Algorithm Flowchart

Algorithm: To add two numbers

1. start
2. Read and save first number a.
3. Read and save second number b.
4. Calculate sum, sum = a+b;
5. Write the sum
6. End

Q To find sum of numbers from 1 to 100.

1. start
2. Initialize sum=0
3. for i = 1 to 100

```
begin
    sum = sum + i
    i = i+1
End
```

4. Write the sum, sum
5. End

Q Algorithm: ① To find factorial of a number.

n!	n!
3! = 3 × 2 × 1	n = 3
0! = 1	fact = 1
	while (fact <= n)
~~n = 3~~	begin
~~fact = 0~~	fact = n × fact
	fact = fact + 1
	end

Page No.
Date : / /

Alogrithm : (1) find factorial of a number.

```
fact = 1,  i = 1
until (i <= n)
begin
  fact = fact * i
  i = i + 1
end
```

n = 3, fact = 1, i = 1
i = 1 fact = fact * i
= 1 * 1 = 1
i = 2 fact = 1 * 2 = 2
i = 3 fact = 2 * 3 = 6
i = 4
fail i > n = 3

⇒ (1) Start
(2) Initialize fact = 1, i = 1
(3) until (i <= n)
begin
fact = fact * i
i = i + 1
end
(4) Write fact
(5) End

Page No.
Date : 28/09/23

Flowchart

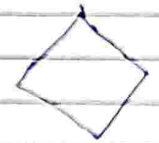
Flowchart is a pictorial representation of an algorithm.

Flowchart Symbols

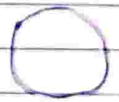
(1)
Terminal (Start / Stop)
(To start & stop of program)

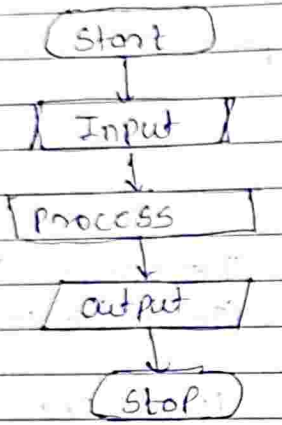
(2)
Processing
used to represent process step in algorithm

(3)
Input / output
used to input/output the data

(4) 
decision / condition box
used to take decision according to condition output in condition box

(5)
Flow lines
To represent the flow of program.

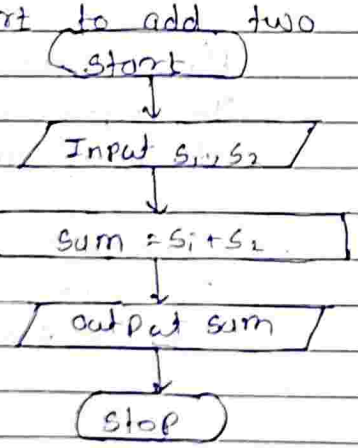
(6) 
Connector
To connect the number of flowchart at one place.



Q Too add two number

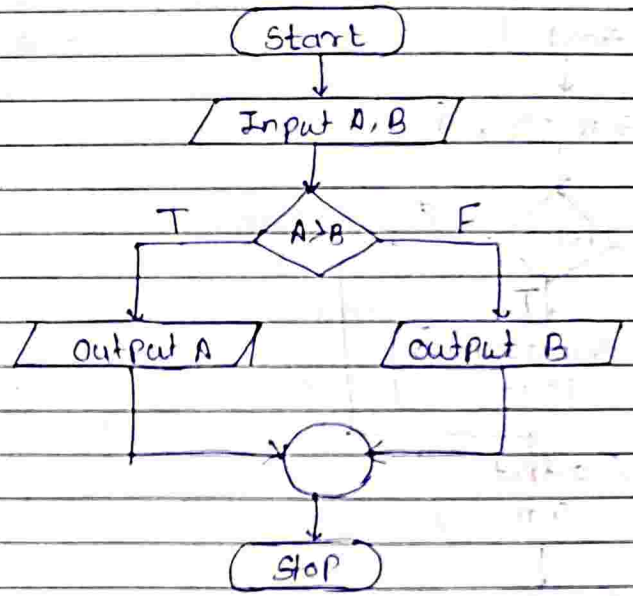
1. Start
2. Input S_1, save
3. Output S_2, save
4. calculate sum, sum, $S_1 + S_2$
5. Write the sum, sum
6. stop

Flowchart to add two number



② To find greatest of two number

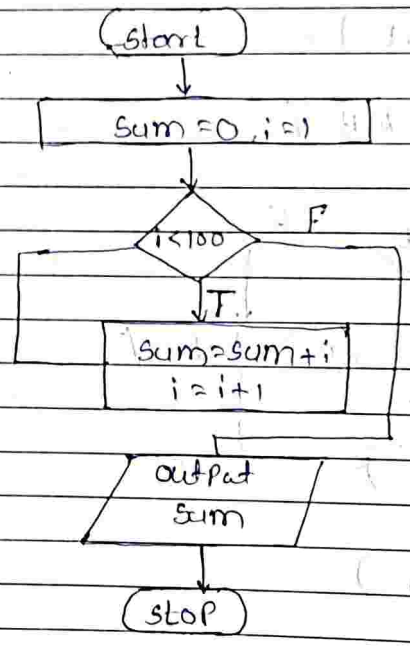
1) Start
2) Input first number A, save A
3) Input second number B, save B
4) If (A > B)
5) output A else
6) output B
7) stop



Page No.
Date : / /

Q To find sum of number 1 to 100

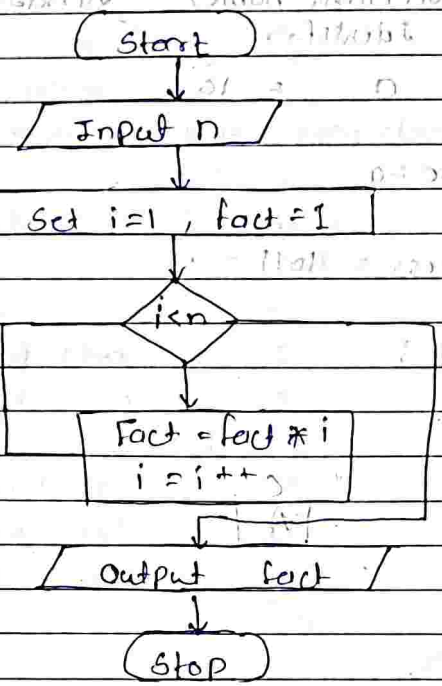
① Start
② Intialize sum = 0
③ increment to i = 1
④ Continue until i<100
   a) Calculate sum, sum = sum + i
   b) i = i + 1
⑤ Write the sum, sum
⑥ Stop



Page No.
Date / /

④ Write algorithm to find factorial of a number.

1) Start
2) Input n
3) Set i = 1, fact = 1
4) Continue until, i<n
   a) Calculate the product
      fact = fact * i
   b) Increment i
5) Write the output, fact
6) Stop



Page No.
Date : / /

Data types

Data types

- primary Data type
  - Integer - int
  - Float - float
  - Character - char
- secondary Data type
  - Array, structure, union, Enumeration, pointer

Data type variable name / Identifier variable / value

int a = 10;

variable name = a
value = 10
memory address = 1011

Ex float b = 19;

b
1010 [19]

char c = 'A' c [A]

2 = 16 Bits
2^{16} = 32768 → 65536/1

1 Bytes = 8 Bits
2^8 = 256
-126 to 127
256

Page No.
Date : / /

* Rules of for creating or variable name:

1. variables name should not exit ~~occured~~ exceed 8 character. some compiler support more character (1 to 31) or 1 to 127.
2. variable name combination of characters number and underscore.
3. variable name should start with characters.
4. Capital & Small letters are distinct (i.e. sensetive)
5. No sense special characters like #, ;, /, etc

	Data type	typical size in Bytes	minimal Range	format specifier
1	char	1	-126 to 127	%c
2	unsigned char	1	0 to 255	
3	Signed char	1	-128 to 127	
4	int	2	-32768 to 32767	%d
5	unsigned int	2	0 to 65536	%d
6	signed int	2	-32768 to 32767	%d
7	short int	2	-32768 to 32767	%d
8	long int	4	2,147,483,608 to 2,147483,647	%d
9	float	4	1E-38 to 1E+37 6 ~~size~~ digit precision	%f
10	~~double~~	8	~~1E-38 to 37 to digit precision~~	

11) ~~long double~~ to ~~1E-38 to 1E+37~~
~~no digit precision~~

10) double 8 1.7E -308 to
1.7E +308

long double 10 3.4E +4932 to
1.1E + 4932

variable can be distrated on the basis of ~~of~~ place of declaration

① local variable :- ~~Outside~~ Declaration inside function
② Global variable - Declare outside function
③ formal parameter - Declare in the definition of

Characteristics of an Algorithm

1. Input : zero or more quantities are extrenally supplied.

2. Output : Atleast one quantity is produced

3. Definiteness : Each instruction is clear and unambigous

4. Finiteness : If we trace out the instructions of an algorithm, them for all cases, the algorithm terminates after finite number of steps

5. Effectiveness : Every instruction most bevery basic so that it can be carried out in principle by a person using only pencil and paper.

Storage classes in C

Type of storage class

	Atomatic storage class	Register storage class	static	External or Global
① Keywords	auto	register	static	extern
② Storage	memory	CPU register	Memory	Memory
③ Default intial value	Garbage value	Garbage value	zero	zero

4) scope	local in which It is defined	local in which it defined	local in which defined	Global
5) Life	Till the control remains inside where it is defined	Till the control remain inside where it is defined	value is perviously incremented value betn different function	Till the Program execution does not come to an end
6) Speed of accessing value	Slow	Fast	Score	Score

Automatic Storage class

```
# include <stdio.h>
{
    auto int a=3;
    {
    auto int a=2;
    {
    auto int a=1;
    printf ("%d",a);
    }
    printf ("%d", a
    }
    printf ("%d,a)
    return 0
}
```



```
# include <stdio.h>
int main()
{
    register int a
    for (a=1; a<=10; a++)
    Printf ("%d\n", a)
    return 0
}
```

Static Storage class

```
# include <stdio.h>
void increase();
void main()
{
increase();
increase();
increase();
}
```

```
void increase()
{
static int a=1
Print f ("%d\n", a);
a=a+1;
}
```

Atomatic Storage class

```
# include <stdio.h>
void increase();
Void main()
{
increase()
increase();
increase()
}
```

```
void increase ()
{
    auto int a=1
    Printf ("%d"\n, a);
    a = a+1;
}
```

07/03/23

Decision Making

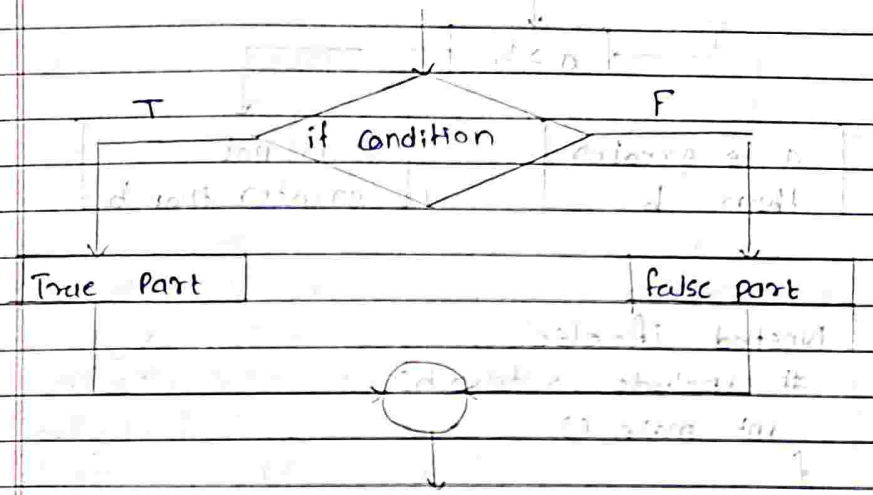
if -else

Syntax : if (condition)

```
{
    Statement 1;
    statement 2,
    statement n
}
else
{
    statement 1;
    :
    statement n;
}
```

True Part

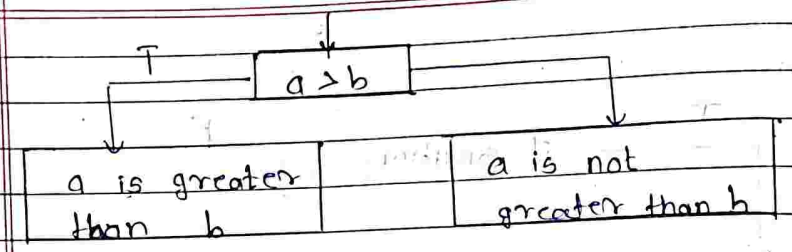
False part



Syntax : if

```
if (condition)
{
    Statement 1
    statement n.
}
```

```
☆ #include <stdio.h>
  int main ()
  {
    int a,b;
    Print f ("Enter two numbers:");
    Scanf ("%d %d" &a, &b);
    if (a>b)
    Print f ("%d is greater than %d", a,b);
    else
    Print f ("%d is not greater than %d", a,b);
    return (a);
  }
```

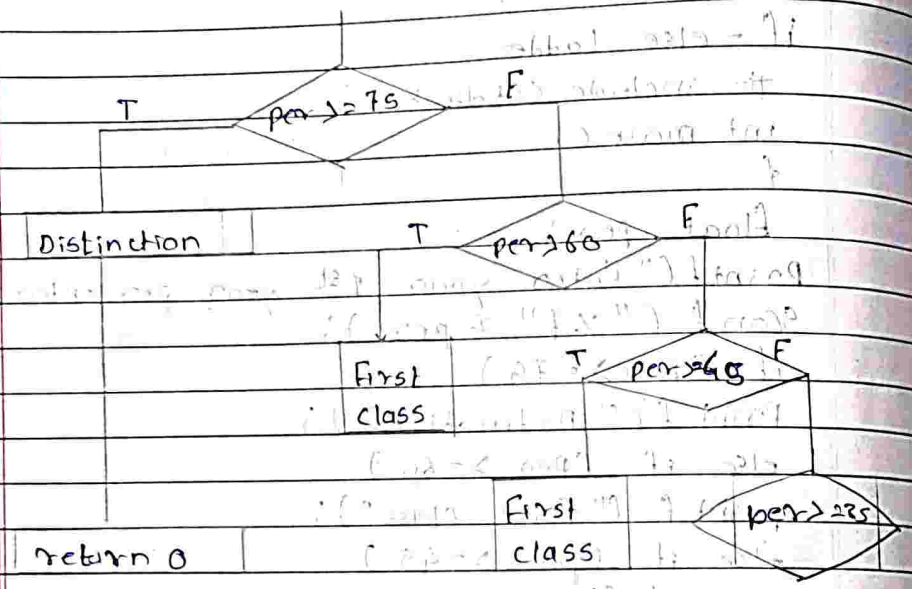


☆ Nested if-else

```
# include <stdio.h>
int main ()
{
int a, b, c;
Printf (" Enter numbers ");
Scanf ("%d %d %d", &a &b &c);
if (a>b)
{
if (a>c)
{
Printf ("%d is greatest ; a);
}
else if (b>c)
{
printf ("%d is greatest ", c);
}
else
Print f ("%d is greatest", c);
}
}
return (0);
}
```


if - else ladder

```
# include <stdio.h>
int main ()
{
float per;
Printf (" Enter your 1st year percentage");
Scanf ("%f" & per);
if (per >= 75)
Print f (" Distinction");
else if (per >= 60)
print f (" first class");
else if (per >= 45)
Print f (" second class);
else if (per >= 35)
Print f ("Third class");
else
Print f (" Fail ")
return (0);
}
```



Q Write a 'C' program to find greater of 2 number using conditional operator

→
```
# include <stdio.h>
int main()
{
    int a,b,c;
```


Write a 'C' Program to point table of 3 using for loop

```
# include <stdio.h>
# include <conio.h>
int main()
{
int i;
clrscr();
Print f ("Table of 3:");
For (i=1; i<=10; i++)
{
Print f("%d\n", 3*i);
}
getch();
return (0);
}
```

Output : Table of 3

3
6
9
⋮
30

* loop variation

1)
```
For (a=0, b=1; a+b<10; a++, b++)
{
 Print f(" a+b = %d", a+b);
}
```

2)
```
For (a=1, b=1; a<5 && b<5; a++, b++)
{
 Print f ("%d", a+b);
}
```

3)
```
For (a=0; a!=20;)
{
 Print f ("Input value of b")
 Scan f ("%d", &b);
  a++;
}
```

4)
```
For (a<10;)
{
 Print f ("%d", a);
  a++;
}
```

5)
```
For (;;)
{
 Ch = gech()
 if (ch == 'A')
 break;
}
```


6)
```
For (;;)
Print f ("Infinite)
```

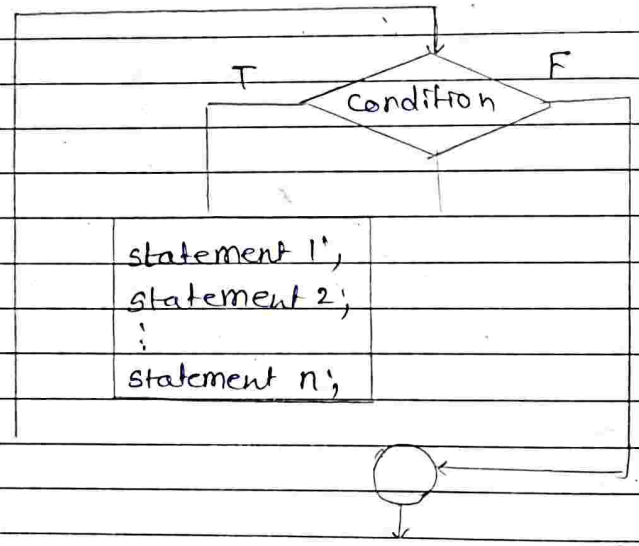
7)
```
int j;
For (int i=9; j=2; i<=10 i--)
{
 Print f ("%d %d", i, j);
  j++;
}
```

8) Write a program to find the number of and sum of all integers greater than 100 and less than 200 that are divisible by 7.

```
# include <stdio.h>
# include <conio.h>
  int main ()
 {
   int i;
   clrscr ();
 For (i=100, i%7 == 0, i<=200, i++);
  Count ++
  Sum = sum + i
 }
}

 Print f ("
```

1) While and do While

Syntax : while (condition)
{
Statement 1;
Statement 2;
⋮
statement n;
}




```
main()
{
  char ch;
  ch = getchar();
  while (ch == 'y')
  {
```



```
  exit (0);
  }
  return (0)
```

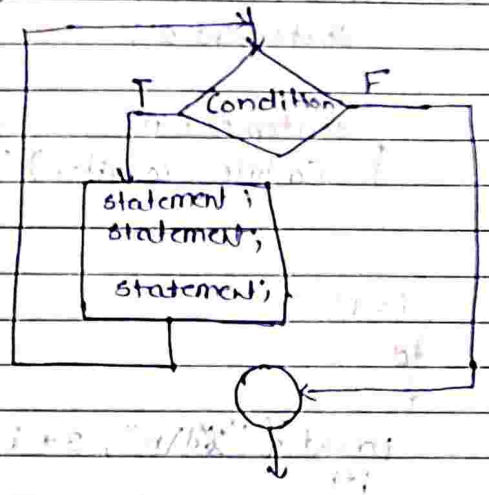
Syntax : do-while
{
Statement 1;
statement 2;
⋮
statement n;
}
while (condition);

```
i = 1;
do
{
Print f ("%d\n", 3*i);
    i++
} while (i <= 10);
```

Loops @While and do-While

Syntax:

```
while (condition)
{
    statement 1;
    statement 2;

    statement n;
}
```




```
main()
{
  char ch;
  ch = get char();
  while (ch == y)
  {
     exit (0);
  }
  return (0);
}


    i = 1;
  while (i<10)
  {
    print ("%d\n", 3*i);
      i++;
  }
```

- Syntax

```
do ~~while~~ while
{
  statement 1;
  statement 2;

  statement n;
} (while condition);
```

```
i=1;
do
{
  printf("%d\n", 3*i);
  i++
} while (i<10);
```

*(2) Jump statement includes

1. return
2. goto
3. break
4. ~~return~~ continue

```
goto Label;
Initialize i=1;
      do
      {
        printf("%d\n", 3*i);
```



```
      i++;
      goto Initialize
      } while (i<=10);
```

(3) switch

```
switch (switch variable)
{
  case 1;
      statement;
      break;
  case 2;
      statement;
      break;
  :
  default;
      statements;
}

main()
{
  cht ch;
  printf("Enter a day of week");
  scanf("%d", &ch);
  switch (ch)
  {
    case 1;
      printf("suday");
```

```
break;
case 2;
    printf ("Monday")
break;
case 3;
    printf (" tuesday")
break;
case 4;
    printf ("Wednesday")
break;
case 5;
    printf (" Thursday")
break;
case 6;
    prin
casebreak;
case 7;
    printf ("saturaday")
break;
default
    printf (wrong Input);
getch();
return (0);
}
```

* Nested Switch

```
switch (a)
{
  case 1;
      printf ("1st case"),
  break;
  case 2;
  switch (b)
  {
    case 1;
      printf ("1st case of 2nd switch),
    break;
    case 2 :
      printf ("2nd case of 2nd switch),
    break
  }
```

Q Write a program to print odd number between 1 to n.

```
# include <stdio.h>
# include < conio.h>
void main ()
{
    int n;
    printf ("\n Please enter the number");
    Scanf ("%d", &number);
```

```
for (i=1); i<=number; i++ }

int main()
{
  int (i,n);
  printf ("Enter any number");
  for (i=1, i<=n; i++)
  {
    if (i%2)==0)
    {
      printf (" Number %d is odd", i);
    }
  }
  getch()
  return (0),
  }
```

☆ Pre-processor directives

Various preprocessor directives

# define	# endif	# ifdef	# line
# elif	# error	# ifndef	#
# else	# if	# include	# undef

☆ Arrays

```
# include <stdio.h>
# include <conio.h>
int main()
{
    int a[10], i;
  printf ("Input ten numbers");
  for (i=0; i<10; i++)
{
  scanf ("%d", & a[i]);
}
  for (i=0; i<10; i++)
  printf ("%d\n", a[i])
  }
  getch ();
  return 0;
  }
```

Page No.
Date : / /

Write a program to find ~~greatest~~ ~~number~~ maximum of an array of 10 numbers

```
# include <stdio.h>
# include <conio.h>
int main ()
{
    int a[10];
    clrscr ();
    printf ("Input ten number");
    for (i=1; i<10; i++)
    {
      scanf ("%d", &a[i]);
    }
    max =a[a]
    for (i=1; i<10; i++)
    {
      if (a[i] > max)
      max = a[i];
    }
    Printf (" Maximum is %d", max);
    getch ()
    return (0),
}
```

Page No.
Date : / /

Searching and Sorting

Searching → linear search
→ Binary search

Sorting → Insertion sort
→ Bubble sort
→ selection sort

Linear search

a	0	1	2	3	4	5	6	7
	20	51	63	5	102	98	75	35

5

int a[8]; (i+1)

int a[8] = {20, 51, 63, 5, 102, 98, 75, 35}

program

```
# include <stdio.h>
# include <conio.h>
int main ()
{
    int a[10], found, i, yes =0;
    clrscr ();
    Printf ("Input array of location :");
    for (i=0; i<10; i++)
      scanf ("%d", &a[i]);
    Printf ("Input element to be searched
            from the array");
    scanf ("%d", found);
    for (i=0; i<10; i++)
```

```
if (a[i] == found)
printf ("Element %d is found at ~~the~~
        location for %d, found i++);
~~break~~; yes = 1;
} else if (yes==0)
Printf ("Element not found");
}
~~break~~ getch()
return(0);
}
```

☆ Binary search

3 cases

① data < a [mid] then low = m+1;

② data > a [mid] then high = mid-1;

③ data = a [mid] ~~then~~ element is found

```
# include <stdio.h>
# include <conio.h>
int main()
{
    int a[10]; i, search, low, high, mid;
    clrscr();
    Printf ("Input sorted array elements");
    for (i=0, i<10; i++)
    scanf ("%d, & a[i]);
    Printf ("Input element to be searched");
```



```
scanf ("%d", & search);
low = 1; high = 10;
mid = (low + high)/2;
while (low <= high)
{
    if (a[mid] == search)
    printf ("Element found");
    if (a [mid] < search)
        low = mid + 1;
    if (a [mid] > search)
        high = (mid-1);
    mid = (low+high)/2;
}
```

☆ Sorting

Bubble Sort

```
# include <stdio.h>
# include <conio.h>
int main()
{
    int a[20] [i] temp n;
    clrscr();
    Printf ("Input size of array");
    scanf ("%d" &n);
    Printf ("Input array element");
    for (i=0; i<n; i++)
    scanf ("%d" 2a[i]);
```

Page No.
Date

```
Printf ("sorted array is ")
for (i=0, i<n, i++)
printf ("%d", a[i])
get ch()
return (0)
}
```

☆ Array of characters

String.h

strlen () - to find the length
strcpy () - copies on string over another
str cat () - concatenate two string
strcmp () - compare two string

- String - A string is nothing but a collection of characters in a linear sequence

① strlen ()

```
char string [20] = "Good"
n = strlen (string)
```

② strcpy ()

```
strcpy (str1, str2)
char str [ ] = one,
char str [ ] = two,
```

Page No.
Date

③ strcat ()

```
strcat (string1, string2)
```

Struct	Union
① The struct keyword is used to define a structure	The union keyword is used to define a union
② Each variable member occupied a unique memory space.	variables members share the memory space of the large size variable.
③ Each variable member will be assessed at a time	Only one variable member will be assessed at a time
④ we can initialize ~~with~~ multiple variable of a structure at a time	In union only the first data member can be initialized.
⑤ It is used to store different data type values	It is used for storing one at a time from different data type values.
⑥	

Page No.
Date : / /

Multidimentional array : 2D array

data type - name

```
# include <stdio.h>
# include <conio.h>
int main ()
{
  int a[2] [3] , i, j;
    clrscr ();
    printf (" Input element of 2-D Array");
      for (i=0; i<2; i++)
    {
      for (j=0; j<3; j++
    {
      scanf ("%d", &a[i], [j] );
    }
    }
  // Display output
  for (i=0 ; j<2 ; i++)
  {
    for (j=0; j<3; i++)
    {
      Print f ("%d , a [i] [j]);
    }
    }

      getch ();
      return (0);
  }
```

Page No.
Date : / /

Input element of 2-D Array
1 2 3

i = 0 T	i = 0	a [0] [0] = 1
i = 0	i = 1	a [0] [1] = 2
i = 0	i = 2	a [0] [2] = 3
i = 1	j = 0	a [1] [0] = 4
i = 1	j = 1	a [1] [1] = 5
i = 0 I	j = 2	a [1] [2] = 6

Transpose matrix

```
a[10] [10], transpose [10] [10] r, c,

Print f ("Input no. of rows");
scan f ("%d" & f );
Print f ("Input no. of columns");
scanf ("%d" & c);
```

Page No.
Date : / /

Function

```
# include <stdio.h>
float salary (void);
int main ()
{
  float payment ;
  char ch;
  Payment = Salary ();
  Printf ("\n Payment = % F", Payments.),
  Printf ("wanted to calculate again (Y/N)")
  Scanf ("% d", & ch );
  if (ch == y);
  {
    Salary ();
  }
    getch ();
    return (0);
  }
Float Salary (void);
{
  int HRA , DA, TA :
  Float BS, S;
printf ("\n Enter data to calculate salary")
Scanf ("%d %d %d", & HRA, & DA, & TA);
Scanf ("% f & BS );
  S = BS + HRA + DA + TA;
  return (S).
  }
```

Page No.
Date : / /

```
# include <stdio.h>
void Brazil ();
void Russia ();
void India ();
void China ();
main (); function calling
{
Printf (" we are BRIC nations \n");
  Brazil (); function called
  Russia ();
  India ();
  China ();
  return (0);

void Brazil ()
{
  Printf ("% n Brazil, B ic BRIC");
}
  void India ()
{
  Printf ("\n Indian I in BRI C ");
void China ()(fun defination)
  Printf ("\n China, C in BRI C ");
}
```

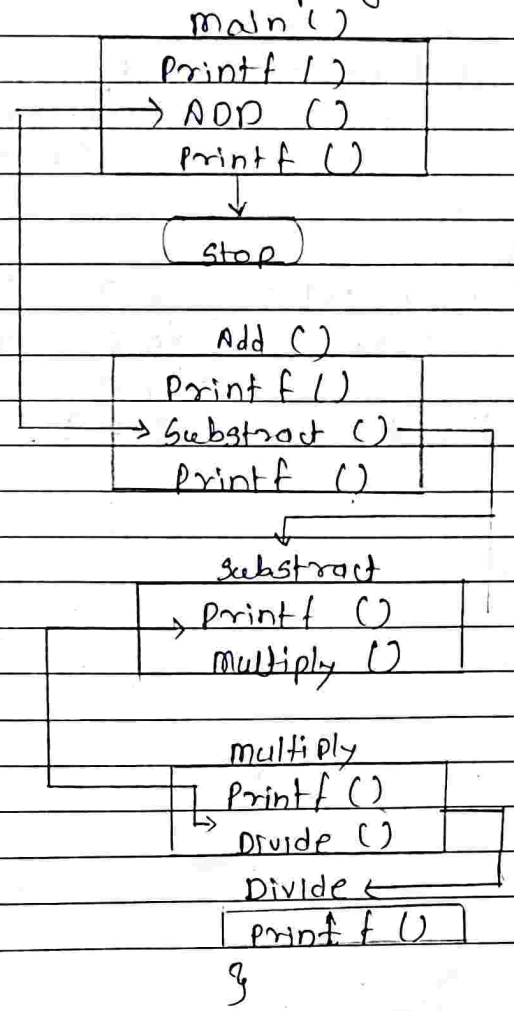
For function prototype

```
# include <stdio.h>
void add ();
void substract ();
void multiply ();
void divide ();
void main ();
{
 Printf ("Arithmatic operations");
 add:
 Printf ("Calculation are over");
 return 0;
}

void add ()
{
 Printf ("This is addition calculation");
 Substract ();
 Printf ("Back in addition calculation");
}

 void Substract ()
{
 Printf ("\n This is Substraction
          Calculation")
 multiply ();
}
 void multiply ()
{
 Printf ("\n This is multiplication
          calculation");
```



```
 divide ();
}
 void divide ()
{
 Printf ("\n This is division calculation");
}
```

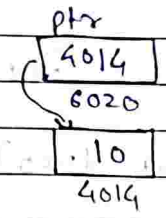
Flowchart of program



Page No.
Date :

Recursion

int * ptr, n=10
ptr &n



* ptr = 10

: function calling itself is called recursion

Recursive function to find factorial of a number.

```
# include <stdio.h>
# include<conio.h>
int fact (int);
main ()
{
  int n, f;
Printf ("Enter n value")
Scanf ("%d", &n);
f = fact (n);
Printf (" Factorial of %d is %d", n, f);
getch ();
return (0);
}

int fact (int n)
{
  int res;
 if (n==0) /* Base condition */
 {
  res = 1
 }
```

```
else
{
 res = n * fact (n-1);
}
return (res),
}
```

* Recursive function to print first n terms of fibonaci series.

```
#include <stdio.h>
# include <conio.h>
 int fib (int);
 int main ()
 {
   int n, i, t;
 printf ("\n Enter value of n:");
 scanf ("%d", &n);
 for (i=0; i<=n; i++)
 {
   t = fib (i);
   printf ("\n %d", f);
 }
 getch ();
 return (0)
 }
 int fib (int n)
 {
   if (n==0)
   return (0);
```

```
}
else
{
   if (n==1)
   return (1);
}
else
{
   return (fib (n-1) + fib (n-2));
```

① Enumeratore
② Typecasting
③ Typedef
④ Bit field operator
⑤ Volatile Qualities

Enumeration

① enum keyword in used to declare the enumerator ② user defined datatype.
for example,

```
enum week { ~~Sday~~ Sunday Monday, Tuesday,
wednesday, thursday, friday saturday };
```

```
int main ()
{
  enum week today,
  today = wednesday
  printf (" Day %d, today+1);
```

Syntax

```
enum    enumtag
{
    enumeration list

}enum variable list; enume
    enum    enum.type enume variable int;
```

```
# include <stdio.h>
enum permission
{
0001        EXECUTE = 1 ;
0010        WRITE   = 2 ;
0100        READ    = 4 ;
0111 = 7 3;
int main ()
{
    int permission = EXECUTE | WRITE | READ;
    Printf (" %d ", permission);
};
```

② Type casting

2 type

Implicit (compiler)	Explicit
Compilation	Programmer
x = 8 bit	float x = 3.435 ;
y = 16 bit	int y = (int) x ;
	y = 3

Case 1: int = 10 , b = 3 ;
float c = a/b ;
10/3
= 3.33

Case 2: int a=10 , b=3 ;
float c=(float) a/b ;

Case 3: int a=10; b=3 ;
float c = (float) (a/b);

3) Typedef

Typedef datatname newname ;

```
typedef int money;
    int a;
    money b;
typedef money coin;
    coin c';
```

ex Typedef int *ptr;
ptr P1;
ptr *P2;

P1 - is pointer to integer entry
P1 is pointer to pointer
P2 - is pointer to pointer integer

* Bit field operator

syntax datatype variable name ~~bitng~~ length in bits

usingned gender 2;

Bit field	Range
1	0-1
2	0-3
3	0-7

Program

```
start struct
{
    usingned r.NO 7;
    usingned ~~numb~~ marks 7;
    usingned 2;
}   st = {34, 98};
void display ()
main ()
{
    int n, m;
```



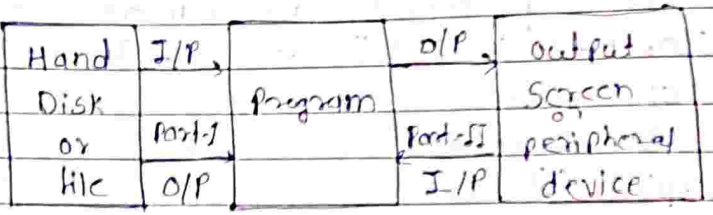
```
display ()
printf ("\n Enter roll no. & marks ");
scanf ("%d%d", &r, &m);
st r.NO = r;
st. marks = m;
display ();
return (0);
}
void display ()
{
printf ("\n Roll NO = %d \n marks = %d",
st r.NO, st marks);
}
```

Output :-
Roll NO = 34
Marks = 98
Enter Roll NO; marks
r = 39, m = 82
Roll NO. = 39
marks = 82

* volatile qualifier

- volatile constant data;

• File Handling in C



IMP

Name of function	Explanation
1) fopen ()	opes the file if already exist.
2)	otherwise creates new file.
2) f close ()	closes the file
3) f gets ()	Read a storing from file.
4) f puts ()	Write a string to file
5) f get c ()	Read a characters from a file.
6) f put (c)	Write a characters to a file
7) f seek (c)	Go to a specified bytes in a file
8) f tell (c)	Tells the current position of file.
9) f scan f (c)	Read data from file
10) f print f (c)	Write data to a file

11) f eof (c)	check if end of the file occured
12) f err (c)	check if error occured
13)	uses the file.
13) f rewind (c)	~~Read a~~ Reset a cursor to the begning of file.
14) Remove (c)	Remove the file.

Keyword or ~~Method~~ macros	file I/O

Macro name	Explanation
1) NULL	Define a null printer
2 Eof	It function tries to read but reached end of the file. file then return.
3)	
3) FOPEN-MAX	open 8 files.
4) SEEK - COR	set cursor to starting of file.
5) SEEK - CUR	present position
6) SEEK - END	set the cursor at the end of file.

- opening a file

Syntax :-

pointer = fopen (char* file name, char* mode);

Ex- FILE
fp = fopen ("Binary.text", "r");

Program

```
# include <stdio.h>
# include <stdlib.h>
void main
{
  FILE *fp;
  fp = fopen ("Binary text", "r");
  if (fp == NULL)
  {
    puts ("can't open file");
  }
  exit (1);
}
```


* Closing a file

Syntax :-
fclose (file pointer);
fclose (fp);

- Writing a character to a file

fput (c)
Syntax :-
fputc (ch, * fp);

- Reading a character from file
fget (c)
Syntax :-
ch = fgetc (* fp);

Write a program to read data from keyboard, write it to a file called input, again read the same data from the Input file and display it on the ~~file~~ screen.

```
# include <stdio.h>
int main
{
  FILE *fp  char c;
  fp = fopen ("INPUT", "w");
  while ((c = get char)) != EOF )
```

```
put c (c, fp);
fclose (fp);
   fp = fopen ("Input", "r");
While (c(c = get c (fp)) != EOF)
{
  printf ("%c", c);
}
fclose (fp);
}
```

Q Write a C program to copy to contain of one file to another file

```
# include <stdio.h>
# include <stdlio.h> // exit fund
int main ()
{
    FILE * fp1, * fp2;
    char c;
    Printf ("enter the file name to open
            for reading");
    Scanf ("%s", filename);
    fp1 = fopen (filename, "r");
    if (fp1 == NULL)
    {
      printf ("scanf open file %s \n", filename);
      exit (0);
    }
    printf ("Enter the filename to open for
            writing");
```

```
scanf ("%s", filename);
fp2 = fopen ("filename, "w");
if (fp2 == NULL)
{
Printf ("cannot open file %s \n", filename);
exit (0);
}
c = fgetc (fp1);
while (c != EOF)
{
   fputc (c, fp2);
   c = fgetc (fp1);
}
Printf ("\n contains copied to %s filename);
fclose (fp1);
fclose (fp2);
return (0);
}
```

Q Write a C program to merge the contains of two file into a third file.

```
# include <stdio.h>
# include <stdlio.h>
int main ()
{
FILE * fp1, *fp2, *fp3;
char ();
fp1 = fopen ("file1. text, "r");
fp2 = fopen ("file2 text, "r");
```

```
fP3 = fopen ("file3.text", "w");
if (fP1 = NULL || fP2==NULL fP3 = NULL)
{
Puts ("could not open files");
exit (0);
}
while (c=fgetc (fP1)) != EOF
{
fputc (c, fP3);
}
while (c=fgetc (cf2)) != EOF)
{
fputc (c, fP3);
}
printf ("magnet file 1.text and file2.text
into file3.text");
fclose (fP1);
fclose (fP2);
fclose (fP3);
return (0);
}
```

* file handling function
fputs() and fgets()
syntax :-
1) fputs (char * str, FILE * fP);
2) * fgets (char * str, int length, FILE * fp);
3) ~~get~~ feof()
syntax :
feof (fP);
This returns true if eof has occurred or returns zero.

4) rewind()
syntax :
rewind (fP);

Q Program to read data from keyboard and write it to file and again read the same bringing using rewind functh.

```
# include <stdio.h>
# include <
# include <string.h>
void main
{
char str [50]; FILE fp;
fp = fopen ("rewind", "wt"); // read/write
if (fp==NULL)
{
printf ("can't open file");
exit (1);
```

```
{
printf("\n Input * string");
    gets (str);
fputs (str, fp);
rewind (fp);
while (! feof (fp))
    gets (str, 50, fp);
        puts (str);
    fclose (fp);
}
```

5) ferror ()
tells about error returns false.

6) remove ()
destroys the specified file
remove ~~file~~ (fp);

7) fflush ()
fflush (stdin);
stdin - keyboard buffer

8) fread () and fwrite ()
syntax :
size of (void * ptr, num bytes, total count, * fp);
size_t fwrite (void * ptr, num bytes, total count * fp);

9) fseek
Syntax
fseek (fp, num bytes, start position)

start position

SEEK - SET — set counter at the begninning
SEEK - COR — set cursor at current position.
SEEK - END — set cursor at the end.

1) fseek (fp, 5, SEEK_SET);
2) fseek (fp, -5, SEEK_END)
fseek (fp, 3*, size of (struct str SEEK.SET),

10) ftell ()
Syntax
ftell (fp),
This tells the current position of file
return -1 if current position not located.

11) fprintf() and fscanf ()
Syntax ;
fprintf(fp, control statement)
fscanf (fp, control statement)

* Command line Arguments

```
# include <stdio.h>
  main (int argc , char * argc [] )
  {
    int i;
    if (argc ==1)
    printf ("No command line argument\n");
  }
  else
  {
    printf (" List of command line argement\n");
    for ( i=1; i<argc; i++ )
    {
    printf ("%s \n" argv [] );
    }
```

```
# include <dish>
# include <stdio.h>
int main (int argc, char* argv [] )
      char * s1 , *s2 ; int x , y ,
      if (argc >2)
  {
      s1 = argv [1];
      s2 = argv [2];
      x = a to c (s1 );
      y = a to c (s2);
      printf ("sum %d \n", x+y );
```



```
  }
  else
  {
    printf (" Insufficient input value\n");
  }
```