Basics of C Programming

- ✓ Content in this is for Windows version and turbo C software.
- ✓ Code might slightly vary based on software we use.

BASIC TERMINOLOGY

Computer: It is an electronic device. It has memory and it performs arithmetic and logical operations.

Input: The data entering into the computer is called input.

Output: The information obtained by the computer is called output.

Program: A sequence of instructions to solve the given problem is known a program.

Software: The group of programs to operate and control the operations of a computer. Software can be classified into 2 types. System software and Application software.

Hardware: All the physical components or units connected to the computer circuit is called hardware.

Operating system: It is an interface between user and the computer. In other words, O.S is a

Complex set of programs, that manage the resources of a computer resources include, input, output, processor etc.

So O.S is a resource manager.

Language: it consists of a set of executable instructions.

Package: It is designed by any other language with limited resources.

ASCII character set

(American standard code for information and interchange)

Character type	no of characters
Capital letters	26
Small letters	26
Digits	10
Special symbols	32
Control characters	34
Graphic characters	128
Total	256

Out of 256 character set the first 128 are called ascii characters

Then next 128 are called graphic characters (or) extended ascii characters.

A to Z	65 - 90
a to z	97 - 122
0 to 9	48 - 57
Space	32
Backspace	8
Tab	9
Esc	27
&	38

Classification of programming language

Programming languages are classified into 2 types.

1. Low level language

2. High level language

1. Low level language

A low level language is called assembly language is design in the beginning. It has some simple instructions. But these instructions are not in binary code but he computer can understand only machine level language. Hence a converter (or) translator is developed to translate a low level language program into machine level language. This translator is called as assembler.

2. High level language:

These are more English like languages hence the programmer very easy to learn.

To convert the programs written in high level language to machine level language compilers and interpreters are used.

Translators

There are 3 types of translators available for the languages.

Assembler: This translator is used to convert the low level language into machine level language.

Compiler: Compilers are used to convert the high level languages into machine level language. It checks per errors in entire program and converts the program into machine language.

Interpreter: It is also used to convert high level language into machine level language. It checks per errors statement by statement and converts the statement into machine language.

Various Steps Involved in Program or Application Development:

The following steps are involved in program development

1. Problem definition

2. Analysis & Design

3. Algorithms

4. Flow chart

5. Coding & implementation

6. Debugging & testing

7. Documentation

1. Problem definition

Problem definition phase is a clear understanding of exactly what is needed for creating a workable solution. It involves three specifications

1. Input specification

2. Output specification

3. Processing

2. Analysis & Design

Before going to make a final solution for the problem, it must be analysed. Outline solution is prepared in the case of simple problems, but in case of complex problems the main problem is divided into sub-problems called 'modules'. These modules can be handled and solved independently. When the task is too big, it is always better to analyse the task such that it can be divided into number of modules and seek solution for each.

3. Algorithm

Once the problem is divided into number of modules, the logic for solving each module can be developed.The logic is expressed step by step. A step by step procedure to solve the given problem is called algorithm. An algorithm is defined as a finite set of instructions which accomplish a particular task. An algorithm can be described in a natural language like English.

4. Flow chart

After completion of algorithm, the program can be visualised by drawing a flow chart. A flow chart is nothing but a graphical or symbolic representation of how instructions will be executed one after another.

5. Coding & Implementation

Coding is a process of converting the algorithm solution of flow chart into a computer program. In this process,each and every step of algorithm is converted to instructions of selected computer programming language.

6. Debugging & Testing

Debugging: Before loading the program into the computer, we must correct all the errors. This process is called Debugging.

There are three types of errors

1. Syntax errors

2. Runtime errors

3. Logical errors

Testing: It is very important to test the program written to achieve a specific task. Testing is running the program with known data and known result.

7. Documentation

It is the most important aspect of programming. It is a continuous process of keeping copy of all phases of a program development.

Algorithm

A step by step procedure to solve the given problem is known as algorithm.

Examples:

1. Find the sum,product,division of given two numbers.

```
step 1 :  Read two numbers a, b

step 2 :  sum = a+b

step 3 :  product=a*b

step 4 :  division=a/b;

step 5 :  print sum, product, division.

step 6 :  stop.
```

----

2. Find the maximum value of given two numbers.

```
step 1 : Read two numbers a and b
step 2 : max=a
step 3 : if max<b then max=b
step 4 : print max
step 5 : stop (or) end
```

----

3. Find maximum value of given three numbers.

```
step 1 : Read three numbers a,b and c
step 2 : max=a
step 3 : if max<b then max=b.
step 4 : if max<c then max=c.
step 5 : print max
step 6 : stop (or) end.
```

----

4. To check whether the given number is even (or) odd

```
step 1 : Read the number n
step 2 : if n%2==0 "print n is even " else "print n is odd"
step 3 : stop (or) end
```

----

5. To print natural numbers from 1 to given number.

```
step 1 : Read the number n
step 2 : i=1
step 3 : print i
step 4 : i=i+1
step 5 : if i<=n then goto step 3
step 6 : stop (or) end
```

6. To calculate sum of 'n natural numbers.

```
step 1 : Read the number n
step 2 : i=1,sum=0
step 3 : sum=sum+i
step 4 : i=i+1
step 5 : if i<=n then goto step 3
step 6 : print sum
step 7 : stop (or) end
```

----

7. To print factors of given number.

```
step 1 : Read the number n
step 2 : i=1
step 3 : if n%i==0 then print i
step 4 : i=i+1
step 5 : if i<=n then goto step 3
step 6 : stop (or) end.
```

----

8. To check whether the given number is prime (or) not.

```
step 1 : Read the number n
step 2 : i=1,c=0
step 3 : if n%i==0 then c=c+1
step 4 : i=i+1
step 5 : if i<=n then goto step 3
step 6 : if c==2 then "print n is prime "
             else "n is not prime"
step 7 : stop (or) end.
```

9.To print factorial of given number

```
step 1 : Read the number n

step 2 : i=1,fact=1

step 3 : fact=fact*i

step 4 : i=i+1

step 5 : if i<=n then goto step 3

step 6 : print fact

step 7 : stop (or) end
```

10.To print reverse number of given number

```
step 1 : Read the number n

step 2 : rev=0

step 3 : rev=rev*10+(n%10)

step 4 : n=n/10

step 5 : if n>0 then goto step 3

step 6 : print rev

step 7 : stop (or) end
```

11.To print sum of digits of given number

```
step 1 : Read the number n

step 2 : sum=0

step 3 : sum=sum+(n%10)

step 4 : n=n/10

step 5 : if n>0 the goto step 3

step 6 : print sum

step 7 : stop (or) end
```

12.To count number of digits in the given number

```
step 1 : Read the number n
step 2 : nd=0
step 3 : nd=nd+1
step 4 : n=n/10
step 5 : if n>0 the goto step 3
step 6 : print nd
step 7 : stop (or) end
```

C-Language

C is a programming language.It is designed by Dennis Ritchie in 1972 at AT & T (American Telephones and telegraphs) BELL labs in U.S.A. It is the most popular general purpose language.

Brief History of C language:-

In 1960's COBOL was being used for Commercial purpose and FORTRAN is used for scientific and Engineering applications. At this stage people started to develop a language which is suitable for all possible applications. There fore An International Committee was setup to develop such a language ALGOL 60 Was released. It was not popular because it seemed too general. To reduce these generality a new language called CPL (Combined programming Language) was developed at Cambridge University. Then some other feature are added to this language and a new language called BCPL developed by Martin Ritchards at Cambridge University. Then B language was devoloped by Ken Thompson at AT & T Bell labs. Dennis Ritchie inherited the features of B and BCPL and added his own features and developed C-Language in 1972.

1. Computer concepts.
2. Introduction to 'c'.
3. 'c' tokens.
4. Control statements.
5. Nested loops
6. Formatting the output, patterns
7. Functions, storage classes, recursive function
8. math.h functions
9. Arrays.

10. Strings.

11. Pointers, Dynamic memory allocation.

12. Structures, unions, bit fields.

13. Files.

Features of C language:-

1. C is a structure programming language with fundamental flow control construction.
2. C is simple and versatile language.
3. Programming written in C is efficient and fast.
4. C has only 32 key words.
5. C has rich set of operators.
6. C permits all data conversions into a mixed mode operations.
7. Dynamic storage allocation is possible in C language.
8. Extensive data types such as arrays, pointer , structures and unions are available in c
9. C improves by itself, It has several built in functions and standard functions are available.
10. Recursive function causes for algorithmic approach is possible in c.
11. Mainly we are using the C language to implement the system software. Those are compilers, text editors, network drivers and data base utility and finally operating system.
12. C compiler combines the capability of an assembly level language with the features of high level language. So it is called as middle level language.

Imp points:-

1. C is a case sensitive programming language.
2. C statements are entered in lowercase letters only.
3. C is a function oriented programming language.

Any C program contains one (or) more functions minimum one function is compulsory by the name called main().

Every C statement must be terminated by semicolon except pre-processor section and function definition.

A function is represented by function name with a pair of Parenthesis.

() parenthesis [] square braces {} curly braces (or) flower braces <> Angular braces

Basic structure of C program

```
            [Document section]
            Pre-processor section
                    (or)
              Link section
      [Global declaration section]
      main ()
      {

            [local declaration section ]


              Statements;
      }
      [Sub-processing section]
                  ³
       User defined functions
```

Document section:

A Document section consist a set of comment lines giving the name of the program, author name and some other details.

Pre-processor or Linking section:

It provides instructions to the compiler to link the functions from the system library.

Global declaration section:

Variables that are used in more than one function are called as Global variables which are declared variables in the Global declaration section.

Main function section:

Every C program must have one function i.e. main function.

This section contains two parts namely:

(1) Local declaration section

(2) Statements part.

 (1) The local declaration section declares all the variables used in statements.

 (2) The statements part contains executable instructions.

These two parts must appear between the opening and closed curly braces. The program execution begins at the opening brace and closing at closing braces.

Sub programming section

It contains all the user defined functions.

Comments

Non-executable lines in the program are called as comment lines. These lines are skipped by the compiler.

```
    /* ---------------------        Group of lines comment

           ---------------------           */            (or)

                                                   Single line comment

    //   ----------------------        Single line comment
```

#include:

It is a preprocessor file inclusion directive and it is used to include header files.

#include "file name"

(or)

#include <file name>

When the file name is included within double quotation marks, the search for the file is made first in current directory and then the standard directory.

When the file name is included within Angular braces, the file is search only in standard directory.

stdio.h	Standard input and output header file
conio.h	Console input and output header file

1) Open 'c' editor

a) Double click on desktop c++ icon.

b) Start->run->c:\tc\bin\tc.exe

c) Select mycomputer -> c -> tc -> bin -> tc

2) Open file menu

press alt+f -> select new

full screen - f5

3)type 'c' program

4)save 'c' program (.c)

press alt+f -> select save

5) compile

press alt+c -> select compile

6)run

press alt+r ->select run

7)output

Press alt+w ->select user screen

8)quit

press alt+f ->select quit

Shortcuts:

save	f2
full screen	f5
compile	alt+f9
run	ctrl+f9
output	alt+f5
quit	alt+x
to close a file	alt+f3

printf <stdio.h> <conio.h>

It is function,which is used to print data on the screen.

Declaration

int printf("contol string"[,arg-1,arg-2,arg--------n]);

³

format string

eg

printf("welcome to c language");

```
// this is first program

#include<stdio.h>

#include<conio.h>

void main()

{

        printf("sachin_aswin");

}
```

clrscr <conio.h>

It is a function, it clears text mode window.

Declaration

void clrscr();

eg

clrscr();

getch <conio.h>

It is a function, to show the output directly without pressing alt+f5.

Declaration

int getch();

eg

getch()

```
#include<stdio.h>
#include<conio.h>
void main()
{
	clrscr();
	printf("sachin_aswin");
	getch();
}
```

Escape sequence characters:

\n	new line
\t	horizontal tab (default 8 spaces)
\b	back space
\v	vertical tab (default 1 line)
\r	carriage return
\a	beep sound

```
#include<stdio.h>
#include<conio.h>
void main()
{
	clrscr();
	printf("sachin\n");
	printf("power star\naswin");
	getch();
}
```

C-Tokens

The smallest individual elements (or) units used in a program are called as tokens.

C has following tokens

1. Identifiers

2. Keywords

3. Constants

4. Operators

5. Special characters

1. Identifiers:

These can be defined as name of the variables, arrays, functions and some other program elements using the combination of the following characters.

Alphabets	A to Z
	a to z
Digits	0 to 9

Underscore.

The first character of an identifier will be an alphabet (or) underscore followed by alphabet, digit (or) underscore.

NOTE

Default identifier length is 32 characters.

2. Key words

Keywords are the words whose meaning has already being explained in the Compiler. That means at the designing a language some words are reserved to do specific tasks such words are called as keywords (or) reserved words. C has 32 keywords.

1.auto	2.break	3.case	4.char
5.const	6.continue	7.default	8.do
9.double	10.else	11.enum	12.extern
13.float	14.far	15.for	16.goto
17.if	18.int	19.long	20.register
21.return	22.short	23.signed	24.sizeof

25.struct 26.static 27.switch 28.typedef

29.union 30.unsigned 31.void 32.while

3. Constants

They refer to fixed values that do not change during the execution of a program. The following constants are availabe in C.

(i) Integer constants

(ii) Real constants (float, double, long double)

(iii) Character constants

(iv) String constants.

4. Operators

It is a symbol which performs a particular operation. C has rich set of Operators.

C operators can be classified into the following types:

1. Arithmetic operators

2. Relational operators

3. Logical operators

4. Bitwise operators

5. Increment and decrement operators.

6. Assignment operators

7. Special operators

(i) ternary (or) conditional operator

(ii) comma operator

(iii) sizeof operator

5. Special characters

All characters other than alphabets, digits and underscore are treated as special characters.

eg

< , > , & ,{ etc

DATA TYPES:-

A kind of data that variables may hold in a programming language is called data type.

C has 3 types of data types.

1. Primary data types (or) primitive (or) predefined

2. User defined data types

3. Derived data types

1. Primary data types

All C compilers support 4 fundamental data types namely int,char,float and double.

(i) int

It is a +ve,-ve and whole values but not a decimal number.

eg

10,30,-34,0,etc....

(ii) char

A single character can be defined as a character data type and it can be represented with in single quotation marks.

eg

'S','A','1','*' etc....

(iii) String

It is also a character data type and it can be defined with in double quotation marks.

eg

"BDPS","WELCOME","12345", etc.....

(iv) float (value with decimal point

The numbers which are stored in the form of floating point representation is called float data type.

eg

10.423, 234.33, etc....

(v) double

The no's which are stored in the form of double precision floating point representation is known as double data type.

eg

10.423,234.33,232242.34434,22154.5455,etc.....

2. User Defined data types

The user defined data types enables a programmer to invent his own data types and define what values it can take on. This data type can help a programmer to reducing the programming errors.

There are two types of user defined data types namely

1. enum (enumerated data type)

2. typedef(type definition)

3. Derived data types

They are built from the basic fundamental data types. The examples of derived data types are arrays, pointers, structures etc...

NOTE

----

void ----------- is an empty data type

Data types , Format specifiers,Memory sizes and their accessibility ranges

data type	format specifier	memory size	accessibility range
1. unsigned char	%c	1 byte	0 to 255
2. char	%c	1 byte	-128 to +127
3. int	%d	2 bytes	-32,768 to 32,767
4. unsigned int	%u	2 bytes	0 to 65,535
5. long int	%ld	4 bytes	-214,74,83,647 to 214,74,83,647
6. unsigned long int	%lu	4 bytes	0 to 429,49,67,295
7. float	%f	4 bytes	3.4*10 to 3.4*10
8. double	%lf	8 bytes	1.7*10 to 1.7*10
9. long double	%Lf	10 bytes	3.4*10 to +4932 1.1*10
10. char [](string)	%s	-------	-----------------

%o   octal base

%x   hexa decimal base

varaiable

A quantity which may vary during the execution of a program is known as variable.

Declaration of a variable

Form 1 : datatype identifier;

eg

int n;

char ch;

float ft;

Form 2: datatype identifier-1,identifier-2--------------------identifier-n;

eg

int a,b,c;

char ch1,ch2;

Initialisation: At the time of declaring a variable we can provide some data into that Variable is known as initialisation.

Declaration

datatype identifier=value;

eg

int a=100;

char ch='s';

datatype identifier-1=value,identifier-2=value,--------identifier-n=value;

eg

int a=100,b=200,c=300;

char ch1='a',ch2='k';

```
#include<stdio.h>

#include<conio.h>

void main()

{

        int n=100;

        clrscr();

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

        getch();

}


#include<stdio.h>

#include<conio.h>

void main()

{

        char ch='s';

        int n=100;

        float ft=123.90;

        double db=12345.890;

        clrscr();

        printf("Character       =  %c",ch);

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

        printf("\n Float value    =  %f",ft);

        printf("\n Double value   =  %lf",db);


        getch();

}
```

Constants

Constants refer to fixed values that do not change during the execution of a program.

Const: It is a keyword,and is used to define constants.

Syntax: const datatype identifier=value;

eg

```
const int n=100;

#include<stdio.h>

#include<conio.h>

void main()

{

        const int a=100;

        clrscr();

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

        //    a=200;  can not modify

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

        getch();

}
```

scanf

It is a function, which is used to read data from keyboard.

Declaration

int scanf("format(s)",address-1,address-2,---------address-n);

eg

int n = scanf("%d",&n) &n is address of n

char ch = &ch   Address of ch

```
#include<stdio.h>

#include<conio.h>

void main()

{

        int n;

        clrscr();

        printf("Enter any integer value : ");

        scanf("%d",&n);

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

        getch();

}
```

```
#include<stdio.h>

#include<conio.h>

void main()

{

        char ch;

        int n;

        float ft;

        double db;

        clrscr();

        printf("Enter any character   :  ");

        scanf("%c",&ch);

        printf("Enter any integer value : ");

        scanf("%d",&n);

        printf("Enter any float value : ");

        scanf("%f",&ft);
```

```
	printf("Enter any double value : ");
	scanf("%lf",&db);
	printf("Given character      =    %c",ch);
	printf("\n Given integer value  =   %d",n);
	printf("\n Float value          =   %.2f",ft);
	printf("\n Double value         =   %.2lf",db);
	getch();
}

#include<stdio.h>
#include<conio.h>
void main()
{
	char ch;
	int n;
	clrscr();
	printf("Enter any integer value : ");
	scanf("%d",&n);
	printf("Enter any character   :  ");
	scanf("%c",&ch);
	printf("\n Given integer value  =   %d",n);
	printf("\n Given character      =   %c",ch);
	getch();
}
```

fflush

It flushes specified stream.

Declaration: int fflush(FILE *stream);

flushall

It flushes all open streams.

Declaration: int flushall();

```
#include<stdio.h>
#include<conio.h>
void main()
{
        char ch;
        int n;
        clrscr();
        printf("Enter any integer value : ");
        scanf("%d",&n);
        printf("Enter any character   :  ");
        fflush(stdin);
        //flushall();
        scanf("%c",&ch);
        printf("\n Given integer value  =   %d",n);
        printf("\n Given character      =   %c",ch);
        getch();
}
```

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

```
void main()
{
            char ch;
            clrscr();
            printf("Enter any character : ");
            scanf("%c",&ch);
            printf("Ascii value of given character  =  %d",ch);
            getch();
}

#include<stdio.h>
#include<conio.h>
void main()
{
            int n;
            clrscr();
            printf("Enter any ascii value ( 0  to  255 ) : ");
            scanf("%d",&n);
            printf("Character of given ascii value =   %c",n);
            getch();
}

#include<stdio.h>
#include<conio.h>
void main()
{
            int n;
```

```
        clrscr();

        printf("Enter any integer value : ");

        scanf("%d",&n);

        printf("Given number            = %d", n);

        printf("\n Octal number          = %o",n);

        printf("\n Hexa decimal number    = %x",n);

        getch();

}
```

String

A group of characters can be defined between double quotation marks is a string.In 'c' language a string is nothing but an array of characters and terminated by a null character (\0)

Declaration: char identifier[size];

Eg

char st[20];

Initialisation: char identifier[size]="string";

eg

char st[20]="welcome";

When the compiler assigns a character string to a character array then it automatically supplies a null character at the end of the string. Therefore the size should be equal to the maximum number of characters in the given string + 1.

Format specifier of string is %s

```
#include<stdio.h>

#include<conio.h>

void main()

{

        char st[80];
```

```
	clrscr();
	printf("Intialised string   =   %s",st);
	getch();
}
```

```
#include<stdio.h>
#include<conio.h>
void main()
{
	char st[80];
	clrscr();
	printf("Enter any string : ");
	scanf("%s",st);
	printf("Given string   =   %s",st);
	getch();
}
```

gets

It is a function which is used to read a string with blank spaces from keyboard into a string variable.

syntax

```
char * gets(string variable);
```

eg

```
char st[80];
gets(st);
```

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

```
void main()

{

        char st[80];

        clrscr();

        printf("Enter any string : ");

        gets(st);

        printf("Given string   =   %s",st);

        getch();

}
```

OPERATORS

Operator

It is a symbol which performs a particular operation.

Operand:

It is an entity on which an operator acts.

Unary operator:

It requires single operand.

Binary operator:

It requires two operands.

'c' operators can be classified into number of categories.

1. Arithmetic operators
2. Logical operators
3. Relational operators
4. Assignment operators
5. Increment and decrement operators

6. Bitwise operators

7. Special operators

   (a)ternary operator (or) conditional operator

   (b)comma operator

   (c)sizeof operator

1. Arthemetic operators

These are basic operators in 'c' language. These operators used for arithmetic operation.

+	Addition
-	Subtraction
*	Multiplication
/	Division
%	modulas

```
#include<stdio.h>
#include<conio.h>
void main()
{
        int a,b;
        clrscr();
        printf("Enter any two integer values : ");
        scanf("%d%d",&a,&b);
        printf("Addition of %d and %d is %d",a,b,a+b);
        printf("\n Subtraction of %d and %d is %d",a,b,a-b);
        printf("\n Multiplication of %d and %d is %d",a,b,a*b);
        printf("\n Division of %d nad %d is %d",a,b,a/b);
        printf("\n Modulas of %d and %d is %d",a,b,a%b)
        getch();
}
```

2. logical operators

These operators are used for combine the result of two or more expressions.

(i) logical AND &&
(ii) logical OR ||
(iii) logical NOT !

EXP1	EXP2	EXP1 && EXP2	EXP1 || EXP2
TRUE	TRUE	TRUE	TRUE
TRUE	FALSE	FALSE	TRUE
FALSE	TRUE	FALSE	TRUE
FALSE	FALSE	FALSE	FALSE

Exp - TRUE !Exp - FALSE

Exp - false !Exp - true

3. Relational operators

These operators are used to test the relation between two values (or) two variables (or) two expressions.

All 'c' relational operators are binary operators and hence requires two operands.

<	less than
>	greater tham
<=	less than or equal to
>=	greater than or equal to
==	equal to
!=	not equal to

4. Assignment operators

These operators are used to assign a value of an expression to an identifier.

= Assignment operator

Compound assignment operators (or) short hand operators

+=

-=

*=

/=

%=

eg

int n=10;

n=n+10; <=> n+=10;

5. Increment and decrement operators

These operators are used to control the loops in an effective method.

Increment operator (++)

The symbol ++ is used for incrementing by 1.

++identifier   : prefix increment

identifier++   : postfix increment

Decrement operator (--):

The symbol -- is used for decrementing by 1

--identifier   : prefix decrement

identifier--   : postfix decrement

decrement operator --

The symbol -- is used for decrementing by 1.

--identifier : prefix decrement

identifier-- : postfix decrement

6. Bitwise operators:

These operators are used for calculating data at bit level. Bitwise operators are two types

(i) bitwise logical operators
(ii) bitwise shift operators

(i).bitwise logical operators:

These operators are used for the bitwise logical decision making.

(i)	bitwise AND	&
(ii)	bitwise OR	|
(iii)	bitwise exclusive OR	^

B1	B2	B1 & B2	B1 | B2	B1 ^ B2
1	1	1	1	0
1	0	0	1	1
0	1	0	1	1
0	0	0	0	0

```
#include<stdio.h>

#include<conio.h>

void main()

{

        int a=5,b=6;

        clrscr();

        printf("a&b     =  %d", a&b);

        printf("\n a|b  =  %d", a|b);

        printf("\n a^b  =  %d", a^b);

        getch();

}
```

(ii).Bitwise shift operators

The shift operators take binary patterns and shift bits to left (or) right.

(i) left shift operator <<
(ii) right shift operator >>

```
#include<stdio.h>

#include<conio.h>

void main()

{

        int a=4,b,c;

        clrscr();

        b=a<<2;

        c=a>>1;

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

        printf("\n b=%d",b);

        printf("\n c=%d",c);

        getch();

}
```

7. Special operators:

(a) ternary operator (or) conditional operator

A ternary operator pair " ?: " is available in 'c' language. It is used to construct conditional expression.

The general form of ternary operator is

exp1 ? exp2 : exp3;

Where exp1, exp2, exp3 are expressions.

If exp1 is true then exp2 is evaluated and its value becomes value of the expression.

Otherwise if exp1 is false then exp3 is evaluated and its value becomes value of the expression.

eg

int a=10,b=20,c;

```
c=a>b ? a : b;

a-10  b-20  c-20
```

Write a program to accept any two integer values and display maximum and minimum values using ternary operator.

```
#include<stdio.h>

#include<conio.h>

void main()

{

	int a,b,max,min;

	clrscr();

	printf("Enter any two integer values : ");

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

	max=a>b?a:b;

	min=a<b?a:b;

	printf("Maximum value       =  %d",max);

	printf("\n Minimum value     =  %d",min);

	getch();

}
```

Write a program to accept any three integer values and display maximum and minimum values using ternary operator.

```
#include<stdio.h>

#include<conio.h>

void main()

{

	int a,b,c,max,min;

	clrscr();

	printf("Enter any three integer values : ");
```

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

        max=a>b?a:b;

        max=max>c?max:c;

        min=a<b?a:b;

        min=min<c?min:c;

        printf("Maximum value      =  %d",max);

        printf("\n Minimum value     =  %d",min);

        getch();

}
```

Write a program to accept any integer value and check whether the given number is even (or) odd using ternary operator

```
#include<stdio.h>

#include<conio.h>

void main()

{

  int n;

  clrscr();

  printf("Enter any integer value : ");

  scanf("%d",&n);

  n%2==0 ? printf("Given number is Even") : printf("Given number is Odd");

  getch();

}
```

(iii) comma operator:

A comma operator can be used to link the related expressions together.

The general form of comma operator is

```
identifier=(identifier-1=value,identifier-2=value------------------------

        -------identifier-n=value,expression);
```

eg

```
int a,b,c;

c=(a=10,b=20,a+b);
```

First assigns the value 10 to 'a' and then assigns the value 20 to 'b'

and finally assigns 30(a+b) to 'c'.

```
#include<stdio.h>
#include<conio.h>
void main()
{
        int a,b,c;
        clrscr();
        c=(a=10,b=20,a+b);
        printf("\n a=%d",a);
        printf("\n b=%d",b);
        printf("\n c=%d",c);
        getch();
}
```

(c) sizeof operator

It is a compiling operator and is used to get the memory size of specified datatype (or) variable (or) expression.

Syntax: sizeof(datatype (or) variable (or) expression);

eg

```
sizeof(int)    -  2 bytes

sizeof(char)   -  1 byte
```

sizeof(float) - 4 bytes.

eg2

```
int n

char ch;

float ft;

sizeof(n)      -   2 bytes

sizeof(ch)     -   1 byte

sizeof(ft)     -   4 bytes
```

eg3

```
int a,b;

sizeof(a)      -  2 bytes

sizeof(b)      -  2 bytes

sizeof(a+b)    -  2 bytes
```

```
#include<stdio.h>

#include<conio.h>

void main()

{

        int n;

        float ft;

        clrscr();

        printf("Size of integer variable    =    %d bytes",sizeof(n));

        printf("\nSize of float variable    =    %d bytes",sizeof(ft));

        getch();

}
```

```
#include<stdio.h>

#include<conio.h>

void main()

{

            int a,b;

            clrscr();

            printf("Size of a      =  %d bytes",sizeof(a) );

            printf("\n Size of b   =  %d bytes",sizeof(b) );

            printf("\n Size of a+b    =  %d bytes",sizeof(a+b) );

            getch();

}


#include<stdio.h>

#include<conio.h>

void main()

{

            int a,b,t;

            clrscr();

            printf("Enter any two integer values : ");

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

            printf("Values before swapping \n ");

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

            printf("\n b  =  %d",b);


            /*  swapping  */


            t=a;
```

```
        a=b;

        b=t;

        printf("\n Values After swapping  ");

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

        printf("\n b  =  %d",b);

        getch();

}


#include<stdio.h>

#include<conio.h>

void main()

{

        int a,b;

        clrscr();

        printf("Enter any two integer values : ");

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

        printf("Values before swapping \n ");

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

        printf("\n b  =  %d",b);

        /* swapping */

        a=a+b;

        b=a-b;

        a=a-b;

        printf("\n Values After swapping  ");

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

        printf("\n b  =  %d",b);

        getch(); }
```

Compound statements

A compound statement consist one or more instructions and those instructions are enclosed with in curly braces.

1. A variable which is declared above the compound statement it is accessible both inside and outside compound statements.
2. A variable which is declared inside the compound statement, it is not available outside the compound statement.

```
#include<stdio.h>

#include<conio.h>

void main()

{

	int n=100;

	clrscr();

	printf("\n Outer compound statement");

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

	{

		int m=200;

		printf("\n Inner compound statement");

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

		printf("\n m=%d",m);

	}

	printf("\n Outer compound statement");

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

	getch();

}
```

3. A variable is declared with the same name in both inner and outer compound statements.

```
#include<stdio.h>

#include<conio.h>
```

```
void main()
{
        int n=100;
        clrscr();
        printf("\n Outer compound statement");
        printf("\n n=%d",n);
        {
                int n=200;
                printf("\n Inner compound statement");
                printf("\n n=%d",n);
        }
        printf("\n Outer compound statement");
        printf("\n n=%d",n);
        getch();
}
```

Type casting

The process of converting one datatype to another dataype is called type casting.

In 'c' language type conversion is an arithmetic expression will be done automaticlly. If you want to store a value of one type into variable of another type we must cast the value will be stored by preceding it with the type name in paranthesis.

eg

```
int a,b;
float c,d;
a-5    b-2
c=a/b;   automatic conversion (or) implicit conversion
c-2.00
d=(float)a/b;  explicit conversion (or) type casting
```

```
d-2.50;


#include<stdio.h>
#include<conio.h>
void main()
{
	int a=5,b=2;
	float c,d;
	clrscr();
	c=a/b;
	d=(float)a/b;
	printf("c=%.2f",c);
	printf("\n d=%.2f",d);
	getch();
}


#include<stdio.h>
#include<conio.h>
void main()
{
	int cno,pmr,lmr,tu;
	char cname[80];
	float bill;
	clrscr();
	printf("Enter consumer number : ");
	scanf("%d",&cno);
	printf("Enter consumer name : ");
	fflush(stdin);

	gets(cname);

	printf("Enter present month reading : ");

	scanf("%d",&pmr);

	printf("Enter last month reading : ");

	scanf("%d",&lmr);

	tu=pmr-lmr;

	bill=tu*3.00;

	printf("Total units          =   %d",tu);

	printf("\n Bill amount       =   %.2f",bill);

	getch();

}


#include<stdio.h>

#include<conio.h>

void main()

{

	int sno,c,cpp,java,tot;

	char sname[80];

	float avg;

	clrscr();

	printf("Enter student number : ");

	scanf("%d",&sno);

	printf("Enter student name : ");

	fflush(stdin);

	gets(sname);

	printf("Enter marks in c cpp java : ");
```

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

        tot=c+cpp+java;

        avg=(float)tot/3;

        printf("Total marks    =  %d",tot);

        printf("\n Average     =  %.2f",avg);

        getch();

}
```

Control statements (or) control structures

'c' is a considered as a structure programming language.one of region for this is having various program control structures.

'c' process the decision making capabilities and supports the statements known as control statements.

There are 3 types of control structures.

(1) Condition control statements

(2) Un condition control statements

(3) Loop control statements

(1) Condition control statements

'c' supports 5 types of condition control statements.

(i) simple if
(ii) if else
(iii) nested if

(iv) else if ladder

(iv) switch

(i) Simple if statement

It is a decision making statement and is used to control the flow of execution.

Syntax:

if(expression)

```
{
        statements;
}
```

If the expression is true, then the statement block will be executed and the control transfers to the next statement.

Otherwise the expression is false then the control directly goes the next statement.

```
#include<stdio.h>
#include<conio.h>
void main()
{
        int a,b,max;
        clrscr();
        printf("Enter any two integer values : ");
        scanf("%d%d",&a,&b);
        max=a;
        if(max<b)
        {
                max=b;
        }
        printf("Maximum value     =   %d",max);
        getch();
}
```

Note:

In any control statement, the statement block contains a single statement the braces are not necessary.

```
#include<stdio.h>

#include<conio.h>

void main()

{

        int a,b,c,max;

        clrscr();

        textmode(2);

        printf("Enter any three integer values : ");

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

        max=a;

        if(max<b)

        {

                max=b;

        }

        if(max<c)

        {

                max=c;

        }

        printf("Maximum value     =   %d",max);

        getch();

}


#include<stdio.h>

#include<conio.h>

void main()

{

        int d,m,y;
        clrscr();
        printf("Enter any date (dd-mm-yyyy)  : ");
        scanf("%d-%d-%d",&d,&m,&y);
        printf("Given date      =    %.2d - %.2d - %d",d,m,y);
        getch();
}


#include<stdio.h>
#include<conio.h>
void main()
{
        int cd,cm,cy,bd,bm,by,y,m,d;
        clrscr();
        textmode(2);
        printf("Enter current date  (dd-mm-yyyy) : ");
        scanf("%d-%d-%d",&cd,&cm,&cy);
        printf("Enter Date of birth (dd-mm-yyyy) : ");
        scanf("%d-%d-%d",&bd,&bm,&by);
        y=cy-by;
        m=cm-bm;
        d=cd-bd;
        if(d<0)
        {
                d=d+30;
                m=m-1;
        }
        if(m<0)
        {
                m=m+12;
                y=y-1;
        }
        printf("Age is  %d years %d months %d days",y,m,d);
        getch();
}
```

(ii) if else statement:

It is an extension of simple if statement.

Syntax

```
if(expression)
{
        statement-1;
}
else
{
        statement-2;
}
```

If the expression is true then if block statements are executed and else block statements are ignored.

Otherwise expression is false then if block statements are ignored and else block statements are executed.

```
#include<stdio.h>
#include<conio.h>
void main()
{
        int a,b,max;
```

```
	clrscr();

	textmode(2);

	printf("Enter any two integer values : ");

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

	if(a>b)

	{

		max=a;

	}

	else

	{

		max=b;

	}

	printf("Maximum value      =   %d",max);

	getch();

}


#include<stdio.h>

#include<conio.h>

void main()

{

	int n;

	clrscr();

	printf("Enter any integer value : ");

	scanf("%d",&n);

	if(n%2==0)

	{

		printf("Given number is even");
```

```
        }
        else
        {
                printf("Given number is odd");
        }
        getch();
}
```

(iii) Nesetd if statement:

using a if statement with in another if is called as nested if. If a series of decisions are involved,we use nested if.

Form1

```
if(expr-1)
{
        if(expr-2)
        {
                ---------
                ---------
                if(expr-n)
                {
                        statements;
                }
        }
}
```

Form2:

```
if(expr-1)
{
        if(expr-2)
        {
                statement-1;
        }
        else
        {
                statement-2;
        }
}
else
{
        if(expr-3)
        {
                statement-3;
        }
        else
        {
                statement-4;
        }
}
```

```
#include<stdio.h>
#include<conio.h>
void main()
{
        int a,b,c,max;
        clrscr();
        printf("Enter any three integer values : ");
        scanf("%d%d%d",&a,&b,&c);
        if(a>b)
        {
                if(a>c)
                        max=a;
                else
                        max=c;
        }
        else
        {
                if(b>c)
                        max=b;
                else
                        max=c;
        }
        printf("Maximum value   =  %d",max);
        getch();
}


#include<stdio.h>
#include<conio.h>
void main()
{
        int sno,c,cpp,java,tot;
        char sname[80];
        float avg;
        clrscr();
        printf("Enter student number : ");
        scanf("%d",&sno);
        printf("Enter student name : ");
        fflush(stdin);
        gets(sname);
        printf("Enter marks in c cpp java : ");
        scanf("%d%d%d",&c,&cpp,&java);
        tot=c+cpp+java;
        avg=(float)tot/3;
        printf("\n Total marks     =   %d",tot);
        printf("\n Average      =   %.2f",avg);
        if(c>=50 && cpp>=50 && java>=50)
        {
                printf("\n Result    =  PASS");
                if(avg>=70)
                        printf("\n Division    =   First");
                else
                        printf("\n Division    =   Second");
        }
        else
        {
                printf("\n Result    =  Fail");
                printf("\n Division  =  No_Division");
        }
        getch();
}
```

(iv) else if ladder:

This statement is also used for the series of decisions are involved.

Syntax

```
if(expr-1)
{
        statement-1;
}
else if(expr-2)
{
        statement-2;
}
else if(expr-3)
{
        statement-3;
}
----------------
else if(expr-n)
{
        statement-n;
}
else
{
        statements;
}
```

In this statement expressions are evaluated top to bottom. If the condition is true the statements associated that block executed and control transfer to the next statement.

```
#include<stdio.h>
#include<conio.h>
void main()
{
	int a,b,c,max;
	clrscr();
	printf("Enter any three integer values : ");
	scanf("%d%d%d",&a,&b,&c);
	if(a>b && a>c)
	{
		max=a;
	}
	else if(b>c)
	{
		max=b;
	}
	else
	{
		max=c;
	}
	printf("Maximum value	=%d",max);
	getch();
}
#include<stdio.h>
#include<conio.h>
void main()
{
        char ch;
        clrscr();
        printf("Enter any character : ");
        scanf("%c",&ch);
        if( (ch>=65 && ch<=90) || (ch>=97 && ch<=122) )
                printf("Given character is alphabet");
        else if(ch>=48 && ch<=57)
                printf("Given character is digit");
        else
                printf("Given character is special character");
        getch();
}

#include<stdio.h>
#include<conio.h>
void main()
{
        char ch;
        clrscr();
        printf("Enter any character : ");
        scanf("%c",&ch);
        if( (ch>=65 && ch<=90) || (ch>=97 && ch<=122) )
        {
                if(ch=='A' || ch=='a' || ch=='E' || ch=='e' || ch=='O' ||
                   ch=='o' || ch=='I' || ch=='i' || ch=='U' || ch=='u')
                        printf("Given character is vowel");
                else
```

```
                        printf("Given character is consonent");
        }
        else
                printf("Given character is not an alphabet");
        getch();
}

/*
  units   rate/unit

  0-50       1.00
  51-100      2.00
  101-200     3.00
  201-400     4.00
  400 above   5.00
*/
#include<stdio.h>
#include<conio.h>
void main()
{
        int cno,pmr,lmr,tu;
        char cname[80];
        float bill;
        clrscr();
        textmode(2);
        printf("Enter consumer number : ");
        scanf("%d",&cno);
        printf("Enter consumer name : ");
        fflush(stdin);
        gets(cname);
        printf("Enter present month reading : ");
        scanf("%d",&pmr);
        printf("Enter last month reading : ");
        scanf("%d",&lmr);
        tu=pmr-lmr;
        if(tu<=50)
                bill=tu*1.00;
        else if(tu<=100)
                bill=(50*1.00)+(tu-50)*2.00;
        else if(tu<=200)
                bill=(50*1.00)+(50*2.00)+(tu-100)*3.00;
        else if(tu<=400)
                bill=(50*1.00)+(50*2.00)+(100*3.00)+(tu-200)*4.00;
        else
                bill=(50*1.00)+(50*2.00)+(100*3.00)+(200*4.00)+(tu-400)*5.00;


        printf("Total units         =   %d",tu);
        printf("\n Bill amount       =   %.2f",bill);
        getch();
}
```

```
#include<stdio.h>
#include<conio.h>
void main()
{
        int m,y,nd;
        clrscr();
        printf("Enter month and year : ");
        scanf("%d%d",&m,&y);
        if(m==2)
        {
                if( (y%4==0 && y%100!=0 )  || (y%400==0 ) )
                        nd=29;
                else
                        nd=28;
        }
        else if(m==1 || m==3 || m==5 || m==7 || m==8 || m==10 || m==12)
                nd=31;
        else
                nd=30;
        printf("Number of days          =   %d",nd);
        getch();
}
```

exit   <process.h>

It terminates the program.

Declaration:

void exit(int status);

eg

```
exit(0);
```

Typically a value of 0 indicates normal exit and a non zero value indicates some error.

```
#include<stdio.h>

#include<conio.h>

#include<process.h>

void main()

{

        int cno,pmr,lmr,tu;

        char cname[80];

        float bill;

        clrscr();

        printf("Enter consumer number : ");

        scanf("%d",&cno);

        printf("Enter consumer name : ");

        fflush(stdin);

        gets(cname);

        printf("Enter present month reading : ");

        scanf("%d",&pmr);

        printf("Enter last month reading : ");

        scanf("%d",&lmr);

        if(pmr<lmr)

        {

                printf("Invalid Reading");

                getch();

                exit(0);

        }
        tu=pmr-lmr;

        bill=tu*3.00;

        printf("Total units         =   %d",tu);

        printf("\n Bill amount       =   %.2f",bill);

        getch();

}


D-regions are N-north  S-south   W-west    E-east

                petrol     diesel   kerosine

----------------------------------------------

NORTH          49.00      34.00    23.00

SOUTH          48.00      33.00    21.00

EAST           51.00      32.00    22.00

WEST           52.00      33.00    23.00

*/

#include<stdio.h>

#include<conio.h>

#include<process.h>

void main()

{

        int dno,np,nd,nk;

        char dname[80],reg;

        float pamt,damt,kamt,tamt;

        clrscr();

        printf("Enter dealer number : ");

        scanf("%d",&dno);

        printf("Enter dealer name : ");
```

```
fflush(stdin);
gets(dname);
printf("Enter dealer region : ");
scanf("%c",&reg);
if(reg>=97 && reg<=122)
        reg=reg-32;
if(reg!='N' && reg!='S' && reg!='E' && reg!='W')
{
        printf("Invalid region");
        getch();
        exit(0);
}
printf("Enter number of litres petrol sold : ");
scanf("%d",&np);
printf("Enter number of litres diesel sold : ");
scanf("%d",&nd);
printf("Enter number of litres kerosine sold : ");
scanf("%d",&nk);
if(reg=='N')
{
        pamt=np*49.00;
        damt=nd*34.00;
        kamt=nk*23.00;
}
else if(reg=='S')
{
        pamt=np*48.00;
```

```
                damt=nd*33.00;
                kamt=nk*21.00;
        }
        else if(reg=='E')
        {
                pamt=np*51.00;
                damt=nd*32.00;
                kamt=nk*22.00;
        }
        else
        {
                pamt=np*52.00;
                damt=nd*33.00;
                kamt=nk*23.00;
        }
        tamt=pamt+damt+kamt;
        printf("\n Petrol Amount     =   %.2f",pamt);
        printf("\n Diesel Amount     =   %.2f",damt);
        printf("\n Kerosine Amount   =   %.2f",kamt);
        printf("\n Total Amount      =   %.2f",tamt);
        getch();
}
```

4. Switch statement:

It is a multi way conditional statement used in 'c' language. It is mainly used in situation where there is need to pick 1 alternative among many alternatives.

syntax

```
switch(exp (or) variable)

{

        case 1:

                statement-1;

                break;

        case 2:

                statement-2;

                break;

        --------------------

        --------------------

        case n:

                statement-n;

                break;

        default:

                default statements;

}
```

The switch statement tests the value of given variable or expression against a list of case values.and when a match is found a block of statements associated that case executed. Otherwise the default block statements will be executed.

Break:

It is unconditional control statement and is used to terminate a switch statement.

syntax

```
break;
```

Note

1. In switch statement the variable (or) expression is an integral value. We cannot pass string and float values.
2. In switch statement the default block is optional.

```
#include<stdio.h>

#include<conio.h>

void main()

{

        int a,b,opt;

        clrscr();

        printf("\nMENU");

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

        printf("\n1.Addition");

        printf("\n2.Subtraction");

        printf("\n3.Multiplication");

        printf("\n4.Division");

        printf("\n5.Modulas");

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

        printf("\nEnter any two integer values : ");

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

        printf("Enter your option : ");

        scanf("%d",&opt);

        switch(opt)

        {

                case 1:

                        printf("Addition of %d and %d is %d",a,b,a+b);

                        break;

                case 2:

                        printf("Subtraction of %d and %d is %d",a,b,a-b);

                        break;

                case 3:
                        printf("Multiplication of %d and %d is %d",a,b,a*b);

                        break;

                case 4:

                        printf("Division of %d and %d is %d",a,b,a/b);

                        break;

                case 5:

                        printf("Modulas of %d and %d is %d",a,b,a%b);

                        break;

                default:

                        printf("Invalid option");

        }

        getch();

}
```

2. Unconditional control statements:

(i) break

It passes control

Syntax

break;

The break statement causes control to pass to the statement following the innermost enclosing while,do,for statements and switch statement.

(iii) continue:

Passes control

Syntax

continue;

It causes control to pass to the end of the innermost enclosing while, do, for statement at which point the loop continuation condition is re-evaluated.

(iv) goto statement:

The goto statement is an unconditional control statement which is off the execution program sequence by transfer of control to some other part of the program.

Syntax

goto lable;

where label is valid 'c' identifier used to label of the destination such that the control could transferred.

syntax of lable:

identifier:

```
#include<stdio.h>

#include<conio.h>

void main()

{

        int n,i=1;

        clrscr();

        printf("Enter any integer value : ");

        scanf("%d",&n);

        printf("Natural numbers from 1 to %d \n",n);

        lb:

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

                i++;

                if(i<=n)

                        goto lb;


        getch();

}
```

```
#include<stdio.h>

#include<conio.h>

void main()

{

	int n;

	clrscr();

	printf("Enter any integer value : ");

	scanf("%d",&n);

	if(n==1)

		goto lb1;

	if(n==2)

		goto lb2;

	printf("welcome");

	lb1:

		printf("\nLABEL1");

	lb2:

		printf("\nLABEL2");

	getch();

}
```

In default text mode there is 25 rows and 80 columns.

gotoxy <conio.h>

It is a function which is used to move the cursor to specified location in text window.

Declaration:

```
void gotoxy(int x,int y);
```

Note:

If the coordinates are invalid, the call to gotoxy is ignored.

```
#include<stdio.h>

#include<conio.h>

void main()

{

        clrscr();

        textmode(2);

        gotoxy(10,5);

        printf("WELCOME");

        gotoxy(30,20);

        printf("BDPS");

        getch();

}
```

Write a program to calculate student result by the following format

```
               -------------------------

                   BDPS SOFTWARE LIMITED

                      VIJAYAWADA

               -------------------------

              SNO :          SNAME :

              MARKS IN C :

              MARKS IN CPP :

              MARKS IN UNIX :

              TOTAL :        AVERAGE :

              RESULT :        DIVISION
```

```
*/

#include<stdio.h>

#include<conio.h>

void main()

{

        int sno,c,cpp,java,tot;

        char sname[80];

        float avg;

        clrscr();

        textmode(2);

        gotoxy(25,4);

        printf("--------------------------------");

        gotoxy(30,5);

        printf("BDPS SOFTWARE LIMITED");

        gotoxy(35,6);

        printf("VIJAYAWADA");

        gotoxy(25,7);

        printf("--------------------------------");

        gotoxy(25,8);

        printf("SNO : ");

        gotoxy(45,8);

        printf("SNAME : ");

        gotoxy(25,10);

        printf("MARKS IN C : ");

        gotoxy(25,12);

        printf("MARKS IN CPP : ");

        gotoxy(25,14);
```

```
printf("MARKS IN JAVA : ");

gotoxy(25,16);

printf("TOTAL : ");

gotoxy(45,16);

printf("AVERAGE : ");

gotoxy(25,18);

printf("RESULT : ");

gotoxy(45,18);

printf("DIVISION : ");

gotoxy(31,8);

scanf("%d",&sno);

gotoxy(53,8);

fflush(stdin);

gets(sname);

lb1:

        gotoxy(38,10);

        scanf("%d",&c);

        if(c<0 || c>100)

        {

                gotoxy(35,22);

                printf("INVALID MARKS");

                getch();

                gotoxy(35,22);

                printf("                ");

                gotoxy(38,10);

                printf("      ");

                goto lb1;
        }
lb2:
        gotoxy(40,12);
        scanf("%d",&cpp);
        if(cpp<0 || cpp>100)
        {
                gotoxy(35,22);
                printf("INVALID MARKS");
                getch();
                gotoxy(35,22);
                printf("                ");
                gotoxy(40,12);
                printf("      ");
                goto lb2;
        }
lb3:
        gotoxy(41,14);
        scanf("%d",&java);
        if(java<0 || java>100)
        {
                gotoxy(35,22);
                printf("INVALID MARKS");
                getch();
                gotoxy(35,22);
                printf("                            ");
                gotoxy(41,14);
                printf("        ");
```

```
                        goto lb3;
                }
        tot=c+cpp+java;
        avg=(float)tot/3;
        gotoxy(33,16);
        printf("%d",tot);
        gotoxy(55,16);
        printf("%.2f",avg);
        if(c>=50 && cpp>=50 && java>=50)
        {
                gotoxy(34,18);
                printf("pass");
                gotoxy(56,18);
                if(avg>=70)
                        printf("First");
                else
                        printf("Second");
        }
        else
        {
                gotoxy(34,18);
                printf("Fail");
                gotoxy(56,18);
                printf("No_DIVISION");
        }
        getch();
}
```

3. loop control statements

Loop:

The process of repeatedly executing a block of statements called loop.

'c' supports 3 types of loop control statements.

1. while loop

2. do while loop

3. for loop.

Any loop has 3 things

(i) Initialise the index.
(ii) Test condition.
(iii) Update the index.

1 While loop:

It is a conditional controlled loop statement in 'c' language.

Syntax

```
while(test condition)

{

        statements;

}
```

First the test condition is evaluated, and if it is true then the statement block will be executed.

After the execution of statements, the test condition is evaluated once again then if it is true the statement block will be executed once again.

The process of repeated execution continuous until the test condition finally becomes false.

```
#include<stdio.h>

#include<conio.h>

void main()

{

        int n,i=1;

        clrscr();
        printf("Enter any integer value : ");
        scanf("%d",&n);
        printf("Natural numbers from 1 to %d \n",n);
        while(i<=n)
        {
                printf("%d\t",i);
                i++;
        }
        getch();
}

#include<stdio.h>
#include<conio.h>
void main()
{
        int n,i;
        clrscr();
        printf("Enter any integer value : ");
        scanf("%d",&n);
        printf("Natural numbers from   %d to 1 \n",n);
        i=n;
        while(i>=1)
        {
                printf("%d\t",i);
                i--;
        }
        getch();
```

```
}

#include<stdio.h>

#include<conio.h>

void main()

{

	int n,i;

	clrscr();

	printf("Enter any integer value : ");

	scanf("%d",&n);

	printf("Even numbers from 1 to %d \n",n);

	i=1;

	while(i<=n)

	{

		if(i%2==0)

		{

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

		}

		i++;

	}

	printf("\n Odd numbers from 1 to %d \n",n);

	i=1;

	while(i<=n)

	{

		if(i%2==1)

		{

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

		}
```

```
                i++;
        }
        getch();
}


#include<stdio.h>
#include<conio.h>
void main()
{
        int n,i=1;
        clrscr();
        printf("Enter any integer value : ");
        scanf("%d",&n);
        printf("Factors of %d \n",n);
        while(i<=n)
        {
                if(n%i==0)
                {
                        printf("%d\t",i);
                }
                i++;
        }
        getch();
}


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

```
void main()
{
	int n,i=1;
	unsigned long fact=1;
	clrscr();
	printf("Enter any integer value : ");
	scanf("%d",&n);
	while(i<=n)
	{
		fact=fact*i;
		i++;
	}
	printf("Factorial of given number   =   %lu",fact);
	getch();
}


#include<stdio.h>
#include<conio.h>
void main()
{
	int n,i=1,c=0;
	clrscr();
	printf("Enter any integer value : ");
	scanf("%d",&n);
	while(i<=n)
	{
		if(n%i==0)
```

```
                    c++;
            i++;
    }
    if(c==2)
            printf("Given number is prime");
    else
            printf("Given number is not prime");
    getch();
}
#include<stdio.h>
#include<conio.h>
void main()
{
    int n,i=1,sum=0;
    clrscr();
    printf("Enter any integer value : ");
    scanf("%d",&n);
    while(i<=n/2)
    {
            if(n%i==0)
                    sum=sum+i;
            i++;
    }
    if(n==sum)
            printf("Given number is perfect ");
    else
            printf("Given number is not perfect");
```

```
        getch();
}

#include<stdio.h>
#include<conio.h>
void main()
{
        int n;
        unsigned long rev=0;
        clrscr();
        printf("Enter any integer value : ");
        scanf("%d",&n);
        while(n>0)
        {
                rev=rev*10+(n%10);
                n=n/10;
        }
        printf("Reverse number      =   %lu",rev);
        getch();
}

#include<stdio.h>
#include<conio.h>
void main()
{
        int n,m;
        unsigned long rev=0;
        clrscr();
        printf("Enter any integer value : ");
        scanf("%d",&n);
        m=n;
        while(m>0)
        {
                rev=rev*10+(m%10);
                m=m/10;
        }
        if(n==rev)
                printf("Given number is pallindrome");
        else
                printf("Given number is not pallindrome");
        getch();
}
```

```
#include<stdio.h>
#include<conio.h>
void main()
{
        int n,sum=0,r,m;
        clrscr();
        printf("Enter any integer value : ");
        scanf("%d",&n);
        m=n;
        while(m>0)
        {
                r=m%10;

                sum=sum+(r*r*r);

                m=m/10;

        }

        if(sum==n)

                printf("Given number is armstrong ");

        else

                printf("Given number is not armstrong");

        getch();

}


#include<stdio.h>

#include<conio.h>

void main()

{

        int n,sum=0;

        clrscr();

        textmode(2);

        printf("Enter any integer value : ");

        scanf("%d",&n);

        while(n>0)

        {

                sum=sum+(n%10);

                n=n/10;

        }

        printf("Sum of digits of given number   =    %d",sum);

        getch();
```

```
}


#include<stdio.h>

#include<conio.h>

void main()

{

        int n,sum;

        clrscr();

        textmode(2);

        printf("Enter any integer value : ");

        scanf("%d",&n);

        sum=n%9;

        if(sum==0)

                printf("Final sum is  9");

        else

                printf("Final sum  =  %d",sum);

        getch();

}


#include<stdio.h>

#include<conio.h>

void main()

{

        int n,pos,i=1,dg;

        clrscr();

        printf("Enter any integer value : ");

        scanf("%d",&n);
        printf("Enter specified position : ");
        scanf("%d",&pos);
        while(i<pos)
        {
                n=n/10;
                i++;
        }
        dg=n%10;
        printf("Specified position value   =   %d",dg);
        getch();
}

#include<stdio.h>
#include<conio.h>
void main()
{
        int a,b,min,i=1,gcd;
        clrscr();
        printf("Enter any two integer values : ");
        scanf("%d%d",&a,&b);
        min=a<b?a:b;
        while(i<=min)
        {
                if(a%i==0 && b%i==0)
                {
                        gcd=i;
                }
                i++;
        }
        printf("GCD of %d and %d is %d",a,b,gcd);
        getch();
}

#include<stdio.h>
#include<conio.h>
void main()
{
        int a,b,lcm;
        clrscr();
        printf("Enter any two integer values : ");
        scanf("%d%d",&a,&b);
        lcm=a>b?a:b;
        while(1)
        {
                if(lcm%a==0 && lcm%b==0)
                        break;
                lcm++;
        }
        printf("LCM of %d and %d is %d",a,b,lcm);
        getch();
}

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

```
void main()
{
	int i=1;
	clrscr();
	while(i<=5)
	{
		printf("%d\t",i);
		i++;
		if(i==3)
			break;
	}
	getch();
}

#include<stdio.h>
#include<conio.h>
void main()
{
	int i=0;
	clrscr();
	while(i<10)
	{
		i++;
		if(i==4 || i==5 || i==6)
			continue;
		printf("%d\t",i);
	}
	getch();
}


#include<stdio.h>
#include<conio.h>
void main()
{
	int n,i;
	clrscr();
	printf("Enter any integer value : ");
	scanf("%d",&n);
	printf("Even numbers from 1 to %d \n",n);
	i=0;
	while(i<n)
	{
		i++;
		if(i%2==1)
			continue;
		printf("%d\t",i);
	}
	printf("\n Odd numbers from 1 to %d \n",n);
	i=0;
	while(i<n)
	{
		i++;
		if(i%2==0)
			continue;
```

```
                printf("%d\t",i);
        }
        getch();
}
```

2 do while loop

It is an alternative form of while loop. Only difference between while and do-while is minimum number of execution of while is 0 minimum number if execution of do-while is 1.

Syntax:

```
do
{
        statements;
}while(test condition);
```

First the statement block will be executed and then test condition will be evaluated. If it is true then the statement block will be executed once again. The process of repeated execution continuous until the test condition finally becomes false.

```
#include<stdio.h>
#include<conio.h>
void main()
{
        int n,i=1;
        clrscr();
        printf("Enter any integer value : ");
        scanf("%d",&n);
        printf("Natural numbers from 1 to %d \n",n);
        do
        {
                printf("%d\t",i);
                i++;
        }while(i<=n);
        getch();
}
```

4. for loop:

It is the most commonly used loop statement in 'c' language. It consists of 3 expressions.

syntax

```
for(exp1;exp2;exp3)
{
        statements;
}
```

The first expression is used to initialise the index. The second expression is used to check whether the loop to be continued (or) not.

The third expression is used to change the index for further iteration (increment (or) decrement).

```
#include<stdio.h>
#include<conio.h>
void main()
{
        int n,i;
        clrscr();
        printf("Enter any integer value : ");
        scanf("%d",&n);
        printf("Natural numbers from 1 to %d \n",n);
        for(i=1;i<=n;i++)
        {
                printf("%d\t",i);
        }
        getch();
}


#include<stdio.h>
#include<conio.h>
void main()
{
        int n;
        unsigned long rev;
        clrscr();
        textmode(2);
        printf("Enter any integer value : ");
        scanf("%d",&n);
        for(rev=0;n>0;n=n/10)
        {
                rev=rev*10+(n%10);
        }
        printf("Reverse number     =   %lu",rev);
        getch();
}


#include<stdio.h>
#include<conio.h>
void main()
{
	int n,t1=0,t2=1,t,i;

	clrscr();

	printf("Enter number of terms : ");

	scanf("%d",&n);

	printf("Fibbonacci series \n");

	printf("%d\t%d",t1,t2);

	for(i=1;i<=n-2;i++)

	{

		t=t1+t2;

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

		t1=t2;

		t2=t;

	}

	getch();

}


#include<stdio.h>

#include<conio.h>

void main()

{

	int n,i;

	clrscr();

	printf("Enter any integer value : ");

	scanf("%d",&n);

	for(i=1;i<=20;i++)

	{

		printf("\n %d * %d = %d",n,i,n*i);
```

```
            }
            getch();
}
```

Textcolor:

textcolor selects a new character color in text mode

Declaration:

void textcolor(int newcolor);

textbackground

textbackground selects a new text background color

Declaration:

void textbackground(int newcolor);

Textmode

Changes screen mode (in text mode)

Declaration:

void textmode(int newmode);

eg

textmode(2);   25 rows 80 columns.

textmode(1);   25 rows 40 columns.

```
  COLORS (text mode)
 ßßßßßßßßßßßßßßßßßßßßß³Back-³Fore-
  Constant     ³Value³grnd?³grnd?
 ÍÍÍÍÍÍÍÍÍÍÍÍÍØÍÍÍÍÍØÍÍÍÍÍØÍÍÍÍÍ
  BLACK        ³  0  ³ Yes ³ Yes
  BLUE         ³  1  ³ Yes ³ Yes
  GREEN        ³  2  ³ Yes ³ Yes
```

CYAN ³ 3 ³ Yes ³ Yes

RED ³ 4 ³ Yes ³ Yes

MAGENTA ³ 5 ³ Yes ³ Yes

BROWN ³ 6 ³ Yes ³ Yes

LIGHTGRAY ³ 7 ³ Yes ³ Yes

DARKGRAY ³ 8 ³ No ³ Yes

LIGHTBLUE ³ 9 ³ No ³ Yes

LIGHTGREEN ³ 10 ³ No ³ Yes

LIGHTCYAN ³ 11 ³ No ³ Yes

LIGHTRED ³ 12 ³ No ³ Yes

LIGHTMAGENTA ³ 13 ³ No ³ Yes

YELLOW ³ 14 ³ No ³ Yes

WHITE ³ 15 ³ No ³ Yes

ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÅÄÄÄÄÄÄÅÄÄÄÄÄÄÄÅÄÄÄÄÄÄÄ

BLINK ³128 ³ No ³ ***

*** To display blinking characters in text mode, add BLINK to the foreground color. (Defined in CONIO.H.)

cprintf:

It is same as printf and is used to apply colors in text mode.

```
#include<stdio.h>

#include<conio.h>

void main()

{

        clrscr();

        textmode(2);

        textbackground(4);
        textcolor(14+128);

        cprintf("Welcome");

        getch();

}
```

delay   <dos.h>

Suspends execution for interval (milliseconds)

Declaration:

void delay(unsigned milliseconds);

_setcursortype

Selects cursor appearance

Declaration:

void _setcursortype(int cur_t);

Remarks

Sets the cursor type to one of the following:

```
 _NOCURSOR       0    (turns off the cursor)

 _SOLIDCURSOR    1    (solid block cursor)

 _NORMALCURSOR   2    (normal underscore cursor)
```

kbhit

Checks for currently available keystrokes

Declaration

int kbhit(void);

```
#include<stdio.h>

#include<conio.h>

#include<dos.h>
```

```
void main()

{

        int c1=1,c2=75,r1=1,r2=25;

        clrscr();

        textmode(2);

        while(!kbhit())

        {

                _setcursortype(0);

                clrscr();

                gotoxy(c1,12);

                printf("LITTLE");

                gotoxy(c2,12);

                printf("MASTER");

                gotoxy(37,r1);

                printf("SACHIN");

                gotoxy(35,r2);

                printf("POWER STAR");

                c1++;

                c2--;

                r1++;

                r2--;

                delay(100);

                if(c1>75)

                {

                        c1=1;

                        c2=75;

                }
                if(r1>25)
                {
                        r1=1;
                        r2=25;
                }
        }
}
```

Nested loops:

Using a loop statement with in another loop is called as nested loop.

```
#include<stdio.h>
#include<conio.h>
void main()
{
        int n,m,i;
        unsigned long rev;
        clrscr();
        printf("Enter any integer value : ");
        scanf("%d",&n);
        printf("Pallindrome numbers from 1 to %d \n",n);
        for(i=1;i<=n;i++)
        {
                m=i;
                rev=0;
                while(m>0)
                {
                        rev=rev*10+(m%10);
                        m=m/10;
                }
                if(rev==i)
                        printf("%d\t",i);
        }
        getch();
}

#include<stdio.h>
#include<conio.h>
void main()
{
        int n,sum,r,m,i;
        clrscr();
        printf("Enter any integer value : ");
        scanf("%d",&n);
        printf("Armstrong numbers from 1 to %d \n",n);
        for(i=1;i<=n;i++)
        {
                m=i;
                sum=0;
                while(m>0)
                {
                        r=m%10;
                        sum=sum+(r*r*r);
                        m=m/10;
                }
                if(sum==i)
                        printf("%d\t",i);
        }
        getch();
}


#include<stdio.h>
#include<conio.h>
void main()
{
        int n,i,j,c;
        clrscr();
        textmode(2);
        printf("Enter any integer value : ");
        scanf("%d",&n);
        printf("Prime numbers form 1 to %d \n",n);
        for(i=1;i<=n;i++)
        {
                c=0;
                for(j=1;j<=i;j++)
                {
                        if(i%j==0)
                                c++;
                }
                if(c==2)
                        printf("%d\t",i);
```

```
	}
	getch();
}


#include<stdio.h>
#include<conio.h>
void main()
{
	int n,i,j,sum;
	clrscr();
	textmode(2);
	printf("Enter any integer value : ");
	scanf("%d",&n);
	printf("Perfect numbers from 1 to %d\n",n);
	for(i=1;i<=n;i++)
	{
		sum=0;
		for(j=1;j<=i/2;j++)
		{
			if(i%j==0)
				sum=sum+j;
		}
		if(sum==i)
			printf("%d\t",i);
	}
	getch();
}
```

```
/*
w
we
wel
welc
welco
welcom
welcome
*/

#include<stdio.h>
#include<conio.h>
void main()
{
	char st[10]="welcome";
	int i;
	clrscr();
	for(i=1;i<=7;i++)
	{
		printf("\n %.*s",i,st);
	}
	getch();
}



/*
```

write a program to generate the following pattern.

n=5

```
* * * * *

* * * * *

* * * * *

* * * * *

* * * * *
```

```
*/
#include<stdio.h>
#include<conio.h>
void main()
{
	int n,i,j;
	clrscr();
	printf("Enter any integer value : ");
	scanf("%d",&n);
	for(i=1;i<=n;i++)
	{
		for(j=1;j<=n;j++)
		{
			printf("%3c",'*');
		}
		printf("\n");
	}
	getch();
}
```

```
/*

Write a program to generate the following pattern.

n=5

*

* *

* * *

* * * *

* * * * *


*/

#include<stdio.h>

#include<conio.h>

void main()

{

        int n,i,j;

        clrscr();

        printf("Enter any integer value : ");

        scanf("%d",&n);

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

        {

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

                {

                        printf("%3c",'*');

                }

                printf("\n");

        }

        getch();
```

```
}

/*

write a program to generate the following pattern.
n=5

1
2 2
3 3 3
4 4 4 4
5 5 5 5 5

*/

#include<stdio.h>
#include<conio.h>
void main()
{
	int n,i,j;
	clrscr();
	printf("Enter any integer value : ");
	scanf("%d",&n);
	for(i=1;i<=n;i++)
	{
		for(j=1;j<=i;j++)
		{
```

```
                    printf("%3d",i);
                }
                printf("\n");
        }
        getch();
}
1 2
1 2 3
1 2 3 4
1 2 3 4 5
*/
#include<stdio.h>
#include<conio.h>
void main()
{
        int n,i,j;
        clrscr();
        printf("Enter any integer value : ");
        scanf("%d",&n);
        for(i=1;i<=n;i++)
        {
                for(j=1;j<=i;j++)
                {
                        printf("%3d",j);
                }
                printf("\n");
        }
```

```
	getch();
}


/*
Write a program to generate the following pattern.
n=5
    *
   * *
  * * *
 * * * *
* * * * *
*/
#include<stdio.h>
#include<conio.h>
void main()
{
	int n,i,j,s;
	clrscr();
	printf("Enter any integer value : ");
	scanf("%d",&n);
	s=n*2;
	for(i=1;i<=n;i++)
	{
		printf("%*c",s,32);
		for(j=1;j<=i;j++)
		{
			printf("%4c",'*');
```

```
                }
                printf("\n");
                s=s-2;
        }
        getch();
}


/*
Write a program to generate the following pattern.
n=5
    *
   * *
  * * *
 * * * *
* * * * *
 * * * *
  * * *
   * *
    *
*/
#include<stdio.h>
#include<conio.h>
void main()
{
        int n,i,j,s,k=1;
        clrscr();
        printf("Enter any integer value : ");
	scanf("%d",&n);

	s=n*2;

	for(i=1;i<n*2;i++)

	{

		printf("%*c",s,32);

		for(j=1;j<=k;j++)

		{

			printf("%4c",'*');

		}

		printf("\n");

		if(i<n)

		{

			k++;

			s=s-2;

		}

		else

		{

			k--;

			s=s+2;

		}

	}

	getch();

}

/*


n=5

1 2 3 4 5
```

```
5 1 2 3 4

5 4 1 2 3

5 4 3 1 2

5 4 3 2 1


*/

#include<stdio.h>

#include<conio.h>

void main()

{

        int n,i,j,m;

        clrscr();

        printf("Enter any integer value : ");

        scanf("%d",&n);

        m=n;

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

        {

                for(j=n;j>m;j--)

                {

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

                }

                for(j=1;j<=m;j++)

                {

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

                }

                m--;

                printf("\n");
        }
        getch();
}
/*
n=5
1  2  3  4  5
10 9  8  7  6
11 12 13 14 15
20 19 18 17 16
21 22 23 24 25
*/

#include<stdio.h>
#include<conio.h>
void main()
{
        int n,i,j,k=1;
        clrscr();
        printf("Enter any integer value : ");
        scanf("%d",&n);
        for(i=1;i<=n;i++)
        {
                for(j=1;j<=n;j++)
                {
                        printf("%d\t",k);
                        if(j<n)
                        {
```

```
                if(i%2==0)
                    k--;
                else
                    k++;
            }
        }
        k=k+n;
        printf("\n");
    }
    getch();
}
/*
A B C D E F G F E D C B A
A B C D E F   F E D C B A
A B C D E       E D C B A
A B C D           D C B A
A B C               C B A
A B                   B A
A                       A
*/

#include<stdio.h>
#include<conio.h>
void main()
{
    int i,j,p,x=7,y=7;
    clrscr();
```

```
        for(i=1;i<=7;i++)
        {
                p=65;
                for(j=1;j<=13;j++)
                {
                        if(i>1 && j>=x && j<=y)
                                printf("%3c",32);
                        else
                                printf("%3c",p);
                        if(j<7)
                                p++;
                        else
                                p--;
                }
                printf("\n");
                if(i>1)
                {
                        x--;
                        y++;
                }
        }
        getch();
}

/*
A B C D E F G F E D C B A
A B C D E F   F E D C B A
```

```
A B C D E      E D C B A

A B C D         D C B A

A B C            C B A

A B               B A

A                  A

A B               B A

A B C            C B A

A B C D         D C B A

A B C D E      E D C B A

A B C D E F   F E D C B A

A B C D E F G F E D C B A

*/

#include<stdio.h>

#include<conio.h>

void main()

{

        int i,j,p,x=7,y=7;

        clrscr();

        textmode(2);

        for(i=1;i<=13;i++)

        {

                p=65;

                for(j=1;j<=13;j++)

                {

                        if(i>1 && j>=x && j<=y)

                        {

                                textcolor(2+128);
                cprintf("%3c",'*');

            }

            else

            {

                textcolor(14);

                cprintf("%3c",p);

            }

            if(j<7)

                p++;

            else

                p--;

        }

        printf("\n");

        if(i>1)

        {

            if(i<7)

            {

                x--;

                y++;

            }

            else

            {

                x++;

                y--;

            }

        }

    }
```

```
	getch();
}
```

FUNCTIONS

Definition:

It is a self contained block statements and it can be used several multiple times of program but defined only once.

Library functions

These functions which are in built with the 'c' compiler is known as library functions.

User defined functions:

User can define functions to do a task relevant to their programs. Such functions are called as user defined functions.

Any function has 3 things

1. Function declaration
2. Function definition
3. Function calling

In case of predefined functions, the function declaration is in headerfiles function definition is in 'c' libraries.calling is in your source program.

But in case of user defined functions all the 3 things are in your source program.

Function declaration:

```
returntype func_name([arg_list]);
```

Function definition:

```
returntype func_name([arg_list])
{
	body;
}
```

Function calling:

```
func_name([para_list]);
```

The arguments which are given at the time of function declaration (or) definition are called as arguments (or) formal arguments.

The arguments which are given at the time of function calling are called as actual arguments (or) parameters.

void empty datatype.

```
#include<stdio.h>

#include<conio.h>

void main()

{

	void func();    //  declaration

	clrscr();

	textmode(2);

	func();       // calling

	getch();

}

void func()          //definition

{

	printf("Welcome to 'c' functions");

}
```

Rules for creating and accessing user defined function:

1. A function can be called by any number of times.

2. A function may or may not receive arguments.

3. A function may or may notreturn value.

4. If a function does not return any value, the function return datatype will be specified as void.

5. If a function returns a value only one value can be returned.

6. If a function returns a value the returning value must be return with a return statement.

7. If a function returns a value the execution of return statement should be last.

8. A function returns an integer value by default.

9. If a function returns a value, the returning value should match with the function datatype.

10. A function is defined after (or) before main function.

11. Before calling a function, the function declaration (or) definition must and should.

12. If a function declaration is specified before the function call, the function definition can be specified any where in the program.

13. If a function definition is specified before the function call, then the function declaration is not necessary.

14. If a function is executed, then the function is call by its name.

15. The function definition should not be terminated with semicolon.

Return:

Exits immediately from the currently executing function to the calling routine, optionally returning a value.

Syntax:

```
return [ <expression> ] ;

#include<stdio.h>

#include<conio.h>

void main()

{

        int func(int,int);    //  declaration

        int a,b,s;

        clrscr();

        printf("Enter any two integer values : ");

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

        s=func(a,b);       // calling

        printf("Sum is   =  %d",s);

        getch();

}

int func(int x,int y)          //definition

{

        return x+y;
}
```

1. Function with no arguments and no return value:

   In this type the function has no arguments, it does not receive any data from the calling function similarly it does not return any value, the calling function does not receive any data from the called function. so there is no data communication between calling and called functions.

```
#include<stdio.h>

#include<conio.h>

void main()

{

        void sum();

        clrscr();

        sum();

        getch();

}

void sum()

{

        int a,b;

        printf("Enter any two integer values : ");

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

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

}
```

2. Function with arguments and no return value:

   In this type the function has some arguments, it receives data from the calling function but it does not return any value, the calling function does not receive any data from the called function. so there is one way data communication between calling and called functions.

```
#include<stdio.h>

#include<conio.h>

void main()

{

        void sum(int,int);

        int a,b;

        clrscr();

        printf("Enter any two integer values : ");

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

        sum(a,b);

        getch();

}

void sum(int x,int y)

{

        printf("Addition is    =  %d",x+y);

}
```

3. Function with arguments and return value:

In this type the function has some arguments; it receives data from the calling function.

Similarly it returns a value the calling function receives data from the called function so there is two way data communication between calling and called functions.

```
#include<stdio.h>

#include<conio.h>

void main()

{

        int sum(int,int);

        int a,b,s;

        clrscr();

        printf("Enter any two integer values : ");
        scanf("%d%d",&a,&b);

        s=sum(a,b);

        printf("Sum is      = %d",s);

        getch();

}

int sum(int x,int y)

{

        return x+y;

}
```

4. Function with no arguments and return a value

In this type the function has no arguments, it does not receive any data from the calling function but it returns a value, the calling function receives data from the called function so there is one way data communication between calling and called functions.

```
#include<stdio.h>

#include<conio.h>

void main()

{

        int sum();

        int s;

        clrscr();

        s=sum();

        printf("Sum is      = %d",s);

        getch();

}

int sum()

{

        int a,b;
        printf("Enter any two integer values : ");

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

        return a+b;

}


#include<stdio.h>

#include<conio.h>

void main()

{

        void nat(int);

        int n;

        clrscr();

        printf("Enter any integer value : ");

        scanf("%d",&n);

        printf("Natural numbers from 1 to %d \n",n);

        nat(n);

        getch();

}

void nat(int x)

{

        int i;

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

        {

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

        }

}
```

```
#include<stdio.h>

#include<conio.h>

void main()

{

        void factors(int);

        int n;

        clrscr();

        printf("Enter any integer value : ");

        scanf("%d",&n);

        printf("Factors of %d \n",n);

        factors(n);

        getch();

}

void factors(int x)

{

        int i;

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

        {

                if(x%i==0)

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

        }

}


#include<stdio.h>

#include<conio.h>

void main()

{
        void fibbo(int);

        int n;

        clrscr();

        printf("Enter number of terms : ");

        scanf("%d",&n);

        printf("Fibbonacci series \n");

        fibbo(n);

        getch();

}

void fibbo(int x)

{

        int i,t1=0,t2=1,t;

        printf("%d\t%d",t1,t2);

        for(i=1;i<=x-2;i++)

        {

                t=t1+t2;

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

                t1=t2;

                t2=t;

        }

}


#include<stdio.h>

#include<conio.h>

void main()

{

        int maxval(int,int);
        int a,b,max;
        clrscr();
        printf("Enter any two integer values : ");
        scanf("%d%d",&a,&b);
        max=maxval(a,b);
        printf("Maximum value   =   %d",max);
        getch();
}
int maxval(int x,int y)
{
        if(x>y)
                return x;
        else
                return y;
}

#include<stdio.h>
#include<conio.h>
void main()
{
        int maxval(int,int);
        int a,b,c,max;
        clrscr();
        printf("Enter any three integer values : ");
        scanf("%d%d",&a,&b,&c);
        /*     max=maxval(a,b);
            max=maxval(max,c);     */
```

```
	max=maxval(maxval(a,b),c);
	printf("Maximum value   =   %d",max);
	getch();
}
int maxval(int x,int y)
{
	if(x>y)
		return x;
	else
		return y;
}

#include<stdio.h>
#include<conio.h>
void main()
{
	unsigned long revnum(int);
	int n;
	unsigned long r;
	clrscr();
	printf("Enter any integer value : ");
	scanf("%d",&n);
	r=revnum(n);
	printf("Reverse number     =   %lu",r);
	getch();
}
unsigned long revnum(int x)
```

```
{
	unsigned long r=0;
	while(x>0)
	{
		r=r*10+(x%10);
		x=x/10;
	}
	return r;
}

#include<stdio.h>
#include<conio.h>
void main()
{
	unsigned long fact(int);
	int n;
	unsigned long f;
	clrscr();
	printf("Enter any integer value : ");
	scanf("%d",&n);
	f=fact(n);
	printf("Factorial of given number     =  %lu",f);
	getch();
}
unsigned long fact(int x)
{
	unsigned long f=1;
	int i;

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

	{

		f=f*i;

	}

	return f;

}

#include<stdio.h>

#include<conio.h>

void main()

{

	unsigned long fact(int);

	int n,r,ncr;

	clrscr();

	printf("Enter n and r values : ");

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

	ncr=fact(n)/(fact(r)*fact(n-r));

	printf("NCR VALUE     =  %d",ncr);

	getch();

}

unsigned long fact(int x)

{

	unsigned long f=1;

	int i;

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

	{

		f=f*i;
	}
	return f;
}
#include<stdio.h>
#include<conio.h>
void main()
{
	int sumdg(unsigned long);
	unsigned long n;
	int sum;
	clrscr();
	printf("Enter any integer value : ");
	scanf("%lu",&n);
	sum=sumdg(n);
	printf("Sum of digits of given number  =  %d",sum);
	getch();
}
int sumdg(unsigned long x)
{
	int sum=0;
	while(x>0)
	{
		sum=sum+(x%10);
		x=x/10;
	}
	return sum;
}
```

```
#include<stdio.h>

#include<conio.h>

void main()

{

	unsigned long fact(int);

	int n,i,j,ncr,s;

	clrscr();

	textmode(2);

	printf("Enter value of n : ");

	scanf("%d",&n);

	s=n*3;

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

	{

		printf("%*c",s,32);

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

		{

			ncr=fact(i) / (fact(j) * fact(i-j) );

			printf("%6d",ncr);

		}

		printf("\n");

		s=s-3;

	}

	getch();

}

unsigned long fact(int x)

{

	unsigned long f=1;
```

```
        int i;

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

        {

                f=f*i;

        }

        return f;

}
```

Storage classes

By the declaration statement the memory is allocated tempararily for all the variables.The size of memory varies with respect to the type of variable.The availability of the variables for access depends on its declaration type.The storage class specifiers are used to specify the life and scope of variables with in block,functions and entire source program.

'c' supports 4 types of storage classes.

1. automatic variables

2. static variables

3. global (or) external variables.

4. register varaiables.

1.Automatic variables:

Thses variables are declared inside a function block.

storage	: main memory
default value	: garbage value
scope	: local to the block in which it is define.
life	: Till the control remains with in the block in which it is defined.
keyword	: auto

```
#include<stdio.h>

#include<conio.h>
```

```
void main()

{

        auto int a,b;

        clrscr();

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

        printf("\n b=%d",b);

        getch();

}
```

Note

If no storage class specifier before the declaration of a variable inside a function block,by default it takes auto storage class.

```
#include<stdio.h>

#include<conio.h>

void main()

{

        auto int n;

        clrscr();

        printf("Enter any integer value : ");

        scanf("%d",&n);

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

        getch();

}
```

3. static varibles:

Memory of static variable remains unchanged until the end of the program.

storage : main memory

default value : 0

scope : local to the block in which it is define.

life : The memory of static varible persists between different function call.that means it can not reintialise between the different function call.

keyword : static.

```
#include<stdio.h>

#include<conio.h>

void main()

{

        static int a,b;

        clrscr();

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

        printf("\nb=%d",b);

        getch();

}
```

```
#include<stdio.h>

#include<conio.h>

void disp()

{

        static int n=1;

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

        n++;

}

void main()

{

        int i;
```

```
	clrscr();

	for(i=1;i<=10;i++)

	{

		disp();

	}

	getch();

}
```

3.Global variables (or) extern varaibles

These variables are both alive and active through out the entire program.

storage	: main memory
default value	: 0
scope	: global
life	: As long as the program execution does not come to end.
keyword	: extern.

```
#include<stdio.h>

#include<conio.h>

int a,b;

void main()

{

	clrscr();

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

	printf("\nb=%d",b);

	getch();

}
```

4.register varibles

we can use register variables for frequently used variables to improve the faster execution of the program.

storage	: cpu
default value	: garbage value
scope	: local to the block in which it is define.
life	: Till the control remains with in the block in which it is defined.
keyword	: register.

```
#include<stdio.h>

#include<conio.h>

void main()

{

            register int a,b;

            clrscr();

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

            printf("\n b=%d",b);

            getch();

}
```

Note

Register storage class support only integral data.

Recursive function (or) recursion function:

calling a function with in the same function is called recursion function (or) recursive function.

If we want to work with recursive function we must following the 2 accepts.

1.calling by itself.

2.termination condition.

```
#include<stdio.h>

#include<conio.h>
```

```
void main()

{

	void nat(int);

	int n;

	clrscr();

	printf("Enter any integer value : ");

	scanf("%d",&n);

	printf("Natural numbers from 1 to %d \n",n);

	nat(n);

	getch();

}

void nat(int x)

{

	if(x>1)

		nat(x-1);

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

}


#include<stdio.h>

#include<conio.h>

void main()

{

	unsigned long fact(int);

	int n;

	unsigned long f;

	clrscr();

	printf("Enter any integer value : ");
```

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

        f=fact(n);

        printf("Factorial of given number   =   %lu",f);

        getch();

}

unsigned long fact(int x)

{

        if(x<=1)

                return 1;

        else

                return x*fact(x-1);

}

#include<stdio.h>

#include<conio.h>

void main()

{

        unsigned long revnum(int);

        int n;

        unsigned long rev;

        clrscr();

        printf("Enter any integer value : ");

        scanf("%d",&n);

        rev=revnum(n);

        printf("Reverse number    =   %lu",rev);

        getch();

}

unsigned long revnum(int x)
```

```
{
        static unsigned long rev=0;
        rev=rev*10+(x%10);
        x=x/10;
        if(x>0)
                revnum(x);
        return rev;
}

#include<stdio.h>
#include<conio.h>
void main()
{
        void fibbo(int);
        int n;
        clrscr();
        printf("Enter number of terms : ");
        scanf("%d",&n);
        printf("Fibbonacci series \n");
        fibbo(n);
        getch();
}
void fibbo(int x)
{
        static int i,t1=0,t2=1,t;
        if(t==0)
        {
                printf("%d\t%d",t1,t2);

                x=x-2;

        }

        t=t1+t2;

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

        x=x-1;

        if(x>=1)

        {

                t1=t2;

                t2=t;

                fibbo(x);

        }

}


#include<stdio.h>

#include<conio.h>

#include<process.h>

void main()

{

        static int i=1,n;

        if(i==1)

        {

                clrscr();

                printf("Enter any integer value : ");

                scanf("%d",&n);

                printf("Natural numbers from 1 to %d \n",n);

        }
```

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

	i++;

	if(i<=n)

		main();

	getch();

	exit(0);

}
```

math.h functions

----

abs	<math.h>

abs (a macro) gets the absolute value of an integer.

declaration

int abs(int x);

```
#include<stdio.h>

#include<conio.h>

#include<math.h>

void main()

{

	int n,a;

	clrscr();

	printf("Enter any integer value : ");

	scanf("%d",&n);

	a=abs(n);

	printf("Absolute value of given number    =   %d",a);

	getch();

}
```

pow <math.h>

It calculates exponential value of given base and power.

declaration

double pow(double x,double y);

```
#include<stdio.h>

#include<conio.h>

#include<math.h>

void main()

{

	int b,p;

	double e;

	clrscr();

	printf("Enter base and power : ");

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

	e=pow(b,p);

	printf("Exponential value of %d and %d is %.3lf",b,p,e);

	getch();

}
```

sqrt <math.h>

It calculates square root of given number.

declaration

double sqrt(double x);

```
#include<stdio.h>

#include<conio.h>
```

```
#include<math.h>
void main()
{
	int n;
	double s;
	clrscr();
	printf("Enter any integer value : ");
	scanf("%d",&n);
	s=sqrt(n);
	printf("Square root of given number    =   %.3lf",s);
	getch();
}
```

ceil	<math.h>

It rounds up the given value.

declaration

double ceil(double x);

floor	<math.h>

It rounds down the given value.

declaration

double floor(double x);

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

```
{
        float n,c,f;
        clrscr();
        printf("Enter any float value : ");
        scanf("%f",&n);
        c=ceil(n);
        f=floor(n);
        printf("Ceil value      =   %.2f",c);
        printf("\n Floor value   =   %.2f",f);
        getch();
}

#include<stdio.h>
#include<conio.h>
#include<math.h>
#include<process.h>
void main()
{
        float a,b,c,s,area;
        clrscr();
        printf("Enter any three sides : ");
        scanf("%f%f%f",&a,&b,&c);
        if( (a+b)<=c  || (b+c) <=a || (c+a)<=b )
        {
                printf("Unable to form triangle ");
                getch();
                exit(0);
```

```
            }
            s=(a+b+c)/2;
            area=sqrt(s*(s-a)*(s-b)*(s-c));
            printf("Area of triangle   =    %.2f",area);
            getch();
}
```

ARRAYS

A group of dataitems of same datatype stored in a continuous memory location and it can be represented by a sigle identifier,that means a group of data items that shared a common name.

A particular value is incated by writing a no called index (or) subscript in square braces after the array name. 'C' supports three types of arrays.

1. Single dimensional array.

2. Two dimensional array.

3. Three dimensional array.

1.Single dimensional array:

A list of items can be given one variable name using only one index.such variable is called single dimensional array.In single dimensional array the elements are represent one after other in edges of memory bytes.

declaration

datatype arr_name[size];

eg

int a[5];

elements

a[0],a[1],a[2],a[3] and a[4].

In any array the array index is 0 to (n-1).

intialisation

datatype arr_name[size]={value-1,value-2,----------------value-n};

optional

eg

```
int a[5]={1,2,3,4,5};

int a[]={100,200,300};

float ft[4]={123.78,1.1,2.3,4.89};
```

```
#include<stdio.h>
#include<conio.h>
void main()
{
	int a[5]={11,34,78,45,21},i;
	clrscr();
	for(i=0;i<5;i++)
	{
		printf("\n a[%d]=%d",i,a[i]);
	}
	getch();
}
```

```
#include<stdio.h>
#include<conio.h>
#include<process.h>
void main()
{
	int a[20],n,i;
	clrscr();
	printf("Enter number of elements : ");
	scanf("%d",&n);
	if(n>20)
	{
		printf("Array index out of bounds ");
		getch();
		exit(0);
	}
	printf("Enter array elements \n");
	for(i=0;i<n;i++)
	{
		printf("Enter value into a[%d]=",i);
		scanf("%d",&a[i]);
	}
	printf("Array elements \n");
	for(i=0;i<n;i++)
	{
		printf("\n Given value in a[%d]=%d",i,a[i]);
	}
	getch();
}

#include<stdio.h>
#include<conio.h>
void main()
{
	int a[20],n,i;
	clrscr();
	printf("Enter number of elements : ");

	scanf("%d",&n);

	printf("Enter array elements \n");

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

	{

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

	}

	printf("Array elements \n");

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

	{

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

	}

	getch();

}


#include<stdio.h>

#include<conio.h>

void main()

{

	int a[20],n,i;

	clrscr();

	printf("Enter number of elements : ");

	scanf("%d",&n);

	printf("Enter array elements \n");

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

	{

		scanf("%d",&a[i]);
	}
	printf("Array elements in reverse order \n");
	for(i=n-1;i>=0;i--)
	{
		printf("%d\t",a[i]);
	}
	getch();
}


#include<stdio.h>
#include<conio.h>
void main()
{
	int a[20],n,i,max,min;
	clrscr();
	printf("Enter number of elements : ");
	scanf("%d",&n);
	printf("Enter array elements \n");
	for(i=0;i<n;i++)
	{
		scanf("%d",&a[i]);
	}
	max=a[0];
	min=a[0];
	for(i=1;i<n;i++)
	{
		if(a[i]>max)
```

```
                max=a[i];
        if(a[i]<min)
                min=a[i];
    }
    printf("Maximum value    =   %d",max);
    printf("\nMinimum value  =   %d",min);
    getch();
}
```

ÄÄÄ

write a program to accept a single dimensional integer array and search specified element in the given array

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int a[20],n,i,se,ck=0;
    clrscr();
    printf("Enter no of elements :");
    scanf("%d",&n);
    printf("Enter array elements \n");
    for(i=0;i<n;i++)
    {
        scanf("%d",&a[i]);
    }
    printf("Enter element to search :");
    scanf("%d",&se);
    for(i=0;i<n;i++)
```

```
	{
		if(a[i]==se)
		{
			ck=1;
			printf("Element is found at a[%d]",i);
		}
	}
	if(ck==0)
		printf("Element not found");
	getch();
}
```

ÄÄ

write a program to accept a single dimensional integer array and delete specified element in the given array.

```
#include<stdio.h>
#include<conio.h>
void main()
{
	int a[20],n,i,j,de,ck=0;
	clrscr();
	printf("Enter no of elements :");
	scanf("%d",&n);
	printf("Enter array elements \n");
	for(i=0;i<n;i++)
	{
		scanf("%d",&a[i]);
	}
	printf("Enter element to delete :");
```

```
        scanf("%d",&de);
        for(i=0;i<n;i++)
        {
                if(a[i]==de)
                {
                        ck=1;
                        for(j=i;j<n;j++)
                        {
                                a[j]=a[j+1];
                        }
                        n--;
                }
        }
        if(ck==0)
                printf("Element not found");
        else
        {
                printf("Array elements after deletion\n");
                for(i=0;i<n;i++)
                {
                        printf("%d\t",a[i]);
                }
         }
        getch();
}
```

ÄÄ

write a program to accept a single dimensional integer array and insert an element in the given array at specified position.

```
#include<stdio.h>

#include<conio.h>

#include<process.h>

void main()

{

        int a[20],n,i,pos,ie,t;

        clrscr();

        printf("Enter no of elements :");

        scanf("%d",&n);

        printf("Enter array elements \n");

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

        {

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

        }

        printf("Enter position:");

        scanf("%d",&pos);

        if(pos>n)

        {

                printf("Invalid position ");

                getch();

                exit(0);

        }

        printf("Array elements before insertion \n");

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

        {

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

        }
```

```
	printf("\nEnter element to insert :");
	scanf("%d",&ie);
	if(pos==n)
	{
		a[pos]=ie;   //a[n]=ie;
		n++;
	}
	else
	{
		for(i=pos;i<=n;i++)
		{
			t=a[i];
			a[i]=ie;
			ie=t;
		}
		n++;
	}
	printf("Array elements after insertion \n");
	for(i=0;i<n;i++)
	{
		printf("%d\t",a[i]);
	}
	getch();
}
```

ÄÄÄ

write a program to accept a single dimensional integer array and display array elements in ascending order.

```
#include<stdio.h>

#include<conio.h>

void main()

{

        int a[20],n,i,j,t;

        clrscr();

        printf("Enter no of elements :");

        scanf("%d",&n);

        printf("Enter array elements \n");

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

        {

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

        }

        printf("Given elements before sorting\n");

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

        {

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

        }

        /* sorting */

        for(i=0;i<n-1;i++)

        {

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

                {

                        if(a[i]>a[j])

                        {

                                t=a[i];

                                a[i]=a[j];
                                a[j]=t;
                        }
                }
        }
        printf("\nGiven elements after sorting\n");
        for(i=0;i<n;i++)
        {
                printf("%d\t",a[i]);
        }
        getch();
}
```

ÄÄ

write a program to accept a single dimensional integer array,it can not accept duplicate elements in the given array.

```
#include<stdio.h>
#include<conio.h>
void main()
{
        int a[20],n,i,j;
        clrscr();
        printf("Enter no of elements :");
        scanf("%d",&n);
        printf("Enter array elements \n");
        for(i=0;i<n;i++)
        {
                lb:
                        scanf("%d",&a[i]);
                        for(j=0;j<i;j++)
                        {
                                if(a[i]==a[j])
                                {
                                        printf("Element already exists\n");
                                        printf("Enter new element :");
                                        goto lb;
                                }
                        }
        }
        printf("Given elements \n");
        for(i=0;i<n;i++)
        {
                printf("%d\t",a[i]);
        }
        getch();
}
```

Two dimensional arrays

Two dimnsional array can store a table of values which contains rows and columns.

In Two dimensional array we use two index values one for rows and another for columns.

declaration

datatype arr_name[rowsize][colsize];

eg

int a[2][3];

elements

```
a[0][0]    a[0][1]    a[0][2]

a[1][0]    a[1][1]    a[1][2]
```

Intialisation

Form1

```
datatype arr_name[rowsize][colsize]={value-1,value-2,value-3,----value-n};
                                    optional
```

eg

```
int a[2][3]={1,2,3,4,5,6};
```

Form2

```
datatype arr_name[rowsize][colsize]={ {value-1,value-2,----value-n},
                                       {value-1,value-2,----value-n},
                                       -----------------------------
                                       {value-1,value-2,----value-n}
                                     };
```

eg

```
int a[3][3]={  {1,2,3},{4,5,6},{7,8,9} };
```

```
#include<stdio.h>

#include<conio.h>

void main()

{

        /*    int a[3][3]={1,2,3,4,5,6,7,8,9};    */


        int a[3][3]={ {1,2,3},{4,5,6},{7,8,9} };


        int i,j;

        clrscr();

        printf("Intialised elements \n");

        for(i=0;i<3;i++)
        {
                for(j=0;j<3;j++)
                {
                        printf("\n a[%d][%d]=%d",i,j,a[i][j]);
                }
        }
        getch();
}
#include<stdio.h>
#include<conio.h>
#include<process.h>
void main()
{
        int a[20][20],r,c,i,j;
        clrscr();
        printf("Enter number of rows and columns : ");
        scanf("%d%d",&r,&c);
        if(r>20 || c>20)
        {
                printf("Array index out of bounds ");
                getch();
                exit(0);
        }
        printf("Enter array elements \n");
        for(i=0;i<r;i++)
        {
                for(j=0;j<c;j++)
                {
                        printf("Enter value into a[%d][%d]=",i,j);
                        scanf("%d",&a[i][j]);
                }
        }
        printf("Array elements \n");
        for(i=0;i<r;i++)
        {
                for(j=0;j<c;j++)
                {
                  printf("\n Given value in a[%d][%d]=%d",i,j,a[i][j]);
                }
        }
        getch();
}

#include<stdio.h>
#include<conio.h>
void main()
{
        int a[20][20],r,c,i,j;
        clrscr();
        printf("Enter number of rows and columns : ");
        scanf("%d%d",&r,&c);
        printf("Enter array elements \n");
        for(i=0;i<r;i++)
        {
                for(j=0;j<c;j++)

                {

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

                }

        }

        printf("Array elements \n");

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

        {

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

                {

                   printf("%4d",a[i][j]);

                }

                printf("\n");

        }

        getch();

}

#include<stdio.h>

#include<conio.h>

void main()

{

        void accept(int [][20],int,int);

        void disp(int [][20],int,int);


        int a[20][20],r,c;

        clrscr();


        printf("Enter number of rows and columns : ");
        scanf("%d%d",&r,&c);
        printf("Enter array elements \n");
        accept(a,r,c);
        printf("Given elements \n");
        disp(a,r,c);
        getch();
}
void accept(int arr[][20],int m,int n)
{
        int i,j;
        for(i=0;i<m;i++)
        {
                for(j=0;j<n;j++)
                {
                        scanf("%d",&arr[i][j]);
                }
        }
}
void disp(int arr[][20],int m,int n)
{
        int i,j;
        for(i=0;i<m;i++)
        {
                for(j=0;j<n;j++)
                {
                        printf("%d\t",arr[i][j]);
                }
```

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

write a program to accept any matrix and display its tranpose matrix.

```
#include<stdio.h>
#include<conio.h>
void main()
{
          void accept(int [][20],int,int);
          void disp(int [][20],int,int);
          void trans(int [][20],int [][20],int,int);
          int a[20][20],at[20][20],r,c;
          clrscr();
          printf("Enter rows and columns :");
          scanf("%d%d",&r,&c);
          printf("Enter any matrix \n");
          accept(a,r,c);
          printf("Given matrix \n");
          disp(a,r,c);
          trans(a,at,r,c);
          printf("Transpose matrix \n");
          disp(at,c,r);
          getch();
}
void accept(int a[][20],int r,int c)
{
          int i,j;
          for(i=0;i<r;i++)
          {
                  for(j=0;j<c;j++)
                  {
                          scanf("%d",&a[i][j]);
                  }
          }
}
void disp(int a[][20],int r,int c)
{
          int i,j;
          for(i=0;i<r;i++)
          {
                  for(j=0;j<c;j++)
                  {
                          printf("%3d",a[i][j]);
                  }
                  printf("\n");
          }
}
void trans(int a[][20],int at[][20],int r,int c)
{
          int i,j;
          for(i=0;i<r;i++)
          {
                  for(j=0;j<c;j++)
                  {
                          at[j][i]=a[i][j];
                  }
          }
}
```

ÄÄ

write a program to display addition of two matrices using functions

```
#include<stdio.h>
#include<conio.h>
void main()
{
        void accept(int [][20],int,int);
        void disp(int [][20],int,int);
        void addition(int [][20],int [][20],int [][20],int,int);
        int a[20][20],b[20][20],add[20][20],r,c;
        clrscr();
        printf("Enter rows and columns :");
        scanf("%d%d",&r,&c);
        printf("Enter first matrix \n");
        accept(a,r,c);
        printf("Enter second matrix \n");
        accept(b,r,c);
        addition(a,b,add,r,c);
        printf("First matrix  \n");
        disp(a,r,c);
        printf("Second matrix \n");
        disp(b,r,c);
        printf("Addition of two matrices \n");

        disp(add,r,c);

        getch();

}

void accept(int arr[][20],int m,int n)

{

        int i,j;

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

        {

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

                {

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

                }

        }

}

void disp(int arr[][20],int m,int n)

{

        int i,j;

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

        {

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

                {

                        printf("%3d",arr[i][j]);

                }

                printf("\n");

        }

}
```

```
void addition(int a[][20],int b[][20],int add[][20],int m,int n)

{

        int i,j;

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

        {

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

                {

                        add[i][j]=a[i][j]+b[i][j];

                }

        }

}
```

ÄÄ

write a program to display multiplication of two matrices using functions

```
#include<stdio.h>

#include<conio.h>

#include<process.h>

void main()

{

        void accept(int [][20],int,int);

        void disp(int [][20],int,int);

        void mul(int [][20],int [][20],int [][20],int,int,int);

        int a[20][20],b[20][20],c[20][20],m,n,p,q;

        clrscr();

        printf("Enter rows and columns for first matrix:");

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

        printf("Enter rows and columns for second matrix:");

        scanf("%d%d",&p,&q);
        if(n!=p)
        {
                printf("Matrix multiplication is not possible ");
                getch();
                exit(0);
        }
        printf("Enter first matrix \n");
        accept(a,m,n);
        printf("Enter second matrix \n");
        accept(b,p,q);
        printf("First matrix  \n");
        disp(a,m,n);
        printf("Second matrix \n");
        disp(b,p,q);
        mul(a,b,c,m,n,q);
        printf("Multoplication of two matrices \n");
        disp(c,m,q);
        getch();
}
void accept(int arr[][20],int m,int n)
{
        int i,j;
        for(i=0;i<m;i++)
        {
                for(j=0;j<n;j++)
                {
                        scanf("%d",&arr[i][j]);
                }
        }
}
void disp(int arr[][20],int m,int n)
{
        int i,j;
        for(i=0;i<m;i++)
        {
                for(j=0;j<n;j++)
                {
                        printf("%3d",arr[i][j]);
                }
                printf("\n");
        }
}
void mul(int a[][20],int b[][20],int c[][20],int m,int n,int q)
{
         int i,j,k;
         for(i=0;i<m;i++)
         {
                for(j=0;j<q;j++)
                {
                        c[i][j]=0;
                        for(k=0;k<n;k++)
                        {
                                c[i][j]=c[i][j]+a[i][k]*b[k][j];
                        }
                        }
              }
}
```

ÄÄ

write a program to accept any matrix and check whether the given matrix is identity or not

```
#include<stdio.h>
#include<conio.h>
void main()
{
        void accept(int [][20],int,int);
        void disp(int [][20],int,int);
        int identity(int [][20],int,int);
        int a[20][20],r,c,k;
        clrscr();
        printf("Enter no of rows and columns :");
        scanf("%d%d",&r,&c);
        if(r!=c)
        {
                printf("Identity matrix is not possible ");
                getch();
                exit(0);
        }
        printf("Enter the matrix \n");
        accept(a,r,c);
        printf("Given matrix \n");
        disp(a,r,c);
        k=identity(a,r,c);
```

```
	if(k==1)
	  printf("Given matrix is identity matrix ");
	else
	   printf("Given matrix is not identity matrix ");
	getch();
}
void accept(int a[][20],int r,int c)
{
	int i,j;
	for(i=0;i<r;i++)
	{
		for(j=0;j<c;j++)
		{
			scanf("%d",&a[i][j]);
		}
	}
}
void disp(int a[][20],int r,int c)
{
	int i,j;
	for(i=0;i<r;i++)
	{
		for(j=0;j<c;j++)
		{
			printf("%4d",a[i][j]);
		}
		printf("\n");
```

```
        }
}
int identity(int a[][20],int r,int c)
{
        int i,j;

        for(i=0;i<r;i++)
        {
                for(j=0;j<c;j++)
                {
                        if(i==j)
                        {
                                if(a[i][j]!=1)
                                  return 0;
                        }
                        else
                        {     if(a[i][j]!=0)
                                  return 0;
                        }
                }
        }
        return 1;
}
```

Multi dimensional arrays

'c' supports arrays of 3 or more dimensions.In multidimensional arrays we use more than two index values.

The general form of Multidimensional array is

datatype arr_name[s1][s2]------[sn];

where s1,s2,--------sn are sizes.

eg

int a[2][2][3];

elements

a[0][0][0]	a[0][0][1]	a[0][0][2]
a[0][1][0]	a[0][1][1]	a[0][1][2]
a[1][0][0]	a[1][0][1]	a[1][0][2]
a[1][1][0]	a[1][1][1]	a[1][1][2]

```
#include<stdio.h>

#include<conio.h>

void main()

{

        int a[2][2][3],i,j,k;

        clrscr();

        printf("Enter array elements \n");

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

        {

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

                {

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

                        {
                                printf("Enter value into a[%d][%d][%d]=",i,j,k);
                                scanf("%d",&a[i][j][k]);
                              }
                    }
          }
          printf("Array elements \n");
          for(i=0;i<2;i++)
          {
            for(j=0;j<2;j++)
            {
              for(k=0;k<3;k++)
              {
               printf("\n Given value in a[%d][%d][%d]=%d",i,j,k,a[i][j][k]);
              }
            }
          }
          getch();
}
```

Strings

A group of characters defined between double quotation marks is a string. But in 'c' language a string is nothing but an array of characters and teminated by a null character(\0).

Declaration

char identifier[size];

eg:

char st[20];

Initialisation

char identifier[size]="string";

eg

char st[20]="welcome";

when the compiler assigns a character string to a character array,it automatically supplies a null character(\0) at the end of the string therefore the size should be equal to the maximum no of characters in the given string+1.

format specifier of string is %s.

```
#include<stdio.h>

#include<conio.h>

void main()

{

        char st[20]="welcome";

        clrscr();

        printf("Given string =%s",st);

        getche();

}
```

ÄÄ

gets

It gets a string from stdin.

declaration

char * gets(char *s);

puts

It outputs a string to stdout.

declaration

int puts(const char *s) (and appends a new line character)

```
#include<stdio.h>

#include<conio.h>
```

```
void main()

{

            char st[40];

            clrscr();

            puts("Enter any string :");

            gets(st);

            puts("Given string");

            puts(st);

            getch();

}
```

ÄÄ

getch

getch gets a character from keyboard but does not echo to the screen.

declaration

int getch();

getche

getche gets a character from keyboard and echoes to the screen.

Declaration

int getche();

```
#include<stdio.h>

#include<conio.h>

void main()

{

            char ch;

            clrscr();
```

```
        printf("Enter any character :");

        ch=getch();

        //ch=getche();

        printf("\nGiven charcter =%c",ch);

        getch();

}
```

ÄÄ

getchar

It is a macro that gets character from stdin.

declaration

int getchar(void);

putchar

putchar is a macro that outputs a character on stdout.

declaration

int putchar(int c);

```
#include<stdio.h>

#include<conio.h>

void main()

{

        char ch;

        clrscr();

        printf("Enter any character :");

        ch=getchar();

        printf("Given character :");

        putchar(ch);
        getch();
}
```

ÄÄ

String handling functions

1.strlen   <string.h>

It calculates length of the given string.

declaration

size_t strlen(const char *s);

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main()
{
        char st[40];
        int len;
        clrscr();
        printf("Enter any string :");
        gets(st);
        len=strlen(st);
        printf("Length of the given string =%d",len);
        getch();
}
```

ÄÄ

2.strcpy   <string.h>

It copies one string to another.

declaration

char *strcpy(char *dest, const char *src);

```
#include<stdio.h>

#include<conio.h>

#include<string.h>

void main()

{

	char st1[40],st2[40];

	clrscr();

	printf("Enter any string :");

	gets(st1);

	strcpy(st2,st1);

	printf("Given string    =%s",st1);

	printf("\nCopied string =%s",st2);

	getch();

}
```

ÄÄ

3.strncpy   <string.h>

It copies specifed number of characters from one string to another.

declaration

char *strncpy(char *dest, const char *src, size_t maxlen);

```
#include<stdio.h>

#include<conio.h>

#include<string.h>

void main()
```

```
{
        char st1[40],st2[40];
        int n;
        clrscr();
        printf("Enter any string :");
        gets(st1);
        printf("Enter specified number of characters to be copied :");
        scanf("%d",&n);
        strncpy(st2,st1,n);
        st2[n]='\0';
        printf("Given string    =%s",st1);
        printf("\nCopied string =%s",st2);
        getch();
}
```

ÄÄÄ

4.strrev   <string.h>

It reverses the given string.

declaration

char *strrev(char *s);

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main()
{
        char st[40];
        clrscr();
```

```
	printf("Enter any string :");
	gets(st);
	printf("Given string  =%s",st);
	strrev(st);
	printf("\nReverse string =%s",st);
	getch();
}
```

ÄÄÄ

5. strcat	<string.h>

It appends one string to another.

declaration

char *strcat(char *dest, const char *src);

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main()
{
	char st1[80],st2[40];
	clrscr();
	printf("Enter first string :");
	gets(st1);
	printf("Enter second string :");
	gets(st2);
	strcat(st1,st2);
	printf("Concatenation of two strings =%s",st1);
	getch();
}
```

ÄÄ

6. strupr   <string.h>

It converts lowercase letters (a to z) in the given string to

upper case (A to Z)

declaration

char * strupr(char *s);

7. strlwr   <string.h>

It converts uppercase letters (A to Z) in the given string to lower case (a to z)

declaration

char * strlwr(char *s);

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main()
{
        char st[80];
        clrscr();
        printf("Enter any string :");
        gets(st);
        printf("Given string =%s",st);
        strupr(st);
        printf("\nGiven string in upper case =%s",st);
        strlwr(st);
        printf("\nGiven string in lower case =%s",st);
        getch();
}
```

ÄÄ

8. strcmp (function)   <string.h>

It compares two strings with case sensitivity.

int strcmp(const char *s1, const char*s2);

9.stricmp (function)   <string.h>

It compares two strings without case sensitivity.

int stricmp(const char *s1, const char*s2);

10.strcmpi (macro)   <string.h>

It compares two strings without case sensitivity.

int strcmpi(const char *s1, const char*s2);

return value

These routines return an int value

```
i.e;  <  0  if  s1 <  s2
     == 0  if  s1 == s2
     >  0  if  s1  > s2
```

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main()
{
        char st1[80],st2[80];
        int k;
        clrscr();
        printf("Enter first string :");
        gets(st1);
        printf("Enter second string :");
        gets(st2);
          /* k=strcmp(st1,st2);
          k=strcmpi(st1,st2);   */
          k=stricmp(st1,st2);
        if(k==0)
          printf("Two strings are equal ");
        else if(k<0)
          printf("First string is less than second string ");
        else
          printf("First string is greater than second string ");
          getch();
}
ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main()
{
        char st1[80],st2[40];
        int k;
        clrscr();
        printf("Enter any string : ");
```

```
            gets(st1);

            strcpy(st2,st1);

            strrev(st2);

            k=strcmp(st1,st2);

            if(k==0)

                        printf("Given string is pallindrome");

            else

                        printf("Given string is not pallindrome");

            getch();

}


#include<stdio.h>

#include<conio.h>

#include<string.h>

void main()

{

            char st[80];

            int ns=0,nw=0,nc=0,na=0,nd=0,nsp=0,i;

            clrscr();

            printf("Enter any string :");

            gets(st);

            for(i=0;i<=strlen(st);i++)

            {

               if(st[i]==32)

               {

                        ns++;

                        nw++;
```

```
  }
  else if( (st[i]>=65 && st[i]<=90) || (st[i]>=97 && st[i]<=122) )
  {
        na++;
        nc++;
  }
  else if(st[i]>=48 && st[i]<=57 )
  {
        nc++;
        nd++;
  }
  else if(st[i]=='\0')
  {
        nw++;
  }
  else
  {
        nc++;
        nsp++;
  }
}

printf("No of spaces              =%d",ns);
printf("\nNo of words               =%d",nw);
printf("\nNo of characters          =%d",nc);
printf("\nNo of alphabets           =%d",na);
printf("\nNo of digits              =%d",nd);
```

```
            printf("\nNof of special characters   =%d",nsp);

            getch();

}

ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ

#include<stdio.h>

#include<conio.h>

#include<string.h>

void main()

{

           char st[80];

           static int a[256],i;

           clrscr();

           printf("Enter any string :");

           gets(st);

           for(i=0;i<strlen(st);i++)

           {

                      a[st[i]]=a[st[i]]+1;

           }

           printf("CHAR    FRE");

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

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

           {

                      if(a[i]!=0)

                      {

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

                      }

           }
```

```
	getch();
}
```

ÄÄÄ

Two dimensional character arrays

A list of names can be treated as table of strings and a two dimensional

character array can be used to store the entire list.

for eg

char st[5][20] may be used to store a list of names,each of length not more than 20 charactres.

```
#include<stdio.h>
#include<conio.h>
void main()
{
	char st[20][20];
	int n,i;
	clrscr();
	printf("Enter anumber of strings : ");
	scanf("%d",&n);
	printf("Enter any %d strings \n",n);
	fflush(stdin);
	for(i=0;i<n;i++)
	{
		gets(st[i]);
	}
	printf("Given strings \n");
	for(i=0;i<n;i++)
```

```
        {
                puts(st[i]);
        }
        getch();
}
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main()
{
        char st[20][20],temp[20];
        int n,i,j,k;
        clrscr();
        printf("Enter anumber of strings : ");
        scanf("%d",&n);
        printf("Enter any %d strings \n",n);
        fflush(stdin);
        for(i=0;i<n;i++)
        {
                gets(st[i]);
        }
        printf("Given strings Before sorting \n");
        for(i=0;i<n;i++)
        {
                puts(st[i]);
        }
        /* sorting */
        for(i=0;i<n-1;i++)

        {

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

                {

                        k=strcmp(st[i],st[j]);

                        if(k>0)

                        {

                                strcpy(temp,st[i]);

                                strcpy(st[i],st[j]);

                                strcpy(st[j],temp);

                        }


                }

        }

        printf("\n Given strings After sorting \n");

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

        {

                puts(st[i]);

        }

        getch();

}
```

POINTERS

C provides the important feature of data manipulations with the address of the variables, and hence the execution time is much reduced. Such concept is possible with the special datatypes called pointers.

Pointer

A pointer is a variable which can store the address of another variable.

Declaration

Pointer declaration is similar to normal declaration but preceded by asterik (*).

datatype *identifier;

eg

int *p;

Initialisation

datatype *identifer = address;

eg

int n;

int *p = &n;

Note

For any type of pointer, the total number of bytes allocated is always 2 bytes. Because the pointer variable stores address of the memory locations.

```
#include<stdio.h>

#include<conio.h>

void main()

{

        int *p1;

        char *p2;

        float *p3;

        double *p4;

        clrscr();

        textmode(2);

        printf("Size of integer pointer      =   %d bytes ",sizeof(p1));

        printf("\n Size of character pointer  =   %d bytes ",sizeof(p2));

        printf("\n Size of float pointer      =   %d bytes ",sizeof(p3));

        printf("\n Size of double pointer     =   %d bytes ",sizeof(p4));
        getch();
}
```

void *

It is a generic pointer refers the address of any type of variable and also it will assign to any type of pointer.

```
#include<stdio.h>
#include<conio.h>
void main()
{
        int n=100;
        int *p=&n;
        clrscr();
        textmode(2);
        printf("Value of n              =    %d",n);
        printf("\n Address of  n          =    %u",&n);
        printf("\n Value of n using p      =    %d",*p);
        printf("\n Address of n using p    =    %u",p);
        *p=500;       /*    n=500    */
        printf("\n Changed value of n        =    %d",n);
        getch();
}
```

Pointers and functions

call by value (pass by value)

The process of passing the value to the function call is known as call by value.

```
#include<stdio.h>

#include<conio.h>

void main()

{

        void func(int);

        clrscr();

        func(100);

        getch();

}

void func(int k)

{

        printf("k=%d",k);

}


#include<stdio.h>

#include<conio.h>

void main()

{

        void func(int,int);

        int a,b;

        clrscr();

        printf("Enter any two integer values : ");

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

        func(a,b);

        getch();

}

void func(int x,int y)
```

```
{
        printf("sum is    %d",x+y);
}
```

call by reference

The process of calling a function using pointers to pass the address of variables is kunown as call by reference.

```
#include<stdio.h>
#include<conio.h>
void main()
{
        void setdata(int *);
        int n;
        clrscr();
        setdata(&n);
        printf("n=%d",n);
        getch();
}
void setdata(int *k)
{
        *k=500;
}
```

```
#include<stdio.h>
#include<conio.h>
void main()
{
        void accept(int *);

        int n;

        clrscr();

        accept(&n);

        printf("n=%d",n);

        getch();

}

void accept(int *p)

{

        printf("Enter any integer value : ");

        scanf("%d",p);

}


#include<stdio.h>

#include<conio.h>

void main()

{

        void accept(int *,char *,float *);

        int eno;

        char ename[80];

        float sal;

        clrscr();

        accept(&eno,ename,&sal);

        printf("Emp number      =   %d",eno);

        printf("\n Emp name     =   %s",ename);

        printf("\n Emp salary   =   %.2f",sal);

        getch();
```

```
}

void accept(int *no,char *name,float *s)

{

        printf("Enter emp number : ");

        scanf("%d",no);

        printf("Enter emp name : ");

        fflush(stdin);

        gets(name);

        printf("Enter salary : ");

        scanf("%f",s);


}
```

Pointers and arrays

When an array is declared the compiler allocates the base address and sufficient amount of storage contain all the array continuous memory location.

The base address is the location of the address is the first element (index 0) of the array.

The compiler also defines the array name as a constant pointer to point to the first element.

suppose we declare an array 'a'(array name) as follows.

int a[5]={1,2,3,4,5};

suppose the base address of 'a' is 1000.and assuming the each integer requires 2 bytes,the five elements will be stored as follows. The name 'a' is defined as constant pointer to point to the first element a[0].

The value of 'a' is 1000,the location where a[0] is stored.

a=&a[0]=1000.

```
#include<stdio.h>

#include<conio.h>

void main()
```

```
{
        int a[5];
        clrscr();
        printf("Base address      =   %u",&a[0]);
        printf("\n Base address   =   %u",a);
        getch();
}
```

if we declared 'p' is an ineteger pointer,then we can make the pointer 'p' to point to the array 'a' by the following assignment.

```
int *p;
p=a;   (or)   p=&a[0];
```

Now we can access every value of 'a' using p++ (or) p+1 to move one element

to another.The relationship between 'p' and 'a' is shown below.

```
p+0 = &a[0] = 1000
p+1 = &a[1] = 1002
p+2 = &a[2] = 1004
p+3 = &a[3] = 1006
p+4 = &a[4] = 1008
```

when handling arrays instead of using array index,we can use pointers to access array elements.

Note that *(p+k) gives the value of a[k]. The pointer accessing method is much faster than array indexing.

```
#include<stdio.h>
#include<conio.h>
void main()
{
        int a[20],i,n,*p;
        clrscr();
        p=a;  /*   p=&a[0]    */

        printf("Enter array elements : ");

        scanf("%d",&n);

        printf("Enter array elements \n");

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

        {

                scanf("%d",p+i);

        }

        printf("Array elements \n");

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

        {

                printf("%4d",*(p+i));

        }

        getch();

}
```

Dynamic memory allocation:

The 'c' language requires the number of elements in an array to be soecified at compile time.But we may not be able to do always.

Our initial judgement of size,if it is wrong may cause failure of the program (or) wastage of memory space.

At this stage we use Dynamic memory allocation.

Definition

The process of allocating memory at runtime is known as Dynamic memory allocation.

malloc   <alloc.h>

It is used to allocating memory at run time.

syntax

```
void * malloc(size_t size);

pointer=(type cast)malloc(memory size);
```

eg

int *p;

p=(int *)malloc(sizeof(int));   1 location

p=(int *)malloc(5*sizeof(int)); 5 locations

calloc   <alloc.h>

syntax

void *calloc(size_t nitems, size_t size);

pointer=(typecast)calloc(no of items,size of each item);

eg

int *p;

p=(int *)calloc(1,sizeof(int));   1 location

p=(int *)calloc(5,sizeof(int)); 5 locations

Note

calloc allocates a block (nitems * size) bytes and clears it to 0.

free

It frees allocated blocks

declaration

void free(void *block);

```
#include<stdio.h>

#include<conio.h>

#include<alloc.h>

void main()

{

        int *a,n,i;
```

```
	clrscr();
	textmode(2);
	printf("Enter number of elements : ");
	scanf("%d",&n);
	a=(int *)malloc(n*sizeof(int));
	printf("Enter elements \n");
	for(i=0;i<n;i++)
	{
		scanf("%d",a+i);
	}
	printf("Given elements \n");
	for(i=0;i<n;i++)
	{
		printf("%d\t",*(a+i));
	}
	free(a);
	getch();
}
```

```
#include<stdio.h>
#include<conio.h>
#include<alloc.h>
void main()
{
	int **a,r,c,i,j;
	clrscr();
	textmode(2);
	printf("Enter number of rows and columns : ");
	scanf("%d%d",&r,&c);
	a=(int **)malloc(r*sizeof(int *));
	for(i=0;i<r;i++)
	{
		*(a+i)=(int *)malloc(c*sizeof(int));
	}
	printf("Enter elements \n");
	for(i=0;i<r;i++)
	{
		for(j=0;j<c;j++)
		{
			scanf("%d",*(a+i)+j);
		}
	}
	printf("Given elements \n");
	for(i=0;i<r;i++)
	{
		for(j=0;j<c;j++)
		{
			printf("%3d",*(*(a+i)+j));
		}
		printf("\n");
	}
	free(a);
	getch();
}
```

realloc

It reallocates main memory at run time.

syntax

```
void * realloc(void *block,size_t size);
```

```
int *p;

p=(int *)malloc(5*sizeof(int));

p=(int *)realloc(p,8*sizeof(int));
```

```
#include<stdio.h>

#include<conio.h>

#include<alloc.h>

void main()

{

        char *st;

        clrscr();

        st=(char *)malloc(10*sizeof(char));

        strcpy(st,"hello");

        printf("String is  =  %s",st);

        st=(char *)realloc(st,25*sizeof(char));

        strcat(st,"welcome to bdps");

        printf("\n New string is    =   %s",st);

        free(st);

        getch();

}
```

Preprocessor statements (or) preprocessor directives

Preprocessor is a program that process the source code before it process through the compiler.

The commands used to control the processor is known as pre-processor directives.

These directives are divided into 3 types.

1.file inclusion directives.

2.macro substitution directives.

3.compiler control directives.

1.File inclusion directives

It is used to include header files.It provides instructions to link the functions with the program from the system library.

```
#include "file name"

            (or)

#include <file name>
```

when the filename is included with in double quotation marks the search for the file is made first the current directory and then the standard directory when the file name is included with in angular braces the file search only in the standard directory.

2.Macro substitution directives

macro is a process where an identifier in a program replaced by a predefined string (or) value.

#define is a preprocessor statement and is used to define macros.

syntax

```
#define  identifier   predefined string (or) value
```

eg

```
#define  p   printf
```

```
#include<stdio.h>

#include<conio.h>

#define   p     printf

#define  cls    clrscr

#define  gt     getch

#define  vm     void main

vm()

{

            cls();

            p("Welcome");

            gt();

}
```

```
#include<stdio.h>

#include<conio.h>

#define   p     printf

#define   s     scanf

#define  cls    clrscr

#define  gt     getch

#define  vm     void main

#define  valz    100


vm()

{

            int n;

            cls();

            p("Enter any integer value : ");
```

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

        p("n=%d",n);

        p("\n value    =   %d",val);

        gt();

}
```

Macro with arguments

Syntax

```
#define      identifier(arg-1,arg-2,----arg-n)       definition
```

eg

```
#define     square(x)   x*x
```

```
#include<stdio.h>

#include<conio.h>

#define square(x)   x*x

/*

int  square(int x)

{

        return x*x;

}

*/

void main()

{

        int n,s;

        clrscr();

        textmode(2);

        printf("Enter any ineteger value : ");
```

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

	s=square(n);

	printf("Square value of given number   =   %d",s);

	getch();

}


#include<stdio.h>

#include<conio.h>

#define square(x)   x*x

void main()

{

	int n;

	clrscr();

	textmode(2);

	n=100/square(5);

	printf("n=%d",n);

	getch();

}
```

Function	Macro
1.It is a self contained block of statements	1.It is a preprocessor statement statements
2.Its replaces its return value.	2.It replaces its definition
3.We can use only specified datatypes only	3.Datatypes are generic.

4.Execution time is more.	4.Execution time is less.
5.It requires less memory.	5.It requires more memory.

```
#include<stdio.h>
#include<conio.h>
#define    maxval(x,y)      x>y ? x : y
void main()
{
        int a,b,max;
        clrscr();
        printf("Enter any two integer values : ");
        scanf("%d%d",&a,&b);
        max=maxval(a,b);
        printf("Maximum value   =  %d",max);
        getch();
}
```

3.compiler control directives

These directives used to control the compiler.

#if

#else

#elif (else if)

#endif

```
#include<stdio.h>
#include<conio.h>
#define n 10
```

```
#if  n<=10

            #define val 100

#else

            #define val 200

#endif

void main()

{

            clrscr();

            printf("value  =   %d",val);

            getch();

}
```

Structures

Definition

A group of data items of different datatypes stored in a continuous memory location is known as a structure.

Struct: It is a keyword and is used to declare a structure.

Declaration of a structure

Form1

```
            struct strut_name

            {

                        datatype item-1;

                        datatype item-2;

                        ---------------

                        ---------------

                        datatype item-n;
```

```
};

eg

--

        struct emp

        {

                int eno;

                char ename[20];

                float sal;

        };
```

Declaration of a structure variable

struct struct_name identifier;

eg

struct emp e;

.(Access operator)

It is used to access the data items in the structure with the help of structure variable.

eg

e.eno

e.ename

e.sal

int n=100;

Intialisation

struct struct_name identifier = {value-1,......value-n};

eg

struct emp e = {101, "abcd", 5000};

Form2

Declaration of a structure & structure variable in a single statement

```
struct struct_name
{
        datatype item-1;
        datatype item-2;
        ----------------
        ----------------
        datatype item-n;
}va_list;
```

eg

```
struct emp
{
        int eno;
        char ename[20];
        float sal;
}e;
```

Initialisation

eg

```
struct emp
{
        int eno;
        char ename[20];
        float sal;
}e={101, "RAJU", 50000};
```

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

```
{
        int eno;
        char ename[80];
        float sal;
}e={300,"xyz",10000.00};
void main()
{
        clrscr();
        printf("Emp number      =   %d",e.eno);
        printf("\n Emp name      =   %s",e.ename);
        printf("\n Emp salary    =   %.2f",e.sal);
        getch();
}


#include<stdio.h>
#include<conio.h>
struct emp
{
        int eno;
        char ename[80];
        float sal;
};
void main()
{
        struct emp e;
        clrscr();
        textmode(2);
```

```
	printf("Enter emp number : ");
	scanf("%d",&e.eno);
	printf("Enter emp name : ");
	fflush(stdin);
	gets(e.ename);
	printf("Enter salary : ");
	scanf("%f",&e.sal);
	printf("Emp number	=   %d",e.eno);
	printf("\n Emp name	=   %s",e.ename);
	printf("\n Emp salary	=   %.2f",e.sal);
	getch();
}

#include<stdio.h>
#include<conio.h>
struct student
{
	int sno,c,cpp,java,tot;
	char sname[80],res[20],div[20];
	float avg;
};
void main()
{	struct student s;
	clrscr();
	textmode(2);
	printf("Enter student number : ");
	scanf("%d",&s.sno);
```

```
printf("Enter student name : ");
fflush(stdin);
gets(s.sname);
printf("Enter marks in c cpp java : ");
scanf("%d%d%d",&s.c,&s.cpp,&s.java);
s.tot=s.c+s.cpp+s.java;
s.avg=(float)s.tot/3;
printf("Total marks     =    %d",s.tot);
printf("\nAverage      =    %.2f",s.avg);
if(s.c>=50 && s.cpp>=50 && s.java>=50)
{
        strcpy(s.res,"pass");
        printf("\n Result      =  %s",s.res);
        if(s.avg>=70)
        {
                strcpy(s.div,"first");
                printf("\n Division   =  %s",s.div);
        }
        else
        {
                strcpy(s.div,"second");
                printf("\n Division   =  %s",s.div);
        }
}
else
{
        strcpy(s.res,"fail");
```

```
                printf("\n Result     =   %s",s.res);

                strcpy(s.div,"NO_DIVISION");

                printf("\n Division   =   %s",s.div);

        }

        getch();

}

#include<stdio.h>

#include<conio.h>

struct item

{

        int ino;

        char iname[80];

        float cost;

};

void main()

{

        struct item it;

        clrscr();

        textmode(2);

        printf("Enter item number : ");

        scanf("%d",&it.ino);

        printf("Enter item name : ");

        fflush(stdin);

        gets(it.iname);


        printf("Enter cost : ");

        scanf("%f",&it.cost);
        printf("Item number      =   %d",it.ino);
        printf("\n Item name     =   %s",it.iname);
        printf("\n Item cost     =   %.2f",it.cost);
        getch();
}
```

Array of structures

Like any other datatype structure arrays can be defined.so that each array element can be of structure datatype.

For eg

```
struct item
{
        int ino;
        char iname[80];
        float cost;
};
```

struct item it[20] defines an array called it, that contains 20 elements each element is defined to be the struct item.

```
#include<stdio.h>
#include<conio.h>
struct item
{
        int ino;
        char iname[80];
        float cost;
};
void main()
```

```
{
        struct item it[20];
        int n,i;
        float *f,f1;
        clrscr();
        textmode(2);
        f=&f1;
        *f=f1;
        printf("Enter number of records : ");
        scanf("%d",&n);
        for(i=0;i<n;i++)
        {
                printf("Enter record   %d \n ",i+1);
                printf("Enter item number : ");
                scanf("%d",&it[i].ino);
                printf("Enter item name : ");
                fflush(stdin);
                gets(it[i].iname);
                printf("Enter cost : ");
                scanf("%f",&it[i].cost);
        }
        printf("%-10s%-15s%s","INO","INAME","COST");
        printf("\n------------------------------");
        for(i=0;i<n;i++)
        {
          printf("\n %-10d%-15s%.2f",it[i].ino,it[i].iname,it[i].cost);
        }
        getch();
}
```

Structures and pointers

A pointer can also point to a structure.

```
struct item
{
        int ino;
        char iname[80];
        float cost;
};
struct item it;
struct item *p;
 p = &it;
```

After making such an assignment, we can access every dataitem of "item"

structure indirectly through "p" as follows:

```
                (*p).ino
                (*p).iname
                (*p).cost
```

Here instead of writing "." and "*", we use the structure pointer

operator "arrow -> " which is a minus (" - ")followed by

greater than sign (">"). This means we can access every dataitem of

"item" structure directly through "p" as follows:

```
                p -> ino
                p -> iname

                p -> cost


#include<stdio.h>

#include<conio.h>

struct item

{

        int ino;

        char iname[80];

        float cost;

};

void main()

{

        struct item it,*p;

        float *f,f1;

        clrscr();

        f=&f1;

        *f=f1;

        p=&it;

        /*      indirect method

        printf("Enter item number : ");

        scanf("%d",&(*p).ino);

        printf("Enter item name : ");

        fflush(stdin);

        gets((*p).iname);

        printf("Enter cost : ");

        scanf("%f",&(*p).cost);
```

```
	printf("Item number     =    %d",(*p).ino);

	printf("\n Item name     =    %s",(*p).iname);

	printf("\n Item cost     =    %.2f",(*p).cost);     */


	/*         Direct method    */

	printf("Enter item number : ");

	scanf("%d",&p->ino);

	printf("Enter item name : ");

	fflush(stdin);

	gets(p->iname);

	printf("Enter cost : ");

	scanf("%f",&p->cost);

	printf("Item number     =    %d",p->ino);

	printf("\n Item name     =    %s",p->iname);

	printf("\n Item cost     =    %.2f",p->cost);

	getch();

}


#include<stdio.h>

#include<conio.h>

struct item

{

	int ino;

	char iname[80];

	float cost;

};

void main()
```

```
{
        struct item *it;
        float *f,f1;
        clrscr();
        textmode(2);
        f=&f1;
        *f=f1;
        it=(struct item *)malloc(sizeof(struct item));
        printf("Enter item number : ");
        scanf("%d",&it->ino);
        printf("Enter item name : ");
        fflush(stdin);
        gets(it->iname);
        printf("Enter cost : ");
        scanf("%f",&it->cost);
        printf("Item number      =   %d",it->ino);
        printf("\n Item name      =   %s",it->iname);
        printf("\n Item cost     =   %.2f",it->cost);
        getch();
}

#include<stdio.h>
#include<conio.h>
struct item
{
        int ino;
        char iname[80];
```

```
	float cost;
};
void main()
{
	struct item *it;
	int n,i;
	float *f,f1;
	clrscr();
	textmode(2);
	f=&f1;
	*f=f1;
	printf("Enter number of records : ");
	scanf("%d",&n);
	it=(struct item *)malloc(n*sizeof(struct item));
	for(i=0;i<n;i++)
	{
		printf("Enter record   %d \n ",i+1);
		printf("Enter item number : ");
		scanf("%d",&(it+i)->ino);
		printf("Enter item name : ");
		fflush(stdin);
		gets( (it+i)->iname);
		printf("Enter cost : ");
		scanf("%f",&(it+i)->cost);
	}
	printf("%-10s%-15s%s","INO","INAME","COST");
	printf("\n-------------------------------");
	for(i=0;i<n;i++)
	{
	 printf("\n %-10d%-15s%.2f",(it+i)->ino,(it+i)->iname,(it+i)->cost);
	}
	getch();
}
```

Functions and Structures

Passing a structure as a function argument (or) parameter.

Like any other datatype, a structure may be used as function argument.

Returning a structure

A function can not only receive a structure as its arguments but also can return them.

```
#include<stdio.h>
#include<conio.h>
struct emp
{
	int eno;
	char ename[80];
	float sal;
};
struct emp accept()
{
	struct emp e;
	printf("Enter emp number : ");
	scanf("%d",&e.eno);
	printf("Enter emp name : ");
```

```
        fflush(stdin);

        gets(e.ename);

        printf("Enter salary : ");

        scanf("%f",&e.sal);

        return e;

}

void disp(struct emp e)

{

        printf("Emp number       =   %d",e.eno);

        printf("\n Emp name      =   %s",e.ename);

        printf("\n Emp salary    =   %.2f",e.sal);

}

void main()

{

        struct emp e;

        clrscr();

        textmode(2);

        e=accept();

        disp(e);

        getch();

}
```

Nested structures (or) structure with in structure

Form 1

Declaring a structure with in another structure is known as Nested structure.

```
struct student                      Accessing dataitems
{                                   -------------------
        int sno,c,cpp,java;             s.sno
        char sname[80];                 s.sname
        struct result                   s.c
        {                               s.cpp
                int tot;                s.java
                float avg;              s.r.tot
                char res[20],div[20];   s.r.avg
        }r;                             s.r.res
}s;                                     s.r.div
```

Form 2

Declaring structure variable with in another structure is known as nested structure.

```
                                        Accessing dataitems
struct student                      -------------------
{                               r.s.sno
        int sno,c,cpp,java;         r.s.sname
        char sname[80];             r.s.c
};                              r.s.cpp
struct result                   r.s.java
{                               r.tot
        struct student s;           r.avg
        int tot;                    r.res
        float avg;              r.div

        char res[20],div[20];

}r;


#include<stdio.h>

#include<conio.h>

struct student

{

        int sno,c,cpp,java;

        char sname[80];

        struct result

        {

                int tot;

                char res[20],div[20];

                float avg;

        }r;

}s;

void main()

{

        struct student s;

        clrscr();

        printf("Enter student number : ");

        scanf("%d",&s.sno);

        printf("Enter student name : ");

        fflush(stdin);

        gets(s.sname);

        printf("Enter marks in c cpp java : ");
```

```
scanf("%d%d%d",&s.c,&s.cpp,&s.java);
s.r.tot=s.c+s.cpp+s.java;
s.r.avg=(float)s.r.tot/3;
printf("Total marks    =    %d",s.r.tot);
printf("\nAverage      =    %.2f",s.r.avg);
if(s.c>=50 && s.cpp>=50 && s.java>=50)
{
        strcpy(s.r.res,"pass");
        printf("\n Result     =  %s",s.r.res);
        if(s.r.avg>=70)
        {
                strcpy(s.r.div,"first");
                printf("\n Division   =  %s",s.r.div);
        }
        else
        {
                strcpy(s.r.div,"second");
                printf("\n Division   =  %s",s.r.div);
        }
}
else
{
        strcpy(s.r.res,"fail");
        printf("\n Result     =   %s",s.r.res);
        strcpy(s.r.div,"NO_DIVISION");
        printf("\n Division   =   %s",s.r.div);
}
```

```
        getch();
}
```

unions

Wnion is similar to struct,except it allows you define variables that share storage space.

syntax

```
union union_name
{
        datatype item-1;
        datatype item-2;
        ----------------
        ----------------
        datatype item-n;
}va_list;
```

Differences between struct and union

----

struct ------	union -----
1.A group of dataitems that belongs to different datatypes.	1.It is also same as structure.but only major difference is memory allocation.
2.It allocates the memory of all declared dataitems in it.	2.It allocates memory of biggest dataitem in it.
3.We can access all dataitems at a time.	3.We can access only one dataitem at a time.

³

4.Each and every dataitem has its ³4.All dataitems are shared common storage

own storage space. ³space.

³

```
#include<stdio.h>
#include<conio.h>
union emp
{
        int eno;
        char ename[80];
        float sal;
};
void main()
{
        union emp e;
        clrscr();
        printf("Enter emp number : ");
        scanf("%d",&e.eno);
        printf("Emp number      =   %d",e.eno);
        printf("\nEnter emp name : ");
        fflush(stdin);
        gets(e.ename);
        printf("Emp name        =   %s",e.ename);
        printf("\nEnter salary : ");
        scanf("%f",&e.sal);
        printf("Emp salary      =   %.2f",e.sal);
```

```
	getch();
}
```

FILES

C language permits the usage of limited input and output functions to read and write data. These functions are used

for only smaller volumes of data and it becomes difficult to handle large data volumes. Also the entire data is lost when the program is over.

To overcome these difficulties, a flexible method was developed by employing the concept of FILES - to store, to read or to write data and to return them even when the program is running.

Definition of file

A file is one, which enable the user to read, write and store a group of related data.

(or)

A file is a collection of related data stored in particular area on the disk.

'C' supports a number of functions to have the ability to perform the basic file operations which include the following

Naming a file

Opening a file

Reading data from the file

Writing data to the file

Closing the file

FILE: It is a predefined structure and is used to declare a file pointer.

Using this pointer, we can perform all file operations.

Declaration:

```
		FILE *identifier;
```

eg: FILE *fp;

fopen():

It opens a file stream.

Declaration:

```
FILE *fopen(const char *filename, const char *mode);
```

filename - specifies the file to be opened.

mode string - used to acces a file in different modes

String	Description
r	Opens the file for reading purpose only
w	Creates a new file for writing purpose
	If a file by that name already exists, it will be overwritten
a	Open for writing at the end of the file (or)
	Creates a new file for writing if the file does not exist
r+	Open an existing file for reading and writing purpose
w+	Creates a new file for reading and writing purpose
	If the file by that name already exists, it will be overwritten
a+	open for updation at the end of file (or)
	Create a new one if the file donot exist

To specify that a given file is being opened or created in textmode, append 't' to the string (rt, at, r+t, w+t etc.)

To specify binary file, append 'b' to the string

(wb, ab, r+b, a+b etc.)

----

fclose()

It closes a file stream.

Declaration:

```
int fclose(FILE *stream);
```

----

fcloseall()

It closes all opened streams.

Declaration:

```
int fcloseall(void);
```

getc:

getc is a macro that gets one character from a file stream

Declaration:

```
int getc(FILE *stream);
```

putc:

putc is a macro that outputs a character to a file stream

Declaration:

```
int putc(int c, FILE *stream);
```

fgetc():

It is a function which gets a character from a file stream.

Declaration:

```
int fgetc(FILE *stream);
```

fputc():

fputc outputs a character to a file stream

Declaration:

```
int fputc(int c, FILE *stream);
```

EOF (ASCII Value - 26)

A constant indicating that end of the file has been reached. To get this character from keyboard press "ctr+z"

```
#include<stdio.h>
#include<conio.h>
void main()
{
	FILE *fp;
	char ch;
	clrscr();
	textmode(2);
	printf("Enter text data into file (ctrl + z ) to stop \n");
	fp=fopen("a.txt","w");
	ch=getchar();
	while(ch!=EOF)
	{
		putc(ch,fp);
		ch=getchar();
	}
	fclose(fp);
	printf("Data stored succesfully");
	getch();
}


#include<stdio.h>
#include<conio.h>
void main()
{
	FILE *fp;
	char ch;
	clrscr();
	textmode(2);
	printf("Enter file name to display : ");
	gets(fname);
	fp=fopen("a.txt","r");
	printf("Reading data from file \n");
	ch=getc(fp);
	while(ch!=EOF)
	{
		printf("%c",ch);
		delay(10);
		ch=getc(fp);
	}
	fclose(fp);
	getch();
}
```

fgets

fgets gets a string from a stream.

Declaration

char *fgets(char *s, int n, FILE *stream);

Remarks

fgets reads characters from stream into the string s. It stops when it reads either n - 1 characters or a newline character, whichever comes first.

fputs

fputs outputs a string to a stream

declaration

int fputs(const char *s, FILE *stream);

Remarks

fputs copies the null-terminated string s to the given output stream. It does not append a newline character, and the terminating null character is not copied.

feof

Macro that tests if end-of-file has been reached on a stream.

Declaration

int feof(FILE *stream);

reurn value

returns non-zero if an end of file indicator was detected

```
#include<stdio.h>

#include<conio.h>

void main()

{
        FILE *fp;

        char st[80];

        clrscr();

        fp=fopen("str.txt","w");

        printf("Enter string data into file ( 'end' to stop ) \n");

        gets(st);

        while(strcmp(st,"end")!=0)

        {

                fputs(st,fp);

                fputs("\n",fp);

                gets(st);

        }

        fclose(fp);

        printf("Data stored successfully");

        getch();

}


#include<stdio.h>

#include<conio.h>

void main()

{

        FILE *fp;

        char st[80];

        clrscr();

        textmode(2);

        fp=fopen("str.txt","r");

        printf("Reading string data from file \n");
        fgets(st,80,fp);
        while(feof(fp)==0)
        {
                printf("%s",st);
                fgets(st,80,fp);
                delay(10);
        }
        fclose(fp);
        getch();
}
```

rewind

Repositions file pointer to stream's beginning.

Declaration

void rewind(FILE *stream);

```
#include<stdio.h>
#include<conio.h>
#include<dos.h>
void main()
{
        FILE *fp;
        char ch;
        clrscr();
        textmode(2);
        printf("Enter text data into file (ctrl + z ) to stop \n");
        fp=fopen("a.txt","a+");
        ch=getchar();
        while(ch!=EOF)
        {
                putc(ch,fp);
                ch=getchar();
        }
        printf("Data stored succesfully");
        printf("Reading data from file \n");
        rewind(fp);
        ch=getc(fp);
        while(ch!=EOF)
        {
                printf("%c",ch);
                delay(10);
                ch=getc(fp);
        }
        fclose(fp);
        getch();
}
```

fprintf

declaration

```
int fprintf (FILE *stream, const char *format [, argument, ...]);
```

fscanf

declaration

```
int fscanf (FILE *stream,      const char *format [, address, ...]);
```

Note

In fprintf (or) fscanf between format specifiers atleast one space is necessary.

```
#include<stdio.h>

#include<conio.h>

void main()

{

	FILE *fp;

	int eno;

	char ename[80];

	float sal;

	clrscr();

	fp=fopen("emp.txt","w");

	printf("Enter emp number : ");

	scanf("%d",&eno);

	printf("Enter emp name : ");

	fflush(stdin);

	gets(ename);

	printf("Enter salary : ");

	scanf("%f",&sal);

	fprintf(fp,"%d %s %f",eno,ename,sal);

	fclose(fp);

	printf("Record stored successfully");

	getch();

}
```

```
#include<stdio.h>

#include<conio.h>

void main()

{

        FILE *fp;

        int eno;

        char ename[80];

        float sal;

        clrscr();

        fp=fopen("emp.txt","r");

        printf("Reading data from file \n");

        fscanf(fp,"%d %s %f",&eno,ename,&sal);

        printf("Emp number      =   %d",eno);

        printf("\n Emp name       =   %s",ename);

        printf("\n Emp salary    =   %.2f",sal);

        fclose(fp);

        getch();

}


#include<stdio.h>

#include<conio.h>

void main()

{

        FILE *fp;

        int eno;

        char ename[80],ch;

        float sal;
```

```
clrscr();
textmode(2);
fp=fopen("emp.txt","a+");
do
{
        printf("Enter emp number : ");
        scanf("%d",&eno);
        printf("Enter emp name : ");
        fflush(stdin);
        gets(ename);
        printf("Enter salary : ");
        scanf("%f",&sal);
        fprintf(fp,"%d %s %f\n",eno,ename,sal);
        printf("Record stored successfully \n");
        printf("Do you want to continue (y/n ) : ");
        fflush(stdin);
        scanf("%c",&ch);
}while(ch!='n');
printf("%-10s%-15s%s","ENO","ENAME","SALARY");
printf("\n----------------------------------");
rewind(fp);
while(feof(fp)==0)
{
        fscanf(fp,"%d %s %f\n",&eno,ename,&sal);
        printf("\n %-10d%-15s%.2f",eno,ename,sal);
}
fclose(fp);
```

```
        getch();
}
```

ftell

Returns the current file pointer position.

Declaration

long ftell(FILE *stream);

fseek

Repositions the file pointer of a stream

Declaration

int fseek(FILE *stream, long offset, int whence);

offset

Difference in bytes between offset and whence.

whence

SEEK_SET	0	seeks from begining of file
SEEK_CUR	1	seeks from current position
SEEK_END	2	seeks from end of file

```
#include<stdio.h>
#include<conio.h>
void main()
{
        FILE *fp;
        char ch;
        clrscr();
        textmode(2);
        fp=fopen("a.txt","w");

        printf("Current file pointer position  = %ld",ftell (fp) );

        printf("\n Enter text data into file  \n");

        ch=getchar();

        while(ch!=EOF)

        {

                putc(ch,fp);

                ch=getchar();

        }

        printf("Present file pointer position    =   %ld",ftell(fp) );

        fseek(fp,-2,1);

        printf("\n New file pointer position     =   %ld",ftell(fp));

        fclose(fp);

        getch();

}
```

Binary files

putw

It outputs an integer on a stream.

declaration

int putw(int w,FILE *stream);

getw

getw gets an integer from stream.

declaration

int getw(FILE *stream);

```
#include<stdio.h>

#include<conio.h>

void main()

{

        FILE *fp;

        int n;

        clrscr();

        textmode(2);

        fp=fopen("num.dat","a+b");

        printf("Enter number data into file ( 0 to stop ) \n");

        scanf("%d",&n);

        while(n!=0)

        {

                putw(n,fp);

                scanf("%d",&n);

        }

        printf("Data stored successfully");

        rewind(fp);

        printf("\n Number data from file \n");

        n=getw(fp);

        while(n!=EOF)

        {

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

                n=getw(fp);

        }

        fclose(fp);

        getch();
```

}

fwrite

It appends specified number of equal size dataitems to an output file.

Declaration

size_t fwrite(const void *ptr, size_t size, size_t n, FILE*stream)

fread

It reads a specified number of equal sized dataitems from an input stream into a block.

Declaration

size_t fread(void *ptr, size_t size, size_t n, FILE *stream);

Argument What It Is/Does

ÍÍÍÍÍÍÍØÍÍÍ

ptr ³ Points to a block into which data is read

size ³ Length of each item read, in bytes

n ³ Number of items read

stream ³ Points to input stream

```
#include<stdio.h>

#include<conio.h>

struct emp

{

            int eno;

            char ename[80];

            float sal;

};

void main()
```

```
{
        struct emp e;
        FILE *fp;
        clrscr();
        fp=fopen("emp.dat","wb");
        printf("Enter emp number : ");
        scanf("%d",&e.eno);
        printf("Enter emp name : ");
        fflush(stdin);
        gets(e.ename);
        printf("Enter salary : ");
        scanf("%f",&e.sal);
        fwrite(&e,sizeof(e),1,fp);
        fclose(fp);
        printf("Record stored Succesfully");
        getch();
}


#include<stdio.h>
#include<conio.h>
struct emp
{
        int eno;
        char ename[80];
        float sal;
};
void main()
```

```
{
	struct emp e;
	FILE *fp;
	clrscr();
	fp=fopen("emp.dat","rb");
	fread(&e,sizeof(e),1,fp);
	printf("Emp number        =    %d",e.eno);
	printf("\n Emp name        =    %s",e.ename);
	printf("\n Emp salary      =    %.2f",e.sal);
	fclose(fp);
	getch();
}
#include<stdio.h>
#include<conio.h>
struct emp
{
	int eno;
	char ename[80];
	float sal;
};
void main()
{
	struct emp e;
	FILE *fp;
	char ch;
	clrscr();
	fp=fopen("emp.dat","a+b");
	do
	{
		printf("Enter emp number : ");
		scanf("%d",&e.eno);
		printf("Enter emp name : ");
		fflush(stdin);
		gets(e.ename);
		printf("Enter salary : ");
		scanf("%f",&e.sal);
		fwrite(&e,sizeof(e),1,fp);
		printf("Record Stored successflluy \n");
		printf("Do u want to continue (y/n) : ");
		fflush(stdin);
		scanf("%c",&ch);
	}while(ch!='n');
	rewind(fp);
	printf("%-10s%-15s%s","ENO","ENAME","SALARY");
	printf("\n----------------------------------");
	fread(&e,sizeof(e),1,fp);
	while(!feof(fp))
	{
		printf("\n%-10d%-15s%.2f",e.eno,e.ename,e.sal);
		fread(&e,sizeof(e),1,fp);
	}
	fclose(fp);
	getch();
}
```

Command line arguments

It is parameter supplied to a program when the program is invocked.This parameter may represent a filename,the program should process.command line arguments are typed by the user.The first argument is always the filename. we know that every 'c' program should have one main function and it can take arguments like other functions.If you want to work with command line arguments the main function can take 2 arguments called argc,argv.and the information obtained in the command line is processed on to the program through these arguments. The variable argc is an argument counter that counts the number of arguments on the command line.The argv is an argument vector and represents an array of character pointers that points to the command line arguments.the size of this array will be equal to the value of argc.

```
#include<stdio.h>

#include<conio.h>

void main(int argc,char *argv[])

{

        int i;

        printf("Number of arguments  =  %d",argc);

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

        {

                printf("\n argv[%d]= %s",i,argv[i]);

        }

}
```