

A NEW GENERATION OF COMPUTER

C++

ADVANCE  
PRINCE SINGH

BY:- PRINCE KUMAR SINGH

# A NEW GENERATION OF COMPUTER

**Program = Algorithm + Flowchart + Language + Translator + Operating System + Datastructure**

**Algorithm:-** Step by step, problem-solving technique is called algorithm.

**Example:-1**

$$\text{Sum} = a + b$$

- Step:-1      Start/Begin
- Step:-2      Read/Input a,b
- Step:-3      Calculate Sum=a+b
- Step:-4      Print/Display Sum
- Step:-5      Stop/End

**Example:-2**

$$Si = p * n * r / 100$$

**Where:-**

- Si      Simple Interest
- p      Principal Amount
- r      Rate of interest
- n      Time Period

- Step:-1      Start/Begin
- Step:-2      Read/Input p, n, r
- Step:-3      Calculate Si= (p\*n\*r)/100
- Step:-4      Print Si
- Step:-5      Stop/End

**Example:-3**      Write algorithm of following formula.

$$A = p * (1 + r/100)^n$$

- A      Amount
- p      Principal Amount
- r      Rate of interest

**BY:- PRINCE KUMAR SINGH**

# A NEW GENERATION OF COMPUTER

n Time Period

- Step:-1 Start/Begin
- Step:-2 Read/Input p, n, r
- Step:-3 Calculate  $A = p \cdot (1 + r/100)^n$
- Step:-4 Print A
- Step:-5 Stop/End

**Example:-4** Write algorithm for checking year is leap or Not Leap.

- Step:-1 Start/Begin
- Step:-2 Read/Input year
- Step:-3 Calculate  $y = \text{year} \text{ Mod } 4$
- Step:-4 If  $y = 0$
- Step:-5 Print "Leap Year"
- Step:-6 If  $y \neq 0$
- Step:-7 Print "Not Leap Year"
- Step:-8 Stop/End

**Example:-5** Write algorithm for checking number is even or odd.

- Step:-1 Start/Begin
- Step:-2 Read/Input number
- Step:-3 Calculate  $y = \text{number} \text{ Mod } 2$
- Step:-4 If  $y = 0$
- Step:-5 Print "Even Number"
- Step:-6 If  $y \neq 0$
- Step:-7 Print "Odd Number"
- Step:-8 Stop/End

**Example:-6** Write algorithm for checking and calculating real roots of any quadratic equation.

- Step 1- Start.
- Step 2- Read coefficient of  $x^2$ , x and Constant.(Say a,b and c respectively).

**BY:- PRINCE KUMAR SINGH**

## A NEW GENERATION OF COMPUTER

Step 3- Calculate  $D=b^2-4ac$ .

Step 4- If  $D \geq 0$

Step 5- Print "Roots are real".

Step 6- Calculate  $x1=(-b+\sqrt{d})/(2a)$ ,  $x2=(-b-\sqrt{d})/(2a)$ .

Step 7- Print root  $x1$  and  $x2$ .

Step 8- If  $D < 0$

Step 9- Print "Roots are Imaginary".

Step 10- Stop.

### Characteristics:-

- ❖ Finiteness.
- ❖ Definiteness.
- ❖ Effectiveness.
- ❖ Input.
- ❖ Output.

**Finiteness** :-Steps of algorithm must be finite.

**Definiteness** :- Each and every steps must be defined.

**Effectiveness** :- Each and every step must be effective.

**Input** :-Algorithm must be associated with inputation.

**Output** :-Algorithm must be associated with output components.

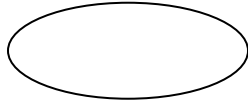
### Flowchart:-

The diagrammatical representation of any algorithm is called flow chart.

Following symbols are used in flowchart.

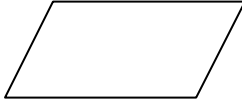
BY:- PRINCE KUMAR SINGH

# A NEW GENERATION OF COMPUTER



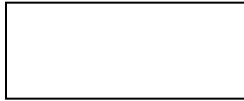
Oval

for [Start/Stop](#)



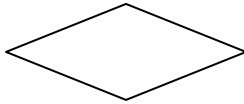
Parallelogram

for [input](#) and [output](#)



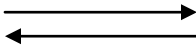
Rectangle

for [process](#)



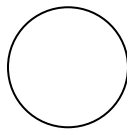
Diamond

for [Decision](#)



Arrow

For [flow direction](#)



Circle

Connector



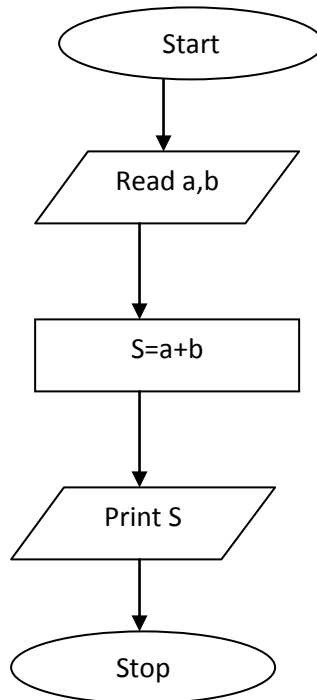
Open Ended Box

[For comment](#)

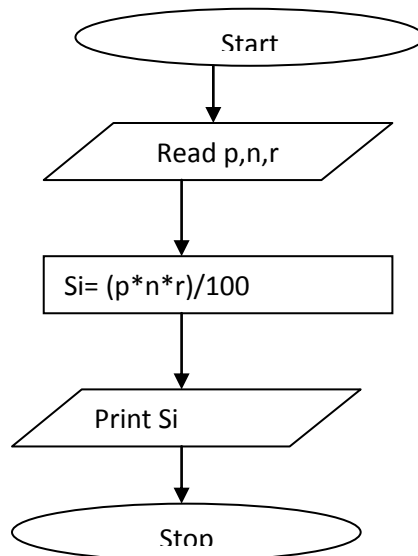
BY:- PRINCE KUMAR SINGH

# A NEW GENERATION OF COMPUTER

**Example:-1** Draw a flow chart for  $s=a+b$



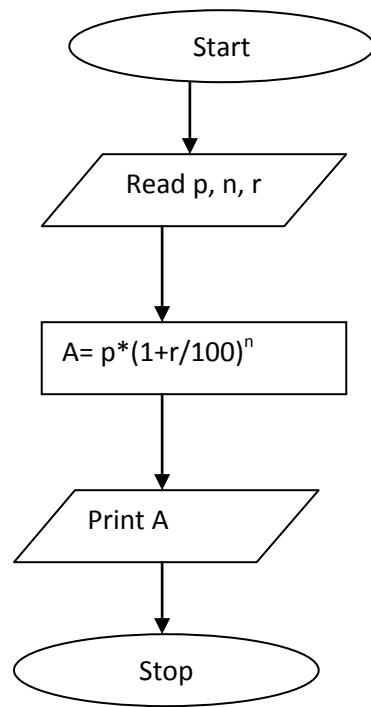
**Example:-2** Draw a flow chart for  $si=p*n*r/100$



**BY:- PRINCE KUMAR SINGH**

# A NEW GENERATION OF COMPUTER

Example:-3 Draw a flow chart for  $a=p*(1+r/100)^n$



## Computer Language:-

Computer languages are categorized into two types

1. **LLL**(Low Level Language)
  - Machine language(0 and 1)
  - Assembly Language(Symbols, codes are used instead of 0 and 1)
2. **HLL**(High Level Language)
  - Natural English like language.

### Example:-

C, C++, JAVA, C#, DOTNET, COBOL, PASCAL, FORTRAN, BASIC, LISP, PROLOG Etc.

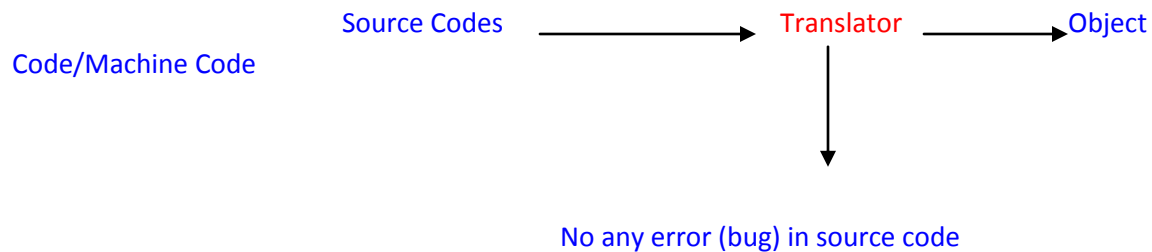
BY:- PRINCE KUMAR SINGH

# A NEW GENERATION OF COMPUTER

## TRANSLATOR:-

It is used for converting source (Program) code into object codes (Machine Codes). There are following three types of translator.

- **Assembler**(Only for assembly language)
- **Interpreter**(Only for Basic Language).
- **Compiler**(All HLL except Basic).



Debug: - To remove error from source codes.

## Advantage of HLL:-

- Easy To understand.
- Fast S/w Development.
- Fast debugging.
- Natural English like language.
- Better portability.

Interpreter	Compiler
1:-Convert source code into object code line by line	1:-Convert entire source code into object code at a time
2:-Debugging is very fast	2:-Debugging is slow.
3:-More Execution time	3:-Less execution Time
4:-Used only in <b>BASIC</b>	4:-Used in all <b>HLL</b> except BASIC

## Operating System :-( OS)

It is collection of s/w which is used for managing computer resources such as :-

BY:- PRINCE KUMAR SINGH



# A NEW GENERATION OF COMPUTER

- ❖ **Memory System**
- ❖ **File System (Heart of application of s/w)**
- ❖ **I/O System**
- ❖ **CPU (Brain of Computer)**

It provides a **platform** for any application s/w. That is, It is **soul** of computer.

Without OS user, never interact with computer hardware to do some work.

## **Example:-**

### CUI (Command User Interface) Based OS:-

- ❖ MS DOS
- ❖ LINUX
- ❖ UNIX

### GUI (Graphical User Interface) Based OS:-

MS Windows 95

MS Windows 98

MS Windows 98 SE

MS Windows ME

MS Windows NT

MS Windows 2000 Professional

MS Windows Advanced Server

MS Windows XP

MS Windows VISTA

MS Windows 2007

## History of 'C' Language:-

Initially **Ken Thompson** developed a language known as **BCPL** (also known as a '**B**' Language). After some time **Dennis Ritchie** modified BCPL language. This modified language was called 'C' language. Since '**C**' is the successor of 'B' language in BCPL.

- B**      Basic
- C**      Combined
- P**      Programming
- L**      Language

**BY:- PRINCE KUMAR SINGH**

# A NEW GENERATION OF COMPUTER

"B" & "C" Language developed at Bell laboratories.

## Programming Elements/Building Blocks of C Language:-

### A:-Data Types:- Gauranteed

1. Simple data Types
  - 1.1. Integer data types
  - 1.2. Real Data type/Floating data Types
  - 1.3. Character data types
2. Structured data types
  - 2.1. Arrays
  - 2.2. Strings
  - 2.3. Structures
  - 2.4. Unions
3. Enumerated data Types
4. Pointer Data type
5. void data Type

### Tables of Integer data types:-

<u>Type</u>	<u>Size</u>	<u>Minimum value</u>	<u>Maximum Value</u>
short int/int	2 Byte	-32768	32767
long int	4 Byte	-2147483648	2147483647

### Tables of Real data/Float data types:-

<u>Type</u>	<u>Size</u>	<u>Minimum value</u>	<u>Maximum Value</u>
float	4 Byte	$3.4 \times 10^{-38}$	$3.4 \times 10^{+38}$
double	8 Byte	$1.7 \times 10^{-308}$	$1.7 \times 10^{+308}$
long double	10 Bytes	$3.4 \times 10^{-4932}$	$1.1 \times 10^{+4932}$

### Character data types:-

It is enclosed into single quote. ASCII range of characters exist between 0 to 255.

### B:-Operators:-

#### Arithmetic Operators

+, -, \*, /, % (Remainder or modulo operator)

BY:- PRINCE KUMAR SINGH

# A NEW GENERATION OF COMPUTER

## Relational Operators

>, <, >=, <=, !=, ==(equal)

## Logical Operators

&&(And), ||(Or), !(Not)

## Bitwise operator

&	Bitwise and
	Bitwise or
^	Bitwise exclusive or
~	Bitwise compliment Operator
<<	Bitwise left shift
>>	Bitwise Right shift

## Increment & decrement Operator

++m	preincrement by one
m++	Post increment by one
--m	predecrement by one
m--	Post decrement by one

## Ternary Operator/Conditional Operator (? :) :-

expression1?expression2:expression3

True

## Address Operator ( & ) :-

Example &a→Address of variable a.

## Indirection Operator ( \* ) :-

\*p→It point address of variable p.

BY:- PRINCE KUMAR SINGH

# A NEW GENERATION OF COMPUTER

## Size of operator:-

`sizeof(variable/expression).`

## Assign operator:-

`=`

## C:-Formatted I/O functions:-

a:-Function for Output:-

`printf("Format String", list of variables);`

b:-Function for Input:-

`scanf("Formate specifire", list of address variables);`

## D:-Formatted specifier:-

`%d` integer

`%f` float

`%ld` Long integer

`%lf` double

`%s` String

`%u` Unsigned decimal

`%e` Exponent notation

`%o` Unsigned octal integer

`%x` Unsigned hexadecimal integer

## E:-Scap Sequence:-

`\n` new line

`\t` tab

`\f` form feed

`\r` carriage return

BY:- PRINCE KUMAR SINGH

# A NEW GENERATION OF COMPUTER

' ' Single quote  
" " double quote  
\\ back slash  
\a alert beep sound

## Program 1:-

```
#include<stdio.h>

main()
{
    printf("Welome in C Programming Language");
}

Where

# Preprocessor directive, which attach c library to header file

stdio.h Standard input/output header file
```

## Program 2:-

```
#include<stdio.h>

main()
{
    int a=2,b=3;
    a=++a + ++b;
    printf("Value of a=%d\n",a);
    printf("Value of b=%d",b);
}
```

BY:- PRINCE KUMAR SINGH

# A NEW GENERATION OF COMPUTER

## Program 3:-

```
#include<stdio.h>

main()

{

    int a=2,b=3;

    a=++a + ++b;

    b=a-- + b--;

    printf("Value of a=%d\n",a);

    printf("Value of b=%d",b);

}
```

## Program 4:-

```
#include<stdio.h>

main()

{

    int a=2,b=3,t1,t2,t3,t4,t5,t6;

    t1=a&b;

    t2=a|b;

    t3=a^b;

    t4=~a;

    t5=a<<2;

    t6=b>>2;

    printf("Value of Bitwise And=%d\n",t1);

    printf("Value of Bitwise OR=%d\n",t2);

    printf("Value of Bitwise Ex OR=%d\n",t3);

    printf("Value of Bitwise Compliment=%d\n",t4);

}
```

**BY:- PRINCE KUMAR SINGH**

## A NEW GENERATION OF COMPUTER

```
printf("Value of Bitwise Left Shift=%d\n",t5);  
printf("Value of Bitwise Right Shift=%d\n",t6);  
}
```

### How To input values Through Keyboard:-

#### Program 5:-

```
# include<stdio.h>  
  
main()  
{  
    int a,b,c1,c2;  
    printf("\nEnter Value of a=");  
    scanf("%d",&a);  
    printf("\nEnter Value of b=");  
    scanf("%d",&b);  
    c1=a+b;  
    c2=a*b;  
    printf("\nSum=%d",c1);  
    printf("\nProduct=%d",c2);  
  
}
```

#### Program 6:-

### Conversion of Celsius into Fahrenheit

$$c=5*(f-32)/9$$

$$f=(9*c)/5+32$$

BY:- PRINCE KUMAR SINGH

# A NEW GENERATION OF COMPUTER

```
# include<stdio.h>

main()

{

float centigrade,foreingn_height;

printf("\nEnter Tempereature in Centigrate=");

scanf("%f",&centigrade);

foreingn_height=(9* centigrade)/5.0 +32;

printf("\nTempereature in Foreignheight=%9.2f", foreingn_height);

}
```

## Program 7:-

### Area of Circle, Volume of Sphere, and Triangle

```
# include<stdio.h>

# include<math.h>

main()

{

float r,a, b,c,s,circle_area,sphere_area,triangle_area;

printf("\nEnter raiouos for circle and Sphere=");

scanf("%f",&r);

printf("\nEnter First side of triangle=");

scanf("%f",&a);

printf("\nEnter Second side of triangle=");

scanf("%f",&b);

printf("\nEnter Third side of triangle=");

scanf("%f",&c);

circle_area=3.14*r*r;
```

**BY:- PRINCE KUMAR SINGH**



## **A NEW GENERATION OF COMPUTER**

```
sphere_area=4.0/3.0*3.14*r*r;  
  
s=(a+b+c)/2;  
  
triangle_area=sqrt(s*(s-a)*(s-b)*(s-c));  
  
printf("\nArea of Circle=%9.3f", circle_area);  
  
printf("\nArea of Sphere=%9.3f", sphere_area);  
  
printf("\nArea of Triangle=%9.3f", triangle_area);  
  
}
```

### **Program 8:-**

#### **Example based on Ternary Operator**

```
# include<stdio.h>  
  
main()  
{  
  
int a,b,c,t1,t2;  
  
printf("\nEnter value of a=");  
  
scanf("%d",&a);  
  
printf("\nEnter value of b=");  
  
scanf("%d",&b);  
  
printf("\nEnter value of c=");  
  
scanf("%d",&c);  
  
t1=a>b?a*c:b*c;  
  
t2=b>c?a*c:b*c;  
  
printf("\nValue of t1=%d",t1);  
  
printf("\nValue of t2=%d",t2);  
  
}
```

### **Program9:-**

**BY:- PRINCE KUMAR SINGH**

# A NEW GENERATION OF COMPUTER

## Example swapping any two numbers

```
# include<stdio.h>

main()

{

int a,b,t;

printf("\nEnter value of a=");

scanf("%d",&a);

printf("\nEnter value of b=");

scanf("%d",&b);

printf("\nValue of a=%d\tValue of b=%d",a,b);

t=a;

a=b;

b=t;

printf("\nValue of a=%d\tValue of b=%d",a,b);

}
```

## Control Statement: - Gauranteed

'C' language provides facilities for controlling the order of execution of the statements, which is referred to as flow control statements/control statements.

There are following three categories of flow control statements.

- ❖ **Decision Control Statements**
  - if statement;
  - if-else statement
  - nested if-else statement
  - else –if construct statement
  - switch case statement
- ❖ **Looping Control Statement**
  - while loop
  - do-while loop
  - for loop
- ❖ **Jumping Control Statement**

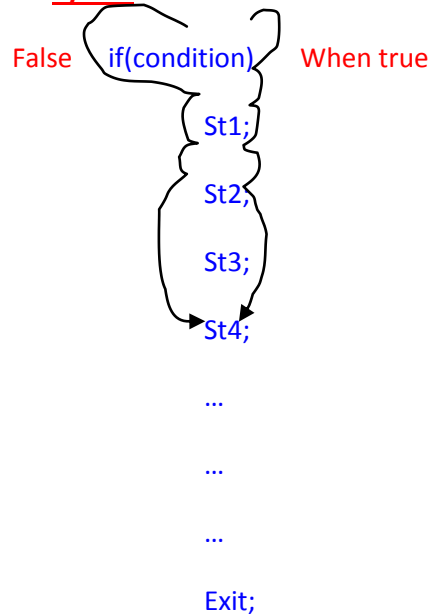
BY:- PRINCE KUMAR SINGH

# A NEW GENERATION OF COMPUTER

- goto
- break
- continue

## if statement:-

### Syntax



### Program10:-

#### Example Check number is even or odd

```
#include<stdio.h>

#include<conio.h>

main()
{
    int n;

    clrscr();

    printf("\nEnter any number=");

    scanf("%d",&n);

    if(n%2==0)
```

BY:- PRINCE KUMAR SINGH

## A NEW GENERATION OF COMPUTER

```
printf("\nNumber is Even=%d",n);

if(n%2!=0)

printf("\nNumber is Odd=%d",n);

getch();

}
```

### Program11:-

#### Example Check year is Leap or Not Leap Year

```
#include<stdio.h>

#include<conio.h>

main()

{

int year;

clrscr();

printf("\nEnter any number=");

scanf("%d",&year);

if(year%4==0)

printf("\nLeap Year=%d",year);

if(year%4!=0)

printf("\nNot Leap Year=%d",year);

getch();

}
```

### Program12:-

#### Example :-Check Profit or Loss or No profit or no Loss

```
#include<stdio.h>

#include<conio.h>

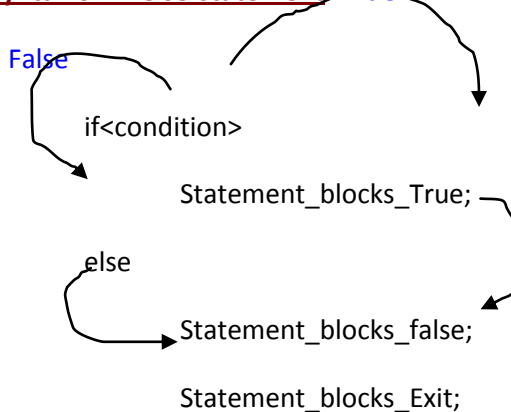
main()
```

BY:- PRINCE KUMAR SINGH

## A NEW GENERATION OF COMPUTER

```
{  
float sale,purchase,m;  
  
clrscr();  
  
printf("\nEnter Purchase cost=");  
  
scanf("%f",& purchase);  
  
printf("\nEnter sale cost=");  
  
scanf("%f",&sale);  
  
m= sale- purchase;  
  
if(m==0)  
  
printf("\nNeither Profit Nor Loss");  
  
if(m>0)  
  
printf("\n Profit=%f",m);  
  
if(m<0)  
  
printf("\n Loss=%f",m);  
  
getch();  
  
}
```

### Syntax of If-else-statement:- True



### Program13:-

#### Example Check year is Leap or Not Leap Year

BY:- PRINCE KUMAR SINGH

## A NEW GENERATION OF COMPUTER

```
#include<stdio.h>

#include<conio.h>

main()

{

int year;

clrscr();

printf("\nEnter any number=");

scanf("%d",&year);

if(year%4==0)

printf("\nLeap Year=%d",year);

else

printf("\nNot Leap Year=%d",year);

printf("\nThank You");

getch();

}
```

### Syntax of Nested If-else-statement:-

```
if<condition1>

    Statement_blocks1;

else

    if<condition2>

        Statement_blocks2;

    else

        if<condition3>

            Statement_blocks3;

        else
```

BY:- PRINCE KUMAR SINGH

## A NEW GENERATION OF COMPUTER

if<condition4>

Statement\_blocks4;

...

...

...

else

Statement\_blocks\_Exit;

### Program14:-

#### Example Largest of any three Numbers:-

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

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

```
main()
```

```
{
```

```
int a,b,c;
```

```
clrscr();
```

```
printf("\nEnter First number=");
```

```
scanf("%d",&a);
```

```
printf("\nEnter Second number=");
```

```
scanf("%d",&b);
```

```
printf("\nEnter Third number=");
```

```
scanf("%d",&c);
```

```
printf("\n\nFirst Number=%d\tSecond Number=%d\tThird Number=%d", a,b,c);
```

```
if(a==b && b==c )
```

```
printf("\nAll Numbers are equal");
```

```
else
```

```
if(a>b && b>c || a>c && c>b )
```

```
printf("\nA is the largest numbers");
```

**BY:- PRINCE KUMAR SINGH**

## A NEW GENERATION OF COMPUTER

```
else

    if(b>c && c>a || b>a && a>c )

printf("\nB is the largest numbers");

else

    printf("\nC is the largest nubers");

getch();

}
```

### Program15:-

#### Example Solving quadratic Equation:-

```
#include<stdio.h>

#include<conio.h>

#include<math.h>

main()

{

float a,b,c,X1,X2,d;

clrscr();

printf("\nEnter Coefficient of X^2=");

scanf("%f",&a);

printf("\nEnter Coefficient of X=");

scanf("%f",&b);

printf("\nEnter Constant Value=");

scanf("%f",&c);

printf("\n\n%3.2fX^2 +%3.2fX+%3.2f=0",a,b,c);

d=b*b-4*a*c;

if(d==0)

{
```

BY:- PRINCE KUMAR SINGH



## A NEW GENERATION OF COMPUTER

```
printf("\nRoots are real equal");

X1=-b/(2*a);

X2=-b/(2*a);

printf("\nReal & Equal Roots=%3.2F",X1);

}

else

    if(d>0)

    {

        printf("\nRoots are real unequal");

        X1=(-b+sqrt(d))/(2*a);

        X2=(-b-sqrt(d))/(2*a);

        printf("\nFirst Real Root=%3.2f\tSecond real root=%3.2f",X1,X2);

    }

    else

        printf("\nRoots are Imaginary & Not Possible");

        getch();

}
```

### Program16:-

#### Example Solving Grading system Problem:-

```
#include<stdio.h>

#include<conio.h>

#include<math.h>

main()

{

    int h,e,m,p,c,tot;

    float per;
```

BY:- PRINCE KUMAR SINGH

## A NEW GENERATION OF COMPUTER

```
clrscr();

printf("\nEnter Marks Obtained In Hindi=");

scanf("%d",&h);

printf("\nEnter Marks Obtained In English=");

scanf("%d",&e);

printf("\nEnter Marks Obtained In Maths=");

scanf("%d",&m);

printf("\nEnter Marks Obtained In Physics=");

scanf("%d",&p);

printf("\nEnter Marks Obtained In Chemistry=");

scanf("%d",&c);

printf("\n-----");

printf("\n\nHindi=%d\tEnglish=%d\tMaths=%d\tPhysic=%d\tChemistry=%d\t",h,e,m,p,c);

printf("\n-----");

tot=h+e+m+p+c;

per=tot/5.0;

if(per>=85 &&per<=100 && h>=33 && e>=33 && m>=33 && p>=33 &&c>=33)

    printf("\n\nA Grade & Passed \t Tot=%d\tPer=%3.2f",tot,per);

else

    if(per>=75 &&per<=100 && h>=33 && e>=33 && m>=33 && p>=33 &&c>=33)

        printf("\n\nB Grade & Passed \t Tot=%d\tPer=%3.2f",tot,per);

    else

        if(per>=65 &&per<=100 && h>=33 && e>=33 && m>=33 && p>=33 &&c>=33)

            printf("\n\nC Grade & Passed \t Tot=%d\tPer=%3.2f",tot,per);

        else

            if(per>=45 &&per<=100 && h>=33 && e>=33 && m>=33 && p>=33 &&c>=33)

                printf("\n\nD Grade & Passed \t Tot=%d\tPer=%3.2f",tot,per);
```

**BY:- PRINCE KUMAR SINGH**

## A NEW GENERATION OF COMPUTER

```
else

if(per>=33 &&per<=100 && h>=33 && e>=33 && m>=33 && p>=33 &&c>=33)

printf("\n\nE Grade & Passed \t Tot=%d\tPer=%3.2f",tot,per);

else

printf("\n\nF Grade & Failed \t Tot=%d\tPer=%3.2f",tot,per);

printf("\n-----");

getch();

}
```

### Syntax of -else-if-Construct Statement/Ladder Statement:-

It is used for solving choice based problem.

```
...

...

...

else

    if<condition>

        Statement_Blocks

        ...

        ...

        ...
```

It is used for solving choice based problem.

### Program17:-

Example Solving Arithmetical ,Year & Number CheckingProblem:-

BY:- PRINCE KUMAR SINGH

## A NEW GENERATION OF COMPUTER

```
#include<stdio.h>

#include<conio.h>

#include<math.h>

main()

{

int ch,n,year;

float a,b,s1,p1,d1,sub;

clrscr();

printf("\n\n\n\t\t-----");

printf("\n\t\tChoice Based Arithmetic,Year & Number Checking\n");

printf("\n\t\t-----");

printf("\n\t\t\t1:-ADDITION");

printf("\n\t\t\t2:-SUBTRACTION");

printf("\n\t\t\t3:-PRODUCT");

printf("\n\t\t\t4:-DIVISION");

printf("\n\t\t\t5:-YEAR CHECKING");

printf("\n\t\t\t6:-NUMBER CHECKING");

printf("\n\t\t\t-----");

printf("\n\t\t\t-----");

printf("\n\t\t\tENTER YOUR CHOICE=");

scanf("%d",&ch);

printf("\n\t\t\t-----");

printf("\n\t\t\tENTER ANY TWO NUMBERS FOR ARITHMETIC OPERATIONS=");

scanf("%f%f",&a,&b);

printf("\n\t\t\t-----");

printf ("\n\t\t\t-----");

if(ch==1)
```

BY:- PRINCE KUMAR SINGH

## A NEW GENERATION OF COMPUTER

```
{  
s1=a+b;  
printf("\n\t\tSUM=%6.2f",s1);  
}  
  
else  
  
if(ch==2)  
  
{  
sub=a-b;  
printf("\n\t\tDIFFERENCE=%6.2f",sub);  
}  
  
else  
  
if(ch==3)  
  
{  
p1=a*b;  
printf("\n\t\tPRODUCT=%6.2f",p1);  
}  
  
else  
  
if(ch==4)  
  
{  
d1=a/b;  
printf("\n\t\tDIVISION=%6.2f",d1);  
}  
  
else  
  
if(ch==5)  
  
{  
printf("\n\t\tENTER YEAR=");  
scanf("%d",&year);
```

**BY:- PRINCE KUMAR SINGH**

## A NEW GENERATION OF COMPUTER

```
if(year%4==0)

printf("\n\t\tLEAP YEAR=%d",year);

else

printf("\n\t\tNOT LEAP YEAR=%d",year);

}

else

if(ch==6)

{

printf("\n\t\tENTER NUMBER=");

scanf("%d",&n);

if(n%2==0)

printf("\n\t\tEVEN NUMBER=%d",n);

else

printf("\n\t\tODD NUMBER=%d",n);

}

else

printf("\n\t\tWRONG CHOICE AGAIN ENTER...!");

getch();

}
```

### **Syntax switch-case statement :-**

It is also used for solving choice based problems. It is an alternative of else-if construct statement.

#### **Syntax:-**

```
switch(expression)

{

case <value1>:

Statement_Blocks_1;
```

**BY:- PRINCE KUMAR SINGH**

## A NEW GENERATION OF COMPUTER

break;

case <value2>:

Statement\_Blocks\_2;

break;

case <value3>:

Statement\_Blocks\_3;

break;

...

...

...

default:

Statement\_Blocks\_False;

}

Exit\_Statement;

### Program18:-

#### Example Solving Day code into day

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

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

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

```
main()
```

```
{
```

```
int day_code;
```

```
clrscr();
```

```
printf("\n\n\n\t\t-----");
```

BY:- PRINCE KUMAR SINGH

## A NEW GENERATION OF COMPUTER

```
printf("\n\t\tCONVERSION OF DAY CODE INTO DAY");
```

```
printf("\n\t\t-----");
```

```
printf("\n\t\tENTER DAY CODE=");
```

```
scanf("%d",&day_code);
```

```
switch(day_code)
```

```
{
```

```
case 1:
```

```
printf("\n\n\t\tSUNDAY");
```

```
break;
```

```
case 2:
```

```
printf("\n\n\t\tMONDAY");
```

```
break;
```

```
case 3:
```

```
printf("\n\n\t\tTUESDAY");
```

```
break;
```

```
case 4:
```

```
printf("\n\n\t\tWEDNESDAY");
```

```
break;
```

```
case 5:
```

```
printf("\n\n\t\tTHURSDAY");
```

```
break;
```

```
case 6:
```

```
printf("\n\n\t\tFRIDAY");
```

```
break;
```

```
case 7:
```

```
printf("\n\n\t\tSATURDAY");
```

```
break;
```

**BY:- PRINCE KUMAR SINGH**



# A NEW GENERATION OF COMPUTER

default :

```
printf("\n\n\t\tWrong day Code");  
  
}  
  
getch();  
  
}
```

## Looping Control Statement/Iteration Statements/Repetition Statement:-

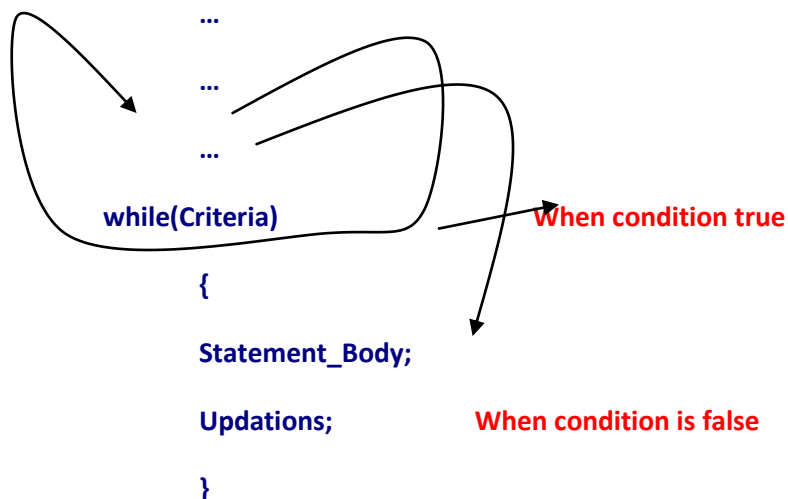
It allows the execution of some set of statements repeatedly till either for a known number of times or till certain conditions are met. There are following three types of looping statements.

- while loop
- do-while loop
- for loop

### while loop:-

It executes looping body when condition is true.

### Syntax of While loop:-



BY:- PRINCE KUMAR SINGH

# A NEW GENERATION OF COMPUTER

Exit\_Statement\_False\_criteria;

## Program19:-

### Example Generating a series of natural numbers:-

```
#include<stdio.h>

#include<conio.h>

#include<math.h>

main()

{

int n,i=1;

clrscr();

printf("\n\nEnter Any Positive integer=");

scanf("%d",&n);

while(i<=n)

{

printf("%d\t",i);

i++;

}

printf("Thank You I Exit because condition is false=%d",i);

getch();

}
```

## Program20:-

BY:- PRINCE KUMAR SINGH

# A NEW GENERATION OF COMPUTER

## Example Generating a series of odd and Even numbers:-

```
#include<stdio.h>

#include<conio.h>

#include<math.h>

main()

{

int n,i=1,j=2;

clrscr();

printf("\n\nEnter Any Positive integer=");

scanf("%d",&n);

printf("\nSeries of Even Numbers=\n");

while(j<=n)

{

printf("%d\t",j);

j=j+2;

}

printf("\nSeries of Odd Numbers=\n");

while(i<=n)

{

printf("%d\t",i);

i=i+2;

}

printf("\n\nThank You I Exit because condition is false=%d",i);

getch();

}
```

## Program22:-

BY:- PRINCE KUMAR SINGH

# A NEW GENERATION OF COMPUTER

## Example Generating a series of $n^n$ :-

```
#include<stdio.h>

#include<conio.h>

#include<math.h>

main()

{

long int n,i=1,m1;

clrscr();

printf("\n\nEnter Any Positive integer=");

scanf("%ld",&n);

printf("\nSeries of Power of Same Number=\n");

while(i<=n)

{

m1=pow(i,i);

printf("%ld\t=%ld\t\n",i,m1);

i++;

}

getch();

}
```

## Program23:-

## Example Generating a series of Reverse Natural Numbers :-

```
#include<stdio.h>

#include<conio.h>
```

BY:- PRINCE KUMAR SINGH

## A NEW GENERATION OF COMPUTER

```
#include<math.h>

main()
{
int n,m;

clrscr();

printf("\n\nEnter Any Positive integer m>n=");

scanf("%ld",&m);

printf("\n\nEnter Any Positive integer m<n=");

scanf("%ld",&n);

while(m>=n)
{
printf("%d\t",m);

m--;

}

getch();

}
```

### Program24:-

#### Example Generating a series of Factorial Numbers :-

```
#include<stdio.h>

#include<conio.h>

#include<math.h>

main()
{

long int n,i=1,fact=1;
```

BY:- PRINCE KUMAR SINGH

# A NEW GENERATION OF COMPUTER

```
clrscr();

printf("\n\nEnter Any Positive integer =");

scanf("%ld",&n);

while(i<=n)

{

fact=fact*i;

printf("\nFactorial of Number=%ld=%ld\t",i,fact);

i++;

}

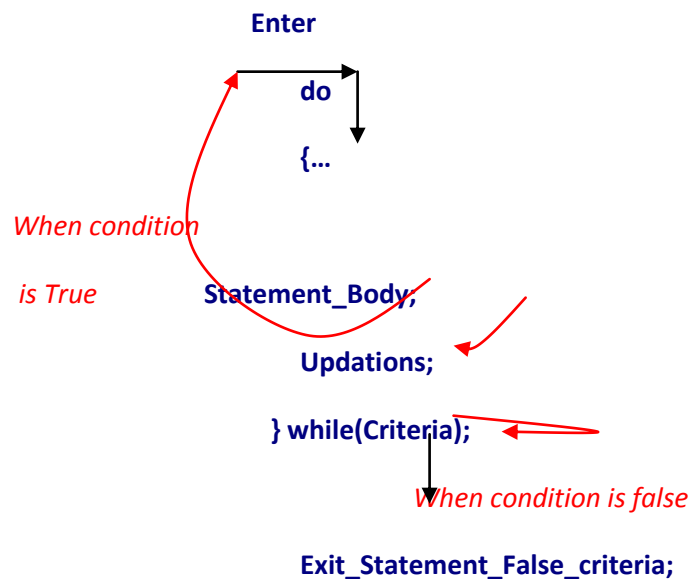
getch();

}
```

## do-while loop:-

It executes looping body one time when condition is false. and further execute when condition is true.

### Syntax of do-while loop:-



BY:- PRINCE KUMAR SINGH


# A NEW GENERATION OF COMPUTER

## Program25:-

### Example Generating a Fibonacci Numbers :-

0      1      1      2      3      5      8      13      21      34      55      ...

Logic:-



a	b	s=1+b
a	b	s=a+b
0	1	0
1	0	1
0	1	1
1	1	2
1	2	3
2	3	5
3	5	8
5	8	13
8	13	21
.	.	.
.	.	.
.	.	.

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

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

```
main()
```

```
{
```

```
int i=0,a=0,b=1,s=0,n;
```

```
clrscr();
```

BY:- PRINCE KUMAR SINGH

## A NEW GENERATION OF COMPUTER

```
printf("\n\nEnter Value of n=");
```

```
scanf("%d",&n);
```

```
do
```

```
{
```

```
printf("%d\t",s);
```

```
a=b;
```

```
b=s;
```

```
s=a+b;
```

```
i++;
```

```
}
```

```
while(i<=n);
```

```
getch();
```

```
}
```

### Program26:-

#### Example Generating a Series of Armstrong Numbers :-

1      153      370      371      407      .      .      .

Logic:-

$$1^3=1$$

$$153=1^3+5^3+3^3=153$$

$$370=3^3+7^3+0^3=370$$

$$371=3^3+7^3+1^3=371$$

$$370=3^3+7^3+0^3=370$$

$$407=4^3+0^3+7^3=407$$

...

...

BY:- PRINCE KUMAR SINGH



# A NEW GENERATION OF COMPUTER

...

## Program26:-

### Example Generating a Series of Armstrong Numbers :-

```
#include<stdio.h>

#include<conio.h>

main()

{

    int n=0,m=0,d=0,s=0,a=1,k;

    clrscr();

    printf("\n\nEnter Value of k=");

    scanf("%d",&k);

    while(a<=k)

    {

        n=a;

        s=0;

        while(n>0)

        {

            m=n%10;

            n=n/10;

            s=s+m*m*m;

        }

        if(s==a)

            printf("Series of Armstrong Numbers=%d\n",a);

        a++;

    }

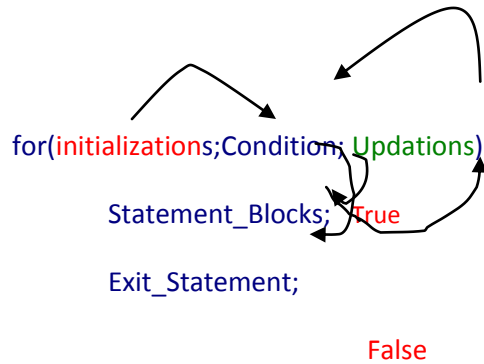
    getch();
```

BY:- PRINCE KUMAR SINGH

# A NEW GENERATION OF COMPUTER

}

## Syntax of for loop:-



- Intializations** :-There are many initializations by using commas.
- Updatons** :-There are many Updatons by using commas.
- Condition** :-Only one Condition will be defined.
- Semicolan** :-There are two semicolons must be inside for loop.

## Program27:-

### Example Generating a Series of natural Numbers and there Sum :-

```
#include<stdio.h>

#include<conio.h>

main()
{
    int n,i,s;

    clrscr();

    printf("\nEnter Value of n=");

    scanf("%d",&n);
```

BY:- PRINCE KUMAR SINGH

## A NEW GENERATION OF COMPUTER

```
for(i=1,s=0;i<=n;i++)  
{  
    printf("%d\t",i);  
    s=s+i;  
}  
printf("\n\nSum=%d",s);  
getch();  
}
```

### Program28:-

#### Example Generating fibonacci series using for loop

```
#include<stdio.h>  
#include<conio.h>  
main()  
{  
    int a,b,n,i,s;  
    clrscr();  
    printf("\nEnter Value of n=\n\n");  
    scanf("%d",&n);  
    for(a=0,b=1,s=0,i=0;i<=n;a=b,b=s,s=a+b,i++)  
  
    printf("%d\t",s);  
    getch();  
}
```

#### Syntax of nested for loop:-

BY:- PRINCE KUMAR SINGH

# A NEW GENERATION OF COMPUTER

```
for(initializations1;Condition1; Updations1)
{
    ....
    ....
    for(initializations2;Condition2; Updations2)
    {
        ....
        ....
        for(initializations3;Condition3; Updations3)
        {
            ....
            ....
            for(initializations4;Condition4; Updations4)
            {
                ....
                ....
                ....
            }
        }
    }
}

Exit_Statement;
```

Program29:- (Pattern of Numbers and Stars)

BY:- PRINCE KUMAR SINGH

## A NEW GENERATION OF COMPUTER

```
#include<stdio.h>

#include<conio.h>

main()

{

int i,j,n;

clrscr();

printf("\nEnter Value of n=\n\n");

scanf("%d",&n);

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

{

for(j=i;j<=n;j++)

printf("%d",i);

printf("\n");

}

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

{

for(j=i;j<=n;j++)

printf("%d",j);

printf("\n");

}

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

{

for(j=i;j<=n;j++)

printf("*",i);

printf("\n");

}
```

BY:- PRINCE KUMAR SINGH

## A NEW GENERATION OF COMPUTER

```
getch();  
}
```

### Program30(Smiling face Triangle:-

```
#include<stdio.h>  
  
#include<conio.h>  
  
main()  
{  
  
int i,j,n;  
  
clrscr();  
  
printf("\nEnter Value of n=\n\n");  
  
scanf("%d",&n);  
  
printf("\nSmiling Face Triangle\n");  
  
for(i=1;i<=n;i++)  
{  
  
for(j=i;j<=n;j++)  
  
printf(" ",i);  
  
printf("\n");  
  
}  
  
getch();  
}
```

### Program31(Series of Prime Numbers="):-

2 3 5 7 11 13 17 19 23 29 ...

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

BY:- PRINCE KUMAR SINGH

## A NEW GENERATION OF COMPUTER

```
int i,j,n,f;

clrscr();

printf("\nEnter Value of n=\n\n");

scanf("%d",&n);

printf("\nSeries of Prime Numbers\n\n");

for(i=2;i<n;i++)

{

    for(f=0,j=2;j<i;j++)

    {

        if(i%j==0)

        {

            f=1;

            break;

        }

    }

    if(f==0)

    {

        printf("\t%d",i);

    }

}

getch();

}
```

**Program32(Series of ASCII):-**

```
#include<stdio.h>

#include<conio.h>
```

**BY:- PRINCE KUMAR SINGH**

## A NEW GENERATION OF COMPUTER

```
main()
{
    int i;

    clrscr();

    printf("\nSeries of ASCII Codes\n\n");

    for(i=0;i<=255;i++)

    printf("%d==%c\t",i,i);

    getch();
}
```

### Jumping Control Statement:-

goto statement:-

Syntax:-

goto <label>;

### Syntax for define label:-

<label\_name>:

### Program 33

```
#include<stdio.h>

#include<conio.h>

main()
{

    float a,b,c1,c2,c3;

    clrscr();

    printf("\nEnter Value of a and b=");
```

BY:- PRINCE KUMAR SINGH



## A NEW GENERATION OF COMPUTER

```
scanf("%f%f",&a,&b);

m1 :

{

c1=a+b;

printf("\nSum=%f",c1);

goto m4;

}

m2:

{

c2=a*b;

printf("\nProduct%f",c2);

}

m3:

{

c3=a/b;

printf("\nDivision%f",c3);

}

m4:

printf("\n\nThank You");

getch();

}
```

Program 34 (Program for adding any five digits)

#include<stdio.h>

BY:- PRINCE KUMAR SINGH

## A NEW GENERATION OF COMPUTER

```
#include<conio.h>

main()
{
clrscr();

int num,a,n ;

int sum=0;

printf("\nEnter Value of five digits=");

scanf("%d",&num);

a=num%10;

n=num/10;

sum=sum+a;

a=n%10;

n=n/10;

sum=sum+a;

a=n%10;

n=n/10;

sum=sum+a;

a=n%10;

n=n/10;

sum=sum+a;

a=n%10;

sum=sum+a;

printf("\nSum=%d=\t%d",num,sum);

getch();

}
```

Program 35 (Program for reverse any five digits)

BY:- PRINCE KUMAR SINGH

# A NEW GENERATION OF COMPUTER

```
#include<stdio.h>

#include<conio.h>

main()

{

clrscr();

int n,a,b;

long int rev=0;

clrscr();

printf("\nEnter any five digits=");

scanf("%d",&n);

a=n%10;

n=n/10;

rev=rev+a*10000L;

a=n%10;

n=n/10;

rev=rev+a*1000;

a=n%10;

n=n/10;

rev=rev+a*100;

a=n%10;

n=n/10;

rev=rev+a*10;

a=n%10;

rev=rev+a;

printf("\nRev number=%ld",rev);

getch();
```

**BY:- PRINCE KUMAR SINGH**

# A NEW GENERATION OF COMPUTER

}

## Program 36 (Program for checking Number is prime or not)

```
#include<stdio.h>

#include<conio.h>

main()

{

clrscr();

int n,i;

clrscr();

printf("\nEnter any number=");

scanf("%d",&n);

for(i=2;i<n;i++)

{

if(n%i==0)

{

printf("\nNot prime");

break;

}

}

if(i==n)

printf("Prime Number");

getch();

}
```

**Note:-**break statement used only in switch case control and looping control statement.

**Continue Control Statement:-**

**BY:- PRINCE KUMAR SINGH**

## **A NEW GENERATION OF COMPUTER**

This statement skips the remainder of the current iteration and initiates the execution of the next iteration. When this statement is encountered in a loop, the rest of the statements in the loop are skipped and the control passes to the condition, which is evaluated, and if true, the loop is entered again.

Syntax:-

```
continue;
```

**BY:- PRINCE KUMAR SINGH**

# A NEW GENERATION OF COMPUTER

## Function:- Full question

It is subprogram which is used for performing some well defined specific task.

Function may or may not consist of arguments. Arguments enclosed within parenthesis.

Or

A function is a set of program statements that can be processed independently. A function can be invoked which behaves as though its code is inserted at the point of the function call. The communication between caller (Calling function) and callee (called function) takes place through parameter.

### Example:-

$f(x)$  Function with single argument/Parameter  $x$ .

$f()$  Function with no argument/Parameter  $x$ .

$f(x_1, x_2, x_3, \dots)$  Function with multiple arguments/Parameters.

## Advantage of Function:-

- ❖ Modular programming.
- ❖ Reduction in the amount of work and development time.
- ❖ Program and function debugging is easier.
- ❖ Reduction in the size of program due to code reusability
- ❖ Library of functions can be implemented by combining well designed, tested and proven functions.

## Types of functions:-

- ✓ Built In function/System defined functions
- ✓ User Defined Functions
  - A function without argument and no return value.
  - A function without argument and return value.
  - A function with argument and no return value.
  - A function with argument and return value.
  - A function call by Value and call by reference.
  - Recursive function/Calling itself function.

## Built In function/System defined functions:-

### Example:-

$\text{pow}(m, n)$        $m^n$

BY:- PRINCE KUMAR SINGH

## A NEW GENERATION OF COMPUTER

log(m)	m>0	
sqrt(x)	x>=0	
ln(x)	x>0	Natural log to the base 2<e<3
strlen(string)		For measuring length of string
strrev(string)		For reversing of string
gets(string)		Accept a string from standard input device
puts(string)		This function outputs a string constant or a string variable to the standard output device.
getchar()		It return a character that has been recently typed.
getche()		It also return a character that has been recently typed.The typed character is echoed to the computer screen.
getch()		This function too returns a character that has been recently typed.But neither the user is required to type enter key after entering character nor the typed character echoed to the computer screen.
putchar()		This function output a character constant or a character variable to the standard output device.
printf("format of string",list of variables)		Console output
scanf("format specifier",list of address variables)		Console input
main()		
clrscr()		
etc.		

### Function Components:-

- ❖ Function declaration or prototype.
- ❖ Function parameter(Formal Parameter)
- ❖ Combination of function declaration and its definition
- ❖ Function definition(function declarator and function body)
- ❖ Return statement.
- ❖ Function call.

### User Defined Functions:-

BY:- PRINCE KUMAR SINGH

# A NEW GENERATION OF COMPUTER

- A function without argument and no return value.

## Program 37

```
#include<stdio.h>

#include<conio.h>

main()

{

clrscr();

sum();

leap();

getch();

}

sum()

{

int a,b,s;

printf("\nEnter Any Two numbers=");

scanf("%d%d",&a,&b);

s=a+b;

printf("\n Sum=%d",s);

}

leap()

{

int year,p;

printf("\nEnter Year=");

scanf("%d",&year);

p=year%4;
```

BY:- PRINCE KUMAR SINGH



## A NEW GENERATION OF COMPUTER

```
if(p==0)

printf("\nLeap year");

else

printf("\nNot Leap Year");

}
```

- A function without argument and return value.

### Program 38

```
#include<stdio.h>

#include<conio.h>

main()

{

int m;

clrscr();

m=pro();

printf("\n Product of any two numbers=%d",m);

getch();

}

pro()

{

int a,b,d;

printf("\nEnter Any Two numbers=");

scanf("%d%d",&a,&b);

d=a*b;

return d;
```

BY:- PRINCE KUMAR SINGH

# A NEW GENERATION OF COMPUTER

}

- A function with argument and no return value.

## Program 39

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

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

```
main()
```

```
{
```

```
int a,b;
```

```
printf("\nEnter Any Two numbers=");
```

```
scanf("%d%d",&a,&b);
```

```
pro(a,b); //a and b are actual arguments
```

```
getch();
```

```
}
```

```
pro(int a1,int b1) //a1 and b1 are Formal arguments/referencing variables
```

```
{
```

```
int d;
```

```
d=a1*b1;
```

```
printf("\n\nProduct of any two digits=%d",d);
```

```
}
```

- A function with argument and return value.

## Program 40

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

BY:- PRINCE KUMAR SINGH

## A NEW GENERATION OF COMPUTER

```
#include<conio.h>

main()
{
    int a,b,d1;

    printf("\nEnter Any Two numbers=");

    scanf("%d%d",&a,&b);

    d1=pro(a,b);                //a and b are actual arguments

    printf("\nProduct of any two integer=%d",d1);

    getch();
}

pro(int a1,int b1)              //a1 and b1 are Formal arguments/referencing variables
{
    int d;

    d=a1*b1;

    return d;
}
```

❖ A function call by Value and call by reference

### Program 41

```
#include<stdio.h>

#include<conio.h>

main()
{
    int a,b;

    printf("\nEnter Any Two numbers=");

    scanf("%d%d",&a,&b);
```

BY:- PRINCE KUMAR SINGH

## A NEW GENERATION OF COMPUTER

```
pro(&a,&b);           //a and b are actual arguments

getch();

}

pro(int *a1,int *b1)   //a1 and b1 are Formal arguments/referencing variables
{
int d1;
d1=(*a1)*(*b1);
printf("\nProduct of any two integer=%d",d1);
}
```

**Note:-** \*a1,\*b1 both are pointer variables which point address of formal arguments.

These arguments reference address of actual arguments.

### ○ Recursive function/Calling itself function.

A function that contains a function call to itself or a function call to a second function which eventually calls the first function is known as recursive function.

### Condition for recursion:-

- ❖ Each time a function calls itself it must be nearer, in some sense to a solution.
- ❖ There must be a decision criterion for stopping the process or computation.

Example:-

### Recursive function for factorial:-

```
fact(n)=1           if n=0
fact(n)=n*fact(n-1) if n>0
```

### Program 42 factorial using recursion method

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

BY:- PRINCE KUMAR SINGH

## A NEW GENERATION OF COMPUTER

```
#include<conio.h>

main()
{
    long int n,m;

    clrscr();

    printf("\nEnter value of n=");

    scanf("%ld",&n);

    m=fact(n);

    printf("\n\nFactorial of number=%ld=%ld",n,m);

    getch();
}

fact(long int n1)
{
    if(n1==0)
        return 1;
    else
        return n1*fact(n1-1);
}
```

### Program 43 Series of factorial using recursion method

```
#include<stdio.h>

#include<conio.h>

main()
{
    long int n,m,i;

    clrscr();
```

BY:- PRINCE KUMAR SINGH

## A NEW GENERATION OF COMPUTER

```
printf("\nEnter value of n=");  
  
scanf("%ld",&n);  
  
for(i=0;i<=n;i++)  
{  
    m=fact(i);  
  
    printf("\n\nFactorial of number=%ld=%ld\n",i,m);  
}  
  
getch();  
  
}  
  
fact(long int n1)  
{  
    if(n1==0)  
        return 1;  
    else  
        return n1*fact(n1-1);  
}
```

### Recursive function for Fibonacci series:-

```
fib(n)=0           if    n=0  
fib(n)=1           if    n=1  
fib(n)=fib(n-1)+fib(n-2) if    n>1
```

### Program 44 Series of fibonacci using recursion method

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

BY:- PRINCE KUMAR SINGH

## A NEW GENERATION OF COMPUTER

```
long int n,m,i;

clrscr();

printf("\nEnter value of n=");

scanf("%ld",&n);

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

for(i=0;i<=n;i++)

{

m=fib(i);

printf("%ld\t",m);

}

getch();

}

fib(long int n1)

{

if(n1==0 || n1==1)

return 1;

else

return fib(n1-1)+fib(n1-2);

}
```

### Tower of Hanoi Using Recursion Method:-

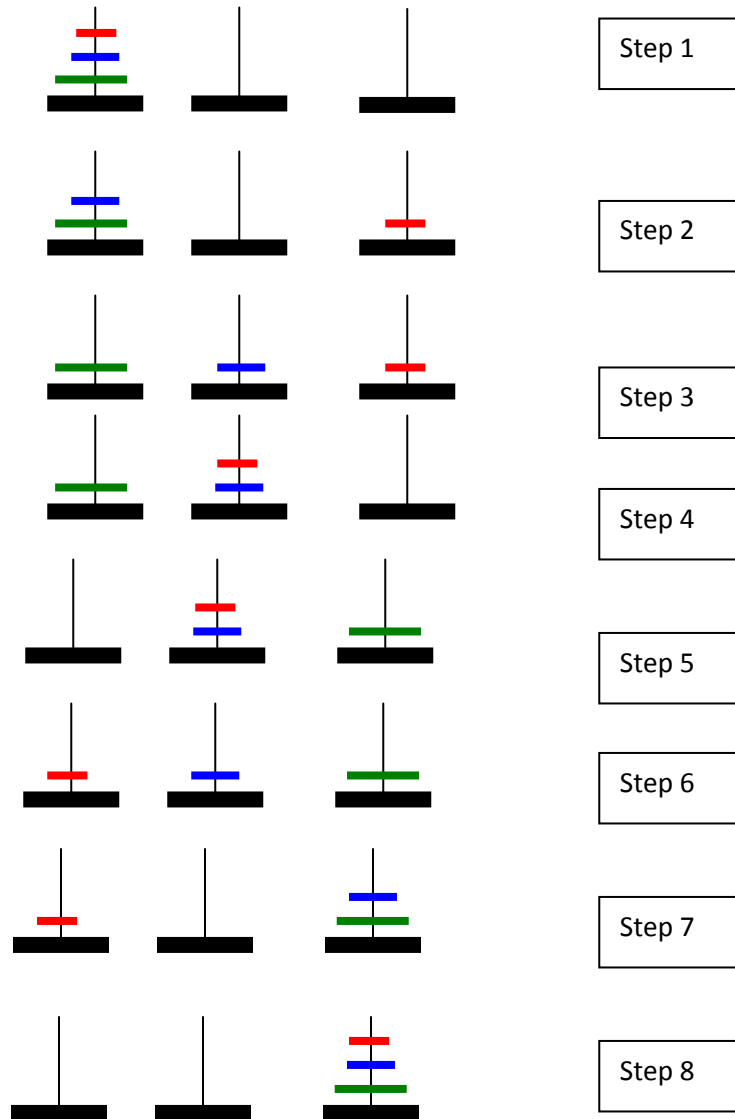
Tower of Hanoi is a historical problem, which can be easily expressed using recursion. There are n disks of decreasing size stacked on one needle, and two other empty needles, It is required to stack all disks onto a second needle in the decreasing order of size. Third needle can be used, as temporary storage. The movement of disks must confirm to the following rules.

- ❖ Only one disk may be moved at a time.
- ❖ A disk can be moved from any needle to any other.
- ❖ At no time ,A larger disks rests upon a smaller one.

**BY:- PRINCE KUMAR SINGH**

# A NEW GENERATION OF COMPUTER

Let number of disks  $n=3$



BY:- PRINCE KUMAR SINGH



# A NEW GENERATION OF COMPUTER

## Program 45 Solving of tower of hanoi using recursion method

```
#include<stdio.h>

#include<conio.h>

main()

{

    int n;

    char Source='A',Middle='B',Target='C';

    void hanoi(int,char,char,char);

    clrscr();

    printf("\nEnter Number of Disks=");

    scanf("%d",&n);

    printf("\nTower of Hanoi with Disk=%d",n);

    hanoi(n,Source,Middle,Target);

    getch();

}

void hanoi(int n1,char left,char mid,char right)

{

    if(n1!=0)

    {
```

BY:- PRINCE KUMAR SINGH

# A NEW GENERATION OF COMPUTER

```
hanoi(n1-1, left,right,mid);  
  
printf("\nMove Disk=%d\tFrom\t%c\tTo\t%c",n1,left,right);  
  
hanoi(n1-1,mid,left,right);  
  
}  
  
}
```

## Array:-

Collection of similar data types element is called array.

## Types of array:-

1:-Single dimensional array.

2:-Double Dimensional Array.

## Syntax:-

### Single dimensional array

<Data\_types> <Array\_Name>[Size];

## Example:-

```
int m[10]={4,7,1,9,2,2,2,2,2,3};
```

4	7	1	9	2	2	2	2	2	3
---	---	---	---	---	---	---	---	---	---

m[0] m[1] m[2] m[3] m[4] m[5] m[6]  
m[7] m[8] m[9]

### Double dimensional array

BY:- PRINCE KUMAR SINGH

# A NEW GENERATION OF COMPUTER

<Data\_types> <Array\_Name>[Size1][Size2];

Or

<Data\_types> <Array\_Name>[Row][Column];

int m[3][3]={{2,5,9},{8,9,4},{8,6,5}};

Row Column

m[0][0]	m[0][1]	m[0][2]
m[1][0]	m[1][1]	m[1][2]
m[2][0]	m[2][1]	m[2][2]

2	5	9
8	9	4

8	6	5
---	---	---

## Example 46 of Single Dimensional Array:-

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

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

```
main()
```

```
{
```

```
int m[12],i;
```

BY:- PRINCE KUMAR SINGH

## A NEW GENERATION OF COMPUTER

```
clrscr();

printf("\nEnter elements of Single Dimensional arrays=\n");

for(i=0;i<=11;i++)

scanf("%d",&m[i]);

printf("\nElements of Single dimensional arrays=\n");

for(i=0;i<=11;i++)

printf("%d\t",m[i]);

getch();

}
```

### Example 47 Sum of Single Dimensional Array:-

```
#include<stdio.h>

#include<conio.h>

main()

{

int m[12],i,s=0;

clrscr();

printf("\nEnter elements of Single Dimensional arrays=\n");

for(i=0;i<=11;i++)

scanf("%d",&m[i]);

printf("\nElements of Single dimensional arrays=\n");

for(i=0;i<=11;i++)

{

printf("%d\t",m[i]);

s=s+m[i];

}
```

BY:- PRINCE KUMAR SINGH

## A NEW GENERATION OF COMPUTER

```
printf("\nSum Of Arrays Elements=%d",s);  
  
getch();  
  
}
```

### Example 48 Double Dimensional Array(Matrix):-

```
#include<stdio.h>  
  
#include<conio.h>  
  
main()  
{  
  
    int m[4][4],row,col;  
  
    clrscr();  
  
    printf("\nEnter elements of Double Dimensional arrays=\n");  
  
    for(row=0;row<=3;row++)  
  
        for(col=0;col<=3;col++)  
  
            scanf("%d",&m[row][col]);  
  
    printf("\nElements of Double dimensional arrays=\n\n");  
  
    for(row=0;row<=3;row++)  
  
    {  
  
        for(col=0;col<=3;col++)  
  
            printf("[%d]\t",m[row][col]);  
  
            printf("\n");  
  
        }  
  
    getch();  
  
}
```

### Example 49 Transpose Matrix:-

BY:- PRINCE KUMAR SINGH

## A NEW GENERATION OF COMPUTER

```
#include<stdio.h>

#include<conio.h>

main()

{

    int m[4][4],row,col;

    clrscr();

    printf("\nEnter elements of Double Dimensional arrays=\n");

    for(row=0;row<=3;row++)

        for(col=0;col<=3;col++)

            scanf("%d",&m[row][col]);

    printf("\nElements of Double dimensional arrays=\n\n");

    for(row=0;row<=3;row++)

    {

        for(col=0;col<=3;col++)

            printf("[%d]\t",m[row][col]);

        printf("\n");

    }

    printf("\nElements of Transpose Matrix=\n\n");

    for(row=0;row<=3;row++)

    {

        for(col=0;col<=3;col++)

            printf("[%d]\t",m[col][row]);

        printf("\n");

    }

    getch();

}
```

BY:- PRINCE KUMAR SINGH

# A NEW GENERATION OF COMPUTER

## Example 50 Sum of two Matrix:-

```
#include<stdio.h>

#include<conio.h>

main()

{

    int m[3][3],m1[3][3],m2[3][3],row,col;

    clrscr();

    printf("\nEnter elements of Matrix1=\n");

    for(row=0;row<=2;row++)

    for(col=0;col<=2;col++)

    scanf("%d",&m[row][col]);

    printf("\nEnter elements of Matrix2=\n");

    for(row=0;row<=2;row++)

    for(col=0;col<=2;col++)

    scanf("%d",&m1[row][col]);

    printf("\nElements of Matrix1=\n\n");

    for(row=0;row<=2;row++)

    {

        for(col=0;col<=2;col++)

        printf("[%d]\t",m[row][col]);

        printf("\n");

    }

    printf("\nElements of Matrix2=\n\n");

    for(row=0;row<=2;row++)

    {

        for(col=0;col<=2;col++)
```

**BY:- PRINCE KUMAR SINGH**

## A NEW GENERATION OF COMPUTER

```
printf("[%d]\t",m1[row][col]);

printf("\n");

}

for(row=0;row<=2;row++)

{

for(col=0;col<=2;col++)

m2[row][col]=m[row][col]+m1[row][col];

}

printf("\nSum of Above Two of Matrix=\n\n");

for(row=0;row<=2;row++)

{

for(col=0;col<=2;col++)

printf("[%d]\t",m2[row][col]);

printf("\n");

}

getch();

}
```

### Sorting:-

It is a technique for ordering elements either ascending order or descending order.

- ❖ Selection Sort.
- ❖ Bubble Sort.
- ❖ Insertion Sort.
- ❖ Heap Sort.
- ❖ Quick Sort.
- ❖ Radix Sort/Bucket Sort.
- ❖ Merge Sort.

### Selection Sort:-

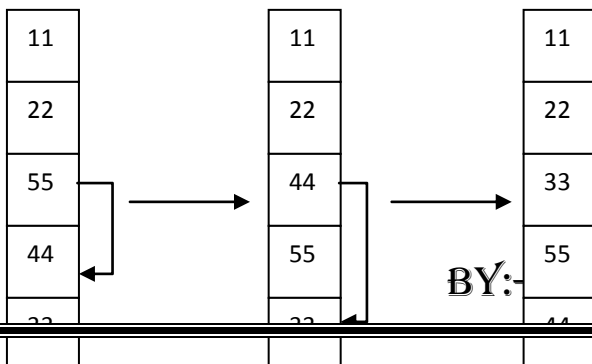
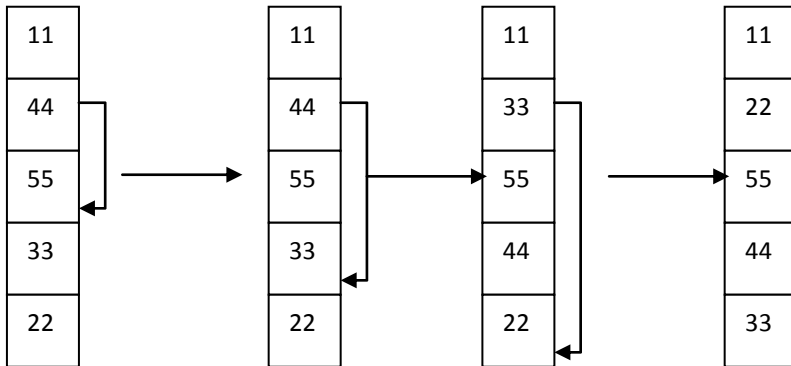
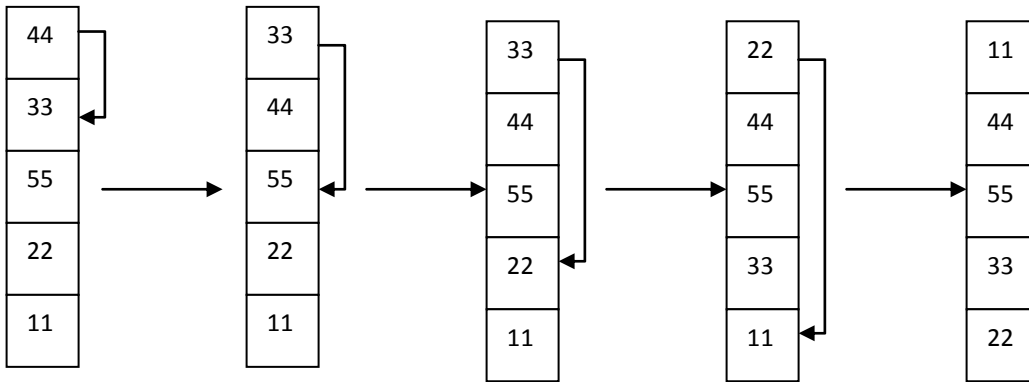
Sort the following unordered elements in either ascending order or descending order.

44, 33,55,22,11

BY:- PRINCE KUMAR SINGH

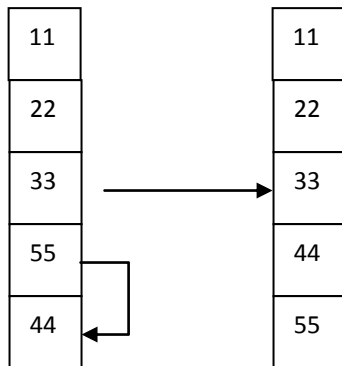


## A NEW GENERATION OF COMPUTER



BY: VINCE KUMAR SINGH

# A NEW GENERATION OF COMPUTER



## Example 51:-

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

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

```
void main()
```

BY:- PRINCE KUMAR SINGH

## A NEW GENERATION OF COMPUTER

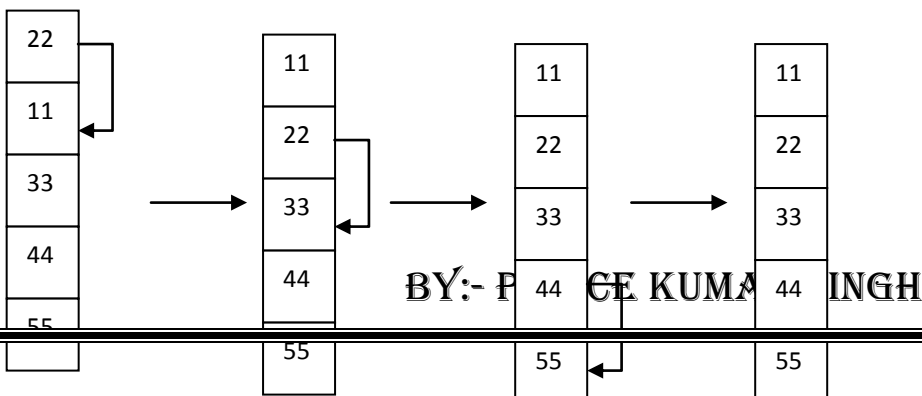
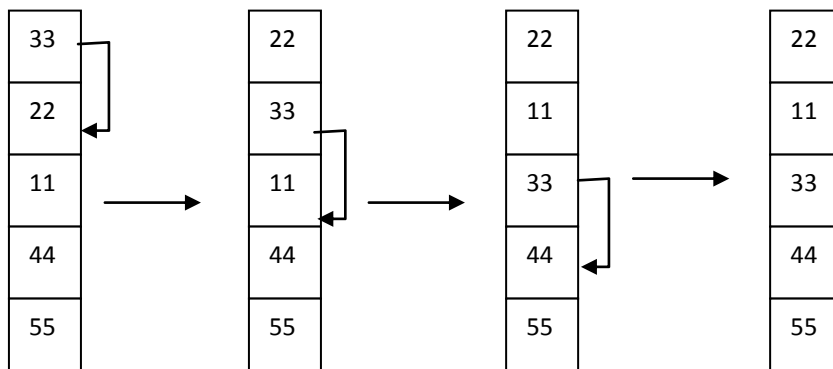
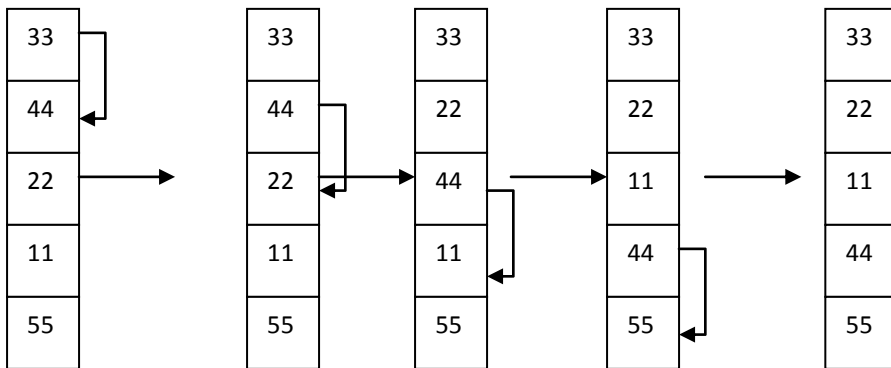
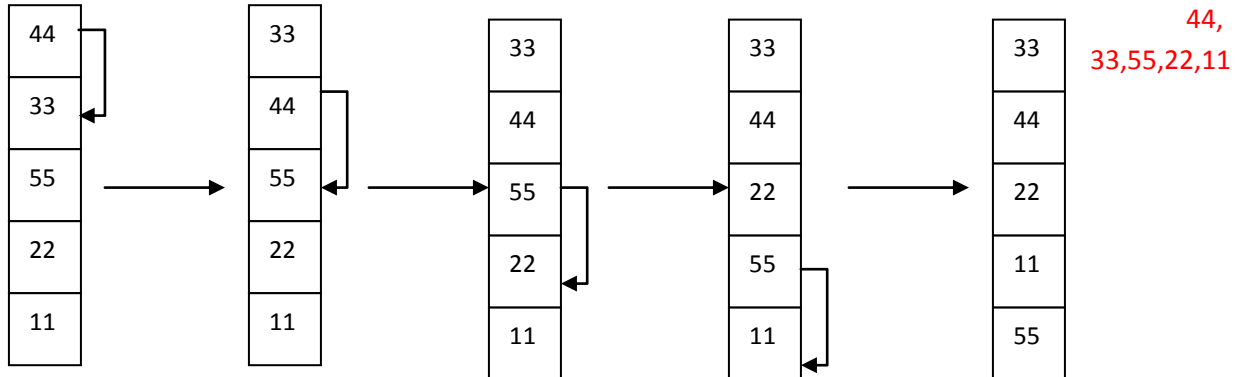
```
{  
int a[10],i,j;  
  
clrscr();  
  
printf("\nEnter Unordered Elements of Arrays= ");  
  
for(i=0;i<=9;i++)  
  
scanf("%d",&a[i]);  
  
printf("\nUnOrdered Elements of Arrays=\n ");  
  
for(i=0;i<=9;i++)  
  
printf("%d\t",a[i]);  
  
printf("\nOrdered Elements of Arrays=\n ");  
  
for(i=0;i<=9;i++)  
  
{  
  
for(j=i+1;j<=9;j++)  
  
{  
  
if(a[i]>a[j])  
  
{  
  
int t;  
  
t=a[i];  
  
a[i]=a[j];  
  
a[j]=t;  
  
}  
  
}  
  
}  
  
for(i=0;i<=9;i++)  
  
printf("%d\t",a[i]);  
  
getch();  
  
}
```

BY:- PRINCE KUMAR SINGH

# A NEW GENERATION OF COMPUTER

## Bubble Sort:-

Sort the following unordered elements in either ascending order or descending order.



BY:- P. GE KUMAR INGH

# A NEW GENERATION OF COMPUTER

BY:- PRINCE KUMAR SINGH

# A NEW GENERATION OF COMPUTER

## Example 52:-

```
#include<stdio.h>

#include<conio.h>

main()

{

int a[10],i,j;

clrscr();

printf("\nEnter Unordred Elements of Arrays= ");

for(i=0;i<=9;i++)

scanf("%d",&a[i]);

printf("\nUnOrdred Elements of Arrays=\n ");
```

**BY:- PRINCE KUMAR SINGH**

## A NEW GENERATION OF COMPUTER

```
for(i=0;i<=9;i++)  
printf("%d\t",a[i]);  
printf("\nOrdered Elements of Arrays=\n ");
```

```
for(i=0;i<=9;i++)  
{  
for(j=0;j<=9-i;j++)  
{  
if(a[j]>a[j+1])  
{  
int t;  
t=a[j];  
a[j]=a[j+1];  
a[j+1]=t;  
}  
}  
}
```

```
for(i=0;i<=9;i++)  
printf("%d\t",a[i]);  
getch();  
}
```

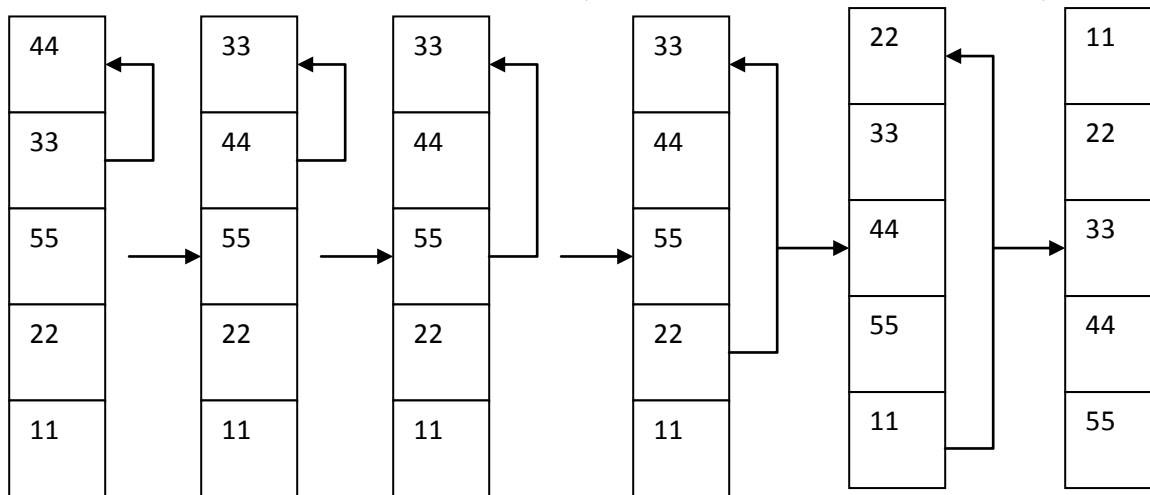
### Insertion Sort:-

Sort the following unordered elements in either ascending order or descending order.

44, 33, 55, 22, 11

BY:- PRINCE KUMAR SINGH

## A NEW GENERATION OF COMPUTER



### Example 53:-

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

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

```
main()
```

```
{
```

```
int a[10],i,j,k,t;
```

```
clrscr();
```

```
printf("\nEnter Unordred Elements of Arrays= ");
```

**BY:- PRINCE KUMAR SINGH**



## A NEW GENERATION OF COMPUTER

```
for(i=0;i<=9;i++)
scanf("%d",&a[i]);

printf("\nUnOrdred Elements of Arrays=\n ");

for(i=0;i<=9;i++)

printf("%d\t",a[i]);

printf("\nOrdred Elements of Arrays=\n ");

for(i=0;i<=9;i++)

{

t=a[i];

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

{

if(t<a[j])

{

for(k=i;k>=j;k--)

a[k]=a[k-1];

a[j]=t;

break;

}

}

}

for(i=0;i<=9;i++)

printf("%d\t",a[i]);

getch();

}
```

### Heap Sort:-

The elements of the heap tree are represented by an array. The root will be the largest elements of the heap tree. Since it is maintained in array, so the largest value should be the last element of array.

BY:- PRINCE KUMAR SINGH

# A NEW GENERATION OF COMPUTER

## Steps for Heap Sort:-

Step 1:- Replace the root with last node of the heap.

Step 2:- Keep the last node at the proper position, means do the delete operation in heap tree

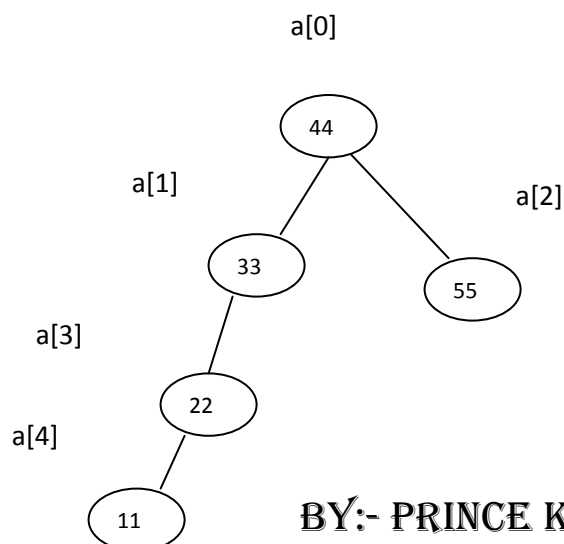
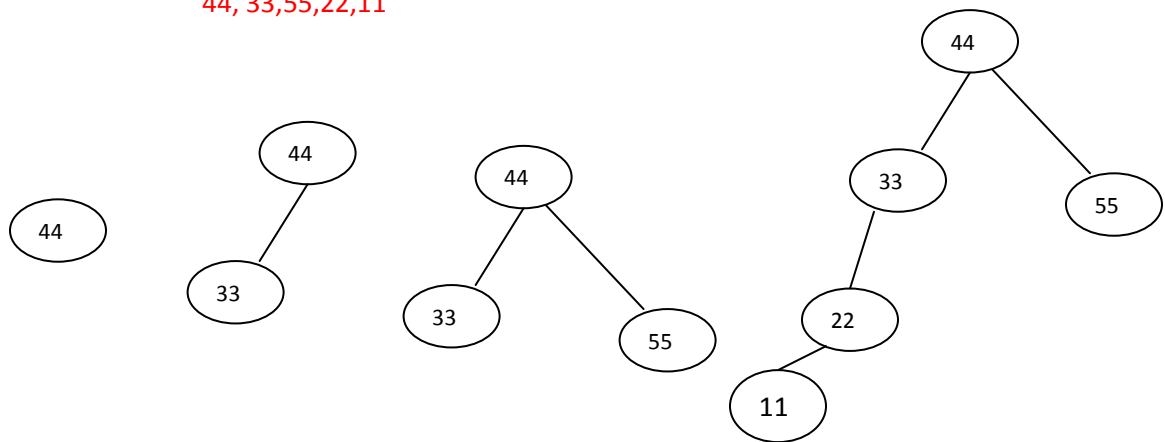
but here deleted node is root

## Example:-

Sort the following unordered elements in either ascending order or descending order.

44, 33, 55, 22, 11

Step1:-

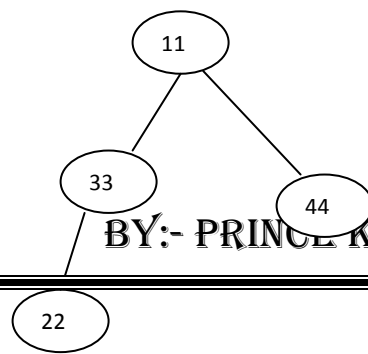
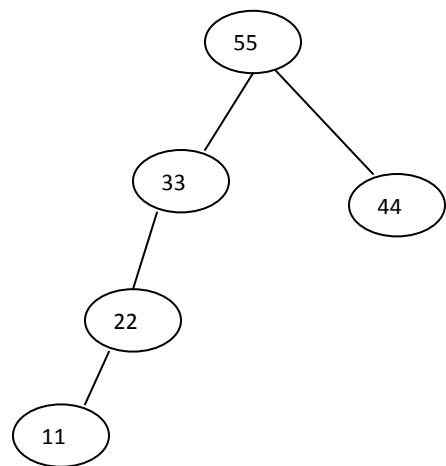


BY:- PRINCE KUMAR SINGH

44	33	55	22	11
----	----	----	----	----

# A NEW GENERATION OF COMPUTER

Step2:- Make Heap Tree of above binary tree



BY:- PRINCE KUMAR SINGH

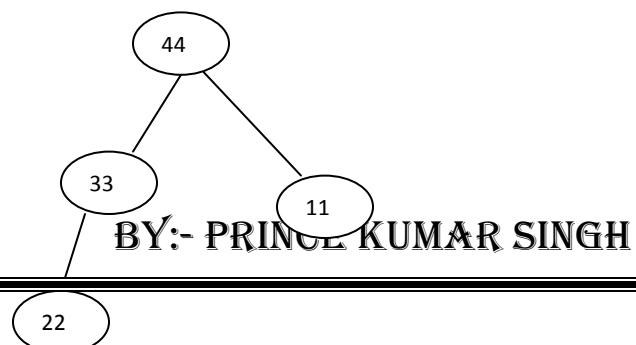
# A NEW GENERATION OF COMPUTER

55	33	44	22	11
11	33	44	22	55

11 replace with 55

11	33	44	22	55
----	----	----	----	----

Step3:- Make Heap  
Tree of above tree

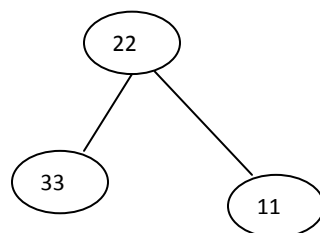


# A NEW GENERATION OF COMPUTER

44	33	11	22	55
----	----	----	----	----

22 replace with 44

22	33	11	44	55
----	----	----	----	----



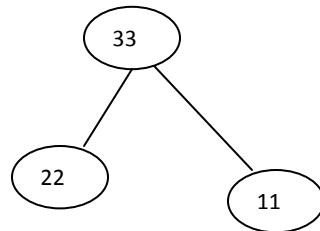
22	33	11	44	55
----	----	----	----	----

BY:- PRINCE KUMAR SINGH

# A NEW GENERATION OF COMPUTER

Step4:- Make Heap Tree of above tree

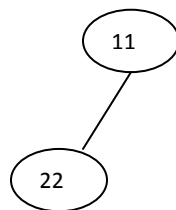
22	33	11	44	55
----	----	----	----	----



33	22	11	44	55
----	----	----	----	----

11 replace with 33

11	22	33	44	55
----	----	----	----	----

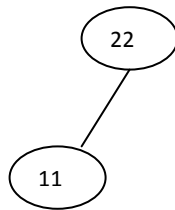


11	22	33	44	55
----	----	----	----	----

BY:- PRINCE KUMAR SINGH

# A NEW GENERATION OF COMPUTER

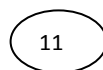
Step5:- Make Heap Tree of above tree



22	11	33	44	55
----	----	----	----	----

22 replace 11

11	22	33	44	55
----	----	----	----	----



11	22	33	44	55
----	----	----	----	----

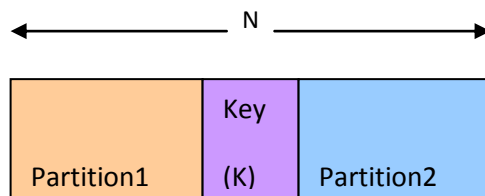
BY:- PRINCE KUMAR SINGH

# A NEW GENERATION OF COMPUTER

## Quick Sort/Partition Exchange technique:-

It is one of the most popular sorting techniques. It was developed by **C.A.R. Hoare**. The quick sort algorithm works by partitioning the array to be sorted and each partition is in turn sorted recursively. In partition, one of the array elements is chosen as key (**pivot**) elements. This key value can be the first element of an array. That is, if  $a$  is an array then  $\text{key} = a[0]$  and rest of the array elements are grouped into two partitions such that:-

- ❖ One partition contains elements smaller than the key value.
- ❖ Another partition contains elements larger than the key value.



Elements < Key

Elements > Key

## Rule:-

- ❖ All the elements on the left side of pivot should be smaller or equal to pivot.
- ❖ All the elements on the right side of pivot should be greater to pivot.

Similarly, we choose the pivot for dividing the sub lists until there are 2 or more elements in the list.

**Example:-** Sort the following elements by using quick sort technique.

44	33	55	22	11
----	----	----	----	----

BY:- PRINCE KUMAR SINGH



## A NEW GENERATION OF COMPUTER

44	33	55	22	11
----	----	----	----	----

Key(Pivot) Value

11 is smaller than 44 we interchange 44 with 11 (Right To Left Move & Search  
Value<44)

11	33	55	22	44
----	----	----	----	----

Now we start from 11 will be from left to right.

The first element greater than 44 is 55. So interchange it with key.

11	33	44	22	55
----	----	----	----	----

Now the comparison will start from 55 and will be from right to left

The first element Smaller than 44 is 22. So interchange it with key.

The first element less than 44 is 22. So interchange it with key.

11	33	22	44	55
----	----	----	----	----

Now the comparison will start from 22 and will be from left to right.

The first element Smaller than 44 is 33. So interchange it with key.

11	44	22	33	55
----	----	----	----	----

BY:- PRINCE KUMAR SINGH

## A NEW GENERATION OF COMPUTER

Now the comparison will start from 33 and will be from Left to Right

The first element smaller than 44 is 22. So interchange it with key.

11	22	44	33	55
----	----	----	----	----

Now the comparison will start from 22 and will be from Right to Left

The first element smaller than 44 is 33. So interchange it with key.

The first element smaller than 44 is 33. So interchange it with key.

11	22	33	44	55
----	----	----	----	----

### Merge Sorting technique:-

If there are two sorted lists of array then process of combining these sorted lists into sorted order is called merging. There are two approaches for merge sorting.

- ❖ First approach.
- ❖ Second approach.

### First approach:-

### Example:-

$m1[4] = \{4, 2, 8, 1\}$

$m2[4] = \{5, 8, 11, 9\}$

Using any sorting technique on both arrays elements. Then we combine both.

$m1[4] = \{4, 2, 8, 1\}$

By using selection sort  $m1[4] = \{1, 2, 4, 8\}$

$m2[4] = \{5, 8, 11, 9\}$  By using bubble sort

$m2[4] = \{5, 8, 9, 11\}$

$m3[8] = \{1, 2, 4, 5, 8, 8, 9, 11\}$

after merging

BY:- PRINCE KUMAR SINGH

# A NEW GENERATION OF COMPUTER

## Second approach:-

$m[10]=\{3,2,7,6,9,55,44,33,23,27\}$

3	2	7	6	9	55	44	33	23	27
---	---	---	---	---	----	----	----	----	----



3	2	7	6	9	55	44	33	23	27
---	---	---	---	---	----	----	----	----	----

After Sorting

2	3	6	7	9	55	33	44	23	27
---	---	---	---	---	----	----	----	----	----



After Sorting

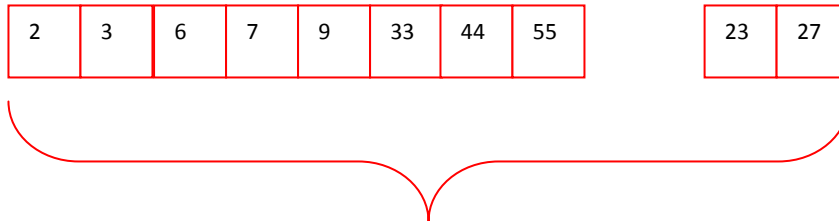
2	3	6	7	9	33	44	55	23	27
---	---	---	---	---	----	----	----	----	----



BY:- PRINCE KUMAR SINGH

# A NEW GENERATION OF COMPUTER

After Sorting



After Sorting

2	3	6	7	9	23	27	33	44	55
---	---	---	---	---	----	----	----	----	----

**Example 54:-**

```
#include<stdio.h>

main()
{
    int a1[5],a2[6],a3[11],i,j,i1,j1;

    printf("\nEnter Unordered Elements of Arrays1= ");

    for(i=0;i<=4;i++)

        scanf("%d",&a1[i]);
```

**BY:- PRINCE KUMAR SINGH**

## A NEW GENERATION OF COMPUTER

```
printf("\nUnOrdered Elements of Arrays1=\n ");
```

```
for(i=0;i<=4;i++)
```

```
printf("%d\t",a1[i]);
```

```
printf("\nEnter Unordered Elements of Arrays2= ");
```

```
for(i=0;i<=5;i++)
```

```
scanf("%d",&a2[i]);
```

```
printf("\nUnOrdered Elements of Arrays2=\n ");
```

```
for(i=0;i<=5;i++)
```

```
printf("%d\t",a2[i]);
```

```
printf("\nMerging of Array elements=\n ");
```

```
for(i=0,i1=0;i<=4;i++,i1++)
```

```
{
```

```
    a3[i1]=a1[i];
```

```
}
```

```
for(j=0,i1=6;j<6;j++,i1++)
```

```
{
```

```
    a3[i1]=a2[j];
```

```
}
```

```
for(i1=0;i1<=10;i1++)
```

```
printf("%d\t",a3[i1]);
```

```
printf("\nOrdered Elements of Arrays1=\n ");
```

```
for(i1=0;i1<=10;i1++)
```

```
{
```

```
    for(j1=i1+1;j1<=10;j1++)
```

```
{
```

```
        if(a3[i1]>a3[j1])
```

```
        {
```

BY:- PRINCE KUMAR SINGH

## A NEW GENERATION OF COMPUTER

```
int t;  
  
t=a3[i1];  
  
a3[i1]=a3[j1];  
  
a3[j1]=t;  
  
}  
  
}  
  
}  
  
for(i1=0;i1<=10;i1++)  
  
printf("%d\t",a3[i1]);  
  
}
```

### Radix/Bucket Sorting technique:-

This sorting technique based on the logic of alphabetical order process.

Example:-

Sort the following elements either ascending order or descending order.

44	33	55	22	11
----	----	----	----	----

Pass1:-

	[0]	[1]	[2]	[3]	[4]	a[5]
44					44	
33				33		
55						55
22			22			
11		11				

BY:- PRINCE KUMAR SINGH

# A NEW GENERATION OF COMPUTER

11, 22,33,44,55

Example:-

Sort the following elements either ascending order or descending

944	633	355	722	811
-----	-----	-----	-----	-----

Pass1

	a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]
944					944					
633				633						
355						355				
722			722							
811		811								

811,722,633,944,355

Pass2

	a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]
811		811								
722			722							
633				633						
944					944					
355						355				

# A NEW GENERATION OF COMPUTER

811,722,633,944,355

Pass3

a[0]   a[1]   a[2]   a[3]   a[4]   a[5]   a[6]   a[7]   a[8]   a[9]

811									811	
722								722		
633							633			
944										944
355				355						

355,633,722,811,944

Introduction of String:-

BY:- PRINCE KUMAR SINGH



# A NEW GENERATION OF COMPUTER

## String:- Important

It is sequence of characters enclosed within double quote.

### Example:-1

“VARANASI” Consist of 9 characters

‘V’,‘A’,‘R’,‘A’,‘N’,‘A’,‘S’,‘I’,‘\0’ → Null Character

### Example:-2

V		A		R		A		N		A		S		I	‘\0’
---	--	---	--	---	--	---	--	---	--	---	--	---	--	---	------

“V A R A N A S I” Consist of 16 characters

Note:- String terminated by null character ( ‘\0’ ).

## Syntax:-

Data\_Types <String\_name>[Size];

### Example:-3

char name[12];

### Example:-4

“9”=Two Bytes ‘9’=One Byte

BY:- PRINCE KUMAR SINGH

# A NEW GENERATION OF COMPUTER

"a" =Two Bytes 'a'=One Byte

## Inputation and Displaying of strings:-

### Methode First 55:-

```
#include<stdio.h>

#include<string.h>

#include<conio.h>

main()

{

char str[10];

clrscr();

printf("\nEnter Any String=");

scanf("%s",&str);

printf("\nInput String=%s",str);

getch();

}
```

### Methode Second 56:-

```
#include<stdio.h>

#include<string.h>

#include<conio.h>

main()

{

char str[10];

clrscr();

printf("\nEnter Any String=");
```

BY:- PRINCE KUMAR SINGH

## A NEW GENERATION OF COMPUTER

```
gets(str);  
printf("\nInput String=");  
puts(str);  
getch();  
}
```

### String Functions:-

#### a:-strlen(String):-

It return length of string.

#### Example 57

```
#include<stdio.h>  
#include<string.h>  
#include<conio.h>  
main()  
{  
char str[10];  
int m;  
clrscr();  
printf("\nEnter Any String=");  
gets(str);  
printf("\nInput String=");  
puts(str);  
m=strlen(str);  
printf("\nLength of string=%s=%d",str,m);  
getch();  
}
```

BY:- PRINCE KUMAR SINGH

# A NEW GENERATION OF COMPUTER

## b:-strrev(String):-

It reverses string.

### Example 58

```
#include<stdio.h>

#include<string.h>

#include<conio.h>

main()

{

char str[10];

clrscr();

printf("\nEnter Any String=");

gets(str);

printf("\nInput String=");

puts(str);

printf("\nReverse String=%s",strrev(str));

getch();

}
```

## C:-strcat(String1,String2):-

It is used for joining string2 into string1.

### Example 59

```
#include<stdio.h>

#include<string.h>

#include<conio.h>

main()

{

char str1[20],str2[10];
```

BY:- PRINCE KUMAR SINGH

## A NEW GENERATION OF COMPUTER

```
clrscr();

printf("\nEnter Any String1=");

gets(str1);

printf("\nEnter Any String2=");

gets(str2);

printf("\nInput String1=");

puts(str1);

printf("\nInput String2=");

puts(str2);

strcat(str1,str2);

printf("\nConcatenated String=%s",str1);

getch();

}
```

### D:-strcpy(String1,String2):-

It copy string2 into string1.

### Example 60

```
#include<stdio.h>

#include<string.h>

#include<conio.h>

main()

{

char str1[10],str2[20];

clrscr();

printf("\nEnter Any String1=");

gets(str1);
```

BY:- PRINCE KUMAR SINGH

## A NEW GENERATION OF COMPUTER

```
printf("\nEnter Any String2=");  
  
gets(str2);  
  
printf("\nInput String1=");  
  
puts(str1);  
  
printf("\nInput String2=");  
  
puts(str2);  
  
strcpy(str1,str2);  
  
printf("\nString1 after copy String=%s",str1);  
  
getch();  
  
}
```

### E:-strcmp(String1,String2):-

This function compares string1, string2, and return following values.

- ❖ Less than zero (It means string1 is less than string2)
- ❖ Equal zero (It means both string1 and string2 are identical)
- ❖ Greater than zero (It means string1 is greater than string2)

### Example 61

```
#include<stdio.h>  
  
#include<string.h>  
  
#include<conio.h>  
  
main()  
{  
  
char str1[10],str2[20];  
  
int m;  
  
clrscr();  
  
printf("\nEnter Any String1=");  
  
gets(str1);
```

BY:- PRINCE KUMAR SINGH

## A NEW GENERATION OF COMPUTER

```
printf("\nEnter Any String2=");  
  
gets(str2);  
  
printf("\nInput String1=");  
  
puts(str1);  
  
printf("\nInput String2=");  
  
puts(str2);  
  
m=strcmp(str1,str2);  
  
if(m==0)  
  
printf("\nBoth strings are identical");  
  
else  
  
if(m>0)  
  
printf("\nstring1 is greater than string2");  
  
else  
  
    printf("\nstring1 is Less than string2");  
  
getch();  
  
}
```

### Example 62(Program for palindrome)

```
#include<stdio.h>  
  
#include<string.h>  
  
#include<conio.h>  
  
main()  
  
{  
  
char str1[10],str2[15];  
  
int m;  
  
clrscr();  
  
printf("\nEnter Any String1=");  
  
gets(str1);
```

**BY:- PRINCE KUMAR SINGH**

## A NEW GENERATION OF COMPUTER

```
printf("\nInput String1=");  
  
puts(str1);  
  
strcpy(str2,str1);  
  
strrev(str2);  
  
m=strcmp(str1,str2);  
  
if(m==0)  
  
printf("\nString Is palindrome");  
  
else  
  
printf("\nstring1 is Not palindrome");  
  
getch();  
  
}
```

### Structures and Unions:- Gauranteed

Structure combine logically related data items into a single unit. The data items enclosed within a structure are known as members and they can be same as different data types. It is defined by users as per their requirements.

#### Syntax:-

```
struct <structure_Name>  
  
{  
  
Data_Type1 member1;  
  
Data_Type2 member2;  
  
Data_Type3 member3;  
  
...  
  
...  
  
...}
```

BY:- PRINCE KUMAR SINGH



## A NEW GENERATION OF COMPUTER

><object1>,<object2>...;

### Example 63

```
#include<stdio.h>

#include<string.h>

#include<conio.h>

struct st1

{

int eno;

char name[15];

char job[15];

};

struct st2

{

char fname[15];

};

main()

{

struct st1 obj1;

struct st2 obj2;

clrscr();

printf("\nEnter Employee Number=");

scanf("%d",&obj1.eno);

printf("\nEnter Employee Name=");

scanf("%s",&obj1.name);

printf("\nEnter Employee Father Name=");

scanf("%s",&obj2.fname);
```

**BY:- PRINCE KUMAR SINGH**

## A NEW GENERATION OF COMPUTER

```
printf("\nEnter Employee Job=");  
  
scanf("%s",&obj1.job);  
  
printf("\nEno=%d\nName=%s\nFather=%s\nJob=%s\n",obj1.eno,obj1.name,obj2.fname,obj1.job);  
  
printf("\n\nSize of structure1=%d\tSize of structure2=%d",sizeof(obj1),sizeof(obj2));  
  
getch();  
  
}
```

### Union:-

It is the collection of different types of data item. It is different from structure i.e. in the structure all the members use individual memory locations where in the case of union all the members are pointing to the same memory locations. The size of memory will be according to the size of that data members whose memory size is the biggest among all.

### Syntax:-

```
union <Union_Name>  
{  
  
Data_Type1 member1;  
  
Data_Type2 member2;  
  
Data_Type3 member3;  
  
...  
  
...  
  
...  
  
}<object1>,<object2>...;
```

### Example 64

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

BY:- PRINCE KUMAR SINGH

## A NEW GENERATION OF COMPUTER

```
#include<conio.h>

struct st1

{

int eno;

char name[25];

char job[15];

};

struct st2

{

char fname[15];

};

union un1

{

struct st1 obj1;

struct st2 obj2;

};

main()

{

union un1 obj3;

clrscr();

printf("\nEnter Employee Number=");

scanf("%d",&obj3.obj1.eno);

printf("\nEnter Employee Name=");

scanf("%s",&obj3.obj1.name);

printf("\nEnter Employee Father Name=");

scanf("%s",&obj3.obj2.fname);

printf("\nEnter Employee Job=");
```

**BY:- PRINCE KUMAR SINGH**

## A NEW GENERATION OF COMPUTER

```
scanf("%s",&obj3.obj1.job);

printf("\nEno=%d\nName=%s\nFather=%s\nJob=%s\n",obj3.obj1.eno,obj3.obj1.name,obj3.obj2.fname,
obj3.obj1.job);

printf("\nSize of Union=%d",sizeof(obj3));

getch();

}
```

### Pointer:-      Most Important

It is variable which point the address of variable. Ponter variable represented by using indirection operator (\*).

### Syntax:-

<Data\_type> <variable1>,<variable2>,<variable3>...,\*ptr1, \*ptr2,\*ptr3...;

### Where :-

ptr1=address of variable1 (&variable1);

ptr2=address of variable2 (&variable2);

ptr3=address of variable3 (&variable3);

...

...

...

\*ptr1=Value at address variable1;

\*ptr2=Value at address variable2;

\*ptr3=Value at address variable3;

...

...

...

### Example:-

```
int a=2,b=6,*p1,*p2;
```

BY:- PRINCE KUMAR SINGH

## A NEW GENERATION OF COMPUTER

p1=&a;

p2=&b;

\*p1=value at address a;

\*p2=value at address b;

### Example 65:-

```
#include<stdio.h>

#include<conio.h>

main()
{
int a=2,b=6,*p1,*p2;

p1=&a;

p2=&b;

printf("\nValue at address a=%d",a);

printf("\nValue at address b=%d",b);

printf("\nValue at address a=%d",*p1);

printf("\nValue at address b=%d",*p2);

printf("\nAddress of a=%u",p1);

printf("\nAddress of b=%u",p2);

getch();

}
```

### Example 66 (Generating series using pointer of single dimensional Array):-

```
#include<stdio.h>

#include<conio.h>

main()
{
```

BY:- PRINCE KUMAR SINGH

## A NEW GENERATION OF COMPUTER

```
int m[10],i;

clrscr();

printf("\nEnter Elements of Array=");

for(i=0;i<=9;i++)

scanf("%d",&m[i]);

printf("\n Elements of Array=\n");

for(i=0;i<=9;i++)

printf("%d\t",*(m+i));

printf("\n Address of Array=\n");

for(i=0;i<=9;i++)

printf("%u\t",&m[i]);

getch();

}
```

### Example 67 (Generating Matrix using pointer of double dimensional Array):-

```
#include<stdio.h>

#include<conio.h>

main()

{

int m[4][4],row,col;

clrscr();

printf("\nEnter elements of Double Dimensional arrays=\n");

for(row=0;row<=3;row++)

for(col=0;col<=3;col++)

scanf("%d",&m[row][col]);

printf("\nElements of Double dimensional arrays=\n\n");
```

**BY:- PRINCE KUMAR SINGH**

## A NEW GENERATION OF COMPUTER

```
for(row=0;row<=3;row++)  
{  
    for(col=0;col<=3;col++)  
        printf("[%d]\t",*(m+row)+col));  
    printf("\n");  
}  
  
printf("\nAddress of Double dimensional arrays=\n\n");  
  
for(row=0;row<=3;row++)  
{  
    for(col=0;col<=3;col++)  
        printf("[%u]\t",&m[row][col]);  
    printf("\n");  
}  
  
getch();  
}
```

### Use of Pointer:-

- ❖ A pointer allows a function or a program to access a variable outside the preview of the function or program.
- ❖ Use of pointer increases makes the program execution faster.
- ❖ Using pointers, arrays and structures can be handled in more efficient way.
- ❖ Without pointers, it will be impossible to create complex data structure such as linked lists, trees and graphs

### Arithmetic Operations Using Pointers:-

#### Example 68

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

BY:- PRINCE KUMAR SINGH

## A NEW GENERATION OF COMPUTER

```
#include<conio.h>

main()

{

float a,b,sum,pro,div,minus,*ptr1,*ptr2;

ptr1=&a;

ptr2=&b;

clrscr();

printf("\nEnter values of a and b=");

scanf("%f%f",&a,&b);

printf("\nValue of a=%f\tValue of b=%f",a,b);

sum=(*ptr1)+(*ptr2);

minus=(*ptr1)-(*ptr2);

pro=(*ptr1)*(*ptr2);

div=(*ptr1)/(*ptr2);

printf("\nSum=%f",sum);

printf("\nDifference=%f",minus);

printf("\nProduct=%f",pro);

printf("\nDivision=%f",div);

printf("\nAddress of a=%u",ptr1);

printf("\nAddress of b=%u",ptr2);

getch();

}
```

### Pointer To Pointer:-

A variable that hold an address of a variable that in turn holds an address of another variable. This type of variable will be known as pointer to pointer.

**BY:- PRINCE KUMAR SINGH**



# A NEW GENERATION OF COMPUTER

Example:-

```
int a=2,*ptr1,**ptr2;

ptr1=&a;

ptr2=&ptr1;
```

## Example 69

```
#include<stdio.h>

#include<conio.h>

main()

{

int a=2,*ptr1,**ptr2;

ptr1=&a;

ptr2=&ptr1;

clrscr();

printf("\nValue at address ptr1=%d",*ptr1);

printf("\nValue at address ptr2=%d",*ptr2);

printf("\nAddress of First pointer=%u",ptr1);

printf("\nAddress of Second pointer=%u",ptr2);

getch();

}
```

## Why C is called Structured Programming Language:-

It consists of multiple modules that have a set of functions of related types.

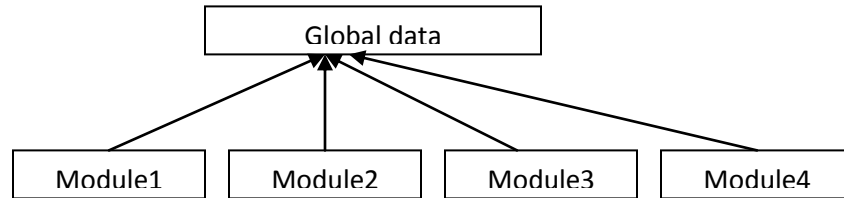
It includes following features.

- It emphasis on algorithm rather than data.
- Programs are divided into individual procedures that perform discrete task.
- Procedures are independent of each other as far as possible.
- Procedures have their own local data and processing logic.
- Parameter passing facility between the procedures for information communication.
- Support for modular programming.

**BY:- PRINCE KUMAR SINGH**

## A NEW GENERATION OF COMPUTER

- Maintenance of a large s/w system is tedious and costly.



BY:- PRINCE KUMAR SINGH