



NAME : T.Prashanth Reddy STD. : B.Tech SEC. : IT ROLL NO. : 12BIT0077

ITE 101 (Problem solving using C):

Computer: commonly oriented machine particularly used for trade and educational research

Von-Neumann Architecture of Computer

machine language
0 - signals do not flow in the circuit (off state)
1 - signal flows in the circuit (ON state)

Hardware: physical parts of computer

Software : set of instructions

Firmware: combination of hardware and software (eg: mobiles, washing machines, Bluetooth devices)

Software

System software

Application software

Application software: software which we use and create

C → compiler

founder of C: Dennis Ritchie

Compiler is a system software which converts high level language to machine language

- Interpreter is also a system software just like compiler
- Interpreter compiles line by line but compiler compiles the whole program in one go.
- BASIC (Beginners allpurpose symbolic instruction language)
- is an interpreter

Assembler:-

It is a system software which converts mnemonic codes to machine language. Assembly level language requires to remember a lot of instructions.

INTRODUCTION TO 'C'

features:

- Robust language
- Fast and efficient
- Structured language
- Portable (capable of running in any system)
- Extendability
- middle level language (supports high level language & middle level language)

Structure of code:

- Documentation section
- Link section
- Definition section
- Global declaration section
- main()
 - variable declaration part
 - Executable part;

→ Sub programs

{
variable declaration part;
executable part;

}

Documentation section:

non-executable part
statement comments

// Sample program (single line comment)

/*

----- *!
→ (multiple line comment)

Link section:

#include <stdio.h>

#include <conio.h>

→ preprocessor (does some process before execution
of program . It includes b header file)

Some header files:

1) stdio.h → standard input output
(supporting input & functions of 'c')
[scanf () → i/p f'n
printf () → o/p f'n]

2) conio.h: console input output
(defines o/p for monitor)
{ clrscr();
getch();
getche(); }
(key typed is reflected on the screen
and then the screen vanishes)

return 0; (conio.h is not required)

→ 3) stdlib.h

exit()
malloc()
calloc()
free()
fflush()
atof()

| system("Pause")

↳ displays (press any key to
continue)

4) math.h

5) string.h

strcpy() } strstr()
strcat() | strcmp()
strlent()

→ Definition section:

used for #define. It is used to define a constant value

e.g. #define Pi 3.14

→ Global declaration:

variables declared outside the main and other functions is called global variable

main() → is a function not a keyword

II Sample program

#include <stdio.h>

#include <conio.h>

main()

{ printf("This is C programming"); }

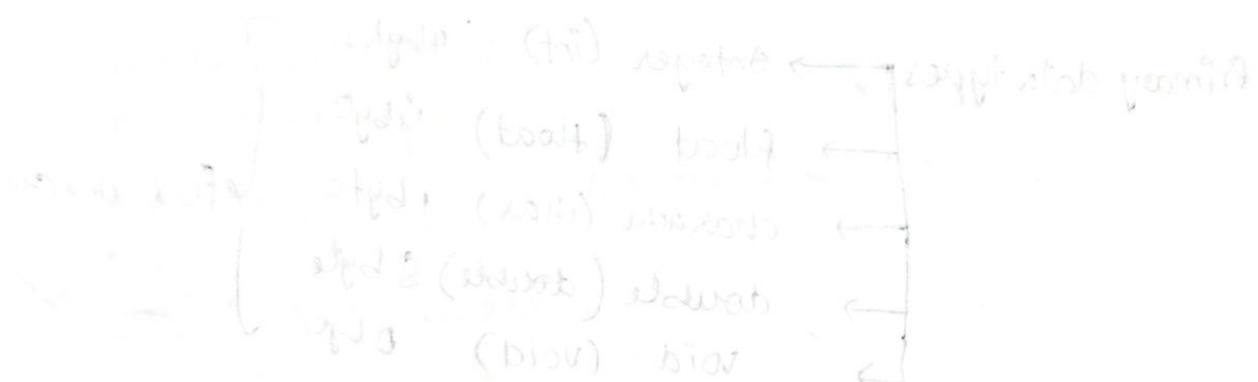
getch();

Y

Escaping Sequences & applications:

In

It



Flowchart showing the execution process:

Source code → Preprocessor → Compiler → Linker → Output file

Sample.c
(compiler)

obj

cout << "Hello world";
exit(0);

Linker → (system software)
(add all library functions)

exe (stored in main memory)

ctrl+F10 (enables loader)

System software

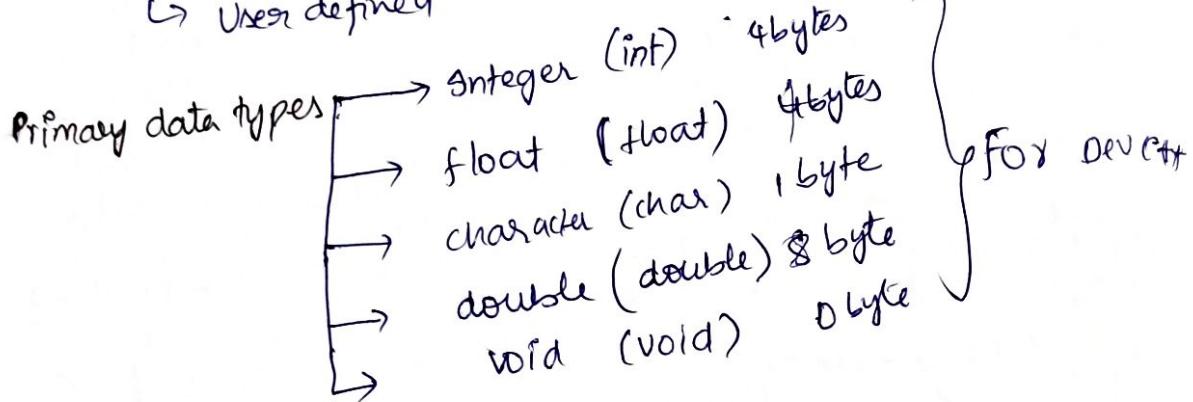
Run

DVC - 32 bit compiler

TurboC - 16 bit compiler

Data types:

- Primary / fundamental
- Secondary / derived
- User defined



Derived

- data Types
- Arrays
 - functions
 - structures
 - Pointers

User defined

- `typedef`
- `enum (enumerated data types)`

0,1 → 1 bit

8 bits → 1 byte

4 bits → nibble

16 bits / 2 bytes → 1 word

range of numbers that "int"

can store = -2^{n-1} to $2^{n-1} - 1$

= -2^{31} to $2^{31} - 1$

To find minimum range and max range ~

`int -MIN;`

`int MAX;`

→ When the program executes successfully then the value of 0 is returned when not, then (-1) is returned

→ // sum of two numbers

```
#include < stdio.h >
#include < conio.h >
main() {
    int a=10, b=20, c;
    c = a+b;
    printf ("The sum of two numbers is %d + %d = %d",
           a, b, c);
    getch();
}
```

address operator

%d = for integer format

%f = for float value

%c = for char value

%lf = for double value

%s = for string "

→ sum of two numbers

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

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

```
main()
```

```
{ printf int a, b, c;
```

printf ("Enter the value of a and b");

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

locates the address of 'a' already allocated

c = a+b;

printf ("%d + %d = %d", a, b, c);

getch();

}

Rules for declaring variables / identifiers:

- The first character of a variable should be a character or a underscore (-)
- Except the first character, the rest of the character can be a digit or a letter or an underscore (-)
- No keywords can be used as variables
- No whitespaces or blank spaces can be used
- No other special characters are not allowed
- The variables can be of any length but only first letters are considered by the compiler.

NOTE:- constants

Keywords

variables

operators

special characters ~~are called as 'tokens'~~
~~(also called as 'lexical elements')~~

→ WAP to exchange the values of two variables using a third variable and without using a third variable

a) using a third variable:

```
#include<stdio.h>
#include<conio.h>
main()
{
    int a,b,c;
    printf("Enter the values of 'a' and 'b'");
    scanf("%d %d", &a, &b);
    printf("The values of 'a' and 'b' before swapping = %d and %d", a, b);
}
```

~~a = a + b~~
~~b = a~~
~~c = b~~
~~b = c~~

printf("The values of 'a' and 'b' after swapping is %d and %d", a, b);

b) without using a third variable :-

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

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

```
main()
```

```
{  
    int a, b;
```

```
    printf("enter the values of a and b.");
```

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

```
    printf("The values of 'a' and 'b' before swapping = ");
```

Swap

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

```
b = a-b;
```

```
a = a-b;
```

H.W

- which function is used to update remainder value after dividing a float

ans. ~~fmod~~. fmod(a,b) (supported by header file math.h)

- which special symbol can be used in variable list

A. '_' (underscore)

- By default compiler treat a real number

A. as double - to get more accuracy

- What is the output of the following

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

```
int x=40;
```

```
int main(){
```

```
    int x=20;
```

```
    printf("%d", x);
```

```
}
```

- A. Here $x=40$ is a global variable (since it is not in main function)
 $int x=20;$ is a local variable

→ local variable has higher priority than global variable

→ Hence output is 20.

14.

Q 5. A float is 4 byte wide and double is 8 byte wide

A.

→ A. True

15.

G 6. size of datatype may vary from one platform to another

1

i A. True (^{for} Dev++ int = 4 byte for other -)

A.

7. Global variables are variables available to all functions.

16.

A. TRUE

8. Does there exist a mechanism by ^{the way of} which global variables is available to some function but not others functions

A.

A. NO.

Scan

17.

F 9. what format specifiers are used to print the values of float and double

A.

A. %f and _____

18.

10. what is preprocessor directive

A.

A. #

19.

11. A preprocessor is a msg from a programmer to the pre-processor

A.

A. TRUE

12. Every 'c' program will contain at least one preprocessor directive.

A.

A. FALSE

B. The preprocessor can trap simple errors like missing declaration, missing comments or mismatch of braces

A.

A. FALSE

14. A pre-processor directive is a msg from a compiler to linker
- A. FALSE
15. One pre-processing is over and the program ^{is} sent for compilation the back ticks are removed from expanded source code
- A. TRUE
16. which header files should be included for use of the functions like malloc(), calloc()
- A. stdlib.h
17. what is the range of "character" data values
- A. -128 to +127
18. every executable program in 'C' must contain
- A. main function
19. A program block is enclosed with pair of
- A. Curly braces { }
20. what is the output of
- main()
{ printf ("1.d", out);
}
- A. Error

Operators in C

1. Arithmetic Operator (+, -, *, /, %)
2. logical (&&, ||, !) True = 1 ; False = 0
3. Relational (>, <, <=, >=, ==, !=)
4. Conditional (?) (Ternary operator)
5. Assignment (=)
6. Increment and decrement operators (++, --) Unary operator
7. Bitwise operator (only for integer values)
(&, |, ^) ($<<$, $>>$, ~)
8. Special operator
(, (comma), sizeof())

→ Sample program

```
#include <stdio.h>
main()
{
    int a,b;
    scanf("%d %d", &a, &b);
    a>b? printf("A is greatest"): printf("B is greatest");
    true
}
```

→ Sample program

```
#include <stdio.h>
main()
{
    int a,b,c;
    Scanf("%d %d %d", &a, &b, &c);
    a>b&&a>c? printf("A"): printf("B");
    printf("C");
}
```

Bitwise op

Expressions:-

An expression is a collection of operators and operands.

- An expression can be prefix, infix (or) postfix. In prefix, the operator precedes the operands. In infix, the operator is in between the operands and the operators and for postfix, the operator follows the operands.

$a+b \rightarrow$ prefix

$a+b \rightarrow$ infix

$a+b \rightarrow$ postfix

Sample program:-

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

```
main()
```

```
{ int m=5, x, y;
```

```
x = m++ ; // x = 5m, m = 6m+1
```

```
printf("%d %d", x, m);
```

```
y = ++m ; // m = 7m+1, y = 7m
```

```
printf("%d %d", y, m);
```

```
}
```

```
z = m++ + ++m // m = 8m+1 ; z = 15m+1
```

```
x = ++x + y-- ; // x = 6x+1, x = 6+713, y = 6y-1
```

OR:

Bitwise operators

and $\vdash \phi$

$$0x = 1$$

negation = \sim → unary operator

left shift = <<

Right shift = >>

c2 a & b

AND

a	b	o/p
0	0	0
1	0	0
0	1	0
1	1	1

$$a=12 \quad ; \quad b=6$$

$$c = 12 \neq 6$$

$$(12)_{16} = (1100)_2$$

$$(6)_{10} = (110)$$

$$(42) \begin{array}{r} 40 \\ + 6 \\ \hline 46 \end{array}$$

$$(0110)_2 = (4)_{10}$$

OR:

OR

a	b	o	p
o	o	o	
o	l	l	
l	o	l	
l	l		

120

$$(12)_{10} = (1100)_2$$

$$(6)_{10} = (110)_2$$

$$(12)_{10} : (6)_{10} = \underline{\underline{1100}} \\ \underline{\underline{0110}} \\ \underline{\underline{1110}}$$

$$C = 12 \quad ; \quad G = 14$$

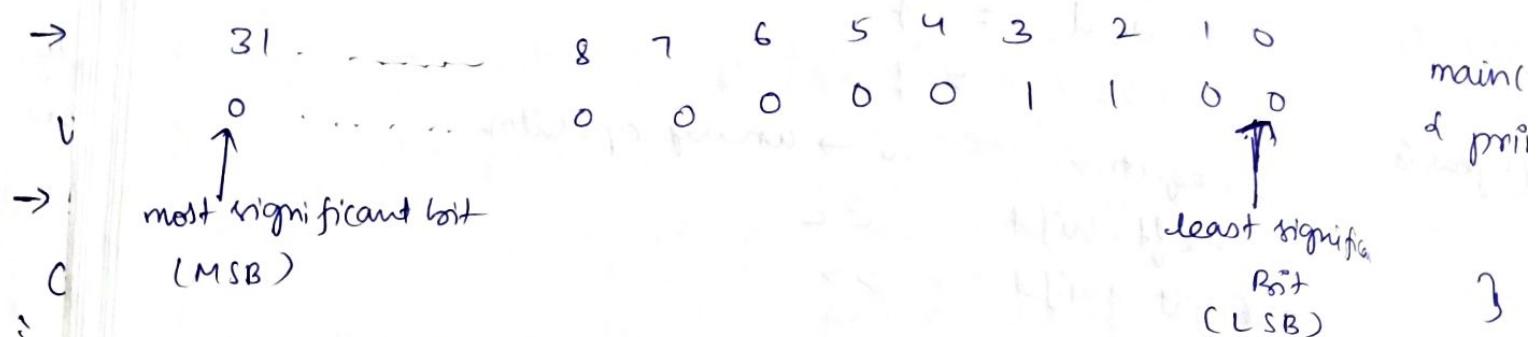
XOR

a	b	o/p
0	0	0
0	1	1
1	0	1
0	0	0

$$\begin{array}{r} 1100 \\ 0110 \\ \hline 1010 \end{array}$$

$$(12)_{10} \wedge (8)_{10} = (10)_{10} = (1010)_2$$

left shift :-



$12 \ll 2$

i $a \ll s \rightarrow$ no. of shift bits $\Rightarrow (12)_0 \rightarrow (11000)_2$

$a \ll s$ output is $[a \times 2^s]$

$$12 \gg 2 = 3$$

$$1100 \xrightarrow{1st \text{ shift}}$$

$$0110 \xrightarrow{2nd \text{ shift}}$$

$a \gg s$ output is $\boxed{\frac{a}{2^s}}$

only 31 shifts are possible

negation:-

$$c = 2^n a$$

\hookrightarrow 2's complement of the output

$$i/p 12 o/p = -13$$

$$i/p 100 o/p = -101$$

Special Operators:-

" , " and "sizeof()"

main()

{ printf ("%d.%d.%d.%d.%d.%d", sizeof(int), sizeof(float),
sizeof(char), sizeof(double), sizeof(void));
}

main()

{

int x=10, y=20, z; // separator

z = x, y, x+y; // operator

printf ("%d", z);

}

→ Comma has lowest precedence amongst all

- a) write a program using conditional operators
- b) To find a given number is odd or even

#include <stdio.h>

#include <conio.h>

main()

{

int a, num;

printf ("Enter the number");

scanf ("%d", &num);

a = num % 2;

(a == 0) ? printf ("number is even") : printf ("number is odd");

getch();

}

2. To find the given year is leap year (or) not

#include <stdio.h>

#include <conio.h>

main()

{ int a, num;

printf ("Enter the number of days in the year");

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

```
a = num % 4;
```

```
(a == 2) ? printf("year is not a leap year") : printf("year  
a leap year");
```

```
getch();
```

```
}
```

iii) To find a number is +ve (or) -ve

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

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

```
main()
```

```
int num;
```

```
printf("Enter the number");
```

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

```
(num > 0) ? printf("number is positive") : printf("number is  
negative");
```

```
getch();
```

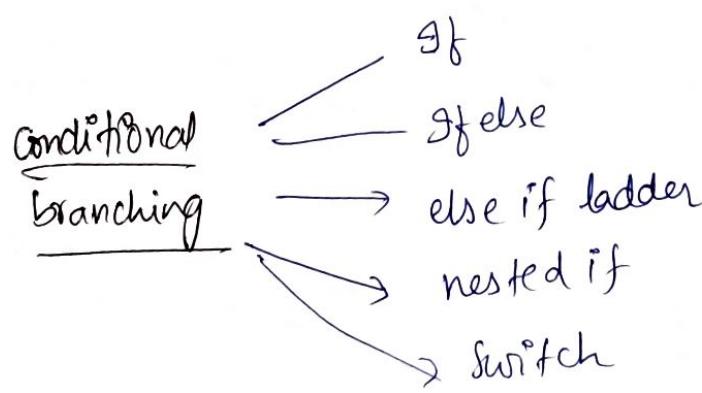
```
}
```

④ To find an input character is vowel or consonant

Q) WAP to perform multiplication operation and division operation without using '*' and '/'.

```
#include < stdio.h>
#include < conio.h>
main()
{
    int num,a,b;
    printf("enter the number");
    scanf("%d", &num);
    printf(" multiplying the number with 4\n");
    a = num<<2;
    printf("%d\n", a);
    printf("dividing the number with 4\n");
    b = num>>2;
    printf("%d\n", b);
    getch();
}
```

Decision making and branching



unconditional
branching — goto

else if ladder:

→ Syntax:

if (condition)

{

 =

 }

else if (condition)

{

 =

 }

 ;

 ;

 ;

 ;

 ;

 ;

 ;

 ;

 ;

 ;

 ;

 ;

 ;

 ;

 ;

 ;

 ;

 ;

 ;

 ;

 ;

→ sample Program:

main()

{

 char op;

 int a, b, c;

 printf ("press + or - or * or /");

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

 if (op == '+')

{

 c = a + b;

 printf ("sum is %d", c);

}

 else if (op == '-')

{

 c = a - b;

 printf ("difference is %d", c);

}

 else if (op == '*')

{

 c = a * b;

 printf ("product is %d", c);

}

 else if (op == '/')

{

 c = a / b;

 printf ("quotient is %d", c);

}

 else if (op == '%')

{

 c = a % b;

 printf ("remainder is %d", c);

}

 else

 printf ("entered wrong operator");

}

Switch Case:

Syntax:

switch (variable(s) expression)

{
 case label;
 —
 case label2;
 —
 case label3;
 :
 :
 default:
 —
}

→ Sample Program:

main()

{ int op;

int a,b,c;

printf("Press 1. Add 2. Subtract
3. Multiplication 4. Division 5. Modulus\n");

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

switch(op)

{ case 1: c = a+b;

printf("sum is %d", c);

break;

case 2: c = a-b;

printf("Difference is %d", c);

break;

case 3: c = a*b;

printf("Product is %d", c);

break;

case 4: c = a/b;

printf("Quotient is %d", c);

break;

case 5:

c = a%b;

printf("Remainder is %d", c);

break

default:

printf("Entered wrong choice");

}

Unconditional branching:

Syntax:

→ goto label;

↳ → main()

d

→ : int i=1;

G loop: printf ("%d\n", i);

i = i+1;

i if (i <= 10)

goto loop;

y

loop:

==

goto loop;

==

goto loop;

==

loop

==

==

} backward jump

==

loop

==

} forward jump

WAP to check whether the given no is divisible by 7 or not.

```
#include < stdio.h >
#include < conio.h >
main()
{
    int num, a, b;
    printf ("Enter the number");
    scanf ("%d", &num);
    a = num % 7;
    if (a == 0)
        printf ("number is divisible by 7");
    else
        printf ("number is not divisible by 7");
    getch();
}
```

2. WAP to check whether a candidate is eligible for voting or not

3. WAP to find the roots of an quad eqn"

4. WAP using goto to find the sum of first 10 numbers

continuation

See After

3 papers

ARRAYS

→ Arrays are ~~defined~~ derived data types which holds data items of same data type

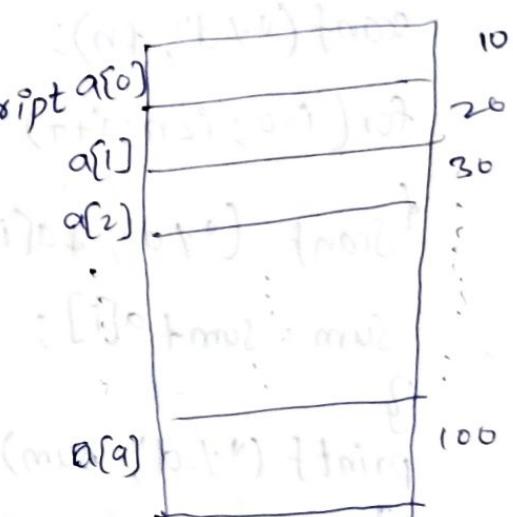
Syntax:

datatype variable [size]

e.g.

int a[10];

↳ index (or) subscript



→ Static Array:

→ Dynamic Array:

pointers

malloc() } <stdlib.h>
calloc()

main()

{ int i, a[100], n;

scanf("%d", &n);

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

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

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

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

}

Q) Write a program to find sum of any array values

```
#include <stdio.h>  
main()  
{  
    int i, a[n], n, sum=0;  
    scanf("%d", &n);  
    for(i=0; i<n; i++)  
    {  
        scanf("%d", &a[i]);  
        sum = sum + a[i];  
    }  
    printf("%d", sum);  
}
```

Q) Scan for ' n ' values and get an element 'x' and check whether the element is present in array or not. if present print 'element is present' let the msg be printed only once.

main()

{

→ This is called "searching" & method is called

"linear (or) Sequential Searching".

main()

```
{ int i, a[50], flag=0, x, n;  
    scanf("%d", &n);  
    scanf("%d", &x);
```

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

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

```
    printf("Enter the element to be searched");
```

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

```

for(i=0; i<n; i++)
{
    if(a[i]==x)
        flag=1;
    break;
}
if(flag==1)
    printf("element found");
else
    printf("element not found");
}

```

→ To find duplicates.

```

main()
{
    int i, a[50], flag=0, x, n, c=0;
    scanf("%d", &n);
    for(i=0; i<n; i++)
        scanf("%d", &a[i]);
    printf("enter the element to be searched");
    scanf("%d", &x);
    for(i=0; i<n; i++)
    {
        if(a[i]==x)
            flag=1;
        c++;
    }
}

```

```
if (flag == 1)
    printf ("Element found %d times (%d)\n");
else
    printf ("Element not found");
}
```

→ H.W
WAP to find the largest & smallest element in

an array

SORTING:-

Arranging of data items either in ascending order (or) descending order

Bubble sort:

5

3

1

4

2

no. of passes

```

for( i=0 ; i<n-1 ; i++ )
  {
    for( j=0 ; j<n-i ; j++ )
      {
        if( a[j] > a[j+1] )
          {
            temp = a[j];
            a[j] = a[j+1];
            a[j+1] = temp;
          }
      }
  }
}

```

Bubble sort:

i=0

j=0

5
3

1

4

2

j=1

3

5
1

4

2

j=2

3

1

4

2

j=3

3

1

4

2

j=4

3

1

4

2

5

i=1

j=0

3
1

4

2

5

j=1

1

3
4

2

5

j=2

1

3

4

2

5

j=3

1

3

2

4

5

i=2

j=0

1)
3)

2

4

5

j=1

1)
3)

2)

4)

5)

j=2

1)
2)

3

4

5

array to an

(+i; i<5; i++)

{
new val, print
for loop
}

(+i; i<5; i++)

{
new val, print
for loop
}

(+i; i<5; i++)

{
new val, print
for loop
}

Decision

while

for

do-while

while

syntax

ed

initializ

while(

{

=

For:

Syntax

For(ini

d

3

i=3

j=0

1)
2)

3

4

5

j=1

1)
2)

3

4

5

{[i]; i>5; i])

4

5

2

1

3

5

4

2

1

3

5

4

2

1

3

5

4

2

1

3

5

4

2

1

3

5

4

2

1

3

5

4

2

1

3

5

4

2

1

3

→

sy

Decision making and looping:

while } entry controlled loops
for

do-while } exit controlled loops

while

syntax

{
initialisation

while (condn)

{
=

increment }

sample Program:

```
main()
{ int i=1, n, sum=0;
  scanf("%d", &n);
  while (i<=n)
  {
    sum = sum + i;
    i = i+1;
  }
  printf("%d", sum);
}
```

For:

Syntax:

for (initialisation; condition; increment/decrement)

{
=

sample Program:

```
main()
{
  int i, sum=0, n;
  scanf("%d", &n);
  for (i=1; i<=n; i++)
  {
    sum = sum + i;
  }
  printf("%d", sum);
}
```

→ do-while

syntax:

initialisation;

do

{
=

increment;

} while (condition);

→ main()

{ int i=1, sum=0, n;

scanf("%d", &n);

do

{ sum = sum + i;

i = i+1;

} while (i<=n);

printf("%d", sum);

}

`kblit()` → keyboard hit
↳ comb.h

Sentinel controlled loops:

→ `main()`

{ int i=0;

while(1){

{ printf("i.d", i);

i=i+1;

}

}

} → `main()`

{ int i=0;

while (!kblit())

{ printf("i.d", i),

i=i+1;

}

}

ALGORITHMS

Algorithm - design

Program - implementation

Algorithm: is a mathematical model of any solution

→ Algorithm is the logical representation of a program defined in definite steps.

Properties:

1. Simple
2. Concise and compact
3. Effective
4. Free of ambiguity

Algorithm for sum of two numbers:

- Declare the variables a, b, c
- Read the values of a and b
- Compute $c = a + b$
- Display the value of c
- Algorithms are written using Pseudo code language

Pseudo code: It is a tool that allows adaptation

to any high level language

→ PROCEDURE / PROGRAM / FUNCTION: SUM

130.

W.

```
→      BEGIN
       VAR   A,B,C: INTEGER
       READ  A,B
       C := A+B
       WRITE C
       END    SUM
```

Home-work:

1Q. WAP to find factorial of a given number

2Q. WAP to generate the fibonacci sequence

0, 1, 1, 2, 3, 5, 8, ... natural numbers

3Q. Sum of square of 'n' natural numbers

4Q. Reversing the digit

5Q. Sum of digit

6Q. No. of digits in a given no

7Q. To check the given no is palindrome or not

8Q. To check the given no is Armstrong or not

9Q. To find the sum of +ve and negative no's

until '0' is pressed, then if '0' is pressed then next

10Q. To generate odd no. b/w 100 and 1000

11Q. To generate first 100 even no

12Q. To check the no. given is prime or not

Q. To generate first 50 prime no's

Cat. 1

MARCH

writing algorithms for greatest.

N230

967

```
PROGRAM GREATEST  
BEGIN  
VAR A,B:INTEGER  
READ A,B  
IF A>B THEN  
WRITE "A IS GREATEST"  
ELSE  
WRITE "B IS GREATEST"  
ENDIF  
END GREATEST
```

→ For finding summation (using while)

```
PROGRAM SUMMATION  
BEGIN  
VAR I,N, SUM: INTEGER  
I:=1  
SUM:=0  
READ N  
WHILE I<=N DO  
SUM:= SUM+I  
I:=I+1  
END WHILE  
WRITE "THE SUM OF N NATURAL NOS IS ", SUM  
END SUMMATION
```

writing for:

PROGRAM SUMMATION

BEGIN

VAR

I := 1

SUM := 0

READ N

~~WHILE~~ ~~DO~~

FOR I := 1 TO N INCR BY 1 DO

SUM := SUM + I

END FOR

WRITE "THE SUM OF N NATURAL NO'S IS", SUM

END SUMMATION

using do ... while

PROGRAM

SUMMATION

BEGIN

VAR

I := 1

SUM := 0

READ N

REPEAT

SUM := SUM + I

I := I + 1

UNTIL I . LE N

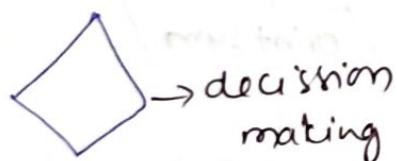
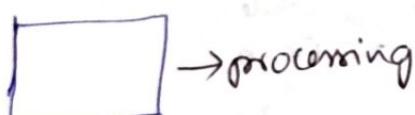
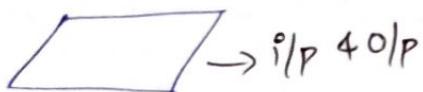
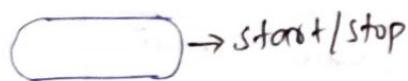
WRITE "THE SUM OF N NATURAL NO'S IS", SUM

END SUMMATION

FLOW CHART

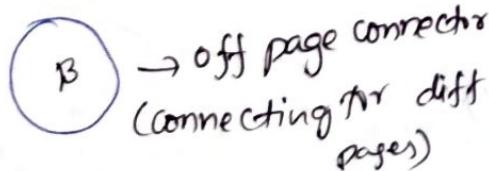
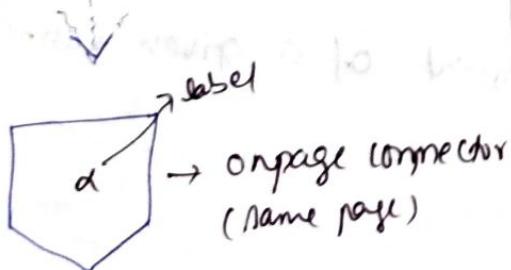
→ Pictorial representation of an algorithm is flow chart

Symbols:

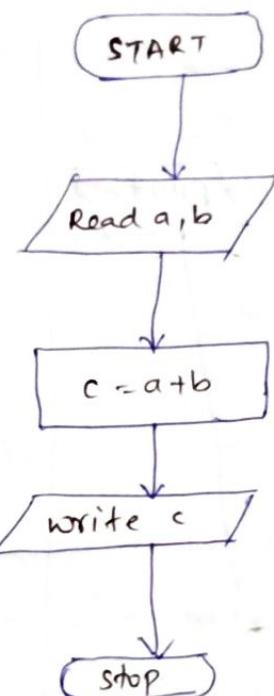


↓
→ data flow

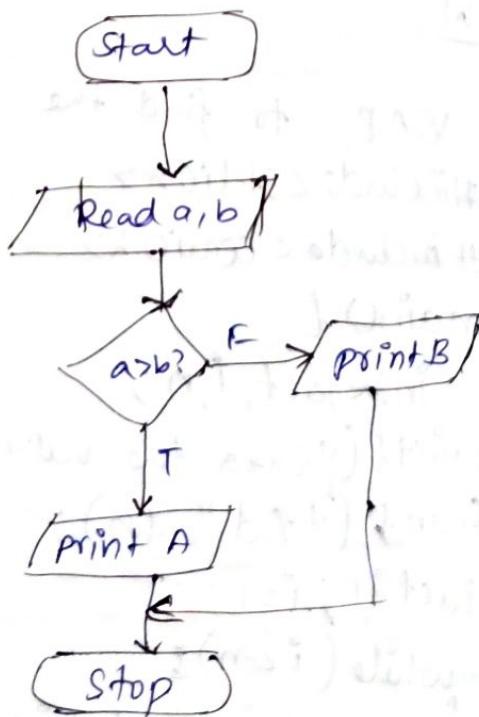
→ control flow



Flow chart for adding two numbers

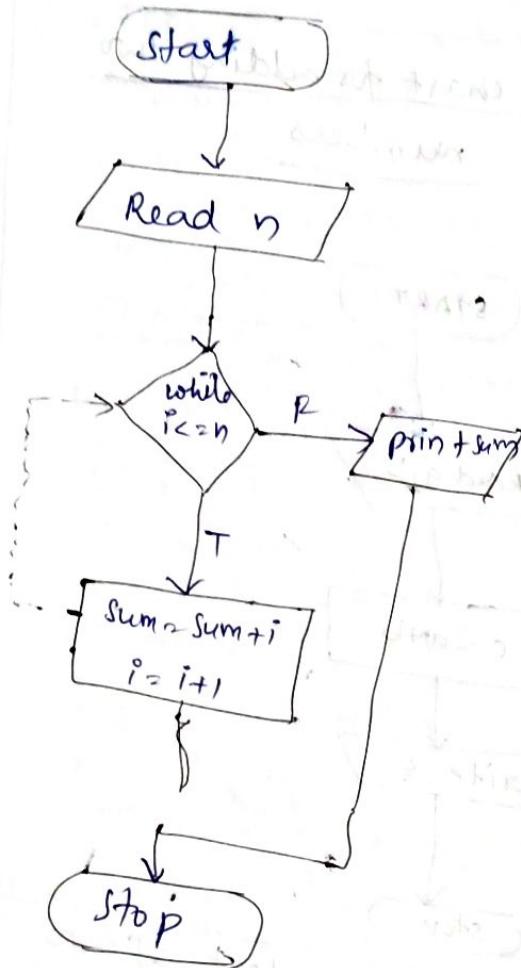


for testing greatest of two
two numbers



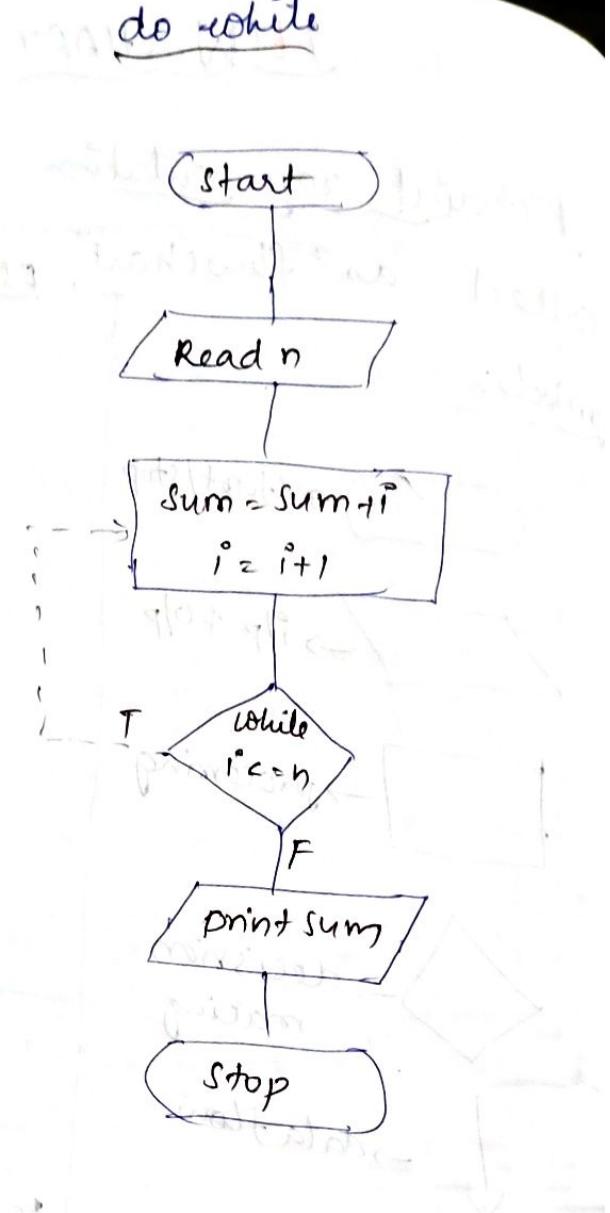
looping flow chart

while



do-while

start



H.W

10. WAP to find the factorial of a given number

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

```
    int fact, i, n;
    printf("Enter the value of n");
    scanf("%d", &n);
    fact = 1; i = 1;
    while (i <= n)
    {
        fact = fact * i;
        i++;
    }
    getch();
}
```

20. WAP to generate the fibonacci sequence
 $0, 1, 1, 2, 3, 5, 8, 13, 21, \dots$

```
#include <stdio.h>
#include <conio.h>
main()
{
    int a=-1, b=1, c, i=0, n;
    printf("enter the value of n");
    scanf("%d", &n);
    while (i<n)
    {
        c=a+b;
        a=b;
        b=c;
        i++;
        printf("%d", c);
    }
    getch();
}
```

30. WAP to find sum of squares of 'n' natural numbers

```
#include <stdio.h>
#include <conio.h>
main()
{
    int sum=0, i=0, n;
    printf("enter the value of n");
    scanf("%d", &n);
    while (i<=n)
    {
        sum = sum + i*i;
        i++;
    }
    printf("%d", sum);
    getch();
}
```

QUESTION

REVERSING THE DIGIT

```
#include <stdio.h>
#include <conio.h>
main()
{
    int m, num, rev=0;
    printf("enter the number: ");
    scanf("%d", &num);
    while(num!=0)
    {
        m = num%10;
        rev = rev*10+m;
        num = num/10;
    }
    printf("The reversed number is %d", rev);
    getch();
}
```

To find sum of digits

```
#include <stdio.h>
#include <conio.h>
main()
{
    int num, sum=0, m;
    printf("enter any number: ");
    scanf("%d", &num);
    while(num!=0)
```

```
{   m = num%10;
```

```
    sum = sum + m;
```

```
    num = num/10;
```

```
}
```

```
printf("sum of digits = %d", sum);
```

```
getch();
```

```
}
```

To find no. of digits:

```
#include <stdio.h>
#include <conio.h>
main()
{
    int num, count=1, sum=0, m;
    printf("Enter any number:");
    scanf("%d", &num);
    while(num!=0)
    {
        num = num/10;
        if(num!=0)
            count = count + 1;
        else
            printf("no. of digits=%d\n", count);
    }
    getch();
}
```

To FIND GIVEN NO' IS A PALINDROME (OR) NOT

```
#include <stdio.h>
#include <conio.h>
main()
{
    int sum=0, m, num, temp;
    printf("Enter the number:");
    scanf("%d", &num);
    temp = num;
    while(num!=0)
    {
        m = num%10;
        num = num/10;
        sum = sum*10+m;
    }
    if(temp==sum)
        printf("given number is a palindrome");
    else
        printf("given number is not a palindrome");
    getch();
}
```

Armstrong number:

Those numbers which sum of its digits to power of number of its digits is equal to that number are known as 'armstrong numbers'.

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

$1634 = 1^4 + 6^4 + 3^4 + 4^4 = 1634$

i) WAP to check whether given number is Armstrong number (for three digits)

```
#include<stdio.h>
#include<conio.h>
main()
{
    int sum=0, m, num, temp;
    printf("Enter the number: ");
    scanf("%d", &num);
    temp = num;
    while (num != 0)
    {
        m = num % 10;
        num = num / 10;
        sum = sum + m * m * m;
    }
    if (temp == sum)
        printf("Given number is Armstrong number");
    else
        printf("Given number is not an Armstrong number");
}
```

Strong number:

Sum of its digits factorial is equal to the number.

$$145 = 1! + 4! + 5!$$

Adam's no.:

11.

Step 1: $11^2 = 121$

Step 2: rev = 121

Step 3: $\sqrt{121} = 11$

10Q. To generate odd numbers b/w 100 and 1000

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

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

```
main()
```

```
{
```

```
int i, n;
```

```
i=100; n=1000;
```

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

```
{
```

```
x = i * 2;
```

```
if (x != 0)
```

```
printf ("%d", i);
```

```
i++;
```

```
}
```

```
getch();
```

```
}
```

To generate first 100 even numbers

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

```
main()
```

```
{ int i,x,n;
```

```
i=1; n=100;
```

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

```
{ x = i%2;
```

```
if(x==0)
```

```
printf("%d",i);
```

```
i++;
```

```
}
```

```
getch();
```

```
}
```

To check whether the given year is leap year or not

Definition of leap year:-

Rule 1: A year is called leap year if it is divisible by 400

For ex: 1600, 2000 etc. leap year while 1500, 1700 are not leap year.

Rule 2:

If year is not divisible by 400 as well as 100 but it is divisible by 4 then that year are also leap year.

For ex: 2004, 2008, 2012, 1012 etc are leap year

Algorithm for finding a leap year

```

IF Year MODULER 400 IS 0
    THEN leap-year
ELSE IF year MODULER 100 IS 0
    THEN not leap-year
ELSE IF year moduler 4 IS 0
    THEN leap-year
ELSE
    not - leap year

```

program using conditional operators:-

```

#include <stdio.h>
#include <conio.h>
main()
{
    int year;
    printf("enter the year");
    scanf("%d", &year);
    if (year % 400 == 0) ? printf("It is a leap year") : (year % 100 == 0) ? printf("It is not a leap year") : (year % 4 == 0) ? printf("It is a leap year") : printf("It is not a leap year");
    getch();
}

```

To check whether the given number is prime number
or not.

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

```
int num, i=2, count=0;
```

```
printf("enter the number\n");
```

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

```
while (i<=num/2)
```

```
{ if (num%i == 0)
```

```
count++;
```

```
i++;
```

```
}
```

```
if (count == 0 && num != 1)
```

```
printf("number is prime");
```

```
else
```

```
printf("number is not a prime number");
```

```
getch();
```

```
y
```

To print prime numbers from 0 to 300.

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

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

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

```
main()
```

```
{ int i, j;
```

```
i = 1;
```

```
while (i < 300) {
```

```
    j = 2;
```

```
    while (j < sqrt(i)) {
```

```
        if (i % j == 0)
```

```
            break;
```

else {

j++;

continue;

y
y

→

if (j > sqrt(i))

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

i++;

y
y

→

getch();

y

else
exit function

<stdio.h> include

<math.h> include

union

int i, j, k, l, m, n, p, q, r, s, t;

float a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t;

char ch1, ch2, ch3, ch4, ch5, ch6, ch7, ch8, ch9, ch10;

double d1, d2, d3, d4, d5, d6, d7, d8, d9, d10;

long l1, l2, l3, l4, l5, l6, l7, l8, l9, l10;

long long ll1, ll2, ll3, ll4, ll5, ll6, ll7, ll8, ll9, ll10;

unsigned int ui1, ui2, ui3, ui4, ui5, ui6, ui7, ui8, ui9, ui10;

unsigned long ul1, ul2, ul3, ul4, ul5, ul6, ul7, ul8, ul9, ul10;

unsigned long long ull1, ull2, ull3, ull4, ull5, ull6, ull7, ull8, ull9, ull10;

float f1, f2, f3, f4, f5, f6, f7, f8, f9, f10;

double df1, df2, df3, df4, df5, df6, df7, df8, df9, df10;

long double ld1, ld2, ld3, ld4, ld5, ld6, ld7, ld8, ld9, ld10;

char ch11, ch12, ch13, ch14, ch15, ch16, ch17, ch18, ch19, ch20;

double dd1, dd2, dd3, dd4, dd5, dd6, dd7, dd8, dd9, dd10;

long double ldd1, ldd2, ldd3, ldd4, ldd5, ldd6, ldd7, ldd8, ldd9, ldd10;

float ff1, ff2, ff3, ff4, ff5, ff6, ff7, ff8, ff9, ff10;

double dff1, dff2, dff3, dff4, dff5, dff6, dff7, dff8, dff9, dff10;

long double ldff1, ldff2, ldff3, ldff4, ldff5, ldff6, ldff7, ldff8, ldff9, ldff10;

char ch21, ch22, ch23, ch24, ch25, ch26, ch27, ch28, ch29, ch30;

double dd21, dd22, dd23, dd24, dd25, dd26, dd27, dd28, dd29, dd30;

long double ldd21, ldd22, ldd23, ldd24, ldd25, ldd26, ldd27, ldd28, ldd29, ldd30;

float ff21, ff22, ff23, ff24, ff25, ff26, ff27, ff28, ff29, ff30;

double dff21, dff22, dff23, dff24, dff25, dff26, dff27, dff28, dff29, dff30;

long double ldff21, ldff22, ldff23, ldff24, ldff25, ldff26, ldff27, ldff28, ldff29, ldff30;

char ch31, ch32, ch33, ch34, ch35, ch36, ch37, ch38, ch39, ch40;

double dd31, dd32, dd33, dd34, dd35, dd36, dd37, dd38, dd39, dd40;

long double ldd31, ldd32, ldd33, ldd34, ldd35, ldd36, ldd37, ldd38, ldd39, ldd40;

float ff31, ff32, ff33, ff34, ff35, ff36, ff37, ff38, ff39, ff40;

double dff31, dff32, dff33, dff34, dff35, dff36, dff37, dff38, dff39, dff40;

long double ldff31, ldff32, ldff33, ldff34, ldff35, ldff36, ldff37, ldff38, ldff39, ldff40;

char ch41, ch42, ch43, ch44, ch45, ch46, ch47, ch48, ch49, ch50;

double dd41, dd42, dd43, dd44, dd45, dd46, dd47, dd48, dd49, dd50;

long double ldd41, ldd42, ldd43, ldd44, ldd45, ldd46, ldd47, ldd48, ldd49, ldd50;

float ff41, ff42, ff43, ff44, ff45, ff46, ff47, ff48, ff49, ff50;

double dff41, dff42, dff43, dff44, dff45, dff46, dff47, dff48, dff49, dff50;

long double ldff41, ldff42, ldff43, ldff44, ldff45, ldff46, ldff47, ldff48, ldff49, ldff50;

char ch51, ch52, ch53, ch54, ch55, ch56, ch57, ch58, ch59, ch60;

double dd51, dd52, dd53, dd54, dd55, dd56, dd57, dd58, dd59, dd60;

long double ldd51, ldd52, ldd53, ldd54, ldd55, ldd56, ldd57, ldd58, ldd59, ldd60;

float ff51, ff52, ff53, ff54, ff55, ff56, ff57, ff58, ff59, ff60;

double dff51, dff52, dff53, dff54, dff55, dff56, dff57, dff58, dff59, dff60;

long double ldff51, ldff52, ldff53, ldff54, ldff55, ldff56, ldff57, ldff58, ldff59, ldff60;

char ch61, ch62, ch63, ch64, ch65, ch66, ch67, ch68, ch69, ch70;

double dd61, dd62, dd63, dd64, dd65, dd66, dd67, dd68, dd69, dd70;

long double ldd61, ldd62, ldd63, ldd64, ldd65, ldd66, ldd67, ldd68, ldd69, ldd70;

float ff61, ff62, ff63, ff64, ff65, ff66, ff67, ff68, ff69, ff70;

double dff61, dff62, dff63, dff64, dff65, dff66, dff67, dff68, dff69, dff70;

long double ldff61, ldff62, ldff63, ldff64, ldff65, ldff66, ldff67, ldff68, ldff69, ldff70;

char ch71, ch72, ch73, ch74, ch75, ch76, ch77, ch78, ch79, ch80;

double dd71, dd72, dd73, dd74, dd75, dd76, dd77, dd78, dd79, dd80;

long double ldd71, ldd72, ldd73, ldd74, ldd75, ldd76, ldd77, ldd78, ldd79, ldd80;

float ff71, ff72, ff73, ff74, ff75, ff76, ff77, ff78, ff79, ff80;

double dff71, dff72, dff73, dff74, dff75, dff76, dff77, dff78, dff79, dff80;

long double ldff71, ldff72, ldff73, ldff74, ldff75, ldff76, ldff77, ldff78, ldff79, ldff80;

char ch81, ch82, ch83, ch84, ch85, ch86, ch87, ch88, ch89, ch90;

double dd81, dd82, dd83, dd84, dd85, dd86, dd87, dd88, dd89, dd90;

long double ldd81, ldd82, ldd83, ldd84, ldd85, ldd86, ldd87, ldd88, ldd89, ldd90;

float ff81, ff82, ff83, ff84, ff85, ff86, ff87, ff88, ff89, ff90;

double dff81, dff82, dff83, dff84, dff85, dff86, dff87, dff88, dff89, dff90;

long double ldff81, ldff82, ldff83, ldff84, ldff85, ldff86, ldff87, ldff88, ldff89, ldff90;

char ch91, ch92, ch93, ch94, ch95, ch96, ch97, ch98, ch99, ch100;

double dd91, dd92, dd93, dd94, dd95, dd96, dd97, dd98, dd99, dd100;

long double ldd91, ldd92, ldd93, ldd94, ldd95, ldd96, ldd97, ldd98, ldd99, ldd100;

float ff91, ff92, ff93, ff94, ff95, ff96, ff97, ff98, ff99, ff100;

double dff91, dff92, dff93, dff94, dff95, dff96, dff97, dff98, dff99, dff100;

long double ldff91, ldff92, ldff93, ldff94, ldff95, ldff96, ldff97, ldff98, ldff99, ldff100;

char ch101, ch102, ch103, ch104, ch105, ch106, ch107, ch108, ch109, ch110;

double dd101, dd102, dd103, dd104, dd105, dd106, dd107, dd108, dd109, dd110;

long double ldd101, ldd102, ldd103, ldd104, ldd105, ldd106, ldd107, ldd108, ldd109, ldd110;

float ff101, ff102, ff103, ff104, ff105, ff106, ff107, ff108, ff109, ff110;

double dff101, dff102, dff103, dff104, dff105, dff106, dff107, dff108, dff109, dff110;

long double ldff101, ldff102, ldff103, ldff104, ldff105, ldff106, ldff107, ldff108, ldff109, ldff110;

char ch111, ch112, ch113, ch114, ch115, ch116, ch117, ch118, ch119, ch120;

double dd111, dd112, dd113, dd114, dd115, dd116, dd117, dd118, dd119, dd120;

long double ldd111, ldd112, ldd113, ldd114, ldd115, ldd116, ldd117, ldd118, ldd119, ldd120;

float ff111, ff112, ff113, ff114, ff115, ff116, ff117, ff118, ff119, ff120;

double dff111, dff112, dff113, dff114, dff115, dff116, dff117, dff118, dff119, dff120;

long double ldff111, ldff112, ldff113, ldff114, ldff115, ldff116, ldff117, ldff118, ldff119, ldff120;

char ch121, ch122, ch123, ch124, ch125, ch126, ch127, ch128, ch129, ch130;

double dd121, dd122, dd123, dd124, dd125, dd126, dd127, dd128, dd129, dd130;

long double ldd121, ldd122, ldd123, ldd124, ldd125, ldd126, ldd127, ldd128, ldd129, ldd130;

float ff121, ff122, ff123, ff124, ff125, ff126, ff127, ff128, ff129, ff130;

double dff121, dff122, dff123, dff124, dff125, dff126, dff127, dff128, dff129, dff130;

long double ldff121, ldff122, ldff123, ldff124, ldff125, ldff126, ldff127, ldff128, ldff129, ldff130;

char ch131, ch132, ch133, ch134, ch135, ch136, ch137, ch138, ch139, ch140;

double dd131, dd132, dd133, dd134, dd135, dd136, dd137, dd138, dd139, dd140;

long double ldd131, ldd132, ldd133, ldd134, ldd135, ldd136, ldd137, ldd138, ldd139, ldd140;

float ff131, ff132, ff133, ff134, ff135, ff136, ff137, ff138, ff139, ff140;

double dff131, dff132, dff133, dff134, dff135, dff136, dff137, dff138, dff139, dff140;

long double ldff131, ldff132, ldff133, ldff134, ldff135, ldff136, ldff137, ldff138, ldff139, ldff140;

char ch141, ch142, ch143, ch144, ch145, ch146, ch147, ch148, ch149, ch150;

double dd141, dd142, dd143, dd144, dd145, dd146, dd147, dd148, dd149, dd150;

long double ldd141, ldd142, ldd143, ldd144, ldd145, ldd146, ldd147, ldd148, ldd149, ldd150;

float ff141, ff142, ff143, ff144, ff145, ff146, ff147, ff148, ff149, ff150;

double dff141, dff142, dff143, dff144, dff145, dff146, dff147, dff148, dff149, dff150;

long double ldff141, ldff142, ldff143, ldff144, ldff145, ldff146, ldff147, ldff148, ldff149, ldff150;

char ch151, ch152, ch153, ch154, ch155, ch156, ch157, ch158, ch159, ch160;

double dd151, dd152, dd153, dd154, dd155, dd156, dd157, dd158, dd159, dd160;

long double ldd151, ldd152, ldd153, ldd154, ldd155, ldd156, ldd157, ldd158, ldd159, ldd160;

float ff151, ff152, ff153, ff154, ff155, ff156, ff157, ff158, ff159, ff160;

double dff151, dff152, dff153, dff154, dff155, dff156, dff157, dff158, dff159, dff160;

long double ldff151, ldff152, ldff153, ldff154, ldff155, ldff156, ldff157, ldff158, ldff159, ldff160;

char ch161, ch162, ch163, ch164, ch165, ch166, ch167, ch168, ch169, ch170;

double dd161, dd162, dd163, dd164, dd165, dd166, dd167, dd168, dd169, dd170;

long double ldd161, ldd162, ldd163, ldd164, ldd165, ldd166, ldd167, ldd168, ldd169, ldd170;

float ff161, ff162, ff163, ff164, ff165, ff166, ff167, ff168, ff169, ff170;

double dff161, dff162, dff163, dff164, dff165, dff166, dff167, dff168, dff169, dff170;

long double ldff161, ldff162, ldff163, ldff164, ldff165, ldff166, ldff167, ldff168, ldff169, ldff170;

char ch171, ch172, ch173, ch174, ch175, ch176, ch177, ch178, ch179, ch180;

double dd171, dd172, dd173, dd174, dd175, dd176, dd177, dd178, dd179, dd180;

long double ldd171, ldd172, ldd173, ldd174, ldd175, ldd176, ldd177, ldd178, ldd179, ldd180;

float ff171, ff172, ff173, ff174, ff175, ff176, ff177, ff178, ff179, ff180;

double dff171, dff172, dff173, dff174, dff175, dff176, dff177, dff178, dff179, dff180;

long double ldff171, ldff172, ldff173, ldff174, ldff175, ldff176, ldff177, ldff178, ldff179, ldff180;

char ch181, ch182, ch183, ch184, ch185, ch186, ch187, ch188, ch189, ch190;

double dd181, dd182, dd183, dd184, dd185, dd186, dd187, dd188, dd189, dd190;

long double ldd181, ldd182, ldd183, ldd184, ldd185, ldd186, ldd187, ldd188, ldd189, ldd190;

float ff181, ff182, ff183, ff184, ff185, ff186, ff187, ff188, ff189, ff190;

double dff181, dff182, dff183, dff184, dff185, dff186, dff187, dff188, dff189, dff190;

long double ldff181, ldff182, ldff183, ldff184, ldff185, ldff186, ldff187, ldff188, ldff189, ldff190;

char ch191, ch192, ch193, ch194, ch195, ch196, ch197, ch198, ch199, ch200;

double dd191, dd192, dd193, dd194, dd195, dd196, dd197, dd198, dd199, dd200;

long double ldd191, ldd192, ldd193, ldd194, ldd195, ldd196, ldd197, ldd198, ldd199, ldd200;

float ff191, ff192, ff193, ff194, ff195, ff196, ff197, ff198, ff199, ff200;

double dff191, dff192, dff193, dff194, dff195, dff196, dff197, dff198, dff199, dff200;

long double ldff191, ldff192, ldff193, ldff194, ldff195, ldff196, ldff197, ldff198, ldff199, ldff200;

char ch201, ch202, ch203, ch204, ch205, ch206, ch207, ch208, ch209, ch210;

double dd201, dd202, dd203, dd204, dd205, dd206, dd

~~Program to~~ ^{check} the given number is strong number
without functions

```
#include <stdio.h>
#include <conio.h>
main()
{
    int i, num, copy=0, r, f=1, sum=0;
    printf("enter the number");
    scanf("%d", &num);
    copy = num;
    while (num != 0)
    {
        int f=1;
        r = num % 10;
        i = 1;
        while (i <= r)
        {
            f = f * i;
            i++;
        }
        sum = sum + f;
        num = num / 10;
    }
    printf("if (%d == sum)\n", copy);
    printf("number is a strong number");
    else
    printf("number is not a strong number");
    getch();
}
```

With functions

```
#include <stdio.h>
#include <conio.h>
int fact(int r);
main()
{
    int i, num, r, copy=0, f=1, sum=0;
    printf("enter the number");
    scanf("%d", &num);
```

```
copy = num;
while (num != 0)
{ r = num % 10;
  sum = sum + fact(r);
  num = num / 10;
}
if (copy == sum)
  printf("number is a strong number");
else
  printf("number is not a strong number");
getch();
return 0;
}

int fact(int n)
{
  int i=1, f=1;
  for (i=1; i<=n; i++)
  {
    f = f * i;
  }
  return f;
}
```

Find the max element in an array

main()

```
{  
    int a[25], i, n, max;  
  
    scanf ("%d", &n);  
  
    for(i=0; i<n; i++)  
        scanf ("%d", &a[i]);  
  
    max = a[0];  
  
    for(i=1; i<n; i++)  
    {  
        if (a[i] > max)  
            max = a[i];  
    }  
  
    printf ("%d", max);  
}
```

→ To get an array and printing it in the reverse order

Ex: Input
4
5
2
3
1

Output
1
3
2
5
4

Logic
swapping
 $a[1] \rightarrow a[n]$
 $a[2] \rightarrow a[n-2]$

main()

```
{  
    int a[5], i, n, p;  
  
    scanf ("%d", &n);  
  
    for
```

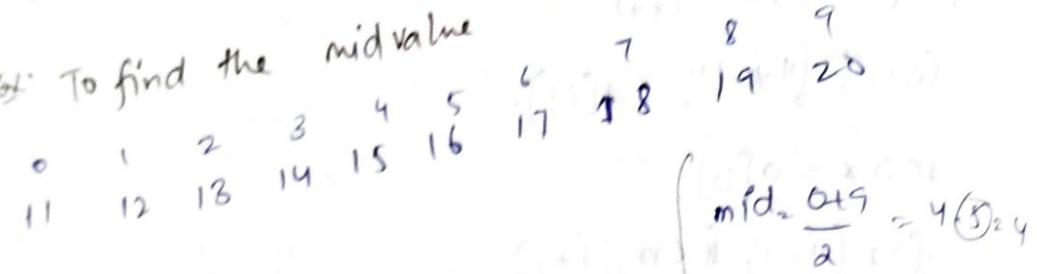
```
    for (i=0; i<n/2; i++)  
    {  
        c = a[i];  
        a[i] = a[n-i-1];  
        a[n-i-1] = c;  
    }
```

divide and conquer "policy"

Binary Search:

→ binary search is efficient only if the input is in sorted order

→ To find the mid value



low = 0;

high = n-1;

while (high >= low)

{ mid = (low + high) / 2

if (a[mid] == key)

{ printf ("element found");

break;

}

else if (key > a[mid])

low = mid + 1;

else

high = mid - 1;

}

From a[mid] = 15
Key = 18

→ low = 4 + 1 = 5

mid = (5+9)/2 = 7 = n

→ H.W

1) ~~using~~ convert decimal number to binary

a) Removal of duplicates from

Ordered array
and
unordered array

ordered array

1
1
2
2
3
4
5
5

unordered array

2
2
1
3
5
5
4
4

TWO DIMENSIONAL ARRAYS

Ex:- matrix

```

main()
{
    int a[10][10], n, i, j;
    scanf("%d", &n);
    for(i=0; i<n; i++)
        for(j=0; j<n; j++)
            scanf("%d", &a[i][j]);
    for(i=0; i<n; i++)
        for(j=0; j<n; j++)
            printf("%d", a[i][j]);
}

```

Program for find the addition of two matrices

```
→ main()
  {
    int a[10][10], b[10][10], i, j, n, c[10][10];
    → Scanf ("%.d", &n);
    for (i=0; i<n; i++)
      for (j=0; j<n; j++)
        Scanf ("%.d", &a[i][j]);
    for (i=0; i<n; i++)
      for (j=0; j<n; j++)
        Scanf ("%.d", &b[i][j]);
    for (i=0; i<n; i++)
      for (j=0; j<n; j++)
        c[i][j] = a[i][j] + b[i][j];
    for (i=0; i<n; i++)
      for (j=0; j<n; j++)
        printf ("%d", c[i][j]);
  }
```

1(a) WAP To find the transformation of the matrix
 2(a) WAP To point the diagonal elements of a matrix
 3(a) WAP for the multiplication of matrix

```

main()
{
    int a[10][10], b[10][10], c[10][10], n, i, j, k;
    Scanf ("%.d", &n);
    for (i=0; i<n; i++)
        for (j=0; j<n; j++)
            Scanf ("%d", &a[i][j]);
    for (i=0; i<n; i++)
        for (j=0; j<n; j++)
            Scanf ("%d", &b[i][j]);
    for (i=0; i<n; i++)
        for (j=0; j<n; j++)
            c[i][j] = 0;
    for (k=0; k<n; k++)
        for (i=0; i<n; i++)
            for (j=0; j<n; j++)
                c[i][j] = c[i][j] + a[i][k] * b[k][j];
    printf ("%d", c[i][j]);
}
  
```

Working of the loop:-

Transpose of a matrix:-

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

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

```
main()
```

```
{
```

```
int a[10][10], b[10][10], i, j, n;
```

```
printf("enter the number of elements");
```

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

```
printf("enter the elements of A matrix");
```

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

```
{
```

```
for(j=0; j<=n; j++)
```

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

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

```
y
```

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

```
{
```

```
for(j=0; j<=n; j++)
```

```
b[i][j] = a[j][i];
```

```
y
```

```
printf("transpose of the matrix 'A' is b=\n");
```

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

```
{
```

```
for(j=0; j<=n; j++)
```

```
printf("%d", b[i][j]);
```

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

```
y
```

```
getch();
```

```
}
```

- Q) WAP to find the sum of diagonal elements
- a) WAP to fill upper right triangular matrix by '1' and lower right triangular matrix by '-1' and diagonals by '0': $\begin{bmatrix} 0 & 1 & 1 \\ -1 & 0 & 1 \\ -1 & -1 & 0 \end{bmatrix}$

- b) WAP to generate multiplication table of following format

									10
									10
									20
									30
									40
									50
									60
									70
									80
									90
									100
10	10	20	30						

WAP to convert decimal to binary

```
→ main()
{ int n, a[100], i=0, j=0;
  printf ("Enter a number");
  scanf ("%d", &n);
  while (n != 0)
  {
    a[i] = n % 2;
    n = n / 2;
    i++;
  }
  for (j = i-1; j >= 0; j--)
  {
    printf ("%d", a[j]);
  }
  getch();
}
```

→ WAP for removal of duplicates from ordered array

```
main()
{
  int i, j, n, a[100];
  printf ("Enter the size of array");
  scanf ("%d", &n);
  printf ("Enter values of array in sorted manner\n");
  for (i = 0; i < n; i++)
  {
    scanf ("%d", &a[i]);
  }
  for (i = 0; j = 0; i < n-1; i++)
  {
    if (a[i] != a[i+1])
    {
      j++;
      a[j] = a[i+1];
    }
  }
}
```

```
for (i=j+1; i<n; i++)  
    a[i]=0;  
    getch();  
}
```

→ Find the square root of a number

```
int main(void)  
{  
    float g1, g2, m;  
    printf("Enter a number: ");  
    scanf("%f", &m);
```

$g_2 = m/2$; // initial guess all square roots from
1-9 are close to their half value

do

 {
 g1=g2; // g1 is previous guess

 g2 = $(g_1 + m/g_1)/2$; // g2 is current guess, m/g_1
 is complementary

 printf("g2=%f\n", g2);

}

 while (fabs(g1-g2)>0.001);

 printf("Square root is %f", g2);

 system("pause");

→ Find the smallest divisor:

→ 9 int main(void)

{ int n, d=3, s1, sdivisor;

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

$$\text{if } (n \neq 0) = 0$$

sdivisor = 2;

else

$$s = \text{sgn}(\eta);$$

while $((n + d) \neq 0) \& (d < 2)$

$$f(d-d+2)$$

if $(n \cdot 1 \cdot d = 0)$

§divisor = d

else.

advisors

3

```
printf("smallest divisor is %d ", a%divisor);
```

System ("pause").

13

\rightarrow n^{th} power of number:-

```
int main(void)
```

3

inf $n, x, p=1$

```
printf("enter the values of x and n");
```

$\text{Scanf}((d + d), \&x, \&y)$

while ($n > 0$)

d if ($n \neq 2 = -1$)

$p = p * x$

$n = n / 2;$

$x = x * x;$

}

`printf ("%d", p),`

`system ("pause"),`

}

Assignment :-

1. WAP to compute $\frac{1+x+x^2}{2!} + \frac{x^3}{3!} + \dots$ terms

2. WAP to print the numbers that do not appear in the fibonacci . The no. of such terms to be printed should be given by the user

3. WAP to print the second largest no. in the given array

4. WAP to find the intersection of two arrays

5. WAP to remove duplicates from an ordered array

Conversion from decimal to binary number

```
#include <stdio.h>
#include <conio.h>
main()
{
    long int dnumber, quotient, n;
    int bnumbers[50], i=1, j;
    printf("Enter the decimal number:");
    scanf("%d", &dnumber);
    quotient = dnumber;
    while (quotient != 0)
    {
        bnumbers[i] = quotient % 2;
        quotient = quotient / 2;
        i++;
    }
    printf("Binary number is = ");
    for (j=i-1; j>0; j--)
        printf("%d", bnumbers[j]);
    getch();
}
```

Binary to decimal conversion

```
#include <stdio.h>
#include <conio.h>
#include <math.h>
main()
{
    long int decimalnumber=0;
    int i=1, j, binarynumber[50], n, k;
    printf("Enter the number of elements:");
    scanf("%d", &n);
}
```

```

printf("enter the binary number");
for(i=0; i<=n; i++)
    scanf("%d", &binarynumber[i]);
for(j=i-1, k=0; j>=0; j--, k++)
    decimalnumber = binarynumber[j] * pow(2, k) + decimalnumber;
printf("decimal number equivalent of the given
      binary number is");
printf("1d", decimal number);
getch();
}

```

1) To compute the series $1+x+\frac{x^2}{2!}+\frac{x^3}{3!}+\frac{x^4}{4!}+\dots$ n terms

```

#include <stdio.h>
#include <conio.h>
int main()
{
    int i, j, k, x, fact, prod, n;
    float sum=0, term=0;
    printf("enter the value of x");
    scanf("%f", &x);
    printf("enter the value upto which series to be calculated");
    scanf("%d", &n);
    for(i=1; i<=n; i++)
    {
        fact=1;
        prod=1;
        for(j=1; j<=i; j++)
        {
            fact=fact*j;
            prod=prod*x;
        }
        term=(float)prod/(float)fact;
        sum+=term;
    }
    printf("sum = %f", sum);
}

```

```
sum = sum + term;
```

```
} printf("the sum of the series is %f", sum);
```

```
getch();
```

```
→ return 0; (return 0; if i < n) then
```

```
(n >= 0) then return 0; (else return 1;)
```

```
else (n < 0) then return 1; (else return 0;)
```

```
else (n < 0) then return 1; (else return 0;)
```

```
else (n < 0) then return 1; (else return 0;)
```

```
else (n < 0) then return 1; (else return 0;)
```

```
else (n < 0) then return 1; (else return 0;)
```

```
else (n < 0) then return 1; (else return 0;)
```

```
else (n < 0) then return 1; (else return 0;)
```

```
else (n < 0) then return 1; (else return 0;)
```

```
else (n < 0) then return 1; (else return 0;)
```

```
else (n < 0) then return 1; (else return 0;)
```

```
else (n < 0) then return 1; (else return 0;)
```

```
else (n < 0) then return 1; (else return 0;)
```

```
else (n < 0) then return 1; (else return 0;)
```

```
else (n < 0) then return 1; (else return 0;)
```

```
else (n < 0) then return 1; (else return 0;)
```

```
else (n < 0) then return 1; (else return 0;)
```

```
else (n < 0) then return 1; (else return 0;)
```

Functions are subprograms

library functions

built-in

ex: <math.h>

<ctype.h>

<stdlib.h>

<string.h>

user defined

ex: main()

function
declaration

int sum (int, int);

main()

{ int a, b, c;

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

actual arguments
c = sum(a, b); → function call

printf ("%d", c);

getch();

}

function prototype (or) function declaration

format arguments
(no semicolon)

int sum(int x, int y)

int z; } function body
z = x + y; } function definition

return(z);

z

→ If simple return is used it acts as closed curly braces

→ stack:

A register in the ALU unit for function calls

(Arithmetic)
(logical)

calls

→ The arguments which are passed by "function call" to the "function" are called "call by value".

→ In this only the copy of the values are passed in function any changes that do not reflect in

the actual values of the variables

→ 5

→ call by Reference or address:

→ 6

→ possible with pointers

→ 7

* In this address (memory cell) is passed to the function. Any changes will be reflected in the original value

→ 8

Categories of functions

- 1) No argument . no return
 - 2) with argument no return
 - 3) no argument with return
 - 4) with argument with return
- 2) with argument no return:

void sum (int, int);

main()

d. int a, b,

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

sum (a, b); → function call

getch();

by

void sum(int x; int y)

```
{ int z;  
z = x+y;  
printf("%d\n", z);  
}
```

3) No argument with return:

```
int sum (void);
```

main()

```
{ int c;  
c = sum(x, y);  
printf ("%d\n", c);
```

getch();

```
}
```

int sum(void)

```
{ int x, y, z;
```

```
scanf ("%d %d", &x, &y);
```

$\rightarrow z = x + y;$

```
printf ("%d", z);
```

```
}
```

M.W

1. Write a 'c' program to sort the given character

Example of passing one-D-array
into functions

→
→ i
→ :
G
: I
i
:
F
—
int sum (int [] , int);
main()
{
 int a[50] , i, n, val;
 scanf (" %d ", &n);
 for (i=0 ; i<n ; i++)
 scanf (" %d ", &a[i]);
 val = sum (a, n);
 printf (" %d ", val);
}

int sum (int a[] , int n)
{
 int i, tot = 0 ;
 for (i=0 ; i<n ; i++)
 tot = tot + a[i];
 return tot ;
}

1st is optional & remaining
should be given see

፩፻፲፭

```

void addmatrix( int [ ] [50], int [ ] [50], int n)
main()
{
    int a[50][50], b[50][50], i, j, n;
    Scanf (" %d", &n);
    for ( i=0; i<n; i++)
        for ( j=0; j<n; j++)
            Scanf (" %d", &a[i][j]);
    for ( i=0; i<n; i++)
        for ( j=0; j<n; j++)
            Scanf (" %d", &b[i][j]);
    addmatrix(a, b, n);
}

void addmatrix( int a[ ] [50], int b[ ] [50], int n)
{
    int K, L;
    for ( K=0; K<n; K++)
        for ( L=0; L<n; L++)
            printf (" %d", a[K][L] + b[K][L]);
}

```

Recursion

- 1. * a function calling onto itself called stack
- 2. Uses a data structure
- 3. a conditional statement is required to
- 4. solves complex problems easily by using "divide and conquer" strategy

Eg:

```
main()
{
    printf ("Recursion");
    if (!kbhit())
        main();
}
```

Finding factorial using recursion

```
int fact(int);
main()
{
    int val, n;
    scanf ("%d", &n);
    val = fact(n);
    printf ("%d", val);
}
```

```
int fact(int m)
```

```
{ int f;
```

```
if(m == 1)
```

```
return 1;
```

```
else
```

```
f = m * fact(m-1);
```

```
return f;
```

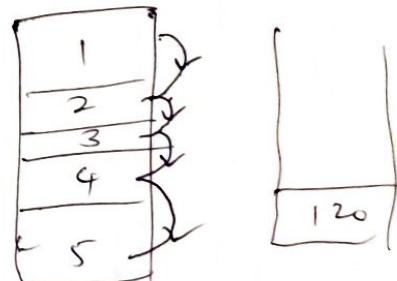
```
}
```

* a single return can return
only a single value
* for returning multiple values we
have to use pointers
* my expression

gets converted

into postfix expression
and it is executed
from right to left

datastructure
linear property of stack
STACK - Last in first out
ex: wearing bangles



Q Home work

1. Using functions WAP to compute factorial of "n".
2. to find largest element in an array
3. To display diagonal elements
4. To perform transpose of a matrix
5. To print fibonacci sequence upto 'n' terms using recursion

Program to print the diagonal elements

```
#include <stdio.h>
#include <conio.h>
main()
{
    int a[10][10], i, j, n;
    printf("enter the number of elements ");
    scanf("%d", &n);
    printf("enter the elements of a matrix");
    for (i=0; i<=n; i++)
    {
        for (j=0; j<=n; j++)
        {
            scanf("%d", &a[i][j]);
        }
    }
    for (i=0; i<=n; i++)
    {
        for (j=0; j<=n; j++)
        {
            if (i==j)
                printf("%d", a[i][j]);
        }
    }
    getch();
}
```

STRINGS (char, int) $bos \# 2$ (?)

```
main()
{
    char name[100]; → no "4"
    scanf("%s", name);
    printf("%s", name);
}
```

→ this will not read
the string with
a space

→ To read string
with space

`<stdio.h> { gets(); puts(); }`
functions are used

→ not
except what ever given in the brackets all the things
are accepted (but [not] "name")

$\rightarrow \text{scanf} \{ \text{*}, [\text{abc}]^*, \text{name} \}$

$\rightarrow a, b, c$ are not accepted

$\rightarrow \text{scanf}(\text{"%.}[\sim]\text{"},$

STRING HANDLING FUNCTIONS

String: combining two strings

1) Street - string Concatenation

27 ~~scopy~~ - string copy

3) `strcmp` - string compare

47 string - string length

5) ~~string~~ - string reverse

Q. 6. = To search the values in a array

c) **Strokr** - To Search the value
in the pointer value

2) strstr - it will return the pos

7) strcat(str1, str2)

str1 = VIT ; str2 = university

str1 = VIT university

2) String copy:

strcpy(str1, str2)

str1 = VIT ; str2 = university

Syntax:

strcpy(destination_string,
source_string, number_of_chars)

str1 = university

3) String compare:

str1 = ABCD

str2 = abCD

* If the two strings given are equal then it will return '0'.

* It subtracts the ASCII values of two strings

4) String length:

strlen(str1);

String length returns no. of characters in the string

strlen(VIT) =

3

String reverse:

strrev(v17) → TIV

strcmp

strcmp(str1, I);

To search the values in a string

strstr

strstr(str1, UNN);

it will return the pointer value

MAP to copy the string without using
string handling function

main()

{ char [50], b[50];

int i=0;

getchar();

while [a[i]]{5}

{

 b[i]=a[i];

 i++;

}

 if b[i] == '0'

 break;

 puts(b);

3

To read a string

Q) WAP to read a string with spaces and print it without spaces

a) WAP to concatenate two strings

b) WAP to compare two strings

POINTERS

new

- * derived data type
- * a pointer provides away of accessing to a variable without referring to the variable directly
- * the mechanism used for this is the address of the variable
- * Thus, a pointer variable is a variable that holds the memory address of another variable
- * This variables are called as pointers because by storing an address they point to a particular value in a memory

Advantages:-

- * makes efficient programs
- * it is a powerful programming construct

Disadvantages:-

- * difficult to understand
- * errors produced by this is difficult to debug

Uses:

- * used to call by address
- * facilitating the changes made to a variable in the call function to become permanent, available in the function from where it is called
- * Used to return more than one value from a function indirectly
- * used to pass arrays and strings more conveniently from one function to another
- * it communicates information about memory using malloc function which return the location of free memory
- * Compiles coding faster and creates efficient code than any other derived data types such as arrays
- * can be used to create complex data structures such as 'linked lists' and 'binary trees'

main()
{
 int a=5, *ptr;
 ptr = &a;

 printf ("%d", a); → 5
 printf ("%f", *ptr); → 5

 printf ("%f", a); → address of a
 printf ("%f", &a); → address of a
 printf ("%f", 4ptr); → address of ptr

 3

points to "the value at address"
indirection operator (*)

decrementing operator

* integer pointer should
point to int address
* float pointer to float address
float (1.0000)

* The size of a pointer is always 4 bytes (depending on O.S)

* Memory address will be in hexadecimal
format

"%p", "%x" → used to print the address of the variable

→ main()
{
 int a=5, *ptr;
 ptr = &a;
 printf ("%d", a); → 5
 a=10;
 printf ("%d", a); → 10

```

*ptr=15; (no output) → 15
→ printf("%d", a); → 15
|
|
|
→
(
|
|
|
;
main()
{
    int a, b, *pa = &a, *pb = &b;
    scanf("%d %d", pa, pb);
    printf("sum = %d", *pa + *pb);
}

```

$\star(++)P$ — increments the memory address
 $++(\star P)$ — increments the value stored in memory address

H.W
 ① WAP using pointers, to exchange the value of two variables using & without using ~~too~~ variables

② WAP using pointer to compute factorial of n.

③ Differences b/w Arrays & Pointers

Swapping using functions

```
void swap(int *x, int *y);  
main()  
{  
    int x, y;  
    scanf("%d %d", &x, &y);  
    printf("values before swapping\n");  
    swap(&x, &y);  
    printf("values after swapping\n");  
    system("pause");  
}  
void swap(int *x, int *y)  
{  
    int temp;  
    temp = *x;  
    *x = *y;  
    *y = temp;  
}
```

pointers.

Returning multiple values using

```
float compute(float, float *);  
int main()  
{  
    float area, circum;  
    printf("enter the radius in'')";  
    scanf("%f", &r);  
    area = compute(r, &circum);  
    printf("Area = %f in", area);  
    printf("circumference = %f in", circum);  
    system("pause");  
}  
float compute(float r, float *p)  
{  
    float a;  
    a = 3.14 * r * r;  
    *p = 2.0 * 3.14 * r;  
    return a;  
}
```

Returning Pointers address

return data type
int * pointmax(int *, int *);
int main()
{
 int a, b, *p;

printf ("enter the values of a &b\n");
scanf ("%d %d", &a, &b);
p = pointmax (&a, &b);
printf ("The largest value is %d", *p);
system ("pause");

{
int * pointmax (int *x, int *y)
{
 if (*x > *y)
 return x;
 else
 return y;
}

* Array is a pointer constant
scaling
[base address + i * size of the data type]

```
int main()
{
    int a[5], i;
    printf ("enter array A\n");
    for (i=0; i<5; i++)
        scanf ("%d", &a[i]);
    for (i=0; i<5; i++)
        printf ("%d\n", *(&a[i])); // a[i]
    system ("pause");
}
```

→ When array is used the array name(s) holds the base address, the compiler adds the base address to subscript \times size of data type to get the offset (balance) address. This process is called scaling.
Hence, the subscript notation is converted to a pointer notation. Use of pointer notation works fastly reducing the conversion time.

(Given)

int arr[5]

arr[0] = 10

arr[1] = 20

arr[2] = 30

arr[3] = 40

arr[4] = 50

arr[5] = 60

arr[6] = 70

arr[7] = 80

arr[8] = 90

arr[9] = 100

arr[10] = 110

arr[11] = 120

26/10/2012

USER DEFINED DATA TYPES

STRUCTURES AND UNIONS

keyword - struct

struct tag-name

```
{  
    members 1;  
    members 2;  
    members 3;  
}
```

(list of variables separated by comma)

or
struct tag-name (, ,);

without tag-name :- variable structure

including , " : Tagged structure

Type definition structure

↳ typedef
↳ can be given an alternative name
(alias)

for existing data type.

Access operators: connects variables & tag name

• Dot operator

→

* The extra space (memory) added by compiler
(padding)

is "byte" in order to round the memory

to multiples of "4"

diff b/w "UNION" & "STRUCTURES".

* key word "UNION"

* difference in memory allocation

* only one ^{member} can be accessed at a time in union. (it is loss the choice of union)
drawback

* In union whichever the size is most it is the size of array

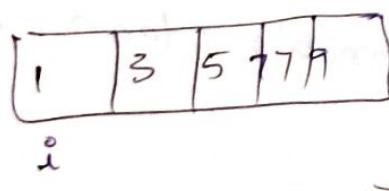
Home Work:-

1Q. WAP to sort a list of ten names in alphabetical order

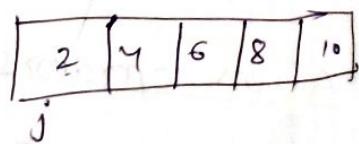
2Q. WAP to merge two arrays in sorted order so that if an integer is in both arrays and gets added to final array only once

3Q.

a



b



3Q. WAP to convert decimal to hexadecimal

4Q. WAP using user define function ~~not the~~ to check whether the given no is strong ^{not}

5. for hospital keeps, a list of blood donors
in which each record has the following fields
name, age, address, blood type.

6. wAP to create an array of structures to
store all these details and print the list
of all blood donors whose age is below
35. and blood type is OT.

FILES

read mode - To retrieve data

write mode -

append mode - to update file

FILE → <stdio.h>

→ inbuilt data structure in 'C'

→ To access file we should use pointers

Syntax: FILE *f1;

f1 = fopen("Input", "w");
→ opening a file

while (c = getchar() != EOF)
→ end of file
(until we press
ctrl+z)

Comment for writing a character into a

file = putc(c, f1);

fclose(f1); → close the file input.

reading
charact
putc, getc 3 file operators

getw, putw 3 reading integers

1. Difference between
 - a. structures and Union
 - b. structures and Arrays
2. Write a 'C' program to create records of a book store with following fields
3. Create a file structure called TEXT . Read the contents of text file character by character if the character read is a vowel write it to file called vowel otherwise, write it to a file called consonant .

structure

C++

→ SIMULA 67

→ SMALL TALK

→ Bjarne Stroustrup - developed C++
- 1979

→ To solve larger problems

→ 'C' with classes

→ 'C++' superset of 'C'

OO approach:

Real world entities
↓ mapping

(object)

↓

data

functions

interface implement

Features of obj:

a) encapsulation

b) polymorphism

c) Inheritance

a) encapsulation :-

program :-
#include <iostream.h>
I/O stream off layers
ipdev ← C is
device cont
@rr
clog

scans (---) functions
printf (→) extracts from I/O device
cin → extraction operator
{ cout << insertion " "
objects

std::in
std::out
std::endl linked
during
(program execution)

① #include <iostream.h>

void main()

{

cin >>

int a, b, c;

cout << "enter a";

cin >> a;

cout << "enter b";

cin >> b;

c = a + b;

cout << "sum of " << a << " and " << b
" is " << "is" << ;

int - 2
char - 1
float - 4
double - 8
void - 0

- ① WAP to display the series
2 2
3 3 3
4 4 4 4
- ② WAP to display the sum of the series

$$1+2+3+\dots+n$$

- ③ finding greatest of 3 numbers

① #include <iostream.h>

void main()

{

int n, i;

cin >> n;

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

cout << i;

}

② #include <iostream.h>

main()

{

int n, sum=0;

cin >> n;

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

sum = i + sum;

cout << sum;

}

```
#include <iostream.h>
void main()
```

```
{ for (int i=1; i<=9; i++)
```

```
{ for (int j=1; j<=i; j++)
```

```
{ cout << " " << j;
```

```
} //j
```

```
cout << "\n";
```

```
} //i
```

```
} //main.
```