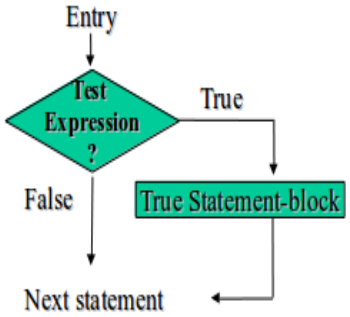
The **if** statement is used to express conditional expressions. If the given condition is true then it will execute the statements; otherwise it will execute optional statements. The braces { and } are used to group declarations and statements into compound statement or a block.

Syntax:

```
if(condition)
{
        //set of stmt1
}
//set of stmt2
```

If the condition is true set of stmt1 is executed and then after only set of stmt2 are executed otherwise set of stmt1 are skipped and directly set of stmt2 is executed.

Flowchart



Example:

```
#include <stdio.h>
#include <conio.h>
void main()
{
int marks;
clrscr();
printf("Enter marks of a student\n");
scanf("%d",&marks);

if(marks>=60)
        printf("Student is passed\n");
if(marks<60)
        printf("Student is Failed:\n");
```

```
getch();
}
```

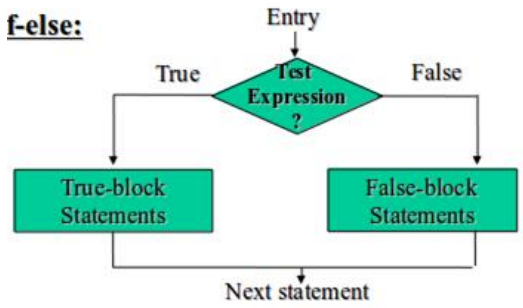
1.2.2 if..else statement:

In this case, either of the two statement is executed depending upon the value of the expression. The first statement is executed if the expression is true; otherwise the else statement is executed.

Syntax:

```
if(condition)
{
        //statement  set1
}
else
{
        //statement  set2
}
```

Flowchart



Q. WAP to find a number which is even or odd.

```
#include <stdio.h>
#include<conio.h>
void main()
{
        int n;
        clrscr();
        printf("Enter a number\n");
        scanf("%d",&n);
        if(n%2==0)
        {
                printf("The number is even\n");
        }
        else
        {
                printf("The number is odd \n");
        }
        getch();
}
```

Q. A program to read any two numbers from the keyboard and to display the larger value.

```
#include <stdio.h>
#include <conio.h>
void main()
{
        float x,y;
        clrscr();
        printf("Enter first number:");
        scanf("%f",&x);
        printf("Enter second number:");
        scanf("%f",&y);
```

```
        if(x>y)
        {
                printf("Larger value is %f",x);
        }
        else
        {
                printf("Larger value is %f",y);
        }
        getch();
 }
```

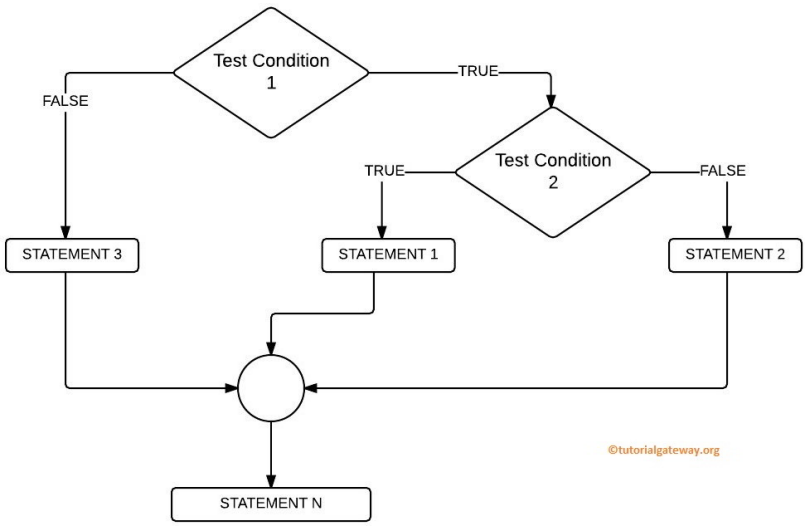
1.2.3 Nested if- else statements:

The if...else statement which is again inside if…else statements, is called nested if…else statements which are as follows.

Syntax:

```
if (expression)
{
        if(expression)
        {
                //statements
        }
        else
        {
                //statements
        }
}
else
{
        if(expression)
        {
                //statements
        }
        else
        {
                //statements
        }
}
```

Flowchart



Q. WAP to find the largest number among 3 numbers.

```
#include <stdio.h>
#include <conio.h>
void main()
{
        float x,y,z;
        clrscr();
        printf("Enter first number:");
        scanf("%f",&x);
        printf("Enter second number:");
        scanf("%f",&y);
        printf("Enter third number:");
        scanf("%f",&z);
        if(x>z)
        {
                if(x>y)
                {
                        printf("Largest value is %f",x);
                }
                else
                {
                        printf("Largest value is %f",y);
                }
        }
        else
        {
                if(z>y)
                {
                        printf("Largest value is %f",z);
                }
                else
                {
                        printf("Largest value is %f",y);
                }
        }
        getch();
}
```

1.2.4 if … else if ladder:

In else if ladder (or multi-way conditional statements), the condition expression is evaluated in order. If any of these expressions is formed to be true, the statement associated with it is executed and this terminates the whole chain. If none of the expression is true then the statement associated with final else is executed.

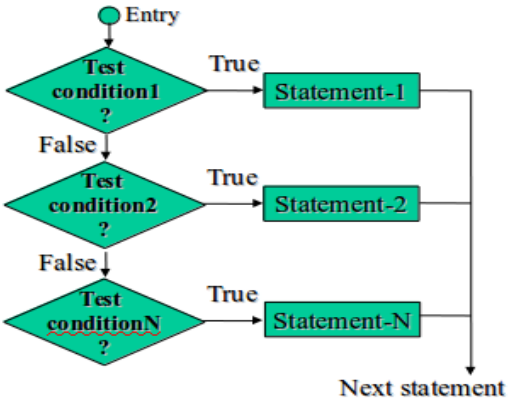
Syntax:

```
if(condition1)
{
        ……
}
else if (condition2)
{
        ……..
}

else if (condition3)
{
        ……
}
…
…
…
else
{
```

```
//default statement
}
```

Flowchart



Example1:

```
#include <stdio.h>
#include <conio.h>
void main()
{
int marks;
clrscr();
printf("Enter marks of a student\n");
scanf("%d",&marks);

if(marks>=80)
{
        printf("Student is passed in distinction:\n");
}
else if(marks>=60)
{
        printf("Student is passed in first division:\n");
}
else if(marks>=45)
{
        printf("Student is passed in second division:\n");
}
else if(marks>=32)
{
        printf("Student is passed in third division:\n");
}
else
{
        printf("Student is failed:\n");
}
getch();
}
```

Example2:

An electricity board charges according to following rates.
For the first 100 units----------Rs. 40 per unit
For the next 200 units----------Rs. 50 per unit
Beyond 300 units----------Rs. 60 per unit
All users are charged meter charge also, which is also Rs. 50. Write a program to read the number of units consumed, and print out the charges.

```
#include<stdio.h>
#include<conio.h>
#define meter_charge 50
void main()
```

```
{
int units, totalcharge, charge;
clrscr();
printf("enter units of electricity used");
scanf("%d",&units);

if(units>300)
        charge =(units-300)*60+200*50+100*40;
else if(units>100)
        charge =(units-100)*50+100*40;
else
        charge =units*40;
totalcharge =charge + meter_charge;
printf("\nThe payable amount is %d",totalcharge);
getch();
}
```

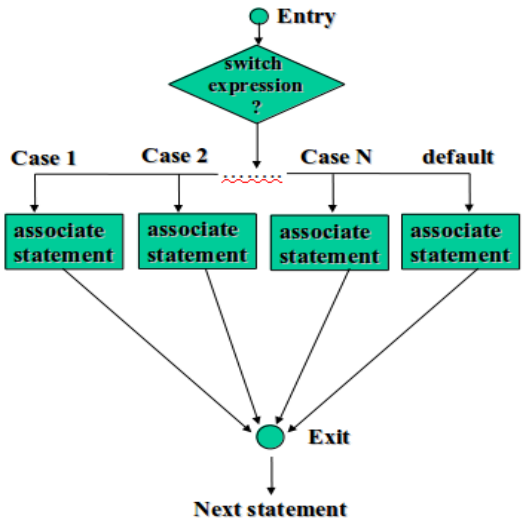
1.2.5 Case Control Instruction (Switch-Case statement)

The control statement that allows us to make a decision from the number of choices is called a switch statement. It chooses a particular group of statements from several available groups. It matches the variable with the case value. And the set of statements inside the case that matches the variable are executed. If none of the case value matches the expression then the default statement is executed. The value may be either integer or character.

Syntax:-

```
switch(variable)
{
case 1:
//statement set1
……..
break;
case 2:
//statement set2
……..
break;

case n:
//statement set_N
break;
default:
//default statement set ……..
break;
}
```



Example1:
WAP to determine whether a user entered number is odd or even.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a;
clrscr();
printf("Enter any number\n");
scanf("%d",&a);
a=a%2;
switch(a)
{
        case 0:
                printf("The number is even\n");
                break;
        default:
                printf("The number is odd");
}
getch();
}
```

Example2:
WAP to find the day on entering a number less than or equal to 7.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int day;
clrscr();
printf("Enter numeric day of the week:\n");
scanf("%d",&day);
switch(day)
{
        case 1:
                printf("Day is Sunday\n");
                break;
        case 2:
                printf("Day is Monday\n");
                break;
        case 3:
                printf("Day is Tuesday\n");
                break;
        case 4:
                printf("Day is Wednsday\n");
                break;
        case 5:
                printf("Day is Thursday\n");
                break;
        case 6:
                printf("Day is Friday\n");
                break;
        case 7:
                printf("Day is Saturday\n");
                break;
        default:
                printf("Your choice is wrong!!!!!!!!");

}
```

```
getch();
}
```

Example3:
Write a program to calculate area or perimeter of a rectangle.

```
#include<stdio.h>
#include<conio.h>
void main()
{
float length, breadth, Area, P;
int choice;
clrscr();
printf("Enter the value of length & breadth:");
scanf("%f %f", &length, &breadth);
printf("1> Area\n");
printf("2> Perimeter\n");
printf("Enter your choice:");
scanf("%d",&choice);
switch(choice)
{
        case 1:
                Area=length*breadth;
                printf("The area is %f", Area);
                break;
        case 2:
                P=2*(length+breadth);
                printf("The Perimeter is %f", P);
                break;
}
getch();
}
```

1.3 Repetition or Iteration or Loop Control Instruction

The repetitive structure consists of program statements that are repeatedly executed while some condition hold true. The computer has the ability to perform a set of instructions repeatedly. This involves repeating some portion of a program either for a specified number of times or until a given condition is satisfied. This repetitive operation is done by loop control statement.

There are three methods for generating repetition of a certain part of the program.

1.3.1 for statement
1.3.2 while statement
1.3.3 do…while statement

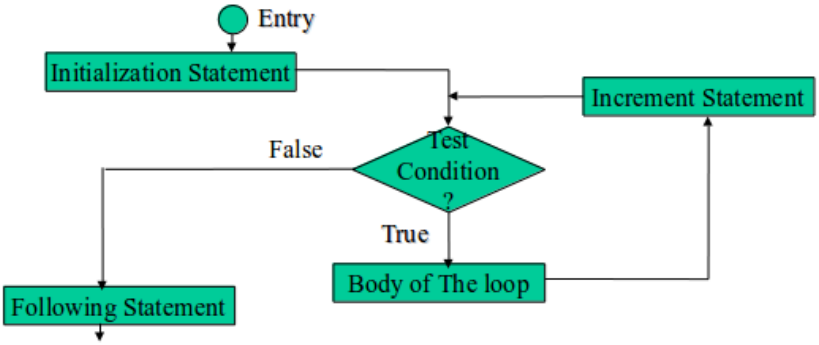
1.3.1 for loop:

The for loop is mostly used loop for repetitive structure. It is an entry control loop because it checks condition at the entry point.The for loop consists of three expressions. The first expression is ued to initialize the index value, the second to check whether the loop is to be continued again and third to change the index value for further repetition.

Syntax:

```
for(initialization; test condition;increment or decrement)
```

Flowchart



Example1: A program to display the numbers from 0 to 10 using for loop.

```
#include <stdio.h>
#include<conio.h>
void main()
{
int i=0;
clrscr();
for(i=0;i<=10;i++)
        printf("%d\n",i);
getch();
}
```

Example2: WAP to print all even numbers from 0 to 20.

```
#include <stdio.h>
#include <conio.h>
void main()
{
int i=0;
clrscr();
for ( ;i<=20; )
{
        printf("%d\t",i);
        i=i+2;
}
getch();
}
```

Example3: WAP to count all the integers greater than 100 and less than 200 that are divisible by 7. Also find the sum of these numbers.

```
#include <stdio.h>
#include <conio.h>
void main()
{
int i, count=0, sum=0;
clrscr();
for(i=101;i<200;i++)
{
        if (i%7 = = 0)
        {
                sum = sum + i;
                count = count + 1;
        }
}
printf("The sum is = %d\n", sum);
printf("The count is = %d", count);
getch();
```

}

Example4: A program to create the multiplication table of a number.

```
#include <stdio.h>
#include <conio.h>
void main()
{
int a,b,c;
clrscr();
printf("Enter a number:");
scanf("%d",&a);
for(b=1;b<=10;b++)
{
        c=a*b;
        printf("\n%d * %d=%d",a,b,c);
}
getch();
}
```

Nested Loops:

If we define loop insides a loop, such loops are called nested loops. For example:

```
for(i=0;i<10;i++)
{
        for(j=0;j<5;j++) //nested loop
        {
                ……
        }
}
```

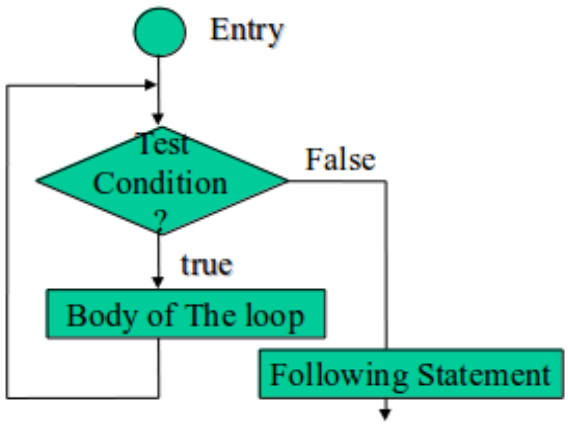
1.3.2 while loop statement:

The second type of loop statement is while loop. While loop first checks whether the initial condition is true or false and finding it to be true, it will enter the loop and execute the statement. Thus, it is also an entry control loop (perform pre-test).

Syntax:

```
initialization;
while(test condition)
{
        Statement 1;
        …
        Statement n;
increment or decrement;
}
```

lowchart



Example1: Program to print 10 natural numbers.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int number=1;
while(number<=10)
{
        printf("%d\t",number);
        number++;
}
getch();
clrscr();
}
```

Example2: Write a program to find factorial using while loop.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int fact=1, i=1,n;
printf("Enter the number:");
scanf("%d",&n);
while(i<=n)
{
        fact=fact*i;
        i++;
}
printf("Factorial of %d is %d",n,fact);
getch();
clrscr();
}
```

Difference between for loop and while loop

for Loop	**while Loop**
It is definite loop i.e. exact number of iterations	It may be definite and may not be definite.
The loop expression is within one block.	The loop expression is scattered throughout the program
It uses the keyword **for**.	It uses the keyword **while**.

For example:
void main()
{
int n;
for(n=0;n<10;n++)
{
printf("Nepal");
}
}
	For example:
	void main()
	{
	int n=10;
	while(n!=0)
	{
	printf("Nepal");
	n--;
	}
	}

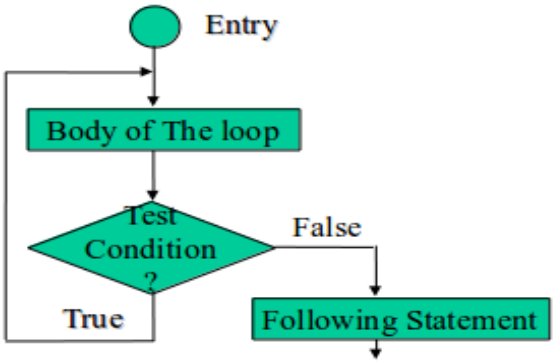
1.3.3 do…while loop statement:

The do...while loop is another repetitive loop used in C programs. It differs from the for loop and while loop. The do…while loop is an exit control (post-test) loop because it tests the condition at the exit point. The while loop tests the condition before executing any of the statements inside the while loop but the do…while loop test the condition after executing the statements. This means that the do…while loop would execute its statements at least once even if the condition fails for first time. The while loop will not execute statements if the condition fails for the first time.

Syntax:

```
do
{
        Statement 1;
        Statement n;
}
while(condition);
```

Flowchart



Example1:

A program to find the sum of odd numbers using do..while loop.

```
#include <stdio.h>
#include<conio.h>
void main()
{
int n, sum=0, i=1;
clrscr();
printf("Enter the value of n:");
scanf("%d",&n);
do
{
        sum=sum + i;
        i=i+2;
}while(i<=n);
printf("The sum of odd number is %d", sum);
getch();
}
```

Difference between while loop and do-while loop

while Loop	**do-while Loop**
The while loop is entry controlled loop.	The do-while loop is exit control loop.
It has a keyword **while**.	It has two keyword **do** and **while**.
Loop is not terminated with semicolon.	Loop is terminated with a semicolon.
It doesn't execute the body of loop until and unless the condition is true.	Body of the loop will always be executed at least once since the test condition is not checked until the end of the loop.

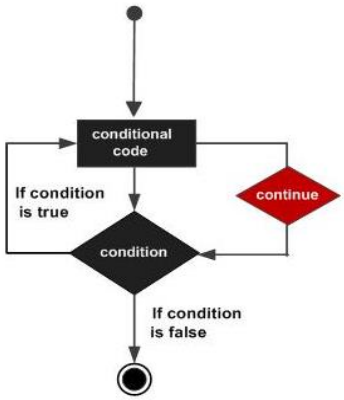
Example: Print "Hello"
void main()
{
int i=0;
while (i>=4)
	Example: Print "Hello"
	void main()
	{
	int i=0;
	do

{
printf("Hello \n");
i++;
}
}
Output: Noting will display.
	{
	printf("Hello \n");
	i++;
	}while(i>=4);
	}
	Output: Hello

2. Continue statements

Continue statement is used to bypass the execution of the statements inside the loop. In some situations we want to jump to the next iterations of the loop by ignoring the execution of statements specified within the body of loop. The keyword "continue" allows us to do this.

When the keyword is encountered inside the loop, control automatically passes to the next iteration of the loop. A continue is specified using if() statement.



Example:

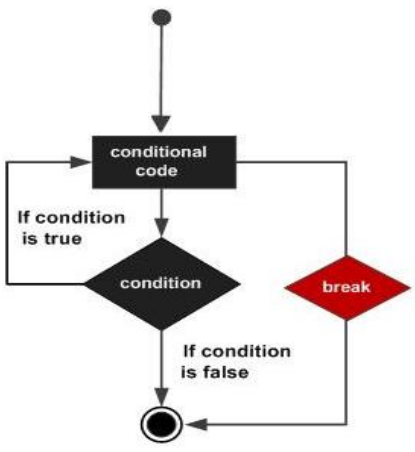
```
void main()
{
int i;
for (i =1; i<=5;i++)//The continue statement is always inside loop
{
        if (i==3)
        {
                continue;
        }
        printf("%d",i);
}
}
```

Output: 1 2 4 5

Thus, continue statement is used to skip a part of the body of loop and continue with next iteration. Note that when the value of i equal to 3, the continue statement takes the control to the next iteration bypassing the execution of the printf() statement.

3. Break Statement

The break statement is used as a statement inside the body of loop. Generally the break statement is associated with a condition specified with if () statement. As soon as the condition evaluates within if (), the execution of break statement takes the control outside the loop and therefore stops the further iterations of the loop.



Example:

```
void main()
{
int i;
for (i =1; i<=5;i++)
{
        printf("%d",i);
        if ( i = =3)
                break;
}
printf("\n loop terminates here");
}
```

Output:

1 2 3

Q. **What is the similarity and difference between break and continue statements?**

Similarity: Both break and continue are jump statements.

Difference: The break statement terminates the entire loop execution whereas continue statement terminates single pass of the loop.

4. goto statement

The goto statement is used to alter the program execution sequence by transferring the control to some other part of the program. It is used to jump to a specific location.

Example: Program to demonstrate the usage of goto statement.

```
#include <stdio.h>
#include<conio.h>
void main()
{
 int a,b;
printf("Enter two numbers:");
scanf("%d %d",&a,&b);
if(a>b)
        goto label1;
else
        goto label2;
label1:
        printf("Largest value=%d",a);
label2:
        printf("Largest value=%d",b);
getch();
}
```

Programs

1. For Loop Programs

Q. A program to find the sum and the average of given numbers.

```
#include <stdio.h>
void main()
{
int n, i;
float sum=0.0, a, average;
printf("enter the amount of number we want to add");
scanf("%d",&n);
for(i=1; i<=n; i++)
{
        printf("Enter a number to be added:");
        scanf("%f ",&a);
        sum=sum+a;
}
average=sum/n;
printf("Sum=%f ", sum);
printf("\nAverage= %f ", average);
}
```

Q. A program to display the Fibonacci series.

```
#include <stdio.h>
#include <conio.h>
void main()
{
int fibo3,fibo1=0,fibo2=1,num,i;
clrscr();
printf("enter the number of Fibonacci series to be displayed");
scanf("%d",&num);
printf("%d\t%d",fibo1,fibo2);
for(i=3; i<=num; i++)
{
        fibo3=fibo1+fibo2;
        fibo1=fibo2;
        fibo2=fibo3;
        printf("\t%d",fibo3);
}
getch();
}
```

Q. If two numbers n1 & n2 are input through the keyboard. WAP to find sum of those numbers which is exactly divisible by 5 between n1 and n2.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int n1, n2, sum=0, i;
printf("enter n1 & n2");
scanf("%d%d",&n1, &n2);
for(i=n1; i<=n2; i++)
{
        if(i%5==0)
        sum=sum+i;
}
printf("required sum=%d",sum);
getch();
}
```

2. While Loop Program

Q. Write a program to check whether the given number is prime number or not.

```
#include <stdio.h>
#include <conio.h>
void main()
{
int num , i;
printf("Enter the number:");
scanf("%d",&num);
i=2;
while(i<=num-1)
{
        if (num% i = = 0)
        {
                printf("%d is not a prime number",num);
                break;
        }
i++;
}
if (i = = num)
        printf("%d is prime number",num);
getch();
}
```

Q. WAP to check whether it is palindrome or not. (Hint: A number is palindrome if it is equal to its reverse. For example: 4224 is palindrome but 4223 is not palindrome. To check whether a number is palindrome or not, first we should find the reverse of a number and check whether it is equal to its reverse or not.)

```
#include<stdio.h>
#include<conio.h>
void main()
{
int n,rev=0,y,d;
printf("enter any number:");
scanf("%d",&n);
y=n;
while(n!=0)
{
        d=n%10;
        rev=rev*10+d;
        n=n/10;
}
if(rev==y)
        printf("%d is palindrome",y);
else
        printf("%d is not a palindrome",y);
getch();
}
```

Q. WAP to check whether a number is Armstrong number or not.

```
#include<stdio.h>
#include<conio.h>
void main()
```

```
{
        int n,m,sum=0,digit;
        printf("enter any number");
        scanf("%d",&n);
        m=n;
        while(n!=0)
        {
                digit=n%10;
                sum=sum+digit*digit*digit;
                n=n/10;
        }
        if(m==sum)
        {
                printf("%d is armstrong number",m);
        }
        else
        {
                printf("%d is not armstrong number",m);
        }
        getch();
}
```

Q. WAP to print all ASCII values & their equivalent characters using a while loop.

```
#include<stdio.h>
#include<conio.h>
void  main()
{
int  i=0;
while(i<=255)
{
        printf("%d %c\t",i,i);
        i++;
}
getch();

}
```

3. Nested Loop Program

Q. WAP to print the following output:

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

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

Q. WAP to print the following output:

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

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

Q. WAP to print the following output:

```
1	2	3	4	5
2	4	6	8	10
3	6	9	12	15
4	8	12	16	20
5	10	15	20	25
```

```
#include<stdio.h>
#include<conio.h>
void main()
{
	int i, j, m;
	for(i=1; i<=5; i++)
	{
		for(j=1; j<=5; j++)
		{
			m=i*j;
			printf("%d\t",m);
		}
		printf("\n");
	}
	getch();
}
```

Q. WAP to print the following output:

```
1
2	3
4	5	6
7	8	9	10
```

```
#include<stdio.h>
#include<conio.h>
```

```
void main()
{
int i, j, k=1;
for(i=1; i<=4; i++)
{
        for(j=1; j<=i; j++)
        {
                printf("%d\t",k);
                k++;
        }
        printf("\n");
}
getch();

}
```

Q. WAP to print the following output

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

```
#include <stdio.h>
#include <conio.h>
void main()
{
int i, j, k;
for(i=1; i<=4; i++)
{
        for(k=3; k>=i; k--) //this loop is used to provide space
        {
        printf ("\t");
        }
                for(j=1; j<=i; j++)
                {
                printf ("*\t");
                }
        printf("\n");
}
getch();
}
```

Q. WAP to print the following output

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

```
#include <stdio.h>
#include <conio.h>
void main()
{
int i, j;
for(i=1; i<=4; i++)
{
        for(j=4; j>=i; j--)
        {
                printf("*\t");
        }
        printf("\n");
}
```

```
getch();
}
```