Navlakhi®

# Learning C Programming


By Abhishek Navlakhi

Tel: 9820246760 / 9769479368

Enquiry: 9820009650/51/52/53 (10am to 6pm)

# Certificate of Registration

This is to Certify that
Quality Management System of

## NAVLAKHI

708, A. C. MARKET BULDING,TARDEO ROAD, MUMBAI- 400034,
MAHARASHTRA, INDIA.

has been assessed and found to conform to the requirements of

**ISO 9001:2015**

for the following scope :

COACHING FOR MATHEMATICS, SCIENCE & TECHNOLOGY FOR SCHOOLS AND COLLEGES.

| | | | |
|---|---|---|---|
| Certificate No | : **22EQIY73** | | |
| Initial Registration Date | : 03/11/2022 | Issuance Date | : 03/11/2022 |
| Date of Expiry | : 02/11/2025 | | |
| 1st Surve. Due | : 03/10/2023 | 2nd Surve. Due | : 03/10/2024 |

EGAC Accredited
QMS Certification
CAB # 118005

Director

**Magnitude Management Services Pvt. Ltd.**
B-55, Lower Ground Floor, Sector 02, Noida-201301, U.P, India
e-mail: info@mmscertification.com, website: www.mmscertification.com

* Subject to Successful Surveillance Audit and case surveillance audit is not allowed to be conducted, this certificate shall be suspended/withdrawal.

# Navlakhi®

An Intellectual Development

ISO 9001:2015 Certified Institution

**For Enquiry**
**9769479368 / 9820246760**

## (FOR MU colleges)

| Semester 1 | | Semester 2 | |
|---|---|---|---|
| MATHEMATICS – 1 | Rs.11000 | MATHEMATICS – 2 | Rs.11000 |
| BASIC ELECTRICAL ENGG | Rs.11000 | ENGG. GRAPHICS | Rs.13500 |
| ENGINEERING MECHANICS | Rs.13500 | C PROGRAMMING | Rs.11000 |
| TOTAL | Rs. 35500 | TOTAL | Rs. 35500 |

Combined semester discount 6 Subjects **Rs. 55000** only

## (FOR DJ Sanghvi)

| Semester 1 | | Semester 2 | |
|---|---|---|---|
| MATHEMATICS – 1 | Rs.11000 | MATHEMATICS – 2 | Rs.11000 |
| C PROGRAMMING | Rs.11000 | JAVA | Rs.11000 |
| ENGINEERING MECHANICS (Sem1 or 2) | Rs.13500 | ENGG. GRAPHICS (Sem2 or 1) | Rs.13500 |
| BEE & DIGITAL (Sem1 or 2) | Rs.11000 | | |
| TOTAL | Rs. 46500 | TOTAL | Rs. 35500 |

Combined semester discount 7 Subjects **Rs. 65000** only

## (FOR SPIT)

| Semester 1 | | Semester 2 | |
|---|---|---|---|
| CALCULUS | Rs.11000 | DIFF EQN+COMPLEX | Rs.11000 |
| C PROGRAMMING | Rs.11000 | OOP (JAVA/C++) | Rs.11000 |
| ENGINEERING MECHANICS | Rs.13500 | ENGG. GRAPHICS | Rs.13500 |
| DIGITAL+MICROPROCESSOR | Rs.11000 | BEE | Rs.11000 |
| TOTAL | Rs. 46500 | TOTAL | Rs. 46500 |

Combined semester discount 8 Subjects **Rs. 74000** only

* Students of SNDT/KJ Vidyavihar/Other autonomous colleges will have few different subjects. They can contact us personally for the same

*Decision of Confirmation and rejection of admission and cancellation lies solely with us and it shall be final and our decision cannot be objected in any case

***FEATURES:***

**Maximum classroom batch Size 10 students**

Over 300 questions solved in class for Mechanics & BEE
Over 100 programs solved in class for SPA (C Prog)
Over 250 questions solved in class for engineering drawing
Over 300 problems solved in class for Maths1 & Maths2
Detailed Understanding of Object-Oriented Core Concepts
Exhaustive practice done in class for all subjects
Personal attention with think – & – solve approach
In case of different subjects in college the student will be shifted to that subject batch
100% refund of fees in case student does not get admission into engineering

# KNOW YOUR INSTRUCTORS

| NAME: | KUNAL NAVLAKHI |
|---|---|
| QUALIFICATION: | ➢ B.E. Electronics,<br>➢ Entrepreneurship Management, Wellingkar,<br>➢ International Executive Management, UBS,Belgium |
| EXPERIENCE: | ➢ 2 yrs in Educational Technology Unit at NCST<br>➢ More than 20 yrs of coaching experience |
| SOME ACADEMIC ACHIEVEMENTS: | ➢ Distinction in MBA from Wellingkar<br>➢ First class each year in engineering<br>➢ Second Rank in Engineering Degree College<br>➢ 90/100 in maths in ICSE<br>➢ 97/100 in maths in HSC |

| NAME: | ABHISHEK NAVLAKHI |
|---|---|
| QUALIFICATION: | ➢ B.E. Computers |
| EXPERIENCE: | ➢ 2 years as programmer at CMC Ltd.<br>➢ More than 20 yrs of coaching experience |
| SOME ACADEMIC ACHIEVEMENTS: | ➢ First class each year in engineering<br>➢ 92 percentile in Data Structures at the all India NCST G level exam<br>➢ 100/100 in physics at hsc (1st in maharashtra)<br>➢ 89.5% aggregate In hsc<br>➢ 192/200 in electronics at hsc<br>➢ 96/100 in maths @ ICSE<br>➢ 94/100 in Science @ ICSE |

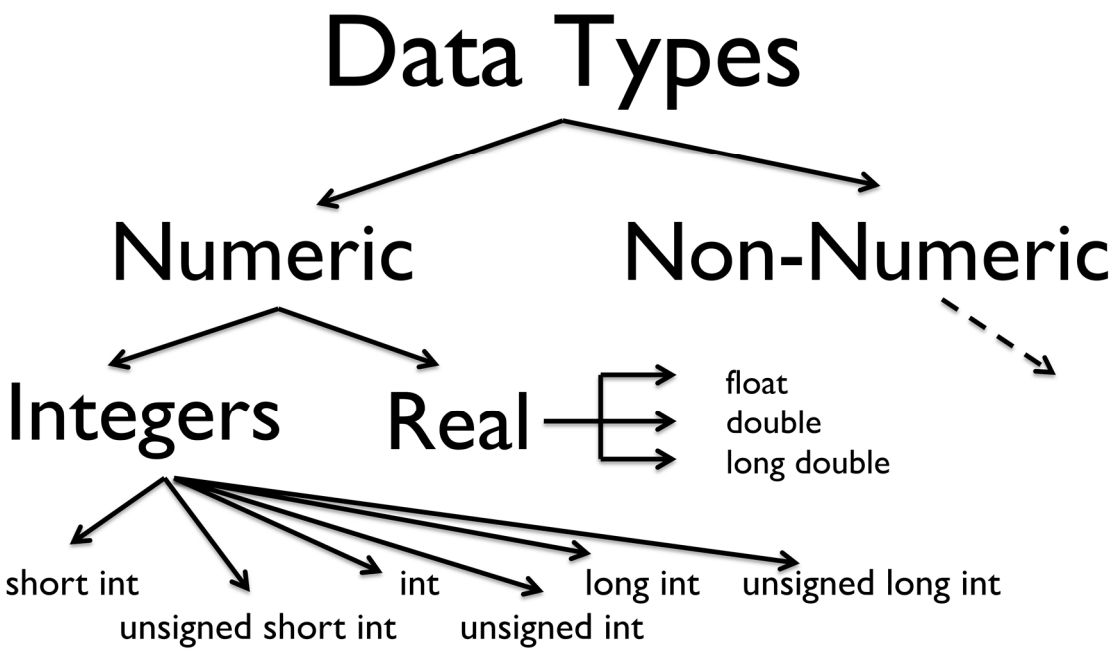
Navlakhi®

# Chapter 1

# Sequential Programming

## C Functions

| Purpose | C Function | Header file |
|---|---|---|
| • Display / Print on monitor/console | printf | stdio.h |
| • Read from keyboard | scanf | stdio.h |
| • Mathematical Operation | | |
| Sine of an angle x | sin(x) | math.h |
| Cosine of an angle x | cos(x) | math.h |
| Tangent of an angle x | tan(x) | math.h |
| $Sin^{-1}$ | asin(x) | math.h |
| $Cos^{-1}$ | acos(x) | math.h |
| $Tan^{-1}$ | atan(x) | math.h |
| Square root of x | sqrt(x) | math.h |
| $x^y$ | pow(x,y) | math.h |
| $\log_{10}x$ | log10(x) | math.h |
| $\log_e x$ | log(x) | math.h |
| Anti-log of $\log_e x$ OR $e^x$ | exp(x) | math.h |
| Round off to next higher integer | ceil(x) | math.h |
| Round off to next lower integer | floor(x) | math.h |




**Data Types**

Variables are used to store information. Variables can be of the following (basic or primitive) types.

char – just one letter

int – integer (-32768 to 32767)
unsigned int – non-negative integers (0 to 65535)

short int – integer +ve as well as –ve
unsigned short int – non-negative integers

long int – integer (-2,147,483,648 to 2,147,483,647)
unsigned long integer – non-negative integers ( 0 to 4,294,967,295)

float – real numbers (3.4E-38 to 3.4E+38)

double – real numbers (1.7E-308 to 1.7E+308)

long double – real numbers (3.4E-4932 to 1.1E+4932)

**e.g.**

| | | |
|---|---|---|
| int x; | unsigned int y; | unsigned z; |
| long int a; | long b; | float c; |
| double d; | long double e; | |

**Basic Structure of C Program**

#include <header file name>
#include <header file name>
void main( void )
{
……..
……..
}

**Displaying Output**

printf("control string",arg1, arg2,……, argn);

*e.g. printf("%d %f %c %s",a, b, c, d);*

In the control string the format specifications should match the variables in number, order & type.

| Format Code | Meaning |
|---|---|
| %c | Prints a single character |
| %d | Prints a decimal integer |
| %i | Prints a signed decimal integer |
| %u | Prints an unsigned decimal integer |
| %e | Prints a floating point value in exponent form |
| %f | Prints a floating point value without exponent |
| %g | Prints a floating type value in e-type or f-type |
| %o | Prints an octal integer without the leading zero |
| %x | Prints a hexadecimal integer without the loading 0X |
| %s | Prints a string |

Following prefix can be added to some of the format codes:

h : for small integers *l :* for long or double L : for long double

In addition to these we can specify output format flags:

| Flag | Meaning |
|---|---|
| - | Left – justified output within the field limit. |
| + | + or – will precede the signed number |
| 0 | Causes leading zeroes to appear |
| # (with 0 or X) | Octal numbers will be preceded by 0<br>Hexadecimal numbers will be preceded by 0x |
| # (with e, f or g) | Each floating point number is represented by a decimal point, even if it is a whole number.<br>Prevents truncation of trailing zeros in g-type conversion. |

Formatting Output:

- Formatting using %wd – discussed in class
- Formatting using %w.pf – discussed in class
- Formatting using %w.pe – discussed in class
- Formatting using %*.*f – discussed in class
- Formatting using %wc – discussed in class
- Formatting using %w.ps – discussed in class
- Formatting using \n and \t – discussed in class

**Reading Input**

To read data we need to

1. Have an already created variable (memory location) of the correct type.
2. Use scanf to accept data from keyboard & transfer it to the variable location.

Syntax of scanf is as follows:

scanf("control string",arg1 address, arg2 address,....., argn address);

*e.g. scanf("%d %f %s",&tel, &marks, name);*

| Format Code | Meaning |
|---|---|
| %c | Reads a single character |
| %d | Reads a decimal integer |
| %i | Reads a decimal, hexadecimal or octal integer |
| %u | Reads an unsigned decimal integer |
| %e | Reads a floating point value |
| %f | Reads a floating point value |
| %g | Reads a floating type value |
| %o | Reads an octal integer |
| %x | Reads an hexadecimal integer |
| %s | Reads a string |
| %[ ] | Reads a string of words |

Following prefix can be added to some of the format codes:

h : for small integers *l* : for long or double L : for long double

Return value:

scanf returns the number of successful inputs read.

Formatting Input:

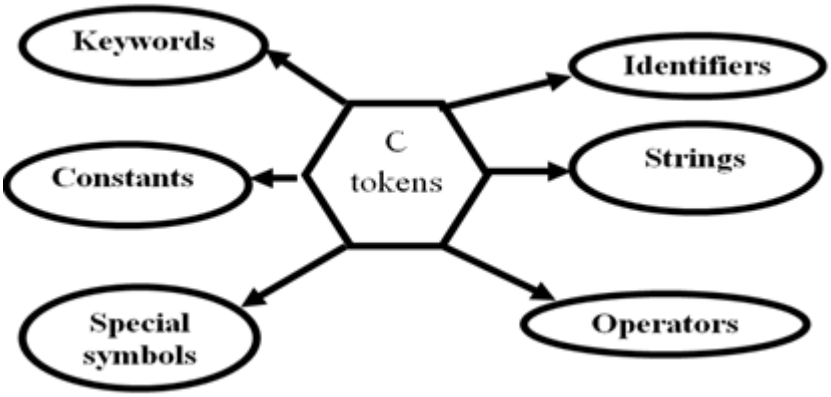
- Formatting using %wd – discussed in class
- Formatting using %* d – discussed in class
- Formatting using %ws – discussed in class
- Formatting using %wc – discussed in class

NOTE: Default symbol for differentiating input in a multiple input scanf statement is enter or space(s).

**Programming Example 2:**
Read the principle, Rate and Time from user and calculate the Simple Interest & Compound Interest.

```
#include <stdio.h>
#include <math.h>
void main()
{
  float P,t,R,CI,SI;
  printf("Enter Principal, Rate and Time");
  scanf("%f%f%f",&P,&R,&t);
  SI=P*t*R/100;
  CI=P*pow(1+R/100,t)-P;
  printf("Simple Int=%f\nCompd Int=%f",SI,CI);
}
```

**C tokens**




**Operators, Expressions, Type Conversion and Precedence**

➔Arithmetic Operators:

- Integer Arithmetic

14 – 4 = 10 14 + 4 = 18 14 * 4 = 56
14 / 4 = 3 14 % 4 = 2 –14 % 3 = -2
–14 % –3 = -2 14 % –3 = 2

- Real Arithmetic

1.0 / 3.0 = 0.333333
–2.0 / 3.0 = -0.666667
NOTE: % operator cannot be used with real numbers (i.e. with float, double)

- Mixed – mode Arithmetic

15 / 10.0 = 1.5

➔Relational Operators

4.5 < = 5 True –10 < –11 False
20 > = 20 True 10 + 2 ! = 12 False
30 – 1 = = 29 True –35 > 10 False

NOTE that a relational operation evaluates to a TRUE (1) or FALSE (0)

➔Logical Operators

(TRUE is any number other than 0. FALSE is 0)
AND &&
OR ||

e.g. a > b && d = = 10
The above statement is TRUE if both a>b and d=10
e.g. a > b || d = = 10
The above statement is TRUE if either a>b OR d=10 OR both
e.g. a > b || d = =10 && c < d
NOTE that AND has a higher precedence than OR
Thus the above example implies
a > b || (d = =10 && c < d)
(T/F) || ( T/F && T/F)

(Relational statements evaluate to a TRUE/FALSE)

e.g. a >b || 1
The above statement is always TRUE
e.g. a > b && 0
The above statement is always FALSE

➔Assignment Operators

- x = 5;
- x = x + y * 3;
  x = x + (y * 3);
  x + = y * 3;
- x = x + 1; x += 1;
- x = x – 3; x –= 3;
- x = x * 3; x *= 3;
- x = x / 3; x /= 3;
- x = x % y; x %= y;
- x = x * x; x *= x;

Advantages of Shorthand assignment operators:

- Whatever appears on the left need not be repeated & hence it becomes easier to write
- Statements become more concise & easier to read
- Statement becomes more efficient

➔Increment Decrement Operators

- x = x + 1; x + +; + + x;
- x = x – 1; x – –; – – x;
- x = a++ + 5; x = a + 5; a = a + 1;
- x = ++a + 5; a = a + 1; x = a + 5;

y = m + + is not same as y = + + m WHY?

➔Conditional Operator
More about it in the next chapter…..

Navlakhi®

→Objectives

- y = 100 Find x and y after execution of

(a) x = y – –y * 4
(b) x = y = y + +
(c) x = y = = 100
(d) x = y = = y + +

→Write the following in correct C++ notation

$$a + b / c - d$$

$$\frac{a + b}{c - d}$$

$$\frac{ab}{c - d}$$

$$\frac{ab}{c} - d$$

$$\frac{a}{cd} - b$$

$$a + \frac{1}{1 + \frac{1}{1 + a}}$$

→Determine the value of the following C expression:
–2 * –3 / 4 % 5 – –6 + 4

→Determine the value of i , j, and k after execution
int i, j, k;
i = j = k = 1;
i –= –j – – – – –k;

→Determine the value of x, y, and z after execution
int x, z;
float y;
x = 5;
x /= y = z = 1 + 1.5;
→Determine the output of the following code section
a=100;
printf(“%d%d%d”,a,+ + a,a + +);
printf(“%d”,a);

**Type Conversion**

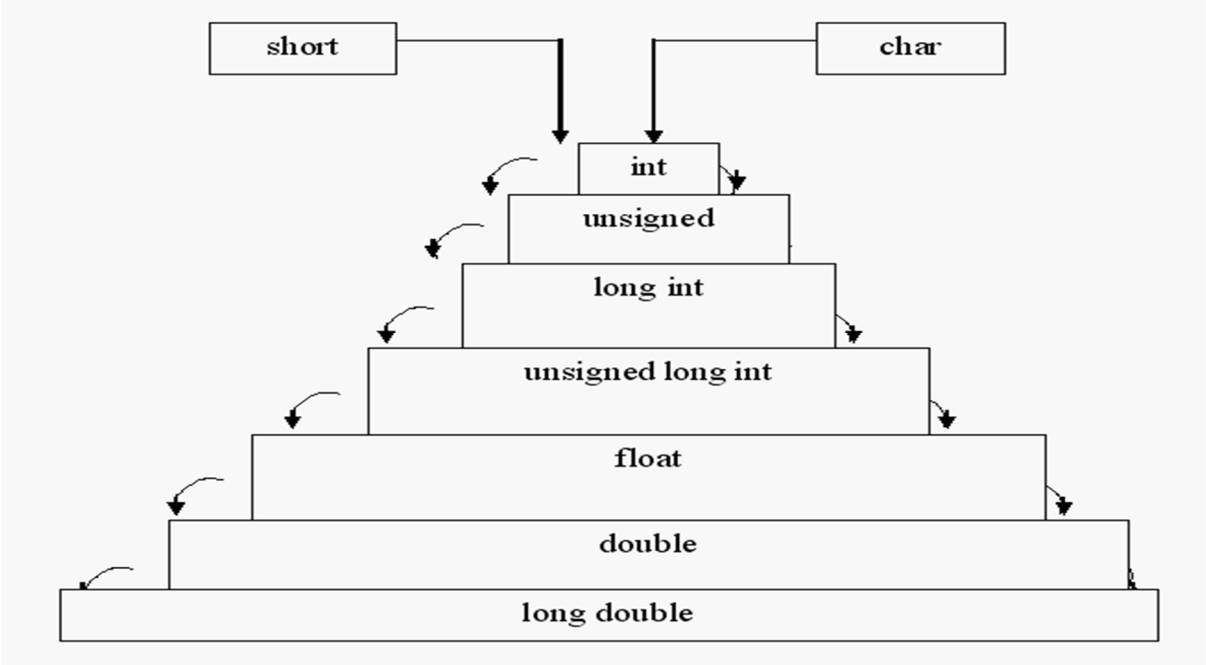
Type conversion are of two types

- Implicit
- Explicit

Implicit Type Conversion: Implicit means automatic. This takes place when we are operating on data of different types.

e.g. 15.0/10 = 1.500000

In the above example a double is divided by an integer and the answer is a double.
The integer data is automatically (implicitly) converted to a double & two double numbers divided will always give a double.




- Determine the type of result

(c/u – l) + s * f

where c = char u = unsigned int
l = long int s = short int
f = float

Explicit Type Conversion

Say we want to find 10/3, obviously the mathematical result is 3.333333333 but since both the numbers are integers (& hence on the same step of the previous diagram), hence the result would also be an integer i.e. 3 & not 3.333333

In order to get the correct answer we need to push atleast one number to either float, double or long double. This can be done using explicit type conversion or casting.

e.g.

(float)10/3 10/(float)3 (float)10/(float)3

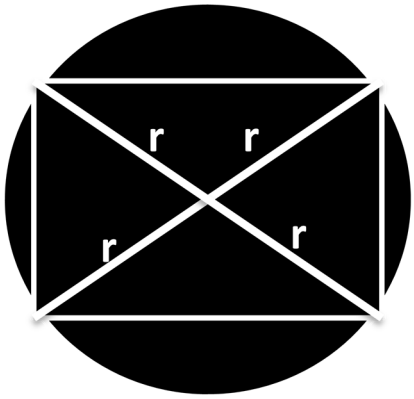
**Programming Example 3: PE3**
Write an interactive program that computes kmpl (km per litre) and cost per km for the operation of a vehicle based on the km traveled, gasoline consumed, cost of gasoline, and other operating costs

```
#include <stdio.h>
void main()
{
  float km,gas,cgas,ocost,kmpl,cpkm;
  printf("Enter km,gas consumed,cost of gas, other costs");
  scanf("%f%f%f%f",&km,&gas,&cgas,&ocost);
  kmpl=km/gas;
  cpkm=(gas*cgas+ocost)/km;
  printf("km/l=%f\ncost/km=%f",kmpl, cpkm);
}
```

Navlakhi®

**PE4:**
Write a program that reads the radius of a circle & determines its area, the area of the largest square contained within it, & the ratio of the two.

```
#include <stdio.h>
void main()
{
  float r,AC,ALSQ,Ratio;
  printf("Enter Radius of circle");
  scanf("%f",&r);
  AC=22.0/7*r*r;
  ALSQ= 2*r*r;
  Ratio=AC/ALSQ;
 printf("Area of Circle=%f\n Area of largest sq=%f\nRatio=%f",AC,ALSQ,Ratio);
}
```



**PE5:** Write a program that determines the difference in the value of an investment after a given number of years when (i) the investment earns simple interest and (ii) the interest is compounded annually at the same rate.

```
#include <stdio.h>
#include <math.h>
void main()
{
  float P,t,R,CI,SI,diff;
  printf("Enter Principal, Rate and Time");
  scanf("%f%f%f",&P,&R,&t);
  SI=P*t*R/100;
  CI=P*pow(1+R/100,t)-P;
  diff=CI – SI;

  printf("Simple Int=%.2f\nCompd Int=%.2f\ndifference=%.2f",SI,CI,diff);
}
```

**PE6:** A projectile fired at an angle θ with an initial velocity v travels a distance d given by

$d = \frac{v^2}{g} \sin 2\theta$

It stays in motion for a time $t = \frac{2v}{g} \sin\theta$

Maximum height $h = \frac{v^2}{g} \sin\theta$

Write a program that computes d, t and h, given v and θ.

```
#include <stdio.h>
#include <math.h>
void main()
{
  float d,t,h,v,theta;
  printf("Enter vel and angle ");
  scanf("%f%f",&v,&theta);
  theta=theta*3.14/180;
  d=v*v/9.81*sin(2*theta);
  t=2*v/9.81*sin(theta);
  h=v*v/9.81*sin(theta);
  printf("d=%f\nt=%f\nh=%f",d,t,h);
}
```

**PE7:** Write a program that interchanges the value of x and y so that x has y 's value y has x's value.

```
#include <stdio.h>
void main()
{
  float x,y,z;
  printf("Enter x and y ");
  scanf("%f%f",&x,&y);
  z=x;
  x=y;
  y=z;

  printf("x=%f\ny=%f",x,y);
}
```

**PE8:** Write a program to convert a given number of seconds into hours, minutes and seconds.

```
#include <stdio.h>
void main()
{
 int s,h,m;
 printf("Enter seconds ");
 scanf("%d",&s);
 h=s/3600;
 s=s%3600;
 m=s/60;
 s=s%60;
 printf("hours=%d\tmin=%d\tseconds=%d",h,m,s);
}
```

**PE9:** Suppose that a particle gets into motion from rest & has constant acceleration 'a' for t seconds. The distance traveled 'd' by the particle & the final velocity 'v' are given by
$d=\frac{1}{2}at^2$ and $v = at$
Write a program that reads 'a' and 't', and prints 'd' and 'v'.

```
#include <stdio.h>
void main()
{
  float a,t,d,v;
  printf("Enter a and t ");
  scanf("%f%f",&a,&t);
  d=1/2.0*a*t*t;
  v=a*t;
  printf("d=%f\nv=%f",d,v);
}
```

**PE10:** Write a program that reads three numbers, $x_1$, $x_2$, and $x_3$, and computes their average μ, their standard deviation σ, and the relative percentage RP for each numbers, where $\mu = \frac{x1 + x2 + x3}{3}$

$\sigma^2 = \frac{x_1^2+x_2^2+x_3^2-3\mu^2}{3}$

RP1 = $\frac{x_1}{x_1+x_2+x_3}$.100 similarly RP2 & RP3

```
#include <stdio.h>
#include <math.h>
void main()
{
 float x1,x2,x3,M,S,RP1,RP2,RP3;
 printf("Enter x1,x2,x3 ");
 scanf("%f%f%f",&x1,&x2,&x3);
 M=(x1+x2+x3)/3;
 S=sqrt((x1*x1+x2*x2+x3*x3-3*M*M)/3);
 RP1=x1/(x1+x2+x3)*100;
 RP2=x2/(x1+x2+x3)*100;
 RP3=x3/(x1+x2+x3)*100;

printf("avg=%f\n std deviation=%f\n rel percentage1=%f\n rel percentage2=\n rel
percentage3=%f",M, S, RP1,RP2,RP3);
}
```

**PE11:** The volume v of a sphere of radius r is given by $v = (4/3)\ \pi r^3$
Write a program that computes the thickness of a hollow ball, given the total volume occupied by the ball and the volume inside the ball.

```
#include <stdio.h>
#include <math.h>
void main()
{
  float V,v,R,r,t;
  printf("Enter outer vol & inner vol");
  scanf("%f%f",&V,&v);
  R=pow(3*V/(4*3.14),1.0/3);
  r=pow(3*v/(4*3.14),1.0/3);
  t=R-r;
  printf("Thickness=%f",t);
}
```

**PE12:** The distance d between points $(x_1,y_1)$ and $(x_2,y_2)$ is given by
$d=((x_1-x_2)^2+(y_1-y_2)^2)^{1/2}$
and the perimeter 'p' and the area 'a' of a triangle of sides of length u, v, and w are given by
$p=u+v+w$ and $a=(s(s-u)(s-v)(s-w))^{1/2}$
where s=p/2. Write a program that reads the coordinates of three points of a triangle & prints its perimeter & area.

```
#include <stdio.h>
#include <math.h>
void main()
{
  float x1,y1,x2,y2,x3,y3,p,s,a,u,v,w;
  printf("Enter x1,y1,x2,y2,x3,y3");
  scanf("%f%f%f%f%f%f",&x1,&y1,&x2,&y2,&x3,&y3);
  u=sqrt((x1-x2)*(x1-x2)+(y1-y2)*(y1-y2));
  v=sqrt((x1-x3)*(x1-x3)+(y1-y3)*(y1-y3));
  w=sqrt((x2-x3)*(x2-x3)+(y2-y3)*(y2-y3));
  p=u+v+w;
  s=p/2;
  a=sqrt(s*(s-u)*(s-v)*(s-w));
  printf("Area=%f\n perimeter=%f",a,p);
}
```

Navlakhi®


# Chapter 2

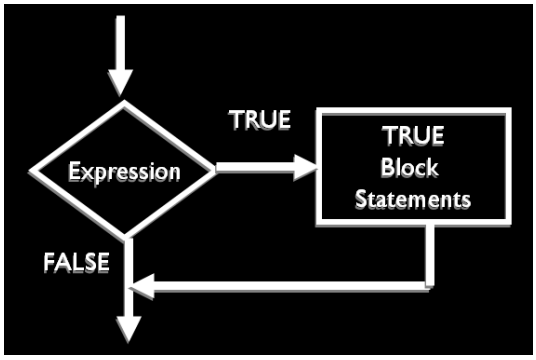
# Selective Programming

**if Statement**

```
if (expression)
{
        ……..
}
```

The expression can be a simple relational, or a more complex one (involving many relational expressions combined using one or more logical operators or just a numeric expression).

If the expression evaluates to a true (non – zero value) then the block of code is executed ELSE the block of code is skipped & the statements below it will yet be executed.



**PE14:** Write a program that finds the absolute value of a number.

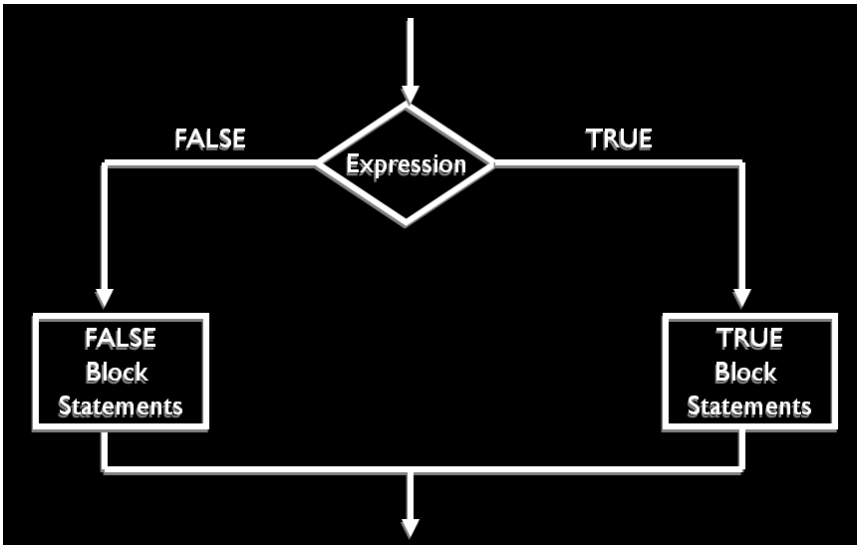
```
#include <stdio.h>
void main()
{
  float x;
  printf("Enter a number ");
  scanf("%f",&x);
  if(x<0)
  {
          x= –x;
  }
  printf("Absolute value=%f",x);
}
```

**Alternate Solution**

```
#include <stdio.h>
void main()
{
  float x,y;
  printf("Enter a number ");
  scanf("%f",&x);
  if (x<0)
  {
          y= –x;
   }
   else
    {
          y=x;
     }
  printf("Absolute value=%f",y);
}
```

**if – else Statement**

```
if (expression)
{
        ……../*TRUE block*/
}
else
{
        ……../*FALSE block*/
}
```




If the expression evaluates to a true (non – zero value) then the 'if' block of code is executed ELSE the 'if' block of code is skipped & 'else' block of code is executed. In either case the statements below it will yet be executed.

**Conditional Operator (Ternary)**

```
if (expression)
{
        x = …(i)….;
}                                   ➔x =(expression)?(i) : (ii);
else
{
        x = …(ii)…;
}
```

Navlakhi®

**Dangling Else Problem**

```
if (expression1)
        if (expression2) {…….}
else
{
        if (expression3) {……}
        else {……}
}
```

Each 'else' is associated with the closet 'if ' (without an else and in the same level/block). Hence the 1st 'else' is for expr2 & not for expr1.

**PE15:** Write a program that finds the larger of two numbers.

```
#include <stdio.h>
void main()
{
  float x,y;
  printf("Enter 2 numbers ");
  scanf("%f%f",&x,&y);
  if (x>y)
  {
        printf("Larger Number=%f",x);
   }
   else
   {
        printf("Larger Number=%f",y);
   }
}
```

**Alternate Solution**

```
#include <stdio.h>
void main()
{
  float x,y,z;
  printf("Enter 2 numbers ");
  scanf("%f%f",&x,&y);
  if (x>y)
  {
        z=x;
   }
   else
   {
        z=y;
   }
printf("Larger Number=%f",z);
}
```

**PE16:** Write a program to check whether a number is divisible by 3.

```
#include <stdio.h>
void main()
{
 int x;
 printf("Enter a number ");
 scanf("%f",&x);
 if(x%3==0)
 {
        printf("Number is divisible by 3");
  }
  else
  {
        printf("Number is not divisible by 3");
   }
}
```

**PE17:** Write a program to check whether a number is divisible by 3 or 7.

```
#include <stdio.h>
void main()
{
 int x;
 printf("Enter a number ");
 scanf("%d",&x);
 if(x%3==0 || x%7==0)
 {
        printf("Number is divisible by 3 or 7");
  }
  else
  {
        printf("Number is not divisible by 3 and 7");
   }
}
```

**PE18:** Write a program to check whether a number is divisible by 3 and 7.

```
#include <stdio.h>
void main()
{
 int x;
 printf("Enter a number ");
 scanf("%d",&x);
 if(x%3==0 && x%7==0)
 {
        printf("Number is divisible by 3 and 7");
  }
  else
  {
        printf("Number is not divisible by 3 or 7");
   }
}
```

**PE19:** Write a program to check whether a number is even or odd.

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

**PE20:** Write a program to check whether a year is a leap year or not.

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

Navlakhi®

**Multiway Conditional Statement**

**if (expression1)**

**statement-1**

**else if (expression2)**

**statement-2**

**……**

**else**

**statement-n**

Only one statement block will be executed.

**PE21:** Write a program to accept the 5 subject marks of a student & calculate the total & display the grade depending on percentage [60+ = A; 50 – 60=B; 40 – 50 =C; below 40=F]

```
#include <stdio.h>
void main()
{
 int m1,m2,m3,m4,m5,t;
 float p;
 printf("Enter 5 subject marks ");
 scanf("%d%d%d%d%d",&m1,&m2,&m3,&m4,&m5);
 t=m1+m2+m3+m4+m5;
 p=t/5.0;
 printf("Total=%d\n Percentage=%.2f",t,p);

  if(p>=60)
  {
        printf("Grade A");
  }
  if(p<60 && p>=50)
  {
        printf("Grade B");
  }
  if(p<50 && p>=40)
  {
        printf("Grade C");
  }
  if(p<40)
  {
        printf("Grade F");
  }
}
```

## Alternate Solution

```
#include <stdio.h>
void main()
{
 int m1,m2,m3,m4,m5,t;
 float p;
 printf("Enter 5 subject marks ");
 scanf("%d%d%d%d%d",&m1,&m2,&m3,&m4,&m5);
 t=m1+m2+m3+m4+m5;
 p=t/5.0;
 printf("Total=%d\n Percentage=%.2f",t,p);

  if(p>=60)
  {
        printf("Grade A");
  }
  else if(p>=50)
  {
        printf("Grade B");
  }
  else if(p>=40)
  {
        printf("Grade C");
  }
  else
  {
        printf("Grade F");
  }
}
```

Navlakhi®

**Constant Multiway Conditional Statement**

```
switch(expression)
{
case value-1:  ……
               break;
case value-2:  ……
               break;
……
default:       …….
               break;
}
```

Only one statement block will be executed, because of the break.
Switch should not be used for a range of values.
If distinct values have different actions to be performed then you can use switch.
Case acts as an entry point & break acts like the exit point for the switch.
The expression is executed & the value of the expression is matched with the case values to decide which case can be executed. If none satisfy then it chooses the default case.

**PE22:** Write a program to simulate a basic arithmetic expression solver.
The user enters say
2.3+5.1
The result should be 7.400000
Similarly for - , * , / and ^.
General form
operand1 operator operand2

```
#include <stdio.h>
#include <math.h>
void main()
{
 float a,c,t;
 char b;
 printf("Enter Expression ");
 scanf("%f%c%f",&a,&b,&c);

 if(b=='+')
 {
        t=a+c;
        printf("Answer is =%f",t);
 }

 if(b=='–')
 {
        t=a–c;
        printf("Answer is =%f",t);
 }
 if(b=='*')
 {
        t=a*c;
        printf("Answer is =%f",t);
 }
if(b=='^')
 {
        t=pow(a,c);
        printf("Answer is =%f",t);
 }
```

```
 if(b=='/')
 {
        if(c==0)
        {
                printf("Infinity");
        }
        else
        {
                t=a/c;
                printf("Answer is =%f",t);
        }
  }
}
```

Alternate Solution

```
#include <stdio.h>
#include <math.h>
void main()
{
 float a,c,t;
 char b;
 printf("Enter Expression ");
 scanf("%f%c%f",&a,&b,&c);

 switch(b)
 {
        case '+':       t=a+c;
                        printf("Answer is =%f",t);
                        break;

        case '–':       t=a–c;
                        printf("Answer is =%f",t);
                        break;
        case '*':       t=a*c;
                        printf("Answer is =%f",t);
                        break;
        case '^':       t=pow(a,c);
                        printf("Answer is =%f",t);
                        break;
```

```
        case '/':       if(c==0)
                        {
                                printf("Infinity");
                        }
                        else
                        {
                                t=a/c;
                                printf("Answer is =%f",t);
                        }
                        break;
        default:        printf("Invalid Operator");
                        break;
}
}
```

**PE23:** Modify the above program to handle X and x as the alternative signs for *.

```
#include <stdio.h>
#include <math.h>
void main()
{
 float a,c,t;
 char b;
 printf("Enter Expression ");
 scanf("%f%c%f",&a,&b,&c);

 if(b=='+')
 {
        t=a+c;
        printf("Answer is =%f",t);
 }

 if(b=='–')
 {
        t=a–c;
        printf("Answer is =%f",t);
 }

 if(b=='*' || b=='x' || b=='X')
 {
        t=a*c;
        printf("Answer is =%f",t);
 }

 if(b=='^')
 {
        t=pow(a,c);
        printf("Answer is =%f",t);
 }
```

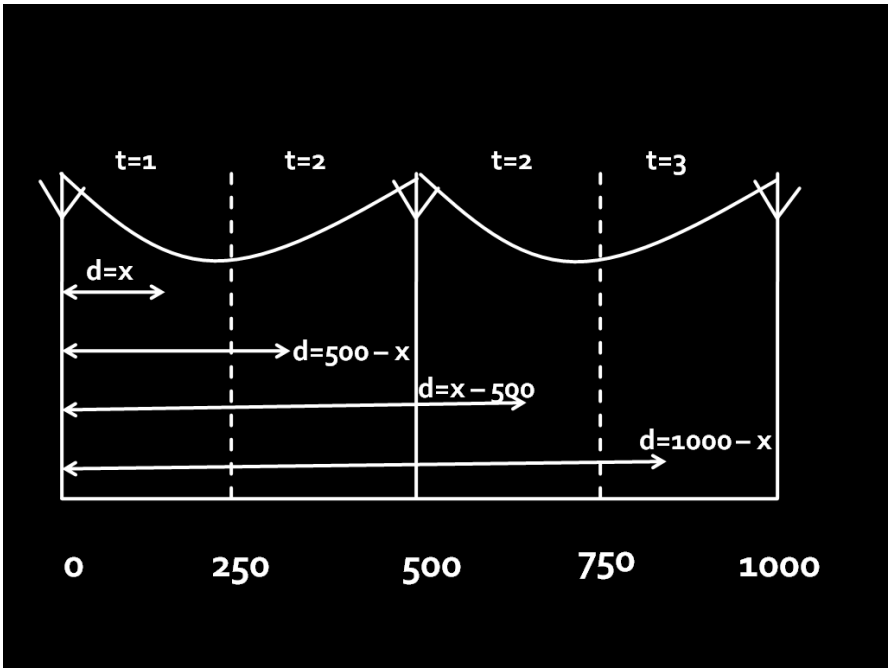
```
 if(b==‘/’)
 {
        if(c==0)
        {
                printf(“Infinity”);
        }
        else
        {
                t=a/c;
                printf(“Answer is =%f”,t);
        }
  }
}
```

**Alternate Solution**

```
#include <stdio.h>
#include <math.h>
void main()
{
 float a,c,t;
 char b;
 printf(“Enter Expression ”);
 scanf(“%f%c%f”,&a,&b,&c);

 switch(b)
 {
        case ‘+’:       t=a+c;
                        printf(“Answer is =%f”,t);
                        break;

        case ‘–’:       t=a–c;
                        printf(“Answer is =%f”,t);
                        break;
        case ‘*’:
        case ‘x’:
        case ‘X’:       t=a*c;
                        printf(“Answer is =%f”,t);
                        break;
        case ‘^’:       t=pow(a,c);
                        printf(“Answer is =%f”,t);
                        break;
```

```
        case '/':       if(c==0)
                        {
                                printf("Infinity");
                        }
                        else
                        {
                                t=a/c;
                                printf("Answer is =%f",t);
                        }
                        break;
        default:        printf("Invalid Operator");
                        break;
}
}
```

**PE24:** A 1000-foot cable for a cable car is stretched between two towers with a supporting tower midway between the two end towers. The velocity of the cable car depends on its position on the cable. When the cable car is within 30 ft of a tower its velocity v is given in ft/sec as $v = 2.425 + 0.00175d^2$ where d is the distance in feet from the cable car to the nearest tower. If the cable car is not within 30ft of a tower its velocity if given by $v = 0.625 + 0.12d - 0.00025d^2$
Write a program that reads the position of the cable car as the distance in feet from the 1st tower and outputs the number of the nearest tower & velocity.




```
#include <stdio.h>
#include <math.h>
void main()
{
 float x,d,v;
 printf("Enter distance from 1st tower ");
 scanf("%f",&x);

 if(x<=250)
 {
        printf("Closest tower is 1\n");
        d=x;
 }
 else if(x<=500)
 {
        printf("Closest tower is 2\n");
        d=500 – x;
 }
```

```
else if(x<=750)
 {
        printf("Closest tower is 2\n");
        d=x – 500;
 }
else
 {
        printf("Closest tower is 3\n");
        d=1000 – x;
 }

 if(d<=30)
 {
        v=2.425+0.00175*d*d;
 }
else
 {
        v=0.625+0.12*d – 0.00025*d*d;
 }
 printf("Velocity= %f ",v);
}
```

**PE25:** Write a program that reads the real coefficients a, b and c of the quadratic equation $ax^2+bx+c=0$ and computes the real roots.

```
#include <stdio.h>
#include <math.h>
void main()
{
  float a,b,c,r1,r2;
  printf("Enter Coefficients ");
  scanf("%f%f%f",&a,&b,&c);

  if(a==0)
 {
        printf("Not  a quadratic");
 }
 else if(b*b – 4*a*c<0)
{
        printf("Roots are not real");
}
else
{
        r1=(-b+sqrt(b*b-4*a*c))/(2*a);
        r2=(-b-sqrt(b*b-4*a*c))/(2*a);
        printf("Roots are %f and %f",r1,r2);
}
}
```

**PE26:** Write a program that reads three positive numbers a,b,c and determines whether the triangle will be an obtuse-angled, or a right-angled, or an acute-angled triangle. If the triangle is an acute-angled triangle, determine further whether the triangle is equilateral, isosceles, or scalene.

```
#include <stdio.h>
void main()
{
 float a,b,c;
 printf("Enter sides ");
 scanf("%f%f%f",&a,&b,&c);

 if(a+b>c && b+c>a && a+c>b)
 {
        if(a*a>b*b+c*c||b*b>a*a+c*c||c*c>a*a+b*b)
                printf("Obtuse triangle ");
        else if(a*a==b*b+c*c||b*b==a*a+c*c|| c*c==a*a+b*b)
                printf("Rt angled triangle ");
        else
        {
                printf("Acute triangle ");
                if(a==b && b==c)
                        printf("Equilateral triangle");
                else if(a==b||b==c||a==c)
                        printf("Isosceles triangle");
                else
                        printf("scalene triangle");
        }
 }
else
{
        printf("Not a triangle");
}
}
```

**PE27:** An electronic component vendor supplies three products: transistors, resistors and capacitors. The vendor gives a discount of 10% on orders for transistors if the order is more than Rs.1000. On orders of more than Rs.100 for resistors, a discount of 5% is given and a discount of 10% is given on orders for capacitors of value more than Rs.500. Assume numeric codes of 1,2 and 3 are used for transistors, capacitors and resistors respectively. Write a program that reads the product code and the order amount, and prints out the net amount that the customer is required to pay after discount.

**PE28:** Write a temperature conversion program. The program should read the temperature measured and the type of scale used in the measurement (F for Fahrenheit and C for Celsius) and convert it to the other scale.

```
#include <stdio.h>
void main()
{
  float t,ans;
  char s;
  printf("Enter temperature ");
  scanf("%f%c",&t,&s);

  if(s=='C')
 {
        ans=t/5*9+32;
        printf("%fF",ans);
 }
else if(s=='F')
 {
        ans=(t – 32)/9*5;
        printf("%fC",ans);
 }
else
{
        printf("Invalid scale");
}
}
```

**P29:** A function f is defined as follows:

$f(x)=ax^3 – bx^2 + cx –d,$ if $x > k$

$f(x)=0$ if $x = k$

$f(x)= –ax^3+bx^2-cx+d,$ if $x < k$

Write a program that reads the values of a, b, c, d, k, and x, and prints the value of f(x).

**P30:**

In a simple thin lens, the optical axis is the line through the center of the lens and joining the centers of curvature of the two surfaces. If the lens is used to form an image of an object, then the relation 1/u + 1/v = 1/f holds, where u, the distance of object from lens, v, the distance of image from lens, and f, the principle focal length of the lens, are measured along the optical axis. Note that if u=∞ then v=f, and if u=f then v=∞.

Write a program that accepts the focal length of the lens an the position of the object, and prints out the position of the image.

**P31**

The commission on a salesman's total sales is as follows:
☺Sales<100 => No commission
☺100 ≤ Sales <1000 => Commission = 10% of the sales value
☺Sales ≥ 1000 => Commission = 100 + 12% of sales above 1000.
Write a program that reads sales and prints out the salesman's commission.

```
#include <stdio.h>
void main()
{
 float s,c;
 printf("Enter sales ");
 scanf("%f",&s);

 if(s<100)
 {
        printf("No commission");
 }
 else if(s<1000)
 {
        c=0.1*s;
        printf("Commission=%.2f",c);
 }
else if(s>=1000)
 {
        c=100+0.12*(s – 1000);
        printf("Commission=%.2f",c);
 }
}
```

**P33** COOL Inc. has classified its employees into 4 categories:
Class 1: Rs.10 per hours for regular hours and no overtime.
Class 2 or 3: Rs.7 per hour for regular hours, and overtime hours at the rate of 1.5 times the rate for the regular hours.
Class 4: Rs.5 per hour for regular hours, and overtime hours at the rate of 2.0 times the rate for regular hours for overtime hours, up to a time equal to regular hours. The rate is 2.5 times the regular rate for excess overtime hours.
Write a program that reads an employee's classification and regular and overtime hours and prints the employee's pay. An error message should be printed if a classification number other than 1, 2, 3, or 4 is read.

```
#include <stdio.h>
void main()
{
 float r,o,s;
 int  c;
 printf("Enter class, regular hrs & overtime hrs ");
 scanf("%d%f%f",&c,&r,&o);
 if(c==1)
 {
        s=10*r;
        printf("Wages=%.2f",s);
 }
  else if(c==2||c==3)
 {
        s=7*r+1.5*7*o;
        printf("Wages=%.2f",s);
 }
 else if(c==4)
 {
        if(o>r)
              s=5*r+2*5*r+2.5*5*(o –r);
        else
              s=5*r+2*5*o;
        printf("Wages=%.2f",s);
 }
 else
        printf("Invalid class");
}
```

**P34** A semiconductor manufacturer sells three types of microprocessors: 8-bit, 16-bit, and 32-bit. It differentiates between three types of customers: industrial, government, and university. It has the following discount policy that depends on the type of microprocessor, the amount of the order, and type of customer: For 32-bit microprocessor, if the order is for less that 50,000, allow 5% discount to industrial customers and 6.5% discount to government agencies. If the order is 50,000 or more, discounts of 7.5% and 8.5% are given to industrial customers and government agencies respectively. A discount of 10% is given to both industrial and government agencies if the order is more than 1,00,000. Universities get a discount of 7.5% irrespective of the amount of order. For 16 – bit microprocessors, no discount is given for orders less than 10,000. For orders of 10,000 or more, 5% discount is given to industrial customers and universities, and 6% discount to government agencies. For 8-bit microprocessors, a flat discount of 10% is given to all types of customers for any order. Write a program that reads the type of the customer, the type of the product, and the amount of the order, and print the net amount payable by the customer.

Navlakhi®


# Chapter 3

# Repititive Programming

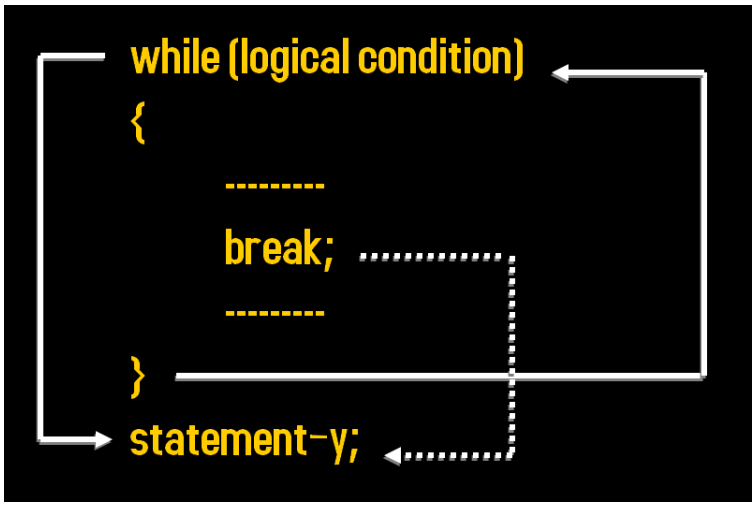
**while loop**

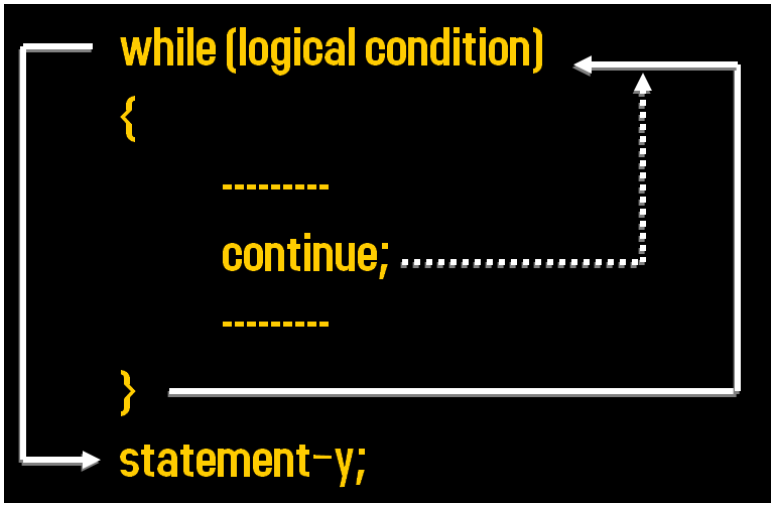
```
while (logical condition)
{
        body of loop
}
statement–y;
```

The logical condition is evaluated & on success the loop body is executed & the condition is checked again. On success the loop body is executed again & again the loop condition is checked….. and so on… till this logical condition fails. Once the logical condition fails the loop body is skipped and program execution continues from statement-y.
If the logical condition had failed the very first time it was checked then the program would have directly jumped to statement-y.

***Loop interruption***

Loop execution can be interrupted using either break or continue.
'break' takes the control out of the loop & 'continue' take the control back to the start of the loop



**do - while loop**

```
do
{
        body of loop
} while (logical condition);
statement–y;
```

The body of the loop is executed first, then the logical condition is checked. On success the control transfers to the 'do' and on then the Loop body is executed again & again the condition is checked & if its TRUE again then again the control transfers to the 'do' and so on….. This stops when the logical condition fails. When it fails the control transfers to statement-y.

***do - while loop interruption***

The rules of 'break' and 'continue' are the same for do-while as they were for while.

**for loop**

Another form of loop structure is the for loop.

It consists of 3 parts

- The initialization section (executed 1st and only once)
- The condition
- The step

```
for (initialization ; logical condition ; step)
{
        loop body
}
        statement–y;
```

```
for (initialization ; logical condition ; step)
{
        loop body
}
        statement–y;
```

First the initialization takes place

```
for (initialization ; logical condition ; step)
{
        loop body
}
        statement–y;
```

Next, the logical condition is checked for TRUE/FALSE

```
for (initialization ; logical condition ; step)
{
        loop body
}
        statement–y;
```

If TRUE then the loop body is executed

```
for (initialization ; logical condition ; step)
{
        loop body
}
        statement–y;
```

After the loop body is executed the step comes into play, which is usually an increment/decrement.

```
for (initialization ; logical condition ; step)
{
        loop body
}
        statement–y;
```

Next, the logical condition is check again for TRUE/FALSE. If TRUE the loop body is executed again, then the step & the logical check once again.

The procedure repeats again and again. This cycle (loop) stops when the logical condition fails. Then the program control transfers to the statement-y.

***Interruption of for loop***

The for loop can be interrupted the same way as while & do-while i.e. using the break or continue. The break will take us out of the for loop (to statement-y) and the continue will take us to the step & then the logical condition & then depending on the TRUE/FALSE value the loop body will be executed or the control will transfer to statement-y.

**PE35:** Print all integers from 1 to 100

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

**PE36:** Print the 1st N natural numbers

```
/* PE36 */
#include <stdio.h>
void main()
{
  int i,N;
  printf("Feed in an integer ");
  scanf("%d",&N);
  for(i=1;i<=N;i=i+1)
  {
    printf("%d\t",i);
  }
}
```

**PE37:** Print even numbers from 1 to N

```
#include <stdio.h>
void main()
{
 int i,N;
 printf("Feed in an integer ");
 scanf(“%d”,&N);
 for(i=2;i<=N;i=i+2)
 {
  printf(“%d\t”,i);
 }
}
```

**ALTERNATE METHOD**

```
#include <stdio.h>
void main()
{
 int i,N;
 printf("Feed in an integer ");
 scanf(“%d”,&N);
 for(i=1;i<=N;i=i+1)
 {
  if (i%2==0)
  {
   printf(“%d\t”,i);
  }
 }
}
```

**PE38:** Print all odd numbers from 1 to N

```
#include <stdio.h>
void main()
{
 int i,N;
 printf("Feed in an integer ");
 scanf(“%d”,&N);
 for(i=1;i<=N;i=i+2)
 {
  printf(“%d\t”,i);
 }
}
```

**ALTERNATE METHOD**

```
#include <stdio.h>
void main()
{
 int i,N;
 printf("Feed in an integer ");
 scanf(“%d”,&N);
 for(i=1;i<=N;i=i+1)
 {
  if (i%2!=0)
  {
   printf(“%d\t”,i);
  }
 }
}
```

**PE39:** Print all multiples of 3 from 1 to N.

```
#include <stdio.h>
void main()
{
 int i,N;
 printf("Feed in an integer ");
 scanf(“%d”,&N);
 for(i=3;i<=N;i=i+3)
 {
  printf(“%d\t”,i);
 }
}
```

**ALTERNATE METHOD**

```
#include <stdio.h>
void main()
{  int i,N;
 printf("Feed in an integer ");
 scanf(“%d”,&N);
 for(i=1;i<=N;i=i+1)
 {
  if (i%3==0)
  {
   printf(“%d\t”,i);
  }
 }
}
```

**PE40:** Print all multiples of 3 and 7 from 1 to N

```
#include <stdio.h>
void main()
{
 int i,N;
 printf("Feed in an integer ");
 scanf(“%d”,&N);
 for(i=21;i<=N;i=i+21)
 {
  printf(“%d\t”,i);
 }
}
```

**ALTERNATE METHOD**

```
#include <stdio.h>
void main()
{  int i,N;
 printf("Feed in an integer ");
 scanf(“%d”,&N);
 for(i=1;i<=N;i=i+1)
 {
  if (i%3==0 && i%7==0)
  {
   printf(“%d\t”,i);
  }
 }
}
```

**PE41:** Print all multiples of 3 or 7 from 1 to N

```
#include <stdio.h>
void main()
{  int i,N;
  printf("Feed in an integer ");
  scanf(“%d”,&N);
  for(i=1;i<=N;i=i+1)
  {
    if (i%3==0 || i%7==0)
    {
      printf(“%d\t”,i);
    }
  }
}
```

**PE42:** Print the sum of the 1st N natural numbers

```
#include <stdio.h>
void main()
{  int i,N,sum;
  printf("Feed in the last term ");
  scanf(“%d”,&N);
  sum=0;
  for(i=1;i<=N;i=i+1)
  {
    sum=sum+i;
  }
  printf("Sum=%d",sum);
}
```

**PE43:** Find the factorial of a number.

```
#include <stdio.h>
void main()
{  int i,N;
  long int p;
  printf("Feed in the term ");
  scanf(“%d”,&N);
  p=1;
  for(i=1;i<=N;i=i+1)
  {
    p=p*i;
  }
  printf("Factorial=%ld",p);
}
```

**PE44:** Print the multiplication table of 13. The output should be displayed as:

13 X 1 = _____
13 X 2 = _____
….
13 X 12=_____

```
#include <stdio.h>
void main()
{
  int i;
  for(i=1;i<=12;i=i+1)
  {
    printf("13X%d=%d\n",i,13*i);
  }
}
```

**PE45:** Print the multiplication table of 13 from 13 X 1 to 13 X N.

```
#include <stdio.h>
void main()
{
 int i,N;
 printf("Feed in the upper limit ");
 scanf(“%d”,&N);
 for(i=1;i<=N;i=i+1)
 {
  printf("13X%d=%d\n",i,13*i);
 }
}
```

**PE46:** Print the multiplication table from P X M to P X N

```
#include <stdio.h>
void main()
{  int i,p,M,N;
 printf("Feed in the multipler and the range of multiplicand ");
 scanf(“%d%d%d”,&p,&M,&N);
 if (M<N)
 {
        for(i=M;i<=N;i=i+1)
        {
                printf(“%dX%d=%d\n”,p,i,p*i);
        }
 }
 else
 {
        for(i=M;i>=N;i=i-1)
        {
                printf(“%dX%d=%d\n”,p,i,p*i);
        }
 }
}
```

**PE47:** Print the sum of the following series
(i) 1+2+3+….. N terms
(ii) 1+3+5+….. N terms
(iii) 2+4+6+….. N terms
(iv) 1+4+9+16+…. N terms
(v) 1+8+27+….N terms
(vi) 1+4+16+64+….. N terms
(vii) 1*3+2*6+3*9+….. N terms
(viii) 1+3+5+….. N terms
2+4+6+…. N terms
(ix) 1*3+2*6+3*9+…. N terms
1*4+3*8+5*12+… M terms

(i)

```
#include <stdio.h>
void main()
{ int i,   N,term=1,sum=0;
  printf("Feed in the number of terms:");
  scanf(“%d”,&N);
  for(i=1;i<=N;i=i+1)
  {
        sum=sum+term;
        term=term+1;
  }
  printf("The sum =%d",sum);
}
```

(ii)

```
#include <stdio.h>
void main()
{ int i,   N,term=1,sum=0;
  printf("Feed in the number of terms:");
  scanf(“%d”,&N);
  for(i=1;i<=N;i=i+1)
  {
        sum=sum+term;
        term=term+2;
  }
  printf("The sum =%d",sum);
}
```

(iii)

```
#include <stdio.h>
void main()
{ int i,   N,term=2,sum=0;
  printf("Feed in the number of terms:");
  scanf(“%d”,&N);
  for(i=1;i<=N;i=i+1)
  {
        sum=sum+term;
        term=term+2;
  }
  printf("The sum =%d",sum);
}
```

(iv)

```
#include <stdio.h>
void main()
{ int i,   N,term=1,sum=0;
  printf("Feed in the number of terms:");
  scanf(“%d”,&N);
  for(i=1;i<=N;i=i+1)
  {
        sum=sum+term*term;
        term=term+1;
  }
  printf("The sum =%d",sum);
}
```

(v)

```
#include <stdio.h>
void main()
{ int i,   N,term=1,sum=0;
  printf("Feed in the number of terms:");
  scanf(“%d”,&N);
  for(i=1;i<=N;i=i+1)
  {
        sum=sum+term*term*term;
        term=term+1;
  }
  printf("The sum =%d",sum);
}
```

(vi)

```
#include <stdio.h>
void main()
{
  int i,   N,term=1,sum=0;
  printf("Feed in the number of terms:");
  sacnf(“%d”,&N);

  for(i=1;i<=N;i=i+1)
  {
        sum=sum+term;
        term=4*term;
  }

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

(vii)

```
#include <stdio.h>
void main()
{
  int i,   N,term1=1,term2=3,sum=0;
  printf("Feed in the number of terms:");
  scanf(“%d”,&N);

  for(i=1;i<=N;i=i+1)
  {
        sum=sum+term1*term2;
        term1=term1+1;
        term2=term2+3;
  }

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

(viii)

```
#include <stdio.h>
void main()
{
  int i,   N,term1=1,term2=2,sum1=0,sum2=0;
  float sum;
  printf("Feed in the number of terms:");
  scanf(“%d”,&N);

  for(i=1;i<=N;i=i+1)
  {
   sum1=sum1+term1;
   sum2=sum2+term2;
   term1=term1+2;
   term2=term2+2;
  }

  sum=(float)sum1/sum2;
  printf("The Answer =%f",sum);
}
```

(ix)

```
#include <stdio.h>
void main()
{
   int i,   N,M,term1=1,term2=3,term3=1, term4=4,sum1=0,sum2=0;
   float sum;

  printf("Feed in the number of terms in the numerator:");
  scanf(“%d”,&N);
  printf("Feed in the number of terms in the denominator:");
  scanf(“%d”,&M);

  for(i=1;i<=N;i=i+1)
  {
        sum1=sum1+term1*term2;
        term1=term1+1;
        term2=term2+3;
  }

  for(i=1;i<=M;i=i+1)
  {
        sum2=sum2+term3*term4;
        term3=term3+2;
        term4=term4+4;
  }

  sum=(float)sum1/sum2;
  printf("The Answer =%f",sum);
}
```

**PE48:** Read N number and check how many are +ve, -ve and zero.

```
#include <stdio.h>
void main()
{  int i, N, pcount=0, ncount=0, zcount=0, n;
  printf("Feed in the number of terms ");
  scanf(“%d”,&N);
  for(i=1;i<=N;i++)
  {
    printf("Feed in the data ");
    scanf(“%d”,&n);
    if(n<0)
    {
      ++ncount;
    }
else if(n>0)
       {
         ++pcount;
       }
       else
       {
         ++zcount;
       }
  }
  printf("Number of positive terms=%d\nNumber of -ve terms=%d\nNumber of
zero's=%d",pcount,ncount,zcount);
}
```

Navlakhi®

**PE49:** Read N numbers are check how many are even and odd.

```
#include <stdio.h>
void main()
{
  int i, N, ecount=0, ocount=0, n;
  printf("Feed in the number of terms ");
  scanf(“%d”,&N);
  for(i=1;i<=N;i++)
  {
    printf("Feed in the data ");
    scanf(“%d”,&n);
    if(n%2==0)
    {
      ++ecount;
    }
   else
    {
      ++ocount;
    }
  }
  printf("Number of even numbers=%d\nNumber of odd numbers=%d",ecount,ocount);
}
```

**PE50:** Read N numbers & find maximum.

```
#include <stdio.h>
void main()
{
 int i, N, max, n;
 printf("Feed in the number of terms ");
 scanf(“%d”,&N);
 for(i=1;i<=N;i++)
 {
  printf("Feed in the data ");
  scanf(“%d”,&n);
  if(i==1)
  {
    max=n;
  }
 else
  {
        if(n>max)
        {
              max=n;
        }
  }
 }
 printf("Maximum=%d",max);
}
```

**PE51:**Read N numbers and find minimum

```
#include <stdio.h>
void main()
{
 int i, N, min, n;
 printf("Feed in the number of terms ");
 scanf(“%d”,&N);
 for(i=1;i<=N;i++)
 {
  printf("Feed in the data ");
  scanf(“%d”,&n);
  if(i==1)
  {
   min=n;
  }
 else
  {
        if(n<min)
        {
              min=n;
        }
  }
 }
 printf("Minimum=%d",min);
}
```

**PE52:** Read N numbers and find the maximum, minimum & their positions.

```
#include <stdio.h>
void main()
{  int i, N, min, max, maxpos, minpos, n;
  printf("Feed in the number of terms ");
  scanf(“%d”,&N);
  for(i=1;i<=N;i++)
  {
    printf("Feed in the data ");
    scanf(“%d”,&n);
    if(i==1)
    {
      max=n;
      maxpos=1;
      min=n;
      minpos=1;
    }
    else
    {
          if(n>max)
          {
                max=n;
                maxpos=i;
          }
          if(n<min)
          {
                min=n;
                minpos=i;
          }
    }
  }
  printf("Minimum=%d\tPosition of min=%d\n",min,minpos);
  printf("Maximum=%d\tPosition of max=%d\n"max,maxpos);
}
```

**PE53:** Read N numbers & find the highest & second highest.

```
#include <stdio.h>
void main()
{
  int i, N, max1, max2, n;
 printf("Feed in the number of terms ");
 scanf(“%d”,&N);
 for(i=1;i<=N;i++)
 {
  printf("Feed in the data ");
  scanf(“%d”,&n);
  if(i==1)
  {
    max1=n;
  }
  else  if(i==2)
      {
          if(n>max1)
          {
                  max2=max1;
                  max1=n;
          }
          else
                  max2=n;
     }

   else if(i>=3)
    {
          if(n>max1)
          {
                  max2=max1;
                  max1=n;
          }
          else
                  if(n>max2)
                  {max2=n;}
          }
   }
 }
 printf("Maximum=%d\n2nd Maximum=%d",max1,max2);
}
```

**PE54:** Check if a number is prime or not.

```
#include <stdio.h>
void main()
{
 int i,n;
 printf("Feed in a number ");
 scanf(“%d”,&n);

 if (n==1)
        printf("1 is non - prime");
 else
 {
        for(i=2;i<=n-1;i++)
        {
                if(n%i==0)
                        break;
        }

          if(i<=n-1) {printf("%d  is not prime",n);}
          else    {printf("%d is prime",n);}
}
}
```

**PE55:** Print all prime numbers from 1 to N

```
#include <stdio.h>
void main()
{
 int i,n,N;
 printf("Feed in the upper limit ");
 scanf(“%d”,&N);

 for(n=2;n<=N;n++)
 {
        for(i=2;i<=n-1;i++)
        {
                if(n%i==0)
                        break;
        }
        if(i<=n-1)
                {}
        else
                printf(“%d”,n);
 }
}
```

**PE56:** Print all perfect squares from 1 to N

```
#include <stdio.h>
void main()
{
 int i,N;
 printf("Feed in the upper limit ");
 scanf(“%d”,&N);

 for(i=1; ;i++)
 {
  if(i*i>N)
        break;
  else
        printf(“%d\n”,i*i);
 }
}
```

**PE57:** Print a rectangle of * with 60 *’s on each row & totally n rows, where n is a user input.

```
#include <stdio.h>
void main()
{
 int n,r,s;
 printf("Feed in the number of rows ");
 scanf(“%d”,&n);
 for(r=1;r<=n;r++)
 {
        for(s=1;s<=60;s++)
        {
              printf("*");
        }
        printf(“\n”);
 }
}
```

Navlakhi®

**Basic steps for design questions:**

```
Loop for Row numbering                                  //Step 1
{
        Loop for printing leading spaces on one row     //Step 2
        {
                ……
        }

        Loop for design printing of one row             //Step 3
        {
                …….
        }

        Move to next line                               //Step 4
}
```

Navlakhi®

**PE58:** Print the following designs

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

```
#include<stdio.h>
void main()
{
  int r,s,sp;
  for(sp=0,r=6;r>=1;r--,sp++)
  {
    for(s=1;s<=sp;s++)
    {printf(" ");}

    for(s=1;s<=r;s++)
    {printf( "*");}

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

```
11111
 2222
  333
   44
    5
```

```
#include<stdio.h>
void main()
{
  int r,s,sp,data;
  data=1;


  for(sp=0,r=5;r>=1;r--,sp++)
  {
    for(s=1;s<=sp;s++)
    { printf(" ");}

    for(s=1;s<=r;s++)
    {
        printf("%d",data);
    }
    printf("\n");
    data=data+1;
  }
}
```

```
12345
 1234
  123
   12
    1
```

```
#include<stdio.h>
void main()
{
  int r,s,sp,data;


  for(sp=0,r=5;r>=1;r--,sp++)
  {
    data=1;

    for(s=1;s<=sp;s++)
    { printf(" ");}

    for(s=1;s<=r;s++)
    {
        printf("%d",data);
        data=data+1;
    }
    printf("\n");
  }
}
```

```
abcde
 abcd
  abc
   ab
    a
```

```
#include<stdio.h>
void main()
{
  int r,s,sp;
  char data;


  for(sp=0,r=5;r>=1;r--,sp++)
  {
    data='a';

    for(s=1;s<=sp;s++)
    { printf(" ");}

    for(s=1;s<=r;s++)
    {
        printf("%c",data);
        data=data+1;
    }
    printf("\n");
  }
}
```

```
abcde
 fghi
  jkl
   mn
    o
```

```
#include<stdio.h>
void main()
{
  int r,s,sp;
  char data;

  data='a';


  for(sp=0,r=5;r>=1;r--,sp++)
  {

    for(s=1;s<=sp;s++)
    { printf(" ");}

    for(s=1;s<=r;s++)
    {
        printf("%c",data);
        data=data+1;
    }
    printf("\n");
  }
}
```

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

```
#include<stdio.h>
void main()
{
  int r,s,sp;
  for(sp=0,r=6;r>=1;r--,sp++)
  {
    for(s=1;s<=sp;s++)
    {printf(" ");}

    for(s=1;s<=r;s++)
    {   if(s==1||s==r||r==6)
        {printf( "*");}
        else printf(" ");
    }
    printf("\n");
  }
}
```

NOTE: ‘s’ is the column number (or star number, in that row).

*
**
***
****
*****
******

```
#include<stdio.h>
void main()
{
  int r,s;

  for(r=1;r<=6;r++)
  {

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

1
22
333
4444
55555
666666

```
#include<stdio.h>
void main()
{
  int r,s,data;
  data=1;
  for(r=1;r<=6;r++)
  {

    for(s=1;s<=r;s++)
    {
        printf("%d",data);
    }
    printf("\n");
    data=data+1;
  }
}
```

1
12
123
1234
12345
123456

```
#include<stdio.h>
void main()
{
  int r,s,data;

  for(r=1;r<=6;r++)
  {
    data=1;
    for(s=1;s<=r;s++)
    {
        printf("%d",data);
        data=data+1;
    }
    printf("\n");
  }
}
```

a
ab
abc
abcd
abcde
abcdef

```
#include<stdio.h>
void main()
{
  int r,s;
  char data;

  for(r=1;r<=6;r++)
  {
    data='a';
    for(s=1;s<=r;s++)
    {
        printf("%c",data);
        data=data+1;
    }
    printf("\n");
  }
}
```

a
bc
def
g hi j
k lmno
pqr s tu

```
#include<stdio.h>
void main()
{
  int r,s;
  char data;
  data='a';
  for(r=1;r<=6;r++)
  {
    for(s=1;s<=r;s++)
    {
        printf("%c",data);
        data=data+1;
    }
    printf("\n");
  }
}
```

*
**
* *
*  *
*   *
******

```
#include<stdio.h>
void main()
{
  int r,s;

  for(r=1;r<=6;r++)
  {
    for(s=1;s<=r;s++)
    {
        if(s==1||s==r||r==6)
        {
            printf("*");
        }
        else
        {
            printf(" ");
        }
    }
    printf("\n");
  }
}
```

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

```
#include<stdio.h>
void main()
{
  int r,s,sp;
  for(sp=40,r=1;r<=6;r++,sp--)
  {
    for(s=1;s<=sp;s++)
    {printf(" ");}

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

```
     1
    22
   333
  4444
 55555
666666
```

```
#include<stdio.h>
void main()
{
  int r,s,sp,data;
  data=1;
  for(sp=40,r=1;r<=6;r++,sp--)
  {
    for(s=1;s<=sp;s++)
    {printf(" ");}

    for(s=1;s<=r;s++)
    {
        printf("%d",data);
    }
    printf("\n");
    data=data+1;
  }
}
```

```
     1
    12
   123
  1234
 12345
123456
```

```
#include<stdio.h>
void main()
{
  int r,s,sp,data;

  for(sp=40,r=1;r<=6;r++,sp--)
  {
    data=1;
    for(s=1;s<=sp;s++)
    {printf(" ");}

    for(s=1;s<=r;s++)
    {
        printf("%d",data);
        data=data+1;
    }
    printf("\n");
  }
}
```

```
     a
    ab
   abc
  abcd
 abcde
abcdef
```

```
#include<stdio.h>
void main()
{
  int r,s,sp;
  char data;

  for(sp=40,r=1;r<=6;r++,sp--)
  {
    data='a';
    for(s=1;s<=sp;s++)
    {printf(" ");}

    for(s=1;s<=r;s++)
    {
        printf("%c",data);
        data=data+1;
    }
    printf("\n");
  }
}
```

```
     a
    bc
   def
  gh i j
 klmno
pqrs t u
```

```
#include<stdio.h>
void main()
{
  int r,s,sp;
  char data;

  data='a';
  for(sp=40,r=1;r<=6;r++,sp--)
  {
    for(s=1;s<=sp;s++)
    {printf(" ");}

    for(s=1;s<=r;s++)
    {
        printf("%c",data);
        data=data+1;
    }
    printf("\n");
  }
}
```

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

```
#include<stdio.h>
void main()
{
  int r,s,sp;

  for(sp=40,r=1;r<=6;r++,sp--)
  {
    for(s=1;s<=sp;s++)
    {printf(" ");}

    for(s=1;s<=r;s++)
    {
        if(s==1||s==r||r==6)
                printf("*");
        else
                printf(" ");
    }
    printf("\n");
  }
}
```

Navlakhi®

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

```
#include<stdio.h>
void main()
{
  int r,s,sp;

  for(sp=40,r=1;r<=4;r++,sp--)
  {
    for(s=1;s<=sp;s++)
    {printf(" ");}

    for(s=1;s<=2*r-1;s++)
    {
        printf("*");
    }
    printf("\n");
  }
}
```

```
   1
  222
 33333
4444444
```

```
#include<stdio.h>
void main()
{
  int r,s,sp,data;

  data=1;
  for(sp=40,r=1;r<=4;r++,sp--)
  {
    for(s=1;s<=sp;s++)
    {printf(" ");}

    for(s=1;s<=2*r-1;s++)
    {
        printf("%d",data);
    }
    printf("\n");
    data=data+1;
  }
}
```

```
   1
  123
 12345
1234567
```

```
#include<stdio.h>
void main()
{
  int r,s,sp,data;

  for(sp=40,r=1;r<=4;r++,sp--)
  {
    data=1;
    for(s=1;s<=sp;s++)
    {printf(" ");}

    for(s=1;s<=2*r-1;s++)
    {
        printf("%d",data);
        data=data+1;
    }
    printf("\n");
  }
}
```

```
   1
  121
 12321
1234321
```

```
#include<stdio.h>
void main()
{
  int r,s,sp,data;

  for(sp=40,r=1;r<=4;r++,sp--)
  {
    data=1;
    for(s=1;s<=sp;s++)
    {printf(" ");}

    for(s=1;s<=2*r-1;s++)
    {
        if(s<r)
        {
                printf("%d",data);
                data=data+1;
        }
        else
        {
                printf("%d",data);
                data=data-1;
        }

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

```
   a
  abc
 abcde
abcdefg
```

```
#include<stdio.h>
void main()
{
  int r,s,sp;
  char data;

  for(sp=40,r=1;r<=4;r++,sp--)
  {
    data='a';
    for(s=1;s<=sp;s++)
    {printf(" ");}

    for(s=1;s<=2*r-1;s++)
    {
        printf("%c",data);
        data=data+1;
    }
    printf("\n");
  }
}
```

```
   a
  aba
 abcba
abcdcba
```

```
#include<stdio.h>
void main()
{
  int r,s,sp;
  char data;

  for(sp=40,r=1;r<=4;r++,sp--)
  {
    data='a';
    for(s=1;s<=sp;s++)
    {printf(" ");}

    for(s=1;s<=2*r-1;s++)
    {
        if(s<r)
        {
                printf("%c",data);
                data=data+1;
        }
        else
        {
                printf("%c",data);
                data=data-1;
        }

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

Navlakhi®

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

```
#include<stdio.h>
void main()
{
  int r,s,sp;

  for(sp=40,r=1;r<=4;r++,sp--)
  {
    for(s=1;s<=sp;s++)
    {printf(" ");}

    for(s=1;s<=2*r-1;s++)
    {
        if(s==1||s==2*r-1||r==4)
                printf("*");
        else
                printf(" ");
    }
    printf("\n");
  }
}
```

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

```
#include<stdio.h>
void main()
{
  int r,s,sp;

  for(sp=40,r=1;r<=2;r++,sp--)
  {
    for(s=1;s<=sp;s++)
    {printf(" ");}

    for(s=1;s<=2*r-1;s++)
    {
        printf("*");
    }
    printf("\n");
  }

  for(r=3;r>=1;r--,sp++)
  {
        for(s=1;s<=sp;s++)
        {
                printf(" ");
        }

        for(s=1;s<=2*r-1;s++)
        {
                printf("*");
        }

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

**PASCAL'S TRIANGLE**

```
     1
    1 1
   1 2 1
  1 3 3 1
1 4 6 4 1
```

```
#include<stdio.h>
int fact(int n)
{
        int p=1,i;

        for(i=1;i<=n;i++)
        {
                p=p*i;
        }
        return p;
}

void main()
{
  int r,s,sp;

  for(sp=40,r=0;r<=4;r++,sp--)
  {
    for(s=1;s<=sp;s++)
    {printf(" ");}

    for(s=0;s<=r;s++)
    {
        printf("%d ",fact(r)/(fact(s)*fact(r-s)));
    }

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

```
MAHARASHTRA
 MAHARASHTR
  MAHARASHT
   MAHARASH
    MAHARAS
     MAHARA
      MAHAR
       MAHA
        MAH
         MA
          M
```

```
#include<stdio.h>
void main()
{
  int r,s,sp,start_pos;
  char data[]="MAHARASHTRA";
  start_pos=0;
  for(sp=0,r=11;r>=1;r--,sp++)
  {
    for(s=1;s<=sp;s++)
    {printf(" ");}

    for(s=1;s<=r;s++)
    {
        printf("%c",data[start_pos+s-1]);
    }
    printf("\n");
  }
}
```

```
MAHARASHTRA
 AHARASHTRA
  HARASHTRA
   ARASHTRA
    RASHTRA
     ASHTRA
      SHTRA
       HTRA
        TRA
         RA
          A
```

```
#include<stdio.h>
void main()
{
  int r,s,sp,start_pos;
  char data[]="MAHARASHTRA";
  start_pos=0;
  for(sp=0,r=11;r>=1;r--,sp++)
  {
    for(s=1;s<=sp;s++)
    {printf(" ");}

    for(s=1;s<=r;s++)
    {
        printf("%c",data[start_pos+s-1]);
    }
    printf("\n");
    start_pos++;
  }
}
```

**NOTE:** Start position is the array slot number where we start printing. In the first row we start with 'M' hence start_pos=0, then with each row start_pos increases by 1 (i.e. next row starts with 'A'). Hence, start_pos++ after 'endl'.

# Similarly you should be able to do MAHARASHTRA in all other shapes

**PE60:** Finding the sum of digits of a number

```
#include <stdio.h>
void main()
{
 int sum=0,n,digit;
 printf("Feed in a number ");
 scanf(“%d”,&n);
 while(n!=0)
 {
  digit=n%10;
  n=n/10;
  sum=sum+digit;
 }
 printf("Sum of digits=%d",sum);
}
```

**PE61:** Finding product of digits of a number.

```
#include <stdio.h>
void main()
{
  int prod=1,n,digit;
 printf("Feed in a number ");
 scanf(“%d”,&n);
 while(n!=0)
 {
  digit=n%10;
  n=n/10;
  prod=prod*digit;
 }
 printf("Product of digits=%d",prod);
}
```

**PE62:** Finding the sum of squares of the digits of a number.

```
#include <stdio.h>
void main()
{
 int sum=0,n,digit;
 printf("Feed in a number ");
 scanf(“%d”,&n);
 while(n!=0)
 {
  digit=n%10;
  n=n/10;
  sum=sum+digit*digit;
 }
 printf("sum of squares of the digits=%d",sum);
}
```

**PE63:** Finding the sum of cubes of the digits of a number.

```
#include <stdio.h>
void main()
{
 int sum=0,n,digit;
 printf("Feed in a number ");
 scanf(“%d”,&n);
 while(n!=0)
 {
  digit=n%10;
  n=n/10;
  sum=sum+digit*digit*digit;
 }
 printf("sum of cubes of the digits=%d",sum);
}
```

**PE64:** Check whether a number is a Armstrong number or not.

```
#include <stdio.h>
void main()
{
  int sum=0,n,digit,c,y;
 printf("Feed in a number ");
 scanf(“%d”,&n);

 c=0;
 y=n;
 while(y!=0)
 {
        y=y/10;
        c=c+1;
 }

 x=n;
 while(n!=0)
 {      digit=n%10;
        n=n/10;
        sum=sum+pow(digit,c);
 }

 if(x==sum)
        printf("%d  is an Armstrong number",x);
 else
        printf(“%d  is not an Armstrong number",x);
}
```

**PE65:** Print all Armstrong numbers from 1 to N.

```
#include <stdio.h>
void main()
{
  int sum,n,digit,N,y,c;
 printf("Feed in the upper limit ");
 scanf(“%d”,&N);
 for(x=1;x<=N;x++)
 {
        c=0;
        y=x;
        while(y!=0)
        {
              y=y/10;
              c=c+1;
        }

        sum=0;
        n=x;
        while(n!=0)
        {
              digit=n%10;
              n=n/10;
              sum=sum+pow(digit,c);
        }
        if(x==sum)      {printf(“%d\n”,x);}
 }
}
```

**PE66:** Print reverse of a number.

```
#include <stdio.h>
void main()
{
  int r=0,n,digit;
 printf("Feed in a number ");
 scanf(“%d”,&n);

 while(n!=0)
 {
        digit=n%10;
        n=n/10;
        r=r*10+digit;
 }

 printf("Reverse is %d",r);
}
```

**PE67:** Write a menu driven program to perform the basic operations of a calculator.

```
#include<stdio.h>
#include<math.h>
void main()
{
        float a,b,c;
        int choice;
        do
        {
                printf("1. Add\n");
                printf("2. Subtract\n");
                printf("3. Multiply\n");
                printf("4. Divide\n");
                printf("5. Power\n");
                printf("6. Exit\n");
                printf("Enter choice:");
                scanf("%d",&choice);
                switch(choice)
                {
                        case 1: printf("Enter 2 numbers: ");
                                scanf("%f%f",&a,&b);
                                c=a+b;
                                printf("Sum=%f\n",c);
                                break;

                        case 2: printf("Enter 2 numbers: ");
                                scanf("%f%f",&a,&b);
                                c=a-b;
                                printf("Difference=%f\n",c);
                                break;

                        case 3: printf("Enter 2 numbers: ");
                                scanf("%f%f",&a,&b);
                                c=a*b;
                                printf("Product=%f\n",c);
                                break;

                        case 4: printf("Enter 2 numbers: ");
                                scanf("%f%f",&a,&b);
                                if (b==0)
                                        printf("Infinity\n");
```

```
                    else
                    {
                        c=a/b;
                        printf("Diviion=%f\n",c);
                    }
                    break;

                case 5: printf("Enter 2 numbers: ");
                    scanf("%f%f",&a,&b);
                    c=pow(a,b);
                    printf("a^b=%f\n",c);
                    break;
            }
        }while(choice!=6);
}
```

**PE68:** Write a program that determines the GCD of two positive integers.

```
#include <stdio.h>
void main()
{
 int i,m,n;
 printf("Feed in two positive integers ");
 scanf(“%d%d”,&m,&n);
 for(i=m; ;i--)
 {
  if(m%i==0 && n%i==0)
        break;
 }
 printf("GCD=%d\n",i);
 printf("LCM=%d",m*n/i);
}
```

**PE69:** Write a program that determines the GCD by Euclidean algorithm. (The algorithm begins by dividing the 1st number by the 2nd and retaining the remainder. At each successive stage, the previous divisor is divided by the previous remainder. The algorithm stops when the remainder becomes zero. The GCD is the last non-zero remainder (or the last divisor).

```
#include<stdio.h>
void main()
{
 int m,n,r;
 printf("feed in 2 numbers ");
 scanf(“%d%d”,&m,&n);

 while(1)
 {
  r=m%n;

  if (r==0)
  {    break;   }

  m=n;
  n=r;
 }

 printf("GCD=%d",n);
}
```

**PE70:** Write a program that reads a positive integer & classifies it as being deficient, perfect, or abundant. A number is said to be deficient, perfect, or abundant if the sum of its proper divisors is less than the number, equal to the number, or more than the numbers. Example: 4 is deficient since 1+2 <4; 6 is perfect since 1+2+3=6; 12 is abundant since 1+2+3+4+6 > 12.

```
#include<stdio.h>
void main()
{
 int sum=0,n,i;

 printf("feed in the number ");
 scanf("%d",&n);
 for(i=1;i<n;i++)
 {
  if (n%i==0)
  {
   sum=sum+i;
  }
 }
 if (sum<n)
 {
  printf("Number is deficient");
 }
 if(sum==n)
 {
  printf("Number is perfect");
 }
 if(sum>n)
 {
  printf("Number is abundant");
 }
}
```

**PE71:** Three integers a,b and c are such that they form a Pythagorean triplet i.e. $a^2+b^2=c^2$. Write a program that generates all Pythagorean triplets a,b,c, where a, b, c ≤ 25.

```
#include<stdio.h>
void main()
{
 int a,b,c;
 for(a=1;a<=25;a++)
 {
  for(b=1;b<=25;b++)
  {
   for(c=1;c<=25;c++)
   {
    if(a*a+b*b==c*c)
    {
     printf("%d,%d,%d\n",a,b,c);
    }
   }
  }
 }
}
```

**PE72:** Modify the above program to generate triplets a, b, c such that a < b <c and a, b ≤ 25.

```
#include<stdio.h>
void main()
{
 int a,b,c;
 for(a=1;a<=25;a++)
 {
  for(b=1;b<=25;b++)
  {
   for(c=1;c*c<=a*a+b*b;c++)
   {
    if(a*a+b*b==c*c && c>b && b>a)
    {
     printf("%d,%d,%d\n",a,b,c);
    }
   }
  }
 }
}
```

**PE73:** Print the value of sin(x) using sin series

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

float fact(int n)
{
        int i;
        float p=1;

        for(i=1;i<=n;i++)
                p=p*i;

        return p;
}


void main()
{
        float x,sum=0;
        int s,i,N,term;

        printf("Feed in the value whose sin is to be found:");
        scanf("%f",&x);
        x=x*3.14159/180;
        printf("Enter the number of terms:");
        scanf("%d",&N);

        s=1;
        term=1;
        for(i=1;i<=N;i++)
        {
                sum=sum+s*pow(x,term)/fact(term);
                term=term+2;
                s=s*-1;
        }
        printf("sin(%f)=%f\nReal Answer=%f\nError=%f", x*180/3.14159, sum, sin(x),
                                                        fabs(sum-sin(x)));
}
```

**PE74:** Print the value of cos(x) using cos series

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

float fact(int n)
{
        int i;
        float p=1;

        for(i=1;i<=n;i++)
                p=p*i;

        return p;
}


void main()
{
        float x,sum=0;
        int s,i,N,term;

        printf("Feed in the value whose cos is to be found:");
        scanf("%f",&x);
        x=x*3.14159/180;
        printf("Enter the number of terms:");
        scanf("%d",&N);

        s=1;
        term=0;
        for(i=1;i<=N;i++)
        {
                sum=sum+s*pow(x,term)/fact(term);
                term=term+2;
                s=s*-1;
        }
        printf("cos(%f)=%f\nReal Answer=%f\nError=%f",x*180/3.14159,sum,cos(x),
                                                        fabs(sum-cos(x)));
}
```

**PE75:**

(i) Print the first N terms of a Fibonacci series

(ii) Fibonacci series up to N

(i)

```
#include<stdio.h>

void main()
{
	int N,a,b,c,i;

	printf("Enter number of terms:");
	scanf("%d",&N);

	if (N==1) printf("1");
	else
	{
		a=b=1;
		printf("1\t1\t");

		for(i=3;i<=N;i++)
		{
			c=a+b;
			printf("%d\t",c);
			a=b;
			b=c;
		}
	}
}
```

(ii)

```
#include<stdio.h>

void main()
{
        int N,a,b,c;

        printf("Enter upper limit of the term value:");
        scanf("%d",&N);

        if (N==1) printf("1\t1");
        else
        {
                a=b=1;
                printf("1\t1\t");

                for(;a+b<=N;)
                {
                        c=a+b;
                        printf("%d\t",c);
                        a=b;
                        b=c;
                }
        }
}
```

**PE:** Calculate $e^x$

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

float fact(int n)
{
        int i;
        float p=1;

        for(i=1;i<=n;i++)
                p=p*i;

        return p;
}


void main()
{
        float x,sum=0;
        int i,N,term;

        printf("Feed in the value whose exponent is to be found:");
        scanf("%f",&x);
        printf("Enter the number of terms:");
        scanf("%d",&N);

        term=0;
        for(i=1;i<=N;i++)
        {
                sum=sum+pow(x,term)/fact(term);
                term=term+1;
        }
        printf("e^%f=%f\nReal Answer=%f\nError=%f",x,sum,exp(x),fabs(sum-exp(x)));
}
```

## LOOP PRACTICE QUESTION

1. Calculate a raised to b without pow, assume a & b are integers.
2. Convert any given number to ROMAN

| Decimal | ROMAN |
|---|---|
| 1 | I |
| 5 | V |
| 10 | X |
| 50 | L |
| 100 | C |
| 500 | D |
| 1000 | M |

Example Roman equivalent of 1988 is MDCCCCLXXXVIII

3. A positive integer is entered through the keyboard. Write a program to obtain the prime factors of this number. E.g. prime factors of 24 are 2, 2, 2 and 3, whereas prime factors of 35 are 5 and 7.
4. Write a program to obtain the first N terms of the Fibonacci sequence 1,1,2,3,5,8,13,21,34,…….
5. Write a program to print the binary equivalent of a number.
6. Find the sum of the series
   $\sin(x) = x - (x^3/3!) + (x^5/5!) - (x^7/7!) + \ldots\ldots$ N terms
7. Find the sum of the following series:
   $\frac{1}{1!} + \frac{2}{2!} + \frac{3}{3!} + \ldots\ldots$ N terms
8. Write a program to generate all combinations of 1, 2 and 3
9. According to a study, the approximate level of intelligence of a person can be calculated using the following formula:
   $i = 2 + (y + 0.5x)$
   Write a program that produces a table os values of i, y, x where y varies from 1 to 6, and, for each value of y, x varies from 5.5 to 12.5 in steps of 0.5

---

10. Suppose you place a given sum of money, A, into a saving account at the beginning of each year for n years. If the account earns interest at the rate of i% anually, then the amount of money that has accumulated after n years, F is given by
    $F=A[(1+i/100) + (1+i/100)^2 + (1+i/100)^3 + \ldots\ldots + (1+i/100)^n]$
    Write an interactive program to determine the following:
    i. How much money gets accumulated after n years
    ii. How much money needs to be deposited at the beginning of the five year to get an amount $F_{amt}$.
11. Write a program which finds four digit perfect squares, where the number represented by the first two digits and last two digits are also perfect squares.
    Example: $1681 = 41^2$, $16=4^2$, $81=9^2$
12. Write a program that reads a number with a decimal point & it prints out the number of digits to the left & right of the decimal point.
13. Write a program to find the sum of prime numbers between the limits specified by the user.
14. Read numbers from a user till he enters a negative value. Find the number of values entered,

the highest & the least

15. Mohan has invested Rs.1000/- at 10% simple interest, wheras Vidhya has invested Rs.750 at 7.5% interested compounded annually. Write a program to determine the number of years it will take for the value of Vidhya's investment to exceed that of Mohan's.
16. Write a program to convert a binary number to decimal number system.
17. Print the Fibonacci series of N terms
18. Print the Fibonacci series upto N
19. Read a number from the user & check if it is a Fibonacci number or not.
20. Write a program to generate every third positive integer from 2 to 100 & find the sum of the generated integers which are divisible by 5.
21. An object starting at rest and travels a diatance of 'S' given by $S=at^2/2$. 'a' changes by 2 $m/s^2$ in the range of $1 \le t \le 10$ and decrements by 1$m/s^2$ for next 5sec. Write a programs that prints a table of time, acceleration and distance, where time is to be taken in intervals of 1s.
22. Write a program that checks if two numbers are same upto 2 decimal place.
23. Write a program that prints out all the factors of a number e.g. for 150 the output should show
    2
    3
    5
    5
24. Write a program to compute the distance S fallen by an object in free fall. The formula is $S=So+Vo \cdot t+1/2.at^2$
    Make a table of S for t=1,5,10,15,……100.
25. Write a program that, give a input number, prints its square , cube, and fourth power without doing any unnecessary calculations.
26. A famous conjecture holds that all positive integers converge to 1 (one) when treated in the following fashion.
    i. If the number is odd, it is multiplied by 3 and 1 is added.
    ii. If the number is even, it is divided by 2
    iii. Continuously apply the above operations to the intermediate results until the number converges to 1.

    Write a program to read an integer from the keyboard & implement the above mentioned algorithm and display all the immediate values until the number converges to 1. Also count the number of iterations required for convergence.

Navlakhi®

# Chapter 4a

# Numeric 1 D Array Programming

**When to use Array**

If a large number of memory locations (variables) are needed & all are of the same type then arrays can be used.

e.g. Say we want to read 100 integers. Now we need 100 memory locations

int a,b,c,d,e,…….;

scanf("%d%d%d…….",&a,&b,&c,…..);

Above method is not NOT PRACTICAL

```
int x[100];
for(i=0;i<100;i=i+1)
{
        scanf("%d",&x[i]);
}
```

NOTE: array size is 100. All are integers. First variable is called x[0] & the last is called x[99].

**Reading elements into an array**

Read into x[0], then read into x[1], then x[2] and so on till x[99]. Repetitive means loops….

```
int x[100],i;
printf("Feed in 100 numbers ");
for(i=0;i<100;i=i+1)
{
        scanf("%d",&x[i]);
}
```

**Printing elements of an array**

printf x[0], then x[1], then x[2]….. Till x[99]

AGAIN LOOPs…….

.....

```
printf("Feed in 100 numbers are\n");
for(i=0;i<100;i=i+1)
{
        printf("%d\t",x[i]);
}
```

**PE76:**
Write a program to read 100 numbers and find the sum and average.

```
#include<stdio.h>
void main()
{
  int x[100],i,sum=0;
  float avg;

 for(i=0;i<100;i++)
 {
  printf("Feed in a number ");
  scanf("%d",&x[i]);
 }

 for(i=0;i<100;i++)
 {
        sum=sum+x[i];
  }

 avg=sum/100.0;

 printf("Sum=%d\n",sum);
 printf("Average=%f",avg);
}
```

**ALTERNATE METHOD**

```
#include<stdio.h>
void main()
{
 int x,i,sum=0;
 float avg;

 for(i=0;i<100;i++)
 {
  printf("Feed in a number ");
  scanf("%d",&x);
  sum=sum+x;
 }

 avg=sum/100.0;

 printf("Sum=%d\n",sum);
 printf("Average=%f",avg);
}
```

Navlakhi®

**PE77:**
Write a program to read N number and print them. Also display the sum & average

```
#include<stdio.h>
void main()
{
 int x[100],i,N,sum=0;
 float avg;

 printf("feed in number of terms ");
 scanf(“%d”,&N);

 for(i=0;i<N;i++)
 {
  printf("Feed in a number ");
  scanf(“%d”,&x[i]);
 }

 for(i=0;i<N;i++)
 {
  sum=sum+x[i];
 }

 avg=(float)sum/N;

  printf("the numbers are\n");
 for(i=0;i<N;i++)
 {
  printf(%d\n”,x[i]);
 }

 printf("Sum=%d\n",sum);
 printf("Average=%f",avg);
}
```

**PE78:**
Write a program to read N number and print how many are above average and how many are below average and print the two lists.

```
#include <stdio.h>
void main()
{
 int x[100],n,i,aa=0,ba=0,sum=0;
 float avg;
 printf("Feed in the number of elements ");
 scanf(“%d”,&n);
 printf("Feed in the elements ");

 for(i=0;i<n;i++)
 {
  scanf(“%d”,x[i]);
  sum=sum+x[i];
 }
 avg=(float)sum/n;

 for(i=0;i<n;i++)
 {
  if(x[i]>avg)
                ++aa;
  if(x[i]<avg)
                ++ba;
 }

 printf("Number of elements above average=%d and they are as follows\n",aa);
 for(i=0;i<n;i++)
 {
  if(x[i]>avg)
    printf(“%d\n”,x[i]);
 }

 printf("Number of elements below average=%d and they are as follows\n",ba);
 for(i=0;i<n;i++)
 {
  if(x[i]<avg)
    printf(“%d\n”,x[i]);
 }
}
```

PE79:
Write a program to read N marks and prepare a frequency distribution table 0-40, 40-60, 60-80,80-100

```
#include<stdio.h>
void main()
{
 int i,N,x[100],b1=0,b2=0,b3=0,b4=0;
 printf("Feed in the number of terms ");
 scanf(“%d”,&N);
 printf("Feed in the terms ");
 for (i=0;i<N;i++)
 {
  scanf(“%d”,&x[i]);
 }
 for(i=0;i<N;i++)
 {
  if (x[i]>=0 && x[i]<40)
  {    b1=b1+1; }
  else
        if(x[i]<60)
        {    b2=b2+1; }
        else
        if(x[i]<80)
                {    b3=b3+1; }
                  else
        if(x[i]<=100)
                        {    b4=b4+1; }
 }
 printf("0-40\t%d\n40-60\t%d\n60-80\t%d\n80-100\t",b1,b2,b3,b4);
}
```

Navlakhi®

**ALTERNATE METHOD**

```
#include<stdio.h>
void main()
{
 int i,N,x,b1=0,b2=0,b3=0,b4=0;
 printf("Feed in the number of terms ");
 scanf(“%d”,&N);
 printf("Feed in the terms ");
 for (i=0;i<N;i++)
 {
  scanf(“%d”,&x);

  if (x>=0 && x<40)
  {
   b1=b1+1;
  }

else
        if(x<60)
        {    b2=b2+1;}
        else
        if(x<80)
                {    b3=b3+1;}
                  else
        if(x<=100)
                        { b4=b4+1; }
 }
 printf("0-40\t%d\n40-60\t%d\n60-80\t%d\n80-100\t",b1,b2,b3,b4);
}
```

**PE80:**
Write a program to read N number and cycle the elements of the array i.e. 1st move into 2nd , 2nd moves into 3rd,…… last moves into 1st slot.

```
#include <stdio.h>
void main()
{
 int x[100],n,i,t;
 printf("Feed in the number of elements ");
 scanf(“%d”,&n);
 printf("Feed in the elements ");
 for(i=0;i<n;i++)
 {
  scanf(“%d”,&x[i]);
 }
 t=x[n-1]; /*backing up the last */
 for(i=n-2;i>=0;i=i-1)
 {
  x[i+1]=x[i]; /*2nd last into last, 3rd last into 2nd last.....*/
 }
 x[0]=t; /*putting last into 1st slot*/
 printf("The new array is \n");
 for(i=0;i<n;i++)
 {
  printf(“%d\t”,x[i]);
 }
}
```

**PE81:**
Write a program to read N numbers and swap the adjacent elements i.e. swap 1st & 2nd, 3rd and 4th,……

```
#include <stdio.h>
void main()
{
 int x[100],n,i,t;
 printf("Feed in the number of elements ");
 scanf(“%d”,&n);
 printf("Feed in the elements ");
 for(i=0;i<n;i++)
 {
  scanf(“%d”,&x[i]);
 }
/* Note we will interchange some i with i+1 & i goes 0,2,4,6,....N-2*/
 for(i=0;i<=n-2;i=i+2)
 {
  t=x[i];
  x[i]=x[i+1];
  x[i+1]=t;
 }
 printf("The new array is \n");
 for(i=0;i<n;i++)
 {
  printf(“%d\n”,x[i]);
 }
}
```

**PE82:**
Write a program to read N number and find an element in that list using linear or sequential search

```
#include<stdio.h>
void main()
{
 int i,N,x[100],s;
 printf("Feed in the number of elements ");
 scanf(“%d”,&N);
 printf("feed in the elements \n");
 for(i=0;i<N;i++)
 {
  scanf(“%d”,&x[i]);
 }
 printf("feed in the element to find... ");
 scanf(“%d”,&s);

 for(i=0;i<N;i++)
 {
  if(x[i]==s)
            break;
 }
 if (i<N)
 {
  printf("Found at position %d",i+1);
                        //since arrays start at 0 hence +1
 }
 else
 {
  printf("NOT FOUND");
 }
}
```

**PE83:**
Write a program to read N numbers and find an element using binary search

```
#include<stdio.h>
void main()
{
 int i,n,x[100],s,low,high,mid;
 printf("Feed in the number of elements ");
 scanf(“%d”,&n);
 printf("feed in the elements in sorted order\n");
 for(i=0;i<n;i++)
  scanf(“%d”,&x[i]);

 printf("feed in the element to find... ");
 scanf(“%d”,&s);

 low=0;
 high=n-1;

 while(low<=high)
 {
  mid=(low+high)/2;

  if(x[mid]==s)
  {  break;  }

  if(s<x[mid])
  {  high=mid-1;  }

  if(s>x[mid])
  {  low=mid+1;  }
 }

if(low<=high)
 {
  printf(Found at location %d",mid+1);
 }
 else
 {
  printf("NOT FOUND");
 }
}
```

**PE84:**
Write a program to read N number and arrange in ascending order using bubble sort

```
#include <stdio.h>
void main()
{
 int x[100],n,i,pos,cpos,t;
 printf("Feed in the number of elements ");
 scanf("%d",&n);
 printf("Feed in the elements ");
 for(i=0;i<n;i++)
 {
  scanf("%d",&x[i]);
 }
```

```
for(pos=0;pos<=n-2;pos++)
 {
  for(cpos=pos+1;cpos<=n-1;cpos++)
  {
   if(x[pos]>x[cpos])
   {
    t=x[pos];
    x[pos]=x[cpos];
    x[cpos]=t;
   }
  }
 }
```

-OR-

```
for(pos=1;pos<=n-1;pos++)
{
        for(i=0;i<=n-pos-1;i++)
        {
                if(x[i]>x[i+1])
                {
                        t=x[i];
                        x[i]=x[i+1];
                        x[i+1]=t;
                }
        }

}
```

```
 printf("The sorted list is\n");
 for(i=0;i<n;i++)
    printf("%d\n",x[i]);

}
```

**PE85:**
Write a program to read N number and arrange in descending order using bubble sort

```
#include <stdio.h>
void main()
{
 int x[100],n,i,pos,cpos,t;
 printf("Feed in the number of elements ");
 scanf("%d",&n);
 printf("Feed in the elements ");
 for(i=0;i<n;i++)
 {
  scanf("%d",&x[i]);
 }
```

```
for(pos=0;pos<=n-2;pos++)
 {
  for(cpos=pos+1;cpos<=n-1;cpos++)
  {
   if(x[pos]<x[cpos])
   {
    t=x[pos];
    x[pos]=x[cpos];
    x[cpos]=t;
   }
  }
 }
```

-OR-

```
for(pos=1;pos<=n-1;pos++)
{
        for(i=0;i<=n-pos-1;i++)
        {
                if(x[i]<x[i+1])
                {
                        t=x[i];
                        x[i]=x[i+1];
                        x[i+1]=t;
                }
        }

}
```

```
 printf("The sorted list is \n");
 for(i=0;i<n;i++)
 {
        printf("%d\n",x[i]);
 }
}
```

**PE86:**
Write a program to read N number and arrange in ascending order using selection sort

```
#include <stdio.h>
void main()
{
 int x[100],n,i,pos,large,j;
 printf("Feed in the number of elements ");
 scanf("%d",&n);
 printf("Feed in the elements ");
 for(i=0;i<n;i++)
 {
  scanf("%d",&x[i]);
 }

 for(i=n-1;i>=0;i=i-1) //loop for position of large
 {
  large=x[0];
  pos=0;                    //assumption

  for(j=1;j<=i;j++)
  {
   if(x[j]>large) //found new large
   {
    large=x[j];
    pos=j;
   }
  }

  x[pos]=x[i];
  x[i]=large;
 }

 printf("The sorted list is \n");
 for(i=0;i<n;i++)
 {
  printf("%d\n",x[i]);
 }
}
```

**PE87:**

Write a program to read N number and arrange in descending order using selection sort

```
#include <stdio.h>
void main()
{
 int x[100],n,i,pos,least,j;
 printf("Feed in the number of elements ");
 scanf("%d",&n);
 printf("Feed in the elements ");
 for(i=0;i<n;i++)
 {
  scanf("%d",&x[i]);
 }

 for(i=n-1;i>=0;i=i-1) //loop for position of least
 {
  least=x[0];
  pos=0;                    //assumption

  for(j=1;j<=i;j++)
  {
   if(x[j]<least) //found new least
   {
    least=x[j];
    pos=j;
   }
  }

  x[pos]=x[i];
  x[i]=least;
 }

 printf("The sorted list is \n");
 for(i=0;i<n;i++)
 {
  printf("%d\n",x[i]);
 }
}
```

**PE88:**
Write a program to read N co-ordinates (x1,y1), (x2,y2),…… (xn,yn) and then calculates the length of the shortest line between any two coordinates and displays the two coordinates & the shortest length between them.

```
#include <stdio.h>
#include <math.h>
void main()
{  int minx_start,miny_start, minx_end, miny_end,x[100],y[100],i,j,n;
  float minl;
  printf("Feed in n, the number of co-ordinates ");
  scanf(“%d”,&n);
  printf("feed in the co-ordinates \n");
  for(i=0;i<n;i++)
  {
   printf("x=");
   scanf(“%d”,&x[i]);
   printf("y=");
   scanf(“%d”,&y[i]);
  }

  for(i=0;i<=n-2;i++)
  {
   for(j=i+1;j<=n-1;j++)
   {
    if(i==0 && j==1)
    {
     minl=sqrt((y[1]-y[0])*(y[1]-y[0])+(x[1]-x[0])*(x[1]-x[0]));
     minx_start=x[0];
     miny_start=y[0];
     minx_end=x[1];
     miny_end=y[1];
    }
    else
    {
     if (sqrt((y[i]-y[j])*(y[i]-y[j])+(x[i]-x[j])*(x[i]-x[j]))<minl)
     {
      minl=sqrt((y[i]-y[j])*(y[i]-y[j])-(x[i]-x[j])*(x[i]-x[j]));
      minx_start=x[i];
      miny_start=y[i];
      minx_end=x[j];
      miny_end=y[j];
     }
```

```
    }
   }
  }
  printf("Minimum length=%f and between (%d,%d) and (%d,%d)",minl,minx_start,
                                                    miny_start,minx_end,miny_end);
}
```


# Chapter 4b

# Numeric 2D Array Programming

**When to use 2D – Array**

If data is to be organized in a tabular form then data can be stored in a 2D array.
e.g. Matrices, weekly weather of 5 cities

**Reading elements into a 2D Array**

```
int x[3][3];
for(r=0;r<3;r=r+1)
{       for(c=0;c<3;c=c+1)
        {
                scanf("%d",&x[r][c]);
        }
}
```

Creating and reading a 3X3 matrix

```
int x[5][7];
for(r=0;r<5;r=r+1)
{       for(c=0;c<7;c=c+1)
        {
                scanf("%d",&x[r][c]);
        }
}
```

7 day weather of 5 cities

Note that the first element of a 2D array is x[0][0] & last will be x[9][9] (Assuming it's a 10X10 array).
To read elements into a 2D array you need 2 loops, one for the rows and the other for the columns of each row. Hence, we fill the matrix (2D array) row-wise.

**Printing elements of an array**

printf ROW 0 i.e. x[0][0], then x[0][1]……, then printf ROW 1 i.e. x[1][0], then x[1][1]….. And so on
AGAIN LOOPs…….

```
printf("The Matrix is\n");
for(r=0;r<10;r=r+1)
{
        for(c=0;c<10;c=c+1)
        {
                printf("%d\t",x[r][c]);
        }
        printf("\n"); /*when row ends do endl*/
}
```

**PE89:**
Write a program to read MXN matrix and print it.

```
#include<stdio.h>
void main()
{
 int m,n,i,j,x[100][100];
 printf("Feed in the number of rows and columns ");
 scanf(“%d%d”,&m,&n);
 printf("Enter the elements \n");
 for(i=0;i<m;i++)
 {
  for(j=0;j<n;j++)
  {
   scanf(“%d”,&x[i][j]);
  }
 }

 printf("The matrix you have entered is \n");

 for(i=0;i<m;i++)
 {
  for(j=0;j<n;j++)
  {
   printf(“%d\t”,x[i][j]);
  }
  printf(“\n”);
 }
}
```

**PE90:**
Write a program to read 2 matrices and add them

```
#include<stdio.h>
void main()
{
 int m,n,i,j,x[100][100],y[100][100],z[100][100];
 printf("Feed in the number of rows and columns ");
 scanf(“%d%d”,&m,&n);
 printf("Enter the elements of matrix1 \n");
 for(i=0;i<m;i++)
 { for(j=0;j<n;j++)
  {
   scanf(“%d”,&x[i][j]);
  }
 }
 printf("Enter the elements of matrix2 \n");
 for(i=0;i<m;i++)
 {
  for(j=0;j<n;j++)
  {
   scanf(“%d”,&y[i][j]);
  }
 }

 for(i=0;i<m;i++)
 {
  for(j=0;j<n;j++)
  {
   z[i][j]=x[i][j]+y[i][j];
  }
 }
 printf("The SUM is \n");
 for(i=0;i<m;i++)
 {
  for(j=0;j<n;j++)
  {
   printf(“%d\t”,z[i][j]);
  }
  printf(“\n”);
 }
}
```

**PE91:** Write a program to read 2 matrices and subtract them

```
#include<stdio.h>
void main()
{
 int m,n,i,j,x[100][100],y[100][100],z[100][100];
 printf("Feed in the number of rows and columns ");
 scanf("%d%d",&m,&n);
 printf("Enter the elements of matrix1\n");
 for(i=0;i<m;i++)
 {
  for(j=0;j<n;j++)
  {
   scanf("%d",&x[i][j]);
  }
 }
 printf("Enter the elements of matrix2\n");
 for(i=0;i<m;i++)
 {
  for(j=0;j<n;j++)
  {
   scanf("%d",&y[i][j]);
  }
 }
 for(i=0;i<m;i++)
 {
  for(j=0;j<n;j++)
  {
   z[i][j]=x[i][j]-y[i][j];
  }
 }

 printf("The DIFFERENCE is\n");
 for(i=0;i<m;i++)
 {
  for(j=0;j<n;j++)
  {
   printf("%d\t",z[i][j]);
  }
  printf("\n");
 }
}
```

**PE92:** Write a program to read 2 matrices and multiply them. Check if they are conformal for multiplication before actually multiplying them.

```
#include<stdio.h>
void main()
{
 int m,n,p,q,i,j,k,x[100][100],y[100][100],z[100][100];
 printf("Feed in the number of rows and columns of matrix 1: ");
 scanf(“%d%d”,&m,&n);
 printf("Feed in the number of rows and columns of matrix 2: ");
 scanf(“%d%d”,&p,&q);

if(n==p)
{
   printf("Enter the elements of matrix1:\n");
   for(i=0;i<m;i++)
   {
     for(j=0;j<n;j++)
     {
       scanf(“%d”,&x[i][j]);
     }
   }
 printf("Enter the elements of matrix2:\n");
 for(j=0;j<n;j++)
 {
   for(k=0;k<q;k++)
   {
     scanf(“%d”,&y[j][k]);
   }
 }
 for(i=0;i<m;i++)
 {
   for(k=0;k<q;k++)
   {
     z[i][k]=0;
     for(j=0;j<n;j++)
     {
       z[i][k]=z[i][k]+x[i][j]*y[j][k];
     }
   }
 }
```

```
 printf("The PRODUCT is\n");
 for(i=0;i<m;i++)
 {
  for(k=0;k<q;k++)
  {
   printf("%d\t",z[i][k]);
  }
  printf("\n");
 }
}
else
{
 printf("Matrices NOT CONFORMAL FOR MULTIPLICATION");
}
}
```

**PE93:**Write a program to read a matrix and find its transpose

```
#include<stdio.h>
void main()
{
 int m,n,i,j,x[100][100],y[100][100];
 printf("Feed in the number of rows and columns ");
 scanf("%d%d",&m,&n);
 printf("Enter the elements \n");
 for(i=0;i<m;i++)
 {
  for(j=0;j<n;j++)
  {
   scanf("%d",&x[i][j]);
  }
 }

 for(j=0;j<n;j++)
 {
  for(i=0;i<m;i++)
  {
   y[j][i]=x[i][j];
  }
 }
 printf("The matrix transpose is \n");
 for(j=0;j<n;j++)
 {
  for(i=0;i<m;i++)
  {
   printf("%d\t",y[j][i]);
  }
  printf("\n");
 }
}
```

-OR-

**Transpose without the use of second matrix**

```
for(j=0;j<n;j++)
{
        for(i=0;i<m;i++)
        {
                if(i<j)
                {
                        t=x[i][j];
                        x[i][j]=x[j][i];
                        x[j][i]=t;
                }
        }

}

 printf("The matrix transpose is \n");
 for(j=0;j<n;j++)
 {
  for(i=0;i<m;i++)
  {
   printf("%d\t",x[j][i]);
  }
  printf("\n");
 }
```

**PE94:**Write a program to read a matrix and find the highest and least element and their position

```
#include<stdio.h>
void main()
{
 int m,n,i,j,x[100][100],max,maxr,maxc,min,minr,minc;
 printf("Feed in the number of rows and columns ");
 scanf(“%d%d”,&m,&n);
 printf("Enter the elements\n");
 for(i=0;i<m;i++)
 {
  for(j=0;j<n;j++)
  {
   scanf(“%d”,&x[i][j]);
  }
 }

 for(i=0;i<m;i++)
 {
  for(j=0;j<n;j++)
  {
   if(i==0 && j==0)
   {
    max=x[i][j];
    maxr=i;
    maxc=j;
    min=x[i][j];
    minr=i;
    minc=j;
   }

   if(x[i][j]>max)
   {
    max=x[i][j];
    maxr=i;
    maxc=j;
   }
```

```
    if(x[i][j]<min)
    {
      min=x[i][j];
      minr=i;
      minc=j;
    }
   }
  }
  printf("Maximum=%d and at (%d,%d)\n",max,maxr,maxc);
  printf("Maximum=%d and at (%d,%d)\n",min,minr,minc);
}
```

**PE95:** Write a program to read a matrix and check if it's an identity, upper triangular, lower triangular, or neither.

```
#include<stdio.h>
void main()
{
 int m,n,i,j,x[100][100],success1,success2,success3;
 printf("Feed in the number of rows and columns ");
 scanf("%d%d",&m,&n);
 if(m==n)
 {
        printf("Enter the elements\n");
        for(i=0;i<m;i++)
        {
                for(j=0;j<n;j++)
                        scanf("%d",&x[i][j]);
        }

        success1=1;
        for(i=0;i<m;i++)
        {
                for(j=0;j<n;j++)
                {
                        if((i==j && x[i][j]!=1) ||(i!=j && x[i][j]!=0))
                                success1=0;
                }
        }
        if(success1==0)
                printf("Its not an Identity Matrix...\n");
        else
          printf("It is an identity Matrix...\n");
```

```
        success2=1;
        for(i=0;i<m;i++)
        {
                for(j=0;j<n;j++)
                {
                        if((i>j && x[i][j]!=0))
                                success2=0;
                }
        }
        if(success2==0)
                printf("Its not an Upper Triangular Matrix...\n”);
        else
                printf("It is an Upper Matrix...\n");

        success3=1;
        for(i=0;i<m;i++)
        {
                for(j=0;j<n;j++)
                {
                        if((i<j && x[i][j]!=0))
                                success3=0;
                }
        }
        if(success3==0)
                printf("Its not an Lower Triangular Matrix...\n");
        else
                printf("It is an Lower Triangular Matrix...\n");

        if(success1==0 && success2==0 && success3==0)
                printf("THE MATRIX HAS NONE OF THE THREE PROPERTIES");
 }
 else
 {
  printf("NOT a square MATRIX... ");
 }
}
```

**EX:** Write a program to find the sum of the diagonals elements of a matrix

**PE96:**Write a program to read a 3X3 matrix and check if it is a magic square or not.

```
#include<stdio.h>
void main()
{
 int i,j,x[3][3],asum=0,sum=0,success=1;
 printf("feed in the 3X3 matrix ");
 for(i=0;i<3;i++)
 {
  for(j=0;j<3;j++)
  {
   scanf("%d",&x[i][j]);
  }
 }
 asum=x[0][0]+x[0][1]+x[0][2];

for(i=0;i<3;i++)
{
        for(j=0;j<3;j++)
        {
                pick=x[i][j];
                c=0;
                for(i1=0;i1<3;i1++)
                {
                        for(j1=0;j1<3;j1++)
                        {
                                if(x[i1][j1]==pick)
                                        c++;
                        }
                }
                if(c>1)  success=0;
        }
}
```

```
//Checking for rows
for(i=0;i<3;i++)
{
 sum=0;
 for(j=0;j<3;j++)
 {
  sum=sum+x[i][j];
 }
 if(sum!=asum)
 {
  success=0;
 }
}

//Checking for columns
for(j=0;j<3;j++)
{
 sum=0;
 for(i=0;i<3;i++)
 {
  sum=sum+x[i][j];
 }
 if(sum!=asum)
 {
  success=0;
 }
}

//Checking diagonal 1
sum=0;
for(i=0;i<3;i++)
{
 sum=sum+x[i][i];
}
if(sum!=asum)
{
 success=0;
}
```

```
//Checking diagonal 2
 sum=0;
 for(j=2,i=0;i<3;i++,j--)
 {
  sum=sum+x[i][j];
 }
 if(sum!=asum)
 {
  success=0;
 }


 if(success==0)
 {   printf("Its not a MAGIC SQUARE");  }
 else
 {   printf("Its a MAGIC SQUARE");  }
}
```

**PE97:**Write a program to read the 7 day temperature of 5 cities and print the average temperature of the week on a per city basic and find the hottest city on average.

```
#include<stdio.h>
void main()
{
 int x[5][7],i,j,sum=0,maxpos;
 float avg[5],max;
 for(i=0;i<5;i++)
 {
  printf("Feed in the 7 temp. for city %d\n",i);
  sum=0;
  for(j=0;j<7;j++)
  {
   scanf(“%d”,&x[i][j]);
   sum=sum+x[i][j];
  }
  avg[i]=sum/7.0;
 }

 for(i=0;i<5;i++)
 {
  if (i==0)
  {
   max=avg[0];
   maxpos=0;
  }
  if(avg[i]>max)
  {
   max=avg[i];
   maxpos=i;
  }
 }
 printf("Hottest city is  %d and the avg. temperature is %f",maxpos,max);
}
```

Navlakhi®

# Chapter 4c

# Character Array Programming

Like 1D numeric arrays character arrays will store a list character e.g. name/city/country
e.g. char city[100];
has the capacity to hold the name of 1 city (say Mumbai)
city[0]='M'
city[1]='u'
city[2]='m'
city[3]='b'
city[4]='a'
city[5]='i'
city[6]='\0'

**Significance of '\0' (NULL)**

Each character array has to be atleat 1 more than the maximum allowed data (for NULL) e.g. if we want to enter a 10 letter word, then, the array should be created of size 11 (at least).
e.g. char w[11];

**Input into a character Array**

Unlike a numeric 1D array (which needs a loop to enter data), here there is no need of a loop

```
e.g.    char x[100];
        printf("Feed in a word");
        scanf("%s",x);
```

**String Manipulation Functions**

Note: We can manipulate with each character by running a loop (like 1D numeric arrays).
Some string manipulation functions are

- strlen

→finds the length (number of characters in the character array)
→Example: l = strlen(x);

- strcat

→To join (concatenate) two arrays
→Example: strcat(s1,s2)
→Like saying s1=s1+s2 (BUT THIS IS A WRONG WAY TO WRITE)

- strcpy

→To copy one array into another
→Example: strcpy(s1,s2)
→Like saying s1=s2 (BUT THIS IS A WRONG WAY TO WRITE)

❖ strcmp

→To Compare one array into another. This returns the difference in the ASCII values of the 1st mismatching character. (NOTE: It's a case sensitive comparison)

→Example: strcmp(s1,s2)

→Answer can be <0, means s1 is alphabetically before s2 (e.g. s1 was 'cat' and s2 was 'dog')

>0 , means s1 is alphabetically after s2

=0 , means s1 and s2 are both same

❖ strcmpi / stricmp

→To Compare one array into another. This returns the difference in the ASCII values of the 1st mismatching character. (NOTE: It's a case insensitive comparison)

→Example: strcmp(s1,s2)

→Answer can be <0, means s1 is alphabetically before s2 (e.g. s1 was 'cat' and s2 was 'dog')

>0 , means s1 is alphabetically after s2

=0 , means s1 and s2 are both same

❖ strchr

→To find the first occurrence of a particular character

→Example: strcpy(m,strchr(s1, 'h'))

→Say if s1 was "Abhishek", then m will be "hishek"

NOTE: m has to be char array (more about it later)

❖ strstr

→To find the first occurrence of a particular string (pattern) in a string

→Example: strcpy(m,strstr(s1, s2))

→Say if s1 was "Abhishek" and s2 was "hi", then m will be "hishek"

NOTE: m has to be a char array (more about it later)

❖ strupr

→To convert the whole string into upper case

→Example: strupr(s1);

→Say if s1 was "Abhishek", then s1 will now be "ABHISHEK"

NOTE: NOT supported by all compilers.

❖ strlwr

→To convert the whole string into lower case

→Example: strlwr(s1);

→Say if s1 was "AbhiSHek", then s1 will now be "abhishek"

NOTE: NOT supported by all compilers.

❖ MANY MORE FUNCTIONS ARE AVAILABLE

❖ NOTE: ALL the above String manipulation functions need a new header file called
**#include <string.h>**

**Character Manipulation Functions**

❖ toupper

→Convert the character argument to uppercase
→Example: c2=toupper(s1[i])
→NOTE: The argument has to be a char (or a single element of the array) NOT THE COMPLETE ARRAY.
LHS will be a char variable
char c2;

❖ tolower

→Convert the character argument to lowercase
→Example: c2=tolower(s1[i])
→NOTE: The argument has to be a char (or a single element of the array) NOT THE COMPLETE ARRAY.
LHS will be a char variable
char c2;

❖ is_____ are functions used to check if the character possesses a particular property
Return Value: non-zero value on success. Success is defined as follows:

■ isalpha: c is a letter (A to Z or a to z)
■ isascii: the low order byte of c is in the range 0 to 127
■ isdigit: c is a digit (0 to 9)
■ islower: c is a lowercase letter (a to z)
■ isspace: c is a space, tab, carriage return, new line, vertical tab, or form feed
■ isupper: c is an uppercase letter (A to Z)
■ isxdigit: c is a hexadecimal digit (0 to 9, A to F, a to f)

→ Example: int l = isdigit(s[i]);

NOTE: ALL the above character manipulation functions need a new header file called
**#include <ctype.h>**

**PE98:**
Write a program to read 2 words and check if they are same

```
#include<stdio.h>
#include<string.h>
void main()
{
  char x[100],y[100];
  printf("Feed in 2 words ");
  scanf("%s%s",x,y);
  if(strcmpi(x,y)==0)
  {
    printf("Both words are the same");
  }
  else
  {
    printf("Both words are different");
  }
}
```

**PE99:**Write a program to read a word and check if it is a palindrome or not

```
#include<stdio.h>
#include<string.h>
void main()
{
  char x[100],y[100];
  int i,j,l;

  printf("feed in a word ");
  scanf(“%s”,x);

  l=strlen(x);

  for(i=0,j=l-1;i<=l-1;i++,j--)
  {
    y[j]=x[i];
  }
y[l]=’\0’;

  if(strcmpi(x,y)==0)
  {
    printf("The word is a palindrome ");
  }
  else
  {
    printf("NOT a palindrome");
  }
}
```

**PE100:**
Write a program to read a word and replace every occurrence ‘a’ by ‘j’.

```
#include<stdio.h>
#include<string.h>
void main()
{
  char x[100],i,l;
  printf("Feed in a word ");
  scanf(“%s”,x);
  l=strlen(x);
  for(i=0;i<=l-1;i++)
  {
    if(x[i]=='a')
    {
      x[i]='j';
    }
  }
  printf("New word is %s",x);
}
```

Navlakhi®

**Reading a sentence**

The problem with %s, the way we used it in the precious example, is that it cannot reads words (can read only one word).

Solution:

→ scanf("%[^\n]",x);

→ gets(x);

**PE101:**
Write a program to read any sentence & find its length, number of vowels, number of consonants, number of white spaces, number of capital letters, number of small letters, number of digits. Then display the sentence in upper case and in lower case

```
#include<stdio.h>
#include<string.h>
#include<ctype.h>
void main()
{
 char x[100];
 int i,l,vc=0,cc=0,wspc=0,capc=0,smallc=0,nodc=0;
 printf("feed in a sentence ");
 scanf("%[^\n]",x);              OR   gets(x);
 l=strlen(x);
 for(i=0;i<=l-1;i++)
 {
        if(toupper(x[i])=='A'||toupper(x[i])=='E'||toupper(x[i])=='I'||toupper(x[i])=='O'||
                                                                toupper(x[i])=='U')
        {       vc++;           }
        else if(isalpha(x[i]))
        {       cc++;           }

        if (isspace(x[i]))
        {       wspc++;         }

        if (x[i]>='A' && x[i]<='Z')
        {       capc++;         }

        if(x[i]>='a' && x[i]<='z')
        {       smallc++;       }

        if(isdigit(x[i]))
        {       nodc++;         }
 }
 printf("Number of vowels=%d\nNumber of consonents=%d\nNumber of white
spaces=%d\nNumber of capital letters=%dNumber of small case letters=%dNumber of
digits=%d\n",vc,cc,wspc,capc,smallc,nodc);
strupr(x);
printf( "Upper case=%s\n",x);
strlwr(x);
printf("Lower case=%s",x);
}
```

**PE102:**
Write a program to read the roll number, name and 3 subject marks of a student and calculate the percentage and class.

```
#include<stdio.h>
void main()
{
 int r,m1,m2,m3;
 float avg;
 char name[100];
 printf("feed in the roll number ");
 scanf(“%d”,&r);
 printf("Feed in the name ");
 scanf(“%s”,name); //Just 1 word name
 printf("Feed in the 3 subject marks ");
 scanf(“%d%d%d”,&m1,&m2,&m3);

 avg=(m1+m2+m3)/3.0;
 printf("Name=%s\nPercentage=%.2f\n",name,avg);

 if(avg>=60)
 {   printf(“Grade=A”);  }
 else  if(avg>=50)
  {     printf(“Grade=B”);    }
  else  if(avg>=40)
    {       printf(“Grade=C”);      }
    else
    {       printf(“Grade=F”);      }
}
```

**ALTERNATE METHOD**

```
#include<stdio.h>
void main()
{
 int r,m1,m2,m3;
 float avg;
 char name[100];
 printf("feed in the roll number ");
 scanf(“%d”,&r);
 printf("Feed in the name ");
 gets(name);   -- OR –  scanf(“%[^\n]”,name);
 gets(name);
 printf("Feed in the 3 subject marks ");
 scanf(“%d%d%d”,&m1,&m2,&m3);
 avg=(m1+m2+m3)/3.0;

 printf("Name=%s\nPercentage=%.2f\n",name,avg);

 if(avg>=60)
 {   printf(“Grade=A”);  }
 else  if(avg>=50)
  {     printf(“Grade=B”);    }
  else  if(avg>=40)
    {       printf(“Grade=C”);      }
    else
    {       printf(“Grade=F”);      }
}
```

Navlakhi®

# Chapter 5

# Structures Programming

A collection of known data types

e.g.

```
struct student
{
        char name[40];
        int s1,s2,s3;
        float p;
};
```

**Allocating structures**

```
void main()
{
        struct student x;
        ……..
}
```

Now x contains name,s1,s2,s3,p

Each member can be accessed using the . (dot) operator e.g. x.name, x.s1, x.s2, x.s3, x.p

**Array of structures**

Array of structures

```
void main()
{
        struct student x[100];
        ……
}
```

100 instances of student x[0] to x[99] each containing 5 members (name,s1,s2,s3,p).

**PE104:**
Write a program to read the name, 3 subject marks, of N students. Calculate the percentage & display the entire list in
(a) Ascending order of name
(b) Descending order of percentage
Use structures and a menu driven approach.

```
#include<stdio.h>
#include<string.h>
struct student
{
 char name[40];
 int m1,m2,m3;
 float avg;
};

void main()
{
 struct student x[100],t;
 int N,i,choice,pos,cpos;
 printf("feed in the number of students ");
 scanf(“%d”,&N);

 for(i=0;i<=N-1;i++)
 {
  printf("Feed in name & 3 subject marks ");
  scanf(“%s%d%d%d”,x[i].name,&x[i].m1,&x[i].m2,&x[i].m3);
  x[i].avg=(x[i].m1+x[i].m2+x[i].m3)/3.0;
 }

 do
 {
  printf("1. List by name\n2. List by percentage\n3.Exit\nenter choice ");
  scanf(“%d”,&choice);
```

```
  if(choice==1)
  {
   for(pos=0;pos<=N-2;pos++)
   {
    for(cpos=pos+1;cpos<=N-1;cpos++)
    {
     if(strcmpi(x[pos].name,x[cpos].name)>0)
     {
      t=x[pos];
      x[pos]=x[cpos];
      x[cpos]=t;
     }
    }
   }

printf("%40s%5s%5s%5s%15s\n","Name","Sub1","Sub2","Sub3","Percentage");
printf("-------------------------------------------------------------------------------------\n");


   for(i=0;i<=N-1;i++)
   {
        printf("%-40s%5d%5d%5d%15.2f\n",x[i].name,x[i].m1,x[i].m2,x[i].m3,x[i].avg);
   }
  }
```

```
  if(choice==2)
  {
   for(pos=0;pos<=N-2;pos++)
   {
    for(cpos=pos+1;cpos<=N-1;cpos++)
    {
     if(x[pos].avg<x[cpos].avg) //doing descending
     {
      t=x[pos];
      x[pos]=x[cpos];
      x[cpos]=t;
     }
    }
   }


printf("%40s%5s%5s%5s%15s\n","Name","Sub1","Sub2","Sub3","Percentage");
printf("---------------------------------------------------------------------------------------\n");


   for(i=0;i<=N-1;i++)
   {
        printf("%-40s%5d%5d%5d%15.2f\n",x[i].name,x[i].m1,x[i].m2,x[i].m3,x[i].avg);
   }
  }
 }while(choice!=3);
}
```

**Explain Nested Structures**

```
struct name
{
  char fname[20];
  char mname[20];
  char sname[20];
};

struct student
{
  struct name n;
  int rno;
};

void main()
{
        struct student x[100];
        ……

        for(i=0;i<N;i++)
        {

        ……..
          scanf("%s%s%s%d",x[i].n.fname, x[i].n.mname, x[i].n.sname, &x[i].rno);

        ……
        }

        ……

}
```

Navlakhi®


# Chapter 6

# Pointers

**Example 1:**

```
int x;
x=5;
printf("%d\n",x);
printf("%u\n",&x);
```

**Example 2:**

```
int *p;
int x;
p=&x;
x=5;    --OR--   *p=5;

printf("%d",x);
printf("%u",&x);
printf("%d",*p);
printf("%u",p);
printf("%u",&*p);
```

**Explain dynamic memory allocation**

```
#include<alloc.h>
#include<stdio.h>
void main()
{
        int *p;
        p=(int *)malloc(sizeof(int));
        *p=10;
        printf("%d",*p);
        free(p);
}
```

malloc : memory allocation at runtime (default type is char)

free: free the memory created during malloc

Since the memory is created & free at runtime, malloc is called dynamic memory allocation function and free is dynamic memory deallocation function.

**Single dimensional arrays using pointers (OR dynamic 1D arrays OR Array of user defined size):**

**Example:** To find the sum of N-elements using arrays

```
#include<stdio.h>
#include<alloc.h>
void main()
{
        int *p,N,i,sum=0;
        printf("Feed in the number of elements");
        scanf("%d",&N);

        p=(int *)malloc(sizeof(int)*N);

        for(i=0;i<N;i++)
        {
                printf("Feed in an interger ");
                scanf("%d",&p[i]);
                sum=sum+p[i];
        }

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

NOTE: p[i] is also *(p+i)

**Double dimensional array using pointers OR Dynamic 2D arrays**

```
int **p,i;
p=(int **)malloc(sizeof(int)*m);        /*where m is user input number of rows*/
for(i=0;i<m;i++)
        p[i]=(int *)malloc(sizeof(int)*n);        /*where n is user input number of columns*/
```

**NOTE: Total number of rows 'mn'**

**p: pointer to 1st row**
**p+i: pointer to i'th row**
***(p+i): pointer to 1st element of i'th row**
***(p+i)+j: pointer to j'th element of i'th row**
**i.e. p[i][j]**
***(*(p+i)+j): value of element p[i][j]**

**Pointer to Structures:**

```
#include<stdio.h>
#include<alloc.h>
struct student
{
        char name[20];
        int id,s1,s2,s3,t;
        float per;
};

void main()
{
        struct student *p;
        p=(struct student *)malloc(sizeof(struct student));

        printf("Feed in the name,id & 3 subject marks\n");
        scanf("%s%d%d%d%d",p→name, &p→id, &p→s1, &p→s2, &p→s3);

        p→t = p→s1 + p→s2 + p→s3;
        p→per=p→t/3.0;

        printf("Percentage=%.2f",p→per);
}
```

**NOTE:**
p→name can also be written as (*p).name
p→id can also be written as (*p).id  and so on….
Note that the brackets is mandatory as . (dot) has higher precedence than *

**Command Line Arguments**

Command line arguments allows the user to interact (enter input) via dos/UNIX/LINUX command prompt.

```
#include<stdio.h>
void  main(int argc, char** argv)
{       int i;
        printf("Total number of arguments=%d\n",argc);
        for(i=0;i<argc;i++)
                printf("Argument %d=%s\n",i,argv[i]);
}
```

NOTE: Say program name is comeg.exe then we run the program at command prompt as:
comed.exe 1 10 abc

Navlakhi®

Total number of arguments=4
Argument 0=comeg.exe *(---Program name---)*
Argument 1=1
Argument 2=10
Argument 3=abc

**Programming example:** Find larger of 2 numbers using command line arguments.

Navlakhi®


# Chapter 7

# Function Programming

A Function contains 4 components

- Return type
- Function name
- Function arguments
- Function body

**Syntax**

```
Return_type  Name_of_fn(arg_type arg1, arg_type arg2,…….)
{
        Function Body
}
```

Return_type signifies the type of data the function will send back to its caller (int, float, double, etc). In case it doesn't then return_type is **void**.

**Arguments**

Arguments, is what the function is expecting as input (pre-requisite) before it can begin the logic. A good approach will be to think of which scanf is needed and that can be assumed as arguments.

The main program will read the data from the user & forward it to the function as arguments. That's why arguments is what a function expects as available (or what a function wants is the arguments).

**Can Arguments be avoided**

One way to avoid arguments (no considered a good practice) is to use global variables. When the main & function both want to operate on the same data then make it global, since all functions including main can use global data.

**Types of Arguments**

Arguments can be of basic types like: int, float, double, char, unsigned int, etc.

The can be of derived type too, like: structures, Arrays

**Call be Value**

Whenever the caller (e.g. main) sends arguments called actual parameters(other than arrays) to a function (i.e. callee), and if the function changes the values of these arguments (called formal parameters) in the body of the function, then these changes will NOT BE REFLECTED IN THE MAIN arguments. Hence, the function arguments are copies of the main arguments (NOT THE ORIGINALS).

```
#include <stdio.h>
void f(int x)
{
        x=x+1 ;
}
void main()
{       int x=3;
        f(x);
        printf("%d",x);
}
```

Output:

3

Formal Parameters can be of different name than the actual Parameter

```
void f(int y)
{
        y=y+1 ;
}
void main()
{       int x=3;
        f(x);
        printf("%d",x);
}
```

Output:

3

**Call be Address/Reference**

When the function arguments (formal) need to represent the original(actual), hence changes in formal should reflect in the actual then call by value cannot be used. Here, we have to use call by reference.

Here formal refers to the actual parameters & NOT A COPY OF THE ACTUAL parameters.

```
#include <stdio.h>
void f(int* x)
{
        *x=*x+1 ;
}
void main()
{       int x=3;
        f(&x);
        printf("%d",x);
}
```

Output:

4

**ALTERNATE METHOD**

```
#include <stdio.h>
void f(int* x);  ← Function prototype
void main()
{       int x=3;
        f(&x);
        printf("%d",x);
}
void f(int* x)
{
        *x=*x+1 ;
}
```

**ALTERNATE METHOD**

```
#include <stdio.h>
void f(int*);  ← Function prototype
void main()
{       int x=3;
        f(&x);
        printf("%d",x);
}
void f(int* x)
{
        *x=*x+1 ;
}
```

If the callee functions are written below the calling function, then we need to use function prototype, i.e. the function header needs to be copied anywhere within the calling function (above the function call) or globally above the calling function.
Here the name of the formal parameter can be different or absent

**PE105:**
Write a program to interchange 2 numbers using functions

```
#include<stdio.h>
void interchange(int* x, int* y)
{
  int t;
  t=*x;
  *x=*y;
  *y=t;
}

void main()
{
  int m,n;
  printf("feed in 2 integers ");
  scanf("%d%d",&m,&n);
  interchange(&m,&n);
  printf("Now 1st no.=%d and second=%d",m,n);
}
```

**PE106:**
Write a program that does a temperature conversion $^{o}$C to F and vice versa. The function should return the value.

```
#include<stdio.h>
float CtoF(float C)
{
  return (C/5*9+32);
}

float FtoC(float F)
{
  return ((F-32)/9*5);
}

void main()
{
  float t,C,F;
  printf("enter temp in C ");
  scanf(“%f”,&C);
  t=CtoF(C);
  printf("Temp in farenheit is %f\n",t);
  printf("enter temp in Farenheit ");
  scanf(“%f”,&F);
  t=FtoC(F);
  printf("Temp is Celcius=%f",t);
}
```

**PE107:** Write a program to create a function sort which sorts the student data based on percentage. Ask the users to enter the data. Use structures.

```
#include<stdio.h>
struct student
{
 char name[40];
 int m1,m2,m3;
 float avg;
};

void sort(struct student X[],int n)
{
 int pos,cpos;
 struct student t;
 for(pos=0;pos<=n-2;pos++)
 {
  for(cpos=pos+1;cpos<=n-1;cpos++)
  {
   if(X[pos].avg<X[cpos].avg)
   {
    t=X[pos];      X[pos]=X[cpos];      X[cpos]=t;
   }
  }
 }
}

void main( )
{  struct student x[100];
 int n,i;
 printf("feed in number of students ");
 scanf("%d",&n);
 for(i=0;i<n;i++)
 { printf(<"feed in the name & 3 subject marks ");
  scanf("%s%d%d%d",x[i].name,&x[i].m1,&x[i].m2,&x[i].m3);
  x[i].avg=(x[i].m1+x[i].m2+x[i].m3)/3.0;
 }
 sort(x,n);
 printf("Sorted list is\n");
 for(i=0;i<n;i++)
 {   printf("%s\t%f\n",x[i].name,x[i].avg);  }
}
```

**PE108:** Write a function to add, subtract, multiply, transpose, print a matrix. Write a menu drive program for the same.

```
#include<stdio.h>
void add(int x[ ][100], int y[ ][100], int z[ ][100], int m, int n)
{
 int i,j;
 for(i=0;i<m;i++)
 {
  for(j=0;j<n;j++)
  {
   z[i][j]=x[i][j]+y[i][j];
  }
 }
}

void subtract(int x[][100], int y[][100], int z[][100], int m, int n)
{  int i,j;
 for(i=0;i<m;i++)
 {
  for(j=0;j<n;j++)
  {
   z[i][j]=x[i][j]-y[i][j];
  }
 }
}

void multiply(int x[][100],int y[][100], int z[][100], int m, int n, int q)
{
 int i,j,k;
 for(i=0;i<m;i++)
 {
  for(k=0;k<q;k++)
  {
   z[i][k]=0;
   for(j=0;j<n;j++)
   {
    z[i][k]+=x[i][j]*y[j][k];
   }
  }
 }
}
```

```
void transpose(int x[][100], int y[][100], int m, int n)
{
 int i,j;
 for(j=0;j<n;j++)
 {
  for(i=0;i<m;i++)
  {
   y[j][i]=x[i][j];
  }
 }
}

void print(int x[][100],int m,int n)
{
 int i,j;
 for(i=0;i<m;i++)
 {
  for(j=0;j<n;j++)
  {
   printf("%d\t",x[i][j]);
  }
  printf("\n");
 }
}

void main( )
{
 int m,n,p,q,x[100][100],y[100][100],z[100][100],i,j,choice;
 printf("feed in the number of rows & columns of mat 1 ");
 scanf("%d%d",&m,&n);
 printf("feed in matrix 1\n");
 for(i=0;i<m;i++)
 {
  for(j=0;j<n;j++)
  {
   scanf("%d",&x[i][j]);
  }
 }
 printf("feed in the number of rows & columns of mat 2 ");
 scanf("%d%d",&p,&q);
```

```
printf("feed in matrix 2\n");
for(i=0;i<p;i++)
{
 for(j=0;j<q;j++)
 {
  scanf(“%d”,&y[i][j]);
 }
}

do
{
 printf("---------------------------------\n");
 printf("Menu\n1. Add\n2. Subtract\n3. Multiply\n4. Transpose\n5. Print\n6.Exit\n”);

 printf("Enter choice ");
 scanf(“%d”,&choice);

 if(choice==1)
 {
  if(m!=p || n!=q)
  {
   printf("Matrices not conformal for addition\n");
  }
  else
  {
   add(x,y,z,m,n);
   print(z,m,n);
  }
 }
```

```
  if(choice==2)
  {
    if(m!=p || n!=q)
    {
      printf("Matrices not conformal for subt.\n");
    }
    else
    {
      subtract(x,y,z,m,n);
      print(z,m,n);
    }
  }


  if(choice==3)
  { if(n!=p)
    {
      printf("Matrices not conformal for multiplication\n");
    }
    else
    {
      multiply(x,y,z,m,n,q);
      print(z,m,q);
    }
  }

  if(choice==4)
  {
    transpose(x,z,m,n);
    print(z,n,m);
    transpose(y,z,p,q);
    print(z,q,p);
  }

 if(choice==5)
  {
    print(x,m,n);
    print(y,p,q);
  }
 }while (choice!=6);
}
```

## STRINGS MANIPULATION FUNCTIONS

```
int strlen(char *s)
{
        int i=0;
        while(s[i]!=NULL)
                i++;
        return i;
}
```

```
void strcpy(char *s1, char *s2)
{
        int i=0;
        while(s2[i]!=NULL)
        s1[i++]=s2[i];
        s1[i]=NULL;
}
```

```
void strcat(char *s1, char *s2)
{
        char *t;
        t=new char[strlen(s1)+1];
        strcpy(t,s1);
        strcpy(s1,t);
        int j=0;
        int i=strlen(s1);
        while(s2[j]!=NULL)
                s1[i++]=s2[j++];
        s1[i]=NULL;
}
```

```
int strcmp(char *s1,char *s2)
{
        int i=0;
        while(s1[i]!=NULL&&s2[i]!=NULL)
        {
                if(s1[i]!=s2[i])
                        return s1[i]-s2[i];
                i++;
        }
        if(s1[i]!=NULL)
                return s1[i];
        else if(s2[i]!=NULL)
                return -s2[i];
        else return 0;
}
```

**ARRAY PRACTICE SHEET**

# SET I

1. Write a function to find the norm of a matrix. The norm is defined as the square root of the sum of squares of all elements in the matrix.
2. Find the co-relation coefficient of a set of points (x,y), where correlation coefficient is given as

$$r = \frac{\sum xy - \sum x \sum y}{\sqrt{[n\sum x^2 - (\sum x)^2][n\sum y^2 - (\sum y)^2]}}$$

3. A straight line y=ax+b can be fit to a set of points (x,y). Write a program that reads the x & y values & finds a & b are given as

$$a = \overline{y} - b\overline{x} \quad \text{and}$$

$$b = \frac{n\sum yx - \sum x \sum y}{[n\sum x^2 - (\sum x)^2]}$$

4. The **X** and **Y** coordinates of 10 different points are entered through the keyboard. Write a program to find the distance of last point from the first point (sum of distance between consecutive points)
5. Write a program that extracts part of the given string from the specified position. For example, if the sting is "Working with strings is fun", then if from position 4, 4 characters are to be extracted then the program should return string as "king". Moreover, if the position from where the string is to be extracted is given and the number of characters to be extracted is 0 then the program should extract entire string from the specified position
6. Write a program that replaces two or more consecutive blanks in a string by a single blank. For example, if the input is Grim return to the planet of apes!! **the output should be** Grim return to the planet of apes!!
7. Write a program to sort a set of names stored in an array in alphabetical order.
8. Write a program to count the number of occurrences of any two vowels in succession in a line of text.

---

Navlakhi®

# SET 2

9. Write a program to compute the mean, variance and standard deviation of a set of numbers using the following formula

$$\text{Mean} = 1/n \sum_{i=1}^{n} Xi$$

$$\text{Variance} = 1/n \sum_{i=1}^{n} (Xi-Xmean)^2$$

$$\text{Standared Deviation} = \sqrt{\text{Variance}}$$

10. **A minimax or saddle point in a two dimensional array is an element that is the minimum of its row and the maximum of its column, or vice verse. For example , in the following array A[I][J]**

| | | | |
|---|---|---|---|
| 11 | 22 | 33 | 24 |
| 99 | 55 | 66 | 77 |
| 77 | 44 | 88 | 22 |

**The element A[1][3]= 33 is a minimax**

**The element A[2][2]= 55 is a minimax**

**Write a program to identify and display all such saddle points and its location with in the two dimensional array, which is read from the keyboard**

11. Write a program to calculate the frequency of a given letter in a string.
12. Write a program to read a string & generate a frequncy table with the letter in one column & number of times it apeared in the other column.
13. Write a program to find all permutations of a string. (Find all possible combinations of the different characters of the string)
14. Decimal to binary conversion
15. Read a string & capitalize the first letter of every word
    Input: programming is fun
    Output: Programming Is Fun
16. Write a program to read a string & print it in alphabetical order. E.g. if the word if STRING the output should be GINRST
17. Read a 2D matrix and sort each row in an ascending order & print the new matrix
18. Write a program to interchange the words of a sentence
    Input: Insects eats apples
    Output: apples eats Insects

Navlakhi®

# Chapter 8

# Recursion

Recursion means a function that calls itself.
Advantage: Smaller code size
Disadvantage: Slower & high amount of stack memory required.

**Example 1:**

Factorial using recursion
As we all know

$$5! = 5 \text{ X } 4 \text{ X } 3 \text{ X } 2 \text{ X } 1 = 5 \text{ X } 4!$$

Similarly

$$4! = 4 \text{ X } 3!$$
$$3! = 3 \text{ X } 2!$$
$$2! = 2 \text{ X } 1!$$
$$1! = 1 \text{ X } 0!$$

And

$$0! = 1$$

Thus factorial(n)= n X factorial(n – 1)
This completes the first step (i.e. designing your iterative formula). Now we have to design the stopping criteria is when will the above formula not be applicable. The answer is when n=0. This completes the second step.

Now we have to design the function. The function will have n as the pre-requirement & the stopping criteria i.e. n=0 will have to be tested first & then the recursive condition. Always the criteria which does not satisfy the recursive formula is put first…. Note in factorial the answer will be a long int & not a int (since int is only upto 32767).

```
#include<stdio.h>
long factorial(int n)
{       if(n==0)
        {
                return 1;
        }
        else
        {
                return n*factorial(n-1);
        }
}
void main( )
{       int x;
        long p;

        printf( "Enter the number ");
        scanf("%d",&x);

        p=factorial(x);
        printf( "Factorial of &d=%ld\n", x , p);
}
```

NOTE: In the factorial function we have put the non-recursive criteria first & then the recursive criteria.

**Example 2:**

Let's try GCD using recursion….

If you recollect we had done GCD/HCF by taking a reverse loop & the moment we found the first number dividing both we stopped & termed it as the GCD.

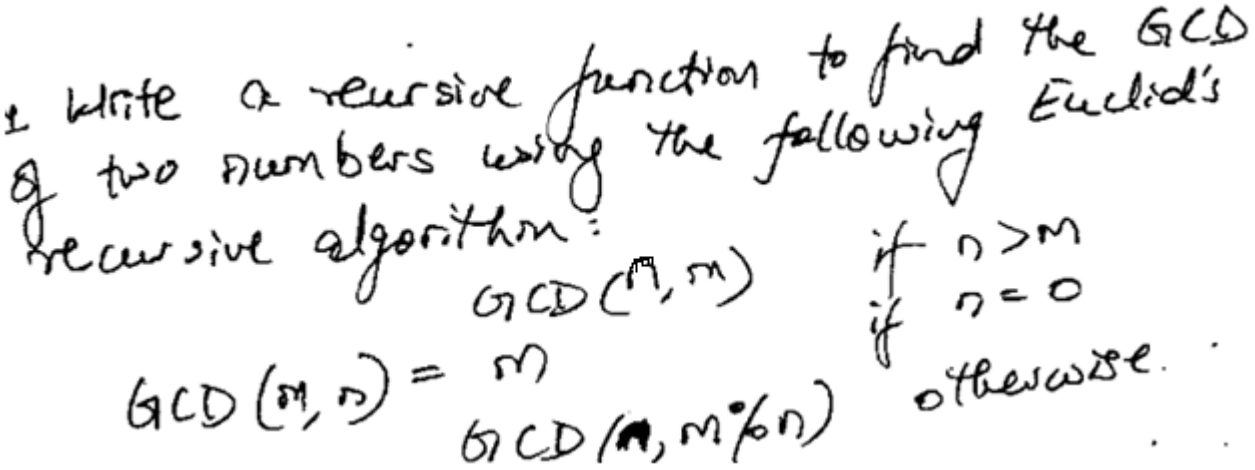
Hence the repetitive criteria is dividing by 'i' (from m to 1) in steps of -1 till we find an 'i' dividing both.

```
#include<stdio.h>
int GCD(int m, int n, int i)
{
        if(m%i==0 && n%i==0)
        {
                return i;
        }
        else
        {
                return GCD(m,n,i-1);
        }
}
void main( )
{
        int m,n,ans;

        printf( "Enter the 2 numbers ");
        scanf("%d%d",&m,&n);

        ans=GCD(m,n,m);

        printf( "GCD=%d",ans);
}
```

NOTE: the function needs m,n & intial value of 'i' which can be m or n.

**Example 3:**

In some question we don't have to think of the login…. As can be seen in the question given below, there is on non-recursive condition, & that has to be first & the remaining below it. Don't care to think if the logic works or not….




```
#include<stdio.h>

int GCD(int m, int n)
{
        if(n==0)
        {
                return m;
        }
        else if(n>m)
        {
                return GCD(n,m);
        }
        else
        {
                return GCD(n,m%n);
        }
}
void main( )
{
        int m,n,ans;

        printf( "enter 2 numbers ");
        scanf("%d%d",&m,&n);

        ans=GCD(m,n);

        printf( "GCD=%d",ans);
}
```

**Example 4:**

Recurive fibonacci…. (it is a series 1, 1, 2, 3, 5, 8, 13,……., basically next term is sum of previous 2 terms).

Here we have to repeatedly do c=a+b (initial value of a=1 & b=1), then we shift by 1 i.e. a becomes current b & b becomes current c, so we get a new c using the same formula c=a+b. We stop when we have printed the necessary number of terms (say n).

```
#include<stdio.h>
void fibo(int a, int b, int done, int n)
{
        if(done==n)
        {
                return;
        }
        else
        {
                int c=a+b;
                printf("%d\t",c);
                a=b;
                b=c;
                fibo(a,b,done+1,n);
        }
}
void main( )
{
        int n;

        printf( "Feed in the number of terms ");
        scanf("%d", &n);

        if (n==1)
                printf( "1");
        else if(n==2)
                printf("1    1");
        else
        {
                printf( "1    1     ");
                fibo(1,1,2,n);
        }
}
```

NOTE: The logic of c=a+b can only begin if n >=3. Hence special cases of n=1 & n=2 have been handled separately in main.

**Example 5:**

(b) Write a program to find value of y using recursive function, where $y = x^n$, x & y are real number & 'n' is an integer.

**NOTE:**

$$y = x^n$$
$$y = x [ x^{n-1}]$$
$$y = x[x[x^{n-2}]]$$
$$\text{.....}$$
$$\text{.....}$$
$$y = x[x[x[x\text{.....}[x^0]\text{....}]]]]]]$$

**Recursion will stop when the power of x becomes 0.**

**Program:**

```
#include <stdio.h>
float power(float x, int n)
{
        if (n==0)
        {
                return 1;
        }
        else
        {
                return (x * power(x,n – 1));    //power reduces by 1, hence n – 1
        }
}

void main( )
{
        float x,y;
        int n;

        printf( "Enter x,n");
        scanf("%f%d",&x,&n);

        y = power(x,n);

        printf( "Answer=%f",y);
}
```

**Example 6:**

Recurive function to find sum of digits of a number

```
#include<stdio.h>
int sod(int n)
{
        if(n==0)        return 0;
        else    return (n%10+sod(n/10));
}

void main()
{
        int ans,n;
        printf("Enter a number ");
        scanf("%d",&n);

        ans=sod(n);

        printf("Sum of digits=%d",ans);
}
```

**Example 7:**

Recurive function to reverse a number

```
#include<stdio.h>
int rev(int n)
{
        static int r;
        if(n==0)
                return r;
        else
        {
                r=r*10+n%10;
                return rev(n/10);
        }
}

void main()
{
        int ans,n;
        printf("Enter a number ");
        scanf("%d",&n);

        ans=rev(n);

        printf("reverse=%d",ans);
}
```

**Example 8:**

Recurive function to convert decimal number to binary

```
#include<stdio.h>
int dbin(int n)
{
        static int r;
        if(n==0)
                return 0;
        else
                 return (n % 2 + 10 * dbin(n / 2));
}
void main()
{
        int ans,n;
        printf("Enter a number ");
        scanf("%d",&n);

        ans=dbin(n);

        printf("Binary equivalent=%d",ans);
}
```

Navlakhi®


# Chapter 9

# File Handling

```
/*Read N students name & 3 subj marks & write it in a file*/

#include <stdio.h>

void main()

{

 FILE *fp;

 char name[40];

 int m1,m2,m3,n,i,j;


 printf("Feed in number of students ");

 scanf("%d",&n);


 fp=fopen("prog1.dat","w");


 for(i=1;i<=n;i++)
 {
  printf("feed in the name ");

  for(j=0;j<40;j++)

        name[j]='\0';

  scanf("%s",name);

  printf("feed in the 3 subject marks ");

  scanf("%d%d%d",&m1,&m2,&m3);

  fprintf(fp,"%s\n%d\n%d\n%d\n",name,m1,m2,m3);

 }

 fclose(fp);

}
```

```
/*      Read N students name & 3 subj marks from a file & print tabular result*/

#include <stdio.h>

void main()
{
 FILE *fp;

 char name[40];

 int m1,m2,m3,j;


 fp=fopen("prog1.dat","r");


 printf("%-30s%6s%6s%6s%6s\n","Name","Sub1","Sub2","Sub3","Total");


printf("=================================================================\n");


 while(!feof(fp))
 {
  for(j=0;j<40;j++)
        name[j]='\0';

  fscanf(fp,"%s%d%d%d\n",name,&m1,&m2,&m3);

  printf("%-30s%6d%6d%6d%6d\n",name,m1,m2,m3,m1+m2+m3);

 }

 fclose(fp);

}
```

```
/*      Read N students name & 3 subj marks & write it in a file*/

#include <stdio.h>

void main()
{
 FILE *fp;

 char name[40];

 int m1,m2,m3,n,i,j;


 printf("Feed in number of students ");

 scanf("%d",&n);


 fp=fopen("prog3.dat","w");


 for(i=1;i<=n;i++)
 {
  printf("feed in the name ");

  getchar();

  for(j=0;j<40;j++)
        name[j]='\0';

  gets(name);

  printf("feed in the 3 subject marks ");

  scanf("%d%d%d",&m1,&m2,&m3);


  fprintf(fp,"%s\n%d\n%d\n%d\n",name,m1,m2,m3);

 }

 fclose(fp);

}
```

Navlakhi®

/*Read N students name & 3 subj marks from a file & print tabular result*/

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

void main()
{
 FILE *fp;
 char name[40];
 int m1,m2,m3,j;

 fp=fopen("prog3.dat","r");

 printf("%-30s%6s%6s%6s%6s\n","Name","Sub1","Sub2","Sub3","Total");

 printf("=================================================================\n");


 while(!feof(fp))
 {
  for(j=0;j<40;j++)
        name[j]='\0';

  fgets(name,40,fp);

  fscanf(fp,"%d%d%d\n",&m1,&m2,&m3);

  name[strlen(name)-1]='\0';


  printf("%-30s%6d%6d%6d%6d\n",name,m1,m2,m3,m1+m2+m3);

 }

 fclose(fp);

}
```

/* Write into a file N student data (high level IO)*/

```
#include <stdio.h>
struct student
{
 char name[40];
 int m1,m2,m3;
};

void main()
{
 FILE *fp;
 int i,n,j;
 struct student x;


 fp=fopen("prog5.dat","w");

 printf("Feed in number of students ");

 scanf("%d",&n);


 for(i=0;i<n;i++)
 {
  printf("Feed in the name of student ");

  for(j=0;j<40;j++)
        x.name[j]='\0';

  scanf("%s",x.name);


  printf("feed in the 3 subject marks ");

  scanf("%d%d%d",&x.m1,&x.m2,&x.m3);


  fprintf(fp,"%s\n%d\n%d\n%d\n",x.name,x.m1,x.m2,x.m3);

 }

 fclose(fp);
}
```

```
/*      Write into a file N student data (low level IO)*/

#include <stdio.h>
struct student
{
 char name[40];
 int m1,m2,m3;
};

void main()
{
 FILE *fp;
 int i,n,j;
 struct student x;

 fp=fopen("prog7.dat","w");

 printf("Feed in number of students ");
 scanf("%d",&n);


 for(i=0;i<n;i++)
 {
  printf("Feed in the name of student ");

  for(j=0;j<40;j++)
        x.name[j]='\0';

  scanf("%s",x.name);

  printf("feed in the 3 subject marks ");
  scanf("%d%d%d",&x.m1,&x.m2,&x.m3);


  fwrite(&x,sizeof(struct student),1,fp);

 }

 fclose(fp);

}
```

```
/*      read from a file N student data (low level IO)*/

#include <stdio.h>

struct student
{
 char name[40];
 int m1,m2,m3;
};

void main()
{
 FILE *fp;
 int  j;
 struct student x;

 fp=fopen("prog7.dat","r");

 printf("%-30s%6s%6s%6s%6s\n","Name","Sub1","Sub2","Sub3","Total");
 printf("=================================================================\n");


 while(1)
 {

  for(j=0;j<40;j++)
        x.name[j]='\0';

  fread(&x,sizeof(struct student),1,fp);

  if (feof(fp)) break;

  printf("%-30s%6d%6d%6d%6d\n",x.name,x.m1,x.m2,x.m3,x.m1+x.m2+x.m3);


 }

 fclose(fp);

}
```

```
/*Copy a file*/
#include <stdio.h>
#include <stdlib.h>

void main(int argc, char **argv)
{
 FILE *fp1,*fp2;
 char c;

 if (argc!=3)
 {
        printf("Insufficient number of arguments ");
        exit(0);
 }

 fp1=fopen(argv[1],"r");
 fp2=fopen(argv[2],"w");

 while(1)
 {
        fread(&c,sizeof(char),1,fp1);
        if(feof(fp1)) break;
        fwrite(&c,sizeof(char),1,fp2);
 }
 fcloseall();
}
```

Navlakhi®

```
/*View contents of a file*/
#include <stdio.h>
#include <stdlib.h>
void main(int argc, char **argv)
{
 FILE *fp1;
 char c;

 if (argc!=2)
 {
        printf("Insufficient number of arguments ");
        exit(0);
 }

 fp1=fopen(argv[1],"r");

 while(1)
 {
        fread(&c,sizeof(char),1,fp1);
        if(feof(fp1)) break;
        printf("%c",c);
 }
 fclose(fp1);
}
```

Navlakhi®

```
/*File TYPE Command*/
#include <stdio.h>

void main(int argc, char** argv)
{
 int i;
 char c;

 FILE *fp;

 if(argc>=2)
 {
    for(i=1;i<argc;i++)
    {
            fp=fopen(argv[i],"r");
            if (fp==NULL)
                    printf("file does not exists...");
            else
            {
                    while(!feof(fp))
                    {
                    c=fgetc(fp);
                    putc(c,stdout);
                    }
            }
            fclose(fp);
  }
 }
 else
 printf("Invalid number of args.");
}
```

Navlakhi®

**FILE TYPES**

File Handling can be done using two methods
- Formatted file
- Unformatted file

In formatted file the data is represented in its actual form (i.e. in the type it belongs to) i.e. an integer is printed into the file as an integer (%d) etc.

In unformatted file the data is represented in its raw form. The entire data (variable/array/structure) is written into the file using one low level command (fwrite).

| Command | Type | Meaning/Syntax |
|---|---|---|
| fprintf | Formatted | Used to write data into the file |
| | High-level | e.g. fprintf(fp,"%d\%s\n",id,name); |
| fscanf | Formatted | Used to read data from the file |
| | High-level | e.g. fprintf(fp,"%d\%s\n",&id,name); |
| fputs | Formatted | Used to write strings into the file |
| | High-level | e.g. fputs(name,fp); |
| fgets | Formatted | Used to read strings from the file |
| | High-level | e.g. fgets(name,40,fp); |
| fputc | Formatted | Used to write a character into the file |
| | High-level | e.g. fputc(c,fp); |
| fgetc | Formatted | Used to read a character from the file |
| | High-level | e.g. c=fgetc(fp); |
| fwrite | Unformatted | Used to write data into a file in raw form |
| | Low-level | e.g. fwrite(&X,sizeof(struct student),1,fp); |
| fread | Unformatted | Used to read data from a file in raw form |
| | Low-level | e.g. fread(&X,sizeof(struct student),1,fp); |

Navlakhi®

# Chapter 10

# Extra Theory

1. Explain pre-processor directive
2. Explain Control structures in C
3. Explain built in string manipulation functions of C
4. Briefly show the use of storage classes in C
5. Explain Ternary Operators
6. Explain backslash (escape) sequences
7. Compare switch statement with control ladder statement if – else – if
8. Compare while & do – while
9. Explain typedef & enum (enumerated data type)
10. Explain command line arguments with suitable example
11. Explain call by value, call by address/call by reference
12. Explain recursion
13. Explain Macros
14. Explain single indirection & multiple indirection in pointers
15. How does structure differ from a union
16. What is dynamic memory allocation
17. Explain the various function used in file handling

**Answer 1**

The preprocessor is a part of the C compiation process that recognizes special statements (beginning with a #) that may be interspersed throughout the program. As its name implies, the preprocessor actually analyses these statements before analyzing the program.

- #define

  One of the primary uses of #define is for assigning symbolic names to program constants. The statement

  #define TRUE 1

  defines the name TRUE equivalent to 1. The name TRUE can be subsequently used anywhere in the program where the constant 1 is needed. Whereever TRUE appears 1 will be automatically be substituted into the program during pre-processing, the process called MACRO SUBSTITUTION.
  NOTE that the value of the symbolic constant cannot be changed anywhere in the program i.e. TRUE=10 is an invalid statement.
  Thus Macro has a general form

  #define identifier value

  Macros can have arguments (parameters) (**Parameterized Macro**) as shown in the following example

  #define CUBE(x) (x*x*x)

  Here the macro name is CUBE which is defined by the parameter x. When call as

  Y= CUBE(a);

  the above line would get replaced (substituted) as

  y= (a*a*a);

Macros can also be nested (**Nesting of Macro**) as follows

```
#define SQUARE(x)   (x*x)
#define CUBE(x)     (SQUARE(x)*x)
```

Here, y= CUBE(a);
would be substituted as y= (SQUARE(a)*a);
which in turn would get substituted to y=((a*a)*a);

- #include

A programmer can put all the definitions, common functions &/or other statements into a new file & then include that file with his/her program using

#include "myincludefile.h"

Standard manufacturer provided functions can also be used after importing the corresponding header file. e.g. to use sqrt the programmer will have to include math.h as follows

#include<math.h>

- Conditional compilation (#ifdef)

```
#ifdef IBMPC
        #define INT_SIZE 16
#else
        #define INT_SIZE 32
#endif
```

The above will have the effect of defining INT_SIZE 16 if the symbol IBMPC has been previously defined & 32 otherwise. To define IBMPC the programmer can write #define IBMPC. UNIX users have another alternative cc -D IBMPC abc.c

**Answer 2:**

Control structures are statements that affect the flow of execution.

Various control structures of C are as follows

- if
- for
- while
- do-while
- goto
- switch
- ?:
- break & continue

[draw the syntax structures, explain each diagram in your own words, & give an example of (say) loops]

**Answer 3:**

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

void main()
{
  char x[100],y[100];
  int l;

  printf("Enter a string ");
  scanf("%s", x);

  l=strlen(x);

  printf("Length=%d\n", l );

  strcpy(y,x);

  printf("Copy=%s\n", y);

  if (strcmpi(x,y)==0)
        printf("they are equal \n");
   else
        printf("Not equal \n");

  strcat(y,"aaa");

  printf("new string = %s\n",y);

  strcpy(x,strchr(y,'a'));

  printf("After search=%s\n",x);

  strcpy(x,strstr(y,"aa"));

  printf("After search=%s\n", x);
}
```

**Answer 4:**

**Table 9.1 Scope and Lifetime of Declarations**

| Storage class | Where declared | Visibility (Active) | Lifetime (Alive) |
|---|---|---|---|
| None | Before all functions in a file (may be initialized) | Entire file plus other files where variable is declared with **extern**. | Entire program (Global) |
| **extern** | Before all functions in a file (cannot be initialized) | Entire file plus other files where variable is declared **extern** and the file where originally declared as global. | Global |
| **static** | Before all functions in a file | Only in that file | Global |
| None or **auto** | Inside a function (or a block) | Only in that function or block | Until end of function or block |
| **register** | Inside a fucntion or block | Only in that function or block | Until end of function or block |
| **static** | Inside a fucntion | Only in that function | Global |

As mentioned earlier, a variable in C can have any one of the four storage classes:

1. Automatic variables
2. External variables
3. Static variables
4. Register variables

We shall briefly discuss the *scope* and *longevity* of each of the above class of variables. The scope of variable determines over what part(s) of the program a variable is actually available for use (active). Longevity refers to the period during which a variable retains a given value during execution of a program (alive). So longevity has a direct effect on the utility of a given variable.

The variables may also be broadly categorized, depending on the place of their declaration, as *internal* (local) or *external* (global). Internal variables are those which are declared within a particular function, while external variables are declared outside of any function.

**Example of automatic data type**

Automatic variables are declared inside a function in which they are to be utilized. They are *created* when the function is called and *destroyed* automatically when the function is exited, hence the name automatic.

```
#include<stdio.h>
void increment()
{
  auto int x=0;
  x=x+1;
  printf("%d\n",x);
}

void main()
{
  increment();
  increment();
  increment();
}
```

Output:

```
1
1
1
```

**Example is static data type**

```
#include<stdio.h>
void increment()
{
  static int x;
  x=x+1;
  printf("%d\n",x);
}

void main()
{
  increment();
  increment();
  increment();
}
```

Output:

1

2

3

[Static variables are always initialized by default to 0. They are alive till the end of the program].

**Example of extern variables**

```
main()
{
        extern int y; /* external declaration */
        . . . .
        . . . .
}
func1()
{
        extern int y; /* external declaration */
        . . . .
        . . . .
}
int y;  /* definition */
```

Although the variable **y** has been defined after both the functions, the *external declaration* of **y** inside the functions informs the compiler that **y** is an integer type defined somewhere else in the program. Note that the **extern** declaration does not allocate storage space for variables.

---

**Example of register memory**

We can tell the compiler that a variable should be kept in one of the machine's registers, instead of keeping in the memory (where normal variables are stored). Since a register access is much faster than a memory access, keeping the frequently accessed variables (e.g., loop control variables) in the register will lead to faster execution of programs.

```
#include <stdio.h>
void main()
{
  register int i;

  for(i=1;i<100;i++)
        printf("%d\n",i);
}
```

**Answer 5:**

Ternary operators are like if-else control statement

e.g.

```
if(x<0)
        y=x+100;
else
        y=x-100;
```

can be written as

```
y=(x<0)?x+100:x-100;
```

Thus, the general form of ternary operators is as follows:

**Identifier=(logical condition)?true_statement:false_statment;**

**Example:**

```
#include<stdio.h>
void main()
{
  int n;

  printf("Feed in a number ");
  scanf("%d",n);

  (n%2==0)?printf("number is even"):printf("Number is odd");
}
```

**Answer 6:**

| Character constant | Meaning |
|---|---|
| \a | Audible alert (bell) |
| \b | Back space |
| \f | Form feed |
| \n | New line |
| \r | Carriage return |
| \t | Horizontal tab |
| \v | Vertical tab |
| \’ | Single quote |
| \” | Double quote |
| \? | Question mark |
| \\ | Backslash |
| \0 | Null |

```
#include <stdio.h>

void main()

{

  printf("This is a new line\nNow its a tab\tNow a audible bell\aand now a backspace\b");

}
```

**Answer 7:**

[Write the basics of both]

Main difference between the two is that, the if-else-ladder can work for ranges & float values, whereas the switch can work only on integer constants.

[Give any one example, like the one below]

**Example for if-else-if ladder**

```
#include<stdio.h>
void main()
{
  int m;

  printf("feed in the month number ");
  scanf("%d",&m);

  if(m==1)
  {
    printf("January");
  }
  else if(m==2)
  {
    printf("February");
  }
  else if(m==3)
  {
    printf("March");
  }
  else if(m==4)
  {
    printf("April");
  }
  else if(m==5)
  {
    printf("May");
  }
  else if(m==6)
  {
    printf("June");
  }
  else if(m==7)
  {
    printf("July");
  }
  else if(m==8)
  {
    printf("August");
```

```
  }
  else if(m==9)
  {
    printf("September");
  }
  else if(m==10)
  {
    printf("October");
  }
  else if(m==11)
  {
    printf("November");
  }
  else if(m==12)
  {
    printf("december");
  }
  else
  {
    printf("Invalid choice");
  }
}
```

**Example for switch**

```
#include<stdio.h>
void main()
{
  int m;
  printf("Enter month number ");
  scanf("%d", &m);

  switch(m)
  {
    case 1:     printf("January");
                break;
    case 2:     printf("February");
                break;
    case 3:     printf("March");
                break;
    case 4:     printf("April");
                break;
    case 5:     printf("May");
                break;
    case 6:     printf("June");
                break;
    case 7:     printf("July");
                break;
    case 8:     printf("August");
                break;
    case 9:     printf("September");
                break;
    case 10:    printf("October");
                    break;
    case 11:    printf("November");
                    break;
    case 12:    printf("December");
                    break;
    default:    printf("Invaid choice");
                    break;
  }
}
```

**Answer 8:**

While is a pre-check look & do-while is a post-check loop. Hence the minimum number of times the while will execute is zero & the minimum number of times do-while will execute is once.

- **Answer 9:**
  ***User – defined data types:***
  - typedef
  - enum

**typedef** allows the user to define an identifier for an existing data type.

*e.g. typedef int marks;*

The above statement creates a new name for the type int.

NOTE: marks is just a new name for an existing data type & is not a new data type. typedef cannot create a new data type.

Now if you want to declare a variable called maths, physics and chemistry of the type int you can define it as:

*int maths, physics, chemistry;*
or still better
*marks maths, physics, chemistry;*

The main advantage of typedef is to create meaningful data type names for increasing readability.

Example for typedef

```
#include<stdio.h>

typedef int integer;

void main()
{
  integer m1,m2,m3,t;

  printf("enter 3 subject marks ");
  scanf("%d%d%d",&m1,&m2,&m3);

  t=m1+m2+m3;

  printf("total=%d",t);
}
```

**enum** (enumerated data type) is another user defined data type.

*enum identifier { value 1, value 2, ……. value n };*
*enum identifier variable_1, variable_2,….., variable_n;*

Then first line of the above syntax defines a user defined data type called identifier. This data type (identifier) can be used to declare variables which can have one of the values that are in the braces (from value 1 to value n) only (as done in line 2). The values in the braces are called *enumeration constants*.

*e.g.*

```
enum months {January, February,……., December};
emun months start_month, end_month;

start_month = January;
end_month = December;
if (start_month == January) end_month = November;
```

NOTE:
By default the value of
January=0, February=1, March=2 and so on.
Thus start_month = January is can also be stated as start_month=0.
In order to change the default value assignment to the enumerated constants we can do the following:

```
enum months {January=1, February,……., December};
```

Now January=1, February=2, March=3 and so on.

The definition & declaration of variables can be stated in one line as:
*enum months {January,……., December} start_month, end_month;*

Example for enum

```
#include<stdio.h>

enum boolean {false,true};

void main()
{
  int x;
  boolean y;

  printf("feed in an integer ");
  scanf("%d",&x);
```

```
  y=(boolean)x%2;
  if(y==false)
    printf("Number is even");
  else
    printf("Number is odd");
}
```

OR

**(write any one, above example or this one)**

```
#include<stdio.h>

enum no {ONE=1,TWO,THREE,FOUR,FIVE};

void main()
{
  enum no x;

  printf("Feed in a number from 1-5 ");
  scanf("%d",&x);

  if(x==ONE)
    printf("You entered 1 ");

  if(x==TWO)
    printf("You entered 2 ");

  if(x==THREE)
    printf("You entered 3 ");

  if(x==FOUR)
    printf("You entered 4 ");

  if(x==FIVE)
    printf("You entered 5 ");
}
```

**Answer 10:**

Command line arguments.

Here the user can specify arguments on the command line (DOS prompt/UNIX prompt). Hence this can be treated as an alternative to scanf.
main() now has 2 arguments. First,the number of parameters, an integer (which I call argc – argument count). Second, the arguments, which is a 2D character array, (which I have called argv – argument value).

Example:

```
#include <stdio.h>

void main(int argc,char **argv)

{

  int i;

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

  {

        printf("Arg%d=%s\n",i,argv[i]);

  }

}
```

Say I save the above program ass command_line.cpp & then build the executable command_line.exe. To run the program the user will go on the command prompt & type

**command_line 1 2 3**

The output will be as follows:

**Arg0=d:\command_line.exe**

**Arg1=1**

**Arg2=2**

**Arg3=3**

**Answer 11:**

Call by Address

Here, the function gets the address of the actual parameters. Hence, the function parameters will be pointers, pointing to the actual parameters. Hence, any changes made to value after dereferencing the parameters will reflect in the original.

Example:

```
#include<stdio.h>
void fun(int *p)
{
  *p=*p+1;
}

void main()
{
  int x=1;

  fun(&x);

  printf("%d",x);
}
```

Output:
2

Here 'p' contains the address of 'x'. Thus, 'p' points to 'x'. Hence, 'x' can also be called as *p. Any changes made to *p will affect 'x' directly.

**Answer 13:**

**Macro:**

Macro has the following general form

**#define identifier string**

e.g. #define COUNT 100

This implies that wherever COUNT is used in the program it shall be substituted by 100 (during compile time) and there would be no existence of the word COUNT in the final compiled version. This implies that the word COUNT is a constant & is symbolic (**symbolic constant**) and the process of replacement of a symbolic constant by its value (during compilation time) is called MACRO SUBSTITUTION.

Macros can have arguments (parameters) (**Parameterized Macro**) as shown in the following example

#define CUBE(x) (x*x*x)

Here the macro name is CUBE which is defined by the parameter x. When call as

Y= CUBE(a);

the above line would get replaced (substituted) as

y= (a*a*a);

Macros can also be nested (**Nesting of Macro**) as follows

#define SQUARE(x) (x*x)
#define CUBE(x) (SQUARE(x)*x)

Here,
y= CUBE(a);
would be substituted as
y= (SQUARE(a)*a);
which in turn would get substituted to
y=((a*a)*a);

**Example:**

To define a macro MAX to find the maximum of 2 numbers

```
#include <stdio.h>
#define MAX(a,b)  ((a>b)?a:b)                /*macro*/

void main( )
{
        int m,n,z;
        printf( "feed in 2 integers ");
        scanf("%d%d",&m,&n)

        z = MAX(m,n);                  /*MACRO CALL*/

        printf("Maximum=%d",z);
}
```

Here the Macro call would get substituted as

z = ((m>n)?m:n);

**Answer 14:**

Single indirection means single pointers & double indirection means double pointers.

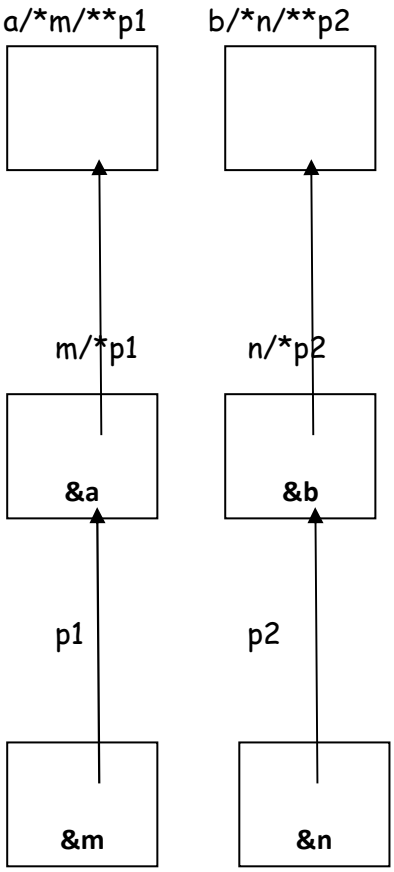
The example below is showing swaping 2 integers using single pointers & the next example used double indirection.

```
#include<stdio.h>
void swap(int *p1,int *p2)
{
  int t;
  t=*p1;
  *p1=*p2;
  *p2=t;
}

void main()
{
  int a,b;

  printf("enter 2 numbers ");
  scanf("%d%d",&a,&b);

  swap(&a,&b);

  printf("Output %d and %d",a,b);
}
```

Double indirection example

```
#include<stdio.h>

void swap(int **p1,int **p2)
{
  int t;
  t=**p1;
  **p1=**p2;
  **p2=t;
}

void main()
{
  int a,b;
  int *m,*n;

  printf("enter 2 numbers ");
  scanf("%d%d",&a,&b);
```

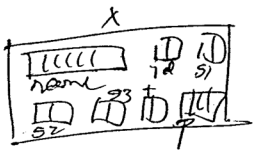
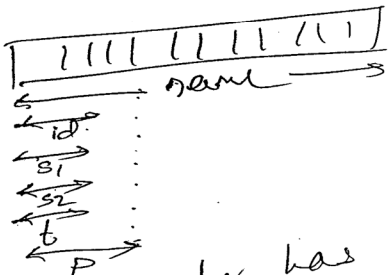
```
  m=&a;
  n=&b;
  swap(&m,&n);
  printf("Output %d and %d",a,b);
}
```




As can be seen from the above examples, single indirection implies the use of single pointers, where the address of a variable will reside in a pointer. In double indirections the address of the data will be in a pointer & the address of that pointer will be in another pointer. Thus, double indirection means double pointers.

**Answer 15:**

7(b)(ii) A union is a collection of items which share a common memory location which has the size same as that of the largest memory location.

| Structure | Union |
|---|---|
| ① struct student<br>{ char name[20];<br>int id, s1, s2, s3, t;<br>float p;<br>}; | ① union student<br>{ char name[20];<br>int id, s1, s2, s3, t;<br>float p;<br>}; |
| ② struct student x; | ② union student x; |
|  |  |
| Each member has a separate memory location | Each member has a common memory location, thus at a given moment only one member will be in memory. |
| ③ strcpy(x.name, "ABCD");<br>x.id = 100;<br>x.s1 = 100;<br>x.s2 = 90;<br>x.s3 = 120;<br>x.t = x.s1+x.s2+x.s3;<br>x.p = x.t/3.0; | ③ strcpy(x.name, "ABCD");<br>x.id = 100;<br>x.s1 = 100;<br>x.s2 = 90;<br>x.s3 = 120;<br>x.t = x.s1+x.s2+x.s3;<br>x.p = x.t/3.0; |

unpredictable behaviour

ISO 9001:2015 Certified

# NOTES

Navlakhi®

ISO 9001:2015 Certified

# NOTES

Navlakhi®