What is computer?

Computer is an advanced electronics device which depends upon two things that is **hardware** and **software** without these two things computer cannot be exist. Computer device takes raw **data** as input from the user and processes these data under the control of set of instructions (called program) and gives the result (output) and saves output for the future use. The word "computer" comes from the word "compute", which means, "to calculate".

A computer is often referred to as a data processor because it can store, process and retrieve data whenever desired.

Data Input Computer Output Information

What is Hardware?

Hardware is a physical device of computer that performs an operation according to the given instruction by software.

Example:

Motherboard, Ram, Processor etc.

What is software?

Software is a collection of program where a program contains several instruction of any specific task.

Example:

Operating System, MS-office, School Management Software etc.

Software is basically categorized into two parts:-

System Software: System software is a set of one or more programs, which controls the operation and extends the processing capability of a computer system.

Example:

Operating system, Compiler, Assembler, Interpreter etc.

Application Software: Application software is designed to help the operator in doing certain type of work. Application software products are designed to satisfy a particular need of a particular environment.

Notes: All Software which runs on operating system known as application software

What is operating System?

An operating system is system software that is used for following task.

i) An operating system is responsible to connect all physical devices logically that immense it decided role of all devices.
ii) An operating system provides an interface between user and hardware. So, a user can interact to hardware easily.
iii) An operating system provides a platform for all application software.

Categorization of programming language:

1. **Low level language**

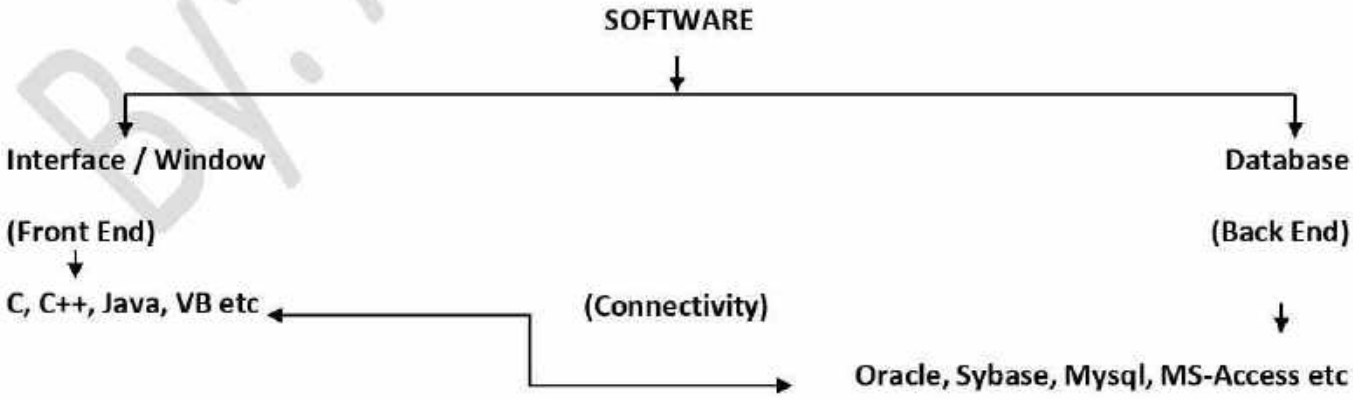
   Example: Assembly (System Software).

2. **Middle level language**

   Example: C, C++, Java etc (Application Software, System Software).

3. **High level language**

   Example: Fortron, Basic, VB, VB.Net etc (Application Software).



INTRODUCTION TO C

In 1972 Denis Ritchie introduce a compiler based language named "Defacto Standard To C" which was influenced to B and BCPL (Basic Combined Programming Language). Where B- language developed by Ken Thomason and BCPL language developed by Richards Martin.

It gains slowly popular in America So, American government assigned a task to check quality of "Defacto Standard For C". To do this requirement American government established a committee named ANSI (American National Standard Institute) in 1983.

ANSI Committee took 6years to check quality of "Defacto Standard For C". So, second version released in 1989 with name C-89, ANSI-C etc in which all drawback remove from "Defacto Standard For C" and the latest version is C-99.

Whole process of development done in AT & T Bell laboratories in USA.

C – Language is a middle level language because it holds/posses some properties of low level language as well as high level language. An application of C – language can be used for develop application software as well as system software but an application of c – language widely used for develop system software such as operating system, Driver software or game etc.

The C- language closely associated with operating system UNIX.

A first application of C- Language:

- **Source code of C –**

  A program is said to be source code of C- Language when it contains following properties.

i) **A file extension must be .C.**

ii) **A reserved word/keyword written in file must be supported by C- Compiler.**

Notes:
An application of C- language can be written in any text editor such as notepad, WordPad, DOS etc.

Example:

Ques: Write a program in 'C' to display a message.

"Welcome To C"

Soln:

Exp.c

```
#include<stdio.h>
#include<conio.h>
Void main ()
{
  Clrscr();
 Printf("Welcome To C");
 Getch();
}
```

OR

exp.c

```
Void main ()
{
  Clrscr();
 Printf("Welcome To C");
 Getch();
}
```

Point to be noted down:

1. In c language a header file including is optional.
2. A header file is a collection of object code of similar type inbuilt function.
3. # include <stdio.h>
   #include <conio.h>
   - **i)** **#:** - is a preprocessor which ensure that statement must be compile and execute before other statement.
   - **ii)** **Include:** - is a directives which is responsible to copy object code of header file into host application.
   - **iii)** **Stdio.h:-**
     Std -> standard

Io -> Input/output

.h -> extension indicates header file.

iv) **Conio.h:-**

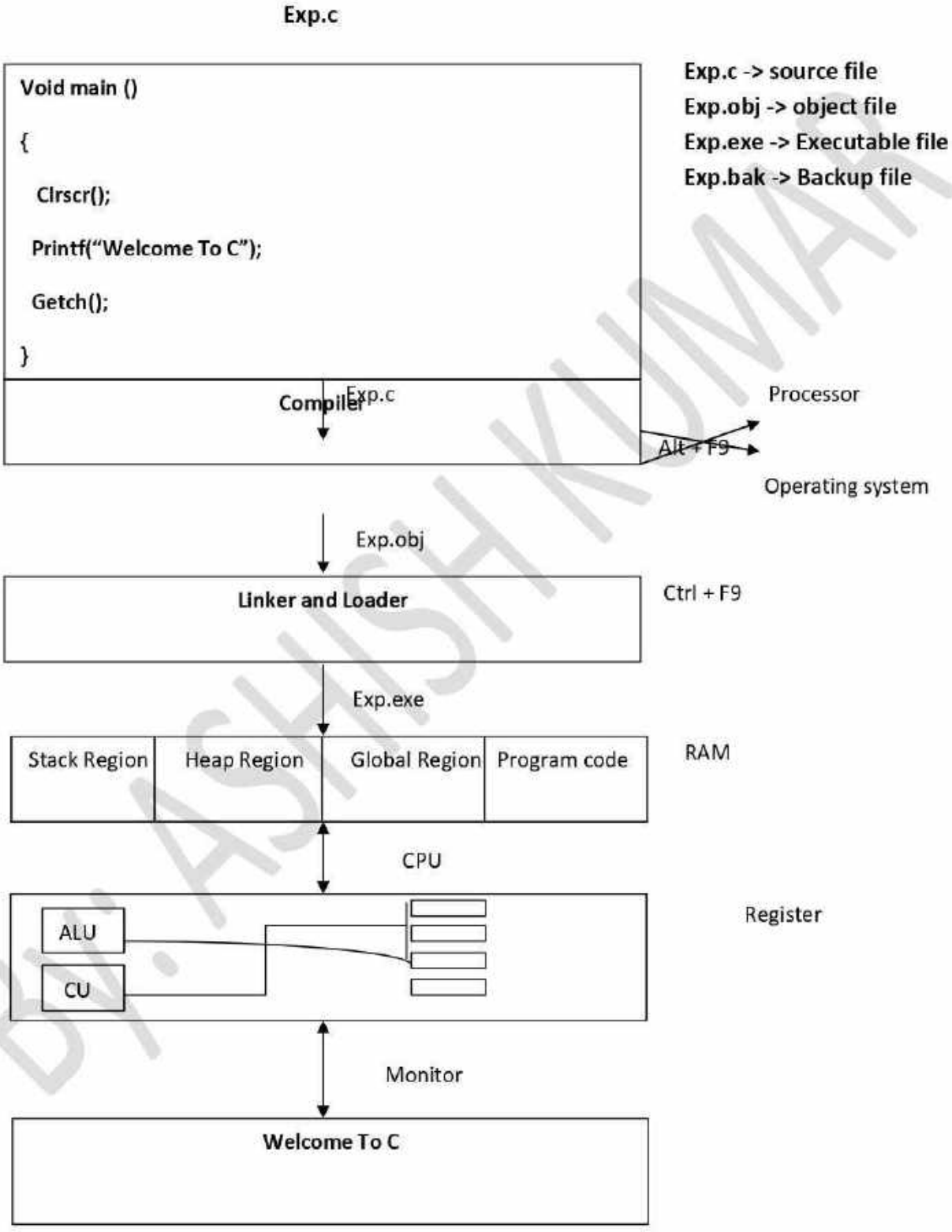
Con -> console

Io -> Input/output

.h -> extension indicates header file

4. In C- Language statements must be enclosed within a function. A function categorized into two types these are:-
   i) **User defined function:-** A function whose definition define by an user is known as user defined function.
   Example: main(), accept(), disp() etc.
   ii) **Inbuilt/ system defined/pre-defined function:** A function whose definition already defined by C compiler is known as Inbuilt/ system defined/pre-defined function.
   Example: clrscr(), printf(), getch() etc.
5. A source file must be contains at least one user defined function named main() because in C-language always execution starts from main() function.
   A main() function automatically called by operating system when we press ctrl + f9.
6. **Void:** is a datatype whose insure that a function is not going to return any value its caller.
7. **Open ({)** curlie braces ensure starting position of function and **close (})** curlie braces ensure termination of definition of function.
8. **Clrscr():-** is an inbuilt function of conio.h that is responsible to remove/clear statement from output buffer.
9. **Printf():-** is an inbuilt function of stdio.h that is responsible to display parenthesized statement on monitor.
10. **Getch():-** is an inbuilt function of conio.h that is responsible to accept a character from input buffer.
11. A statement must be terminates by semicolon (;).
12. All reserved word/keyword must be written into small letter because C is case sensitive language.

 # Compiling, Linking and Executing C Application:



 # Some points regarding application

1. In c language a statement must be terminated by semicolon (;).
2. In c language more than one statement can be separated into single line by using comma operator.

Explanation:

1st method

```
Void main ()
{
  Clrscr();
 Printf("Welcome To C");
 Getch();
}
```

2nd method

```
Void main ()
{
  Clrscr(), Printf("Welcome To C"),  Getch();
}
```

3rd method

```
Void main (){ Clrscr(), Printf("Welcome To C"), Getch();}
```

3. A variable statement must be first statement just after open ({) curlie braces otherwise leads to compile time error.

Explanation:

```
Void main ()
{
 Clrscr();
Int a=20;
  Printf("value of a=%d",a);
  Getch();
}
```

Output :

Error – Declaration is not allowed here.

```
Void main ()
{
 Int a=20;
Clrscr();
  Printf("value of a=%d",a);
  Getch();
}
```

Output: value of a=20

- **About Comment:**

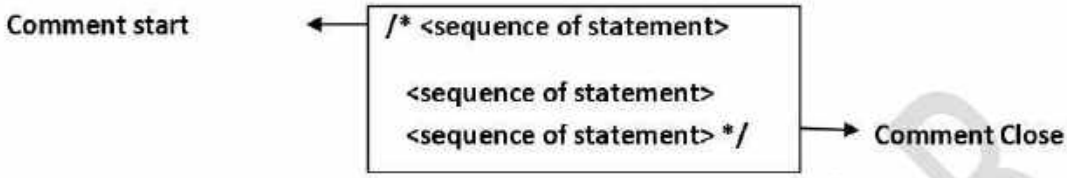
4. **A comment is a statement of application that appear but it does not compiled and execute.**
   A comment line may be two types:
   i) Single line comment
   ii) Multiple line comment

 # Single comment: when it is required to comment single line then statement must be preceded by double forward slash (//) as following format.

Syntax: `// <sequence of statements` → **Commented line**

 # Multiple line comment: when it is required to comment more than one line then you must follow the following structure given below.

Syntax:



Example:

/* write a program in 'c' to assign a value at compilation time and display it on monitor with an appropriate message. */

```
Void main()
{
Int a=5;
Clrscr();
Printf("Value of a=%d",a);
Getch();
}
```

What is error in 'C' Language?

A problem raised at compile time as well as run time due to any syntax or grammatical mistakes are known as error.

An error is categorized into following three types these are:

- Syntactical error
- Grammatical error
- Logical error

Syntactical error: An error raised at compile time due to missing any syntax is known as syntactical error.

Example:

```
Void main()
{
Clrscr();
Printf("Welcome to c")
Getch();
}
```

Error: Semicolon expected (;).

Grammatical error: An error raised at run time when we write an inbuilt function wrong.

Example:

```
Void main()
{
Clrsrc();
Printf("welcome to c");
Getch();
}
```

Error: undefined symbol clrsrc() in module exp.c.

Logical Error: when an application does not produce a correct output due to only logical mistake then it is said to be as logical error.

Example:

```
Void main()
{
Int basic=50,da;
Clrscr();
Da=basic/100*10; //->output will be always zero, because calculation is wrong.
                 //->correct way da=basic*10/100;
Printf("da=%d",da);
Getch();
}
```

Exception: - **An** abnormal condition arises during run-time is known as exception.

Example:

```
Void main()
{
Int a,b;
Clrscr();
Printf("Enter two no::");
Scanf("%d %d",&a,&b);
C=a/b;
Printf("c=%d",c);
Getch()
}
```

Output:

Enter two no:: 5

0

Divide error //Exception manage

Warning:- A warning is a alert message which raised by compiler when we use a statement into applicatiuon but it does not effect/use in application.

Example:

If we define a variable into application but does not used into application then compiler prompt a warning message.

```
Void main()
{
Int a =10;
Clrscr();
Printf("Welcome to world of programming");
Getch();
}
```

Warning: A variable 'a' assigned a value that never used.

 # Some useful shortcut keys:

Shortcut key	Description
ALT + F9	To Compile
CTRL + F9	To Execute Object Code (Linker or Loader)
F2	To Save Source File
F3	To Open an Existing File.
F5	To Show Current Window in Full Screen
F7	To Show Step by Step Process
ALT + F5	To Show Previous/Current output
ALT + Backspace	Undo
ALT + F3	To Close Current Window
ALT + X	To Close Platform of C
Shift + Del	Cut
CTRL + Insert	Copy
Shift + Insert	Paste
F6	To Show Current Open Window one by one
F4	Go to cursor

Screen Related Function:

1) **Textcolor():** is an inbuilt function of conio.h that is responsible to change color of text. It accept a constant value/symbolic constant as following format.

Syntax: To call textcolor();

```
Textcolor(constant/symbolic constant);
```

Example:

Textcolor(4); or textcolor(RED);

2) **Textbackground():()**: is an inbuilt function of conio.h that is responsible to change background color of text. It accept a constant value/symbolic constant as following format.

Syntax: To call textbackground();

```
Textbackground(constant/symbolic constant);
```

Example:

textbackground (1); or textcolor(BLUE);

3) **Cprintf():** is an inbuilt function of conio.h that is responsible to display parenthesized value and change its color.

Syntax: To call cprintf();

Example:

```
Void main()
{
Textbackground(1);
Textcolor(4);
Printf("ONKAR AND SONS\n");
cprintf("ONKAR AND SONS IT ZONE");
getch();
}
```

Output:

ONKAR AND SONS // it will display as default color of compiler

ONKAR AND SONS IT ZONE // it will display in specified color that you have passed

Symbolic constant	Constant	Background	Foreground
BLACK	0	Yes	Yes
BLUE	1	Yes	Yes
GREEN	2	Yes	Yes
CYAN	3	Yes	Yes
RED	4	Yes	Yes
MAGENTA	5	Yes	Yes
BROWN	6	Yes	Yes
LIGHTGRAY	7	Yes	Yes
DARKGRAY	8	No	Yes
LIGHTBLUE	9	No	Yes
LIGHTGREEN	10	No	Yes
LIGHTCYAN	11	No	Yes
LIGHTRED	12	No	Yes
LIGHTMAGENTA	13	No	Yes
YELLOW	14	No	Yes
WHITE	15	No	Yes
BLINK	128	No	No

Example:

```
Void main()
{
Clrscr();
Textcolor(2+128);
Textbackground(4);
Cprintf("welcome to c");
Getch();
}
```

Example:

```
Void main()
{
Clrscr();
Textcolor(GREEN + BLINK);
Textbackground(RED);
Cprintf("welcome to c");
Getch();
}
```

Error: undefined symbol 'GREEN'

Undefined symbol 'BLINK'

Undefined symbol 'RED'

Note:

To resolve this problem a header file conio.h must be include into program manually.

Example:

```
#include<conio.h>
Void main()
{
Clrscr();
Textcolor(GREEN + BLINK);
Textbackground(RED);
Cprintf("welcome to c");
Getch();
}
```

Example:

```
#include<conio.h>
Void main()
{
Clrscr();
Textcolor(GREEN + BLINK);
Textbackground(RED);
Cprintf("welcome to c \n");
Cprintf("welcome to Onkar & Sons IT ZONE");
Getch();
}
```

Example:

```
Void main()
{
Clrscr();
Textbackground(2);
Textcolor(4);
Cprintf("Welcome\n");
Textcolor(1+128);
Cprintf("ONKAR & SONS IT ZONE");
Getch();
}
```

Dealy(): is an inbuilt function of dos.h that is responsible to pause continue execution for certain period it accept an integer value as a millisecond.

Decleration syntax:

Void delay(unsigned milisecond);

Syntax:

To call delay();

```
Delay(unsigned integer value);
```

Example:

Delay(2000);

 # Sleep(): is an inbuilt function of dos.h that is responsible to pause continue execution for certain period. It accept an unsigned integer value as a second.

Decleration syntax:

Void sleep(unsigned second);

Example:

Sleep(2); or delay(2000);

Example: Wap in 'C' to display some message and each message display after 2 second.

```
Void main()
{
Clrscr();
Printf("ONKAR & SONS IT ZONE");
delay(2000); //sleep(2);
printf("\nYour Dream Our Support");
getch();
}
```

Example:

```
Void main()
{
Clrscr();
Textcolor(3);
Cprintf("ONKAR & SONS IT ZONE \t");
sleep(2);
textcolor(4);
cprintf("ONKAR & SONS IT ZONE \t");
textcolor(6);
cprintf("ONKAR & SONS IT ZONE");
getch();
}
```

 # Sound() : is an inbuilt function of dos.h that is responsible to turn on pc speaker at specified frequency.

Decleration of syntax:

```
Void sound(unsigned frequency);
```

 # Nosound(): is an inbuilt function of dos.h that is responsible to turn off pc speaker.

Syntax:

```
Void nosound();
```

Example:

```
Void main()
{
Clrscr();
Sound(400);
Sleep(2);
Nosound();
}
```

 # System: is an inbuilt function of process.h and stdlib.h that is responsible to execute dos command in application of 'C'.

Syntax:

```
Int system(constant char * command);
```

Example:

```
Void main()
{
Clrscr();
Printf("Your Dream Our Support \n");
System("pause");
}
```

Output: **Your Dream Our Support**

Press any key to continue...

FUNDAMENTAL OF C.

About variable:-

A variable is a named location of memory (i.e; RAM) where data will store and manipulated during run-time.

Syntax:- TO define a variable.

<storage specifier> <Modifier> <Qualifier> datatype var1 [= value], var2, ...;

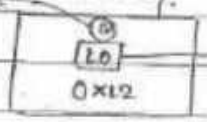
(optional) (optional) (optional) (optional)

↓

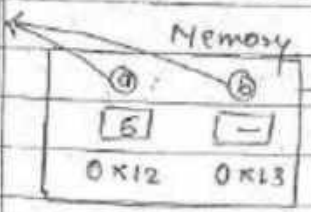
Optional

Ex:-

(i) int a = 10;



(ii) int a = 5, b;



Datatype	Modifier	Storage specifier	Qualifier.
char	signed	auto	const
int	unsigned	static	volatile.
float	short	extern	
double	long	register	
void			

Naming convention of variable.

1 (i) A variable name must be start with an alphabet or underscore and followed by any sequence of characters except embeded space, hypen(-) and period (.).

Ex:-

```
int sal; ✓
int 512; ✓
int 1sal; ✗
int _sal; ✓
int emp.code; ✗
int emp-code; ✗
int emp_code; ✓
int emp code; ✗
```

2. A variable name must not be match to reserve word.

Ex:-

```
int auto; ✗
int Auto; ✓
int static; ✗
int STATIC; ✓
```

3. C language is case sensitive so, we can define more than one variable at a same name by differ it's case.

Ex:-
```
int sal;
int SAL;   } Different
int SaL
```

Date
Page 46

4. A variable name must be authentic and provides readability.

Ex:-

```
int employee code;
int abc;
int e code;
```

→ authentic and ~~variable~~ readable.

Datatype:-

A datatype is responsible to ~~reserve allocate~~ memory space on main memory (i.e: RAM) and it specify following two properties:-

(i) A size of variable in byte

(ii) A type of variable.

Datatype	Size (in byte)	Description (type)
char	1	Ascii value of single character.
int	2	round integer value
float	4	Fractional value.
double	8	Fractional value
void	0	Pointer or function.

Note:-

0/1 => 1 bit

4 bit => 1 nibble

8 bit => 1 byte

1024 byte => 1 KB.

1024 KB => 1 MB

1024 MB => 1 GB.

1024 GB => 1 TB.

Character Ascii

A - Z => 65 - 90

a - z => 97 - 122

0 - 9 => 48 - 57

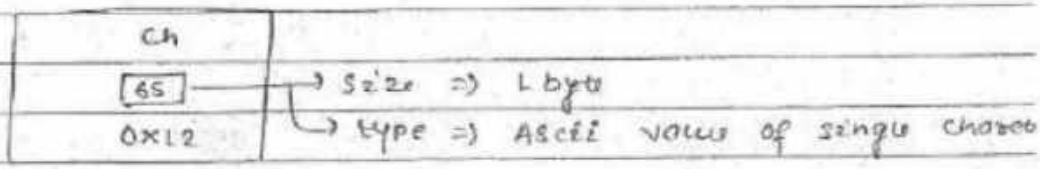
Date
Page 41

Char:- A variable define by char datatype allocates 1 byte memory spou and it can stoke on ascii valw of singu character at a time.

Ex:- char ch;

↓

memory.



(i) ch = A; ✗

(ii) ch = 'A'; ✓

or

ch = 65;

(iii) ch = 'BA'; ✗

↓

Ch

66

0X12.

ch = '\8';

↓

Ch

13

0X12.

(iv) ch = 'ABC'; ✗

Note :-

(i) In c language, a character must be enclosed within singu quotes (') because singu quotes (') is responsibu to drop ascii valw of characters.

(ii) A singu quotes allow singu character or doubl character. It allow doubu character due to esc sequence character.

(v) ch = 'A' + 32;

= 65 + 32;

ch = 97;

(vi) ch = 'A' * 2;

= 65 * 2;

ch = 130

(vii) ch = 260; ↵

↓

ch

| 4 |

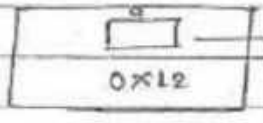
0x12.

int: A variable defined by int datatype, allocates 2 byte memory space and it can store signed round integer value.

Ex: int a;

↓

Memory.



(i) a=20; ↵

(ii) a = -25; ↵

(iii) a = +25; ↵

(iv) a = 'B'; ↵

↓

a

| 66 |

Date
Page 43

(vi) a = 45.25; ←

↓

a

| 45 |

0x12

(vii) a = 45.25f; ←

↓

a

| 45 |

0x12

(viii) a = '8' - 10; ←

↓

a = 66 - 10;

↓

a

| 56 |

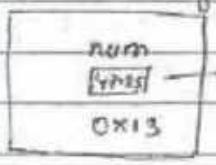
0x12.

Float :- A variable defined by float datatype allocates 4 byte memory space and it can store signed fractional value. It allo 0-9 digit, hypen(-) or period (.).

Ex:- float num;

↓

memory.



(i) num = 47.25; ↵
↓ 8 byte
because a Fractional value by default promoted in double type.

(ii) num = 47.25F; ↵
↓ 4 byte
because it convert in float type due to letter 'F' or 'f'

(iii) num = -85.7; ↵

(iv) num = 85; ↵
↓
num
85.0
0x15

Note:- In C language a Fractional value by default promoted into double type so, when it is required to convert Fractional value into float type then Fractional by value must be postfix by letter F or f.

double:- A variable defined by double datatype, allocates 8 byte memory space and it can store signed Fractional value. It allows 0-9 digit, hypen (-) or period (.).

Ex:- double num;
↓
Memory
num

(i) num = (47.25); ✍

↓ 8 byte

because a fractional value by default promoted in double type

(ii) num = (47.25f); ✍

↓ 4 byte

because it convert in float type due to letter 'f' or 'F'

(iii) num = -85.7; ✍

(iv) num = 25; ✍

↓

num

25.0

0x13.

DIFFERENCES BETWEEN FLOAT AND DOUBLE.

(i) A Float datatype allocate 4 byte memory space. where a double datatype allocates 8 byte memory space.

(ii) A Float datatype uses single precision where double datatype uses double precision

(iii) A Float datatype allow max^m 6 digits in fractional part where a double datatype allow max^m. 15 digits in fractional part.

Date
Page

Inbuilt / predefined / system defined function.

An inbuilt function can be of two types, these are:-

(i) Formatted inbuilt function.

(ii) unformatted inbuilt function.

(i) Formatted inbuilt function:-

An inbuilt function in which format specifier can be use is known as formated inbuilt function.

Ex:- printf(), scanf() etc.

(ii) unformatted inbuilt function:-

An inbuilt function in which format specifier can't be used is known as unformatted function.

Ex:- getch(), getche(), putc(), getchar() etc.

FORMATTED INBUILT FUNCTION TO DISPLAY CONTENTS ON STDOUT DEVICE (i.e; monitor)

An inbuilt function printf() that reside into stdio.h is responsible to display contents / statements / value on standard output device (i.e; Monitor) as a following format:-

SYNTAX:- TO display contents by using inbuilt function printf.

```
printf(" contents/ statements");
```

Ex:-

(1) WAP in 'C' To display a message on monitor

```
void main()
{
clrscr();
Printf("welcome to C");
getch();
}
```

Output

welcome to C.

ESCAPE SEQUENCE CHARACTER:-

A character preceded by backslash (\) is known as escape sequence character. An escape sequence character is known as special character because it made by using two character but internally treated as single character. An escape sequence character is used for change output formate.

Escape sequence	Description
\n	To change cursor to next line.
\t	For tab.
\r	For carriage return.
\b	For backspace.
\\	To display backslash (\) with informatio
\"	To display double quotes (") with info
\	tion.
\'	To display single quotes (') with infor
\a	To display alarm sound with informat

Date
Page 48

Explanation:-

Ex(E) WAP in 'C' to display following statements:-

(a) welcome
TO
C

(b) welcome
TO
C

(c) This is too good.
=
This is too bad.

(d) " Feel the Sensation of your Destination".

(e) 'Honesty' is the best "policy".

(F) welcome TO \ turbo C3.

Soln:-

(a)
```
void main ()
{
    clrscr ();
    Printf (" welcome \n TO \n C");
    getch ();
}
```

Output

welcome
TO
C

(b)
```
void main ()
{
  clrscr ();
  printf (" welcome\n\t TO\n\t\t C");
}
```

Output

welcome
        TO
                C.

(c)
```
void main ()
{
  clrscr ();
  Printf (" This is too good,\n\n\n this is too bad.");
  getch();
}
```

(d)
```
void main ()
{
  clrscr ();
  Printf ("\"Feel the sensation of your destination\");
  getch ();
}
```

(e)
```
void main ()
{
  clrscr ();
  Printf ("\' Honesty\' is best \" Policy\"");
  getch ();
}
```

(F)
```
void main()
{
  clrscr();
  printf(" welcome to \\turboc3");
  getch();
}
```

Explanation - 2.

(i)
```
void main()
{
  clrscr();
  printf(" welcome \rT");
  getch();
}
```

Output

Telcome.

(ii)
```
void main()
{
  clrscr();
  printf("abc\rABc\b\bb");
  getch();
}
```

Output

ABC

Syntax:-

To display a value of variable or outside value on monitor.

```
Printf ("<Format Specifier", varname);
```

Ex:- int a = 5;

(i) printf ("value of a = %d", a);

Output

value of a = 5.

(ii) Printf ("%d", 9);

output

5.

About Format Specifier:-

A character precided by symbol '%' is known as format specifier and a format specifier ensure following properties:-

(i) It ensure total no. of value required fro outside.

(ii) It ensure type of outside value.

(iii) If comming value doesn't given as format specifier then format specifier automatically cast accepted value in it's own compatible type.

Format specifier	Description.
%d	Signed round integer (-32768 to 32767)
%u	Unsigned round integer (0 to 65535).
%c	For character.
%ld	For long int
%f	For Fractional (Float)
~~%lf~~ %lf	For Fractional (double)
%s	For string.
%x	For hexadecimal (capital letter)
%x	For hexadecimal (small letter).
%#X	For hexadecimal (preceded by 0x and capital letter)
%#x	For hexadecimal (preceded by 0x and small letter).
%o	For octal.
%#o	For Octal (Preceded by zero).
%%	To drop symbol % with information.

EXPLANATION

(i) void main()
{
int a=10;
clrscr();
Printf("value of a = %d", a);
getch();
}

Output
value of a=10.

2. void main ()
{
int a=65;
clrscr ();
printf ("%d, %c", a, a);
65 A
getch ();
}

Output
65, A.

3. void main ()
{
clrscr ();
printf ("%c: %d", 65, 'A');
A 65.
getch ();
}

Output
A:65

4. void main()
{
int a=25;
float b=45.35;
clrscr();
printf ("a = %d, b = %f", a, b);
25 45.35
getch();
}

Output
a=25. b=45.35

5. void main()
{
clrscr();
printf("%c%c%c", "welcome", "To", 67);
getch();
}

Output

welcome
To
C

6. void main()
{
int a=31;
clrscr();
Printf("value of a=%d", a);
Printf("\n Hexa =%x, Hexa =%#x", a, a);
Printf("\n hexa =%x, hexa =%#x", a, a);
Printf("\n octal =%o, octal =%#o", a, a);
getch();
}

Output

value of a=31
Hexa =1F, Hexa =0x1F
hexa =1f, hexa =0x1f.
Octal =37, Octal =037

Question:-
WAP in C to assign two integer number calculate and display it sum and average.

Soln:-

```
void main()
{
int a=5, b=10;
clrscr();
Printf("sum = %d", a+b);
Printf("\n Average = %f", (a+b)/2.0);
getch();
}
```

Output

Sum = 15

Average = 7.5.

Explanation (2)

1.
```
void main()
{
clrscr();
Printf("%d", 5.5);
getch();
}
```

Output

0.

2.
```
void main()
{
clrscr();
Printf("%f", 4);
getch();
```

Output:- Abnormal condition Illegal

3. void main ()
{
clrscr ();
Printf ("%f", 5.2);
getch ();
}

Output :- 5.200000

4. void main ()
{
clrscr ();
Printf ("%.0f", 5.2);
getch ();
}

Output :- 5.

5. void main()
{
clrscr ();
Printf ("%.2f", 5.2);
getch ();
}

Output :- 5.20.

6. void main()
{
clrscr();
Printf ("%.2d", 5);
getch ();
}

7. void main()
{
clrscr();
printf("a = %d", 5);
getch();
}

Output: a = 5
(space)

8. void main()
{
clrscr();
printf("a = %d", 5);
getch();
}

Output: a = 05.

9. void main()
{
clrscr();
Printf("a=%3d", 3154);
getch();
}

Output: a= 3154.

10. void main()
{
clrscr();
Printf("sundaram"+5);
getch();
}

Rough
Printf("sundaram"+5);

50	51	52	53	54	55	56	57	58
s	u	n	d	a	r	a	m	\0
0	1	2	3	4	5	6	7	8

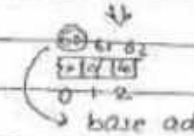
base address → printf

```
// void main()
{
int a=5;
clrscr();
printf("%d",a);
getch();
}
```

Output
5

Rough

printf("%d",a);



printf(&a+1,a);

printf(61,a);

Output
61

WORKING PROCESS OF printf().

1. In printf(), a format specifier associate to outside value left-to-left.
Exp:
```
int a=5, b=10;
printf("a=%d, b=%d", a, b);
```

2. In printf(), a work will be perform right to left.
Exp:
```
int a=10;
printf("a=%d", a, ++a, a=20);
```
Output: a=21

3. When outside value received in printf() then printf() display it on monitor.
Exp:
```
int a=5, b=10;
printf("a=%d b=%d", a, b);
```

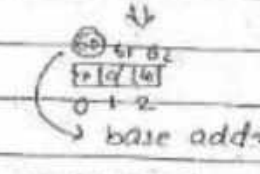
```
void main()
{
int a=5;
clrscr();
printf("%d",a);
getch();
}
```

Output

5.

Rough

printf("%d",a);



printf(&a+1,a);

printf(61,a);

5

5

WORKING PROCESS OF printf().

1. In printf(), a format specifier associate to outside value left-to-left.

Exp:-

```
int a=5, b=10;
printf("a=%d, b=%d", a, b);
```

2. In printf(), a work will be perform right to left.

Exp:-

```
int a=10;
printf("a=%d", a, ++a, a=20);
```

Output:- a=21

3. When outside value received in printf() then printf() display it on monitor.

Exp:-

```
int a=5, b=10;
printf("a=%d b=%d", a, b);
```

Output:- a=5, b=10.

4. Lastly printf() returns total no. of character in displayed statement.

Exp:-

```
Int a=5, x;
x = printf("a = %d", a);
Printf("\n x = %d", x);
```

Output:- a=5
x=3.

EXPLANATION:-

```
Int a = 10, x;
x = Printf("Rays");
Printf("\n x = %d", x);
```

Output:- Rays
x=4.

```
x = printf("man\n go");
Printf("\n x = %d", x);
```

Output:- man
go
x=6.

```
x = printf("man\b go");
Printf("\n x = %d", x);
```

Output:- Man
x=3.

Rough.

printf("man\b go");

50	51	52	53	54	55	56
M	a	n	\b	g	o	\0
0	1	2	3	4	5	6

4. x = printf("education"+3);
printf("\n x = %d", x);

Output
cation
x = 6.

Rough
printf("education"+3);

60	61	62	63	64	65	66	67	68	69
e	d	u	c	a	t	i	o	n	\0

base address.

printf(60+3)
⇓
printf(63);
⇓
cation.

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

Output: a = 10
x = 4.

6. x = 'A' + printf("a = %d", 4.5);
printf("\n x = %c", x);

Output: a = 0
x = A

7. x = 010 + printf("mon%s", "go");
printf("\n x = %d", x);

Output: Mongo
x = 13.

8. x = 0x12 + printf("%o, %d", 020, 0x13);
printf("x = %d", x);
(020 = 16, 0x13 = 19)

Output: 16, 19
x = 27

9. r='a' - printf("a = %d", a==5, ++a, a=15);
printf("\n r = %d", r);

Output:- a=0
r=94.

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

Output:- 10, (85) → garbage.
r=5.

11. r=printf("India", printf("Go"));
Printf("\n r = %d", r);

Output:-
GoIndia.
r=5.

12. r= printf("man", "go");
printf("\n r = %d", r);

Output:-
man go
r=5.

13. r= printf("man", "go");
printf("\n r = %d", r);

Output:-
man.
r=3.

14. x = printf ("mon\t-s", "go");
printf ("\n x = %-d", x);

Output
mango
x = 5.

15. x = printf ("%-d, %-d", printf ("computer"), printf ("Raja"
printf ("\n x = %-d", x);

Output: Raja computer 8, 4.
x = 3

16. x = printf ("\n %-d", printf ("%-x", printf ("India") + 15));
printf ("\n x = %-d", x);

Output: India 14
2
x = 2.

17. x = '10' + printf ("%-d", printf ("%-#0", printf ("India"))
2 05
printf ("Bha %-s\n", "raf"));
printf ("\n x = %-d", x);

Bharat
India 05 2
x = 50.

18. r = 0x15 - printf("%d", printf("\t Talent", printf("India\t" "got")), printf("Hindustan\n"));

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

Hidustan
India got Talent7
r=20.

19. r=020 + printf("%d, %d", printf("James", printf("%s", "Gosling", printf("India%d", printf("Bharat\n" 0x23);

16 and 1+4=20 characters

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

Bharat
Indian7 Gosling James 5,35
r=20.

About Comma (,)

Comma(,)

As an Operator | As a Seperator.

(2) A comma(,) can be used for seperate more than one statement.

Ex:-

int a, b, c, d;

(iii) When a comma(,) is used as an operator then it perform a task as a following way.

(a) It performs a work left to right.

(b) It returns a value of last statement.

Ex:-

```
int a=10, b;
(i) b = 0, 1, 2;
    Printf ("b = %d", b);
```

Output

b=0.

```
(ii) b = (0, 1, 2);
     Printf ("b = %d", b);
```

Output

b=2.

```
(3) b= Printf ("come"), Printf ("wel");
    Printf ("\nb = %d", b);
```

Output

comewel

b=4.

```
(4) b= ( Printf ("come"), Printf ("wel"));
    Printf ("\nb = %d", b);
```

Output

comewel

b=4

(5) b= Printf("%d", Printf("Bharat"), Printf("Hindustan"));
Printf("\nb=%d", b);

Output
Hindustan Bharat6
b=1

(6) b= Printf("%d", (Printf("Bharat"), Printf("Hindustan
Printf("\nb=%d", b);

Output-
BharatHindustan9
b=1

(7) b=20+ Printf("%d, %d", (Printf("Feel"), Printf("Bad")),
Printf("Indian\n"));
Printf("\nb=%d", b);

Output
Indian
Feel Bad 3, 7
b=1

(8) b= Printf("%d, %d", (Printf("mon=5", "go"), Printf("Happy
Printf("Daylight"+8))),
Printf("abc\t ac\b a\nac"));
Printf("\nb=%d", b);

Output
acc mangolight Happy 5, 3.
b=3.

ACCEPTING VALUES FROM KEYBOARD.

The C compiler provides an inbuilt function named scanf() that is responsible to accept values from keyboard as a following format:-

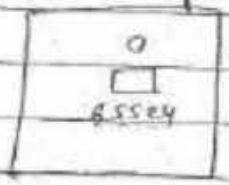
Syntax:-

scanf("<Format specifier>", &var);

Note:- An operator ampersend (&) is responsible to drop address of variable.

Ex:-
```
void main()
{
    int a;
    clrscr();
    printf("%d %x", &a);
    printf("\n %u", &a);
    getch();
}
```

Stack region.



Output:-
```
FFF4
65524
```

Q WAP in 'C' to accept an integer no. from keyboard and display it on monitor with an appropriate message.

Soln:-
```
void main()
{
    int a;
    clrscr();
    printf("enter ano:");
    scanf("%d", &a");
```

```
printf ("value of a = %d", a);
getch ();
}
```

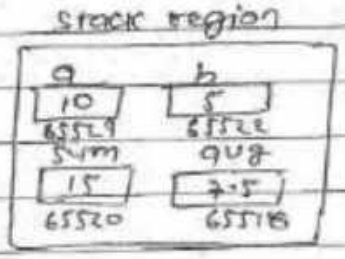
Output

enter a no. = 20 ↵

value of a = 20.

9. WAP in C to accept two integer no. from keyboard. Calculate and display sum and avera

```
void main()
{
int a, b, sum;
float avg;
clrscr ();
printf ("enter two no:");
scanf ("%d%d", &a, &b);
sum = a+b;
avg = sum / 2.0;
printf ("sum = %d\n Average = %f", sum, avg);
getch ();
}
```



Output

enter two no: 10 5 ↵

Sum = 15

Average = 7.5

Enter two no: 10 ↵

5 ↵

Sum = 15

Average = 7.5.

NOTE:- when in scanf() a seprator is not available then more than one inputted value must be seprate by enter or space.

2nd method

```
void main()
{
    int a,b;
    clrscr();
    printf("enter two no.");
    scanf("%d,%d",&a,&b);
    Printf("Sum = %d \n Average = %f",a+b,(a+b)/2.0);
    getch();
}
```

Output

Enter two no = 10, 5,
Sum = 15
Average = 7.5,

WORKING PROCESS OF scanf()

1. In scanf(), a format specifier associate with variable left to left

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

2. In scanf(), a work will be perform left to right.

3. An inputted value must be seperate by seperator which are given into scanf().

Ex:-

```
printf (" enter two no:");
scanf (" %d @ %d", &a, &b);
```

Output

Enter two no: 10 @ 5.

4. Lastly scanf() returns total number of values accepted from keyboard.

Ex:-

```
int a, b, x;
printf (" Enter two no.");
x = scanf (" %d, %d ", &a, &b);
printf (" a = %d, b = %d, x = %d", a, b, x);
```

Output

Enter two no. 10, 15.

a = 10, b = 15, x = 2.

Explanation

```
int a, b, x;
x = printf ("\n %d", scanf("%d %d", &a, &b), printf("enter two
  printf ("a = %d, b = %d, x = %d", a, b, x);
```

Output

enter two no: 10 5 4

—

a = 10, b = 5, x = 2.

(ii)

```
x = 'A' - printf ("%d\n", (scanf ("%d", &a), printf (" value of a
    %d", a)), printf (" enter a no:"));
printf (" value of x = %d", x);
```

Output

Enter a no: A 1 —

TYPES OF VARIABLE

In c language variables categorised as a following way:-

(I) Allocating at compile-time.
   (a) local variable
   (b) globle variable

(II) Allocating at Run-time.

Figure-1

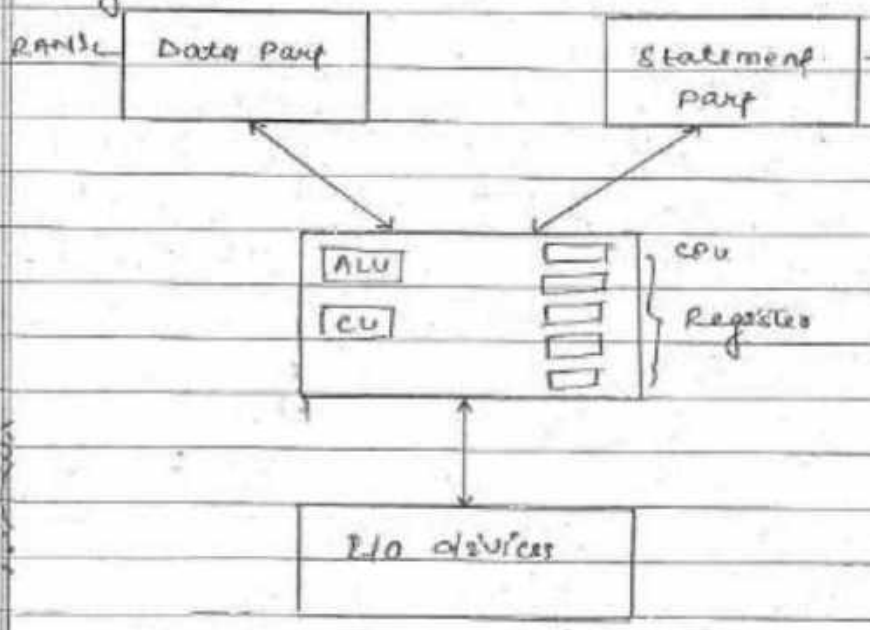
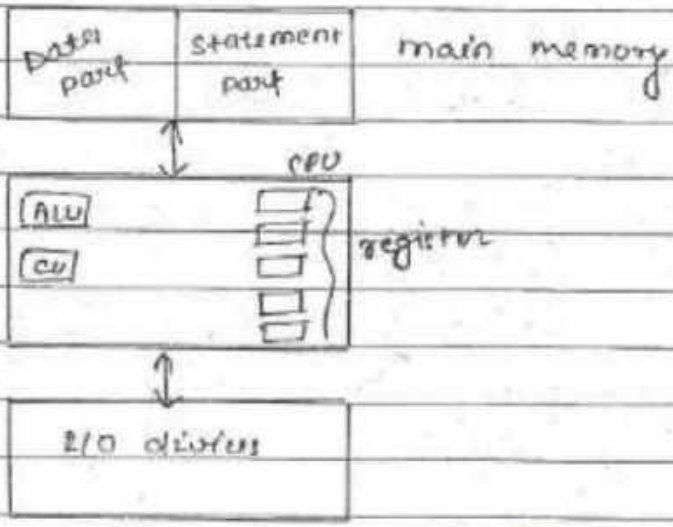
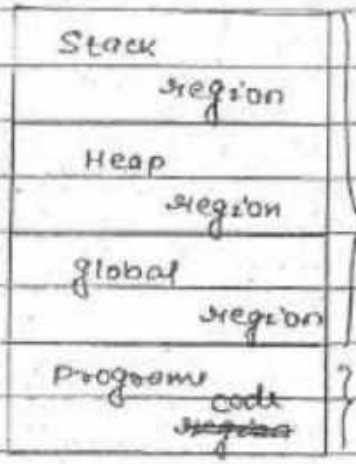


Figure-2



NOTE:- When an application of C successfully compiled then compiler request to operating system for memory space when os divides main memory into four region,



Range of address :) 0 to 65535

1. Allocating at compile-time.

(a) Local variable:-

A variable difined within open(ξ) and clo. curly brace are known as local variable.

Or,

A variable defined within a function is known as local variable.

A local variable posses the following properties:-

(i) It allocates into stack region.

(ii) By default initilised by garbage

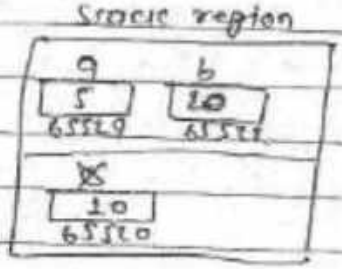
(iii) It can be access by only those statement which are enclosed within it's own open(ξ and close(ξ) curly braces.

(iv) It automatically deallocates when it's ow close (}) curly braces found.

(v) In stack region an operation perform as LIFO (last in first out).

Examples

```
void main()
{
    int a=5, b;
    clrscr();
    {
        int c=10;
        Printf ("a = %d, b = %d, c = %d", a, b, c);
    }
    b=20;
    Printf ("\n value of a = %d", a);
    Printf ("\n value of b = %d", b);
    Printf ("\n value of c = %d", c);
    getch();
}
```



Output

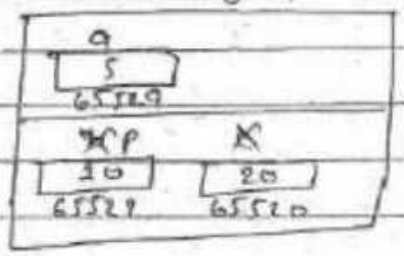
a=5, b=(25), c=10
garbage.

value of a=5.

value of b=20.

GARBAGE VALUE PREDICTE

```
void main ()
{
  int a=5;
  class ();
  {
    int b=10, c=20;
    printf ("a = %d, b= %d, c= %d", a, b, c);
  }
  {
    int P;
    Printf ("\n value of P = %d", P);
  }
  Printf ("\n value of a = %d", a);
// Printf ("\n value of P = %d", P);
   getch ();
}
```



Output

a=5, b=10, c=20.
value of P = 10
value of a = 5.

Note :- STACK REGION.

Datatype	Starting address.
char	65525
int	65524
float	65522
double	65518.

CLASS
Date
Page 41

Global variable:-

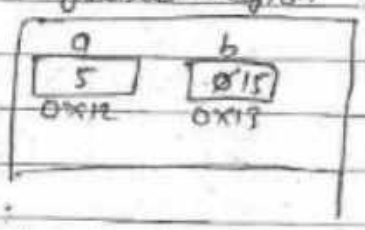
A variable defined outside of open ({) and close (}) curlie brace or outside of function are known as global variable.

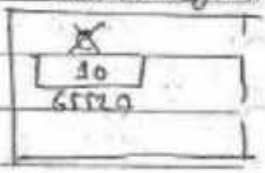
A global variable holds following properties

i) It allocates into global region
ii) By default initialized by zero.
iii) It can be access by all followed statement.
iv) It automatically deallocates when programme terminates.

Ex:- int a=5, b; // global variable.

```
void main()
{
  clrscr();
}
int c=10; // global variable.
printf("a=%d, b=%d, c=%d", a, b, c);
}
b=15;
printf("\n value of a =%d", a);
printf("\n value of b = %d", b);
// printf("\n value of c = %d", c);
  getch();
}
```





Output

a=5, b=0, c=10
value of a = 5.
value of b = 15.

Explanation - 2.

```
int a=5; // global variable
void disp()
{
  printf("\n value of a=%d", a);
// printf("\n value of b=%d", b);
}
int b=10;
void main()
{
  clrscr();
printf(" a=%d, b=%d", a, b);
disp();
getch();
}
```

Output

a=5, b=10

value of a=5.

Note:- In C language, more than one variable can have same name by differ it's scope or region.

Ex:-

```
int a=5;
void main()
{
  int a=10;
  clrscr();
}
  int a=20;
  printf("%d", a);
```

```
{
printf("\n a = %d", a);
printf("\n a = %d", a);
getch();
}
```

Output

20, 10, 10.

Note:- when we define a local variable and global variable as a same name then a local variable automatically hides global variable and this process is known as shadowing.

When it is required to access a global variable in this case of shadowing then C compiler supports and operator named scope resolution operator (::) that can be used as a following format to access global variable.

Syntax:-

```
:: gvar = value / expression;
```

or

```
printf(" message < format specifier)", :: gvar);
```

Ex:- `int a = 5;`

```
void main()
{
int a = 10;
clrscr();
}
```

```
int a=20;
print("%d,",a);
{
printf("%d,",::a);
printf("%d",a);
getch();
}
```

output

20, 5, 10

NOTE:- A scope resolution operator (::) is also known as access operator.

ABOUT MODIFIER:-

The C compiler provides four reserved word named signed, unsigned, short and long which are known as modifier.

A Modifier is responsible to modify type and size of varia

(i) Signed:- When a variable definition statement not preceded by unsigned modifier manually then a variable by default preceded by signed modifier.

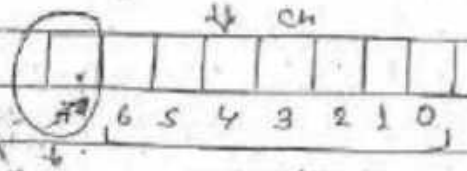
A signed modifier ensure that a variable can store positive as well as negative ~~variable~~ value.

Date
Page 76

Ex:-
(signed) char ch;

or

signed char ch;



left most bit (LMB)

or

Most significant bit (MSB)

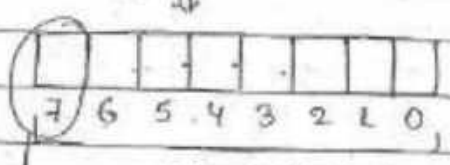
Reserved for sign (+ve/-ve)

0 ⇒ +ve

1 ⇒ -ve.

2. unsigned:- when a variable definition statement preceded by unsigned modifier manually then it ensure that a variable can store only positive value.

Ex:- unsigned char ch;



LMB/MSB

Free to sign (+ve/-ve).

MODIFIER WITH Char.

A datatype char allow only two modifier i.e: signed or unsigned.

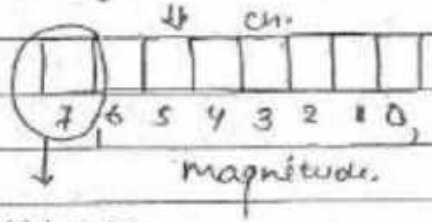
1. Signed with char.

ex:-

signed char ch;

or

signed char ch;



LMB / MSB

reserved to sign.

0 ⇒ +ve

1 ⇒ -ve.

Minimum

1	0	0	0	0	0	0	0
7	6	5	4	3	2	1	0

$\Rightarrow -1\times2^7 + 0\times2^6 + 0\times2^5 + 0\times2^4 + 0\times2^3 + 0\times2^2 + 0\times2^1 + 0\times2^0$

$= -128.$

Maximum.

0	1	1	1	1	1	1	1
7	6	5	4	3	2	1	0

$= 0\times2^7 + 1\times2^6 + \dots + 1\times2^0.$

$= 127.$

range -128 to 127

Date
Page 80

Shortcut Method

Minimum -2^{n-1} Maximum $2^{n-1}-1$ Here, n represent total no. of bit.

Ex:-

$n=8$

Minimum

$-2^{8-1} = -2^7 = -128$

Maximum.

$2^{n-1}-1 = 2^{8-1}-1$

$= 2^7-1 = 128-1$

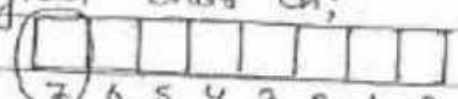
$= 127.$

range -128 to 127

(3) unsigned with char.

Ex:-

unsigned char ch;



LMB / MSB Magnitude

Free To Sign

Minimum

0	0	0	0	0	0	0	0
7	6	5	4	3	2	1	0

$= 0\times2^7 + 0\times2^6 + \cdots + 0\times2^0$

$= 0$

Maximum

1	1	1	1	1	1	1	1

$= 1\times2^7 + 1\times2^6 + \cdots + 1\times2^0$

$= 255$

range 0 to 255

Shortcut Method

Minimum

0

Maximum

$2^n - 1$

Ex:- n28

Minimum

0

Maximum

$2^n - 1$

$= 2^8 - 1$

$= 256 - 1$

$= 255$

range 0 to 255

NOTE:- when we apply short and long mod with char datatype then it does not effect on Types.

MODIFIER WITH int.

A datatype int allow all modifier i.e. signed, unsigned, short or long.

(1) Signed with int.

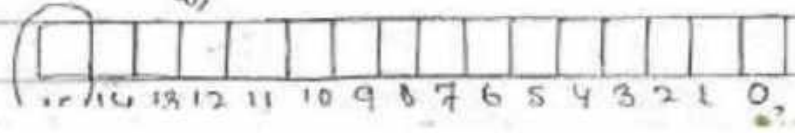
Ex:-

[Signed] int n;

or

signed int n;

↓



class
Date
Page 8

Reserved For sign (+ve/-ve)

0 ⇒ +ve

1 ⇒ -ve.

Minimum.

1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

$\Rightarrow -1 \times 2^{15} + 0 \times 2^{14} + \dots\dots 0 \times 2^0$

$= -32768.$

Maximum

0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

$\Rightarrow 0 \times 2^{15} + 1 \times 2^{14} + 1 \times 2^{13} + \dots\dots 1 \times 2^0$

$\Rightarrow 32767.$

range -32768 to 32767

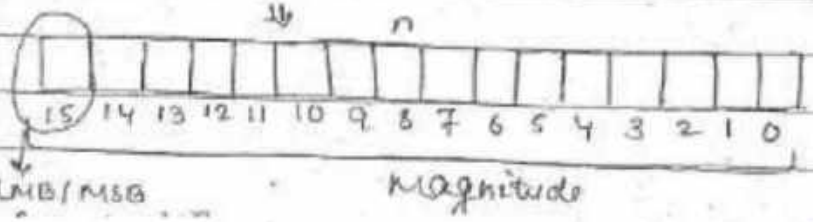
Shortcut Method

Minimum	Maximum.
-2^{n-1}	$2^{n-1}-1$
$= -2^{16-1}$	$= 2^{16-1}-1$
$= -2^{15}$	$= 2^{15}-1$
$= -32768.$	$= 32767$

range -32768 to 32767

2. Unsigned with int.

Eg:- unsigned int n;



minimum.

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

$= 0 \times 2^{15} + 0 \times 2^{14} + \dots + 0 \times 2^{0}$

$= 0$

Maximum.

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

$= 1 \times 2^{15} + 1 \times 2^{14} + 1 \times 2^{13} + \dots + 1 \times 2^{0}$

$= 65535.$

range 0 to 65535.

Shortcut Method

Minimum	Maximum.
0	$2^n - 1$
	$= 2^{16} - 1$
	$= 65536 - 1$
	$= 65535$

range 0 to 65535

(3) Short with int.

When a variable preceded by short modifier then it ensure that a variable can be used for bit oriented operation.

Ex:- short int a;

Or,

signed short int a;

↓

range -32768 to 32767

(4) Long with int:-

When long modifier preceded with then it allocates 4 byte memory space

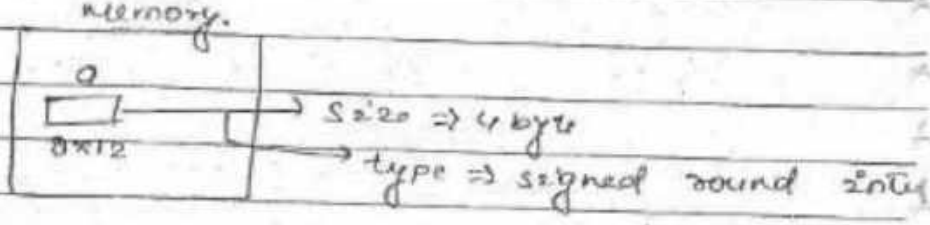
Ex:- long int a;

Or

signed long int a;

⇓

Memory.



Minimum	Maximum.
-2^{n-1}	$2^{n-1}-1$
$=-2^{32-1}$	$=2^{32-1}-1$
$=-2^{31}$	$=2^{31}-1$

NOTE:- A Float and double datatype only allow short and long modifier so, when we try to use signed or unsigned with Float and doubu then leads to compile - time error.

MODIFIER WITH Float.

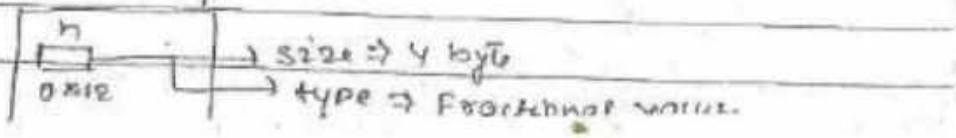
(5) Short with Float:-

Ex:-

Short Float n;

⇓

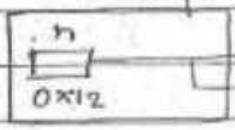
Memory.



(ii) Long with Float:-
When long modifier preceded with Float datatype then it allocates 8 byte memory SP

Ex:- long float n;

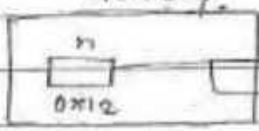
Memory.



MODIFIER WITH Double.

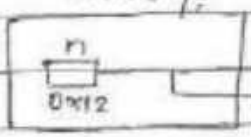
(i) Short with double:-
Ex:- short double n;

Memory.



(ii) Long with double:-
Ex:- long double n;

Memory.



NOTE:- We can omit a data type while variable define variable. So, when we ommit / drop a datatype then by default int will be considered.

Ex:-

(1) signed a; → signed int a;

(2) unsigned a; → unsigned int a;

ABOUT Sizeof():-

A sizeof() is a special operator that is responsible to show memory space required for value, variable, datatype, object etc. in byte.

Syntax to use Sizeof().

var = sizeof value;

↓ Optional

printf("memory=%d", sizeof(var));

Or.

var = sizeof variable

Or

var = sizeof (datatype);

Or

var = sizeof (expression);

Ex:-

```
void main()
{
    int a;
    float b;
    clrscr();
    printf("sizeof a = %d", sizeof(a));
    printf("\n sizeof b = %d", sizeof(b));
    printf("\n sizeof char = %d", sizeof(char));
    printf("\n sizeof short int = %d", sizeof(short int));
    printf("\n long int = %d", sizeof(long int));
```

Number System And Conversion

In this chapter you will learn

- About Number System
- Type of Number System
  - Decimal Number System
  - Binary Number System
  - Octal Number System
  - Hexadecimal Number System
- Conversion between one Number System to Another
  - Decimal to Binary and Binary to Decimal
  - Decimal to Octal and Octal to Decimal
  - Decimal to Hexadecimal and Hexadecimal to Decimal
  - Binary to Octal, Hexadecimal and Octal, Hexadecimal to Binary

---

- **About Number System:** A **numeral system** (or **system of numeration**) is a writing system for expressing numbers, that is, a mathematical notation for representing numbers of a given set, using digits or other symbols in a consistent manner. Numeral systems are sometimes called *number systems*, but that name is ambiguous, as it could refer to different systems of numbers, such as the system of real numbers, the system of complex numbers, the system of *p*-adic numbers etc.

Number System can be classified into following two types:

1. Non-Positional Number System
2. Positional Number System

1. **Non-Positional Number System:** In early days, human being counted on fingers. When counting beyond ten fingers, they used stones, sticks to indicate values. This method of counting uses an additive approach or non-positional number system.
   for eg: Decimal Number System

2. **Positional Number System:** In a positional number system, there are only a few symbols called digits. These symbols represent different values, depending on the position they occupy in a number.

   In computer designs following types of positional number system are supported:
   - Binary Number System
   - Octal Number System
   - Hexadecimal Number System

Note: The value of the base determines the total number of different symbols or digits available in the number system.

- **Decimal Number System:** A number system that uses ten distinct digits (i.e; 0, 1, 2, 3, 4, 5, 6, 7, 8, 9) is known as Decimal number system. In decimal number system, the value of base is 10.
  for eg:
  $(25)_{10}$ or 25

- **Binary Number System:** A number system that uses two distinct digits (i.e; 0, 1) is known as Binary number system. In binary number system, the value of base is 2.
  for eg:
  $(25)_{10} = (11001)_2$

- **Octal Number System:** A number system that uses eight distinct digits (i.e; 0, 1, 2, 3, 4, 5, 6, 7) is known as Octal number system. In octal number system, the value of base is 8.
  for eg:
  $(25)_{10} = (31)_8$

- **Hexadecimal Number System:** A number system that uses sixteen distinct digits (i.e.; 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F) is known as Hexadecimal number system. In hexadecimal number system, the value of base is 16.
  A hexadecimal number system is also known as alphanumeric number system.
  for eg:
  $(31)_{10} = (1F)_{16}$

Note: (i) A sign of octal is zero (0). So, octal number should be preceded by zero (0)
(ii) A sign of hexadecimal is 0x. So, hexadecimal number should be preceded by 0x.

for eg;

25 ⟶ Decimal Number System
025 ⟶ Octal Number System
0x25 ⟶ Hexadecimal Number System

CONVERTING FROM ONE NUMBER SYSTEM TO ANOTHER

There are many methods or techniques which can be used to convert numbers from one base to another. We'll demonstrate here the following:

1. **Decimal to Binary:** In Decimal to Binary conversion, divide integer part of decimal number system by base of binary (2) until the quotient becomes zero and get the remainder from bottom to top for output and multiply fractional part of decimal number system by base of binary (2) and get the carry from top to bottom for output.

Example 1

$25_{10} = ?_2$

	Remainders
25	
12	1
6	0
3	0
1	1
0	1

Hence, $25_{10} = (11001)_2$

Example 2

$42_{10} = ?_2$

	Remainders
42	
21	0
10	1
5	0
2	1
1	0
0	1

Hence, $42_{10} = (101010)_2$

Example 3

$25.75_{10} = ?_2$

Here, $25_{10} = 11001_2$

Now,

0.75 x 2 = 1.50 with carry 1

0.50 x 2 = 1.00 with carry 1

Hence, $(25.75)_{10} = (11001.11)_2$

Example 4

$42.375_{10} = ?_2$

Here, $42_{10} = 101010_2$

Now,

0.375 x 2 = 0.75 with carry 0

0.75 x 2 = 1.50 with carry 1

0.50 x 2 = 1.00 with carry 1

Hence, $(42.375)_{10} = (101010.011)_2$

2. **Binary to Decimal:** In Binary to Decimal conversion, multiply each bit by base of binary (i.e; 2) with power of bit position and add all product term.

Example 1:

$(11001)_2 = ?_{10}$

$(11001)_2 = 1x2^4 + 1x2^3 + 0x2^2 + 0x2^1 + 1x2^0$

$= 16 + 8 + 0 + 0 + 1$

$= 25$

Hence, $(11001)_2 = (25)_{10}$

Example 2:

$(101010.011)_2 = ?_{10}$

$(101010.011)_2 = 1x2^5 + 0x2^4 + 1x2^3 + 0x2^2 + 1x2^1 + 0x2^0 . 0x2^{-1} + 1x2^{-2} + 1x2^{-3}$

$= 32 + 8 + 2 . 0 + 0.25 + 0.125$

$= 42.375$

Hence, $(101010.011)_2 = (42.375)_{10}$

Shortcut Method

Position	2^4	2^3	2^2	2^1	2^0 .	2^{-1}	2^{-2}	2^{-3}	2^{-4}
Weight	16	8	4	2	1	$^1/_2$	$^1/_4$	$^1/_8$	$^1/_{16}$

Example 1:
$(25)_{10} = ?_2$

$(25)_{10} = 16 + 8 + 1$
$= 11001$
Hence, $(25)_{10} = (11001)_2$

Example 2:
$(11001)_2 = ?_{10}$

$(11001)_2 = 1\ 1\ 0\ 0\ 1$
$\downarrow\downarrow\quad\quad\downarrow$
$16\ \ 8\quad\quad 1$
$= 16 + 8 + 1 = 25$
Hence, $(11001)_2 = (25)_{10}$

3. **Decimal to Octal:** In Decimal to Octal conversion, divide integer part of decimal number system by base of octal (8) until the quotient becomes zero and get the remainder from bottom to top for output and multiply fractional part of decimal number system by base of octal (8) and get the carry from top to bottom for output.

Example 1
$25_{10} = ?_8$

8	25	Remainder
	3	1 ↑
	0	3

Hence, $25_{10} = (31)_8$

Example 2
$47_{10} = ?_8$

8	47	Remainder
	5	7 ↑
	0	5

Hence, $47_{10} = (57)_8$

Example 3
$25.75_{10} = ?_8$

Here, $25_{10} = (31)_8$
Now,
$0.75 \times 8 = 6.00$ with carry 6 ↓
$0.00 \times 8 = 0.00$ with carry 0

Hence, $(25.75)_{10} = (31.6)_8$

Example 4
$47.25_{10} = ?_8$

Here, $47_{10} = (57)_8$
Now,
$0.25 \times 8 = 2.00$ with carry 2 ↓
$0.00 \times 8 = 0.00$ with carry 0

Hence, $(47.25)_{10} = (57.2)_8$

4. **Octal to Decimal:** In Octal to Decimal conversion, multiply each digit by base of octal (i.e; 8) with power of digit position and add all product term.

Example 1:
$(31.6)_8 = ?_{10}$

$(31.6)_8 = 3x8^1 + 1x8^0 . 6x8^{-1}$
$= 24 + 1 . 0.75$
$= 25.75$
Hence, $(31.6)_8 = (25.75)_1$

Example 2:
$(57.2)_8 = ?_{10}$

$(57.2)_8 = 5x8^1 + 7x8^0 . 2x8^{-1}$
$= 40 + 7 . 0.25$
$= 47.25$
Hence, $(57.2)_8 = (47.25)$

5. Octal to Binary:

1st Method

Octal → Decimal → Binary

Shortcut Method

Base of Binary = 2
Base of Octal = 8
Hence, $2^3 = 8$ So, represent each digit of octal by using three bit.

Example 1:
$(31.6)_8 = ?_2$

$(31.6)_8 = 011\ 001\ .\ 110$
$= 11001.11$
Hence, $(31.6)_8 = (11001.11)_2$

Example 2:
$(124.14)_8 = ?_2$

$(124.14)_8 = 001\ 010\ 100.001\ 100$
$= 1010100.0011$
Hence, $(124.14)_8 = (1010100.0011)_2$

6. Binary to Octal:

1st Method

Binary → Decimal → Octal

Shortcut Method

In Binary to Octal conversion, divide the binary bits into groups of three starting from the right in integer part and starting from the left in fractional part.

Example 1:
$(11001.11)_2 = ?_8$

$(11001.11)_2 = \underline{011}\ \underline{001}.\ \underline{110}$
$= 31.6$
Hence, $(11001.11)_2 = (31.6)_8$

Example 1:
$(1010100.0011)_2 = ?_8$

$(1010100.0011)_2 = \underline{001}\ \underline{010}\ \underline{100}.\ \underline{001}\ \underline{100}$
$= 124.14$
Hence, $(1010100.0011)_2 = (124.14)_8$

7. Decimal to Hexadecimal: In Decimal to Hexadecimal conversion, divide integer part of decimal number system by base of hexadecimal (16) until the quotient becomes zero and get the remainder from bottom to top for output and multiply fractional part of decimal number system by base of hexadecimal (16) and get the carry from top to bottom for output.

Example 1
$31_{10} = ?_{16}$

16	31	Remainder
	1	16 ↑
	0	1

Hence, $31_{10} = (1F)_{16}$

Example 2
$46_{10} = ?_{16}$

16	46	Remainder
	2	14 ↑
	0	2

Hence, $46_{10} = (2E)_{16}$

Example 3
$46.25_{10} = ?_{16}$

Here, $46_{10} = (2E)_{16}$
Now,
0.25 x 16 = 4.00 with carry 4 ↓
0.00 x 16 = 0.00 with carry 0

Hence, $(46.25)_{10} = (2E.4)_8$

Example 4
$31.15_{10} = ?_{16}$

Here, $31_{10} = (1F)_{16}$
Now,
0.15 x 16 = 2.40 with carry 2 ↓ Approx.
0.40 x 16 = 6.40 with carry 6
0.40 x 16 = 6.40 with carry 6

Hence, $(31.15)_{10} = (1F.266)_8$

8. **Hexadecimal to Decimal:** In Hexadecimal to Decimal conversion, multiply each digit by base of hexadecimal (i.e; 16) with power of digit position and add all product term.

Example 1:
$(2E.4)_{16} = ?_{10}$

$(2E.4)_{16} = 2x16^1 + Ex16^0 . 4x16^{-1}$
$= 2x16^1 + 14x1 . {}^1/_4$
$= 32 + 14 . 0.25$
$= 46.25$
Hence, $(2E.4)_{16} = (46.25)_{10}$

Example 2:
$(2A3.2B)_{16} = ?_{10}$

$(2A3.2B)_{16} = 2x16^2 + Ax16^1 + 3x16^0 . 2x16^{-1} + Bx16^{-2}$
$= 2x16^2 + 10x16^1 + 3x16^0 . {}^1/_8 + {}^{11}/_{256}$
$= 512 + 160 + 3 . 0.125 + 0.042$
$= 675.167$
Hence, $(2A3.2B)_{16} = (675.167)_{10}$

9. **Hexadecimal to Binary:**

1st Method

Hexadecimal → Decimal → Binary

Shortcut Method

Base of Binary = 2
Base of Hexadecimal = 16
Hence, $2^4 = 16$ So, represent each digit of hexadecimal by using four bit.

Example 1:
$(2A.4C)_{16} = ?_2$

$(2A.4C)_{16} = 0010\ 1010.0100\ 1100$
$= 101010.010011$
Hence, $(2A.4C)_8 = (101010.010011)_2$

Example 2:
$(4C.1F)_{16} = ?_2$

$(4C.1F)_{16} = 0100\ 1100.0001\ 1111$
$= 1001100.00011111$
Hence, $(4C.1F)_{16} = (1001100.00011111)_2$

10. Binary to Hexadecimal:

1st Method

Binary → Decimal → Hexadecimal

Shortcut Method

In Binary to Hexadecimal conversion, divide the binary bits into groups of four starting from the right in integer part and starting from the left in fractional part.

Example 1:
$(101010.010011)_2 = ?_{16}$

$(101010.010011)_2 = \underline{0010}\ \underline{1010}.\ \underline{0100}\ \underline{1100}$
$= 2A.4C$
Hence, $(101010.010011)_2 = (2A.4C)_{16}$

Example 2:
$(11001.101)_2 = ?_{16}$

$(11001.101)_2 = \underline{0001}\ \underline{1001.1010}$
$= 19.\ A$
Hence, $(11001.101)_2 = (19.\ A)_{16}$

Note: In Decimal to Another base conversion, follow rule as a decimal to binary conversion and another base to Decimal conversion, follow rule as a binary to decimal conversion.

Example 1
$31_{10} = ?_4$

4	31	Remainder
	7	3 ↑
	1	3
	0	1

Hence, $31_{10} = (133)_4$

Example 1:
$(133)_4 = ?_{10}$

$(133)_4 = 1x4^2 + 3x4^1 + 3x4^0$
$= 16 + 12 + 3$
$= 31$

Hence, $(133)_4 = (31)_{10}$

EXERCISE

1. Why computers use the binary number system?
2. What is the difference between positional and non-positional number system?
3. Convert the following Numbers to their Binary equivalent
   (a) 123 (b) 167 (c) 72.45
   (d) 4097.188 (e) 0.4475

4. Convert the following Binary Numbers to their Decimal, Octal and Hexadecimal equivalent.
   (a) 10110 (b) 10001101 (c) 10111.1011
   (d) 0.0111011 (e) 1011.011

5. Convert the following Decimal Number to Octal and Hexadecimal.
   (a) 102 (b) 547 (c) 72.54
   (d) 675.098 (e) 0.6548

6. Convert the following Hexadecimal number to Binary and Decimal.
   (a) 49 (b) 5C8 (c) FB17
   (d) 4A .67 (e) 8109.4A

7. Convert the following Octal number to Hexadecimal and Binary.
   (a) 137 (b) 4163 (c) 775
   (d) 673 (e) 4453

Ex:-

(1) signed a; → signed int a;

(2) unsigned a; → unsigned int a;

ABOUT Sizeof():-

A sizeof() is a special operator that is responsible to drop memory space required for value, variable datatype, object etc. in byte.

Syntax to use Sizeof().

var = sizeof value;
↓
Optional

printf ("memory=%d", sizeof var);

Or,

var = sizeof variable

Or

var = sizeof (datatype);

Or

var = sizeof (expression);

Ex:-

```
void main()
{
    int a;
    float b;
    clrscr();
    printf("sizeof a = %d", sizeof(a));
    printf("\n sizeof b = %d", sizeof(b));
    printf("\n sizeof char = %d", sizeof(char));
    printf("\n sizeof short int = %d", sizeof(short int));
    printf("\n long int = %d", sizeof(long int));
```

printf("\n long float = %d", sizeof(long float));
printf("\n long double = %d", sizeof(long double));
getch();
}

Output
Size of a = 2
size of b = 4
size of char = 1
Size of short int = 2
long int = 4
long float = 8
long double = 10.

Explanation

(1) printf("%d", sizeof 25);

Output
2

(2) printf("%d", sizeof 25l);

Output
4

(3) printf("%d", sizeof 4.5);

Output
8

(4) printf("%d", sizeof 4.5f);

Output
4.

(5) printf("%d", sizeof 'A');

Output

Rough
sizeof 'A'
1

STORAGE SPECIFIER :-

The C compiler supports four reserved word named auto, static, extern and register which are known as storage specifier.

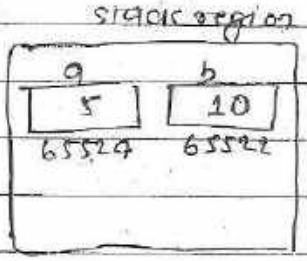
A storage specifier is responsible to specify a place for variable.

(1) auto :- A storage specifier auto specify a memory space into stack region. An auto storage specifier only allowed with local variable.

Ex :-

```
{
    auto int a = 10;
        or
    auto int a = 10;
}
```

Explanation.




```
#include <stdio.h>
#include "conio.h"
    void main()
    {
    auto int a=5;
    auto int b=10;
    clrscr();
        printf("a=%d, b=%d", a, b);
    getch();
    }
```

output

a=5, b=10.

NOTE:- When a local variable defined without storage specifier then it by default preced by auto.

(2) Static:- A Storage specifier static specify a memory space into global region. A static can be use with local variable as well as global variable.

(a) Static with global variable.

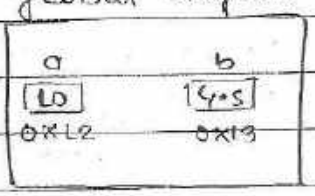
(b) Static with local variable.

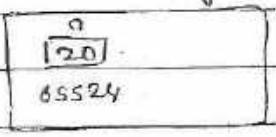
(a) Static with global variable:-

A Static Storage specifier by default precede with global variable so, global variable automatically allocates into global region.

Exp:-

```
static int a=10;
Static float b=4.5;
void main()
{
    auto int a=20;
    clrscr();
    printf("\n value of a = %d", a);
    printf("\n\n value of a = %d", ::a);
    printf("\n\n value of b = %f", b);
    getch();
}
```





Output

value of a = 20

value of a = 10

value of b = 4.500000.

(b) Static with Local variables

When a local variable preceded by static storage specifier manually then it posses / hold following properties.

(i) It allocates into global region.

(ii) By default initialised with 0.

(iii) It deallocates automatically when program terminates.

(iv) It can be access only those statements which are enclosed within it's own open(s) and close(s) curly braces.

Eg:n

```
void main()
{
    int i=1;
    clrscr();
        while(i<=3)
{
  {
    int a=0;
    printf("a = %d\n", a);
    a++;
  }
  //printf("value of a = %d\n",a);
    i++;
}
getch();
}
```

Stack region.

i
1
65524

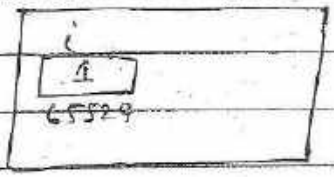
a
10
65522

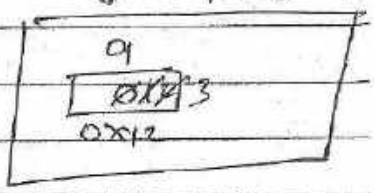
dry run.

i	i<=3	a	output
1	1<=3 ✓	0	a=0
2	2<=3 ✓	0	a=0
3	3<=3 ✓	0	a=0
4	4<=3 ✗	1	

Using Static.

```
void main()
{
    int i=1;
    clrscr();
    while(i<=3)
    {
        {
            static int a;
            Printf("a = %d\n", a);
            a++;
        }
        // Printf("value of a = %d\n", a);
        i++;
    }
    getch();
}
```





dry run.

i	i<=3	a	Output
1	1<=3 ✓	0	a=0
2	2<=3 ✓	1	a=1
3	3<=3 ✓	2	a=2
4	4<=3 ✗	3	

(2) extern: A storage specifier extern only allows with golabe variable. It is used for access a global variable before definition statement.

Syntax: TO use extern.

extern datatype guar;

Ex:n
```
int a=5;
void disp();
void main()
{
    clrscr();
    Printf("a=%d", a);
    //Printf("\nb=%d", b);
    disp();
    getch();
}
int b=10; // global variable
void disp()
{
    Printf("\n value of a=%d", a);
    Printf("\n value of b=%d", b);
}
```

global region.

a
5
0x12

b
10
0x13

Output
a=5
value of a=5
value of b=10.

2 Ex:n
```
int a=5;
void disp();
void main()
{
    extern int b;
    clrscr();
    Printf("a=%d", a);
    Printf("\nb=%d", b);
    disp();
```

```
    getch();
}
int b=10;
void disp()
{
  Printf("\n value of a = %d", a);
  Printf("\n value of b = %d", b);
}
Output
a=5
b=10
value of a=5
value of b=10.
```

Explanation (2)

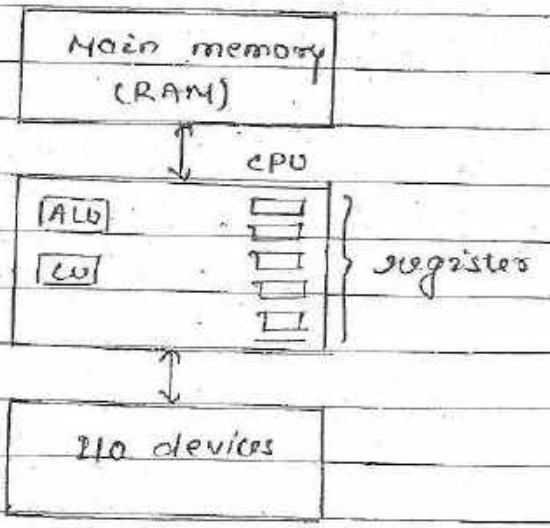
```
int a=5;
void main()
{
  extern int b;
  Printf("a = %d, b= %d", a, b);
  getch();
}
int b=20;
```

Output

a=5, b=20.

(4) Register :- A register storage specifier only allowed with local variable. It is responsible to allocates a local variables direct on register of CPU rather than stack region.

It is logically suggested to define only those variable as register that required in application several times.



NOTE :- (i) When we precede a local variable by register then we can't drop it's address because a register local variable doesn't have a address.

Ex :-

```
void main()
{
    register int a = 10;
    clrscr();
    printf("a = %d", a);
// printf("\n Address of a = %u", &a);
    getch();
}
```

Output

QUALIFIER

The c compiler supports two reserved word named const and volatile which are known as Qualifier.

(1) Const :- when a variable preceded by const then it ensure that a value assign at definition time fixed and it can be access into application but it can't be alter later

Syntax:- | const datatype var [= value];

Ex:-

```
void main()
{
const int a=5;
int b=10; // volatile
clrscr();
printf("a=%d, b=%d", a, b);
// a=20;
b=15;
printf("\n value of b=%d", b);
getch();
}
```

stack segment.

a	b
5	10 / 15
65524	65522

Output

a=5, b=10

value of b=15.

(2) WAP in 'C' to accept radious of circle from keyboard, calculate and display area of circle.

```
void main()
{
    const float PI = 3.14F;
    int r;
    clrscr();
        printf("enter radious of circle:");
        scanf("%d", &r);
        printf("Area of circle = %f", PI * r * r);
    // PI = 7.14F;
    getch();
}
```

Output

Enter radious of circle: 2.
Area of circle = 12.560000

NOTE:- We can alter a value of constant variable by using pointer or scanf.

Exp:

```
void main()
{
    const int a = 5;
    clrscr();
        printf("a = %d", a);
    // a = 20;
    printf("Enter a no:");
    scanf("%d", &a);
    printf("\n a = %d", a);
    getch();
```

Stack region.

a
20
65524

Output

a=5

Enter a no: 20

a=20.

(2) Volatile :- A volatile qualifier ensure that a value of variable can be use into application and it can be alter later on.

Syntax:-

```
volatile datatype var [= value];
```

(optional)

Ex:-

```
void main()
{
    volatile int a=5;
    clrscr();
    printf("\n a=%d", a);
    a=20;
    printf("\n a=%d", a);
    getch();
}
```

Output

a=5

a=20.

CHAPTER-4

classmate
Date
Page 98

OPERATORS.

The C compiler supports following Types OF Operators these are;-

(1) Arithmetic operators.
   (a) Simple arithmetic operators.
   (b) Increment (++) and decrement (--) operat

(2) Assignement operator
   (a) single assignement operator.
   (b) Multiple assignement operator.
   (c) compound / shortcut assignement operator.

(3) conditional / Relational / comparision operators.

(4) Logical operators.

(5) Special operators.

(6) Bitwise operator.

(1) Arithmetic operators.

(a) simple arithmetic operators.

An arithmetic operators are used for perform simple arithmetic operation.

Operators	Description	Example: int a=5, b=2;
+	for addition	printf ("a+b=%d", a+b);
		Output
		a+b=7
-	for subtraction	printf ("a-b = %d", a-b);
		Output
		a-b = 3.
*	for multiplication	printf ("Product = %d", a*b);
		output
		Product = 10.

Operators	Description	Examples int a=5, b=2;
/	For division (Quotient)	printf("result = %d", a/b); Output Result = 2.
%	For division (remainder)	Printf ("remainder = %d", a%b); Output 1

NOTE:- (i) A division(/) operator returns the round integer if numberator and denominator both are round integer otherwise it returns Fractional value.

Ex:-

5/2	5.0/2	5/2.0	5.0/2.0	(Float) 5/2
2	2.5	2.5	2.5	2.5

(i) Printf ("%d", 5/2);

Output

2

(2) printf ("%d", 5.0/2);

Output

0

(3) Printf ("%f", 5.0/2);

Output

2.500000

(4) Printf ("%f", (Float) 5/2);

Output

2.500000

2. A Modulus (%) doesn't allow a fractional value as a numerator and denominator.

Ex:-

5%2	5.0%2	5%2.0	5.5%2
↓	↓	↓	↓
1	Error	Error	Error.

(b) Increment (++) and decrement (--) Operator.

Increment (++):- Operator is used for increase a value of operand by 1.

Ex:-

```
int a = 5;
++a;
 ↓
a = a+1;
  = 5+1;
a = 6;
```

decrement (--):- Operator is used for decrease a value of operand by 1.

Ex:-

```
int a = 5;
--a;
 ↓
a = a-1;
  = 5-1;
a = 4;
```

NOTE:- An increment (++) and decrement (--) Operators can be use as a following two Form.

(i) Prefix Form.

(ii) Postfix form.

(i) Prefix Form:- When an operator comes before an operand then it is known as prefix for

Ex:-

```
++a;
--a;
```

(ii) Postfix Form:- When an operator comes aft and operand then it is known as postfix Form.

Ex:-

```
a++;
a--;
```

Expression:- An expression is a combinatio of more than one operand or operator.

Ex:-

```
++a; x
a--; x
a+b; ✓
y=++a; ✓
y=a+b; ✓
```

NOTE:- When an expression not found then bo Form (i.e; prefix or postfix) produce same resu

```
int a=5;
```

Prefix form.	Postfix form
++a; ⇓ a=a+1 =5+1 a=6	a++; ⇓ a=a+1 =5+1 a=6
--a; ⇓ a=a-1 =5-1 a=4	a--; ⇓ a=a-1 =5-1 a=4

NOTE:- When an expression found then both form (i.e prefix or postfix) produce different result.

~~int a=15;~~ int a=5, y;

y = ++a;	y = a++;
→ a = a+1	↓ → a = a+1
= 5+1	y = 5 = 5+1
a = 6	a = 6
y = 6	
y = --a;	y = a--;
a = a-1	↓ → a = a-1
= 5-1	y = 5 = 5-1
a = 4	a = 4
y = 4	

(2) Assignement Operators.

(a) Single assignement operators:-

An assignement operator(=) is used for assign right side value into left side variable.

Syntax:-

```
Lvar = Rvalue / var / expression;
```

NOTE:- (i) A right side of assignement may be variable, expression or constant but A left side of assignement only allowed a variable.

Ex:-

```
int a=10, b;
```

(i) b=20, ✓

(ii) b = 2*5; ✓

(iii) b = a*4; ✓

↓

b = 10*4;

b = 40

(iv) a*2 = 50; ✗

↓

10*2 = 50;

20 = 50;

(v) 20 = 9; ✗

(b) Multiple assignement operators:-

When it is required to assign same value into more than one variable then we can use multiple assignement as a following way:

Syntax:-

| var1 = var2 = var3 value / constant / expression; |

Ex:-

```
int a, b, c, d;
a = 50;
b = 50;
c = 50;
d = 50;
```

a = b = c = d = 50;

(c) Compound / Shortcut assignement operators:-

When a left side variable just comes after assignement (=) operator then it can be expressed as a following format:-

Syntax:-

| Lvar <Operator> = Rvalue / expression / constant; |

Ex:- int a = 10

a = a+5;

↓

a += 5;

a = a*b-2;

↓

a *= b-2;

10 += 5

15

Compound / Shortcut assignement Operator.

+=, -=, *=, /=, %=.

(3) Conditional / comparison / relational operator

A relational / comparison operator are used for compare two value and returns an integer value (i.e 0 or 1).

If returns 1 if condition met true otherwise it returns 0.

Operators	Description	Example int a=5, b=2;
>	greater than	printf("%d", a>b);
		output
		1
<	less than	printf("%d", a<b);
		output
		0
>=	greater than	Printf("%d", a>=b*5);
	or equal to	output
		0
<=	less than or	Printf("%d", a<=b*5);
	equal to	output
		1
==	equal to	printf("%d", a==b);
		output
		0
!=	not equal	Printf("%d", a!=b);
	to	output
		1

(4) Logical Operators:-
A logical operators are used for combined more than one conditional operator and it also returns 0 or 1

Operators	Description	Example int a=5, b=2;
&&	Logical AND	printf("%d", a>b && b>a);
		Output
		0
||	Logical OR	printf("%d", a>b || b>a);
		Output
		1
!	Logical NOT	printf("%d", !(a>b));
		output
		0

Truth table

OP_1	OP_2	OP_1 && OP_2	OP_1 || OP_2	!OP_1	!OP_2
0	0	0	0	1	1
0	1	0	1	1	0
1	0	0	1	0	1
1	1	1	1	0	0

NOTE:- All operators are categorised into following three types according to nature.
(a) Unary operators
(b) Binary "
(c) Ternary "

(a) Unary operators:- An operators that have requires only one operand to perform an operation is known as unary operators

Ex:- ++, --, *, -, +, &, !, ~, sizeof etc.

(b) Binary Operator:- An Operator that have required two operand to perform an operation is known as binary operator.
Ex:- +, -, %, *, /, >, <=, >=, ==, &&, ||, & etc.

(c) Ternary Operator:- An Operators that have required three operand to perform an operation is known as ternary operator.
Ex:- ?: => Ternary Operator.

APPLICATION BASED ON OPERATORS.

(i) WAP in 'c' to accept two round integer no. from keyboard calculate and display sum and average.

```
Solution:  void main()
           {
              int, a, b, sum;
              Float avg;
              clrscr();
                 Printf(" Enter two no:");
                 scanf("%d, %d", &a, &b);
                 sum = a+b;
                 avg = sum / 2.0;
                 Printf("sum = %d \n Average = %f", sum, avg);
                 getch();
           }
```

Stack region

a	b	sum	avg
5	10	15	7.5
65524	65522	65520	65518.

Output

Enter two no: 5, 10 ↵

sum = 15.

Average = 7.500000.

2nd Method

```
void main()
{
   int a, b;
   clrscr();
      Printf(" Enter two no:");
      scanf("%d, %d", &a, &b);
   Printf(" sum = %d \n Average = %.2f", a+b, (a+b)/2.0);
   getch();
}
```

Output

Enter two no: 5 10 ↵ Average = 7.50

(2) WAP in 'C' to accept a three digit no. From keyboard and display it's reverse.

Soln:-

```
void main()
{
    int n, d1, d2, d3;
    clrscr();
    printf("Enter a three digit no:");
    scanf("%d", &n);
    d1 = n % 10;
    n = n / 10;
    d2 = n % 10;
    n = n / 10;
    d3 = n;
    printf("reverse no: %d%d%d", d1, d2, d3);
    getch();
}
```

dry run.

n	d_1	d_2	d_3	output
245	5	4	2	Enter a three digit no: 245
24				reverse no: 542
2				

10) 245 (24
20
*45
40
*5

2nd Method

```
void main()
{
    int n, d1, d2, rev;
    clrscr();
    printf("Enter a three digit no:");
    scanf("%d", &n);
```

```
d1 = n % 10;
n = n / 10;
d2 = n % 10;
n = n / 10;
rev = d1 * 100 + d2 * 10 + n;
Printf (" reverse = %d", rev);
getch ();
}
```

dry run.

n	d_1	d_2	rev	output.
245	5	4	= 5 * 100 + 4 * 10	Enter a three digit no 245
24			+ 2	
2			= 500 + 400 + 2	reverse = 542.
			= 542	

Method-3th

```
void main ()
{
int n, rev;
clrscr ();
        Printf("Enter a three digit no:");
        scanf ("%d", &n);
rev = n % 10 * 100 + (n/10) % 10 * 10 + n/100;
Printf (" reverse = %d", rev);
getch ();
}
```

Rough.

rev: 245 % 10 * 100 + (245 / 10) % 10 * 10 + 245 / 100;

= 5 * 100 + (245 / 10) % 10 * 10 + 245 / 100;

500 + 24 % 10 * 10 + 245 / 100;

= 500 + 4 * 10 + 245 / 100;

= 500 + 40 + 2

4th Method

```
void main()
{
int n;
clrscr();
printf("Enter a three digit no:");
scanf("%d", &n);
printf("reverse = %d%d%d", n%10, (n/10)%10,
                                    n/100);
getch();
}
```

Q. WAP in 'C' to accept item code, unit price and total quantity from keyboard, calculate and display invoice / bill where 20% discount available on all items.

Item code. ::
Unit price in Rs::
Quantity taken ::
Total amount in::
Discount amount::
Net payable amount::

Soln

```
void main()
{
int icode, qty;
float uprice, total, dist, pamt;
clrscr();
```

```
printf (" enter item code, unit price and total quantity
scanf (" %d %f %d", &icode, &uprice, &qty);
total = uprice * qty;
dist = total * 20/100;
pamt = total - dist;
clrscr ();
printf (" ==========");
printf ("\n Item code :: %d", icode);
printf ("\n ==========");
printf("\n unit price in Rs:: %f", uprice);
printf("\n Quantity taken :: %d", qty);
printf("\n ==========");
printf ("\n Total amount in :: %f", total);
printf ("\n Discount amount :: %f", dist);
printf ("\n ==========");
printf("\n Net Payable amount :: %f", pamt);
getch();
}
```

2nd Method

```
void main()
{
  int icode, qty;
  float uprice;
  clrscr();
  printf (" enter item code, unit price and total quantity
  scanf (" %d %f %d", &icode, &uprice, &qty);
  clrscr();
  printf (" ==========");
  printf ("\n Item code :: %d", icode);
  printf ("\n ==========");
  printf("\n unit price in Rs:: %f", uprice);
```

```
Printf("\n Quantity taken :: %d", qty);
Printf("\n ==========");
Printf("\n Total amount in :: %f", uprice * qty);
Printf("\n Discount amount :: %f", (uprice * qty) * 20/100)
Printf("\n ==========");
Printf("\n Net Payable amount :: %f", (uprice * qty) -
                    (uprice * qty) * 20/100);
getch();
}
```

Q WAP in 'C' to accept two Integer no. From keyboard and swap both to eachother by using third variable.

Soln:

```
void main()
{
    int a, b, temp;
    clrscr();
    Printf("Enter two no:");
    Scanf("%d %d", &a, &b);
    Printf("a = %d, b = %d", a, b);
temp = a;
a = b;
b = temp;
Printf("\n After swapping!");
Printf("\n a = %d, b = %d", a, b);
getch();
}
```

Stack region.

a	b	temp
10	5	5
65524	65522	65520

Output

Enter two no: 5 10

a=5, b=10

After swapping!

WAP in 'C' to accept two fractional no. from keyboard and enter change or swap both to each other without using third variable

```
void main()
{
    float a, b;
    clrscr();
    printf("enter two fractional no:");
    scanf("%f %f", &a, &b);
    printf("a = %f, b = %f", a, b);
a = a+b;
b = a-b;
a = a-b;
printf("\n After swapping!");
printf("\n a = %f, b = %f", a, b);
getch();
}
```

Output

Enter two fractional no: 4.5 2.4

a = 4.5, b = 2.4

After swapping!

a = 2.4, b = 4.5.

classmate
Date
Page 116

OPERATOR PRECEDENCE CHART.

OPERATOR TYPE	OPERATOR	ASSOCIATIVITY
Primary Expression Operators.	() [] . ->	left-to-right.
	exp++ exp--	
Unary Operators.	* & + - !	right-to-left.
	~ ++exp -- exp	
	(typecast) sizeof	
	*/%	
	+ -	
	>> <<	
	< > <= >=	left-to-right.
Binary operators	== !=	
	&	
	^	
	|	
	&&	
	||	
Ternary Operators	?:	right-to-left.
Assignment Operators	= += -= /= >>=	right-to-left
	<<= &= ^= |=	
comma.	,	left-to-Right

Explanation.

int a=5, b=2, z=5, y;

(i) y = a+b*2;
= 5+2*2
= 5+4
y=9.

(ii) y = (a+b)*2;
= (5+2)*2;
= 7*2;
y=14

(iii) y = z++ - z++;
= 5-5;
= y=0
Printf("y=%d, z=%d", y, z);
Output
y=0, z=7.

(iv) y = ++z + ++z + ++z;
= 8+8+8
y=24
Printf("y=%d, z=%d", y, z);
Output
y=24, z=8.

(v) y = z++ - z++;
= ~~z++ - 5~~;
= ~~7~~;
y=~~7~~

= 6-z++;
= 6-6;
= 0

printf("%d, %d", y, z);
Output
2, 7.

(vi) y = a*b/3;
= 5* 2/3;
= 10/3;
y=3

(vii) y = a&&b||0;
= 5&&2||0;
= 1||0;
y=1;

(viii) $y = a \,||\, b \;\&\&\; 20;$
$= 5 \,||\, 2 \;\&\&\; 20;$
$= 5 \,||\, 0;$
$y = 1;$

(ix) int $i = -3, j = 2, k = 0, m;$
$m = ++i \;\&\&\; ++j \,||\, ++k;$
printf("%d, %d, %d, %d", i, j, k, m);

Output
$-2, 3, 0, 1$

(x) int $i = 5, j = -1, y;$
$y = ++j \;\&\&\; ++i;$
$= ++j \;\&\&\; 6;$
$= 0 \;\&\&\; 6;$
$y = 0$

Rough
$M = ++i \;\&\&\; ++j \,||\, ++k;$
$= -2 \;\&\&\; 0 \,||\, ++k;$
$= -2 \;\&\&\; 3 \,||\, ++k;$
$= 1 \,||\, ++k;$
$M = 1$

(xi) int $i = 5, j = 0, y;$
$y = j \;\&\&\; ++i;$
$= 0 \;\&\&\; ++i;$
$y = 0;$
printf("%d, %d, %d", i, j, y);

Output
$5, 0, 0.$

(xii) int $i = 5, j = 0, y;$
$y = j \,||\, ++i;$
$= 0 \,||\, ++i;$
$= 0 \,||\, 6;$
$y = 1$
printf("%d, %d, %d", i, j, y);

Output
$6, 0, 1.$

```
int i=5, j=1, y;
y = j && ++i;
  = 1 && ++i;
  = 1 && 6;
y = 1
printf("%d, %d, %d", i, j, y);
```

Output

6, 1, 1.

```
int i=5, j=1, y;
y = j || ++i;
  = 1 || ++i;
y = 1
printf("%d, %d, %d", i, j, y);
```

Output

5, 1, 1.

```
int i=-3, j=2, k=0, m;
m = ++i || ++j && ++k;
printf("%d, %d, %d, %d", i, j, k, m);
```

Output

-2, 2, 0, 1.

Rough

m = ++i || ++j && ++k;

-2 || ++j && ++k;

m = 1

HOW OVERFLOW AND UNDERFLOW HANDLE IN C.

When a right side value exceed range of left side variable then an overflow and underflow occurred

When an overflow and underflow raised then it must be handle as a following way:-

Syntax:- `Lvar = Rvalue % capacity of Lvar;`

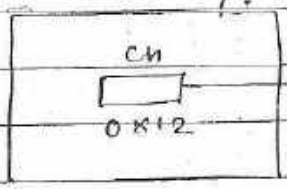
Here, capacity of Lvar = 2^n

and n represent total no. of bit in left side variable.

Ex:- `char ch;`

↓

Memory.



(i) `ch = 20;`
`Printf("%d", ch);`

Output
20

(ii) `ch = 260; // overflow.`
`= 260 % 256;`
`ch = 4;`

capacity of ch = 2^n
$= 2^8$
$= 256$.

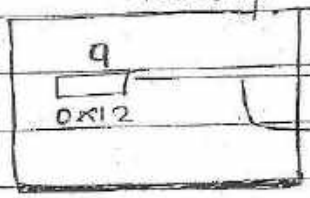
(iii) `ch = -270; // underflow.`
`= -270 % 256;`
`= -14;`

(iv) ch = 130; // Overflow.
= 130 % 256;
= 130 - 256;
ch = -126;

(v) ch = -130; // underflow.
= -130 % 256;
= -130 + 256;
ch = 126;

unsigned int q;

Memory



(i) n = -5; // underflow. capacity of q = 2^n
= -5 % 65536; $= 2^{16}$
= -5 + 65536; = 65536.
n = 65531;

(ii) n = 65570; // overflow.
= 65570 % 65536;
n = 34;

Explanation

%d ⇒ -32768 to 32767 ⇒ signed int. } capacity
%u ⇒ 0 to 65535 ⇒ unsigned int. } ⇓
65536

(i) printf("%d", 40000);

Output
-25536;

Rough.
40000 % 65536
= 40000 - 65536;
= -25536;

(ii) Printf ("%u", -1);

Output

65535.

Rough

-1 %u 65536

= -1 + 65536

= 65535.

CHAPTER-5

CONTROL STATEMENT.

(1) Conditional / composison / selection.
- (a). if - else Structure,
- (b) nested if ~~else~~.else
- (c) switch case
- (d) nested switch case.
- (e) Ternary Operator.

(2) Repetition / iteration / looping structure
- (a) while
- (b) for
- (c) do while.

(3) Jump Statement:-
- (a) goto
- (b) gotoxy
- (c) break
- (d) continue
- (e) exit()
- (f) return.

(1) (a) if- else structure:- An if- else structure is used for execute sequence of statement based on specified condition as a following format:-

syntax:-

```
if(condition)
{
   {sequence - of - stmt};
   =
}
else
{
  {sequence - of - stmt};
```

Point to be noted down.

(1) In c language, a non-zero (i.e. greater than 0 or less than zero) value treated as true and zero treated as false.

(2) When condition met true then if part will be execute otherwise else part will be execute.

(3) Open({) and close(}) curly braces are optional when it is required to enclosed singe statement within if or else part.

(4) else part is also optional, it can be used if needed.

Explanation - 1

(1)

```
void main()
{
    int a=5;
    clrscr();
      if(a)
        printf("man");
      else
        Printf("go");
    Printf("soon");
    getch();
}
```

Output

Mansoon.

Explanation - 2

```
void main()
{
  int a=5;
  clrscr();
  if(!a)
    printf("man");
  else
    printf("go");
  printf("soon");
  getch();
}
```

Output

gosoon.

Explanation - 3.

```
void main()
{
  int a=5;
  clrscr();
  if(a>5)
    printf("man");
    printf("go");
  printf("soon");
  getch();
}
```

output

gosoon.

Explanation -4.

```
void main()
{
int a=5;
clrscr();
if(a>5)
{
printf("man");
printf("go");
}
printf("soon");
getch();
}
```

Output

soon.

Explanation - 5.

```
void main()
{
int a=5;
clrscr();
if(a>5)
printf("man");
printf("go");
else.
printf("soon");
getch();
}
```

Output

~~soon~~. Error - Misplaced else.

NOTE:- An outer statement must not be enclosed between ~~within~~ if or else part otherwise compiler leads to an error i.e; Misplaced else.

Explanation-6

```
void main()
{
  int a=5;
  clrscr();
    if (a>5);
      printf("man");
  else
    Printf("soon");
  getch();
}
```

Output

Error - Misplaced else.

Explanation - 7.

```
void main()
{
  int a=5;
  clrscr();
    if (a>0)
      Printf("man");
  else;
    Printf("soon");
  getch();
}
```

Output

Man soon.

classmate
Date
Page 126

Ans: 26/08/15

Explanation-8.

```
void main()
{
  int a=5;
  clrscr();
  if (a=0,1,2)
     printf("true and a=%d", a);
else
     printf("false and a=%d", a);
getch();
}
```

Stack region

a
[20]
65524

Output

False and a=20.

True and a = 0

Explanation-9.

```
void main()
{
  int a=5;
  clrscr();
  if (a=(0,1,2))
     printf("true and a=%d", a);
else
     printf("false and a=%d", a);
getch();
}
```

Output

true and a=2

26/08/16.

Explanation - 10.

```
void main()
{
    float a=0.7;
    clrscr();
      if (a==0.7)
          printf("India");
    else
      printf("Bihar");
    getch();
}
```

Output

~~Bihar~~ India ~~loading~~ Bihar

Explanation - 11

```
void main()
{
  float a=0.7;
  clrscr();
     if (a==0.7F)
        printf("India");
  else
     printf("Bihar");
  getch();
}
```

Output

India.

Explanation - 12.

```
void main()
{
    float a = 5.0;
    clrscr();
    if (a == 5.0)
        printf("India");
    else
        printf("Bihar");
    getch();
}
```

Output

India.

ALL

Explanation 13.

```
void main()
{
    int a = 10;
    clrscr();
    if (a>10 && (a==20))
        printf("true and a = %d", a);
    else
        printf("False and a = %d", a);
    getch();
}
```

Output

False and a = 10.

Explanation 14.

```
void main()
{
  int a=10;
  clrscr();
  if (a>=10 && (a=20))
    Printf("true and a = %d", a);
  else
    Printf("False and a = %d", a);
  getch();
}
```

Output

true and a = 20.

Ans.

Explanation 15.

```
void main()
{
  int a=10;
  clrscr();
  if (a>5 || (a=20))
    Printf("true and a = %d", a);
  else
    Printf("False and a = %d", a);
  getch();
}
```

Output

true and a = 10.

APPLICATION BASED ON if - else STRUCTURE.

1. WAP in 'C' to accept two no. from keyboard check and display highest no.

Soln:

```
void main()
{
    int a, b;
    clrscr();
    printf("Enter two no:");
    scanf("%d %d", &a, &b);
if (a>b)
    printf("Highest no: %d", a);
else
    printf("Highest no: %d", b);
getch();
}
```

Output

Enter two no: 5 10 ↵

Highest no: 10.

2. WAP in 'C' to accept a number from keyboard check and display a message whether given number is even or odd.

Soln:

```
void main()
{
    int n;
    clrscr();
    printf("Enter a no:");
    scanf("%d", &n);
if (n%2==0)
    printf("Given no: %d: is even!", n);
```

```
else
    Printf (" Given no: %d is odd!",n);
    system (" pause");
}
}
```

Output

Enter a no: 5
Given no: 5 is odd.

3. WAP in 'C' to accept a character from keyboard check and display a message whether given character is capital letter, small letter, dig or other character.

Character		Ascii
A-Z	=>	65-90
small a-z	=>	97-122
0-9	=>	48-57

```
Soln:  void main()
{
    Char ch;
    clrscr();
    Printf (" enter a character:");
    Scanf ("%c", &ch);
    if (ch>=65 && ch<=90)
    Printf (" capital letter!");
    else if (ch>=97 && ch<=122)
        Printf (" small letter!");
    else if (ch>=48 && ch<=57)
    Printf (" digit!");
    else
```

```
printf("other character!");
getch();
}
```

Output

Enter a character: 2 d

Digit!

(b) nested If else:n

When an if else structure enclosed within another if else structure then structure is known as nested if else structure.

Syntax:n

```
if (condition)
    if (condition)
{
    (sequence of statement);
}
    else if (condition)
            (sequence of statement);
        else (sequence of statement);
    else if (condition)
            if (condition)
                (sequence of statement);
            else
                (sequence of statement);
    else
        (sequence of statement);
```

APPLICATION BASED ON nested if else structur

1) WAP in C to accept three no. from keyboa check and display highest no.

Solution

```
void main()
{
    int a,b,c;
    clrscr();
    printf("enter three no");
    scanf("%d%d%d",&a,&b,&c);
    if (a>b && a>c)
        printf("Highest no: %d",a);
    else if (b>a && b>c)
    printf("Highest no: %d",b)
    else
        print("Highest no: %d",c);
    getch();
}
```

2nd Method

```
void main()
{
    int a,b,c;
    clrscr();
    printf("enter three no.");
    scanf("%d %d %d",&a,&b,&c);
if (a>b)
    if (a>c)
        printf("Highest no: %d",a);
    else
        printf("Highest no: %d",c);
```

```
else if (b>c)
        printf("Highest no: %d", b);
    else
        printf("Highest no: %d", c);
getch();
}
```

2. WAP in 'C' to accept four no. from keyboard check and display ~~last~~ least no.

soln:

```
void main()
{
    int a, b, c, d;
    clrscr();
    printf("enter four no:");
    scanf("%d %d %d %d", &a, &b, &c, &d);
    printf("least no:");
    if (a<b)
        if (a<c)
            if (a<d)
                printf("%d", a);
            else
                printf("%d", d);
        else if (c<d)
            printf("%d", c);
        else
            printf("%d", d);
    else if (b<c)
        if (b<d)
            printf("%d", b);
        else
            printf("%d", d);
```

```
else if (c<d)
        printf ("%d", c);
    else
        printf("%d", d);
getch();
}
```

Ans:

3. WAP in 'c' to accept three no from keyboard check and display middle no.

Solution

```
void main()
{
int a, b, c;
clrscr();
    printf("enter three no:");
    scanf("%d %d %d", &a, &b, &c);
if (a>b)
    if (b>c)
        printf(middle no: %d", b);
    else if (c>a)
            printf(" middle no: %d", a);
        else
            printf(" Middle no: %d", c);
else if (a>c)
        printf(" Middle no: %d", a);
    else if (c>b)
            printf(" Middle no: %d", b);
        else
            printf(" Middle no: %d", c);
getch();
}
```

Date
Page 136

4. WAP in C to accept a no. From keyboard check and display a message whether given no: is positive even, positive odd or negative no.

Soln:

```
void main()
{
int n;
clrscr();
printf("enter a no:");
scanf("%d",&n);
if (n>0 && n%2==0)
printf("positive even!");
else if (n>0 && n%2!=0)
printf("positive odd!");
else
printf("Negative!");
getch();
}
```

2nd Method

```
void main()
{
int n;
clrscr();
Printf("enter ano:");
scanf("%d",&n);
if (n>0)
if (n%2==0)
Printf("Positive even!");
else
Printf("positive odd!");
```

```
else if (n<0)
    printf ("Negative!");
else
    printf ("zero!");
getch();
}
```

5. WAP in 'C' to accept year from keyboard check and display message whethers given year is leap year or not.

Soln:

```
void main()
{
    int year;
    clrscr();
    printf ("enter a year:");
    scanf ("%d", &year);
    printf ("given year: %d: is:", year);
    if (year %100 == 0)
    if (year %400 == 0 && year %4 == 0)
        printf ("leap year!");
    else
        printf ("not leap year!");
    else if (year %4 == 0)
        printf ("leap year!");
    else
        printf ("not leap year!");
    getch();
}
```

Output

Enter a year = 1900 N

given year: 1900 is: not leap year!

// Later

6. WAP in C to accept item code, unit Price and total no quantity from keyboard. calculate and display invoice or bill where 10% discount available on an item and a customer must be paid 4% tax when total amount excede 12000.

Soln:

```
void main()
{
    int icode, qty;
    float uprice, tot, dist, tax, pamt;
    clrscr();
    printf("Enter item code:");
    scanf("%d", &icode);
if (icode > 0)
{
printf("Enter unit price:");
scanf("%f", &uprice);
if (uprice > 0)
{
printf("Enter total quantity:");
scanf("%d", &qty);
if (qty > 0)
{
    tot = uprice * qty;
    dist = tot * 10/100;
if (tot > 12000)
    tax = tot * 4/100;
else
    tax = 0;
pamt = tot - dist + tax;
clrscr();
printf("========== Invoice Details ==========");
```

```
printf("\n Item code : %d", code);
printf("\n unit price : %f", uprice);
printf("\n Total quantity : %d", qty);
printf("\n * * * * * * * * * * *");
printf("\n Total amount : %f", tot);
printf("\n Discount amount : %f", dist);
printf("\n tax : %f", tax);
printf("\n * * * * * * * * * * *");
printf("\n payable amount : %f", pamt);
}// qty
else
    printf("\n Quantity must be > 0!");
}// uprice
else
    printf("\n unit price must be > 0!");
}// icode
else
    printf("\n Invalid item code, try again!");
getch();
}
```

7. WAP in 'C' to accept a character from keyboard and display its Ascii value along with chara

Soln:

```
void main()
{
    char ch;
    clrscr();
    printf("enter a character:");
    scanf("%c", &ch);
    printf("%d :: %c", ch, ch);
getch();
```

Output: Enter a character : A

8. WAP in 'C' to display Ascii value along with character until user pressed any key.

Soln:-

```
void main()
{
    int i=0;
    clrscr();
        while (!kbhit())
    {
    printf("Ascii OF: %c is: %d\n", i, i);
    i++;
    delay(1000);
    }
    // getch();
}
```

Output:-

```
Ascii OF: 125: 0
Ascii OF: is: 1
Ascii OF: 0: n: 98
```

9. WAP in C to display Ascii value of any inputed character such as alphabate, digits Functions key, navigation key, combination key etc.

Soln:-

```
void main()
{
    char ch1, ch2;
    clrscr();
        printf("enter a character:");
    ch1 = getch();
    if (ch1==0)
```

```
    ch2 = getch();
    printf("\n Ascii of %c is: %d %d", ch1, ch2);
  }
  else
    printf(" Ascii of %c is: %d", ch1, ch1);
  getch();
}
```

Ternary operator:-

A ternary operator (i.e; ? :) is specially use for replace of else structure as a following formate;-

Syntax:-

```
EXP1 ? EXP2 : EXP3;
```

NOTE:- EXP1 evaluates and if condition met true then EXP2 will be execute otherwise EXP3 will be execute

Example.

1. WAP in 'c' to accept two no. from keyboard check and display highest number by using ternary operator.

```
void main()
{
  int a, b;
  clrscr();
  printf("enter two no:");
  scanf("%d %d", &a, &b);
a>b? printf("Highest no: %d", a): printf("Highest no: %d", b)
  getch();
}
```

Output

Enter two no: 10 58

```
    ch2 = getch();
    Printf("\n Ascii is: %d %d", ch1, ch2);
  }
  else
    Printf(" Ascii OF : %c, is: %d", ch1, ch1);
  getch();
}
```

Ternary operator:-

A ternary operator (i.e; ?:) is specially use for replace of else structure as a following formate:-

Syntax:-

```
EXP1 ? EXP2: EXP3;
```

NOTE:- EXP1 evaluates and if condition met true then EXP2 will be execute otherwise EXP3 will be execute

Example:-

1. WAP in 'c' to accept two no. From keyboard check and display highest number by using ternary operator.

```
Sol:- void main()
{
  int a, b;
  clrscr();
  Printf(" enter two no:");
  scanf(" %d %d", &a, &b);
a>b? Printf(" Highest no: %d", a): printf(" Highest no: %d", b)
  getch();
}
```

Output

Enter two no: 10 5

2. WAP in 'c' to accept three no. From keyboard check and display highest no. by using ternary operators.

Soln:-

```
void main()
{
int a, b, c, h;
clrscr();
  printf("enter three no:");
  scanf("%d %d %d", &a, &b, &c);
h = a>b? a>c? a:c: b>c? b:c;
printf("Highest no: %d", h);
getch();
}
```

Output

Enter three no: 10 15 5 N

Highest no: 15

3. WAP in 'c' to accept a characters from keyboard check and display a message wheather given characters is capital letter, small letter or other characters.

Soln:-

```
void main()
{
  char ch;
  clrscr();
    printf("enter a characters:");
    scanf("%c", &ch);
(ch)=65 && ch<=90)? printf("capital letters"): (ch)=97
&& ch<=122)? printf("small letter"): (ch)=48 && ch<=57);
```

getch();
}

Output
Enter a character 2
Digit.

Switch case:-

Switch case is specially supported for simplify nested - if - else structure as a following formate.

Syntax:-

```
switch(var/ expr/ constant)
{
  case constant: {sequence - of - statement};
  ---
  ---
  {break;}
  case constant: {sequence - of - statement};
  ---
  ---
  {break;}
  ---
  ---
  {default: {sequence - of - statement};}
  ---
  ---
}
```

optional

optional

1. In the parenthesis of switch an integer type variable, expression and constant are allowed.

2. Case reserved word only allowed constant valu

3. Break is optional it can be used when it is required to stop continue execution and resume execution after switch case.

4. Default statement is also optional.

Explanation:-1

1. WAP in 'C' to accept product no. From keyboard and display discount message as a following given chart.

Product no:	Discount.
1	10%
2	15%
3	20%

Soln:-

```
void main()
{
    int pno;
    clrscr()
    printf("enter product no. as 1/2/3:");
    scanf("%d", &pno);
  switch (pno)
  {
  case 1: printf("10% discount available!");
  case 2: printf("15% discount available!");
  case 3: printf("20% discount available!");
  }
  printf("\n Execution complete!");
  getch();
}
```

Output

Enter product no. as 1/2/3:2
15% discount available!
20% discount available!
Execution complete!
Enter product no. as 1/2/3 :4
Execution complete!

2nd Method.

```
    void main()
{
    int pno;
    clrscr();
      printf("Enter product no. as 1/2/3:");
      scanf("%d",&pno);
   switch(pno)
   {
    case1: printf("10% discount available!");
           break;
    case2: printf("15% discount available!");
           break;
    case3: printf("20% discount available!");
           break;
    default: printf("Invalid product no. tryagain!");
   }
    printf("\n Execution complete!");
    getch();
}
```

OUTPUT

Enter product no. as 1/2/3: 2N
15% discount available!
Execution complete!
Enter product no. as 1/2/3: 4N
Invalid product no. try again!
Execution complete!

Explanation – 2.

```
void main()
{
    int x = 0;
    clrscr();
    switch (x+1)
    {
     case 0: x = x+5;
     case 1: x = x+3;
     case 2: x = --x;
     default: x = x+2;
    }
     printf("\n x = %d", x);
     getch();
}
```

OUTPUT

x = 0.

Explanation - 3.

```
void main()
{
  int i=1, j=2;
  clrscr();
    switch(i+j)
    {
      case 1: printf("wel");
      case 3: printf("done");
    }
  getch();
}
```

(a) wel (b) done

(c) weldone (d) error.

Explanation 4.

1. WAP in 'C' to accept month no. from keyboard and display it's corresponding season name as a following given chart.

Month no.	Season.
11, 12, 1, 2	winter
3, 4, 5	Spring.
5, 6, 7	Summer
8, 9, 10	Rainy.

Soln:

```
void main()
{
  int mno;
  clrscr();
  Printf("enter month no:");
```

```
scanf("%d", &mno);
switch(mno)
{
case 11:
case 12:
case 1:
case 2: printf("winter season!");
        break;
case 3:
case 4: printf("spring season!");
        break;
case 5:
case 6:
case 7: printf("summer season!");
        break;
case 8:
case 9:
case 10: printf("Rainy season!");
        break;
default: printf("invalid month no, try again!");
}
getch();
}
```

OUTPUT

Enter month no: 6 ↵

Summer season!

Enter month no: 13 ↵

Invalid month no, try again!

Explanation -5

A default statement may be anywhere in the curlie brace of switch.

```
Ex:-
    void main()
{
    int a;
    clrscr();
     Printf (" enter a no: ");
     Scanf (" %d", &a);
  Switch(a).
  {
   default: printf ("wipro");
            break;
   case1: printf (" Infosys: ");
          break;
   case2: Printf (" Microsoft ");
  }
   getch();
}
```

OUTPUT

Enter a no: 2

Microsoft

Enter a no: 5

wipro.

Explanation - 6

1. WAP in 'C' to accept publishers name from Keyboard and display discount message offes by publishers as a following given chart.

Publishers name	Discount
BPB (B/b)	10%
TNH (T/t)	15%
Pearson (P/p)	20%

Soln:

```
void main()
{
    char Pname;
    clrscr();
    printf("Enter publishers name as BPB(B/b), TNH(T/t), Pearson(P/p):");
    scanf("%c", &Pname);
    switch(Pname)
    {
    case 'B':
    case 'b': printf("10% discount offes!"); break;
    case 'T':
    case 't': printf("15% discount offes!"); break;
    case 'P':
    case 'p': printf("20% discount offes!"); break;
    default: printf("Invalid publishers name, try again!");
    }
    getch();
}
```

OUTPUT

```
Enter publishers name as BPB(B/b), TNH(T/t), Pearson(P/p): P
```

2. WAP in 'C' to accept a no. From keyboard check and display a message whether given no. is positive even, positive odd, negative or zero.

Soln:-

```
void main()
{
int n;
clrscr();
Printf("Enter a no:");
scanf("%d",&n);
switch(n>0)
{
case 1:
   switch(n%2==0)
   {
   case 1: Printf("Even!");
           break;
   default: Printf("odd!");
   }
   break;
default: switch(n<0)
         {
         case 1: Printf("Negative!");
                 break;
         default: printf("zero!");
         }
}
getch();
}
```

OUTPUT

Enter a no: 9

Odd!

Enter a no: -5

Negative!

2. Looping Structure:-

Looping structure are used for repeate sequence of statement until condition met false.

The C language supports following three types of looping structure. these are:-

(1) while } Pretst loop
(2) For }
(3) do while → posttest loop.

1. while :-

Looping structure is used as a following format to repeate sequence of statement until condition met false.

Syntax:-

```
while (condition)
{
  {sequenu - of - stmt};
  ——
  {Modify looping driven variable};
}
```

NOTE:-

1. In looping structure, a non-zero ~~value~~ (i.e >0 or value treated as true and zero treated as False

2. Open ({) and close (}) curlie braces are optional when it is required to enclosed single statement within body of loop.

3. A looping driven variable must be modify to stop repetition at certain period otherwise execution will be infinite.

Explanation 1.

```
void main()
{
    int i=1;
    clrscr();
    while(i<=3)
    {
     printf("loop executes:%d: times!\n",i);
     i++;
    }
  printf("execution complete!");
  getch();
}
```

OUTPUT

Dry Run

i	i<=3	OUTPUT
1	1<=3 ✓	loop executes:1: times!
2	2<=3 ✓	loop executes:2: times!
3	3<=3 ✓	loop executes:3: times!
4	4<=3 ✗	Execution complete!

Explanation - 2.

```
void main()
{
    int i = 0;
    clrscr();
    while (++i <= 3)
        printf("Loop executes : %d : times ! \n", i);
    printf("Execution complete!");
    getch();
}
```

DRY RUN

i	++i <= 3	OUTPUT.
0	1 <= 3 ✓	Loop executes : 1 : times!
1	2 <= 3 ✓	Loop executes : 2 : times!
2	3 <= 3 ✓	Loop executes : 3 : times!
3		
4	4 <= 3 ✗	Execution complete!

Explanation - 3.

1. WAP in 'C' to display statement as a following formate

(a) 10
9
8
7
6
5
4
3
2
1
0

(b) 1 4 9 16 - - - - upto 100.

Soln:- (a)

```
void main()
{
    int i = 10;
    clrscr();
    while (i >= 0)
    {
        printf("%d\n", i);
        i--;
    }
    printf("Blast off!");
    getch();
}
```

(b)

```
void main()
{
    int i = 1, k = 1;
    clrscr();
    while (k <= 100)
    {
        k = i * i;
        printf("%d", k);
        i++;
    }
    getch();
}
```

Dry Run

i	k	k<=100	output
1	1	1<=100	1 4 9 16 25 ------
2	1	1<=100	
3	4	4<=100	
4	9	9<=100	
5	16	16<=100	
	25	25<=100	

2. WAP in 'C' to accept a no. From keyboard , calculate and display it's table as a following Format:-

Hint:- 5*1 = 5
5*2 = 10
5*3 = 15
=
5*10 = 50.

Soln:-

```
void main()
{
    int i=1, n;
    clrscr();
    printf("Enter a no. for table:");
    scanf("%d", &n);
    while (i<=10)
    {
    printf("%d*%d = %d\n", n, i, n*i);
    i++;
    }
    getch();
}
```

Dry Run

n	i	i<=10	OUTPUT.
5	1	1<=10 ✓	Enter a no. for table:5
	2	2<=10 ✓	5*1=5
			5*2=10
			...

3. WAP in 'C' to accept a no. From keyboard, and display its reverse.

Soln:-

```
void main()
{
    int n, d, rev=0;
    clrscr();
    printf("Enter a no.");
    scanf("%d", &n);
    while(n>0)
    {
    d = n%10;
    rev = rev * 10 + d;
    n = n/10;
    }
    printf("\n reverse no: %d", rev);
    getch();
}
```

Dry Run

n	d	n>0	rev	OUTPUT
245	5	245>0 ✓	=0*10+5	Enter no: 245
24	4	24>0 ✓	=5	reverse no: 542
2	2	0>0 ✗	=5*10+4	
0			=54	
			=54*10+2	
			542	

4. WAP in 'c' to accept N no's from keyboard, check and display total count of even and odd no.

Soln:

```
void main()
{
    int n, i=1, num, even=0, odd=0;
    clrscr();
        printf("How many no's u want:");
        scanf("%d", &n);
        printf("Enter your no's:");
    while(i<=n)
    {
        scanf("%d", &num);
        if(num%2==0)
            even++;
        else
            odd++;
        i++;
    }
    printf("\n Total even = %d \n Total odd = %d", even, odd);
    getch();
}
```

Dry run

n	i	i<=n	num	even	odd	OUTPUT.
5	1	1<=5 ✓	3	0	0	How many no u want: 5
	2	2<=5 ✓	7	1	1	Enter your no's: 3, 7, 8, 6, 9
	3	3<=5 ✓	8	2	2	
	4	4<=5 ✓	6		3	
	5	5<=5 ✓	9			Total even = 2.
	6	6<=5 ✗				Total odd = 3.

Date 159

5. WAP in 'C' to accept N no's From keyboard check and display highest and least no.

soln:-

```
void main()
{
int n, i=1, num, gt, lt;
clrscr();
    Printf("How many no's u want:");
    Scanf("%d", &n);
Printf("Enter your no's:");
while(i<=n)
{
scanf("%d", &num);
if(i==1)
    gt = lt = num;
else if(gt<num)
    gt = num;
else if(lt>num)
    lt = num;
i++;
}
printf("\n greatest no: %d\n least no: %d", gt,
getch();
}
```

Dry Run / Traversing.

n	i	i<=n	num	gt	lt	OUTPUT
5	1	1<=5 ✓	8	8	8	How many no
	2	2<=5 ✓	7	10	7	u want: 5
	3	3<=5 ✓	10		2	Enter your no's
	4	4<=5 ✓	2			8 7 10
	5	5<=5 ✓	10			greatest no: 10
	6	6<=5 ✗				least no: 2

Date
Page 160

6. WAP in 'C' to accept a no. from keyboard check and display a message whether given no. is prime or composite no.

Soln:-

```
Void main()
{
    int n, d=2;
    clrscr();
        printf("Enter a no:");
        scanf("%d", &n);
    while (d<=n/2)
    {
      if (n%d==0)
        d=n;
     d++;
    }
    if (n==1)
        printf("\n Neither prime nor composite!");
    else if (d>n)
        printf("\n given no: %d: is composite!", n);
    else
        printf("\n given no: %d: is prime!", n);
    getch();
}
```

Dry Run

n	d	d<=n/2	OUTPUT
1	2	2<=0 ×	Enter a no:1
			Neither prime nos composite!
2	2	2<=1 ×	Enter a no:2
			given no:2: is prime!
7	2	2<=3✓	Enter a no:7
	3	3<=3✓	given no:7: is prime!
	4		

12	2	2<=6	Enter a no:12
	+2	13<=6	given no:12: is composite!
	13		

7. WAP in 'C' to accept a no: From keyboard check and display a message whether given no: is perfect or not.

Soln:-

```
void main()
{
    int n, d=1, sum=0;
  clrscr();
    printf("Enter a no:");
    scanf("%d", &n);
  while(d<=n/2)
  {
    if(n%d==0)
        sum= sum+d;
    d++;
  }
  if(sum==n)
    printf("\n given no:%d: is perfect!", n);
  else
    printf("\n given no:%d: is not perfect!", n);
  getch();
}
```

8. WAP in 'c' to accept a three digit no. from keyboard check and display a message whether given no. is armstrong or not.

Soln:-

```
void main()
{
int n, d, arm=0, temp;
clrscr();
printf("Enter a three digit no:");
scanf("%d",&n);
temp=n;
while(n>0)
{
d=n%10;
arm=arm+d*d*d;
n=n/10;
}
if(temp==arm)
printf("given no:%d is armstrong!",temp);
else
printf("given no:%d is not armstrong!",tem,
getch();
}
```

Traversing

n	temp	n>0	d	arm	OUTPUT
~~153~~	153	153>0 ✓	3	=0+3*3*3	Enter a three
15		15>0 ✓	5	=27	digit no. 153
1		1>0 ✓	1	=27+5*5*5	given no: 153 is
0		0>0 ✗		=152	armstrong.
				=152+1*1*1	
				=153	

(b) FOR:-

Looping structure is used as a following format to repeate sequence of statement until condition met false.

SYNTAX:-

```
FOR(<initilisation stmt>---;<conditional stmt>;<iterat
                                        stmt>,---).
{
   <sequence - OF - stmt>;
   ==
}
```

NOTE:-

1. In FOR LOOP, initilisation statement and iteratio statement may be more than one but conditional statement is only one allowed.

2. All initilisation, conditional and iteration statement are optional but semicolon(;) is required.

3. Open({) and close(}) curlie brous are optional when it is required to enclosed single statemen within body of for loop.

4.

STRUCTURE - 1.

When only one initialisation, conditional and iteration statement are available.

```
void main()
{
    int i;
    clrscr();
    for (i=1; i<=3; i++)
        printf("Loop executes: %d: times: \n", i);
    printf("\n Execution complete!");
    getch();
}
```

Dry Run

i	i<=3	OUTPUT
1	1<=3 ✓	Loop executes: 1: times!
2	2<=3 ✓	Loop executes: 2: times!
3	3<=3 ✓	Loop executes: 3: times!
4	4<=3 ✗	Execution compute!

STRUCTURE - 2.

When more than one initialisation and iteration statement are available.

```
void main()
{
    int i, j;
    clrscr();
    for (i=1, j=3; i<=j; i++, j--)
    {
        printf("i = %d", i);
```

```
        Printf("\nJ = %d\n", J);
    }
    Printf("Execution Complete!");
    getch();
}
```

Dry Run

i	J	i<=J	OUTPUT
1	3	1<=3 ✓	i=1
2	2	2<=2 ✓	J=3
3	1	3<=1 ✗	i=2
			J=2
			Execution complete!

STRUCTURE - 3.

When only conditional statement available.

```
void main()
{
    int i=1;
    clrscr();
    for(; ++i<=3;)
        Printf("%d", i);
    Printf("\n Rays Edutech Pvt. Ltd");
    getch();
}
```

Dry Run

i	++i<=3	OUTPUT
1	2<=3 ✓	2 3
2	3<=3 ✓	Rays Edutech Pvt. Ltd.
3	4<=3 ✗	
4		

STRUCTURE:- 4.

When conditional statement is not available.

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

Dry Run

i	i==3	OUTPUT
1	1==3 x	i=1
2	2==3 x	i=2
3	3==3 ✓	i=3

STRUCTURE -5.

Dummy FOR:- A FOR loop terminated by semicolon(;) without body is known as dummy FOR.

Syntax:-

```
for (<initilisation stmt>,---; <conditional stmt>;
     <iteration stmt>---);
```

Example:-

Date
Page 167

1. WAP in 'c' to accept a no. From keyboard check and display total count of digit.

Soln:-

```
void main()
{
    int i,n;
    clrscr();
    printf("Enter a no:");
    scanf("%d",&n);
    for(i=0; n>0; i++, n=n/10);
    printf("\n Total digit: %d", i);
    getch();
}
```

Dry Run

n	i	n>0	OUTPUT
245	0	245>0 ✓	Enter a no: 245
24	1	24>0 ✓	Total digit: 3.
2	2	2>0 ✓	
0	3	0>0 ✗	

2. WAP in 'c' to accept a no. From keyboard calculate and display its Factorial.

Soln:-

```
void main()
{
    int n, fact;
    clrscr();
    printf("Enter a no:");
    scanf("%d",&n);
    for(fact=1; n>0; fact=fact*n, n--);
    printf("\n Factorial = %d", fact);
    getch();
}
```

1 -

Dry Run

n	n>0	Fact	OUTPUT
4	4>0 ✓	1	Enter a no: 4
3	3>0 ✓	=1*4	Factorial = 24
2	2>0 ✓	=4*3	
1	1>0 ✓	=12*2	
0	0>0 ✗	=24*1	
		=24	

3. WAP in 'C' to accept a no. From keyboard and display sum of it's even alternate digit

Soln:-

```
void main()
{
    long int n;
    int sum=0;
    clrscr();
    printf("Enter a no:");
    scanf("%ld",&n);
for(;n>0;d=(n%10)*10,sum=d,n=n/100);
printf("\n sum of even alternate digit=%d", sum);
getch();
}
```

Dry Run

n	d	n>0	sum	OUTPUT
413725	2	413725>0 ✓	0	Enter a no: 413725
4137	3	4137>0 ✓	0+2	Sum of even alternate
41	7	41>0 ✓	2+3	digit = 9.
0		0>0 ✗	5+4	
			=9	

4. WAP in 'C' to display total leap year found between 1000 and 2013.

SOLN:-

```
void main()
{
    int year;
    clrscr();
    printf("Leap year are:\n");
    for(year=1000; year<=2013; year++)
    {
        if(year%100==0)
        {
            if(year%400==0 && year%4==0)
                printf("%d\n", year);
        }
        else if(year%4==0)
            printf("%d\n", year);
    }
    getch();
}
```

OUTPUT

Leap year are:

1004
1008
1012
1016
1020
1024
1028
1032
1036
1040
1044
1048
1052
1056
1060
1064
1068
1072
1076
1080
1084
1088
1092
1096
1104
1108
1112
1116
1120
1124
1128
1132
1136
1140
1144
1148
1152
1156
1160
1164
1168
1172
1176
1180
1184
1188
1192
1196
1204
1208
1212
1216
1220
1224
1228
1232
1236
1240
1244
1248
1252
1256
1260
1264
1268
1272
1276
1280
1284
1288
1292
1296
1304
1308
1312
1316
1320
1324
1328
1332
1336
1340
1344
1348
1352
1356
1360
1364
1368
1372
1376
1380
1384
1388
1392
1396
1404
1408
1412
1416
1420
1424
1428
1432
1436
1440
1444
1448
1452
1456
1460
1464
1468
1472
1476
1480
1484
1488
1492
1496
1504
1508
1512
1516
1520
1524
1528
1532
1536
1540
1544
1548
1552
1556
1560
1564
1568
1572
1576
1580
1584
1588
1592
1596
1604
1608
1612
1616
1620
1624
1628
1632
1636
1640
1644
1648
1652
1656
1660
1664
1668
1672
1676
1680
1684
1688
1692
1696
1704
1708
1712
1716
1720
1724
1728
1732
1736
1740
1744
1748
1752
1756
1760
1764
1768
1772
1776
1780
1784
1788
1792
1796
1804
1808
1812
1816
1820
1824
1828
1832
1836
1840
1844
1848
1852
1856
1860
1864
1868
1872
1876
1880
1884
1888
1892
1896
1904
1908
1912
1916
1920
1924
1928
1932
1936
1940
1944
1948
1952
1956
1960
1964
1968
1972
1976
1980
1984
1988
1992
1996
2000
2004
2008
2012

Date
Page 1760

5. WAP in 'C' to display 10 terms of Fibonanci series.

Hint:- 0 1 1 2 3 5 8 13 - - - - -

```
void main()
{
int a=0, b=1, n;
clrscr();
    printf("Fibonanci series:");
    printf("%d %d", a, b);
for (i=1; i<=8; i++)
{
n=a+b;
a=b;
b=n;
printf("%d", n);
}
getch();
}
```

Dry Run

a	b	n	i	i<=8	OUTPUT
0	1	1	1	1<=8	Fibonanci series: 0 1 1 2 3
1	1	2	2	2<=8	
1	2	3	3	3<=8	
2	3				

6. WAP in 'C' to accept N no's from keyboard check and display total count of prime no and composite no.

Date
Page 171

Soln:-

```
void main()
{
  int n, num, d, pr=0, cm=0;
  clrscr();
  printf("HOW many no's u want:");
  scanf("%d", &n);
while (n>0)
{
  printf("Enter a no:");
  scanf("%d", &num);
  for (d=2; d<=num/2; d++)
      if (num%d==0)
          d=num;
  if (d>num)
      cm++;
  else
      pr++;
  n--;
}
printf("\n Total prime=%d", pr);
printf("\n Total composite=%d", cm);
getch();
}
```

Dry Run

n	num	d	pr	cm	n>0	OUTPUT
3	2	2	0	0	3>0 T	HOW many nos :
2	7	2	1	1	2>0 T	want:3
1	9	3	2		1>0 T	Enter a no:2
0		4			0>0 F	Enter a no:7
		2				Enter a no:9
		3				Total Prime:2
		4				Total compo =
		10				

2. Pow():- Pow is an inbuilt function of Math.h that accept two arguments and it raised first argument by second argument.

Ex:- pow(2,3);

↓ $2^3 = 8$

(8.0)

↓

Drop result in double

Ex:- printf("%d", pow(2,3));

OUTPUT

0

Ex:- printf("%.1f", pow(2,3));

OUTPUT

8.0.

7. WAP in 'c' to accept a no. From keyboard check and display a message wheather given no. is armstrong or not.

soln:-

```
#include<Math.h>
void main()
{
    int n, temp, arg=0, d, i;
    clrscr();
    printf("Enter a no:");
    scanf("%d", &n);
    temp=n;
    for(i=0; n>0; i++, n=n/10);
    for(n=temp; n>0; n=n/10);
```

```
    d = n % 10;
    arg = arg + pow(d, 3);
}
if (arg == temp)
    printf("given no: %d is armstrong!", temp
else
    printf("given no: %d is not armstrong!", te
getch();
}
```

Assinatura

Data

Page 174

algorithm

Date

Page 175

9. WAP in 'C' to calculate and display the sum of the given series.

$1 + x^2 + x^3 + x^4 + \dots$ upto N terms.

Soln:-

```
#include (Math.h)
void main()
{
int n, x, sum=1, i=2;
clrscr();
Printf("How many terms u want:");
Scanf("%d", &n);
Printf("Enter a value:");
Scanf("%d", &x);
while (i<=n)
{
sum = sum + POW(x, i);
i++;
}
printf("sum of series, %d", sum);
getch();
}
```

Dry Run

n	x	i	i<=n	sum	OUTPUT
5	3	2	2<=5 ✓	1	How many terms u
		3	3<=5 ✓	=1+9	want:5
		4	4<=5 ✓	=10+27	Enter a value:3
		5	5<=5 ✓	=37+81	sum of series:361.
		6	6<=5 ✗	=118+243	
				=361	

STRUCTURE ( ):-

2.(b) FOR(;;):-

this structure by default created condition true and it is well suited for menu driven programe.

Syntax:-

```
FOR(;;)
{
<sequence-of-stmt);
==
}
```

Ex:-

```
void main ( )
{
int n, i; char choice;
FOR(;;)
{
clrscr( );
Printf("==========MENU==========");
Printf("\n Press (F) FOR Factorial application!'
Printf("\n Press (c) For count digit application!
Printf("\n Press (P) For prime application!");
Printf("\n Press (x) For exit application!");
Printf("\n * * * * * * * * * * * *");
Printf("\n Enter your choice:\?\b");
choice = getch( );
clrscr( );
Switch(choice)
{
case 'F':
```

```
case 'f': printf("Enter a no:");
          scanf("%d",&n);
          for(i=1; n>0; i=i*n, n--);
          printf("Factorial = %d", i);
          break;
case 'c':
case 'c': printf("Enter a no:");
          scanf("%d",&n);
          for(i=0; n>0; i++, n=n/10);
          printf("\n Total digit = %d", i);
          break;
case 'p':
case 'p': printf("Enter a no:");
          scanf("%d",&n);
          for(i=2; i<=n/2; i++)
              if(n%i==0)
                  i=n;
          if(i>n)
              printf("given no: %d: is composite!", n);
          else
              printf("given no: %d: is prime!", n);
          break;
case 'x':
case 'x': exit(0);
default: printf("Invalid choice. try again!");
}// switch
getch();
}// for
}
```

2(c) do while:-

do while looping structure is known as posttest loop because in this loop condition will be checked in bottom.

Syntax:-

```
do
{
  (sequence - of - stmt);
  =
} while (condition);
```

NOTE:-

1. In do while looping structure, an execution will be perform at least once.

2. Open(f) and close(f) curlie breaces are optional when it is required to enclosed single statement while do while.

3. It is well suited for menu driven program.

Explanation - 1.

```
void main()
{
  int i = 1;
  clrscr();
  do
    printf("Rajiv\n");
  while (++i <= 3);
  getch();
}
```

OUTPUT

Rajiv

Explanation-2.

```
void main()
{
    int n, i; char choice;
    do
    {
     clrscr();
     printf("========== MENU ==========");
     printf("\n press (F) for Factorial application!");
     printf("\n press(C) for count digit-application!");
     printf("\n press(P) for prime application!");
     printf("\n press(x) for exit application!");
     printf("\n * * * * * * * * * *");
     printf("\n Enter your choice:\t\b");
    choice = getch();
    clrscr();
    switch(choice)
    {
     case 'F':
     case'f': printf("enter a no:");
              scanf("%d", &n);
              for(i=1; n>0; i=i*n, n--);
              printf("Factorial = %d", i);
              break;
    case 'C':
    case'c': printf("Enter a no:");
              scanf("%d", &n);
              for(i=0; n>0; i++, n=n/10);
              printf("\n total digit = %d", i);
             break;
    case'P':
    case'p': printf("enter a no:");
```

```
scanf("%d",&n);
for(i=2; i<=n/2; i++)
    if(n%i==0)
        i=n;
    if(i>n)
        printf("given no:%d is composite!",n);
    else
        printf("given no:%d is prime!",n);
    break;
case 'x':
case 'x': choice='x';
    break;
default: printf("invalid choice, try again!");
}//switch
getch();
} while(choice!='x');
}
```

JUMP STATEMENT

The c compiler supports several jump statement, some are:-

1. gotoxy()
2. goto()
3. break()
4. continue()
5. exit()
6. Return()

1 gotoxy():-

Is an inbuilt function that accept two arguement as a x-co-ordinate and y co-ordi and it is responsible to jump cursor a

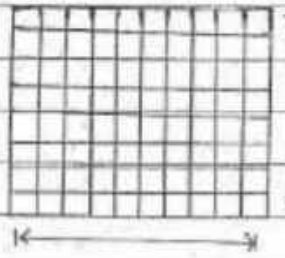
Specified position in current window.

Syntax:- To call gotoxy()

gotoxy (column, row);

column → x co-ordinate

row → y co-ordinate.



Let Assume screen of monitor.

1. WAP in 'C' to display following statements.

Rays	Rays
Education.	
Rays	Rays

Soln:-

```
void main()
{
    clrscr();
    textcolor(2);
    cprintf("Rays");
    sleep(2);
    gotoxy(1,25);
    textcolor(4);
    cprintf("Rays");
    sleep(2);
```

```
gotoxy (76,25);
textcolor (6);
Cprintf ("Rays:");
Sleep (2);
gotoxy (76,1);
textcolor (8);
Cprintf ("Rays");
Sleep (2);
gotoxy (36,13);
textcolor (10+128);
Cprintf ("Education");
getch ();
}
```

2. WAP in 'C' to accept three no. from keyboard and display it's table as a following Formats:-

Hint:-

2*1=2	3*1=3	4*1=4
2*2=4	3*2=6	4*2=8
———	———	———
2*10=20.	3*10=50.	4*10=40.

Soln:-

```
void main ()
{
int i, n, num[3], c=1, r;
clrscr();
printf ("Enter three no's:");
for (i=0; i<3; i++)
scanf ("%d", &num[i]);
clrscr();
for (i=0; i<3; i++, c=c+10)
{
```

```
        n = num[i];
        for (x=1; x<=10; x++)
        {
            gotoxy (c,x);
            printf("%d * %d = %d", n, x, n*x);
        }
    }
    getch();
}
```

Dry Run

i	n	c	x	i<3	OUTPUT.	
0	2	1	1	0<34	2*1=2	3*1=3
1	3	11	2	1<34	2*2=4	3*2=6
			3		2*3=6	3*3=9
			4		=	=
			5		=	=
			6		2*10=20	3*10=30.
			7			
			8			
			9			
			10			
			11			

goto:-

goto is a label jump statement that is used for jump cursor at specified label.

Syntax:- to define a label.

```
label1: <sequence-of-stmt);
   ==
label2: (sequence-of-stmt);
   ==
```

By using goto jump statement we can jump at any specified label as a following format:-

Syntax:- to jump on specified label.

```
goto label name;
```

Ex:-

1. WAP in 'C' to accept a no. from keyboard, chec and display a message wheather given no. is even or odd.

Sol:-

```
void main()
{
  int n;
  clrscr();
    printf("Enter a no:");
    scanf("%d", &n);
  if (n%2==0)
    goto even;
  else
```

```
        goto odd;
    even: printf("given no: %d: is even!", n);
        goto e1;
    odd: printf("given no: %d: is odd!", n);
e1: getch();
}
```

OUTPUT

Enter a no: 8

given no: 8: is even:

Enter a no: 5

given no: 5: is odd:

Q2. WAP in 'c' to accept itemcode, unit price and total quantity from keyboard, calculate and display invoice / bill where 10% discount available on all product and if total amount exceed 12000 then 4% tax must be paid by customer on getting bill.

Soln:-

```
void main()
{
    int icode, qty;
    float uprice, tot, tax, disc, pamt;
    clrscr();
l1: printf("Enter itemcode:");
scanf("%d", &icode);
if (icode <=0)
    goto l1;
l2: printf("Enter unit price:");
scanf("%f", &uprice);
if (uprice <=0)
    goto l2;
```

```
l3; printf("Enter total quantity:");
    scanf("%d", &qty);
    if (qty <=0)
        goto l3;
    tot = uprice * qty;
    dist = tot * 10/100;
    if (tot > 12000)
        tax = tot * 4/100;
    else
        tax=0;
    pamt = tot - dist + tax;
clrscr();
printf("\t\t========== Invoice Details ==========");
printf("\n\t\t Item code : %d", icode);
printf("\n\t\t unit price : %f", uprice);
printf("\n\t\t total quantity: %d", qty);
printf("\n\t\t * * * * * * * * * * *");
printf("\n\t\t total amount : %f", tot);
printf("\n\t\t Discount amount : %f", dist);
printf("\n\t\t tax : %f", tax);
printf("\n\t\t * * * * * * * * *");
printf("\n\t\t payable amount : %f", pamt);
getch();
}
```

3. **Break:-** Jump statement is used for ~~stop~~ control execution of switch or loop and resumed execution just after close (}) write brace of switch or loop.

Examples

1. WAP in 'C' to accept a no. from keyboard, check and display a message whether given no. is positive even or positive odd.

Soln:-

```
void main()
{
  int a;
  clrscr();
   printf("Enter a no:");
   scanf("%d",&a);
 switch (a%2)
 {
 case 0: if (a>0)
          printf("given no:%d is +ve even!",a);
         break;
 case 1: if (a>0)
          printf("given no:%d, is +ve odd!",a);
 }
  printf("\n Execution complete!");
  getch();
}
```

OUTPUT

Enter a no: 8

given no: 8, is +ve even!.

Explanation (2):-

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

4. Continue:-

Continue jump statement only allowed within looping structure and it is responsible to stop continue execution at specified condition and resume execution from top of looping structure.

Ex1. WAP in 'c' to accept five positive no. check and display total count of even or odd.

Soln:-

```
void main()
{
int i=1, even=0, odd=0, n;
clrscr();
    while(i<=5)
    {
    printf("Enter a no:");
```

```
scanf("%d", &n);
if (n<0)
    continue;
if (n%2==0)
    even++;
else
    odd++;
i++;
}
printf("\n Total even = %d \n Total odd = %d",
                                    even, odd);
getch();
}
```

Dry Run

n	i	i<=5	even	odd	OUTPUT
8	1	1<=5	0	0	Enter a no: 8
-4	2	2<=5	1	1	Enter a no: -4
-7	3	2<=5	2	2	——— : -7
3	4	2<=5		3	——— : 3
9	5	3<=5			——— : 9
12	6	4<=5			——— : 12
-12		5<=5			——— : -12
13		5<=5			——— : 13
		6<=5			Total even = 2
					Total Odd = 3.

Explanation-2.

```
void main()
{
    int i=1, even=0, odd=0, n;
    clrscr();
```

```
for(; i<=5; i++)
{
    printf("Enter a no:");
    scanf("%d", &n);
    if(n<0)
    {
        i--;
        continue;
    }
    if(n%2==0)
        even++;
    else
        odd++;
}
printf("\n total even=%d\n total odd=%d", e
                                    .odd
getch();
}
```

5. exit( ):-

exit is an inbuilt function that is responsible to quit / exit application.

Syntax:- To call exit() function.

```
exit(0);
```

Or,

```
exit(1);
```

Ex 1. WAP in 'C' to accept n numbers from keyboard and terminate application when user enter 0 (zero).

Soln:-

```
void main()
{
    int n;
    clrscr();
    printf("Enter no's");
    while(1)
    {
        scanf("%d",&n);
        if(n==0)
            exit(0);
    }
}
```

OUTPUT

Enter no's: 10 5 7 6 7 4 3 1 12 0.

CHAPTER - 6

Date
Page 197

POINTER

A pointer is also a variable that holds and address of another variable or location.

Syntax:- TO define a pointer.

```
datatype *pname;
```

↓

char
int
float
double
user defined datatype

Note:-

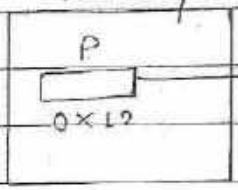
1. A pointer always allocates two byte memory space.
2. A pointer by default considers unsigned.
3. Once pointer holds an address of any local variable then we can access or change a value of that location / variable by using pointer.

Ex:-

(1) `char *p;`

⇓

Memory.



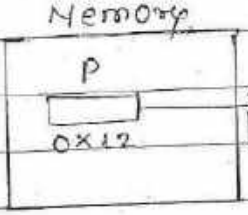
size => 2 byte

type => Address of char type.

2. int *P;

⇓

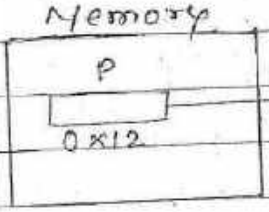
Memory



3. Float *P;

⇓

Memory



NOTE:-

4. A datatype with pointer only ensure type of variable whose address being assigned into pointer.

5. A pointer always allocates two byte unsigned memory space because a memory address range is 0 to 65535.

Syntax:- TO store an address of variables into pointer.

```
Pname = &varname;
```

Ex:- char ch = 'A', *P;
int a=10, *t;
P = &ch; ✓

p = &a; ✗
t = &a; ✓
t = &ch; ✗

Memory

ch	p		
65	0x12		
0x12	0x13		
a	t		
10	0x14		
0x14	0x15		

Syntax:- TO access a value of variable by using pointer.

```
Lvar = * pname;
```

or,

```
printf("Message <Format specifier>", * pname);
```

Ex:-

```
void main()
{
    int a=10, b, *p;
    p = &a;
    b = *p;
      ↓
    b = *(65524);
    b=10;
    printf("value of a = %d", (*p));
                          ↓
                          *p
                          *(65524)
                          ↓
                          10.
}
```

Stack region.

a	b
10	10
65524	65522
p	
65524	
65520	

OUTPUT

value of a = 10;

Date
Page 200

Syntax:- To change a value of variable by using pointers.

```
*Pname = value / expression;
```

Or,

```
scanf("<Format specifiers>", Pname);
```

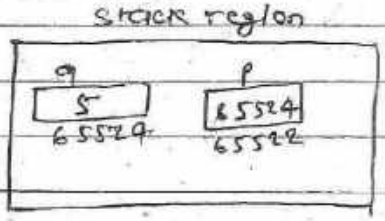
Ex:-

```
int a=5, *P;
P = &a;
```

1. `Printf("a = %d", *P);`

OUTPUT

a=5.



2. `*P = 20;`

↓

`*(65524) = 20;`

`Printf("a = %d", *P);`

OUTPUT

a=20.

3. `Printf("Enter a no:");`

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

Or

`scanf("%d", P);`

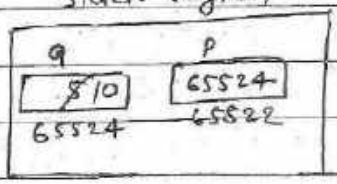
OUTPUT

Enter a no: 15.

Stack region.

a	P
65524	65524
65524	65522

Explanation:-




```
void main()
{
int a=5, *p;
clrscr();
  p=&a;
  printf("value of a=%d", a);
  printf("\n value of a=%d", *p);
  printf("\n value of a=%d", *(&a));
  printf("\n Address of a=%u", &a);
  printf("\n Address of a=%u", p);
  printf("\n Address of p=%u", &p);
*p=10;
 printf("\n a=%d", *p);
  getch();
}
```

OUTPUT

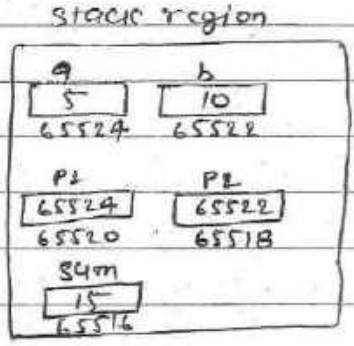
value of a=5
value of a=5
value of a=5
Address of a=65524
Address of a=65524
Address of p=65522
a=10.

Rough

*(&a)	*p
*(65524)	*(65524)
↓	↓
5	5.

WAP in 'C' to accept two no. From keyboard calculate and display. It's sum and average by using pointer.

Soln:-




```
void main()
{
    int a,b,*P1,*P2,sum;
    clrscr();
    P1 = &a;
    P2 = &b;
    printf("Enter two no:");
    scanf("%d %d",P1,P2);
    sum = *P1 + *P2;
    printf("\n sum = %d",sum);
    printf("\n Average = %F", sum/2.0);
    getch();
}
```

OUTPUT

Enter two no: 5 10

sum = 15

Average = 7.5.

WAP in 'C' to accept two no. From keyboard and interchange both to eachother by using pointer.

Soln:-

```
void main()
{
    int a, b, *P1, *P2;
    clrscr();
    P1 = &a;
```

```
    p2 = &b;
    printf("Enter two no:");
    scanf("%d %d", p1, p2);
    printf("a=%d, b=%d", *p1, *p2);
*p1 = *p1 + *p2;
*p2 = *p1 - *p2;
*p1 = *p1 - *p2;
printf("\n After swapping:");
printf("\n a=%d, b=%d", *p1, *p2);
getch();
}
```

Stack region

a	b	p
10	5	65
65524	65522	65

p2.
65522
65518

OUTPUT

Enter two no: 5 10
a=5, b=10
After swapping!
a=10, b=5

LEVEL OF POINTER

int a=10, *p1, **p2, ***p3;

a
10
65524

p1
P1 = &a; | 65524 | → level 0
65522

p2
P2 = &P1; | 65522 | → level 1
65520

p3
P3 = &P2; | 65520 | → level 2.
65518

1. printf ("value of a = %d", *p1);

OUTPUT

value of a = 10.

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

OUTPUT

Address of a = 65524.

2. printf ("value of a = %d", **p2);

OUTPUT

value of a = 10.

Rough

** p2

* (65522)

(65524)

10.

printf ("Address of a = %u", *p2);

OUTPUT

65524

Rough

* p2

(65522)

65524.

printf ("Address of p1 = %u", p2);

OUTPUT

Address of p1 = 65522.

3. printf ("value of a = %d", *** p3);

OUTPUT

value of a = 10

Rough

*** p3

*** (65520)

** (65522)

* (65524)

printf ("Address OF a = %u", **P3);

OUTPUT

Address OF a = 65524

Rough

** P3

** (65520)

* (65522)

↓

65524

printf ("Address OF P1 = %u", *P3);

OUTPUT

Address OF P1 = 65522

Rough

* P3

* (65520)

↓

65522

printf ("Address OF P2 = %u", P3);

OUTPUT

Address OF P2 = 65520.

printf ("Address OF P3 = %u", &P3);

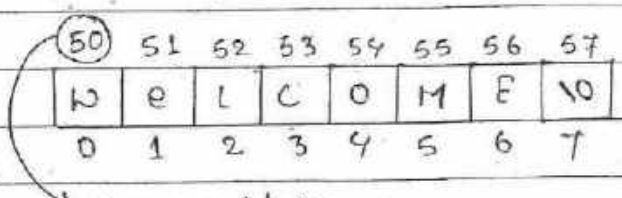
OUTPUT

Address OF P3 = 65518.

EXPLANATION OF POINTER.

1. printf ("welcome");

↓

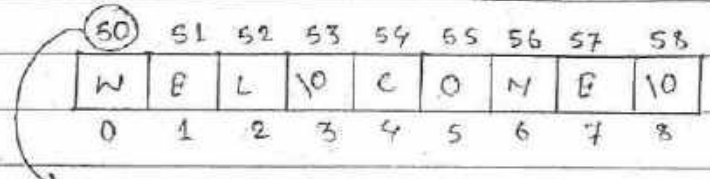


printf (50);

↓

welcome

2. printf ("welcome");



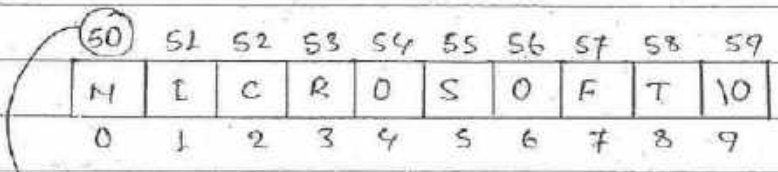
Base address.

printf(50);

↓

wel.

3. printf ("Microsoft" + 5);



Base address

printf (50+5)

printf(55);

↓

soft

4. void main()

{

clrscr();

printf(4);

getch();

}

Address range.

0 to 65535

OUTPUT

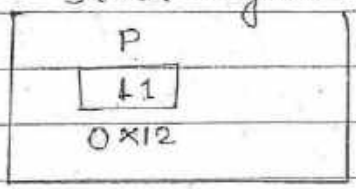
Borland c++ - copyright 1991 Borland intl.

5. printf (7);

OUTPUT

Land C++ - Copyright 1991 Borland Intl.

6. void main()
{
char * P = (char *) 11;
clrscr ();
* P = '\0';
printf (4);
getch ();
}



OUTPUT

Borland.

WAP in 'C' to display value of RAM until user press any key.

Soln:-

```
void main ()
{
    char *P = (char *)1;
    int i = 1;
    clrscr ();
    while (!kbhit())
    {
    textcolor (i);
    cprintf ("%c", *P);
    sleep (1);
    P++;
    i++;
}
```

Explanation - 8.

```
int a=10, *p;
p = &a;
```

*p++;	++*p
⇓	⇓
* [p++]	++ * (65524)
* (p+1*2)	++ 10
* (65524+2)	⇓
* (65526)	11
⇓	
0/ garbage.	

Ex:-

```
void main()
{
    int a = 10, *p;
    p = &a;
    clrscr();
    *p++;
    printf("%d, %u", *p, p);
    getch();
}
```

Stack region.

a	p	
10	65524	65526
65524	65522	

OUTPUT

0, 65526.

2.

```
void main()
{
    int a=10, *p;
    p = &a;
    clrscr();
```

printf("%d", *P++); — stack region.

getch();

}

a	P
10	65524
65524	65522

OUTPUT

10

Rough.

*P++
⇓
*(65524) ⇓ 10
P++ ⇓ P+2 ⇓ (65524+2) ⇓ 65526

3. void main()

{

int a=10, *P;

P = &a;

clrscr();

++ *P;

printf("%d, %u", *P, P);

getch();

}

Stack region.

a	P
10 11	65524
65524	65522

OUTPUT

11, 65524

Rough

++ *P;

++ (10);

⇓

11

4. printf("%u", ++ *P);

OUTPUT

11

5. `Printf ("%d\n", *++P);`

OUTPUT: 0

Rough:
`*++P`
`*(P+2)`
`*(65526)`
⇓
0

6. `Printf ("%d\n", (*P)++);`

OUTPUT: 10

Rough:
`(*P)++`
⇓
10 (++) → Here, ++ doesn't perform any work because it used as postfix
⇓
10

7. `Printf ("%d\n", --(29));`

OUTPUT
~~65522~~
Error :- lvalue required

8. `float a=10, *P;`
`P = &a;`
`Printf ("%d\n", ~~*P~~);`

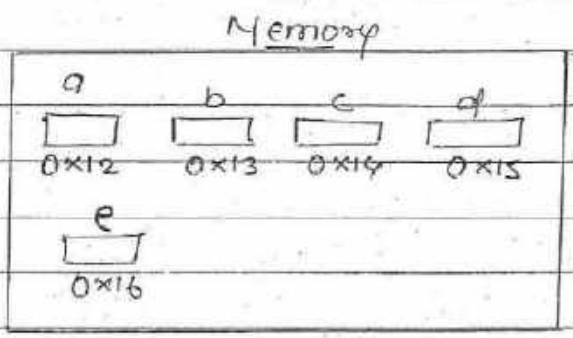
Rough
P++
⇓
P+2

CHAPTER:-7

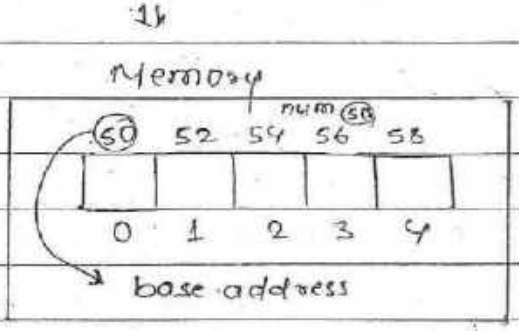
ARRAY

An array is a collection of more th one similar type variable into single name where memory space allocates contiguously.

Ex:- int a, b, c, d, e;



int num [5];



TYPE OF ARRAY.

An array can be categorised as a following way:-

#-Allocation at compile-time.

(1) one dimensional array.

(a) numeric array.

(b) character array.

(2) Two/ multi dimensional array.
(a) numeric array.
(b) character array.

Allocation at run-time.

Allocation at compile-time:-

(1) One dimension array:- An array defined with one dimension / size is known as one dimensional array.

Syntax:- datatype arrayname [size];

datatype	arrayname [size];
int	Must be fixed positive round integer.
Float	
double	
char	
user defined datatype	

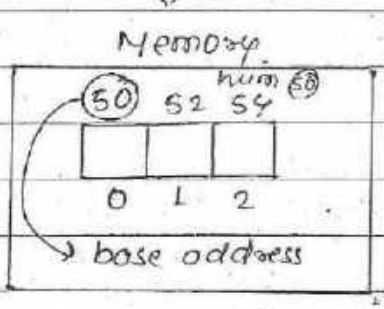
(a) Numeric array:- An array defined by numeric datatype such as int, Float or double is known as numeric array.

Syntax:-

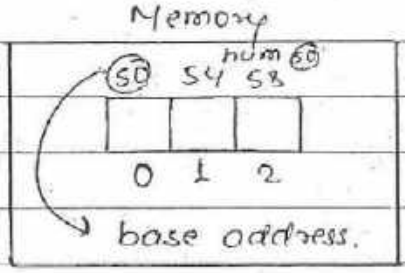
datatype arrayname [size];

datatype	arrayname [size];
int	Must be fixed positive round integer.
Float	
double	

Ex-1. int num [3];



Ex-2. Float num [3];



Notes:-

1. An array name always holds base address and once base address being assigned into array name then we can access all element of array by using array name.

2. An array index always start from zero.

INPUT :-

We can accept / store values into array as following three way:-

(1) Direct initialisation method.
(2) Using assignement operator.
(3) Through keyboard.

1. Direct initialisation method:-

Direct initialisation method is a technique by which we can assign values into any array at defenition time as a following Format:-

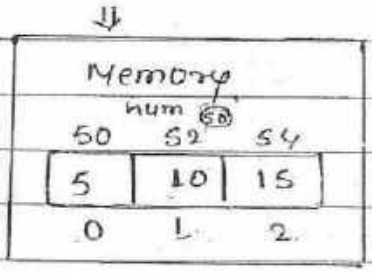
Syntax:-

datatype arrayname [size] = {val1, val2, --- };

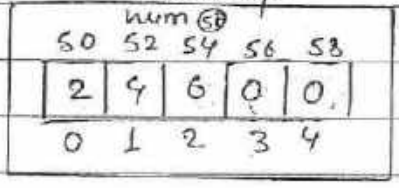
datatype arrayname [ ] = {val1, val2, --- };

Ex:-

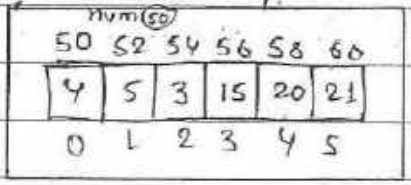
1. int num [3] = {5, 10, 15};



2. int num [5] = {2, 4, 6};



3. int num [ ] = {4, 5, 3, 15, 20, 21};



4. int num [3] = {5, 10, 15, 20}; ✗

NOTE:-

1. In direct initialisation method we can omit size of array.

2. In direct initialisation method a boundation of array will be check.

2. Using Assignement Operator:-

An using assignement operator (=) can be used as a following format to assign a value into particular element of array.

Syntax:-

```
arrayname [index] = value;
```

NOTE:- An index written into square braket as arrayname[index] internally interpreted as.

```
* (arrayname + index * capacity)
```

Ex:-

```
int num [3];
```

⇓

Memory

num 60

60	62	64
0	10	20
0	1	2

1. num [1] = 10;

⇓

* (num + 1 * 2) = 10;

* (60 + 2) = 10;

* (62) = 10;

2. num [2] = 20;

⇓

* (num + 2 * 2) = 20;

* (60 + 4) = 20;

* (64) = 20;

Date
Page 216

NOTE:-

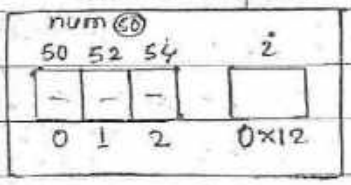
1. An array is a fixed pointer because once it holds address then we can't alter later.

2. When an array dekr allocates into stack region then it by default holds garbage but when we store a value into any elemen of array then it by default holds zero(0).

Ex:-

```
void main()
{
  int num[3], i;
  clrscr();
  printf("value of array are:");
  for(i=0; i<3; i++)
    printf("%d\t", num[i]);
  getch();
}
```

Stack region



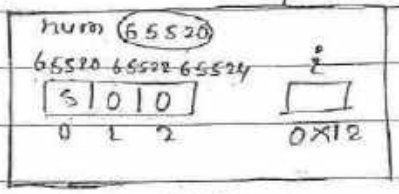
OUTPUT

value of array are: garbage garbage garbage.

Ex:-

```
void main()
{
  int i, num[3] = {5};
  clrscr();
  printf("value of array are:");
  for(i=0; i<3; i++)
    printf("%d\t", num[i]);
  getch();
}
```

Stack region.



OUTPUT value of array are: 5 0 0

Date
Page 217

3. Through keyboard:-

we can accept values into array by using inbuilt function scanf() as a following format

Syntax:-

```
For (i=0; i<size; i++)
{
Printf("Message prompt");
scanf("<Format specifier>", &arrayname[i]);
}
```

Ex:-

1. WAP in 'c' to accept six no. from keyboard and display it same on monitor by using one dimensional numeric array.

Soln:-

```
void main()
{
int num[6], i;
clrscr();
For (i=0; i<6; i++)
{
Printf("Enter a no:");
scanf("%d", &num[i]);
}
For (i=0; i<6; i++)
Printf("num[%d]: %d\n", i, num[i]);
getch();
}
```

Stack region.

num (65514)

65514	65516	65518	65520	65522	65524
5	10	8	15	20	25
0	1	2	3	4	5

OUTPUT

Enter a no:5

Enter a no:10

Enter a no: 8
Enter a no; 15
Enter a no: 20
Enter a no: 25
num[0]: 5
num[1]: 10
num[2]: 8
num[3]: 15
num[4]: 20
num[5]: 25

3.

2. WAP in 'C' to accept 10 no's from keyb check and display total count of even and odd no.

Sol:-

```
Void main()
{
int num[10], i, even=0, odd=0;
clrscr();
    Printf("Enter 10 no's:");
    For (i=0; i<10; i++)
        scanf("%d", &num[i]);
    For (i=0; i<10; i++)
        if(num[i] %2 ==0)
                even++;
        else
                odd++;
    Printf("Total even = %d\n Total odd = %d", even, od
getch();
}
```

OUTPUT

Enter 10 no's:- 10 5 4 3 2 6 7 8 1 15
Total even - 6

Date
Page 29

3. WAP in 'C' to define two array and display it as o following way:-

A[10]

50	52	54	56	58	60	62	64	66	68
5	4	3	8	9	7	6	3	2	1
0	1	2	3	4	5	6	7	8	9

B[5]

70	72	74	76	78
6	6	6	14	16
0	1	2	3	4

B[0] = A[0] + A[9]
B[1] = A[1] + A[8].

Soln:-

```
void main()
{
  int A[10], B[5], i;
  clrscr();
  printf("Enter 10 no's:");
  for(i=0; i<10; i++)
    scanf("%d", &A[i]);
  for(i=0; i<5; i++)
    B[i] = A[i] + A[9-i];
  printf("\n Elements of array A:");
  for(i=0; i<10; i++)
    printf("%d", A[i]);
  printf("\n Element of array B:");
  for(i=0; i<5; i++)
    printf("%d", B[i]);
  getch();
}
```

OUTPUT

Enter 10 no's: 5 4 3 8 9 7 6 3 2 1.
Elements of array A: 5 4 3 8 9 7 6 3 2 1
Elements of array B: 6 6 6 14 16.

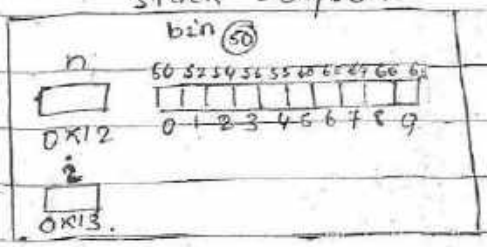
Date
Page 220

4. WAP in 'c' to accept a no. from keyboard, calculate and display it's binary equivalent.

Soln:-

```
void main()
{
    int n, bin[10], i;
    clrscr();
    printf("Enter a no:");
    scanf("%d", &n);
    for(i=0; n>0; bin[i] = n%2, n=n/2, i++);
    printf("Binary equivalent:");
    for(--i; i>=0; i--)
        printf("%d", bin[i]);
    getch();
}
```

Stack region.



Dry Run

n	n>0	i	OUTPUT
12	12>0 ✓	0	Enter a no: 12
6	6>0 ✓	1	Binary equivalent: 1100.
3	3>0 ✓	2	
1	1>0 ✓	3	
0	0>0 ✗	4	
		3	
		2	
		1	
		0	
		-1	

5. WAP in 'c' to accept a no. from keyboard calculate and display it's equivalent Hexa-decimal.

Date
Page 221

Soln:-

```
void main()
{
int n, hex[10], i;
clrscr();
printf("Enter a no:");
scanf("%d", &n);
for(i=0; n>0; hex[i] = n%16, n = n/16, i++);
printf("Hexa equivalent:");
for(--i; i>=0; i--)
{
switch(hex[i])
{
case 10: printf("A");
break;
case 11: printf("B");
break;
case 12: printf("C");
break;
case 13: printf("D");
break;
case 14: printf("E");
break;
case 15: printf("F");
break;
default: printf("%d", hex[i]);
}
}
getch();
}
```

(b) Character array:-

When an array define by char datatype is known as character array.

Syntax:-

```
char arrayname [size];
```

↓

Must be Fixed positive round integer

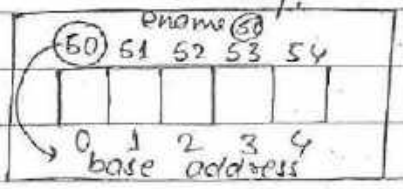
NOTE:-

1. In character array, a string automatically ended by backslash zero('\0') to seperate string from garbage value because a string of character array can be display as a following two way:-
   (a) character by character traverssing.
   (b) whole string at a time.

Ex:- char ename [5];

↓

Memory.



INPUTS

(1) Direct Initialisation Method
(2) Using assignement operator.
(3) Through keyboard.

(1) Direct Initilisation Method.

syntax:-

```
char arrayname[size] = "string";
```

or

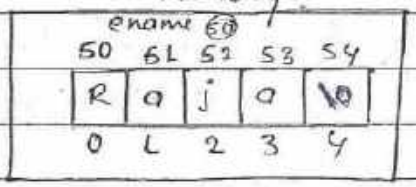
```
char arrayname[] = "string";
```

or

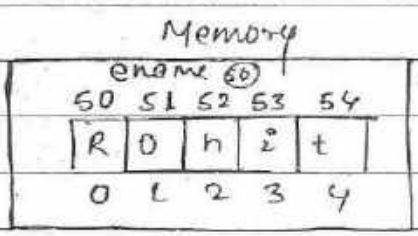
```
char arrayname[size] = {'char1', 'char2', ---};
```

Ex:-

1. `char ename[5] = "Rafa";`

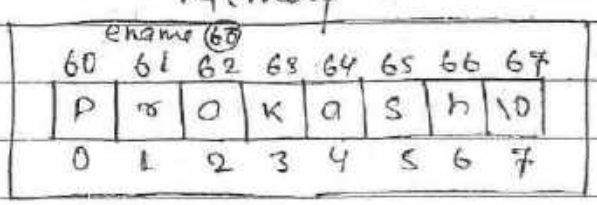


2. `char ename[5] = "Rohit";`



3. `char ename[5] = "Rakesh";` ✗

4. `char ename[] = "prakash";`



Date
Page 224

5. Char ename[5] = {'P', 'a', 'n', 'K', 'U'};

↓

Memory

ename (60)

60	61	62	63	64
P	a	n	K	U
0	1	2	3	4

2. Using Assignement Operator:-

Syntax:-

arrayname [index] = 'character';

NOTE:- An index written as arrayname [index] internally interpreted as.

* (arrayname + index * capacity)

Ex:-

Char Str[5];

↓

Memory

Str (60)

60	61	62	63	64
A	P			
0	1	2	3	4

(i) Str[0] = 'A';

↓

*(Str + 0 * 1) = 'A';

*(60 + 0) = 'A';

*(60) = 'A';

(ii) Str[1] = 'P';

↓

*(Str + 1 * 1) = 'P';

*(60 + 1) = 'P';

*(61) = 'P';

3. Through Keyboard:-

Syntax:-

```
Printf("Message prompt");
scanf("%s", arrayname);
        or
gets(arrayname);
```

OUTPUT

We can display a value of character a by using following two way:-

(i) Character by character traversing.

(ii) Whole string at a time.

(i) Character by character traversing:-

Syntax:-

```
Printf("Message prompt");
For(i=0; arrayname[i] != '\0'; i++)
  Printf("%c", arrayname[i]);
```

(ii) Whole string at a time:-

Syntax:-

```
Printf("<Message>%s", arrayname);
```

Ex:-

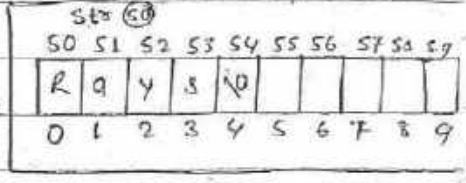
4. WAP in 'C' to accept a string from keyboard and display it on monitor with an appropriate message.

Sol:- Character by character traversing:-

```
void main()
{
  Char str[10];
  clrscr();
```

```
        printf("Enter a string:");
        scanf("%s", str);
        printf("given string is:");
        for(i=0; str[i]!='\0'; i++)
            printf("%c", str[i]);
    getch();
}
```

Stack region.



OUTPUT

```
Enter a string: Rays
given string is: Rays
```

Whole string at a time.

```
void main()
{
    char str[10];
    clrscr();
        printf("Enter a string:");
        scanf("%s", str);
        printf("given string is:%s", str);
    getch();
}
```

OUTPUT

```
Enter a string: Rays
given string is: Rays
```

gets():- gets() is an inbuilt function of stdio.h that is specially used for accept string from keyboard.

Syntax:- to call gets()

```
gets(arrayname);
```

Ex:-

Q. WAP in 'C' to accept your Friend name from keyboard and wish him/her for coming examination.

Soln:-

```
void main()
{
  char Fname[10];
  clrscr();
    printf("Enter your Friend name:");
    scanf("%s", Fname);
    printf("%s, best of luck for coming examination!", Fname);
  getch();
}
```

OUTPUT

Enter your Friend name: Ram ~~Prakash~~, Prakash

Ram, ~~Ram~~ best OF luck for coming examination;

2nd Method

```
void main()
{
  char Fname[10];
  clrscr();
    printf("Enter your Friend name:");
    gets(Fname);
    printf("%s, best of luck For coming examinat
                                    ", Fname
```

OUTPUT

Enter your Friend name: Ram Prakesh.
Ram prakesh, best of luck for coming examination!

2. WAP in 'C' to accept a string from keyboard calculate and display it's length.

Soln:-

```
void main()
{
    char str[10];
    int i;
    clrscr();
    printf("Enter a string");
    gets(str);
    for(i=0; str[i]!='\0'; i++);
    printf("length of str = %d", i);
    getch();
}
```

Dry Run

i	str[i]!='\0'	OUTPUT
0	'R'!='\0' ✓	Enter a string: Rays8
1	'a'!='\0' ✓	length of str = 4.
2	'y'!='\0' ✓	
3	's'!='\0' ✓	
4	'\0'!='\0' X	

Strlen():- It is a Function of string.h that accept a string as an argument and return it's length.

Syntax:- To call strlen()

```
(var)=strlen(str/"string");
```

↳ Optional.

int strlen(char

2nd Method

```
void main()
{
char str[10];
clrscr();
printf("Enter a string:");
gets(str);
printf("length of str=%d", strlen(str));
getch();
}
```

3. WAP in 'C' to accept two string from key and concatenate second string at the end of First string.

Soln:-

```
void main()
{
char str1[20], str2[10]; int i, j;
clrscr();
printf("Enter two string:");
gets(str1);
gets(str2);
```

```
for(i=0; str1[i]!='\0'; i++);
    for(j=0; str2[j]!='\0'; str1[i]=str2[j], i++, j++)
    str1[i]='\0';
printf("\n concatenated string = %s", str1);
getch();
}
```

OUTPUT

Enter two string: Ram
Prakesh.
concatenated string: Ramprakesh.

Strcat():- Is an inbuilt function of string.h that accept two string as an arguments and concatenate source string at the end of destination string.

Decleration syntax:-

```
char * strcat(char *s1, char *s2);
```

s1 → Destination string
s2 → Source string.

(3) WAP in 'C' to accept two string from keyboard and concatenate second string at the end of first string. check and display some message wheather both are same or not.

```
void main()
{
    char str1[20], str2[10];
    clrscr();
    printf("Enter two string:");
```

```
gets(s1);
gets(s2);
for(i=0; s1[i]==s2[i] && s1[i]!='\0'; i++);
if(s1[i]=='\0' && s2[i]=='\0')
        printf("Both are same");
else
        printf("Both are different!");
getch();
}
```

OUTPUT

```
Enter two string: note
note
Both are same!
```

strcmp():- is an inbuilt function of str.h that accept two string as an argument and compare both character by character an returns an integer value as a following condition.

Declaration syntax:-

```
int strcmp(char *s1, char *s2)
```

condition		returns value
s1==s2	⟶	0
s1<s2	⟶	-ve
s1>s2	⟶	+ve.

Explanation

```
char s1[5] = "Bot", s2[5] = "FOX";
int r;
```

(i) `r = strcmp(s1, s2);`
`printf("r = %d", r);`

OUTPUT
r = -4

Rough
strcmp(s1, s2);
⇓
'B' - 'F'
= 66 - 70
= -4

(2) `printf("%d", strcmp(s1, "Bca"));`

OUTPUT
-2

Rough
strcmp(s1, "Bca");
⇓
'B' - 'B' = 0
'o' - 'c' = 97 - 99
= -2

(3) `printf("%d", strcmp("Bca", s1));`

OUTPUT
2

Rough
strcmp("Bca", s1);
⇓
'B' - 'B' = 0
'c' - 'o' = 99 - 97 = 2

(4) `printf("%d", strcmp(s1, "Bot"));`

OUTPUT
0

Rough
strcmp(s1, "Bot");
⇓
'B' - 'B' = 0

2nd Method - Quest - 3.. Page 230

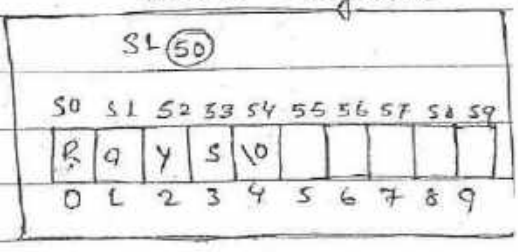
```
#include <string.h>
void main()
{
  char s1[10], s2[10];
  clrscr();
    printf("Enter two string:");
    gets(s1);
    gets(s2);
  if (strcmp(s1,s2) == 0)
      printf("Both are same!");
  else
      printf("Both are different!");
  getch();
}
```

4. WAP in 'C' to accept a string from keyboa and display it's reverse string.

Soln:-

```
void main()
{
  char s1[10];
  int i;
  clrscr();
    printf("Enter a string:");
    gets(s1);
    printf("given string:%s", s1);
    for(j=0; s1[j] != '\0'; j++);
    for(i=0, --j; i<j; i++, j--)
    {
      ch=s1[i];
      s1[i] = s1[j];
```

Stack region.



printf("\n Reverse string : %s", st);
getch();
}

OUTPUT :- Enter a string : Rays
given a string : Rays.
Reverse String : syaR.

Strrev:- is an inbuilt function of string.h that accept one string as an argument and reverse it.

Decleration syntax:-

```
char * strrev (char *s1);
```

Q.1

Sol:-

```
void main()
{
  char s1[10];
  clrscr();
    Printf("Enter a string:");
    gets(s1);
    Printf("given string:%s", s1);
    Strrev(s1);
  Printf("\n Reverse string:%s", s1);
  getch();
}
```

OUTPUT

Enter a string: Rays

given a string: Rays

Reverse string: syaR.

Date
Page 2

Q2. WAP in 'C' to accept a string from keyboard check and display a message whether given string is palindrome or not.

Soln-

```
Void main()
{
    char s1[10];
    clrscr();
    printf("Enter a string:");
    gets(s1);
    for(j=0; s1[j]!='\0'; j++);
    for(i=0, --j; s1[i]==s1[j]; i++, j--);
    if(i>j)
        printf("Palindrome!");
    else
        printf("Not Palindrome!");
    getch();
}
```

Stack region

	s1 (50)									
50	51	52	53	54	55	56	57	58	59	
a	n	n	a	\0						
0	1	2	3	4	5	6	7	8	9	

OUTPUT

Enter a string = anna.

Palindrome;

Enter a string = konak.

Palindrome;

2nd Method

```
# include {string.h}
  void main()
  {
  char s1[10], s2[10];
  clrscr();
    Printf("Enter a string:");
    gets(s1);
    strcpy(s2,s1);
    strrev(s1);
  if(strcmp(s1,s2)==0)
      Printf("Palindrome!");
  else
      Printf("Not palindrome!");
  getch();
  }
```

OUTPUT

Enter a string = anna

Palindrome:

Enter a string = kanak

palindrome:

3. WAP in 'C' to accept a sentence and a character from keyboard count and display total number of inputed character found in given sentence.

Soln:-

```
void main()
{
    char str[20], ch;
    int i, ctr=0;
    clrscr();
    printf("Enter a sentence:");
    gets(str)
    printf("Enter a character:");
    ch=getch();
for(i=0; str[i]!='\0'; i++)
    if(str[i]==ch)
        ctr++;
printf("A character:%c:Found in:%s:%d time
                                    ch, str, ctr);
while(!kbhit());
}
```

OUTPUT

Enter a sentence: Jaipur national university.
Enter a character: i
A Character:i: found in: Jaipur national university: 4 times!

2. Two dimensional array:-

An array define with two dimension/ size is known as two dimensional array.

Syntax:-

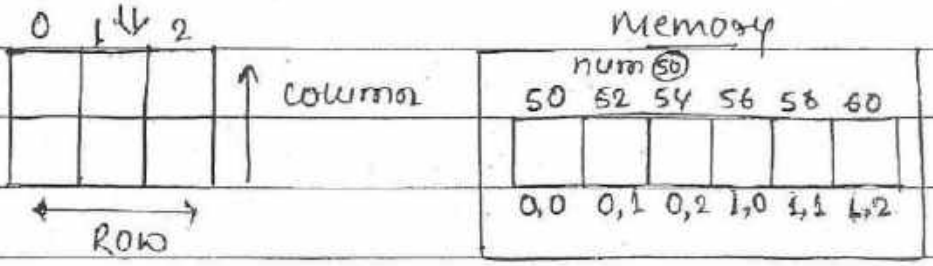
```
datatype array name [size] [size];
```

datatype: char, int, float, double, user defined datatype.

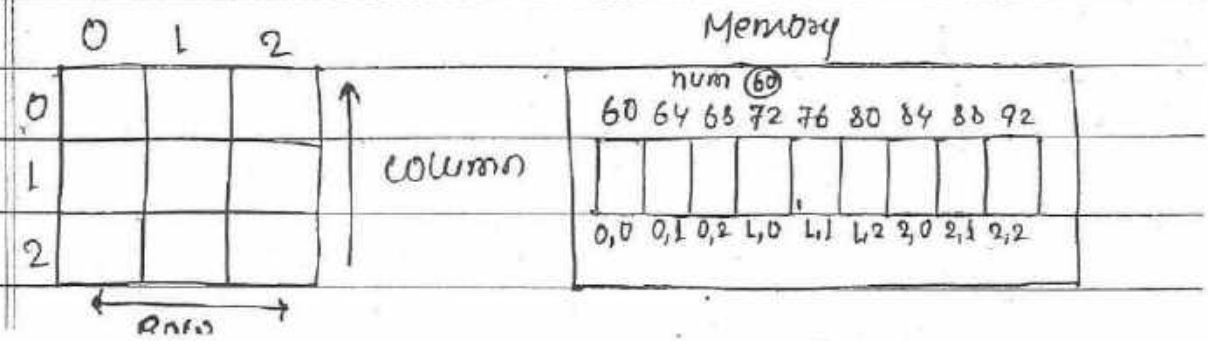
size: Must be fixed positive round integer.

(a) numeric array:-

Ex:- int num [2] [3];



Ex:- float num [3] [3];



INPUT

(i) Direct initialisation method

(ii) using assignement operator.

(iii) Through keyboard.

(i) Direct initialisation method:-

Syntax:-

```
datatype arrayname[size][size] = {{val1,
                val2,--},{val1, val2,---}---};
```

(or)

optional ↑ required ↑

```
datatype arrayname[][size] = {{val1, val2,---},
                {val1, val2,---},---};
```

NOTE:- In C language, an array can be represent into matrix form but non-matrix form doesn't supported.

Ex:- int num[2][3] = {{10, 20, 15}, {3, 6, 9}};

⇓

	0	1	2
0	10	20	15
1	3	6	9

(2) int num[2][3] = {{10,20}, {3,6,9}};

⇓

	0	1	2
0	10	20	0
1	3	6	9

(3) int num[2][3] = {10,20,3,6,9};

⇓

	0	1	2
0	10	20	3
1	6	9	0

(4) int num[3][2] = {{10,20}, {3,6}, {4,5}};

⇓

	0	1
0	10	20
1	3	6
2	4	5

(5) int num[2][3] = {{2,4,6}, {3,9,18,10}}; X

(ii) Using assignement operators:-

Syntax:-

```
arrayname [index] [index] = value;
```

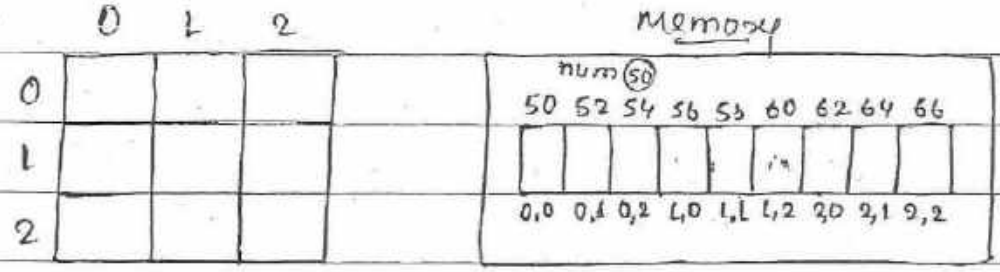
NOTE:- An index written as arrayname [index] [index] internally interpreted as

```
*((arrayname + i * capacity) + j * capacity)
```

Ex:-

```
int num [3] [3];
```

↓



(i) num [0] [1] = 10;

↓

*((num + 0 * 6) + 1 * 2) = 10;

*((50 + 0) + 2) = 10;

*(52) = 10;

(ii) num [2] [1] = 20;

⇓

*((num + 2*6) + 1*2) = 20;

*((50 + 12) + 1*2) = 20;

*(62+2) = 20;

*(64) = 20;

(iii) Through keyboard:-

Syntax:-

```
For (i=0; i<row; i++)
    For (j=0; j<col; j++)
    {
        Printf("Message");
        Scanf("<Format Specifier>", &argument[i][j]);
    }
```

Ex:-

1. WAP in 'C' to create 3×3 matrix and accept values from keyboard and display it some on monitor.

soln:-

```
Void main()
{
    int mat[3][3], i, j;
    clrscr();
    For(i=0; i<3; i++)
        For (j=0; j<3; j++)
```

```
printf("Enter a no:");
scanf("%d", &mat[i][j]);
}
printf("\n Elements of array are!\n");
for(i=0; i<3; i++, printf("\n"))
    for(j=0; j<3; j++)
        printf("%d", mat[i][j]);
getch();
}
```

OUTPUT

```
Enter a no: 5
Enter a no: 8
Enter a no: 3
_____ : 4
_____ : 6
_____ : 9
_____ : 12
_____ : 15
_____ : 20
Elements of array are!
5 8 3
4 6 9
12 15 20
```

	0	1	2
0	5	8	3
1	4	6	9
2	12	15	20

2. WAP in 'c' to create 3×3 matrix and display it -transpose matrix.

soln:-

```
void main()
{
int mat[3][3], i, j;
clrscr();
Printf("Enter element For 3×3 matrix
                                !\n");
For (i=0; i<3; i++, Printf("\n"))
    For (j=0; j<3; j++)
        scanf("%d", & Mat[i][j]);
Printf("* * * * * * * * *");
Printf("\n Transpose matrix:\n");
For( i=0; i<3; i++; Printf("\n"))
    For(j=0; j<3; j++)
        Printf("%d", mat[j][i]);
getch();
}
```

OUTPUT

Enter element For 3×3 matrix:

5 8 10

6 7 3

3 2 1

* * * * * * *

	0	1	2
0	5	8	10
1	6	7	3
2	3	2	1

FUNCTION.

A Function is a block code that provides a facility to combined / collects similar type statements into a single name. In c language a statement must be enclosed within a functions because without function a statements can't be execute.

A source code of c must be contains at least one user defined function named main because in c language an execution always start to main function. where main function automatically called by operating System and a main function also called manualy.

Advantage of Functions.

(i) Using function we can combined / collects similar type statements into a single name.

(ii) Using functions we can divides a larger programme into more than one small modules (Function).

(iii) Using Functions o debugging and maintenance will be easy.

(iv) Using Functions we can execute only desired statement.

(v) Using Functions we can reduce line of coding.

Types of Functions.

A Function can be divides into following two types according to ~~each~~ it's definition.

(i) System defined / pre-defined / inbuilt functions.

(ii) User defined Function.

(i) System defined / Pre-defined / inbuilt Functions. :-

A Function that definition already defined are known as System defined / Pre- defined or inbuilt Functions

A sourc code of inbuilt Function is not visible, that meanc we can use inbuilt Function

but we can't alter it's definition.

Examples:- printf(), scanf(), gets(), clrscr(), etc.

(ii) User defined functions:-

A function whose definitions depends upon an user is known as user-defined functions.

A user can define it own functions name and definition depends on user requirements.

Syntax:- TO define an user defined functions

(<return type>) function name (datatype part1,---)

↓ Optional ↓ Optional ↓ Formal Paramters

{

{ Sequence-of-stmt};

(return (value / expression);)

↓ Optional

}

Point to be noted down:-

1. A return type is optional and it ensure that a function will be return a value to it's caller or ref.

2. A return type may be system defined data type or user defined datatype.

3. A formal parameter is optional and it will be define when it is required to receive value to it's caller.

4. A formal parameter acts as local variable.

5. Return statement also optional and it depends upon return type.

Syntax:- to call a functions.

(Var =) Functionname (arg1, ---);

↓ optional ↓ optional ↓ actual argument

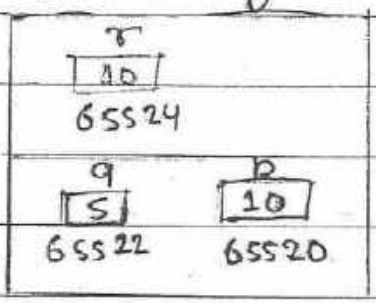
Note:- 1. A variable is optional and it depends upon on user.

2. A function does't work until it call.

3. An actual argument is optional and it depends upon a definition of function.

```
Ex:- void disp()
     {
     printf("\n my first programme!");
}
int check (int a, int b)
{
printf("\n check called!");
if (a>b)
    return a;
else
    return b;
}
void main()
{
int r;
```

Stack regions



```
clrscr();
r = check(5,10);
printf("\n Highest no. is %d", r);
disp();
getch();
}
```

OUTPUT

```
Check called!
Highest no: 10
my first programme!.
```

Note:- 1. A function can return only - one value.

2. In c language a program terminated when close curly braces of main function found.

ABOUT OBJECT ORIENTED LANGUAGE:-

An object is a combination of member data and member function , where member data must be encapsulated to avoid/prevent un-authorized temping and member function made publicly open so, that the end-user access the object through defined member function only to do needful.

Example:-

A car is an example of an object as it consist member data that is like piston, connecting rod enclosed with in engine crank, carburetor, gear box etc perfectly encapsulated and consist member functions like steering, gear, brake etc associated to corresponding parts that is member data and are publicly open so that end user or derive the car through defined member functions only.

EXAMPLE 2:-

A human being is a perfect example of an object as it consist member data that is parts like heart, liver, brain etc perfectly encapsulated and member functions like eyes, ear, hand, legs etc are provided to access the object that is human beings and do the needful.

ENCAPSULATION:-

It is a mechanism in which member data that is parts is bind or associates with its corresponding function/member function in such a way that member data that is parts perfectly encapsulated and member function made publicly open to ensured that the end-user access the object through defined member function only.

Example:-

A Car which is an example of an object go through process/mechanism encapsulation at the time of manufacturing object in which the automobile engineers binds together member data to its corresponding member functions like gear box associated with function gear and gear box perfectly encapsulated and member function gear made publicly open to ensured that the end-user can drive that is access the car through defined member function only.

Abstraction or Data abstraction:-

The outcome of the encapsulation is an object that can be refer to as an example of abstraction or data abstraction.

When an object is properly encapsulated that is member data perfectly hides and member function associated to their member data that is made publicly open at the time of manufacturing object when comes in hand of end-user to be access it is found that it can be access through defined member function only. and there is no way left to tempered the parts that is (member data) accidentally and intentionally and then the object is referred to as an example of abstraction or data abstraction.

Example:-

A T.V when owned by an end-user he/she found that all the parts (member data) like pictures tube ,digital circuit, inbuilt speaker etc are isolated with in insulted PVC cover or jacket and further enclosed with in cabinet perfectly encapsulated and functions like ON/OFF switch, menu button etc were provided externally on TV cabinet and on remote control to watch/access TV through defined function only like set the contrast, brightness, picture quality that suited to your eyes to

watch your favorite program/channels and there is no way left to tempered this part directly and such an object (TV) is refers to as an example of abstraction or data abstraction. And is being saved in long term.

Polymorphism:-

All object/object oriented language strictly follows the rule stated below.

"Similar purpose serving function must contain single interface".

When an object (living or non-living) follows the rule stated above it must consist at least one polymorphic function having single interface producing multiple output and an existence of polymorphic function in an object provides support to features polymorphism.

Which is made of three Latin Words?

POLY + MOR + PHISM

Which means one interface multiple outputs? Once an object posses/holds features polymorphism it save end-user from complexity.

Example:-

A human being which is an example of an object consist single interface that is eyes using for multiple purposes like viewing object placed at larger, moderate and shorter distance referred to as poly mor phic function, which provides support to features polymorphism by an object that is human being.

INHERITANCE:

All object/object oriented language must possess/holds the concept of "reusability" which means code once defined is reusable.

To support reusable features all object and object oriented language supports one technique/mechanism that allowed inherits of exciting class/type in a new class/type and made thing reusable to save time, energy and resources and such a mechanism/technique in object or any object oriented language is referred to as inheritance.

Thus an inheritance is one mechanism support in all object and object oriented language. That provides support to features "reusability".

The class/type that is inherited is called base class and the class/type that does the inheriting is called derived class respectively in java.

The derived class/type inherits all the properly of its base class and contains its own characteristics.