DiSHA

COMPUTER INSTITUTE

(C NOTES)

(SANT TUKARAM NAGAR BRANCH)

History of C Language

C Language is developed by "Dennis Ritchie " at Bell Labs in 1972.
C language is a case sensitive language.
Ex. SUM, sum, Sum
These variables are look like same but systems have a different meaning of them.

Programing Languages are divided into two types

1 **High Level Language**
The high level language is designed to get faster programing development.

2 **Low Level Language**
The low level language is designed to get faster program execution.

The C Language is a combination of both these language therefore it is called a "Middle level language"

Structure of C Program

```
Documentation section
Preprocessor Directive
Definition section
Global Declaration section
Main () Function Section
{
Variable declaration/initialization;
Program statements;
}
Subprogram section

Function1 ()
.
.
.
Function ()
```

Documentation sections consist of comments of the program.

- Preprocessor Directive which links the compiler to the standard Library functions.
- Definition section defines all symbolic constants like #define PI=3.14
- Global Declaration section declares the global variables. These variables can be use more than one function. And they declare outside of all functions.

- main () function is used by c program to tell the compiler where the program start. Every program must have exactly one main()
- The opening '{' and closing '}' braces indicate the begin and end of the main function
- C program may contain zero or more variable declaration and statements.
- Subprogram section contain all user defined functions

Sample of C program

```
/* A c program*/
#include<stdio.h>
main ()
{
Printf ("hello world");
}
```

Header Files

There are following header files to store the standard library function

- **#include<stdio.h>**
  This header file is a standard input output header file. It include the following functions
  Printf () – write argument to stdout format
  Scanf () – Read arguments from stdin in specified format
  fopen () - open the file
  fclose () – close the file

- **#include<conio.h>**
  This header file is a console input output header file. It includes the following functions- getch() and clrscr() functions

Comments in C

Single Line comment is given by a //
Multiline comment is given by /* - - - - -*/

Tokens in C

The smallest individual units in a program are called tokens.

1. **Keywords**

C language has 32 keywords .All are in lowercase letter. The keywords are

auto	extern	sizeof	break	float	static
case	for	struct	char	goto	switch
const	if	typedef	continue	int	union
default	long	unsigned	do	register	void
double	return	volatile	else	short	while
enum	signed				

2. Datatype

Datatype specifies the size and type of the value stored in that variable.

- **Built in data types**

Datatype	**Description**	**Memory requirement**	**Conversion specifier**
Int	Stores whole number	2/4 bytes	%d
Char	Single character	1 byte	%c
Float	Contains fractional part	4 bytes	%f
Double	Double precision floating number	8 bytes	%ld
Bool	Single bit(true or false)	1 bit	-
Void	Not return anything	-	-

- **User Defined datatypes**

The user defined datatypes are union ,structure,typedef,enum.

typedef:-

It represents the type definition that allows the user to define an identifier that would represent the existing datatype

Syntax:

typedef type identifier;

Ex: typedef int item_qty;

Item_qty nut,bolt;

The above ex have item_qty declares as int variable.

enum:-

An enumeration is a list of constant values.

Syntax:

enum identifier {value1, value2,….., valuen};

Ex. enum color {green, blue, red, pink};

enum color flowers, leaves;

flowers=red;
leaves=green;
compiler automatically assigns the integer values beginning with 0 to all enum constants. Enumeration constants can also explicitly assign.

3. Variables

A variable is used to store a data value. It is symbolic name assigned to the memory location where data is stored.

Syntax:

datatype variable_name;

Ex. int a=7;

One or more variables can define by using commas (,).

Ex: int a, b, c;

4. Constants

Syntax:

const datatype variable name=value;

Constants are the identifiers that have a fixed value. There are 4 types of constants

I. Integer constants

It consist of sequence of digits.it have +ve or -ve value of constant from 0 -9.

Ex. -3, 5, 9 etc.

II. Floating point constants

It consists of sequence of digits and having a decimal point. It may be +ve or -ve.

Ex. -9.8, 5.6, .8.0 etc

III. Character constants

It is enclosed in single quotation marks. An integer must be char, special symbols like #, @, $ must be a character.

Ex. 'A','/','#' etc

IV. String constants

It is a sequence of characters enclosed in double quotation mark.

Ex."hello", "23/04/08","(a+b)", etc

5. Operators

I. **Arithmetic operators**

Operator	Meaning
+	Addition
-	Substraction
*	Division
/	Multiplication
%	Modulus(Remainder)

II. **Relational Operators**

Operator	Meaning
<	Is less than
>	Is greater than
<=	Is less than equal to
>=	Is greater than equal to
==	Is equal to
!=	Is not equal to

III. **Logical Operators**

Operator	Meaning	Type
&&	Logical AND	Binary operator
||	Logical OR	Binary operator
!	Logical NOT	Unary operator

Iv. **Unary operators**

Operator	Meaning
-	Unary minus
+	Unary plus
++	Increment
--	Decrement
~	Bitwise complement
&	Address of
*	Indirection
!	Logical not

v. **Increment and Decrement operators**

Operator	Meaning
++m	The prefix unary plus operator .It adds 1 to operand and then assigns result to variable on left
m++	The posfix unary plus operator assigns result to the variable on left then adds 1 to the operand
--m	The prefix unary minus operator .It substract 1 to operand and then assigns result to variable on left
m--	The posfix unary plus operator assigns result to the variable on left then adds 1 to the operand

vi. **conditional operator**

It is used to check conditions and depending on result the statement is executed.
Syntax:
exp1? exp2:exp3;
Ex.

```
int a=10,b=20;
X=(a>b)?a:b;
```

Program for addition of two numbers

```
#include<stdio.h>
Void main()
{
        int a,b,c;
        printf("\n enter the value of a and b");
        scanf("%d\t %d",&a,&b);
        c=a+b;
        printf("addition is=%d",c);

}
```

Program for swapping of two values without using third variable

```
#include<stdio.h>
void main()
{
        int a=5;
        int b=2;
        printf("a=%d\t b=%d\n",a,b);
        a=a+b;
        b=a-b;
        a=a-b;
        printf("After swapping \n a=%d \t b=%d\t b=%d",a,b);
}
```

Type casting

converts the value of one type into another type

Syntax:

(type)expression;

Ex. X=(int) 10.456; //it converts to 10

```
int a=34;
            Z= (float) a; //it converts to 34.0000000
```

Program for type casting.

```
#include<stdio.h>
void main()
{
float num=8.45;
printf("value of num=%f \n",num);
printf("value of num on type casting=%d\n",(int)num);
}
```

Character input output functions

The stdio.h header file provide following functions for Input and output

I.getchar() function

This function accepts a character from keyboard.

Syntax:

Character_variable_name=getchar();

II.putchar() function

This function is used to print one character on screen.
Syntax:
Putchar(Character_variable_name);

Program to accept character from keyboard and print it.

```
#include<stdio.h>
void main()
{
	char c;
	c=getchar();
	putchar(c);
}
```

String input output functions

I.gets() function

This function is used to accept string from the keyboard.

Syntax:

gets (string_variable_name);

II.puts() function

This function is used to print string on screen

Syntax: puts(string_variable_name);

Program to accept a string from keyboard and print it.

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

```
        char s[80];
        gets(s);
        puts(s);
}
```

Formatted console input output functions

The stdio.h header file provide built in functions for reading and writing formmated data from input output devices.

I.scanf() function

This function accept formmated data from keyboard.

Syntax:

scanf("control string",&var1,&var2,……);
The control strinjg consist of

- White space character(\t,\n)
- Coversion specifire characters as

Conversion specifire	Meaning
%c	For character
%d	For decimal integer
%f	For folating point value

Ex.int a;

scanf("%d",&a);

II.printf() function

It is used to print formatted data from keyboard where format of string is depends onto the control string.

Syntax: printf("control string",var1,var2,……varn);

Here var1,var2 ….are the parameters of the function that means the variable names showing the address of memory location

The control string consist of ordinary string, conversion specifire, format specifire

Ex. printf("the no is %d:",a);

Program for swapping of two numbers.

```
#include<stdio.h>
void main()
{
        int a,b,t;
        printf("\nenter value of a and b");
        scanf("%d\t%d",&a,&b);
        t=a;
        a=b;
        b=t;
        printf("value after swapping=%d \t b=%d",a,b);
}
```

Program for calculate area

```
#include<stdio.h>
void main()
{
int l,b;
float area;
printf("\n enter the value area of traingle");
scanf("%d%d",&l,&b);
area=0.5*l*b;
printf("%f ",&area);
}
```

control Structure/Decision making statement

The control statement are used to control the flow of the program.
Decision making is to see the particular condition is occurred or not and direct the compiler to execute other statement.
There are two ways of decision making statement and is used with expression

1. **If statement:**

   It is two way decision making statement depending on the expression is true or false.

   Syntax:

```
 if(expression)
{
    Statement block;
}
Statement x;
```

   Program for if statement

```
#include<stdio.h>
void main()
{
    int a;
    printf("\n enter the value of a");
    scanf("%d",&a);
    if(a>0)
```

```
{
        printf("no is greater");
}
printf("no is less");
    }
```

2. If else statement:

It is allows to selecting one of two available options depending upon the output of the test expression.

Syntax:
```
if(expression)
{
        True block statement;
}
else
{
    False block statement;
    }
Statement x;
```

Program for number is even or odd

```
#include<stdio.h>
void main()
{
        int a;
printf("enter value of a");
scanf(%d",&a);
if(a%2==0)
{
    printf("\n the %d no is even",a);
}
else
{
    printf("\n the %d no is odd",a);
}
}
```

Program for no maximum or minimum

```
#include<stdio.h>
void main()
{
int a,b;
printf("enter value of  a and b");
scanf(%d\t %d",&a,&b);
if(a>b)
{
    printf("\n the %d no is maimum",a);
}
else
{
```

```
        printf("\n the %d no is maximum",b);
    }
    }
```

Program for leap year

3. Nesting if elsete statement

This allows us to perform multiple actions in single instruction.

Syntax:

```
 If (expression1)
{
   If (expression2)
   {
         Statement 1;
   }
   Else
   {
         Statement 2;
   }
}
Else
{
   Statement 3;

 }

    Statement x;
```

Program to find out largest number out of four numbers.

```
#include<stdio.h>
Void main()
{
        int a,b,c,d;
        printf("\n enter the 4 numbers");
        scanf("%d%d%d%d",&a,&b,&c,&d);
        if(a>b)
        {
                if(a>c)
                {
                        if(a>d)
                                Printf("\n maximum number=%d",a);
                        else
```

```
                printf("maximum number =%d",d);
        }
        else
        {
                if(c>d)
                        printf(\n maximum number=%d",c);
                else
                        printf(\n maximum number=%d",d);
        }
}
else
{
        if(b>c)
        {
                if(b>d)
                        printf("\n maximum number=%d",b);
                else
                        printf(\n maximum number is=%d",d);
        }
        else
        {
                if(c>d)
                        printf("\n maximum number=%d",c);
                else
                        printf("\n maximum number=%d",d);
        }

}
```

Program for maximum of three numbers .

```
#include<stdio.h>
void main()
{
  int a,b,c;
  printf("\n Enter the value of a:");
  scanf("%d",&a);
  printf("\n Enter the value of b:");
  scanf("%d",&b);
  printf("\n Enter the value of c:");
  scanf("%d",&c);
  if(a>b)
  {
        if(a>c)
        {
          printf("\n a is maximum");
        }
        else
```

```
        {
          printf("\n c is maximum");
        }
  }
  else
  {
        if(b>c)
        {
          printf("\n b is maximum");
        }
        else
        {
          printf("\n c is maximum");
        }
  }
  getch();
}

}
```

4.Else if ladder:

Syntax:

```
if(expression 1)
Statement 1;
    else if(expression 2)
       Statement 2;
    else if(expression 3)
         Statement 3;
     ....
    else
    Statement x;
```

Program for elseif ladder

```
#include<stdio.h>
void main()
{
char n[40];
int m1,m2,m3,m4,m5,m6,t;
float per;
printf("\nEnter name of the student:");
scanf("%s",&n);
printf("\nEnter marks in 6 subjects:");
scanf("%d%d%d%d%d%d",&m1,&m2,&m3,&m4,&m5,&m6);
t=m1+m2+m3+m4+m5+m6;
per=t/6;
if((m1<40)||(m2<40)||(m3<40)||(m4<40)||(m5<40)||(m6<40))
printf("\n\nName:%s\nTotal:%d\nGrade:Fail",n,t);
else
```

```
if(per>=75)
printf("\n\nName:%s\nTotal:%d\nGrade:Distinction",n,t);
else
if(per>=60)
printf("\n\nName:%s\nTotal:%d\nGrade:First Class",n,t);
else
if(per>=50)
printf("\n\nName:%s\nTotal:%d\nGrade:Second Class",n,t);
else
printf("\n\nName:%s\nTotal:%d\nGrade:Third Class",n,t);
}
```

5.Switch statement:

A multi way decision making known as switch. A case expression can repeatedly use in switch statement. The use of break statement in every case to quit the switch statement after matched.

Syntax:

```
switch(expression)
{
    case value-1:
                Statement 1;
                break;
    case value-2:
            Statement 2;
            break;
        ….
        \’’’
    default:
            default block;
            break;
    }
Statement x;
```

Program for switch case

```
#include<stdio.h>
void main()
{
        int month;
        printf(“\n enter  the month number”);
        scanf(“%d”,&month);
        switch(month)
        {
                case 1:printf(“january”);
                                break;
                case 2:printf(“February”);
                                break;
```

```
            case 3:printf("march");
                                break;
            case 4:printf("april");
                                break;
            case 5:printf("may");
                                break;
            case 6:printf("june");
                                break;
            case 7:printf("july");
                                break;
            case 8:printf("august");
                                break;
            case 9:printf("september");
                                break;
            case 10:printf("octomber");
                                break;
            case 11:printf("november");
                                break;
            case 12:printf("december");
                                break;
            default: printf("wrong input");
                                break;
        }

}
```

Program for switch case addition,multiplication,division,modulo

```
#include<stdio.h>
 #include<conio.h>
 void main()
 {
 int a,b,c,ch;
 printf("enter the value a&b");
 scanf("%d%d",&a,&b);
 printf("\n\t1addtion");
 printf("\n\t2substraction");
 printf("\n\t3multiplication");
 printf("\n\t4division");
printf("\n\t 5modulus ");
 printf("enter any choice");
 scanf("%d",&ch);
 switch(ch)
 {
 case 1:
 c=a+b;
 printf("addtion is %d",c);
 break;
 case 2:
 c=a-b;
```

```
 printf("substraction is %d",c);
 break;
 case 3:
 c=a*b;
 printf("multiplication is %d",c);
 break;
 case 4:
 c=a/b;
 printf("division is %d",c);
case 5:
 c=a%b;
 printf("division is %d",c);

 }
 getch();
 }
```

Program for color choices

```
#include<stdio.h>
#include<conio.h>
void main()
{
int color=1;
printf("\n please choose a color number");
printf("\n 1:red\n2:green\n3:blue");
scanf("%d",&color);
switch(color)
{
case 1:
printf("\n you choose red color");
break;
case 2:
printf("\n you choose green color");
break;
case 3:
printf("\n you choose blue color");
break;
default:
printf("\n you did not choose any color");
break;
}
getch();
}
```

Decision control loops:

1. While loop:

It is a entry controlled loop. The test condition is executed if it is true then the body of loop is executed this process is continue until the test condition is false

Syntax:

```
initialization;
While(condition)
{
    Body of loop;
    Increment/decrement;
}
Statement x;
```

Program for print 1 to 10 numbers using whuile loop

```
#include<stdio.h>
void main()
{
        int a;
            a=1;
            while(a<10)
            {
                    printf("%d",a);
                    a=a+1;
            }
}
```

Program for factorial no

```
#include<stdio.h>
Void main()
{
      int f=1,n,i=1;
printf("enter the no");
scanf("%d",&n);
while(i<=n)
{
      f=f*i;
      i++;
}
printf("factorial no is=%d",f);
}
```

Program for fibbonancci series

```
#include<stdio.h>
Void main()
{
      Int a=0,b=1,c=0,is=0,n;
      Printf("enter the range");
```

```
        scanf("%d",&n);
        While(i<=n)
        {
                printf("fibbonancci series is %d",c);
                a=b;
                b=c;
                c=a+b;
i++;
        }
}
```

Program for sum of given numbers

```
#include<stdio.h>
void main()
{
int n,a,i=1,sum=0;
printf("\how many number:");
scanf("%d",&n);
while(i<=n)
{
printf("\nEnter any number:");
scanf("%d",&a);
sum+=a;
i+=1;
}
printf("\nSum=%d",sum);
getch();
}
```

Program for sum of five digit number

```
#include<stdio.h>
#include<conio.h>
void main()
{
int no,rem=0;
long int sum=0;
clrscr();
printf("enter five digit number");
scanf("%d",&no);
while(no!=0)
{
rem=no%10;
sum=sum+rem;
no=no/10;
}
printf("sum of digits= %d",sum);
getch();
}
```

2. do-while loop:

it is exit controlled loop. The body of loop statement is executed for the first time. Then the test condition in while loop is executed. If the condition is true then the body of loop is executed until the condition becomes false.

Syntax:

```
initialization;
do
{
    Body of loop;
}while(condition);
```

Program for do while loop

```
#include<stdio.h>
void main()
{
    int a=1;
    do
    {

            printf("%d",a);
            a++;

    }while(a<10);

    }
```

Program for numbers in decending order

```
#include<stdio.h>
#include<conio.h>
void main()
{
int x=20;
do
{
printf("\n %d",x);
x=x-1;
}while(x>=1);
getch();
}
```

Program for reverse number

```
#include<stdio.h>
#include<conio.h>
void main()
{
int n,d,resno=0;
clrscr();
printf("enter no");
```

```
scanf("%d",&n);
do
{
d=n%10;
n=n/10;
resno=resno*10+d;
} while(n>0);
printf("reverse no=%d",resno);
getch();
}
```

3.for loop:

It is also entry controlled loop.

syntax:

```
 for(initialization;test condition;increment/decrement)
{
Body of loop;
}
```

Different ways of using for loops :

```
for(p=1;n=0;n<4;++n)

for(p=1;n=0;n<4;++n;++p)

for(;n<4;)

for( ; ; )
```

program for for loop

```
#include<stdio.h>
void main()
{
    int i;
    for(i=1;i<10;i++)
    {
            printf("%d",i);
    }
}
```

4.nested for loop:

syntax:

```
 for(initialization;test condition;increment/decrement)
{

    for(initialization; test condition;increment/decrement)

    {
```

```
        Statement of loop;;

    }

    Statement of outer loop;

    }
```

Program for nested for loop

```
#include<stdio.h>
void main()
{
    int i,j;
    for(i=0;i<=4;i++)
    {
        for(j=0;j<=i;j++)
        {
            printf("*");
        }
        printf("\n");
    }
}
```

Program for nested for loop

```
#include<stdio.h>
void main()
{
    int i,j;
    for(i=0;i<=4;i++)
    {
        for(j=0;j<=i;j++)
        {
            printf("%d",i);
        }
        printf("\n");
    }
}
```

Program for nested for loop

```
#include<stdio.h>
void main()
{
    int i,j;
    for(i=0;i<=4;i++)
    {
        for(j=0;j<=i;j++)
```

```
		{
			printf(“%d”,j);
		}
		printf(“\n”);
	}
}
```

Program fort nested for

```
#include<stdio.h>
#include<conio.h>
void main()
{
int i,j;
clrscr();
for(j=72;j>65;j--)
{
for(i=65;i<j;i++)
{printf("%c",i);}

printf("\n");

}
}
```

Program for nested for loop

```
#include<stdio.h>
void main()
{
	int i,j;
	for(i=0;i<=4;i++)
	{
		for(j=0;j<=4;j++)
		{
			printf(“*”);
		}
		printf(“\n”);
	}
}
```

Program for factorial numbers series.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int i=0,j=0,n=0;
long int f=1;
```

```
clrscr();
printf("enter range");
scanf("%d",&n);
for(i=1;i<=n;i++)
{
printf("\t");
for(j=1;j<=i;j++)
{
f=f*j;
}
printf("%ld",f);
f=1;
}
getch();
}
```

5.break statement:

it is used to exit from loop before the predetermine condition becomes false.it can also be handle error or any exceptional condition.
The general form are:

While(…….)
{
……
…….
If(condition)
 	
break;
…..
……
}
……
	Do
	{
	……
	…….
	If(condition)
	
	break;
	…..
	……
	}while(…..)
	……
		for(…….)
		{
		……
		…….
		If(condition)
		
		break;
		…..
		……
		}
		……

Program for break statement

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

```
            printf("****");
        }
}
```

6. continue statement

the continue statement causes the loop to continue with the next iteration skipping the remaining statement in that loop.

Do
{
……
 	
if( )
 	
continue;
 	
…….
…….
 	
}while( );
	for(; ;)
	{
	……
	
	if( )
	continue;
	…….
	
	……
	}

Program for continue statement

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

Arrays

Array is a collection of data items with same data type.All data items in array have same name. Array is starts with zero (0).

There are two types of array:

One dimensional array

Syntax: data_type array_name[size/subscript];

Ex: int marks[20];

It stores marks in computer as

		…….	

marks[0] marks[1]……..marks[19]

Initialization:

1.Syntax: arrayname[index]=value;

Ex. Marks[0]=35;

Marks[1]=98;

……

…….

Marks[19]=54;

2. **syntax:** type arrayname[]={ list of values };

Ex. Int marks[10]={34,56,78,44,87,88,99,76,66,55};

3. Declaration and initialization of character array

Ex. char a[5]={'l','e','a','r','n'};

The character array must ends with null character means '\0'.

Program for print 1D array element

```
#include<stdio.h>
#include<conio.h>
void main()
{
int arr[5]={10,15,78,90,100};
int i=0;
clrscr();
for(i=0;i<=4;i++)
{
printf("\n%d",arr[i]);
}
getch();
}
```

Program for print 1D array element

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

```
void main()
{
int a[10],i;
printf("\nEnter any ten numbers\n");
for(i=0;i<10;i++)
{
scanf("%d",&a[i]);
}
printf("\nThe inputed numbers are as follows\n");
for(i=0;i<10;i++)
{
printf("%d",a[i]);
}
}
```

Program addition of 1D array elements

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a[10],b[10],c[10],i;
printf("\nEnter any ten numbers for a\n");
for(i=0;i<10;i++)
{
scanf("%d",&a[i]);
}
printf("\nEnter any ten numbers for b\n");
for(i=0;i<10;i++)
{
printf("%d",&b[i]);
}
Printf("addition is:\n");
for(i=0;i<10
{
c[i]=a[i]+b[i];
scanf("\t%d",c[i]);
}
}
```

Two dimensional array

The array which require two subscript to its individual element is called two dimensional array.

Declaration syntax: dtatatype arrayname[row-size][column-size];

Ex. int matrix[3][3];

Initialization: int matrix[3][3]={2,4,5,5,6,1,8,9,3};

The representation of above array is :

2	4	5
5	6	1
8	9	3

Program for print 2D array element

```
#include<stdio.h>

void main()
{
int a[3][3],i,j;
printf("elements for a");
for(i=0;i<3;i++)
{
for(j=0;j<3;j++)
{
scanf("%d",&a[i][j]);
}
}
printf("elements are-");
for(i=0;i<3;i++)
{
for(j=0;j<3;j++)
{
printf("%d\t",a[i][j]);
}
Printf("\n");

}
}
```

Program for matrix elements addition

```
#include<stdio.h>



void main()
{
int a[3][3],b[3][3],c[3][3],i,j;
printf("elements for a");
for(i=0;i<3;i++)
{
for(j=0;j<3;j++)
scanf("%d",&a[i][j]);
}
printf("elements for b");
for(i=0;i<3;i++)
{
```

```
for(j=0;j<3;j++)
scanf("%d",&b[i][j]);
}
for(i=0;i<3;i++)
{
for(j=0;j<3;j++)
c[i][j]=a[i][j]+b[i][j];
}
printf("addition of two matrix");
for(i=0;i<3;i++)
{
for(j=0;j<3;j++)
{
printf("\n%d",c[i][j]);
}
printf("\n");
}
}
```

Program for diagonal addition

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a[3][3],i,j,n;

printf("enter the array elements");
for(i=0;i<3;i++)
{
for(j=0;j<3;j++)
{
scanf("%d",&a[i][j]);
}
}
for(i=0;i<3;i++)
{
for(j=0;j<3;j++)
{
if(i==j)
{
n=n+a[i][j];
}
}
}
printf("diagonal add is %d",n);
}
```

Program for transpose of matrix

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a[20][20],b[20][20],r,c,i,j;
clrscr();
printf("accept rows and column:");
scanf("%d %d",&r,&c);
printf("accept elements");
for(i=0;i<r;i++)
{
for(j=0;j<c;j++)
{
scanf("%d",&a[i][j]);
b[j][i]=a[i][j];
}
}
printf("\n original matrix is:");
for(i=0;i<r;i++)
{
for(j=0;j<c;j++)
printf("%d",a[i][j]);
printf("\n");
}
printf("\n transpose matrix");
for(i=0;i<r;i++)
{
for(j=0;j<c;j++)
{
printf("\t %d",b[j][i]);
}
printf("\n");
}
}
```

Function

A function is a self-contained block of statement that perform particular task and may return a value to the calling program.

- The function declaration is called function prototype or function call.There is a semicolon at the end of the declaration.

  Syntax: return_datatype function_name(type arg1,type arg2,……..)

Ex. int sum(int a,int b); //it takes two argument and return a value.

void sum(float a,float b) //it takes two argument but no return value

void sum(void); //it takes no argument and no return value

The function that do not return any values can be explicitly defined as void.

- **Function definition**

Syntax: return_datatype function_name(type arg1,type arg2,…..)

```
{
        Local variable declaration;
        Statements;
         return(expression);
}
```

- The first line is a function header.
- There is no semicolon at the ends of header.
- The argument list is the mode of communication between the function and the calling function.
- Local variable declaration is necessary only if you are using any local variable.
- Return keyword:

The return keyword is used to end the function body and return a value to the calling function. A function body may contain multiple return statements but the fun execution is stop at the first return statement.

There are three ways to give return:

Syntax	example
return;	return;
return expression;	return b;
return(expression);	return(a+b);

program for square of the numbers upto given range

```
#include<stdio.h>
#include<Conio.h>
void sqr_prn(int);
void main()
{
 int pos=0;
 clrscr();
 printf("\n Enter last position");
 scanf("%d",&pos);
 sqr_prn(pos);
 getch();
```

```
}
void sqr_prn(int x)
{
 int i=1;
 for(i=1;i<=x;i++)
 {
  printf("%d\t",i*i);
 }
}
```

program for area of triangle using function

```
#include<stdio.h>
void main()
{
	triangle();
}
void triangle()
{
	int base, height;
	float area;
	printf("\n enter base");
	scanf("%d",&base);
	printf("\n enter height");
	area=0.5*base*height;
	printf("\n area of triangle=%f",area);
}
```

Program for addition using function

```
#include<stdio.h>
void main()
{
	int p,q,sum;
	p=10,q=20;
	sum=add(p,q);
	display(sum);
}
int add(int x,int y)
{
	int s;
	s=x+y;
	return(s);
}
Int display(int sum)
{
	printf("addition is=%d",sum);
}
```

Program for arithmetic using function

```
#include<stdio.h>
#include<conio.h>
void main()
{
int p,q,sum;
p=40;q=20;
add(p,q);
sub(p,q);
mul(p,q);
div(p,q);
}
int add(int x,int y)
{
int s;
s=x+y;
printf("\n%d",s);
return(s);
}
int sub(int x,int y)
{
int s;
s=x-y;
printf("\n%d",s);
return(s);
}
int mul(int x,int y)
{
int s;
s=x*y;
printf("\n%d",s);
return(s);
}
int div(int x,int y)
{
int s;
s=x/y;
printf("\n%d",s);
return(s);
}
```

Method of parameter passing:

There are two ways to pass parameters to function:

1. call by value:

In this method the content of the arguments in the calling functions are not changed, even if they are changed in the called function.

The content of actual parameter get copied into corresponding formal parameters.

Program for call by value

```
include<stdio.h>
#include<conio.h>
void swap(int a,int b)
{
int t;
printf("within function calling %d %d",a,b);
t=a;a=b;b=t;
printf("after swaping %d %d",a,b);
}
void main()
{
int x,y;
printf("enter two value");
scanf("%d %d",&x,&y);
swap(x,y);
printf("after function call %d %d",x,y);
}
```

2. Call by reference:

In this method the content of the calling functions get changed i.e. The original values are changed.

Instead of passing the value of a variable, we can pass the memory address of the variable to the function.it is called call by reference.

Program for call by reference

```
#include<stdio.h>
#include<conio.h>
void swap(int *a,int *b);
void main()
{
int x,y;
clrscr();
printf("enter value of x,y");
scanf("%d %d:",&x,&y);
printf("before swapping");
printf("x=%d y=%d",x,y);
swap(&x,&y);
printf("\n after swapping");
```

```
printf("\n x=%d,y=%d",x,y);
getch();
}
void swap (int*a,int*b)
{
int p;
p=*a;
*a=*b;
*b=p;
}
```

Scope of variable

1. Local variable:

By default the scope of variable is local to the function in which it defined. Local variable can only be accessed in the function in which they defined.

2. Global variable:

If the variable is defined outside any function definitions is called as external variable. The scope of such a variables will be for whole program. Hence it is called global variable.

Ex. #include<stdio.h>

```
int a=7;//global variable
main ()
{
int b=3;//local to main
…..
….
}
abc()
{
int c=4;//local to abc()
…
}
```

Recursion:

It is a process by which a function calls itself repeatedly until some specified Cond has been satisfied.

Syntax:

```
        return_type function_name(argument list)
    {
```

```
                ……..
……..
Function_name(……);
……
……
}
```

Program for factorial using recursion

```
#include<stdio.h>
int fact(int p)
{
int f=1,i;
if(p==1)
        return 1;
else
        f=p*fact(p-1);
        return f;
}
void main()
{
        int x,n;
        printf("enter the value of n");
        scanf("%d",&n);
        x=fact(n);
        printf("factorial no=%d",x);
}
```

Program for odd numbers upto n numbers using recursion

```
#include<stdio.h>
void odd(int a,int n)
{
        if(a<=n)
        {
                printf("%d\t",a);
                odd(a+2,n);
                }
                return;
}
void main()
{
        int n;
        printf("\n enter the value of n");
        scanf("%d",&n);
        odd(1,n);
}
```

string

A string is an array of characters stored in consecutive memory locations.

In strings the ending character is always the null character **'\0'** .The null character acts as a string terminator. The compiler automatically stores null character at the end of the string.

Declaration of strings:

Syntax: char string_name[length];

Ex. char name [5];

Initialization:

Char greet[10]={ 'h' , 'e' , 'l' , 'l', 'o'};

Char greet[10]="hello";

Char greet[]="hello";

Memory representation of string:

Char greet[]="hello";

h	e	l	l	o	\0

Standard string handling functions:

The following standard library functions are the standard functions with strings. They are declares in the string's header file.

Function	meaning
strlen(s1)	Returns the length of the string
strlwr(s1)	Coverts the string s1 to lower case
strupr(s1)	Converts the string in upper case
strcat(s1,s2)	It joins the two strings
strcmp(s1,s2)	Compares s1 and s2 and returns –ve if s1<s2, +ve if s1>s2, 0 if s1==s2

strrev(s1)	Reverse the string s1
Strcpy(s1,s2)	Copies the string s2 into s1

Program for string strlen()

```
#include<stdio.h>
#include<string.h>
#include<conio.h>
void main()
{
char name [20]="rutu";
int a;
a=strlen(name);
printf("\n %d",a);
}
```

Program for string strrev()

```
#include<string.h>
#include<stdio.h>
#include<conio.h>
void main()
{
char name[20];
printf("enter string");
scanf("%s",name);
printf("%s",strrev(name));
}
```

Program for string strlwr()

```
#include<stdio.h>
#include<string.h>
#include<conio.h>
void main()
{
char str [20];
int len;
printf("enter string");
gets(str);
strlwr(str);
printf("lower case=%s",str);
```

```
}
```

Program for string strupr()

```
#include<stdio.h>
#include<string.h>
#include<conio.h>
void main()
{
char str [20];
int len;
printf("enter string");
gets(str);
strupr(str);
printf("upper case=%s",str);
}
```

Program for string strcmp()

```
#include<stdio.h>
#include<string.h>
#include<conio.h>
void main()
{
char a[20],b[20];
clrscr();
printf("enter string");
gets(a);
printf("enter second string");
gets(b);
if(strcmp(a,b)==0)
printf("string are equal");
else
printf("string are not equal");
}
```

Program for string strcat()

```
#include<stdio.h>
#include<string.h>
void main()
{
char name[20]="Disha";
char a[]="comp";
strcat(name,a);
printf("\n%s",name);
}
```

Program for string Strcpy()

```
#include<stdio.h>
#include<string.h>
void main()
{
char name[]="Disha";
char a[20]
strcpy(a,name);
printf("%s",a);
}
```

Structure:

Structure is a collection of logically related data items of different data types grouped together under a single name.

Structure is analogous to records. The data items that make up a structure are called its members or fields.

Structure definition:

Syntax:

```
struct structure_name
    {
        Data type member1;
        Data type member2;
        ……..
     };
```

The structure definition template is terminated with semicolon.
The members of the structure are enclosed in {}.
Each data member in structure is declared independently with its name and type in separate statement.
Ex. of book database:

```
struct book{
        char title [15];
        char author [10];
        int pages;
        float price;
};
```

Structure declaration:

The structure declaration in 3 ways:

1. Structure variable declaration in structure template.

Syntax	**example**
Struct structure name {	struct book {

Data type member1; Data type member2; …. }var1,var2….;	char title[15]; char author[10]; int pages; float price; }b1,b2,b3;

2. Structure variable declaration anywhere in the program.

Syntax	**Example**

Struct structure_name
{
Datatype member1;
Datatype member2;
….
};
Struct structure_name var1,var2….;
	struct book
	{
	char title[15];
	char author[10];
	int pages;
	float price;
	};
	Struct book b1,b2,b3;

3. Structure variable declaration in structure template.

Example

struct book
{
char title[15];
char author[10];
int pages;
float price;
};
Struct book b[100];

Structure initialization:

Initialization in a structure template	**Initialization out of structure template**

Ex. struct book
{
char title[15];
char author[10];
int pages;
float price;
}b1{“Let Us c” , “kanetkar”
,300,150.50};
	struct book
	{
	char title[15];
	char author[10];
	int pages;
	float price;
	};
	
	Struct book b1{“Let Us c” ,
	
	“kanetkar” ,300,150.50};

Accessing structure members:
The individual members of the structure can be accessed using dot(.) operator. This dot operator is called as structure member operator as it connects the structure variable and the structure member.

Syntax:

structure variable.structure_member;

Ex.

```
b1.title=”Let Us c”;
b1.pages=300;
printf(“price%f”,b1.price);
scanf (“%s”,b1.author);
```

program for structure

```
#include<stdio.h>
#include<conio.h>+
struct book
{
char title[15];
int pages;
}s={"CPP",300};
void main()
{
printf("title=%s",s.title);
printf("pages=%d",s.pages);
}
```

program for employee information

```
#include<stdio.h>
#include<conio.h>
struct emp
{
char n[80];
char address[200];
long phone;
};
struct emp e;
void main()
{
printf("enter emp name");
scanf("%s",e.n);
printf("enter emp address");
scanf("%s",&e.address);
printf("enter phoneno");
```

```
scanf("%ld",&e.phone);
printf("name is %s",e.n);
printf("address is %s",e.address);
printf("phone is %ld",e.phone);
}
```

program for bank account balance.

```
#include<stdio.h>
#include<conio.h>
void main()
{
struct bank_acno
{
 int acno;
 char mem_name[10];
 float balance;
 char ac_ty[10];
 }b_acc;
 clrscr();
 printf("\n Enter Bank Account details\n");
 printf("\n Enter bank Account Number:");
 scanf("%d",&b_acc.acno);
 printf("\n Enter Account holder's Name:");
 scanf("%s",b_acc.mem_name);
 printf("\n Enter balance");
 scanf("%f",&b_acc.balance);
 printf("\n Enter Account type:");
 scanf("%s",b_acc.ac_ty);
 printf("\n\n BANK ACCOUNT DETAILS\n");
 printf("\n Bank account number=%d",b_acc.acno);
 printf("\n Account holders Name=%s",b_acc.mem_name);
 printf("\n Account type=%s",b_acc.ac_ty);
 printf("\n Balence=%f",b_acc.balance);
 if(b_acc.balance<2000)

 {
 printf("\n Miniumum Balance in your account!!!!!!!!!!!!!!!!!!!");
 }
 getch();
}
```

Union:

Union are also a user defined type they also contain the members of datatypesbut all members in union shares same stoiarage area in computer memory while structure assingned its own unique stoarage area

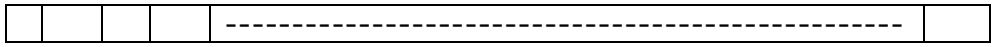
Syntax:

```
        Union union_name
```

```
        {
            Datatype member1;
            ……
            …..
        }var1,var2……;
Ex. union book
    {
        Char title[20];
        Int pages;
    }b1,b2;
```

pointer:

Pointer are the variable that contain the address of another variable .
Every data item is strored in memory in one of the adjacent locations ina cells.each cells is known as byte and has a number that means the address.
memorycells



address 0 1 2 3 ---------- 65535

ex. int x=5;
it stores in memory as:-

X	<---- variable
5	←---- value
1002	<---- address

Address and dereferencing(& and *) operators
Int x=5;
To access the baddress of variable x, the & operator is used it is called "address of " operator.then we assign the address like as:
P=&x;
To access the value of the variable x,then used * operator ,it is called "dereferancing operator"
X=*p;
Here x and *p represents same value. 5

Pointer declaration
Datatype *ptvar;
Here ptvtr is a pointer and it needs memory location and it points to a variable of that datatype.
Ex. int *p;
Float *a;
Char *c;
Pointer initialization
To initialize pointer variable use (&) operator
Ex. p=&a;

Program for pointer

```
#include<stdio.h>
#include<conio.h>
void main()
{
int *p,a;
printf("\n enter the valies");
scanf("%d",&a);
p=&a;
printf("\n *p=%d",*p);
printf("\n p=%x",p);
printf("\n a=%d",a);
}
```

Pointer arithmetic:

increment	Ptr++	It points to the next location of same type ,if ptr=2003 then it gives ptr=2005,if ptr is point to character then ptr=2004
decrement	Ptr--	It points to previous location of same type,if ptr=2003 then it gives ptr=2001
Adding a number to pointer	Ptr=ptr+8	Ptr points to 9 integer locations after current location.if ptr=2003 then ptr+8 gives ptr=2011
Substracting a number from pointer	Ptr=ptr-8	Ptr points to 8 integer before current location. If ptr=2003 then ptr-8 gives ptr=1995

Invalid pointer arithmetic:

Additonh of two numbers
Multiplication of a number with pointer
Division of a pointer with a number

Program for pointer arithmetic

```
#include<stdio.h>
#include<conio.h>
void main()
{
char c='z',*cp;
float f=10.2,*fp;
int i=987,*ip;
long l=345,*lp;
double d=9.99,*dp;
cp=&c;fp=&f;ip=&i;lp=&l;dp=&d;
printf("\n char float int long double");
printf("\n %8x %8x %8x %8x %8x",cp,fp,ip,lp,dp);
cp++;fp++;ip++;lp++;dp++;
printf("\n %8x %8x %8x %8x %8x",cp,fp,ip,lp,dp);
```

```
cp+=12;fp+=12;ip+=12;lp+=12;dp+=12;
printf("\n %8x %8x %8x %8x %8x",cp,fp,ip,lp,dp);
cp--;fp--;ip--;lp--;dp--;
printf("\n %8x %8x %8x %8x %8x",cp,fp,ip,lp,dp);
cp-=12;fp-=12;ip-=12;lp-=12;dp-=12;
printf("\n %8x %8x %8x %8x %8x",cp,fp,ip,lp,dp);
}
```

Pointer to pointer:

A pointer to pointer is a variable that contains the address of pointer variable of specific datatype.

Syntax:

Datatype **ptr_to_ptr;

Ex. int x=12;

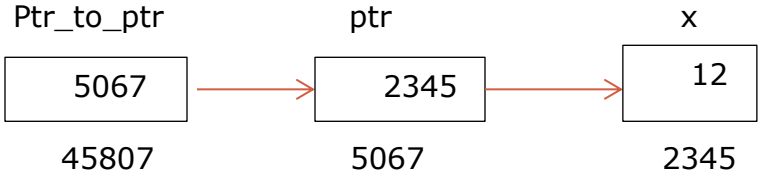
Int *ptr;

Int **ptr_to_ptr;

Ptr=&x;

Ptr_to_ptr=&ptr;

The memory representation:



Program for pointer to pointer

```
#include<stdio.h>
#include<conio.h>
void main()
{
char c='z',*cp,**pcp;
float f=10.2,*fp,**pfp;
int i=987,*ip,**pip;
double d=9.99,*dp,**pdp;
cp=&c;
fp=&f;
ip=&i;
dp=&d;
pcp=&cp;
pfp=&fp;
pip=&ip;
pdp=&dp;
printf("\n DataType   Address_of_pointer Addressof_value values values");
printf("\n         \t    **p \t   *p \t   p");
printf("\n character %8x      %10x   %c",pcp,cp,c);
printf("\n float     %8x      %10x   %f",pfp,fp,f);
printf("\n integer   %8x      %10x   %d",pip,ip,i);
printf("\n double    %8x      %10x   %lf",pdp,dp,d);
```

```
}
```

Program for pointer to pinter arithmetic

```
#include<stdio.h>
#include<conio.h>
void main()
{
char c='z',*cp,**pcp;
cp=&c;
pcp=&cp;
printf("\n character%8x%10x%c",pcp,*pcp,**pcp);
pcp++;
printf("\n **p *p p");
printf("\n character%8x%10x%c",pcp,*pcp,**pcp);
getch();
}
```

File:

It is a collection of information stored in the secondary me3mory having some filename,which stored in the directory.

Operations on files:

1.naming the file
2.opening the file:opening the files in various modes.
3.reading the content of the file:reading the dtata from the file
4.writing the content into the file:writing the datav to the file.
5.updating the file:changing some content of the records of the file
6.appending the file:writing additional data to the end of file
7.closing the file

File pointer:

Before opening the file we have to create file pointer .

Syntax:

```
FILE *fptr;
```

FILEis a structure defined in the "stdio.h" header file.

Opening the file:

Syntax:

```
FILE *fopen(char *fname,char *mode);
```

Ex.

```
FILE *fp;
If((fp=fopen("myfile","r"))==NULL)
{
Printf("error for oprn a file");
Exit(1);
}
```

mode	Meaning

r	Open a text file for reading only
w	Open a file for writing only
a	Appends to existing text file
r+	Open a text file for r/w
w+	Open the existing text fileor crate a text file for r/w
a+	Append or create a text file for read/write

Closing the file:
To close a file use fclose()
Syntax:
fcloase(FILE *fp);
The fclose file close the file associated with fp.
End of file:
A file contains a large amount of data thus some times we cannot detect end of file then use EOF .
Ex. while(!feof(fp))
Fscanf(fp,"%s",line);

Character input and output functions:
The getc() and putc() can be used to handle one charater at a time
Syntax: putc(ch,fptr);
Syntax: Ch=getc(fptr);

String input output functions:
The fgets() and fputs()functions can be used for string I/O.
Syntax: char*fgets(char *str,int n,FILE *fptr);

Syntax: fputs(const char *str,FILE *fptr);

Program for open file in read mode

```
#include<stdio.h>
#include<conio.h>
void main()
{
char ch;
FILE *fp;
fp=fopen("c:\\abcd.txt","r");
while((ch=getc(fp))!=EOF)
{
printf("%C",ch);
}
fclose(fp);
}
```

Program for count number of lines in file

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

```
#include<conio.h>
void main()
{
FILE *fp;
char ch;
int n=0;
fp=fopen("c:\\ab.txt","r");
while (1)
{
ch=fgetc(fp);
if(ch==EOF)
break;
if(ch=='\n')
n++;
}
printf("no of lines=%d",n);
fclose(fp);
}
```

Program for open c file in read mode

```
#include<stdio.h>
#include<conio.h>
void main()
{
FILE *fp;
char ch;
fp=fopen("comp.c","r");
if(fp==NULL)
printf("unable to open comp.c");
else
{
do
{
ch=getc(fp);
putchar(ch);
} while(ch!=EOF);
fclose(fp);
}
}
```

Program for file open in write mode

```
#include<stdio.h>
#include<string.h>
void main()
{
        FILE *fp;
```

```
        char line[80];
        clrscr();
        fp=fopen("c:\\abc.txt","w");
        if(fp==NULL)
        {
                printf("\n file can not open");
                exit(1);
        }
        while(strlen(gets(line))>0)
        {
                fputs(line,fp);
        }
        fclose(fp);
}
```

Program for create a text file for write the content .

```
#include<stdio.h>
#include<string.h>
void main()
{
        FILE *fp;
        int ch,i=0;
        char line[260];
        fp=fopen("c:\\abcd.txt","a");
        if(fp==NULL)
                printf("\n file can not open");
        else
        {
                do
                {
                        do line[i++]=getchar();
                        while(line[i-1]!='*');
                        fputs(line,fp);
                        i=0;
                        printf("\n press  1 to continue");
                        scanf("%d",&ch);
                }while(ch==1);
        fclose(fp);
        printf("\n file is succesfully created");
        }
}
```