Programming in C

By

Dr Ramkumar Krishnamoorthy

kramdharma@gmail.com

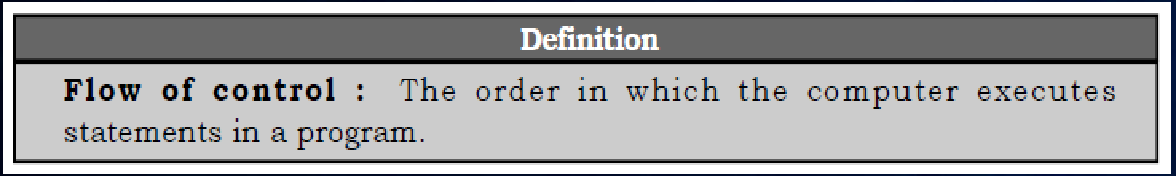
Fundamentals of C

Unit II

Contents

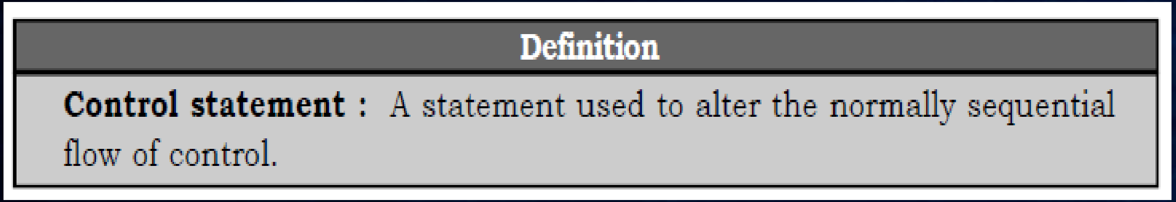
Unit II – Chapter I

❝ *It gives an Idea about the uses of Branching, Looping, Arrays, Strings, Some Advanced Features*



Definition

Flow of control : The order in which the computer executes statements in a program.



Definition

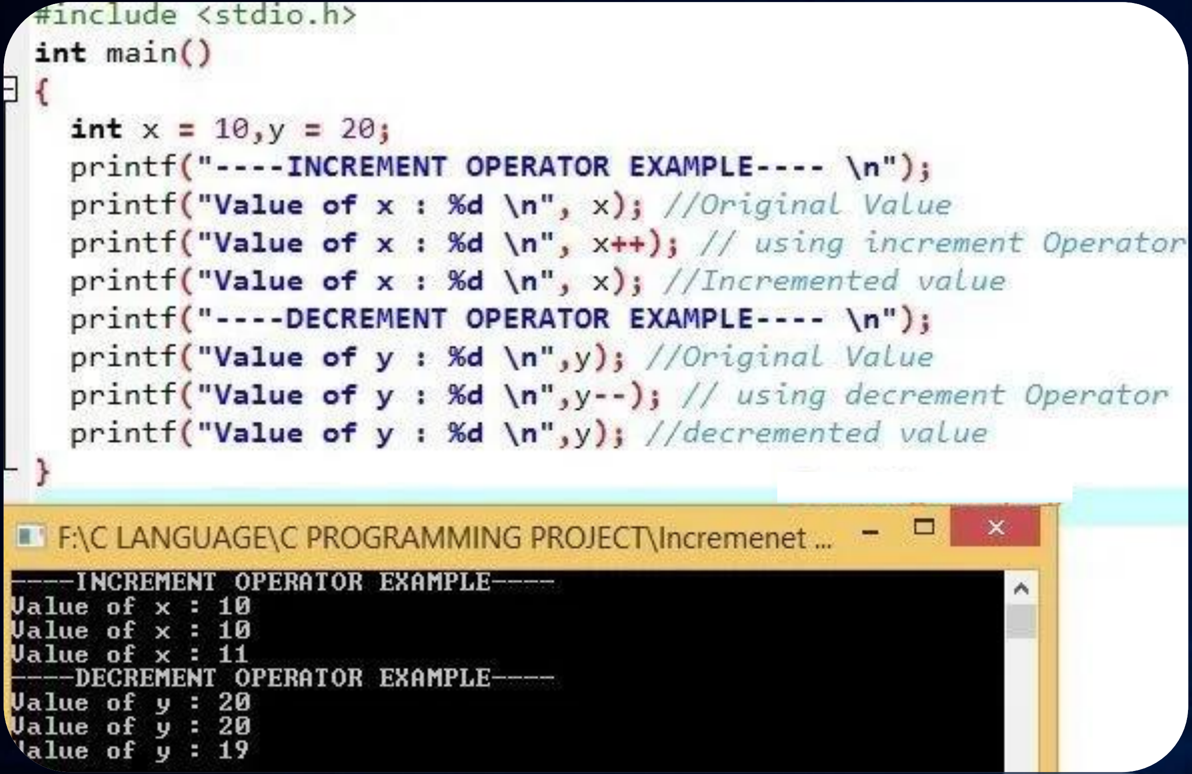
Control statement : A statement used to alter the normally sequential flow of control.

Increment/Decrement Operator

- **Increment – It is used to increase the value by one (1)**
  - Pre increment – It increases the value by 1 and prints the result (++x)
  - Post increment – It prints the value and increases the value by 1 (x++)
- **Decrement – It is used to decrease the value by one (1)**
  - Pre decrement – It decreases the value by 1 and prints the result (--x)
  - Post decrement – It prints the value and decreases the value by 1 (x--)

Increment/Decrement Operators		Let us assume X is a variable
Operator	**Expression**	**Description**
++	++X	Pre-increment
	X++	Post-increment
--	--X	Pre-decrement
	X--	Post-decrement

Increment/Decrement Operator



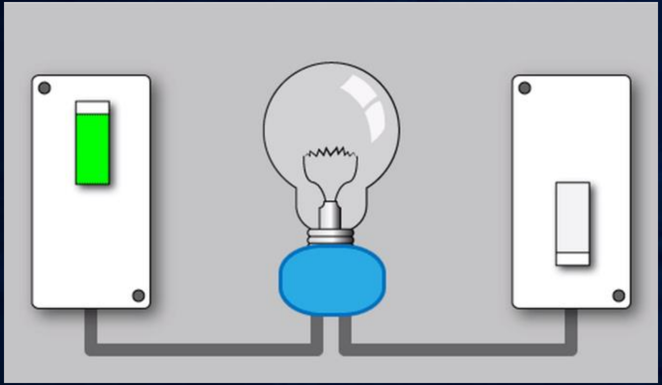
```
#include <stdio.h>
int main()
{
    int x = 10,y = 20;
    printf("----INCREMENT OPERATOR EXAMPLE---- \n");
    printf("Value of x : %d \n", x); //Original Value
    printf("Value of x : %d \n", x++); // using increment Operator
    printf("Value of x : %d \n", x); //Incremented value
    printf("----DECREMENT OPERATOR EXAMPLE---- \n");
    printf("Value of y : %d \n",y); //Original Value
    printf("Value of y : %d \n",y--); // using decrement Operator
    printf("Value of y : %d \n",y); //decremented value
}
```

F:\C LANGUAGE\C PROGRAMMING PROJECT\Incremenet ...

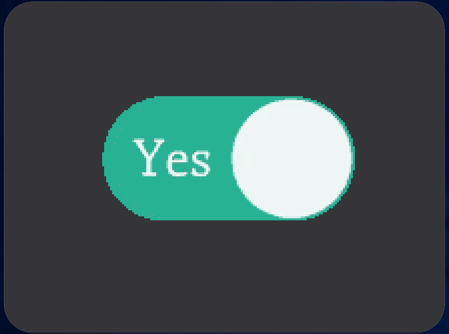
```
----INCREMENT OPERATOR EXAMPLE----
Value of x : 10
Value of x : 10
Value of x : 11
----DECREMENT OPERATOR EXAMPLE----
Value of y : 20
Value of y : 20
Value of y : 19
```

Branching

- When an "Algorithm" makes a choice to do one of two (or more things) this is called branching
- Simply can say **making choices**
- Branching Statements are decision making statements, which decide the flow of program execution
- C language offers following branching statements
  - if
  - if-else
  - else-if
  - nested if
  - switch



Branching



Branching – Simple if

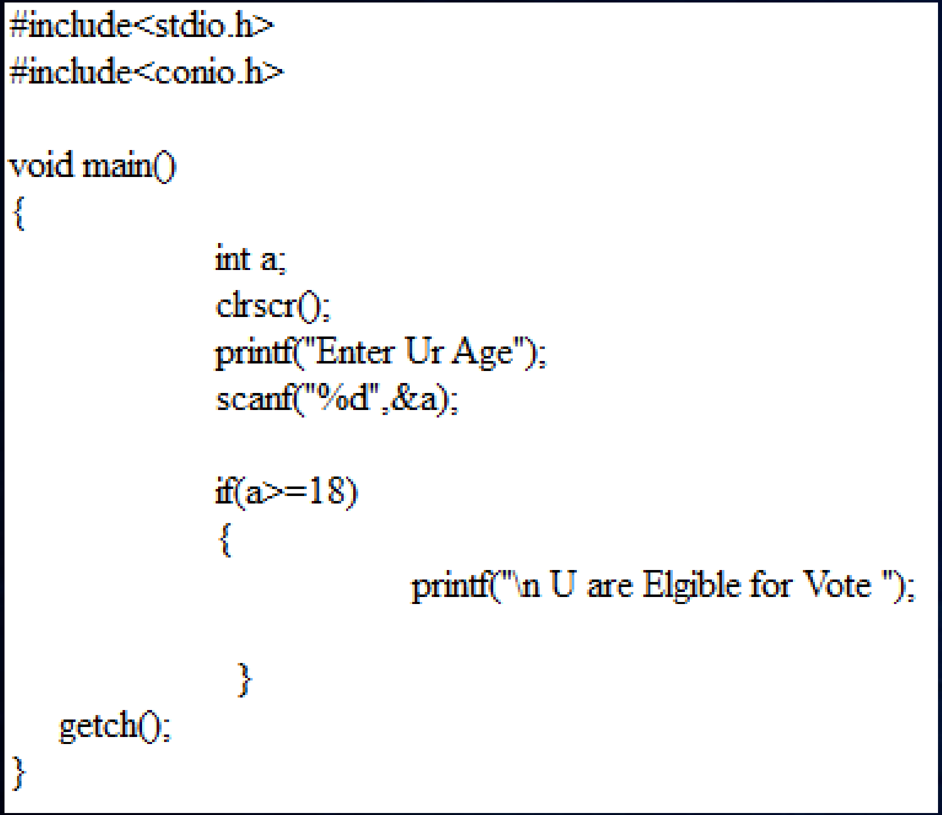
- It takes some conditions and TRUE when it got satisfied else exits from the block
- Syntax

```
if (Condition)
 {
     Statements of the block;
 }
```

- Example

```
if(age>18)
{
    printf("Hey Dear You can vote now");
}
```

Example for If

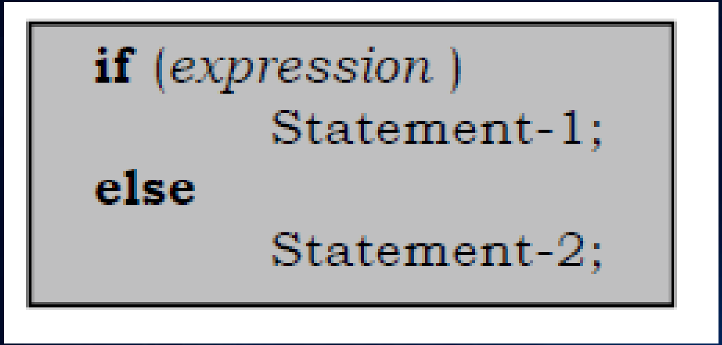


```
#include<stdio.h>
#include<conio.h>

void main()
{
                int a;
                clrscr();
                printf("Enter Ur Age");
                scanf("%d",&a);

                if(a>=18)
                {
                                printf("\n U are Elgible for Vote ");

                 }
    getch();
}
```

if else

- It takes some conditions and TRUE when it got satisfied else exits from the block
- Syntax

```
if (Condition)
 {
        Statements of the block;
 }
else
 {
       Statements of the block;
  }
```



```
if (expression)
        Statement-1;
else
        Statement-2;
```

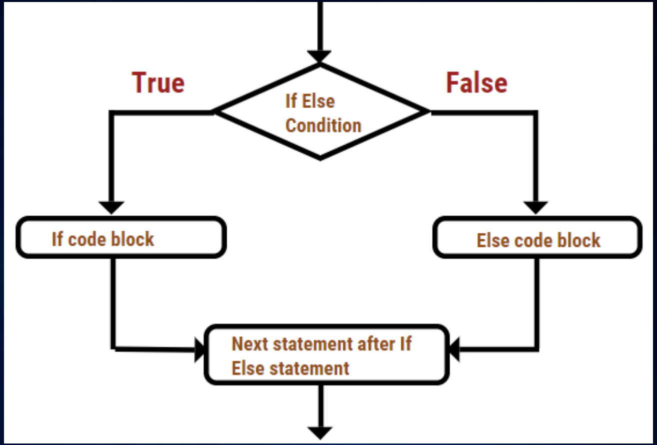
- Example

```
    if(age>18)
   {
       printf("Hey Dear You can vote now");
    }
    else {
       printf("Nope, Finish the 18 first");
     }
```

if else

Example

- Food
- If Food ==
  - Vegeterian

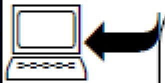


- Else
  - Non-Vegeterian

Example for if else

```
#include <stdio.h>
main()
{
      int   num;
      printf(" Enter a number  ");
      scanf("%d",&num);
      if(num % 2 == 0)
            printf("Number is even ");
      else
            printf("Number is odd ");
}
```

```
if (a>b)
        printf("Largest number is %d",a);
else
        printf("Largest number is %d",b);
```



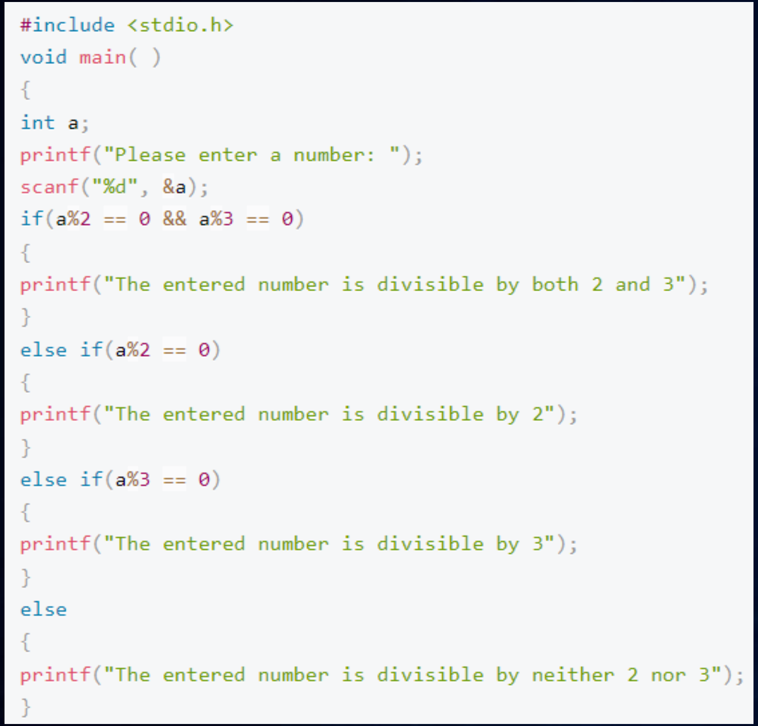
```
Output 1        Enter a number  14
                Number is even
Output 2        Enter a number  13
                Number  is odd
```

Else – If

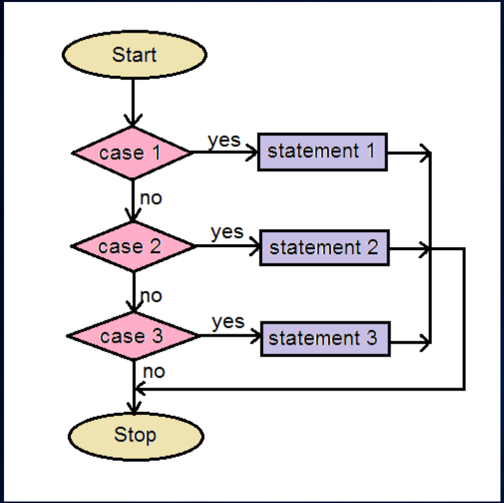
- It takes more than one IF conditions and checks each whichever satisfied that particular block be executed.
- Syntax

```
if (Condition 1)
     Statements of the block;
else if (Condition 2)
    Statement of the block;
else if (Condition 3)
    Statement of the block;
else
        Statement of the block;
```

Example for Else – If

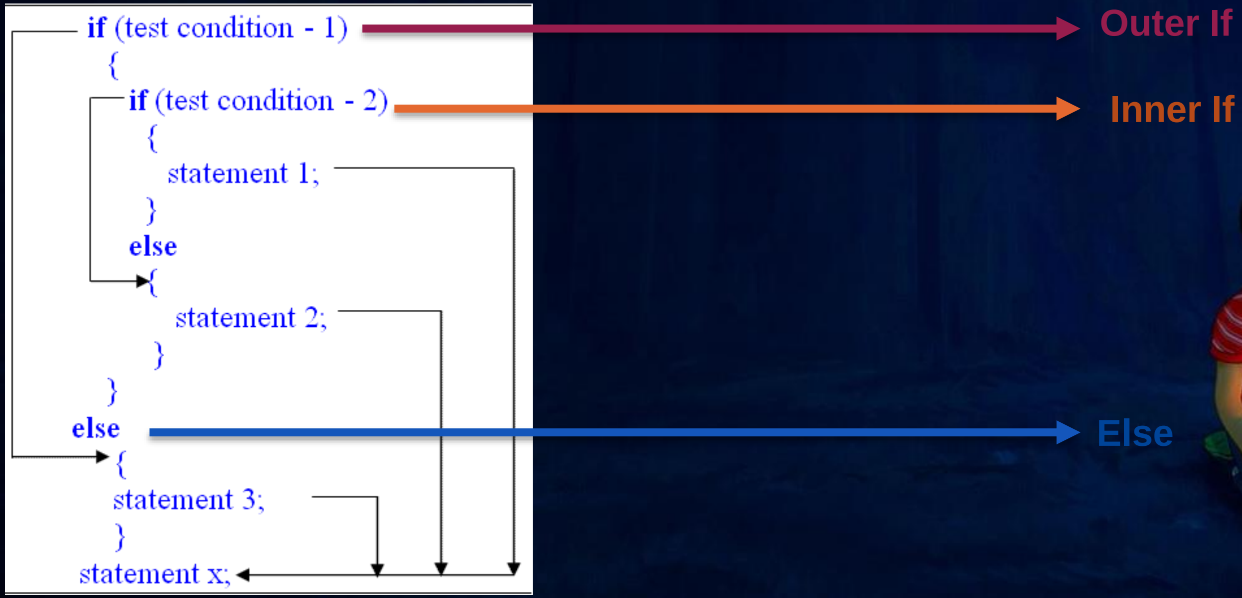


```c
#include <stdio.h>
void main( )
{
int a;
printf("Please enter a number: ");
scanf("%d", &a);
if(a%2 == 0 && a%3 == 0)
{
printf("The entered number is divisible by both 2 and 3");
}
else if(a%2 == 0)
{
printf("The entered number is divisible by 2");
}
else if(a%3 == 0)
{
printf("The entered number is divisible by 3");
}
else
{
printf("The entered number is divisible by neither 2 nor 3");
}
```



Nested – If

- A if statement may itself contain another if statement inside it, then it is known as nested if statement.
- First **outer If statement gets checked**, **if it is true** it allows to **pass into inner if, again true, inner if gets executed** else **comes out from both if and control moves to else part**
- **Syntax**



```
if (test condition - 1)
 {
  if (test condition - 2)
   {
    statement 1;
   }
  else
   {
    statement 2;
   }
 }
else
 {
 statement 3;
 }
statement x;
```

Example for Nested – If

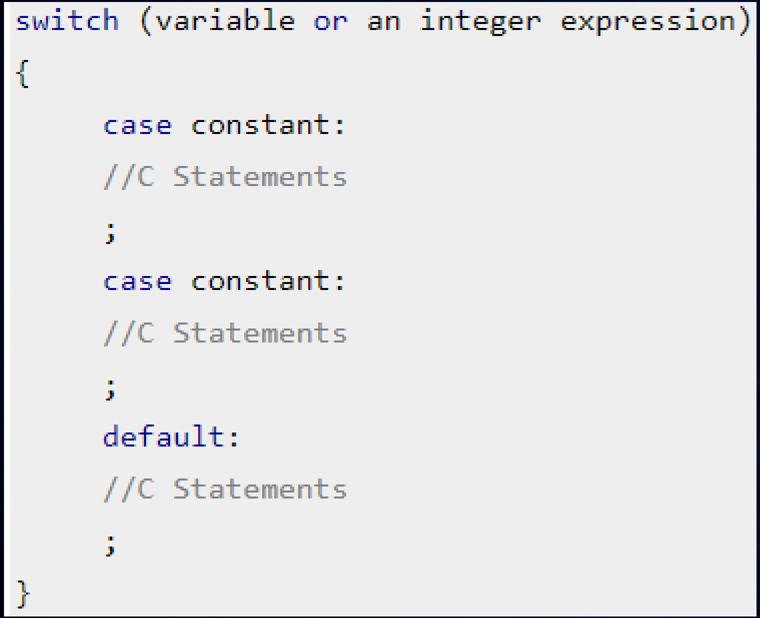
```
#include<stdio.h>
int main()
{
        int num=1;
        if(num<10)
        {
                if(num==1)
                {
                        printf("The value is:%d\n",num);
                }
                else
                {
                        printf("The value is greater than 1");
                }
        }
        else
        {
                printf("The value is greater than 10");
        }
        return 0;
}
```

```
if ( )
{
    if ( )
    {
    }
    else
    {
    }
}
```

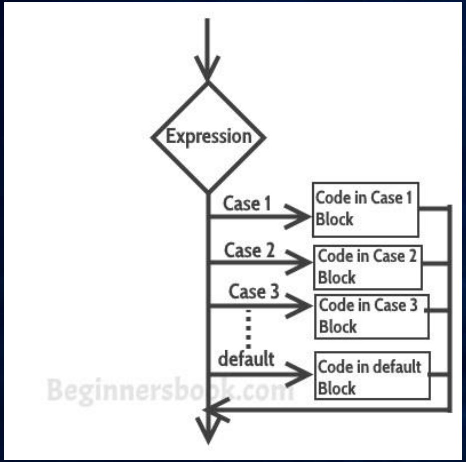
Switch

- The switch case statement is used when we have multiple options and we need to perform a different task for each option.
- It is also known as Multiway decision making statement

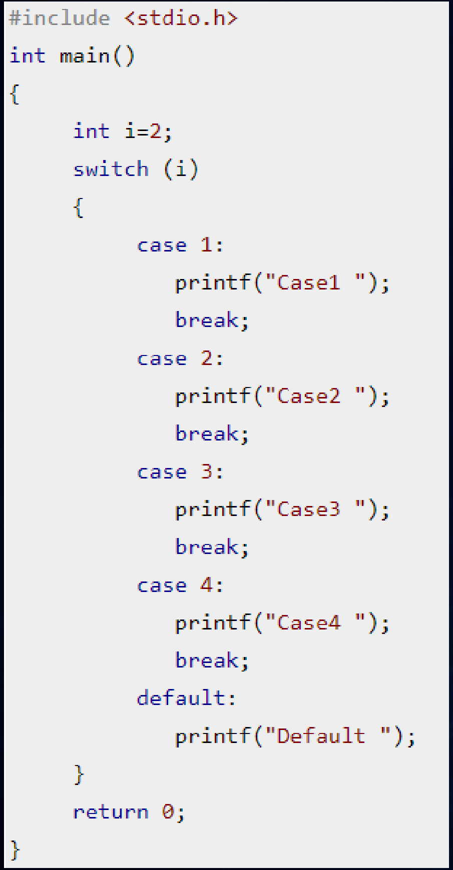
Syntax



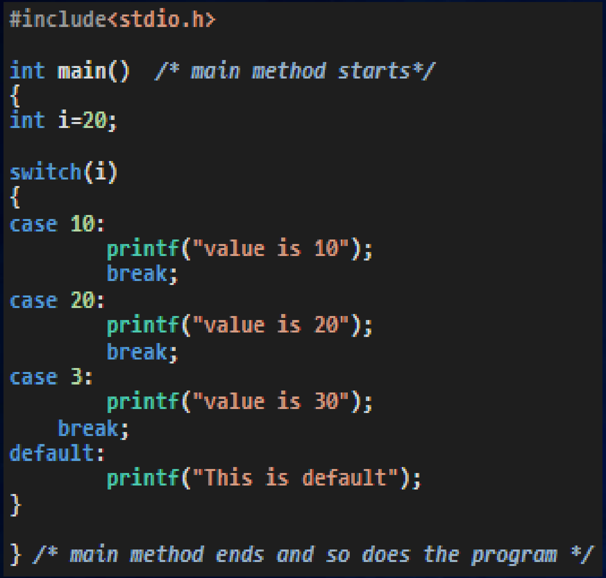
```
switch (variable or an integer expression)
{
     case constant:
     //C Statements
     ;
     case constant:
     //C Statements
     ;
     default:
     //C Statements
     ;
}
```



Example for Switch

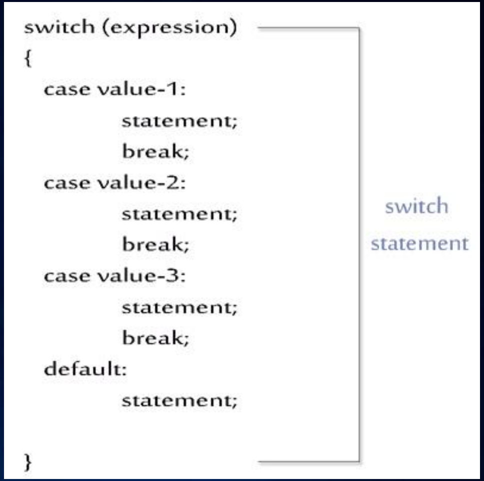


```c
#include <stdio.h>
int main()
{
    int i=2;
    switch (i)
    {
        case 1:
          printf("Case1 ");
          break;
        case 2:
          printf("Case2 ");
          break;
        case 3:
          printf("Case3 ");
          break;
        case 4:
          printf("Case4 ");
          break;
        default:
          printf("Default ");
    }
    return 0;
}
```



```c
#include<stdio.h>

int main()  /* main method starts*/
{
int i=20;

switch(i)
{
case 10:
        printf("value is 10");
        break;
case 20:
        printf("value is 20");
        break;
case 3:
        printf("value is 30");
    break;
default:
        printf("This is default");
}

} /* main method ends and so does the program */
```

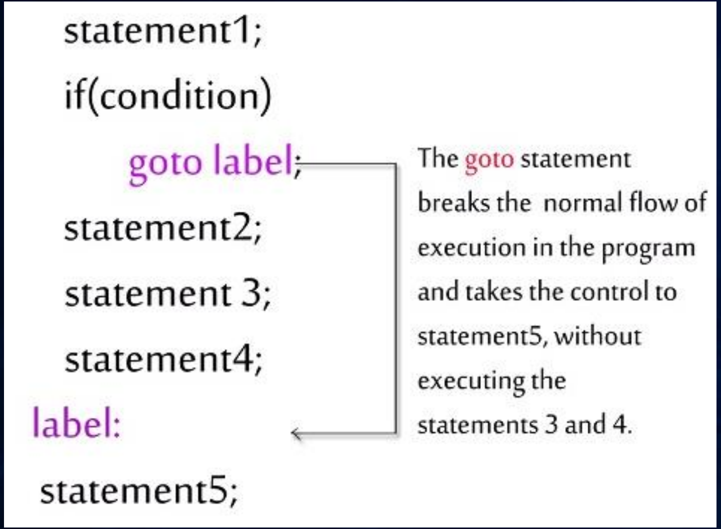
- Syntax



```
switch (expression)
{
  case value-1:
        statement;
        break;
  case value-2:
        statement;
        break;
  case value-3:
        statement;
        break;
  default:
        statement;

}
```

Goto Statement

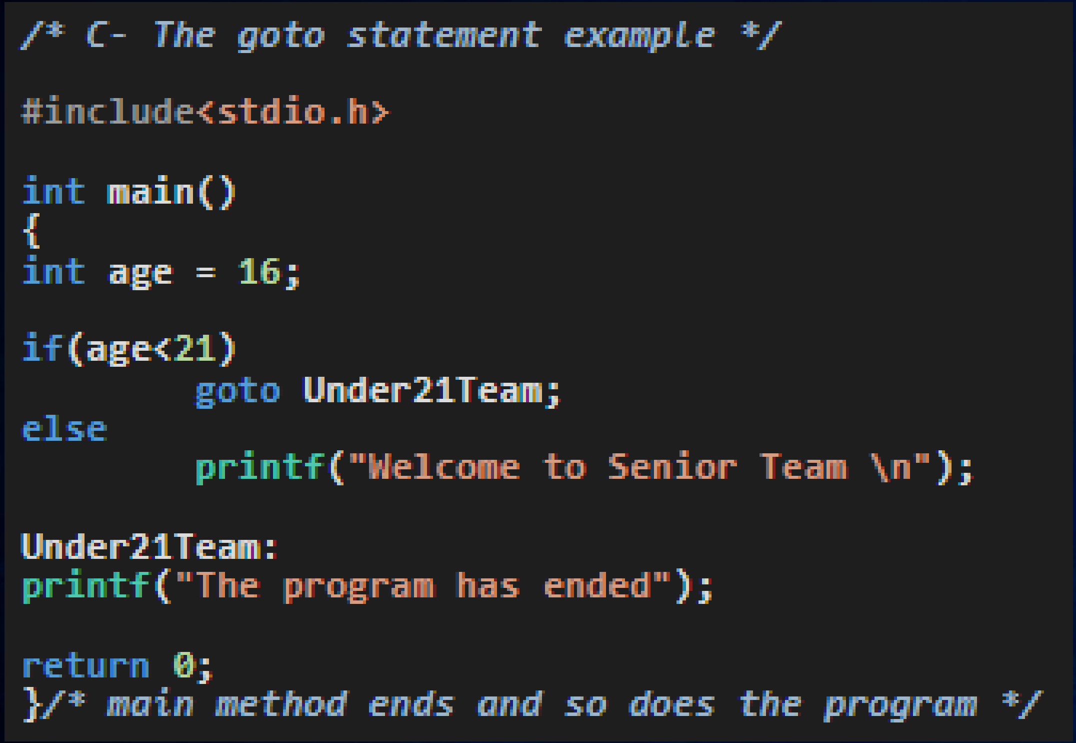
- In C language, the **goto** statement is used to take the control of the program to almost anywhere in the program
- **Syntax**
- **goto label;**



```
statement1;
if(condition)
    goto label;
statement2;
statement 3;
statement4;
label:
statement5;
```

The goto statement breaks the normal flow of execution in the program and takes the control to statement5, without executing the statements 3 and 4.

Example for Goto

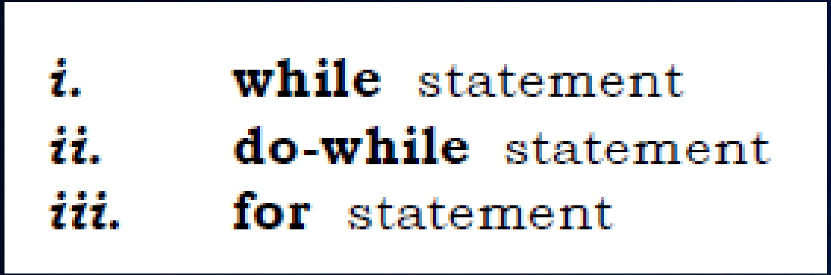


```c
/* C- The goto statement example */

#include<stdio.h>

int main()
{
int age = 16;

if(age<21)
        goto Under21Team;
else
        printf("Welcome to Senior Team \n");

Under21Team:
printf("The program has ended");

return 0;
}/* main method ends and so does the program */
```

Looping

- A loop is a program construct that causes a statement to be executed again and again.
- The process of repeating the execution of a certain set of statements again and again is termed as looping.

- C Language has the following looping statements

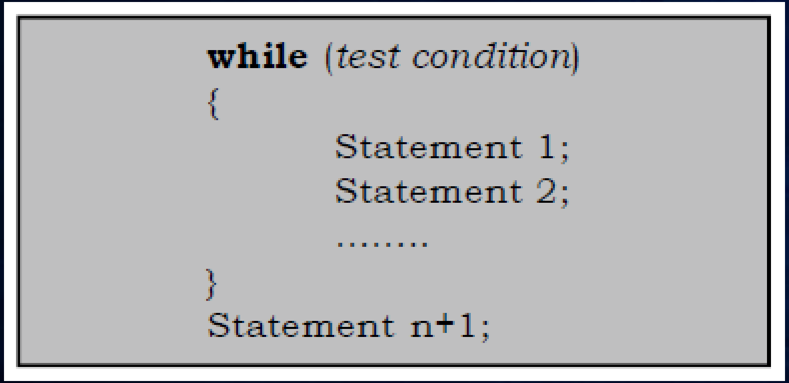


i. **while** statement
ii. **do-while** statement
iii. **for** statement

While Statement

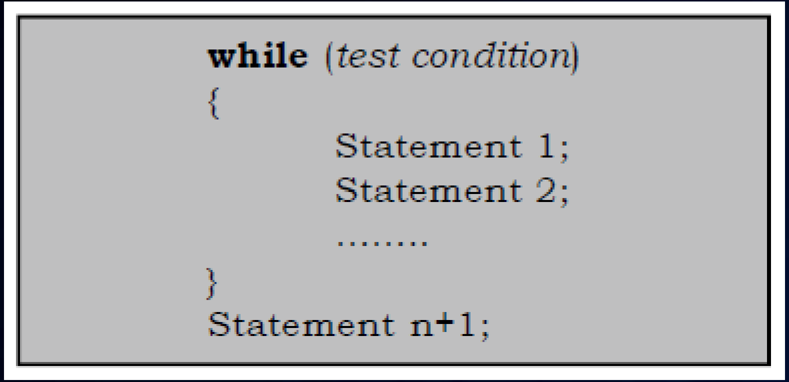
- In this structure the checking of a condition is done at the beginning
- It must have initial value and termination should meet/satisfy the condition for exiting the loop
- The condition must be satisfied before the execution of the statements i.e., the set of statements in the structure(block) is executed again and again until the test condition is true

- If the test condition becomes false control is transferred out of the structure(block).
- Syntax

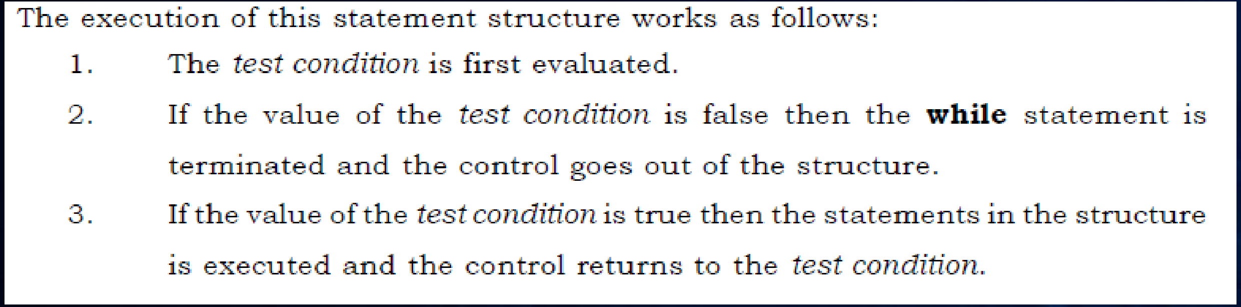


```
while (test condition)
{
        Statement 1;
        Statement 2;
        ........
}
Statement n+1;
```

While Statement



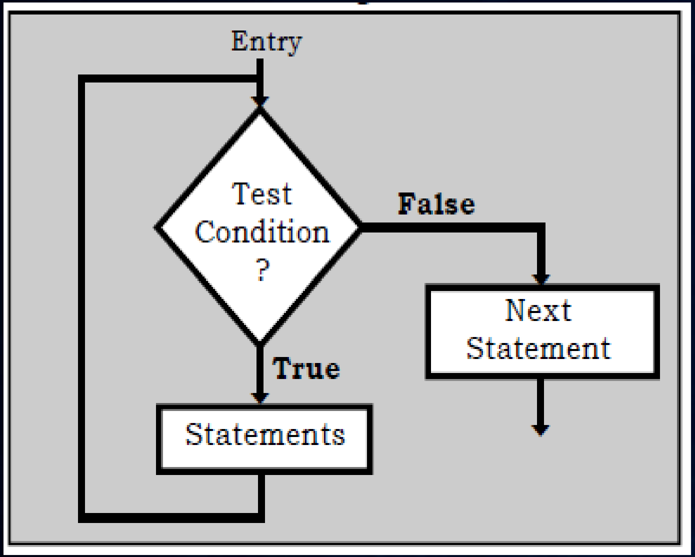
```
while (test condition)
{
        Statement 1;
        Statement 2;
        ........
}
Statement n+1;
```

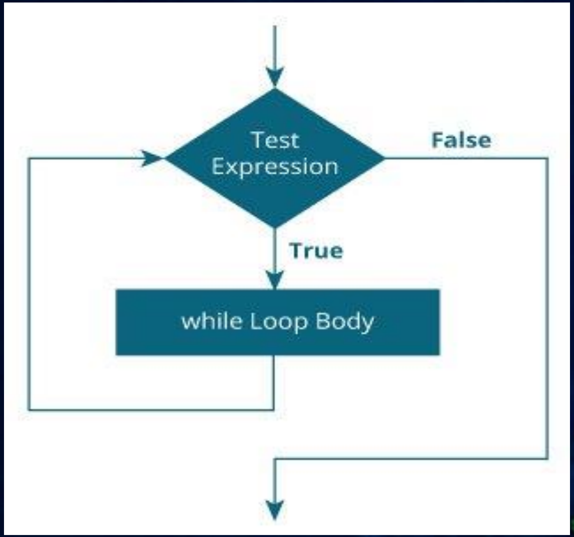


The execution of this statement structure works as follows:

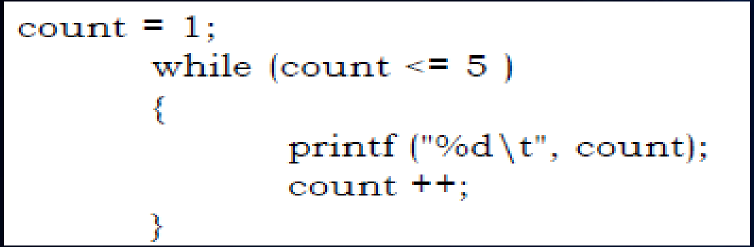
1. The *test condition* is first evaluated.
2. If the value of the *test condition* is false then the **while** statement is terminated and the control goes out of the structure.
3. If the value of the *test condition* is true then the statements in the structure is executed and the control returns to the *test condition*.

While Loop Statement – Flowchart

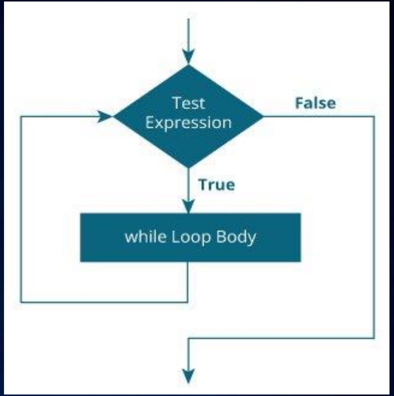




While Loop Statement – Example



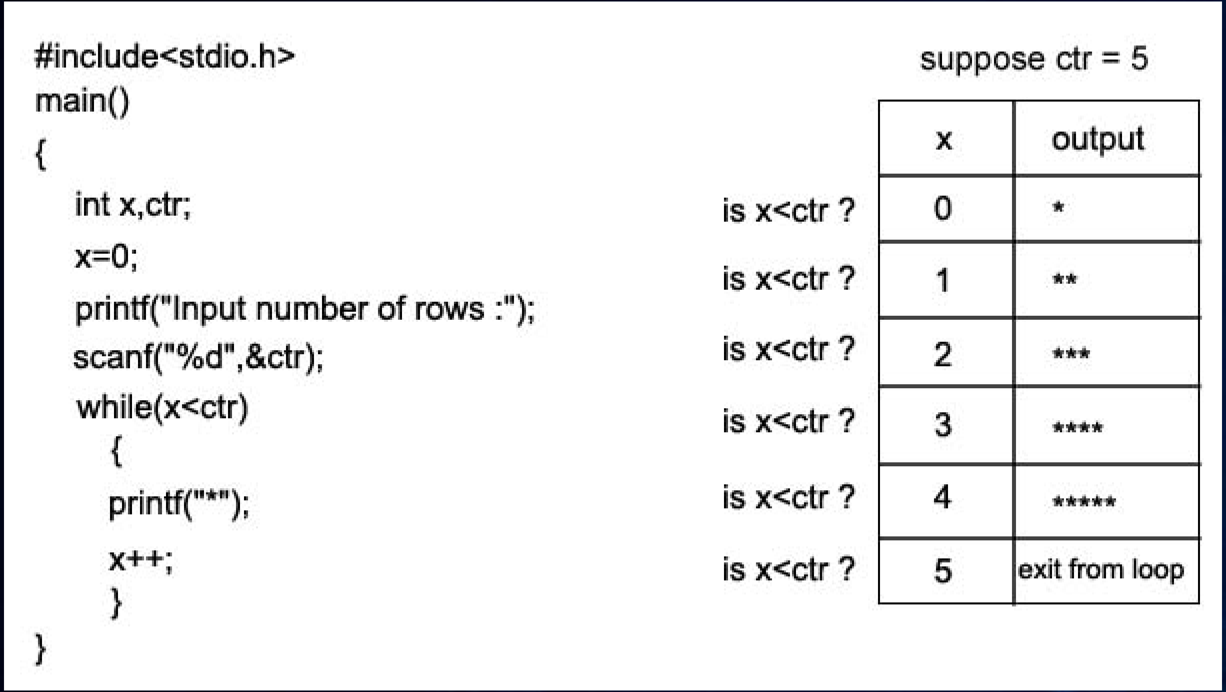
```
count = 1;
        while (count <= 5 )
        {
                printf ("%d\t", count);
                count ++;
        }
```



```
int a = 1;
while ( a < 4 )
{
printf ( "Hello World\n" );
a ++;
}
```

Output

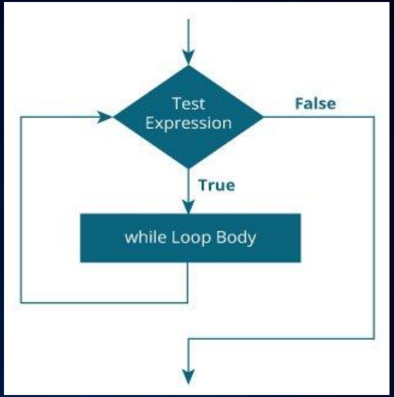
While Loop Statement – Example



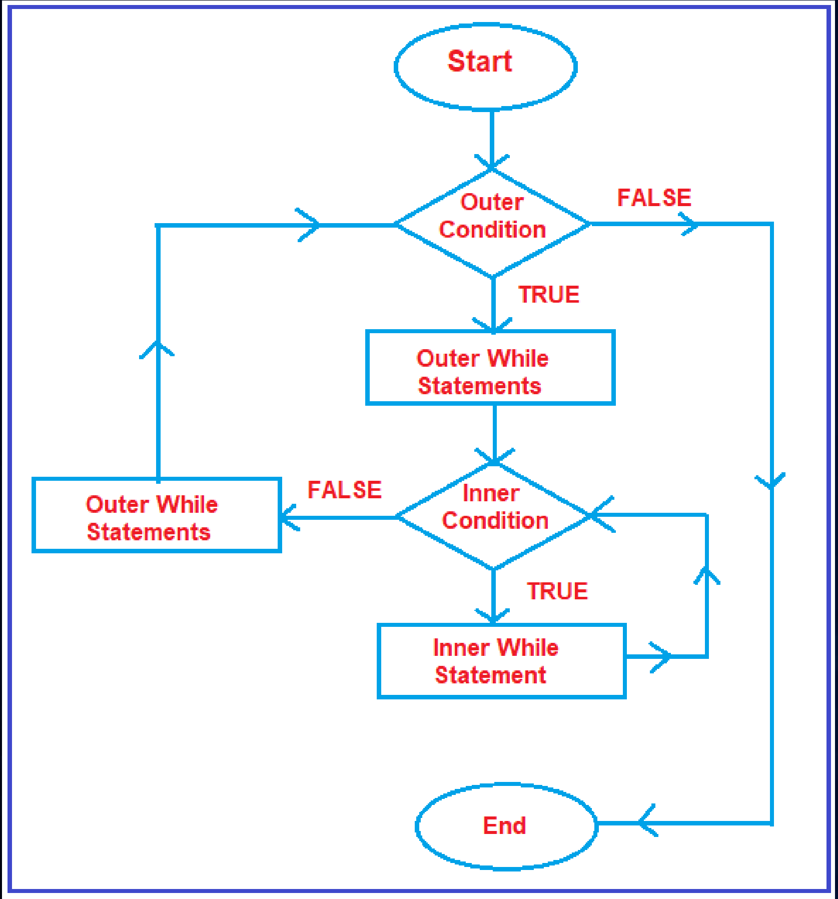
```
#include<stdio.h>
main()
{
    int x,ctr;
    x=0;
    printf("Input number of rows :");
    scanf("%d",&ctr);
    while(x<ctr)
       {
       printf("*");
       x++;
       }
}
```

suppose ctr = 5

	x	output
is x<ctr ?	0	*
is x<ctr ?	1	**
is x<ctr ?	2	***
is x<ctr ?	3	****
is x<ctr ?	4	*****
is x<ctr ?	5	exit from loop



Nested While Loop Example

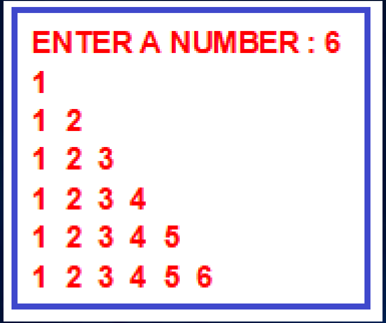


Syntax:

```
while (outer condition)
{
        Outer while Statements;

        while (inner condition)
        {
              Inner while Statements;
        }

        Outer while Statements;
}
```

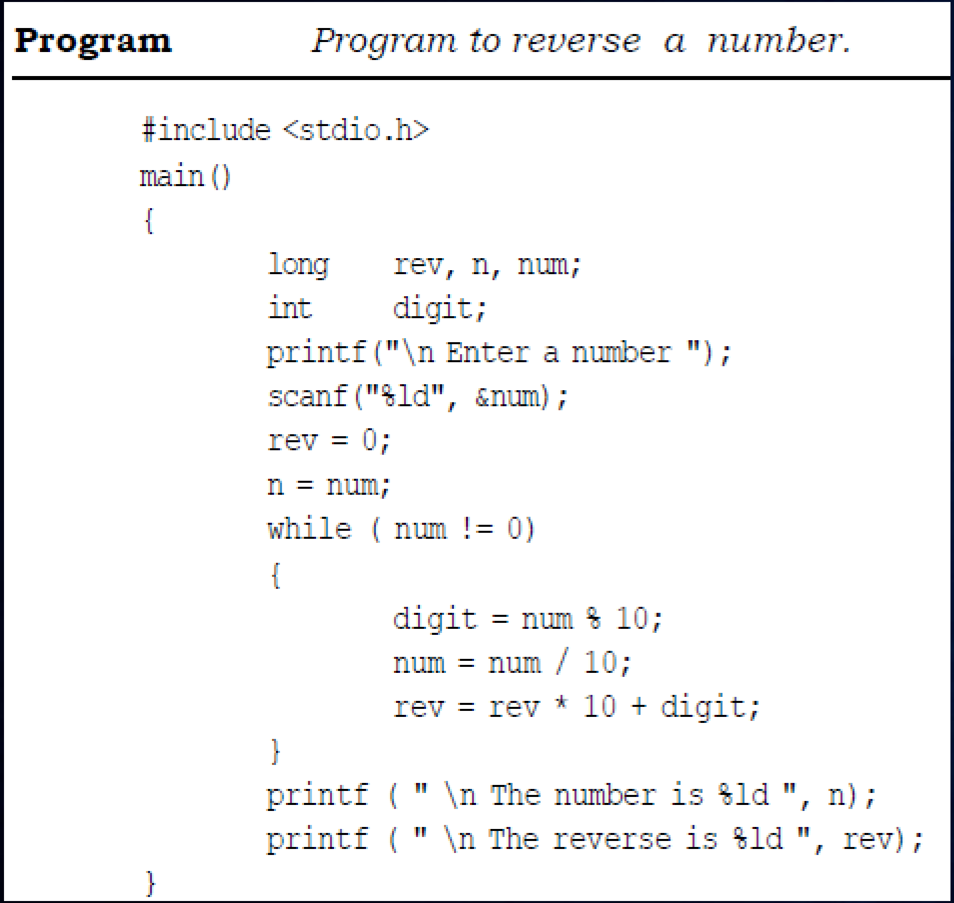
Nested While Loop Example

```
#include <stdio.h>
void main ()
{
        int i, n, in;
        printf ("ENTER A NUMBER ");
        scanf ("%d", &n);
                i = 1;
        while (i <= n)
        {
                printf ("\n");
                in = 1;
                while (in <= i)
                {
                        printf ("%d ", in);
                        in = in + 1;
                }
                i = i + 1;
        }
getch();
}
```



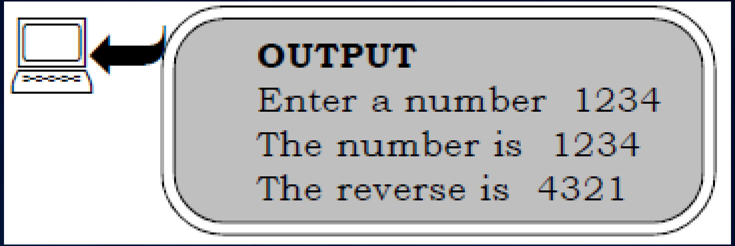
```
ENTER A NUMBER : 6
1
1  2
1  2  3
1  2  3  4
1  2  3  4  5
1  2  3  4  5  6
```

Reverse a Number using While Loop



Program *Program to reverse a number.*

```
#include <stdio.h>
main()
{
        long    rev, n, num;
        int     digit;
        printf("\n Enter a number ");
        scanf("%ld", &num);
        rev = 0;
        n = num;
        while ( num != 0)
        {
                digit = num % 10;
                num = num / 10;
                rev = rev * 10 + digit;
        }
        printf ( " \n The number is %ld ", n);
        printf ( " \n The reverse is %ld ", rev);
}
```

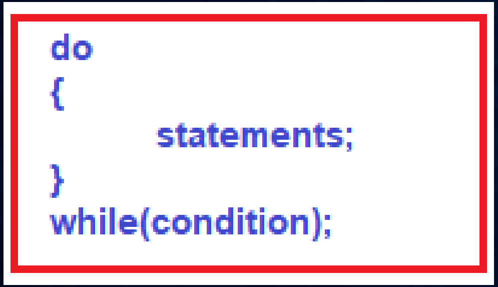


OUTPUT

```
Enter a number  1234
The number is  1234
The reverse is  4321
```

Do While Loop

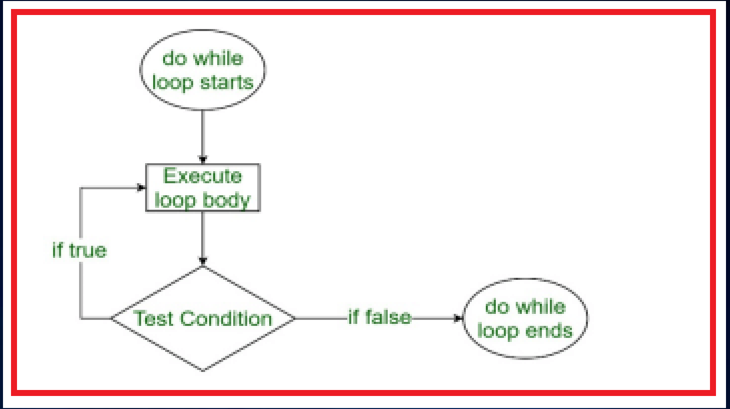
- The do-while loop is a post-tested loop. Using the do-while loop, we can repeat the execution of several parts of the statements.
- **It executes the statements unconditionally atleast once**
- The do-while loop is mainly used in the case where we need to execute the loop at least once.
- The do-while loop is mostly used in menu-driven programs where the termination condition depends upon the end-user
- **Syntax**



```
do
{
        statements;
}
while(condition);
```

Do While Loop

- Flowchart



- Example

```
#include <stdio.h>
void main()
{
int j=0;
            do
            {
                        printf("Value of variable j is: %d\n", j);
                        j++;
            }while (j<=3);
            getch();
}
```

Armstrong Number

Program *Program to find whether a given number is an Armstrong number. ($153 = 1^3 + 5^3 + 3^3$) i.e., the sum of the cubes of all the digits is equal to the original number.*

```
#include<stdio.h>
main()
{
        int     sum, n, num, digit;
        printf("\n Enter a number ");
        scanf("%d", &num);
        sum = 0;
        n = num;
        do
        {
                digit = num % 10;
                num = num / 10;
                sum = sum + digit * digit * digit ;
        } while ( num > 0);
        if ( sum == n)
                printf ( " \n %d is an Armstrong number ", n);
        else
                printf ( " \n %d is not an Armstrong number ", n);
}
```

OUTPUT

```
Enter a number  153
153  is an Armstrong number
Enter a number  193
193  is not an Armstrong number
```

Calculation of Commission of a Product using do-while

Program *A Company's salesmen sell toothpastes and soaps. The company gives 10% commission for toothpaste and 15% commission for soaps. Calculate and output the total commission for each salesman if required.*

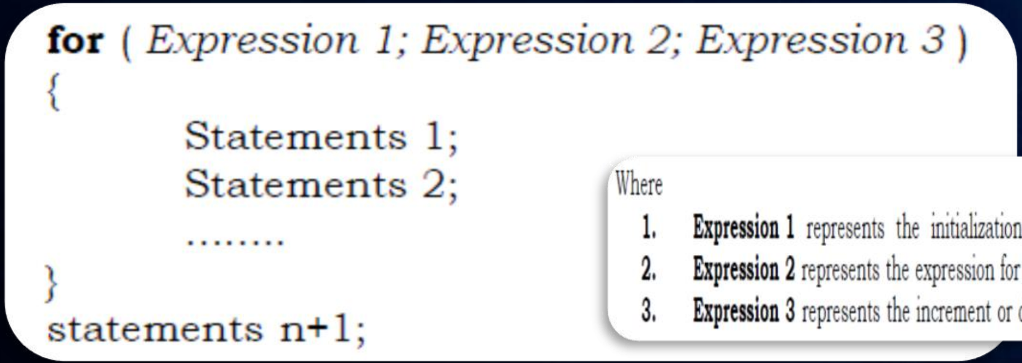
```
#include <stdio.h>
main()
{
        int     slno;
        float   paste, soap, comm;
        char    ch;
        do
        {
                printf(" \n Enter salesman number ");
                scanf("%d",&slno);
                printf(" \n Enter the total sales of toothpastes and soaps ");
                scanf("%f%f",&paste, &soap);
                comm = ( paste * 0.1 + soap * 0.15 ) ;
                printf(" \n Salesman number = %d", slno);
                printf(" \n Commission = %.2f ", comm);
                printf(" \n Continue ( Y / N ) ? ");
                ch=getchar();
        } while( ch == 'y' || ch == 'Y');
}
```

OUTPUT

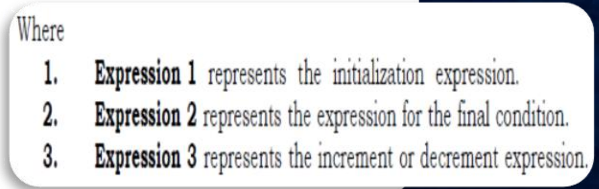
```
Enter salesman number   1743
Enter the total sales of toothpastes and soaps  2100   1700
Salesman number =  1743
Commission =  465.00
Continue ( Y / N ) ? N
```

For Loop

- ‘For Loop’ structure is normally used when we know exactly how many times a particular set of statements is to be repeated again and again.
- The for statement is a looping control structure which will execute a set of statements a specified number of times and automatically keep track of the number of ‘passes’ through the set of statements.
  - Syntax



```
for ( Expression 1; Expression 2; Expression 3 )
{
        Statements 1;
        Statements 2;
        ........
}
statements n+1;
```

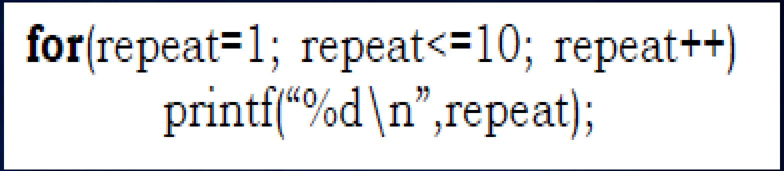


Where

1. **Expression 1** represents the initialization expression.
2. **Expression 2** represents the expression for the final condition.
3. **Expression 3** represents the increment or decrement expression.

For Loop

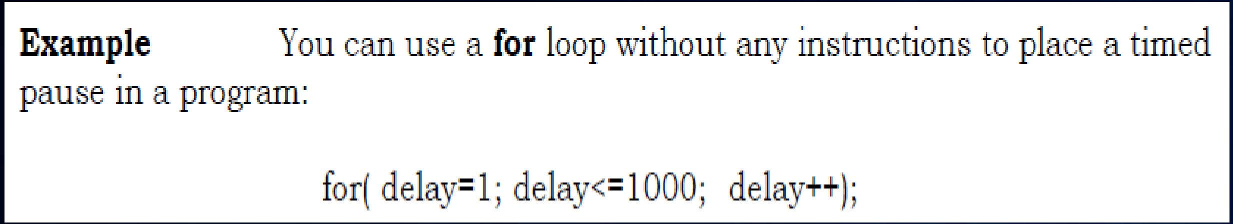
- **Syntax**
- For(Initialization; Condition; Increment/Decrement)
- {
-         Statement block;
- }



```
for(repeat=1; repeat<=10; repeat++)
    printf("%d\n",repeat);
```

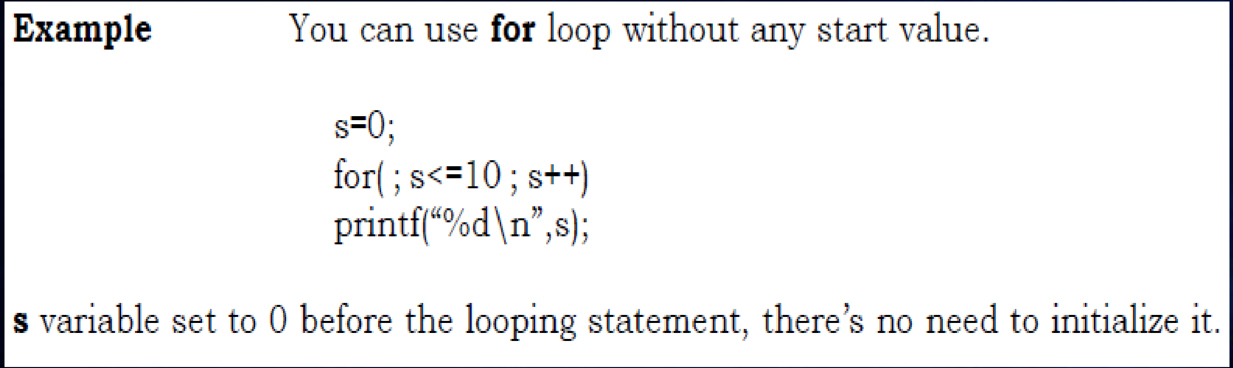
- **Example**
- for (k = 0 ; k < =100 ; k++)
- {
-         printf (" Welcome to the C World, Have fun");
- }

Example Program



Example You can use a **for** loop without any instructions to place a timed pause in a program:

```
for( delay=1; delay<=1000;  delay++);
```

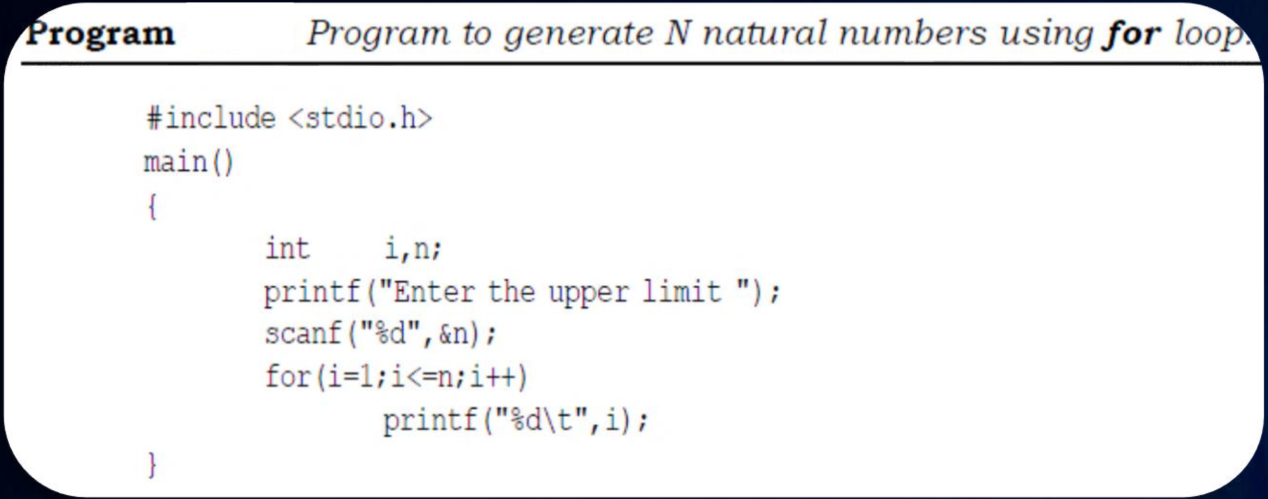


Example You can use **for** loop without any start value.

```
s=0;
for( ; s<=10 ; s++)
printf("%d\n",s);
```

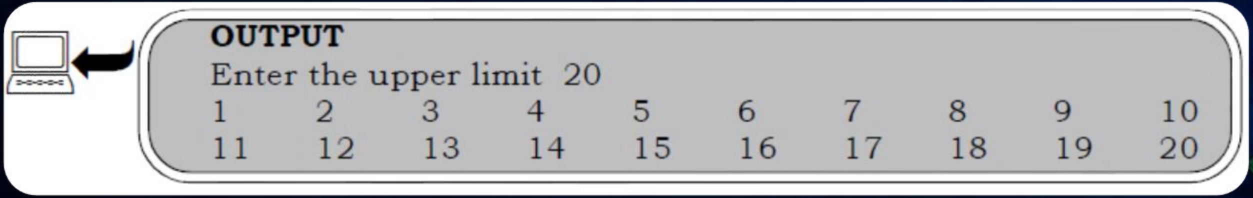
s variable set to 0 before the looping statement, there's no need to initialize it.

Example Program



Program *Program to generate N natural numbers using **for** loop.*

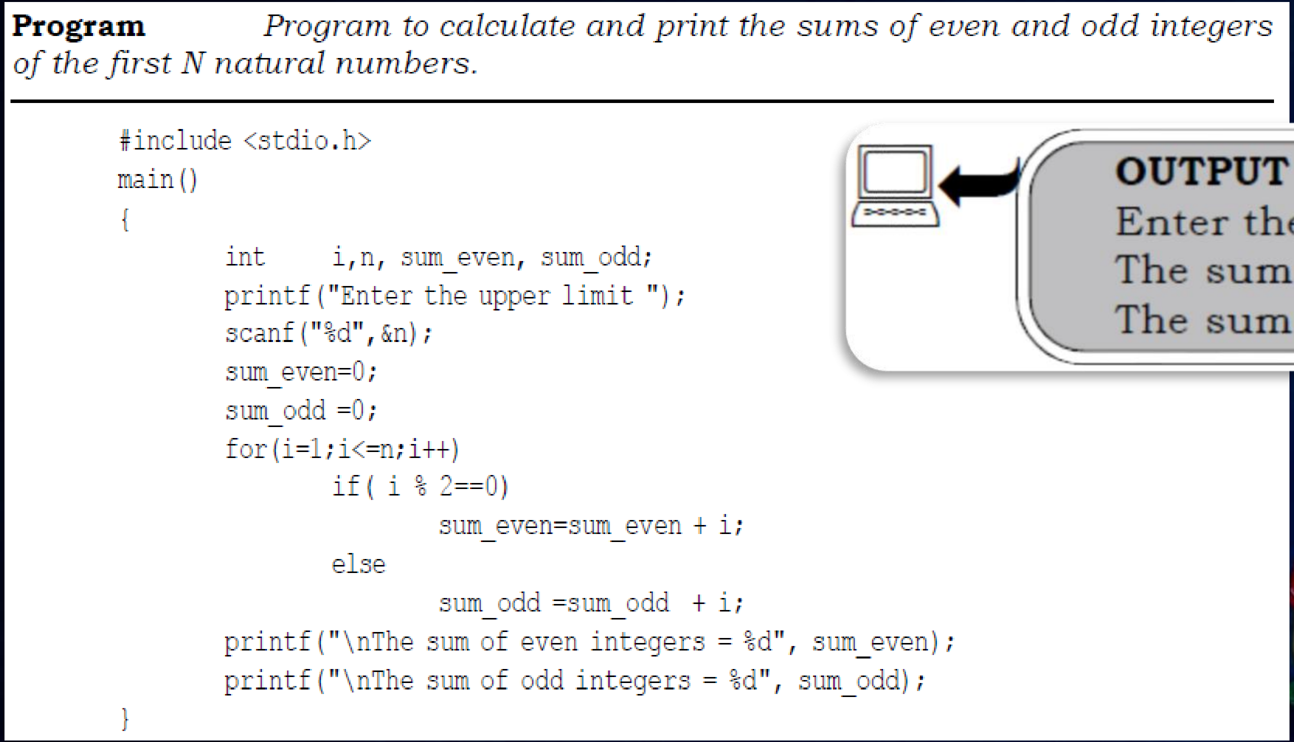
```
#include <stdio.h>
main()
{
        int     i,n;
        printf("Enter the upper limit ");
        scanf("%d",&n);
        for(i=1;i<=n;i++)
                printf("%d\t",i);
}
```



OUTPUT

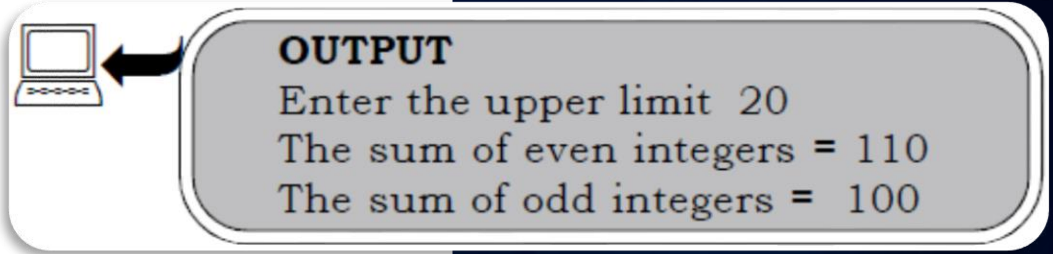
```
Enter the upper limit  20
1      2      3      4      5      6      7      8      9      10
11     12     13     14     15     16     17     18     19     20
```

Example Program



Program *Program to calculate and print the sums of even and odd integers of the first N natural numbers.*

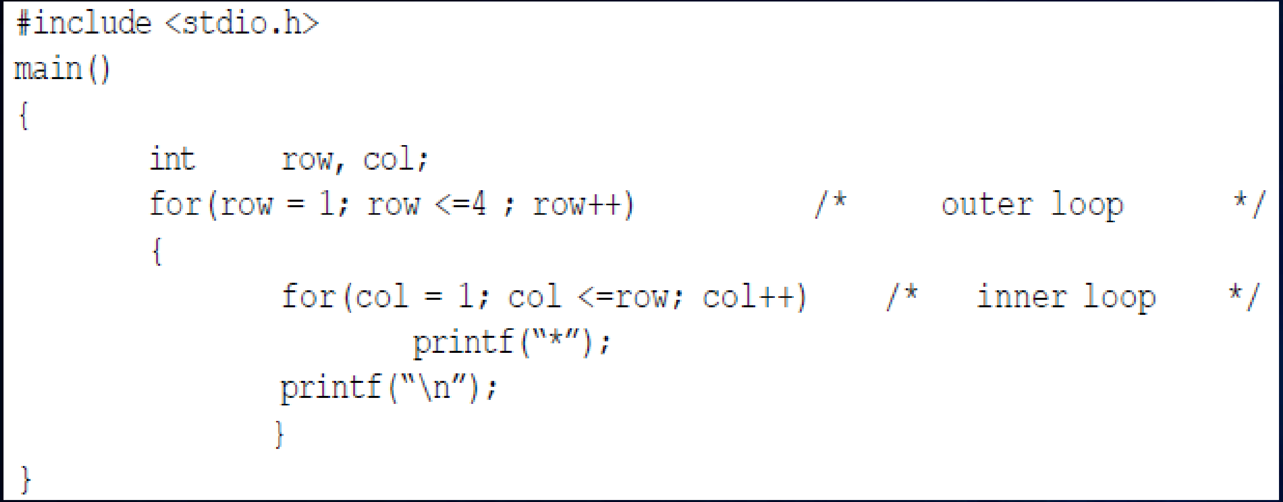
```
#include <stdio.h>
main()
{
        int    i,n, sum_even, sum_odd;
        printf("Enter the upper limit ");
        scanf("%d",&n);
        sum_even=0;
        sum_odd =0;
        for(i=1;i<=n;i++)
                if( i % 2==0)
                        sum_even=sum_even + i;
                else
                        sum_odd =sum_odd  + i;
        printf("\nThe sum of even integers = %d", sum_even);
        printf("\nThe sum of odd integers = %d", sum_odd);
}
```



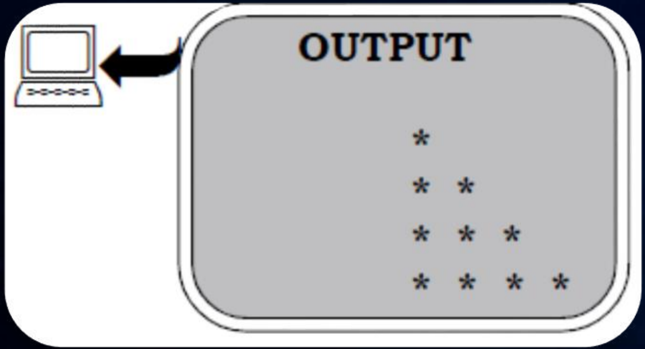
OUTPUT

```
Enter the upper limit  20
The sum of even integers = 110
The sum of odd integers =  100
```

Example Program



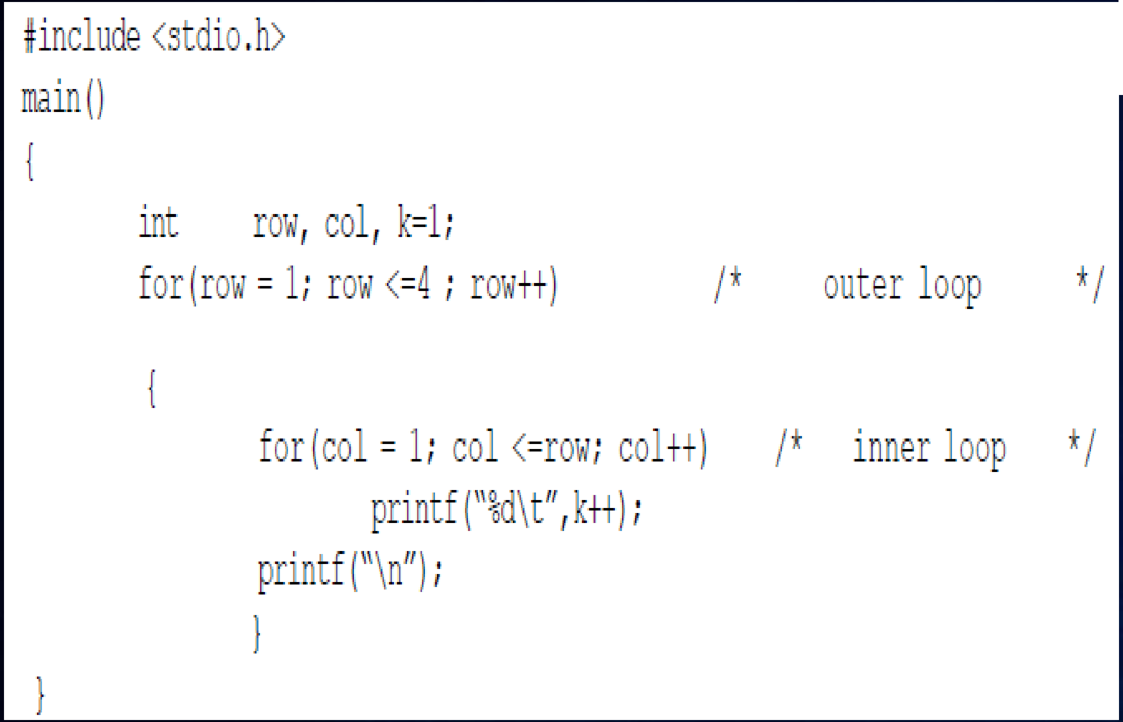
```
#include <stdio.h>
main()
{
        int     row, col;
        for(row = 1; row <=4 ; row++)          /*     outer loop      */
        {
                for(col = 1; col <=row; col++)     /*    inner loop    */
                        printf("*");
                printf("\n");
                }
}
```



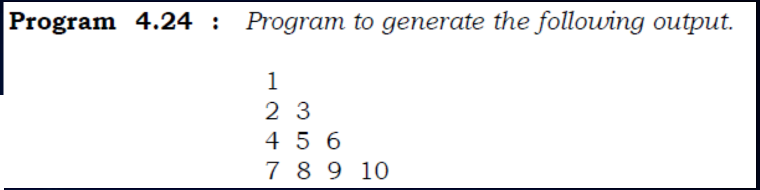
OUTPUT

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

Example Program

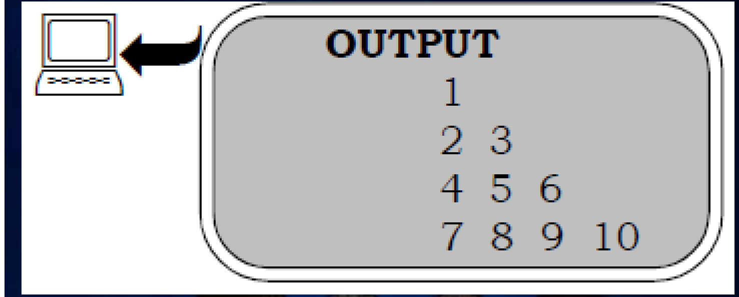


```
#include <stdio.h>
main()
{
        int     row, col, k=1;
        for(row = 1; row <=4 ; row++)          /*     outer loop      */

         {
                for(col = 1; col <=row; col++)    /*   inner loop    */
                        printf("%d\t",k++);
                printf("\n");
                }
 }
```



Program 4.24 : *Program to generate the following output.*

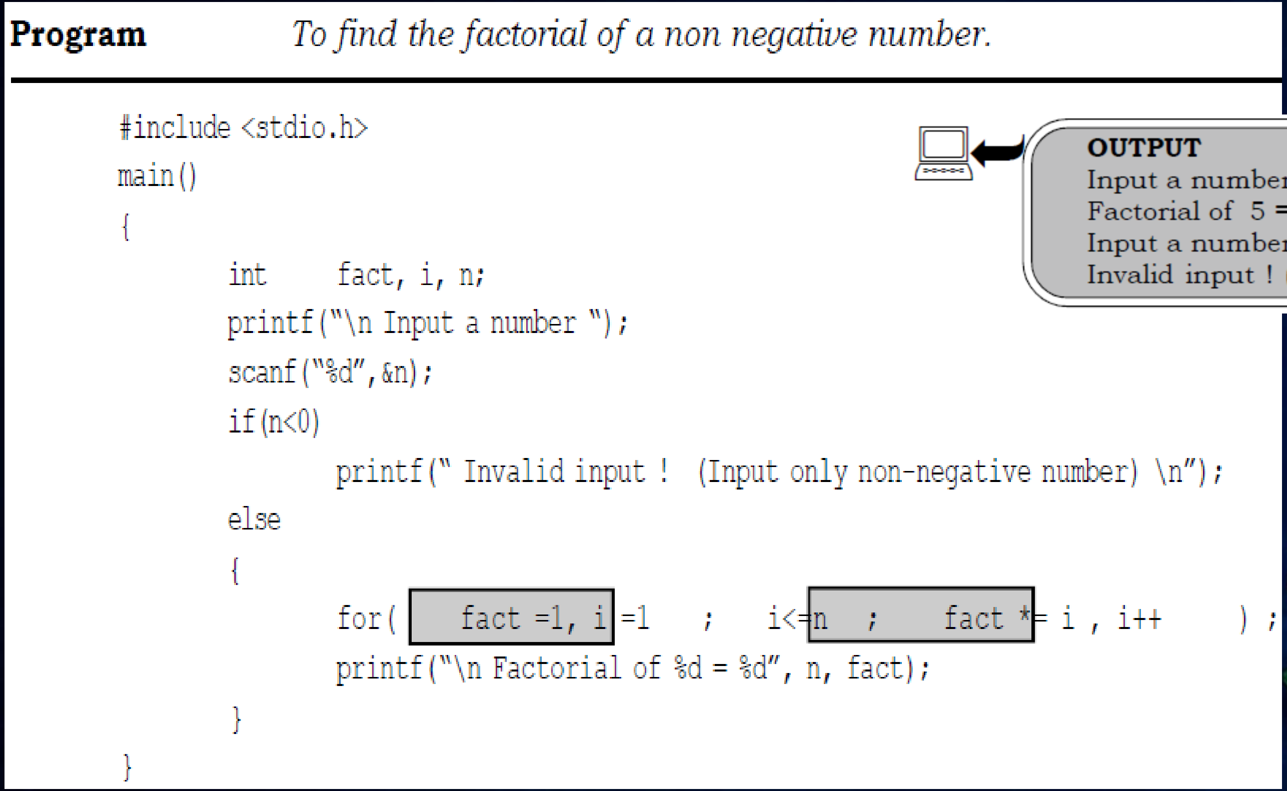
```
1
2  3
4  5  6
7  8  9  10
```



OUTPUT

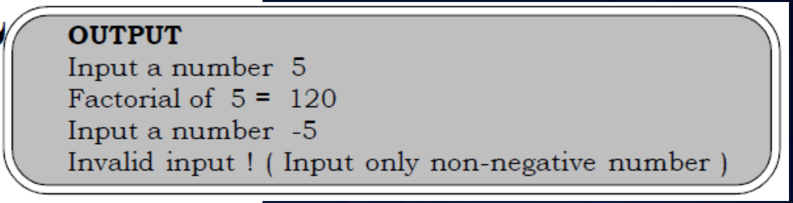
```
1
2  3
4  5  6
7  8  9  10
```

Example Program



Program *To find the factorial of a non negative number.*

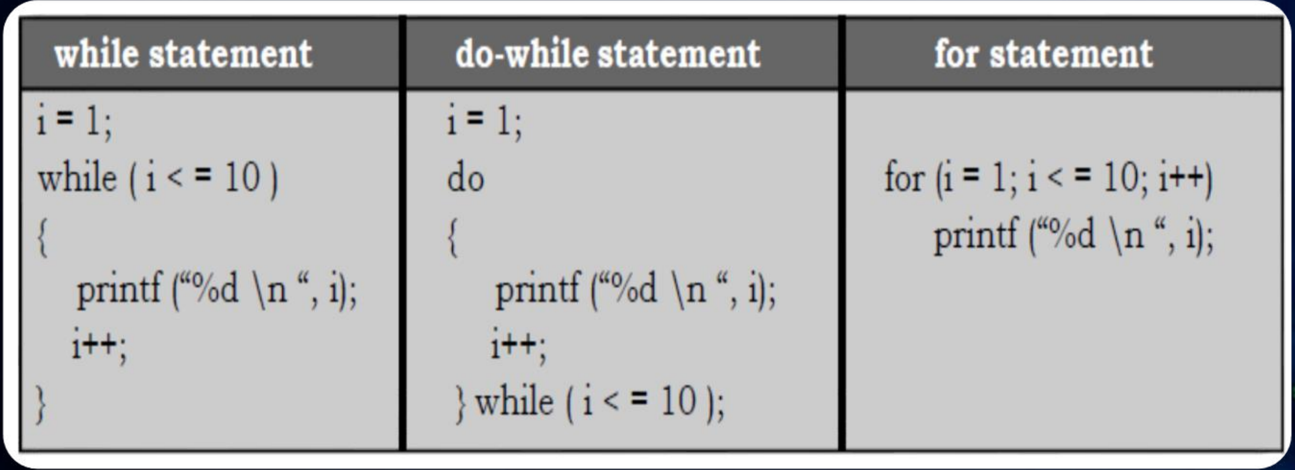
```
#include <stdio.h>
main()
{
        int     fact, i, n;
        printf("\n Input a number ");
        scanf("%d",&n);
        if(n<0)
                printf(" Invalid input !  (Input only non-negative number) \n");
        else
        {
                for(  fact =1, i =1   ;   i<=n  ;    fact *= i , i++     ) ;
                printf("\n Factorial of %d = %d", n, fact);
        }
}
```



```
OUTPUT
Input a number  5
Factorial of  5 =  120
Input a number  -5
Invalid input ! ( Input only non-negative number )
```

Comparison

- The **for loop** is appropriate **when you know in advance how many times the loop will be executed**. The other two loops while and do-while loops are more suitable in the situations where it is not known before-hand when the loop will terminate. The **while** should be preferred **when you may not want to execute the loop body even once**, and the **do-while** loop should be preferred **when you are sure to execute the loop body at least once**

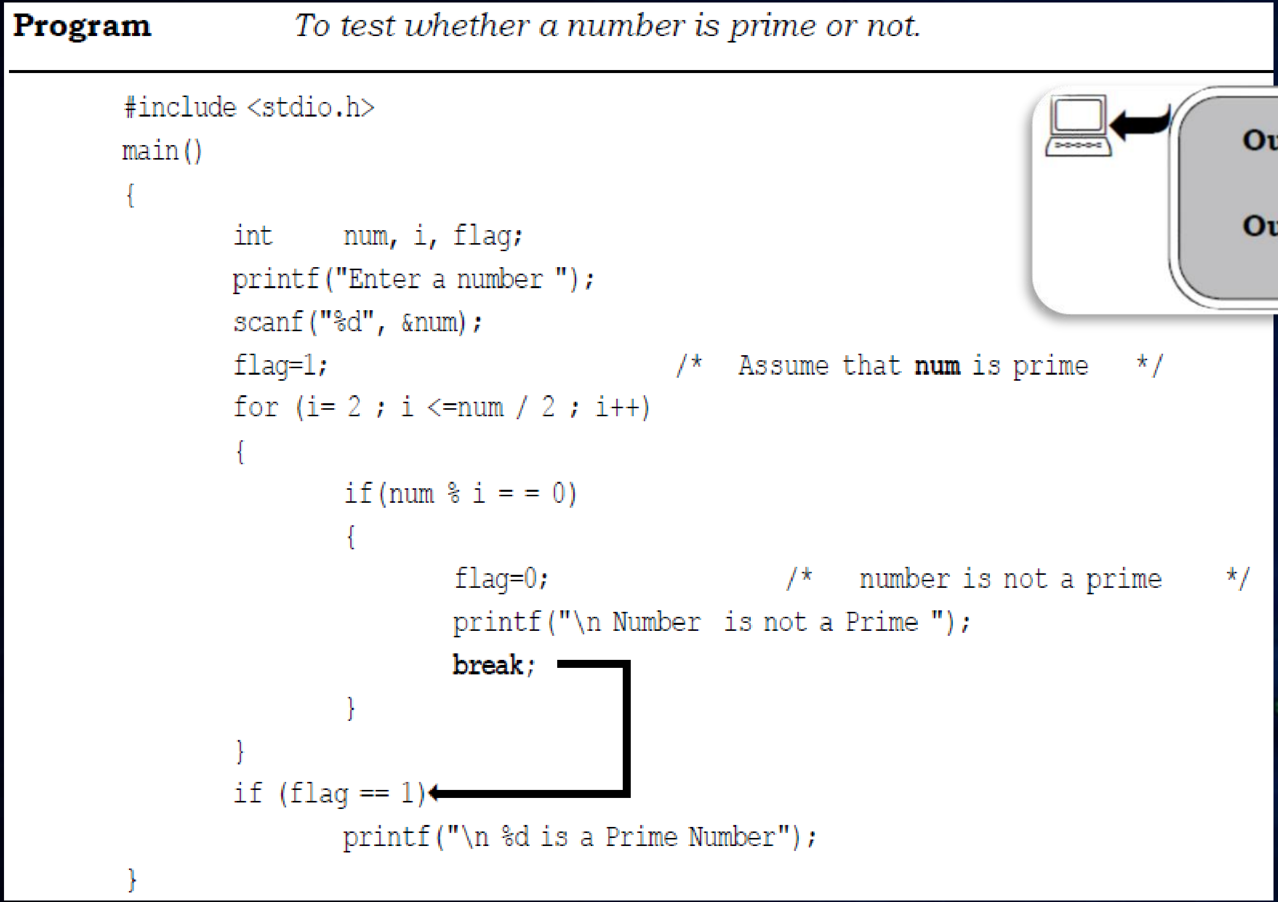


while statement	do-while statement	for statement
i = 1;	i = 1;	
while (i < = 10)	do	for (i = 1; i < = 10; i++)
{	{	printf ("%d \n ", i);
printf ("%d \n ", i);	printf ("%d \n ", i);	
i++;	i++;	
}	} while (i < = 10);	

Jumping Statements

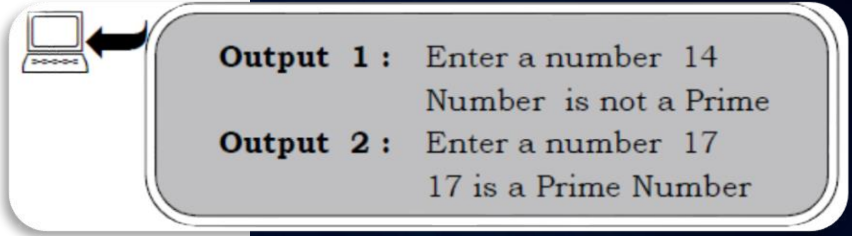
- The jump statements unconditionally transfer program control within a function
- C has four statements that perform an unconditional branching : **goto**, **return**, **break**, and **continue**.
- We can use **goto** and return statements any where in the program whereas **break and continue are used inside the loops**.
- In addition to the above four, C provides a standard library function **exit( )** that helps you break out of a program

Break Statement



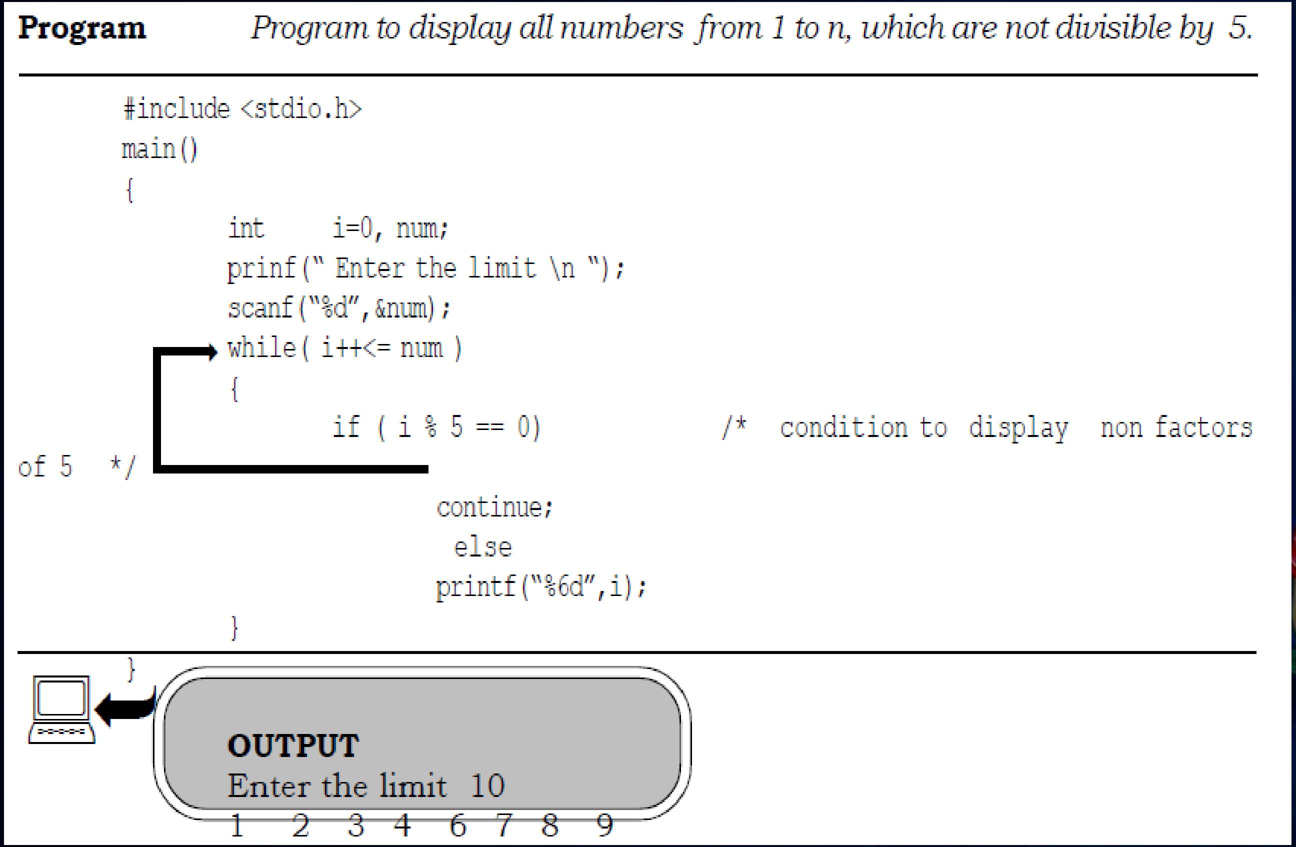
Program *To test whether a number is prime or not.*

```
#include <stdio.h>
main()
{
        int     num, i, flag;
        printf("Enter a number ");
        scanf("%d", &num);
        flag=1;                         /*  Assume that num is prime   */
        for (i= 2 ; i <=num / 2 ; i++)
        {
                if(num % i = = 0)
                {
                        flag=0;                 /*   number is not a prime     */
                        printf("\n Number  is not a Prime ");
                        break;
                }
        }
        if (flag == 1)
                printf("\n %d is a Prime Number");
}
```



```
Output  1 :   Enter a number  14
              Number  is not a Prime
Output  2 :   Enter a number  17
              17 is a Prime Number
```

Continue Statement



Program *Program to display all numbers from 1 to n, which are not divisible by 5.*

```
        #include <stdio.h>
        main()
        {
                int      i=0, num;
                prinf(" Enter the limit \n ");
                scanf("%d",&num);
                while( i++<= num )
                {
                        if ( i % 5 == 0)              /*  condition to  display   non factors
of 5   */
                                continue;
                                 else
                                printf("%6d",i);
                }
        }
```

OUTPUT

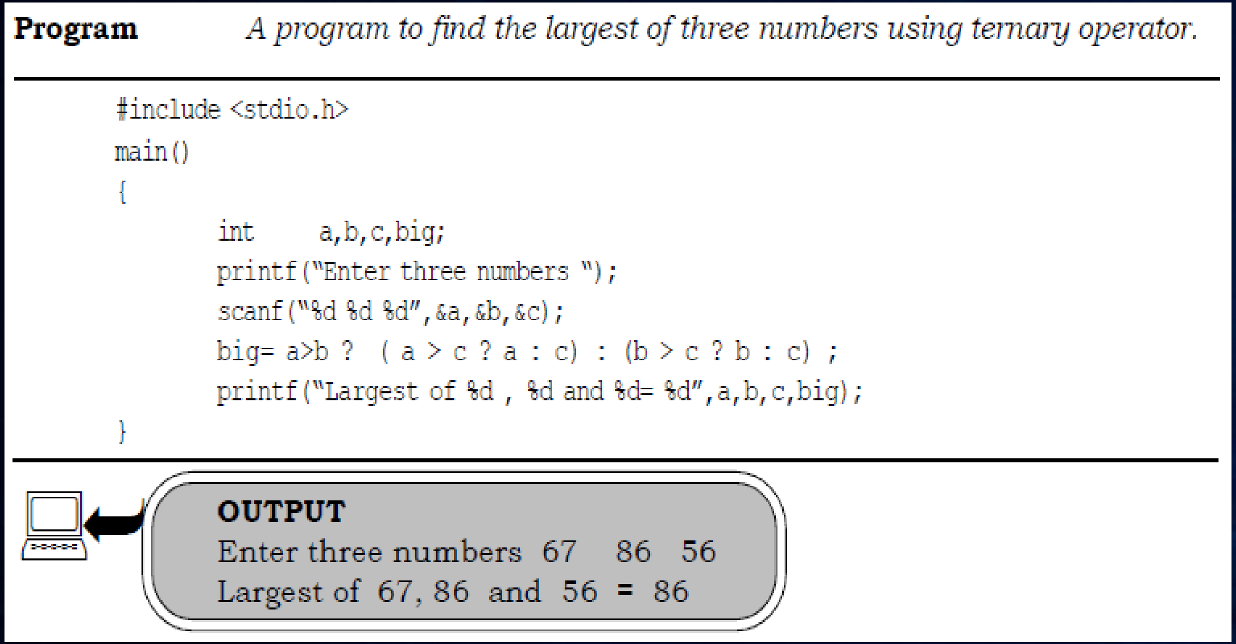
```
Enter the limit  10
1   2   3  4   6  7  8   9
```

exit()

```
#include <stdio.h>
#include <stdlib.h>             /*     For  exit()  function        */
main()
{
        int     num, i;
        printf("Enter a number ");
        scanf("%d", &num);
        for (i= 2 ; i <=num / 2 ; i++)
                if(num % i = = 0)
                {
                          printf("\n Number  is not a Prime ");
                        exit(0);
                }
        }
        printf("\n %d is a Prime Number");
}
```

```
OUTPUT
Enter a number  14
Number  is not a Prime
```

Case Study



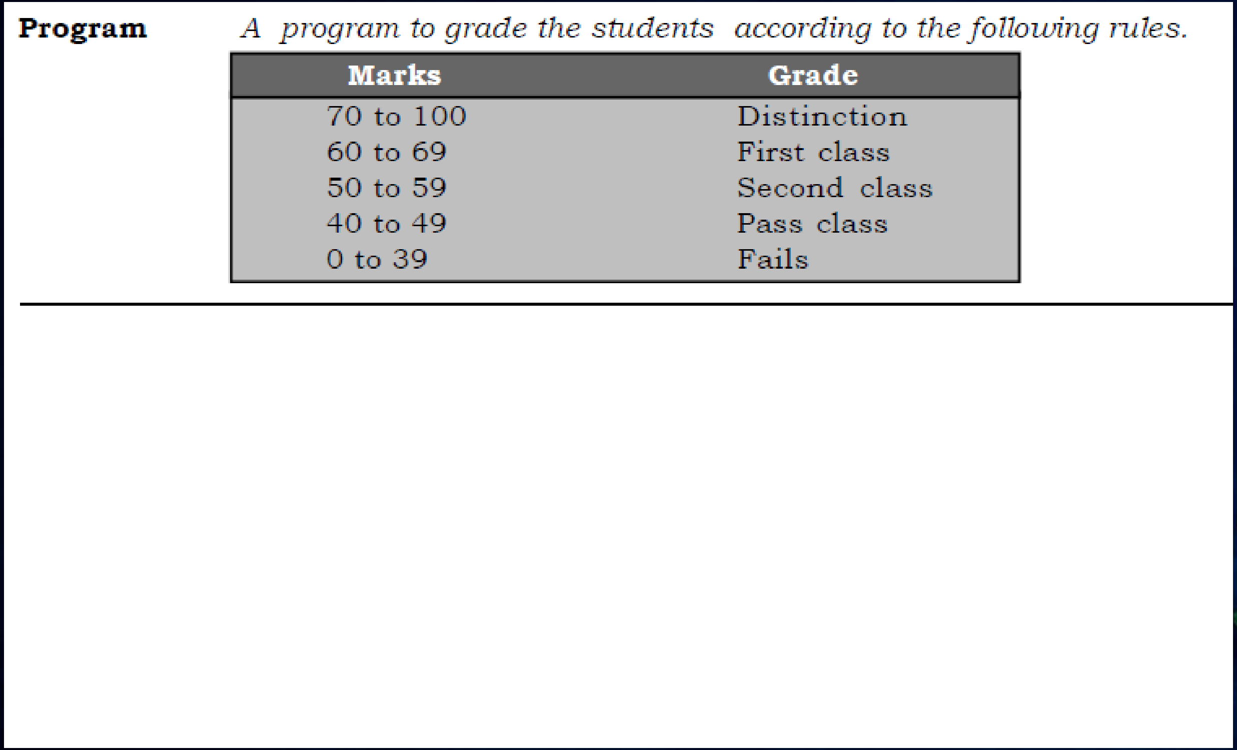
Program *A program to find the largest of three numbers using ternary operator.*

```
#include <stdio.h>
main()
{
        int     a,b,c,big;
        printf("Enter three numbers ");
        scanf("%d %d %d",&a,&b,&c);
        big= a>b ?  ( a > c ? a : c) : (b > c ? b : c) ;
        printf("Largest of %d , %d and %d= %d",a,b,c,big);
}
```

OUTPUT

```
Enter three numbers  67   86   56
Largest of  67, 86  and  56  =  86
```

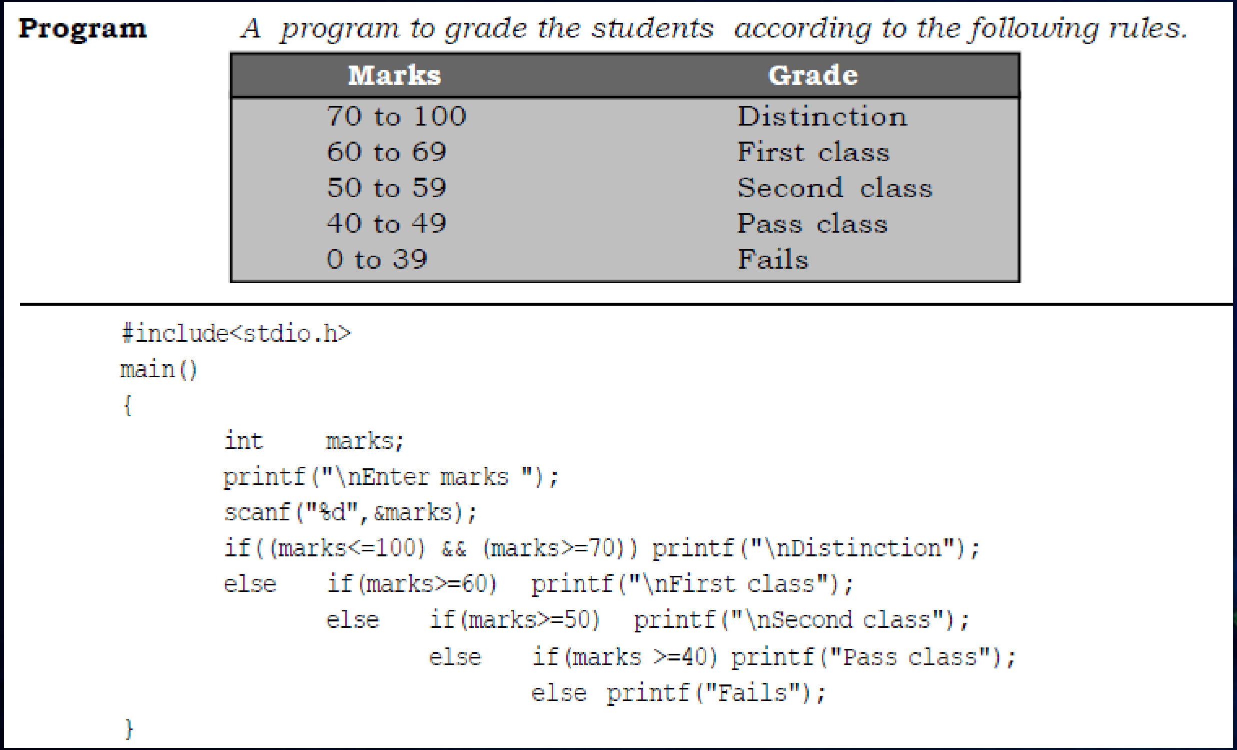
Case Study



Program *A program to grade the students according to the following rules.*

Marks	Grade
70 to 100	Distinction
60 to 69	First class
50 to 59	Second class
40 to 49	Pass class
0 to 39	Fails

Case Study



Program *A program to grade the students according to the following rules.*

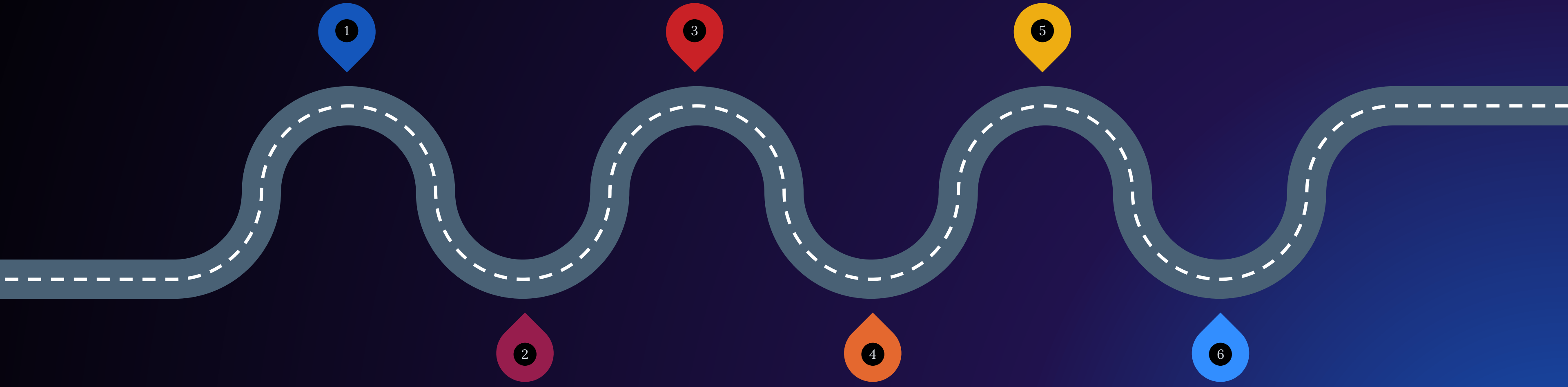
Marks	Grade
70 to 100	Distinction
60 to 69	First class
50 to 59	Second class
40 to 49	Pass class
0 to 39	Fails

```
#include<stdio.h>
main()
{
        int    marks;
        printf("\nEnter marks ");
        scanf("%d",&marks);
        if((marks<=100) && (marks>=70)) printf("\nDistinction");
        else    if(marks>=60)   printf("\nFirst class");
                else    if(marks>=50)   printf("\nSecond class");
                        else    if(marks >=40) printf("Pass class");
                                else  printf("Fails");
}
```

Credits

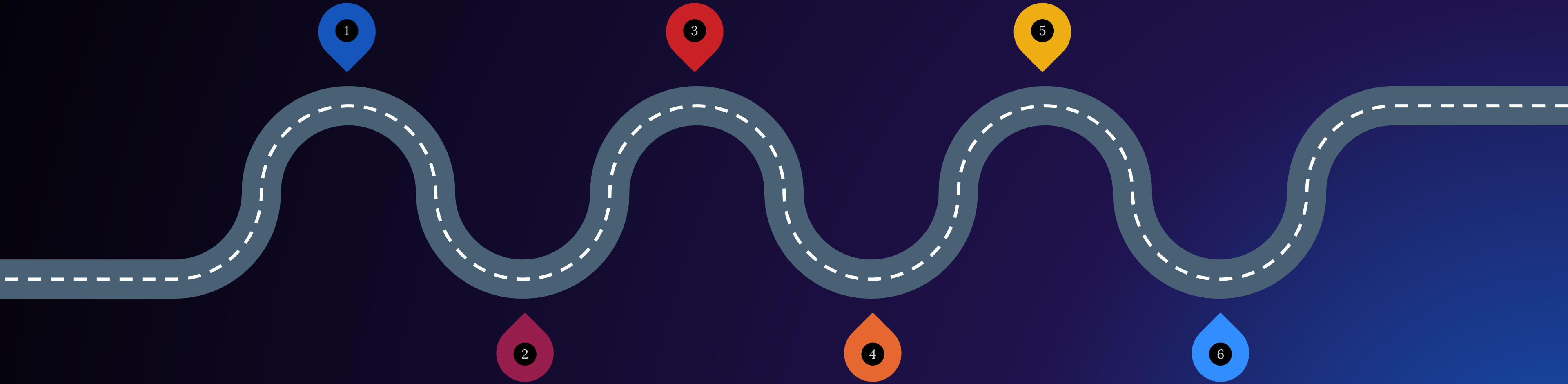
- https://prepinsta.com/c-program/introduction-to-branchingandLooping/

Roadmap for Unit II



1. Branching Statements, Looping Statements Nested
2. Arrays One Dimensional and Two/Multi Dimensional
3. Strings and its Functions
4. Advanced Features, Type Casting and Type Conversion
5. Formatted and Unformatted Input and Output
6. Summary

Roadmap to Programming in C



Unit I
Computer Based Problem Solvingand Overview of C

Unit – II
Branching, Looping, Arrays, Strings

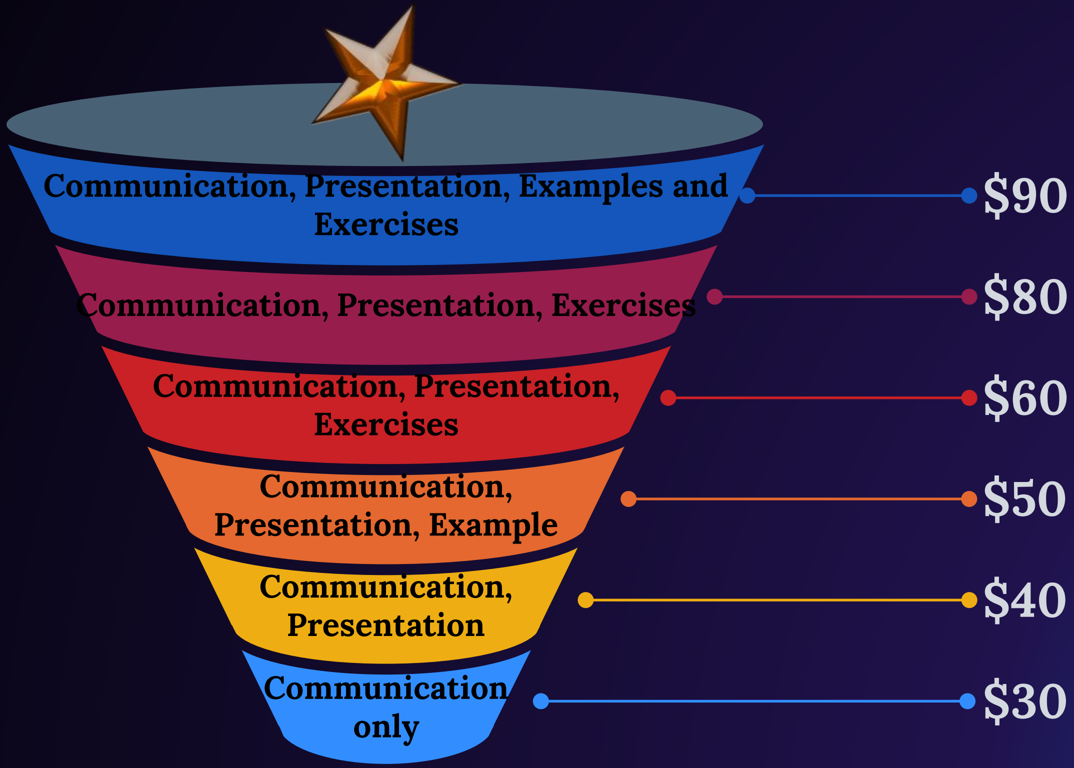
Unit III
Functions and Recursions

Unit IV
Dynamic Data Structures, Pointers, Union

Unit V
File Handling, Preprocessor, Standard Library and Header Files

Revision

Feedback – Rating Star


Communication, Presentation, Examples and Exercises
$90
Communication, Presentation, Exercises
$80
Communication, Presentation, Exercises
$60
Communication, Presentation, Example
$50
Communication, Presentation
$40
Communication only
$30

Thanks!

Any questions?

You can find me at:

- kramdharma@gmail.com