

Unit 3

Control Structures

Introduction:

C Program consists several set of instructions. Control structure is the set of instructions which are used to monitor, regulate or control the flow of instructions in the program. There are three types of control structures used in C programming. They are:

- a) Sequential structure
- b) Selection structure
- c) Iteration structure

Sequential Control Structure

The sequential structure consists of a sequence of programming statements (instructions) which are executed one after another in a linear order. In this structure, the first statement is executed first, after completion of the statement; second statements are executed and so on all the statements of the program are executed in their turn. The sequential order is described in following and algorithm

Step 1: Statement 1
Step 2: Statement 2
Step 3: Statement 3
.....
.....
.....
Step N: Statement N

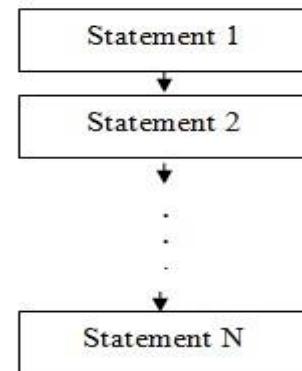


Fig: Flowchart- Sequential Structure

Fig: Algorithm- Sequential structure

```
/*Program to calculate simple interest and mixed amount*/
#include<stdio.h>
void main()
{
int p; float t,
r, i, ma;
printf("\n Input Principal Amount:- ");
scanf("%d", &p);
printf("\n Input Time Period:- ");
scanf("%f", &t);
printf("\n Input Rate of Interest:- ");
scanf("%f", &r); i=(p*t*r)/100;
ma=p+i;
printf("\n Simple interest = Rs %10.2f",i);
printf("\n Mixed Amount = Rs %10.2f",ma);
}
```

Output:

Input Principal Amount: - 1000
Input Time Period: - 2
Input Rate of Interest: - 3

Simple interest = Rs 60.00 Mixed

Amount = Rs 1060.00

Selection Control Structure

The decision making structure is also called selective structure. In this structure either certain block of statements are executed or skipped according to the set of condition applied. In C language there are four keywords: *if*, *if-else*, *if-else if* and *switch-case* for selection control structure and those are discussed in following section.

a) *if* statement

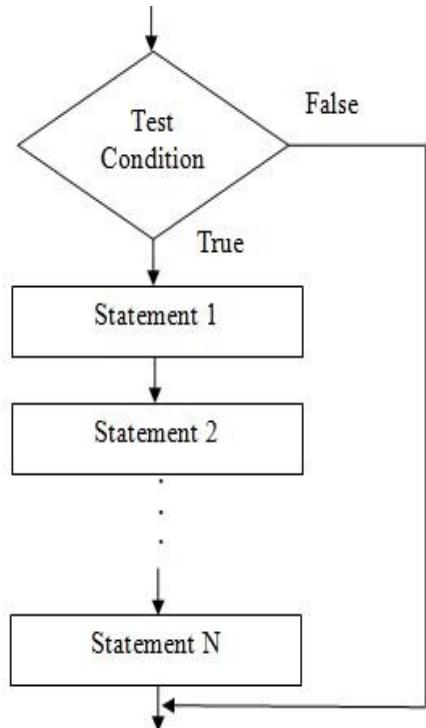
It is simplest form of selection. In this structure only single condition is applied but its alternative is not mentioned. The block of statements associated with the „if“ are executed in sequential order if the given condition has been satisfied otherwise those statements are skipped and the immediate next statements are executed. The syntax and flowchart is shown as below.

Syntax:

```
if (test_condition)
{
    Statement 1;
    Statement 2;
    .....
    Statement N;
}
```

Example: void main()

```
{
    int salary; float bonus;
    printf("\n Input Salary:- ");
    scanf("%d", &salary);
    if(salary>10000)
    {
        bonus=salary*0.10;
        printf("\n Bonus = %10.2f", bonus);
    }
}
```



b) *if-else* statement

In this structure both alternatives are also provide. It is used to carry out a logical test and then take one of two possible actions depending on the outcome of the test. The test condition is applied within the “*if*” and if it is true then the block of statements associated with it are executed in linear order and after completion the control is automatically transferred to the next immediate statement after the structure otherwise the “*else*” part is activated and the statements along with it are executed.

The syntax and flowchart is shown as below.

Syntax:

```
if (test_condition)
{
    Statement T1;
    Statement T2;
    .....
    Statement TN;
```

```

    } else
{
    Statement F1;
    Statement F2;
.....
.....
Statement FN;
}

```

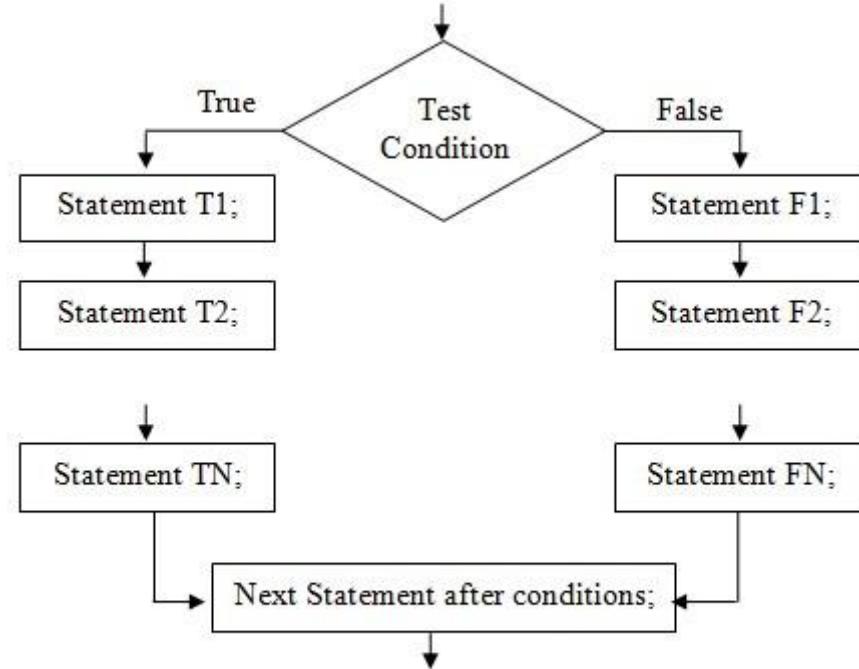


Fig: Flowchart- *if.... else* structure

Example:

```

void main()
{
int salary;
float bonus;
printf("\n Input Salary:- ");
scanf("%d",&salary);
if(salary>10000)
{
    bonus=salary*0.10;
    printf("\n Bonus = %10.2f",bonus);
}
else
{
    bonus=salary*0.5;      printf("\n
Bonus = %10.2f",bonus);
}
}

```

Nested if-else statement

It is a complex structure, where the *if-else* statements could be either within *if* block, *else* block or in both blocks. The inner conditional expressions are said to be nested within outer one. It allows us to check

for multiple test expressions and executes different codes for more than two conditions. The general format of *nested if-else* is:

Syntax;

```
if (condition1)
{
    if (condition2)
    {
        Statement 1;
        Statement 2;

        Statement N;
    }
} else
{
    if (condition3)
    {
        Statement A;
        Statement B;

        Statement Z;
    }
}
else
{
    Statement a;
    Statement b;

    Statement z;
}
```

Example:

```
/*Greater number among three numbers*/
#include<stdio.h>
void main()
{
int a,b,c;
printf("\n Input First Number:- ");
scanf("%d", &a);
printf("\n Input Second Number:- ");
scanf("%d", &b);
printf("\n Input Third number:- ");
scanf("%d", &c); if(a==b && a==c)
printf("All are equal numbers"); else
{
if(a>b)
{
if(a>c)
    printf("%d is greatest number", a);
else
    printf("%d is greatest number", c);
}
else
{
```

```

if(b>c)
    printf("%d is greatest number", b);
else
    printf("%d is greatest number", c);
}
}
}

```

c) if-else if..... else statement

It is used for multiple conditions. It is also known as *multi-way if else* or *if else ladder*. If any one condition among all of them is true then rest of all will be false. Then if all of the given condition are not true, then the *else* statement will be executed. The general format of *nested if-else* is:

Syntax;

```

if (condition1)
{
    Statement 1;
    Statement 2;

    Statement N;
}

else if (condition2)
{
    Statement A;
    Statement B;

    Statement Z;
}

. . else if
(condition N)
{
    Statement a;
    Statement b;

    Statement z;
} else
{
    Statement1;
    Statement2;
.....
    Statement N;
}

```

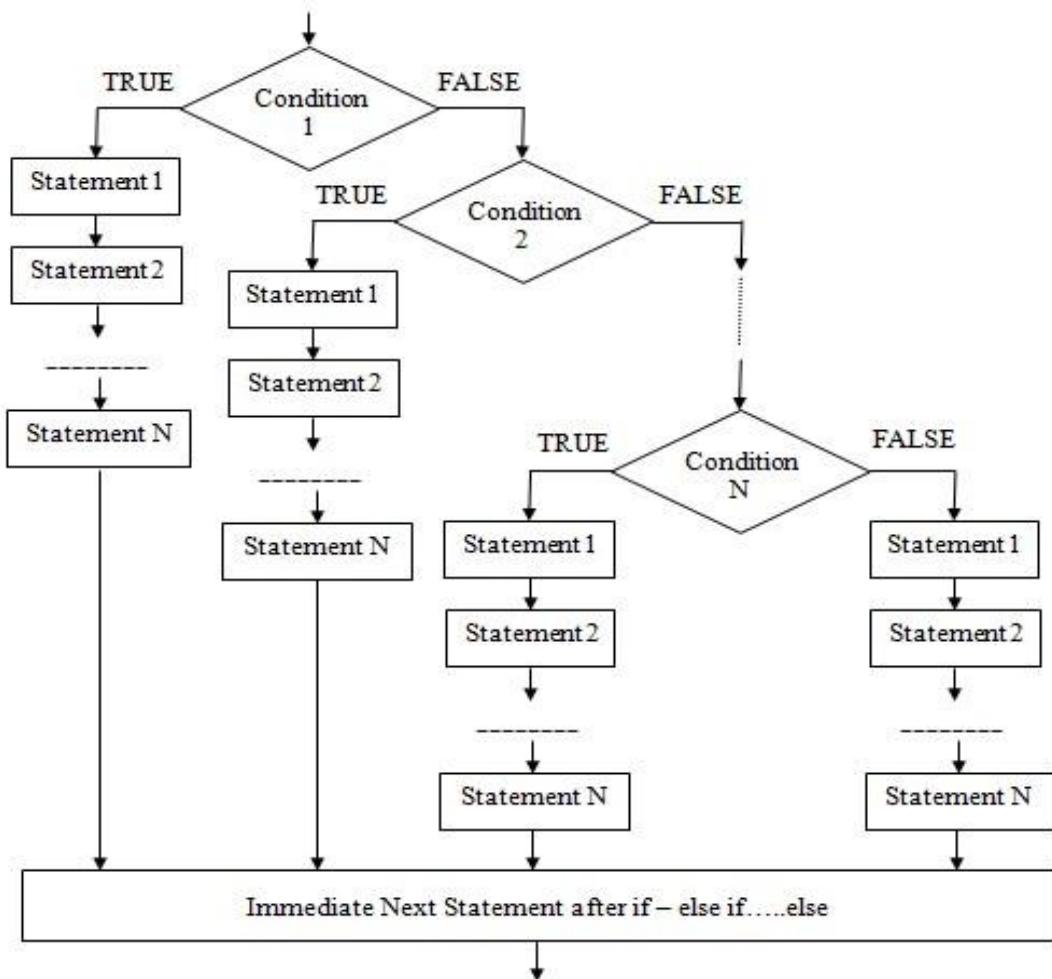


Fig: Flowchart – if else if else

Example:

```

/*Greater number among three numbers*/
#include<stdio.h>
void main()
{
int a, b, c;
printf("\n Input First Number:- ");
scanf("%d", &a);
printf("\n Input Second Number:- ");
scanf("%d", &b);
printf("\n Input Third number:- ");
scanf("%d", &c); if(a==b && a==c)
printf("All are equal numbers");
else if(a>b && a>c)
printf("%d is greatest number", a);
else if(b>a && b>c)
printf("%d is greatest number", b);
else
printf("%d is greatest number", c);
}
  
```

d) switch case statement

The control statement that allows us to make a decision from a number of choices is called a *switch*. Switch is a mechanism of jumping into a series of statements, the exact starting point depending on the value of

the expression. The *switch* expression is evaluated, and then the flow of control jumps to the matching *const – expression case*. The case expressions are typically *int* or *char* constants. If no match is found, the statements following the *default* are executed.

Syntax

```
switch (expression)
{
    case value 1:
        Block of statements;
        exit;
    case value 2:
        Block of statements;
        exit;
    .
    .
    .
    case value n:
        Block of statements;
        exit;

    default:
        do this;
}
```

Each constant need its own *case* keyword and trailing colon (:). Once execution has jumped to a particular case, the program will keep running through all the cases from that point down, called “*fall through*”. So, the process of executing statements in subsequent case clauses is called *fall through*. To prevent fall through, *break* statements can be used, which cause an immediate exit from the switch statement.

Example:

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

```
    printf("\n Invalid Choice");
}
}
```

Looping Control Structure:

The mechanism which repeats some portion of instructions either a specified numbers of times or until a certain condition has been satisfied is called *loop*. This repetitive operation is done through a loop control structure.

There are two types of loops in a program. They are: *finite* and *infinite* loop. The loop which has termination part is known as finite loop where as the loop which doesn't have an ending part is called infinite loop.

Generally each loop has three parts: initial value, loop condition and loop's increment or decrement part. There are three methods through which we can repeat a part of program. They are: a) using **for** loop

- b) using **while** loop
- c) using **do.... while** loop

The **for** loop

It is a pre-test or entry control loop, in which a statement or block of statements are executed for certain number of times. A *for* loop allows us to specify three things about a loop in a single line and they are:

- Loop's initial value i.e. to set loops starting value.
- Test condition i.e. to determine the number of repetitions
- increment or decrement value (loop update)

In this structure, the condition is tested at first, if it is satisfied then only the block of statements associated with the loop are executed till the given condition has been true, otherwise those statements are skipped and the control is automatically transferred to the next immediate statement after the loop. The syntax and flowchart is shown as below.

Syntax:

```
for (initial value; test condition; loop update)

{
Statement 1;
Statement 2;
.....
.....
.....
Statement N;
}
```

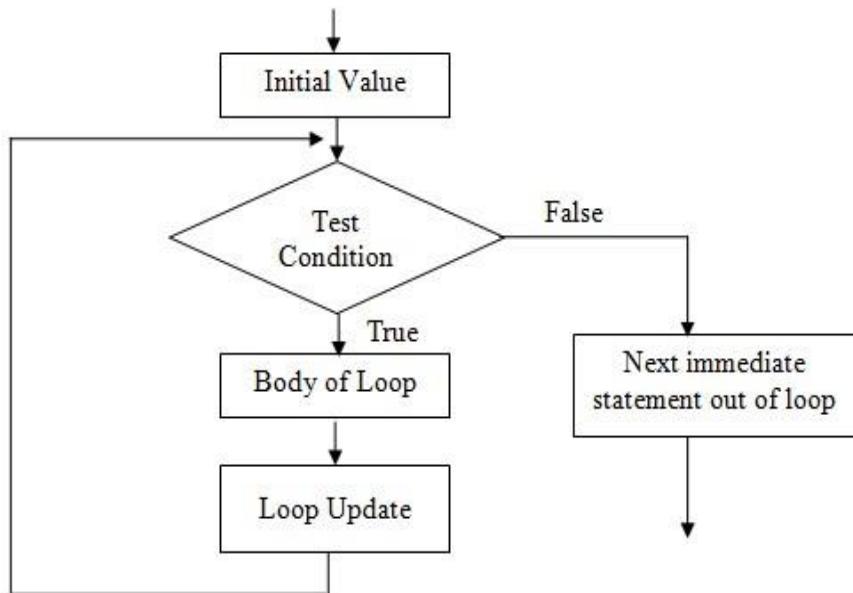


Fig: Flowchart – **for** loop

Example:

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

Output:

0 1 2 3 4

while loop

It is used to repeat a statement or number of statements for specified number of times or certain condition has been satisfied. The statements within the **while** loop gets executed till the condition remains true. When the condition becomes false, the control is passed to the first statement out of the loop. It is also known as *entry control or pre-test* loop because the condition is applied at the top of the loop before the control enters or passes inside the body of the loop. It will not execute its statements if the condition fails for the first time. The execution of a **while** loop is shown in following figure.

Syntax:

```
Initial value of loop; while
(test condition)
{
    Statement 1;
    Statement 2;
    .....
    .....
    .....
    Statement N;
    Loop Update;
}
```

Flowchart:

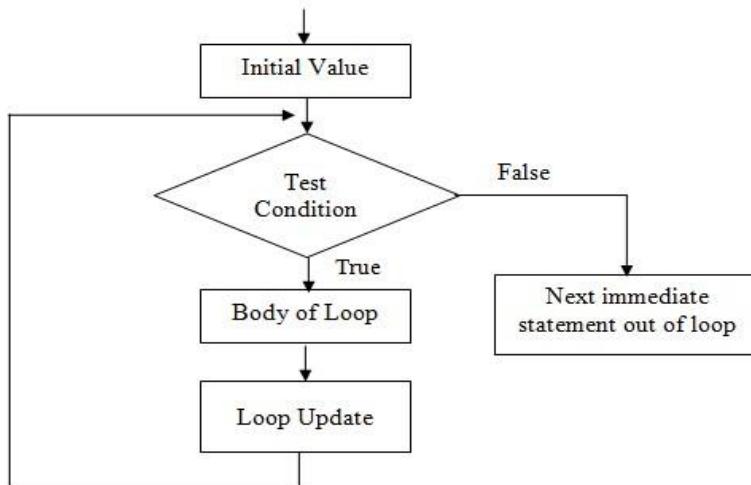


Fig: flowchart- **while** loop

Example:

```
/*Illustration of while loop*/
#include<stdio.h>
#include<conio.h>
void main()
{
int i=1;
clrscr();
while(i<=10)
{
printf("%5d",i); i=i+1;
}
getch(); }
```

Output:

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

do.... while loop

It is also used to repeat a statement or number of statements for specified number of times or certain condition has been satisfied. The statement within the **do-while** loop gets executed till the condition remains true. When the condition becomes false, the control is passed to the first statement out of the loop. The minor difference between **while** and **do-while** loop is- placement of the condition. In **do-while** loop the condition is placed at the bottom of the loop and it is tested after the statements have been executed. This means that **do-while** loop will execute its statements at least once, even if the condition fails for the first time. So it is known as *exit control or post test* loop. The execution of **do-while** loop is shown as:

Syntax:

```
Initial value of loop; do
{
Statement 1;
Statement 2;
.....
.....
Statement N;
Loop Update;
} while (test condition);
```

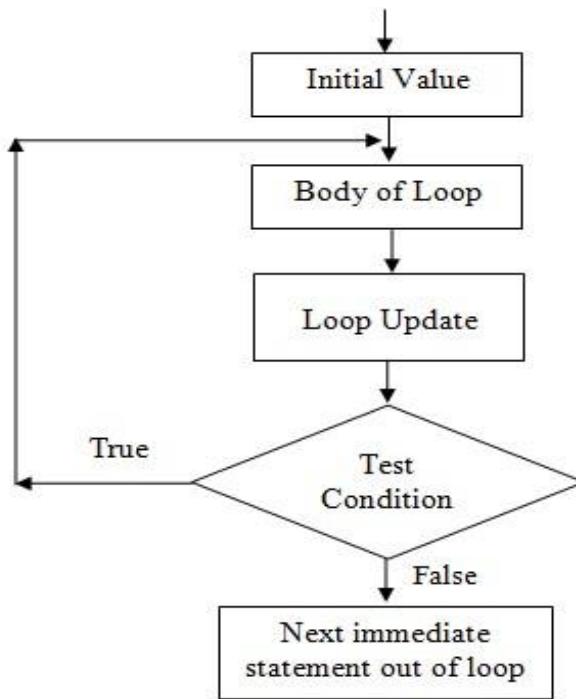


Fig: Flowchart- **do... while** loop

Example:

```

/*Illustration of do-while loop*/
#include<stdio.h>
void main()
{
int i=1;
do
{
printf("%5d",i);
i=i+1;
}while(i<=10);
getch();
}

```

Output:

1 2 3 4 5 6 7 8 9 10

Comparison between while and do-while loop.

while loop	do – while loop
It is used to repeat a statement or number of statements for fixed number of times or until the given condition has been satisfied.	It is also used to repeat a statement or block of statements for number of times or until the given condition has been satisfied.
It is also known as pre-test or entry control loop.	It is also known as post-test or exit control loop.
The condition is tested at the beginning of the loop so the numbers of statements may not be executed for a time if the given condition is false for the first time.	The condition is tested at the bottom of loop, so the number of statements are repeated at least one time even the condition is false for the first time.

Syntax:

```
Initial value of loop; while  
(test condition)  
{  
    Statement 1;  
    Statement 2;  
    .....  
    .....  
    .....  
    Statement N;  
    Loop Update;  
}
```

Syntax:

```
Initial value of loop; do  
{  
    Statement 1;  
    Statement 2;  
    .....  
    .....  
    .....  
    Statement N;  
    Loop Update;  
}while (test condition);
```

Example:

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

Output:

1 2 3 4 5

Example:

```
#include<stdio.h>  
#include<conio.h> void  
main()  
{ int  
i=1;  
clrscr();  
do  
{  
    printf("%5d",i);  
i=i+1; }  
while(i<=5);  
getch(); }
```

Output:

1 2 3 4 5

Flowchart:

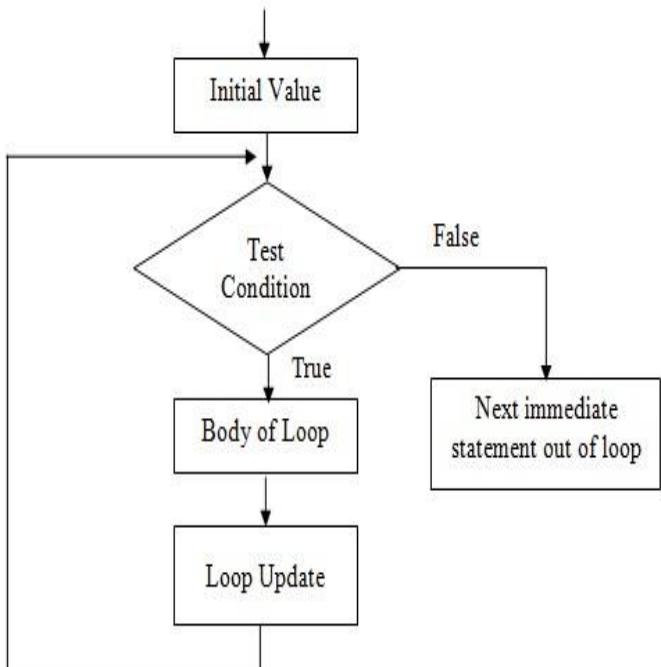


Fig: flowchart- while loop

Flowchart

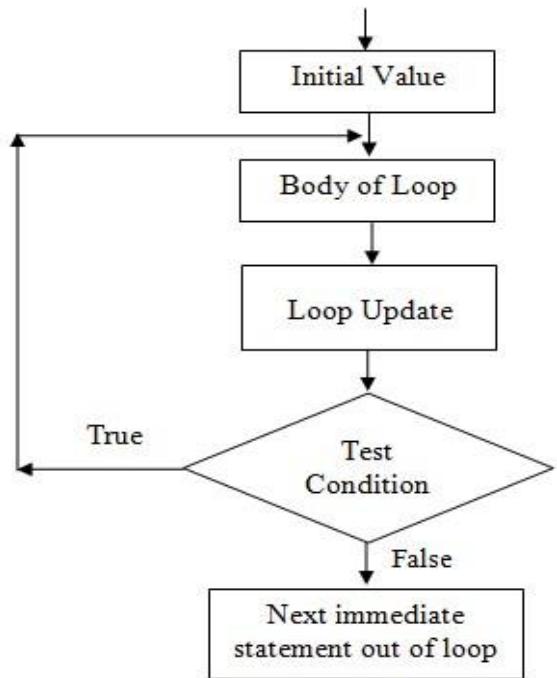


Fig: Flowchart- do... while loop

Loop Interruption:

Loop interruption is a process of skipping a part of the loop or quite the loop abnormally while executing. There are two keywords: *break* and *continue* to interrupt the flow of a loop. Generally both keywords are associated with *if* within a loop.

The Break Statement:

The **break** statement allows us to exit a loop from any point within its body i.e. abnormal termination of the loop. When the **break** statement is encountered inside a loop, the loop is immediately terminated, and program control is transferred to the next statement after the loop. The **break** statement can be used with all three of C's loop as well as in conjunction with functions and **switch-case** statements.

It is demonstrated in following program. **Example:**

```

#include<stdio.h>
#include<conio.h> void
main()
{ int i;
clrscr();
for(i=1;i<=10;i++)
{
  if(i%3==0) break;
  else printf("%5d",i);
}
getch(); }
  
```

Output:

1 2

The continue Statement:

The **continue** statement is somewhat the opposite of the **break** statement. When the **continue** keyword is encountered within a loop, it forces the next iteration of the loop to take place i.e. the control remains inside the loop until the loop condition satisfies and the loop will be normally terminated. The use of **continue** is demonstrated in following program.

```
#include<stdio.h>
#include<conio.h> void
main()
{ int i;
clrscr();
for(i=1;i<=10;i++)
{
    if(i%3==0) continue;
    else printf("%5d",i);
}
getch(); }
```

Output:

```
1 2 4 5 7 8 10
```

C goto Statement

The goto statement allows us to transfer control of the program to the specified label.

Syntax of goto Statement

```
goto label;
```

```
.... . . .
```

```
.... . . .
```

```
label:
```

```
statement;
```

The label is an identifier. When the goto statement is encountered, the control of the program jumps to label: and starts executing the code.

Example: goto Statement

```
// Program to calculate the sum and average of positive numbers
// If the user enters a negative number, the sum and average are displayed.
```

```
#include <stdio.h>

int main() {

    const int maxInput = 100;
    int i;
    double number, average, sum = 0.0;

    for (i = 1; i <= maxInput; ++i) {
        printf("%d. Enter a number: ", i);
        scanf("%lf", &number);

        // go to jump if the user enters a negative number
        if (number < 0.0) {
```

```

        goto jump;
    }
    sum += number;
}

jump:
    average = sum / (i - 1);
    printf("Sum = %.2f\n", sum);
    printf("Average = %.2f", average);

    return 0;
}
Output
1. Enter a number: 3
2. Enter a number: 4.3
3. Enter a number: 9.3
4. Enter a number: -2.9
Sum = 16.60
Average = 5.53

```

C Programs related to control structure

- 1. Write a program to ask number of quantities and price for per quantity then find total price.**

```

#include<stdio.h>
#include<conio.h> void
main()
{
int qty, price, amount; clrscr();
printf("\n Input Number of quantity:- ");
scanf("%d", &qty);
printf("\n Input Price per quantity:- ");
scanf("%d", &price); amount=qty*price;
printf("\n Total Price = Rs %d", amount);
getch();
}

```

- 2. Write a program to read basic salary then find tax and allowance, tax is 15% of basic salary and allowance 25% of basic salary. Also find out net salary.**

```

#include<stdio.h>
void main()
{
int bs;
float tax, allowance, net_salary;
printf("\n Input Basic Salary:- ");
scanf("%d", &bs); tax = bs*0.15;
allowance = bs*0.25;
net_salary = bs + allowance - tax;
printf("\n Tax = Rs %0.2f", tax);
printf("\n Allowance = Rs %0.2f", allowance);
printf("\n Net Salary = Rs %0.2f", net_salary);
}

```

- 3. Write a program to find area of circle.**

```
#include<stdio.h>
void main()
{
float radius, area;
printf("\n Input Radius of a Circle (in cm):- ");
scanf("%f", &radius); area = 2*3.14*radius;
printf("\n Area = %0.2f cm2", area);
}
```

4. Write a program to find square, cube and square root of given number.

```
#include<stdio.h>
#include<math.h>
void main()
{
int num, org, square, root, cube;
printf("\n Input any number:- ");
scanf("%d", &num); org = num;
square = pow(num,2); root =
sqrt(num); cube = pow(num,3);
printf("\n Square of %d is %d", num, square);
printf("\n Square Root of %d is %d", num, root);
printf("\n Cube of %d is %d", num, cube);
}
```

5. Write a program to read two numbers and swap them and display them.

```
#include<stdio.h>
void main()
{
int a, b, temp;
printf("\n Input First Number:- ");
scanf("%d", &a);
printf("\n Input Second Number:- ");
scanf("%d", &b);
printf("\n Numbers before swapping\n");
printf("\n First Number = %d and Second Number = %d", a, b);
temp = a; a = b; b = temp;
printf("\n Numbers after swapping\n");
printf("\n First Number = %d and Second Number = %d", a, b);
}
```

6. Write a program to read a four digit number and display the sum of those individual digits.

[Example: $1234 = 1 + 2 + 3 + 4 = 10$]

```
#include<stdio.h>
void main() {
int num, first, second, third, fourth, sum;
printf("\n Input any four digit number:- ");
scanf("%d", &num);
fourth = num%10;
num = num/10;
third = num%10;
num = num/10;
second = num%10;
num = num/10;
first = num%10;
```

```
sum = first + second + third + fourth;
printf("\n Sum = %d", sum); }
```

7. Write a program to read a four digit number and display its reverse. [E.g.: 1234 = 4321]

```
#include<stdio.h>
void main()
{
int num, first, second, third, fourth, org, reverse;
printf("\n Input any four digit number:- ");
scanf("%d", &num);
org = num;
fourth = num%10;
num = num/10;
third = num%10;
num = num/10;
second = num%10;
num = num/10;
first = num%10;
reverse = fourth*1000 + third*100 + second*10 + first*1;
printf("\n Reverse of %d is %d", org, reverse);
}
```

8. Write a program to read a four digit number then print a new number by adding one to each of its digits.

[Example: 1234 = 2345]

```
#include<stdio.h>
void main()
{
int num, org, first, second, third, fourth, new_num;
printf("\n Input any four digit number:- ");
scanf("%d", &num);
org = num;
fourth = num%10+1;
num = num/10;
third = num%10+1;
num = num/10+1;
second = num%10;
num = num/10+1;
first = num%10;
new_num = first*1000 + second*100 + third*10 + fourth*1;
printf("\n Required number is %d", new_num);
}
```

Decision Control Structure Programs:

1. Write a program to read a number and find whether it is odd or even number.

```
#include<stdio.h>
void main()
{
int num;
printf("\n Input any Number:- ");
scanf("%d", &num);
if(num%2==0)
    printf("\n %d is even number", num); else
    printf("\n %d is odd number", num);
}
```

2. Write a program to input a number and check the number is positive or negative.

```
#include<stdio.h>
void main() { int
num;
printf("\n Input any Number:- ");
scanf("%d", &num); if(num<0)
printf("\n %d is negative number", num);
else
printf("\n %d is positive number", num);
}
```

3. Write a program to input any two numbers and display the largest number between them.

```
#include<stdio.h>
void main()
{
int first, second;
printf("\n Input First Number:- ");
scanf("%d", &first);
printf("\n Input Second Number:- ");
scanf("%d", &second); if(first>second)
printf("\n %d is large number", first);
else
printf("\n %d is large number", second);
}
```

4. Write a program to input any three numbers and display the largest number among them.

```
#include<stdio.h>
void main() {
int first, second, third;
printf("\n Input First Number:- ");
scanf("%d", &first);
printf("\n Input Second Number:- ");
scanf("%d", &second);
printf("\n Input Third Number:- ");
scanf("%d", &third);
if(first>second)
{
    if(first>third)
        printf("\n %d is largest number", first);
    else
        printf("\n %d is largest number", third);
}
else
{
    if(second>third)
        printf("\n %d is largest number", second);
    else
        printf("\n %d is largest number", third);
}
}
```

Alternate method

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

```

void main()
{
int first, second, third;
printf("\n Input First Number:- ");
scanf("%d", &first);
printf("\n Input Second Number:- ");
scanf("%d", &second);
printf("\n Input Third Number:- ");
scanf("%d", &third);
if(first>second && first>third)
    printf("\n %d is largest number", first);
else if(second>first && second>third)
    printf("\n %d is largest number", second);
else
    printf("\n %d is largest number", third);
}

```

5. Write a program to find roots of a quadratic equation.

```

#include<stdio.h>
#include<math.h>
void main()
{
float a,b,c, root1, root2;
printf("\n Input value of a, b, and c :- ");
scanf("%f%f%f", &a, &b, &c);
if((b*b)>(4*a*c))
{
    root1 = -b + sqrt((b*b)-(4*a*c)/(2*a));
    root2 = -b - sqrt((b*b)-(4*a*c)/(2*a));
    printf("\n Roots are:- \n");
    printf("First Root = %.2f and Second Root = %8.2f", root1, root2);
} else
    printf("\n The roots are imaginary");
}

```

6. Write a program to read cost price and selling price of product and determine whether there is loss or profit and also determine how much profit made or loss incurred.

```

#include<stdio.h>
void main()
{
int cp, sp, profit, loss;
printf("\n Input Cost Price:- ");
scanf("%d", &cp);
printf("\n Input Selling Price:- ");
scanf("%d", &sp); if(sp==cp)
    printf("\n No profit No loss"); else
{
    if(sp>cp)
    {
        profit = sp - cp;
        printf("\n Profit = Rs %d", profit);
    }
else
{
    loss = cp - sp;
}
}

```

```

        printf("\n Loss = Rs %d", loss);
    }
}
}

```

7. Write a program that check whether the number entered by user is exactly divisible by 5 but not by 11.

```

#include<stdio.h>
void main()
{
int num;
printf("\n Input any number:- ");
scanf("%d", &num);
if(num%5 == 0 && num %11!=0)
printf("\n %d is a required number", num);
else
printf("\n %d is not required number", num);
}

```

8. Write a program to input any year through keyboard and determine whether it is leap year or not.

```

#include<stdio.h>
void main()
{
int year;
printf("\n Input any year:- ");
scanf("%d", &year);
if(year%4000 == 0 )
printf("\n %d is a leap year", year);
else if (year %100 == 0)
printf("\n %d is a leap year", year);
else if(year%4 ==0)
printf("\n %d is a leap year", year);
else
printf("\n %d is not a leap year", year);
}

```

9. Write a program to read a binary number and display its equivalent decimal number.

```

#include<stdio.h>
void main()
{
long int num, bin_num; int
dec = 0, base = 1, rem;
printf("\n Input a binary number :- ");
scanf("%ld", &num);
bin_num = num;
while(num>0)
{
rem = num%10;
dec = dec + rem * base;
num = num /10;
base = base *2;
}
printf("\n Binary number = %ld and Decimal number = %d", bin_num, dec);
}

```

10. Write a program to read a decimal number and display its equivalent binary number.

```
#include<stdio.h>
void main()
{
long int num,rem, dec_num, base = 1, binary = 0;
printf("\n Input a decimal number :- ");
scanf("%ld", &num);
dec_num = num;
while(num>0)
{
    rem = num%2;
binary = binary + rem * base;
num = num/2;
    base = base * 10;
}
printf("\n Decimal number = %ld and Binary number = %ld", dec_num, binary);
}
```

11. Write a program to read a octal number and display its equivalent decimal number.

```
/*Octal number to decimal number*/
#include<stdio.h>
void main()
{
long int num, oct_num;
int dec = 0, base = 1, rem;
printf("\n Input a octal number :- ");
scanf("%ld", &num); oct_num = num;
while(num>0)
{
    rem = num%10;
dec = dec + rem * base;
num = num /10;
    base = base * 8;
}
printf("\n Octal number = %ld and Decimal number = %d", oct_num, dec);
}
```

12. Write a program to calculate the salary as per following table

Gender	Year of Service	Qualification	Salary
Male	>=10	Post Graduate	15000
	>=10	Graduate	10000
	<10	Post Graduate	10000
	<10	Graduate	7000
Female	>=10	Post Graduate	12000
	>=10	Graduate	9000
	<10	Post Graduate	10000
	<10	Graduate	6000

```
#include<stdio.h>
void main()
{
int year_service, quali;
char sex;
printf("\n Input Sex (\\"M\\" for Male or \\"F\\" for Female):- ");
```

```

scanf("%c", &sex);
printf("\n Input year of service:- ");
scanf("%d", &year_service);
printf("\n Input qualification (0 for Post Graduate or 1 for Graduate):- ");
scanf("%d", &quali);
if(sex=='m' || sex == 'M')
{
    if(year_service>=10)
    {
        if(quali== 0)
            printf("\n Salary = Rs. 15000");
        else
            printf("\n Salary = Rs. 10000");
    } else
    {
        if(quali== 0)
            printf("\n Salary = Rs. 10000");
        else
            printf("\n Salary = Rs. 7000");
    }
}
else
{
    if(year_service>=10)
    {
        if(quali== 0)
            printf("\n Salary = Rs. 12000");
        else
            printf("\n Salary = Rs. 9000");
    }
else
{
    if(quali== 0)
        printf("\n Salary = Rs. 10000");
    else
        printf("\n Salary = Rs. 6000");
}
}
}

```

Alternate method

```

#include<stdio.h>
void main()
{
int year_service, quali, salary;
char sex;
printf("\n Input Sex (\\"M\\" for Male or \\"F\\" for Female):- ");
scanf("%c", &sex);
printf("\n Input year of service:- ");
scanf("%d", &year_service);
printf("\n Input qualification (0 for Post Graduate or 1 for Graduate):- ");
scanf("%d", &quali);

```

```

if(sex=='m' && year_service>=10 && quali==0)
    salary = 15000;
else if (sex=='m' && year_service>=10 && quali==1)
    salary = 10000;
else if ((sex=='m' && year_service<10 && quali==0) || (sex=='f' &&
year_service<10 && quali==0))
    salary = 10000;
else if (sex=='m' && year_service<10 && quali==1)
    salary = 7000;
else if (sex=='f' && year_service>=10 && quali==0)
    salary = 12000;
else if (sex=='f' && year_service>=10 && quali==1)
    salary = 9000; else salary = 6000;
printf("\n Salary = Rs. %d", salary);
}

```

- 13. Any character is entered through the keyboard; Write a program to determine whether the character entered is a capital letter, a small letter, a digit or a special symbol.**

```

#include<stdio.h>
void main()
{
char ch;
printf("\n Input any character:- ");
scanf("%c", &ch); if(ch>='A' &&
ch<='Z')
    printf("\n %c whose value is %d is a capital letter", ch, ch);
else if(ch>='a' && ch <='z')
    printf("\n %c whose value is %d is a small letter", ch, ch);
else if(ch>='0' && ch<='9')
    printf("\n %c whose value is %d is a number", ch, ch);
else
    printf("\n %c whose value is %d is a special letter", ch, ch);
}

```

- 14. Write a program to display the name of day on the basis of entered number from 1 to 7. For example, Sunday for 1.**

```

#include<stdio.h>
void main()
{
int day;
printf("\n Input any number:- ");
scanf("%d", &day);
switch(day)
{
case 1:
printf("\n Sunday");
break;
case 2:
printf("\n Monday");
break;
case 3:
printf("\n Tuesday");
break;
case 4:

```

```

        printf("\n Wednesday");
    break;
case 5:
    printf("\n Thursday");
    break;
case 6:
    printf("\n Friday");
    break;
case 7:
    printf("\n Saturday");
break;
default:
printf("\n Invalid Choice");

}
}

```

15. Write a menu driven program using switch statement having following options.

1. Addition
2. Subtraction
3. Multiplication
4. Division
5. Exit

```

/*Illustration of switch case*/
#include<stdio.h>
void main() {
int a, b, choice, sum, sub, rem, quot, mult;
printf("\n Menu for Arithmetic operations:\n");
printf("\n1. Addition");
printf("\n2. Subtraction");
printf("\n3. Multiplication"); printf("\n4.
Division");
printf("\n5. Exit");
printf("\n Input your choice:- ");
scanf("%d", &choice);

switch(choice)
{
case 1:
printf("\n Input First Number:- ");
scanf("%d", &a);
    printf("\n Input Second Number:- ");
    scanf("%d", &b);
sum=a+b;
    printf("\n Sum = %d", sum);
    break;
case 2:
printf("\n Input First Number:- ");
scanf("%d", &a);
    printf("\n Input Second Number:- ");
    scanf("%d", &b);
sub=a-b;
}
}

```

```

printf("\n Subtraction = %d", sub);
break;
case 3:
printf("\n Input First Number:- ");
scanf("%d", &a);
printf("\n Input Second Number:- ");
scanf("%d", &b);
mult=a*b;
printf("\n Multiplication = %d", mult);
break;
case 4:
printf("\n Input First Number:- ");
scanf("%d", &a);
printf("\n Input Second Number:- ");
scanf("%d", &b); quot=a/b;
rem=a%b;
printf("\n Quotient = %d", quot);
printf("\n Remainder = %d", rem);
break;
case 5:
exit();
}

default:
printf("\n Invalid choice");
}
}

```

The Loop Control Structure Programs:

1. Write a program to display series and sum of the series of followings:

a) 1, 2, 3,, 100

```

/*Natural Number*/
#include<stdio.h>
void main() {
int i, sum = 0;
for(i=1;i<=100;i++)
{
printf("%5d",i);
sum = sum + i;
}
printf("\n Sum = %d", sum);
}

```

b) 2, 4, 6,100

```

/*Sum of first 100 even numbers*/
#include<stdio.h>
void main() {
int i, sum = 0;
for(i=1;i<=100;i++)
{
if(i%2 == 0)
{
printf("%5d",i);
}
}

```

```

        sum = sum + i;
    }
}
printf("\n Sum = %d", sum);
}

```

c) 1, 8, 27, 64, up to nth term.

```

#include<stdio.h>
#include<math.h>
void main() {
    int i, n; long int num, sum =
0;
printf("\n How many terms:- ");
scanf("%d", &n);
for(i=1;i<=n;i++)
{
    num = pow(i,3);
    printf("%5d",num);
    sum = sum + num;
}
printf("\n Sum = %ld", sum);
}

```

d) 1, 11, 111, 1111, up to 10 terms

```

#include<stdio.h>
void main() {
int i,n;
long int sum = 0,term = 0;
printf("\n How many terms:- ");
scanf("%d", &n);
for(i=0;i<n;i++)
{
    term=term*10 + 1;
    sum = sum + term;
    printf("%10ld", term);
}
printf("\n Sum = %ld", sum);
}

```

e) 3, 9, 27, Up to nth terms.

```

#include<stdio.h>
void main() {
int i,n;
long int sum = 0,term = 1;
printf("\n How many terms:- ");
scanf("%d", &n);
for(i=0;i<n;i++)
{
    term=term*3;
    sum = sum+term;
    printf("%10ld", term);
}

```

```

    printf("\n Sum = %ld", sum);
}

f) x + x2/2 + x3/3 + x5/5..... up to n terms
#include<stdio.h>
#include<math.h>
void main() {
    int i,n,x, deno, exp = 1, a = 1, b = 1;
    long int nume;
    long float term = 0, sum = 0;
    printf("\n Input value of x:- ");
    scanf("%d", &x);
    printf("\n How many terms:- ");
    scanf("%d", &n);
    for(i=0;i<n;i++)
    {
        nume = pow(x,exp);
        deno = exp;
        term = nume/deno;
        printf("%ld/%d\t", nume, deno);
        exp = a + b;
        a = b;
        b = exp;
        sum = sum + term;
    }
    printf("\n Sum = %.2lf", sum);
}

```

2. Write a program to display multiplication table of a number which is entered through keyboard.

```

#include<stdio.h>
void main()
{
int i,n;
printf("\n Input any number:- ");
scanf("%d", &n); for(i=1;i<=10;i++)
printf("\n %2d x %2d = %3d", n, i, n*i);
}

```

3. Write a program to read a positive number less than 20 and display the multiplication table.

```

#include<stdio.h>
void main()
{
int i,n;
printf("\n Input any number:- ");
scanf("%d", &n); if(n<20)
{
for(i=1;i<=10;i++)
    printf("\n %2d x %2d = %3d", n, i, n*i);
} else
printf("\n Invalid Number");
}

```

4. Write a program to display all odd numbers from 1 to 500.

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

```

void main()
{
int i;
for(i=1; i<500;i++)
{
if(i%2!=0)
    printf("%5d", i);
}
}

```

5. Write a program to find the sum of numbers from 1 to 1000, which are exactly divisible by 4 but not by 6.

```

#include<stdio.h>
void main()
{
int i;
long int sum=0;
for(i=1;i<=100;i++)
{
if(i%4==0 && i%6!=0)
    sum = sum + (i*i);
}
printf("\n Sum = %ld", sum);
}

```

6. Write a program to find factorial of a given number.

```

#include<stdio.h>
void main() {
    int i,n; long
    int fact=1;
    printf("\n Input any number:- ");
    scanf("%d", &n);
    if(n<0)
        printf("\n Invalid Number");
    else if(n==0 || n==1)
        printf("\n Factorial of %d is 1",n);
    else
    {
        for(i=1; i<=n; i++)
            fact = fact*i;
        printf("\n Factorial of %2d = %ld", n, fact);
    }
}

```

7. Write a program to read a number and determine whether it is a prime number or not.

```

#include<stdio.h>
void main() {
int i, num;
printf("\n Input any number:- ");
scanf("%d", &num);
for(i=2; i<num; i++)
{
if(num%i==0)
{
    printf("%d is non-prime number", num);
}
}

```

```

        break;
    }
}
if(num==i)
    printf("\n %d is prime number", num);
}

```

8. Write a program to display all prime numbers from 1 to 1000.

```

#include<stdio.h>
void main() {
int i, j;
for(i=1;i<1000;i++)
{
for(j=2;j<i; j++)
{
if(i%j==0)
    break;
}
if(i == j)
    printf("%5d",i);
}
}

```

9. Write a program to read any number and determine whether it is Armstrong number or not.

```

#include<stdio.h>
#include<math.h>
void main() { int
rem;
long int num, org, arms=0;
printf("\n Input any number:- ");
scanf("%ld", &num); org=num; do
{
rem = num%10;
arms = arms + pow(rem,3);
num = num/10; }
while(num!=0);
if(org==arms)
    printf("\n %ld is Armstrong number", org);
else
    printf("\n %ld is non-Armstrong number", org);
}

```

10. Write a program to display Armstrong numbers up to nth term

```

#include<stdio.h>
#include<math.h>
void main() {
int i, temp, n, rem, sum = 0;
printf("\n Armstrong number up to:- ");
scanf("%d", &n);
for(i=1;i<=n;i++)
{
sum = 0;
temp = i;
while(temp!=0)

```

```

    {
    rem = temp%10;
    sum = sum + pow(rem,3);
        temp = temp/10;
    }
    if(i == sum)
        printf("%5d", sum);
}
}

```

11. Write a program to find sum of odd and even numbers from 1 to 100 numbers.

```

#include<stdio.h>
void main()
{
int i, even_sum = 0, odd_sum = 0;
for(i=1;i<=100;i++)
{
if(i%2==0)
    even_sum +=i;
else
odd_sum +=i;
}
printf("\n Sum of odd numbers is:- %d", odd_sum);
printf("\n Sum of even numbers is:- %d", even_sum);
}

```

12. Write a program to read a number and display its reverse number.

```

#include<stdio.h>
void main()
{
    int rem;
    long int num, org, reverse=0;
    printf("\n Input any number:- ");
    scanf("%ld", &num);
    org = num;
    do
    {
        rem = num%10;
        reverse = reverse*10+rem;
        num = num/10;
    }
    while(num!=0);
    printf("\n Reverse of %ld is %ld", org, reverse);
}

```

13. Write a program to input any number and determine whether it is palindrome number or not.

```

#include<stdio.h>
void main()
{ int rem;
long int num, org, reverse=0;
printf("\n Input any number:- ");
scanf("%ld", &num); org=num; do
{

```

```

rem=num%10;
reverse=reverse*10+rem;
num=num/10;
}while(num!=0);
if(org==reverse)
printf("\n %ld is palindrome number", org);
else
printf("\n %ld is non-palindrome number", org);
}

```

14. Write a program to display Fibonacci series.

```

#include<stdio.h>
void main() {
int i, n, first=0,second=1,third;
printf("\n How many terms:- ");
scanf("%d", &n);
printf("%5d%5d",first,second);
for(i=0;i<n-2;i++)
{
    third = first + second;
    printf("%5d",third);
    first = second;
    second = third;
}
}

```

a) 5 4 3 2 1

```

5 4 3 2
5 4 3
5 4
5

```

```

#include<stdio.h>
void main() {
int i, j;
clrscr();
for(i=1;i<=5;i++)
{
    for(j=5;j>=i;j--)
        printf("%2d",j);  printf("\n");
}
}

```

b) 1 2 3 4 5

```

1 2 3 4
1 2 3
1 2
1

```

```

#include<stdio.h>
void main() {
int i, j;

```

```

for(i=5;i>=1;i--
)
{
    for(j=1;j<=i;j++)
printf("%2d",j);
    printf("\n");
}
}

```

c) C O M P U T E R

C O M P U T E

C O M P U T

C O M P U

C O M P

C O M

C O

C

```

#include<stdio.h>
void main() {
int i, j,k;
char text[] = "COMPUTER";
for(i=0;i<8;i++)
{
k=0;   for(j=8;j>i;j--)
{
printf("%2c",text[k]);
k++;
}
printf("\n");
}
}

```

d) C

C O

C O M

C O M P

C O M P U

C O M P U T

C O M P U T E

C O M P U T E R

```

#include<stdio.h>
void main() {
int i, j;
char text[] = "COMPUTER";
for(i=0;i<8;i++)
{
for(j=0;j<=i;j++)
printf("%2c", text[j]);
printf("\n");
}
}

```

e) **5**

```

      4 5
      3 4 5
      2 3 4 5
1 2 3 4 5
#include<stdio.h>
void main() {
int i,j,k,n;
for(i=5;i>=1;i--)
{
    for(k=1;k<=i-1;k++)
printf(" ");
    for(j=i;j<=5;j++)
printf("%d",j);
printf("\n");
}
}
```

f) **H**

```

HH
EEE
LLLL
OOOOO
WWWWWW
#include<stdio.h>
#include<string.h>
void main() {
int i, j, length; char
text[50];
printf("\n Input any text:- ");
gets(text);
length = strlen(text);
for(i=0;i<length;i++)
{
    for(j=0;j<=i;j++)
printf("%c", text[i]);
printf("\n");
}
printf("\n"); for(i=0;i<length;i++)
{
    for(j=0;j<=i;j++)
printf("%c", text[j]);
printf("\n");
}
printf("\n"); for(i=length-1;i>=0;i--)
{
    for(j=0;j<=i;j++)
printf("%c", text[j]);
printf("\n");
}
}
```