

C LANGEUAGE

BY.KIRAN

NARESH TECH

FRIENDS XEROX

Gayatri Nagar,Ameerpet.Hyd

Cell:9985011311



C-Language

Content

Page no

Constants.	1
Variables.	2
Data Types.	2
C-Editors. and Installation of Turbo-C	6
→ I/O, O/P Stmts.	7
→ Escape Sequences.	
→ Comments.	11
→ Tracing the program.	14
→ Debugging of a program & diff type of errors.	15
→ Predefined header files.	16
→ Internal Execution of 'c' program.	22
Operators.	19
→ Increment, Decrement operators.	34
Control Structures.	37
→ Conditional Operator (Ternary)	
→ Conditional operator if, ifelse, nested if's.	
→ Loops.	
Switch.	
→ Switch C	53
→ While C	63
→ for C	74

→ do while ()	78
→ low level programming (number system). Binary, Octal, Hexa	80
→ Bitwise operators.	83
5. Preprocessor Directives. Macros.	87
Functions.	93
Storage Classes. → scope of variables → auto, static, extern, register. → Memory Model. Stack, heap, PMA	107
Pointers.	113
Arrays.	121
Dynamic Memory allocation. malloc(), calloc(), realloc(), free(). → formatted I/O op. getchar(), getch(), & getch(), gets(), puts(), gotoxy()	138
Strings.	145
Graphics.	156
Files.	157
→ diff. modes of a file: r mode, w mode, a mode. etc. fputc(), fgetc(), fprintf(), fscanf()	160
→ Command line arguments.	163
Structures.	165
→ Enumeration, → Bitfield.	

FEBRUARY 2012

SUN	M	T	W	T	F	SAT
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29			

20 12

05

JANUARY

005-361-Week 1

Real Constants:

Real constants are also Number type with fractional point.

→ It can be (+ve) (0) (-ve).

→ By default it is +ve.

The real constant numbers doesn't accept any special symbols in b/w the Numbers, but they are separated in Two parts.

1. Mantissa

2. Exponent.

Ex:- 56.678, 9.453, -45.67 etc.

→ The Minimum range of Real constant

-3.4×10^{-38} to 3.4×10^{38}

Character Constant:-

a character constant always enclosed in Single quotation ' '.

→ Each character in Keyboard is 1 byte.

In a Keyboard Every key acts as a character.

The Minimum range of a character

- 128 to 127.

Ex:- 'a', '0' // Here when you place 0 in single quotes it takes it's ASCII val

String Constant:-

a String Constant is a Combination of characters, always the String Constant must be enclosed in double quotes " ".

Ex:- "India"

FRIDAY

06

20 12

006-360-Week1

JANUARY

JANUARY 2012

SUN	M	T	W	T	F	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

Variables:- a Name which is given for any computer memory location is called as Variable.

The purpose of the Variable to store some data.

A Variable Name can be called as alias to the memory location.

Rules for declaring a Variable

1. a Variable Name can be alphanumeric character.
2. A Variable name can be lower case, uppercase, and mixed case.
3. A Variable name can't be a Keyword.
4. a Variable name can't contain any special characters except - (underScore).
5. a Variable name can start with alphabet (e.g) underScore but not a digit.
6. The length of a Variable Max of 256 characters, but The Compiler will read only first 32 characters.

REPLY TO

→ The user access the data by the Variable name but the compiler will access by the address of the locations (i.e physical location)

FEBRUARY 2012

	M	T	W	T	F	S
1		1	2	3	4	
2	6	7	8	9	10	11
3	13	14	15	16	17	18
4	20	21	22	23	24	25
5	26	27	28	29		

20 12

07

2

JANUARY

007-359 • Week 1

Valid variables:-

India, abc, a100, -, -123 → only - is also a variable

Invalid Variables:-

for — because it is a Keyword

1a — " if starts with a Number

e.name — . (dot, Lower, space are not allowed)

e no — space. (in b/w Variable name).

1,2,3 — .

Data types:-

a Data type describes what type of data we can store in a Variable. It also tells, how many bytes are to allocate for the Variable.

Always Variables will be define w.r.t to data type

→ Data types are classified as Primary data types.

① Predefined data types.

② Secondary data types

(a) User defined..

REPLY TO

1. Primary data types:-

Primary data types are the data types which are directly manipulated by the machine instructions.

SUNDAY 08

→ it can also be called as pre defined or primitive data types.

MONDAY

09

2012

009-357 • Week 2

JANUARY

JANUARY 2012

SUN	M	T	W	T	F	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

a primary data type can be defined with Type-modifiers.

a type modifier will alter the meaning of a data type. \downarrow
changing.

There are 4 types of Type modifier.

1. Signed } default These are int
2. Short }
3. Unsigned }
4. long }

Signed:-

~~Not Singed~~

int (signed) (short) (signed int)

format specification : %i @ %d.

size : 2 bytes (16 bits)

Range : -32768 to 32767

Description:-

If accept the No's without fractional point.

Ex:- 78,345, -56, 2342 etc

Macros:-

INT-MAX \rightarrow it get the o/p of max integer range. 32767

INT-MIN \rightarrow it get the o/p of min integer range. -32768

FEBRUARY 2012

SUN	M	T	W	T	F	S
5		1	2	3	4	
6	7	8	9	10	11	
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29			

20 12

JANUARY

10

010-356 • Week 2

unsigned int:-

format specification:- %u

size : 2 bytes

Range : 0 to 65535

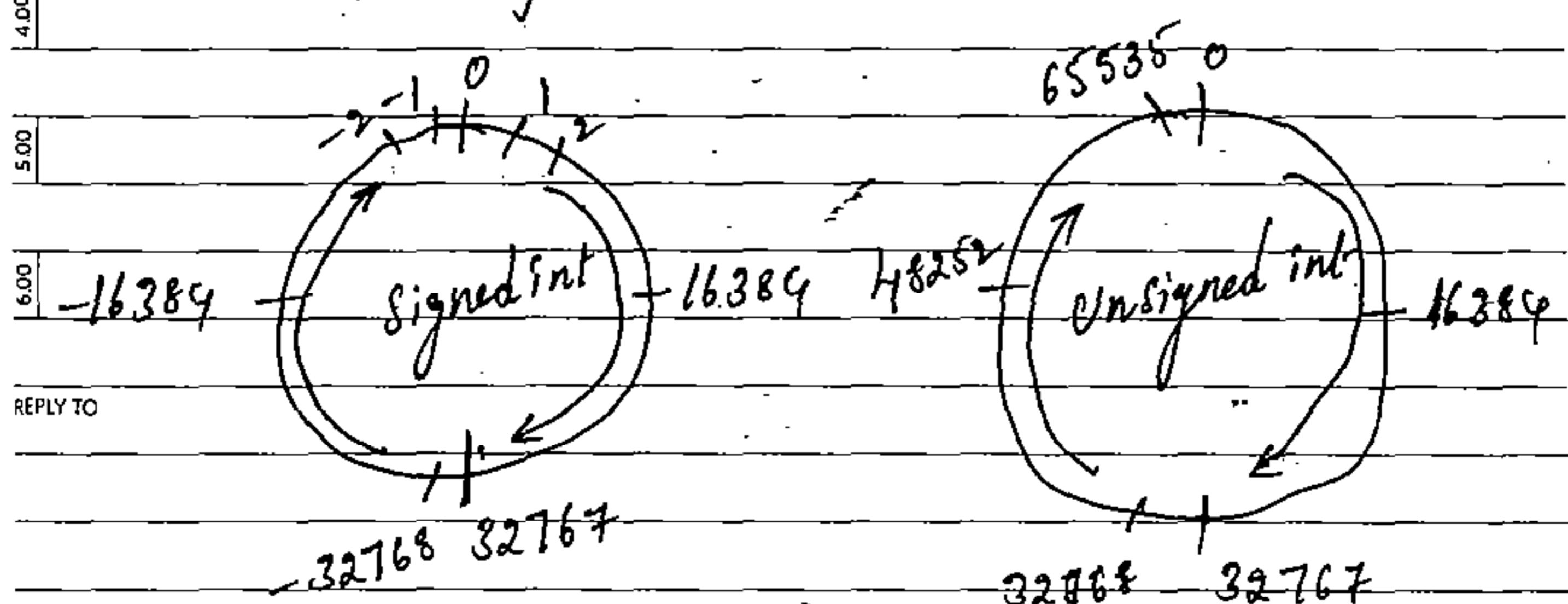
Description:

This datatype will accept only the +ve no's. It will not accept -ve values due to we defined as unsigned type modifier.

Ex:- 78u, 45231u, 5u, etc.

Macros: $\text{UINT_MAX} = 65535$
 $\text{UINT_MIN} = 0$.

Note:- Internally the binary codes are same
 externally Numbers are different.

long (signed long):-

format specification: %ld

size : 4 bytes.

WEDNESDAY

11

2012

011-355 - Week 2

JANUARY

JANUARY 2012

SUN	M	T	W	T	F	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

Range: -2 147 483 648 to 2 147 483 648

Description:-if also accepts integer type of data
with more range of values and it is an
signed integer.Ex:- 78L, 45231L, -6755L etc

l = long

Macros:-

LONG - MAX

LONG - MIN

Unsigned long:-

specification: %lu

size: 24 bytes

Range: 0 to 4 294 967 295

Description:-integer type of data with only
pos values.Ex:- 78lc, 45231lc, 6755lc etc.Macros:- ULONG - MAX

ULONG - MIN

REPLY TO

Char:- (signed char)

specification: %c

size: 1 byte

Range: -128 to 127

Description:-

Every character has to be

FEBRUARY 2012

SUN	M	T	W	T	F	S
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29			

20 12

12

JANUARY

012-354 • Week 2

enclosed in Single quotes & every key in the keyboard acts as a character.

Ex:- 'A', 'i', 'g' etc. '1' etc

Macros:

CHAR - MAX

CHAR - MIN

Unsigned char

Format specification: %c

Size: 1 byte

Range: 0 to 255

Description:

More range of characters.

Ex) 'A', 'i', 'z' etc.

Macros:

UCHAR - MAX

UCHAR - MIN

All these macros are defined from one of the header file called as `<limits.h>`

float:-

Specification:- %f

Size: 4 bytes

Range: -3.4×10^{38} to 3.4×10^{38}

Description:-

It accept the numbers with fractional point & it will display maximum of 6 decimal points.

FRIDAY

13

20 12

013-353 - Week 2

JANUARY

~~float~~ Here the ~~float~~ is accepted how much you give total more than 6 decimal, but output display only upto 6 decimal.

JANUARY 2012						
S	M	T	W	T	F	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

Ex:- $67.54321456321f$, $45.453f$, $-5e.01f$, etc

Macro's:-

~~F~~ LT LT - MAX

LT - MIN.

double:-

specification: %.lf

size : 8 bytes.

Range: -1.7×10^{308} to 1.7×10^{308}

Description:-

It also accept the fractional-point numbers with more range of data.
it will display 12 decimal points. (accepting only more, but display upto only 12 decimals).

Macro's:-

Ex:- 459.127 , 62.97567 , -100.00 .

Macro's:- DBL - MAX

long double:-

DBL - MIN

specification: %.Lf (1) %.LF

size : 10 bytes.

Range: -3.4×10^{4982} to 3.4×10^{4982}

Description:-

If it is also fractional point of no's with 19 decimal points.

Macro's:-

LONG DBL - MAX

LONG DBL - MIN

FEBRUARY 2012

SUN	M	T	W	T	F	S
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29			

2012

14

SATURDAY

JANUARY

014-352 • Week 2

For the above Macros the header file is <float.h>

* Note:- Internally there are 2 types of data-types only present, both are number type.

1. without fraction no's.
2. with fraction no's.

Since, char is also integer type of no. if accepted it's ASCII code.

According to the 16 bit operating system the integer size is 2 bytes.

According to the 32 bit o.s. the integer size is 4 bytes.

2. Derived data types: Ex:- pointers, arrays.

3. Secondary data types: (or) user defined data type.

Ex:- Unions, enumeration,

Structures etc.

Declaration & Initialization of a Variable:-

Syntax:- of variable declaration:

SUNDAY 15

<type> <SIZE> <sign> datatype Val-name = <initialization>

(4) | optional
auto (default)

+ve

-ve

↓ optional

2 sheet
long

MONDAY

16

2012

016-350 - Week 3

JANUARY

TOKEN
Separator

JANUARY 2012

SUN	M	T	W	T	F	S
	2	3	4	5	6	7
	8	9	10	11	12	13
	15	16	17	18	19	20
	22	23	24	25	26	27
	29	30	31			

8.00

int a;

(as)

signed a;

(as)

short a;

float K;

char c;

(as)

signed char c;

int a;

int b;

(as)

int a, b;

float _;

↳ underscore is

also a variable, we can

also assign any value.

_ = 97.8916;

9.00

10.00

11.00

12.00

1.00

→ int a=78; // Both declaration & initialization at the same line

→ char india = 'D';

Variable name Value signing to Variable

2.00

→ int a, b, c;

a = b = c = 10 // valid.

a, b, c are created in diff memory locations and have the same value 10.

REPLY TO

int a = b = 10; // invalid.

int a1; // valid.

int 1a; // invalid because var name not start with no.

FEBRUARY 2012

S	M	T	W	T	F	S
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29			

20 12

17

TUESDAY

JANUARY

017-349 - Week 3

unsigned a : // by default int.

FLAVOURS OF C EDITORS

There are diff types of compilers available & each compiler supports it's own editor.

The editor wch we can write a 'c' program

Editors

OS

1. Turbo c/c++ (16bit) DOS

2. In any editor
every 'c' program save with
.c Extension.
· CPP with C++ programs

3. Borland c/c++ windows

4. Dev c/c++ windows

5. Microsoft VC++ windows

6. Anjuta IDE Linux

7. Berkeley c/c++ Unix

8. pro * c/c++ Unix / oracle etc.

Examples of 'c' Compilers:

TURBO C

QUICK C

MICROSOFT C

WEDNESDAY

18

2012

018-348 - Week 3

JANUARY

JANUARY 2012						
S	M	T	W	T	F	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

AZTECH-C

WATCOM C

GREEN LEAF C

VITAMIN C

All These Compilers provides their own IDE's (integrated development environment).

IDE Contains 5 Jobs.

1. Debugger. (Removal of error)
 2. Editor. (Write / Remove)
 3. Compiler.
 4. Linker.
 5. Preprocessor

→ TURBO C is an editor w it is developed under 8085 16 bit Micro processor.

Teschra

Installation of TURBO C :-

Step 1. TURBO C++ 3.0

Step 2. setup.exe.

~~Sep 3~~ : tc

Step 4: Bin

Step 5: fc-exe

The set path: Go To Run

~~to change direction etc~~

c:\etc\bin\tc.o<

To change editor options → Environment → Editor.

FEBRUARY 2012

S	M	T	W	T	F	S
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29			

20 12

19



JANUARY

019-347 - Week 3

All ways program execution starts from main.

collective

Main()

scope

{ statements }

} // end of Main()

Note: Any programming language (all - c prog execution always takes from main()).

Main() is starting line of program.
Without main() program execution never takes place.

Main is not a Stmt, it contains set of stmts.

Input output stmts:-

Reading, processing (or) writing of data are the three major things of a computer progs.

REPLY TO

Most progs takes some data as i/p & display the processing data as o/p.

They are 2 Methods of providing the data, to the program by declaring the variables & initialize the variables, but every time it is not possible then we can use another

JANUARY 2012

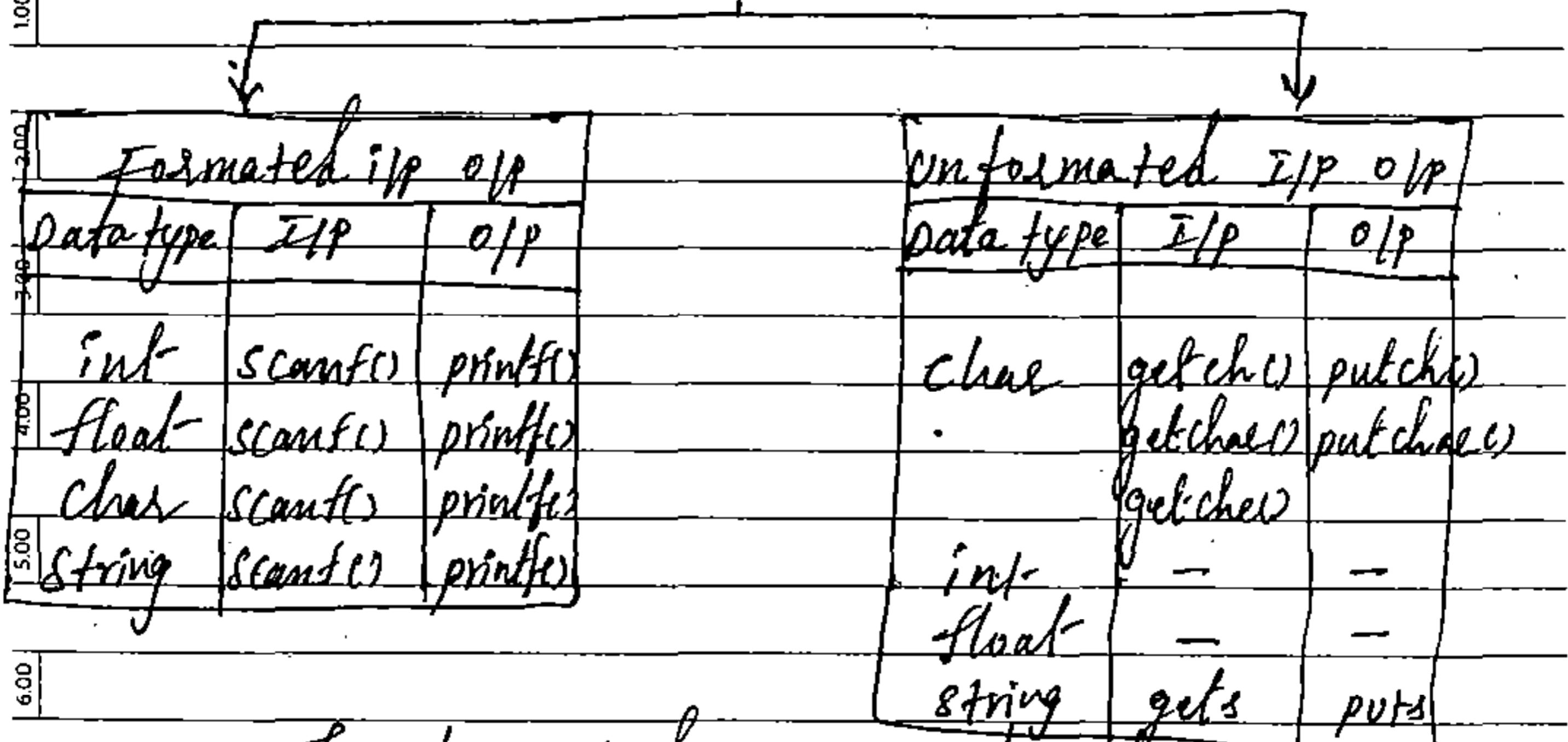
S	M	T	W	T	F	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

Method i/p Stmt. & to display the data on to the terminals we have o/p stmts.

In 'c' language this i/p, o/p operations are carried out through functions, they are called as Console i/p, o/p functions.

This functions receive i/p from Keyboard & write o/p to the VDU (visual display unit)

I/P O/P (I/O Library)



The formatted i/p, o/p functions can capture & display any type of data. The unformatted only for specific type of data.

Syntax :-

`printf("Control String");`

The printf is an o/p stmt to display any type of data we use printf stmt

FEBRUARY 2012

SUN	M	T	W	T	F	S
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29			

20 12

JANUARY

21

021-345 • Week 3

8

The `printf` has diff types of syntax as the first syntax display the data on the console screen. which are the data we are displaying with always place with in the double quotations, it is treated as a string.

Main()

{

```
printf(" Welcome to C ");
printf(" 1234 ");
```

}

After writing the prog to compile & execute the program for seeing the o/p of the program we have to run the program.

ALT + F9 --- Compile

Ctrl + F9 (or) F9 --- Run

ALT + F5 --- To see o/p

Escape Sequence:

In order to design the o/p of the prog in o/p stmts, we can use the Escape sequences.

The Escape seq will be used in order to design the o/p in a clear order.

They are diff types of Esc sequences to design the o/p.

All Esc seq mostly used in the o/p functions.

MONDAY

23

20 12

023-343 - Week 4

JANUARY

JANUARY 2012

SUN	M	T	W	T	F	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

This ESC sequences will be prefixed with back-slash followed by the character.

This ESC sequences can be used anywhere in the printf but mostly will be used either starting (or) Ending so they are called as boundaries (or) delimiters.
In -- Newline character.

In -- Move the cursor to the Next line.

Many ^{metho}ds

1. classic;

printf(" welcome to c\n");

printf("In it is pop's ");

tab character }

tab ← It - Moves the cursor to next horizontal tab position.

Each tab character will makes to move frame by frame, each frame is 8 characters
we have 10 frames

← 1 to 80 columns →

← 8 →
spaces
one frame

Row 1 - 25

like this we have
10 frames

FEBRUARY 2012

SUN	M	T	W	T	F	S
5		1	2	3	4	
6	7	8	9	10	11	
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29			

20 12

24

9

JANUARY

024-342 • Week 4

printf ("WelcomeHai");

O/P: Welcome Hai

The cursor starting from 1st frame
1st frame has 7 characters - 'H' at 8th place
in the frame when it is coming if
going to next frame.

Suppose printf ("wel|tcome Hai");

Here it is coming in the 4th place
Here after wel 5 spaces are coming because
every frame has 8 spaces after 3 characters
tab space is come, so it goes to next frame.

So the result is wel come Hai.

\b -- back space

Ex:- welcome

welc\bome o/p: welcome

Ex:- welcome\b

wel\c\ome

wel\dl\re

O/P:- welcome

ome = welcome

* * * Backspace move back one position.

Ex:- password logic

WEDNESDAY

25

2012

025-341 - Week 4

JANUARY

JANUARY 2012

S	M	T	W	T	F	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

Main()

{

else {

printf (" Kilag");

Kilag-

printf (" l ");

Kilag

printf (" ");

Kilag

printf (" l ");

Kila

printf (" n ");

Kilan

}

→ |r -- carriage return.

The 18 - Makes to move horizontally
to the starting position of the same row.

printf (" Rat\n");

O/P:- Ratprintf (" RAT\n"); // Here R is written
// replaced with A
// & cursor under AO/P:- CAT

REPLY TO

1. What is The o/p?

Main()

{

printf (" fnab"); cursor coming to next line

FEBRUARY 2012

	M	T	W	T	F	S
				1	2	3
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29			

20 12

10

26



JANUARY

026-340 • Week 4

printf("absi"); abbsi, absi

printf("asihai"); asi\lyhe abss;

{

asihai
hai

asi
hai

Note:-

Some characters not possible to display directly then use the backslash ' \ ' & follow the character we can able to watch the characters on the screen.

printf("welcome to \"c\"");

O/P:- welcome to "c"

→ Single ' ' will not display.

give Two " " one ' ' will display on the screen

not display ~~display~~ display.

→ displaying % is completely opposite of ' '.

REPLY TO

→ Main()

{

close();

printf("This is procedure oriented prog\r\nramming");

O/P: This is procedure oriented programming.

TUESDAY

27

2012

027-339 • Week 4

JANUARY

JANUARY 2012

S	M	T	W	T	F	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

Escape Sequences:-

\\t -- tab character

\\b -- backspace

\\r -- carriage return

\\\" -- "

\\ -- \

\\a -- beep sound

\\v -- vertical tab etc

\\n -- New line character

printf("This is \\n a pop's");

O/P: — This is \\n a pop's.Note:-

The Escape Sequence character will occupy one byte of character.

Ex:- 1. main()

{

class();

printf("Hello\\n");

printf("Welcome\\r\\n Naresh");

printf(" Welcome\\r\\n Naresh");

printf(" Welcome\\b\\b");

printf(" Welcome\\b\\bHello");

printf(" Hello\\r");

printf(" Welcome\\rHello");

printf("Hello\\t welcome");

{

REPLY TO

FEBRUARY 2012

SUN	M	T	W	T	F	S
			1	2	3	4
	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29			

20 12

28

SATURDAY

JANUARY

028-338 • Week 4

getch();

It is one of the function which compiler read the getch(); it suspend the program ~~for a while~~ until the user press any key on the keyboard.

* Note:

The actual purpose of getch(); to capture non data bytes.

O/P for Ex:- Hello
Welcome In Walesh

Comments:

C supports a concept called comments to provide the documentation of the program.

which ever the stmt are placed at comments they will not be executed by the compiler. The stmt's will be ignored.

C language supports Two types of Comments

1. Single line Comments. // SUNDAY 29
2. Multipline Comment /* */

MONDAY
30 2012
030-336 - Week 5

JANUARY

JANUARY 2012

SUN	M	T	W	T	F	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

* Note:- Nested comments are Not supported by 'c' language.

9.00 Syntax 2: for printf();

10.00 `printf("Control string with format specification",
arg1, arg2, ..., argn);`

11.00 The second syntax of the printf() will be defined with format specifications.

12.00 The no. of format specifications must be equal to no. of arguments.

13.00 The arguments will be substitute of at in the place of format specifications.

14.00 In printf() - The 1st argument must be in double quotations. & the remaining arguments depends on format specification.

15.00 In printf() any no. of arguments can accepted.

16.00 The printf() works by call by value.

REPLY TO

* Difference b/w syntax 1 & Syntax 2 of printf() */

main()

{

clrscr();

FEBRUARY 2012

SUN	M	T	W	T	F	S
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29			

20 12

JANUARY

31

TUESDAY

031-335 - Week 5

printf("2 + 3\n"); — syntax 1

printf("2 + 3 = %n"); — syn-1

printf("%d\n", 2+3); — syn-2

printf("2 + 3 %d", 2+3); — syn-2.

getch();

}

output:-

2 + 3

2 + 3 =

5

2 + 3 = 5

In printf() — the arguments can be variable type, constant no's / (or) Expressions.

printf("%d %d %d", 100, 200); // 100, 200.

printf("A:%d B:%d", 10, 20); // A:10, B:20

printf("%d %d %d", 10, 20, 30); // 10, 20

read only 10, 20 because a %'s.

printf("%d %d %d %d", 100, 200); // 100, 200, Garbage value

Syntax for scanf() :-

address

scanf() format specification: & arg1, & arg2...& arg

The Scanner also accept any no. of args in this also the 1st argt with in the " "

WEDNESDAY

01

20.12

032-334 • Week 5

FEBRUARY

FEBRUARY 2012

S	M	T	W	T	F	S
1	2	3	4			
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29			

It is used to capture the S/I/P from from
the Console device like Keyboard.

The arguments of the scanf() must be
variable type only.

Every argument must be define with
the address. The address indicate physical memo-
ry location of the variable.

Note:- scanf() works on call by address.

Note:- Any function work flow in 'c' language
(or) C++ will be right to left.

Program: Write a program accept 2 Variables as an S/I/P
& findout sum of the Variables.

int + int = int
The '+' operator can
understand only the No's. float + float = float
char + char = invalid.

main()

{

* declaration block *!
short a, b, sum;

* Execution block *!

C:\SSCC\1\

MARCH 2012

SUN	M	T	W	T	F	S
				1	2	3
	4	5	6	7	8	9
	11	12	13	14	15	16
	18	19	20	21	22	23
	25	26	27	28	29	30
	31					

20 12

02

MONDAY

FEBRUARY

033-333 - Week 5

```

printf(" Enter the values: ");
scanf(" %d %d ", &a, &b); /* if stmt */
sum = a+b;
printf(" sum of Two numbers %d ", sum);
getch();
}

```

O/P: Enter the values : 10 20
20

Sum of Two numbers : 30.

~~Notes:-~~

→ When the user doesn't follow the rules of the programming language Errors will be taken place.

→ After framing the Stmt, if it is not a meaningful Stmt then warning will takes place.

Ex:- a+b; defines a warning code has no effect.

→ In 'C' language Variable declaration can't be taken place in Execution block, but in C++ it is possible.

→ A 'C' language is a block level programming in every block, declaration block is present & Execution block is present, but the first block is Cased declaration, Second block Execution blo

FRI DAY
03 2012
034-332 • Week 5

FEBRUARY

FEBRUARY 2012

SUN	M	T	W	T	F	S
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29			

Main()

{

int a;

printf(" Block 1");

{ int x;

printf(" Block 2");

}

↓

The program will execute. Executed

Main()

{

int a;

printf(" Block 1");

{

printf(" Block 2");

int x;

{

Not

If it is an Error, since after Execution block, we are defining declaration block.

Tracing of the prog:-

A prog can be trace line by line by pressing F7

→ We can also watch the variable values at the time of execution by using Add Watch. If will be get by pressing Ctrl + F7

→ Already the add watch will be opened & going backside (or) off side by pressing F6 we watch on screen (or) we come back to add watch on screen.

→ An ordinary variable can hold only one value at a time.

MARCH 2012

SUN	M	T	W	T	F	S
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

2012

04

14
SATURDAY

FEBRUARY

035-331 • Week 5

→ $\% .2f$ - It is restricted the output with Two decimal points only.

$\% .3f$ - 3 decimal points after . in o/p.

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

displaying left-to-right $a = 10$ $b = 20.$

`printf("%d %d", a, b);`

processing our workflow
right to left

prog 2:-

Main()

{

int a;

float s;

long l;

$a = 32767 + 1;$

$s = 32767 + 1;$

$l = 32767 + 1;$

$a = 32767 + 1$

int int int

int

because it Next-n
of 32767

$s = 32767 + 1$

float int int

inf -32768 after conver-

-32768.00 is their

" `printf("%d %f %ld", a, s, l);` long inf-SUNDAY 05

getch();

int long int long
inside long -32768 long

O/p :- -32768 -32768.0000 -32768

MONDAY

06

20 12

037-329 • Week 6

FEBRUARY

FEBRUARY 2012

S	M	T	W	T	F	S
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29			

l = 32767 + 1 l;

printf

printf(" %d\n")

0 l = 32768

25 4112

Debugging of the program:-

Debugging is a process of rectifying the errors, in any programming language when a program is compiled & execute it.

diff types of errors will be taken place

The errors like 1. Syntax errors.

2. Logical errors (or) Semantic errors

3. Execution errors. (or)

Runtime errors.

Syntax errors:-

Syntax means Correct way (or)

"grammar" of writing a command (or) Series of commands, including all the proper options and Command-Line Stmts.

Syntax errors will be takes place on whenever the user doesn't follow the rules of the prog language. This errors can easily be identified by the programmer, since the compiler will clearly show the help.

MARCH 2012

	M	T	W	T	F	S
1			1	2	3	
2	5	6	7	8	9	10
3	12	13	14	15	16	17
4	19	20	21	22	23	24
5	26	27	28	29	30	31

by using Ctrl + F5 to show
the information about that. 20 12 → only F5 to show details of
FEBRUARY 07

038-328 • Week 6

Ex:- writing the Keywords in uppercase, ; missing, etc.

Logical errors:-

"Semantics" means the logical meaning of the stmt, separate from the grammatical structure.

Logical errors are purely mistake of the user. logical errors will not identify by the Compiler.

Ex:- Net salary = Basic salary +
Allowances - Deductions.
But through outside.

Net salary = Basic salary + Allowances +
Deductions.

Execution Errors:

Whenever the user pass wrong input as values then Runtime errors will be take place. When runtime error is occur the Compiler will take the control & gives predefined error messages. This type of concept is called as Error handling.

Ex:- 1. Dividing the number with Zero. 5%.

2. finding the square root of -ve no's, etc

WEDNESDAY

08

20 12

039-327 • Week 6

FEBRUARY

FEBRUARY 2012

S	M	T	W	T	F	S
				1	2	3
	5	6	7	8	9	10
	12	13	14	15	16	17
	19	20	21	22	23	24
	26	27	28	29		

predefined Header files:-

As the 'C' language source code contains diff types of functions. All these functions will have their own declarations. This declaration has been defined by the programmers & this all the declarations will maintain in category wise in different header files.

Diff types of header files are present & header files contain

1. Declaration of the function.
2. Predefined macros. (#'s)
3. User-defined datatypes.
4. Documentation of Header files etc.

- preprocessor directive

#include → file inclusion Macro.

#include <stdio.h>

This # will include this header file data into our program.

REPLY TO

→ The # is a preprocessor directive which substitute the prog code.

The #include → file inclusion macro which is defined with predefined header file.

MARCH 2012

M	T	W	T	F	S
			1	2	3
5	6	7	8	9	10
12	13	14	15	16	17
19	20	21	22	23	24
26	27	28	29	30	31

20 12

09

FEBRUARY

040-326 - Week 6

→ preprocessor directive
 #include < stdio.h >

↓ ↓
 - folder name header file
 - file inclusion macro

Examples:-

< stdio.h > Standard I/O P.

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

< conio.h > console I/O P.

Ex:- clrscr(), getch(), etc

" math.h "

Ex:- sqrt(), cos() etc

" string.h "

Ex:- strlen(), strcpy(), etc

Help — Ctrl + F1

Note:-In .c programs Header files are optional,
in .cpp if is Mandatory.

FRIDAY

10

2012

041-325 • Week 6

FEBRUARY

FEBRUARY 2012

S	M	T	W	T	F	S
				1	2	3
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29			

26/4/12

Internal Execution of a 'c' program:-

Whenever a .c programming is executing there are diff types of stages will be takes place, each stage will generate diff Extension files.

When a 'c' program is executing There are 4 stages will be takes place.

Text program

preprocessor

Compiler

Linker

The Text program is called as Source code.
Source code:-

The program that User can read Commonly Understandable as the program is called as Source code.

The Source code is an i/p to the preprocessor stage.

MARCH 2012

SUN	M	T	W	T	F	SAT
1	2	3				
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

20 12

11

SATURDAY

FEBRUARY

042-324 • Week 6

If the user clearly observe with the .c files we are sending the supporting files like header files

Our .c program will directly not takes place to the compiler since our program contain a predefined macros.

Main language not understandable for the 'C' compiler directly.

Preprocessors:-

The Preprocessor stage will remove all the Macro's & substitute only the high level language & generates one file with .i extension, called as Intermediate code.

This intermediate code file contains only high level language.

This .i file will pass to the 3rd stage

Compiler:-

The 3rd stage of the 'C' program directly not generate the objective code, the total 'C' program of every instruction,

will be converted into the assembly code, SUNDAY 12

& internally generate one file with an Extension of .ASM (Assembly)

MONDAY

13

20 12

044-322 • Week 7

FEBRUARY

FEBRUARY 2012

S	M	T	W	T	F	S
				1	2	3
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29			

This .ASM will be converted into objective code by the Assembler & generates objective file.

Objective file:-

The translation of the intermediate program into the Machine code which the computer can read & understandable directly is called as objective code.

When we executing the program & Compiling all the extension files wants to generates in your folder

Options

→ Directories.

output directory.

D:\BAT\7top\

Whenever the user press Alt+F9 automatically objective file is created.

REPLY TO

Note:-

→ When syntax errors are present objective file is not created.

MARCH 2012

SUN	M	T	W	T	F	SAT
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

20 12

14

FEBRUARY

045-321-Week 7

9:00 The objective files are i/p to the linker.
 9:00 Linker:-

10:00 A Linker is the 4th stage that links a program of separately compiled modules into one program, it also combines the functions in the standard 'C' library that the code which is already present. — the o/p of the linker is an executable file.

11:00 Note:- The .EXE file is Not generated when linker errors are existed.

12:00 Ex:- Miss spelling of predefined function.

1:00 `printf();`
 1:00 $\rightarrow \text{prntf}(); \rightarrow \text{linker error}$

REPLY TO

WEDNESDAY

15

2012

046-320 • Week 7

FEBRUARY

FEBRUARY 2012

S	M	T	W	T	F	S
				1	2	3
	5	6	7	8	9	10
	12	13	14	15	16	17
	19	20	21	22	23	24
	26	27	28	29		

~~7:30 AM / 12
8:00 AM~~

SUM. C

c preprocessor

SUM. I

Compiler

SUM. ASM

Assembler

SUM. OBJ

Lib. Obj

Linker

SUM. EXE → loader → CPU

REPLY TO

The off file of C program i.e. .EXE is
not portable from one O.S to another O.S.

The .EXE file which is generated
by the Windows O.S editor will not work
in Linux O.S.

Linux generates .out file

MARCH 2012

S	M	T	W	T	F	S
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

20 12

16

THURSDAY

FEBRUARY

047-319 • Week 7

Vice versa also the same problem. So the • Exe file is platform dependent of C & C++.

Operators:- An Operator is a symbol which will be manipulated on the operators.

In 'C' language there are diff categories of operators are present, but they are classify in two types.

1. Binary Operators.
2. Unary Operators.

The Binary Operators are the operators which will be manipulated on minimum of 2 operands.

Ex:- a+b;

Unary operators are the operators which will be manipulated on single operand.

Ex:- -a

Every operator in 'C' lang i.e 44 operators, will have hierarchy (preference) of operators. of some associate (which direction) - they have to solve.

REPLY TO

FRIDAY
17

2012

048-318 • Week 7

FEBRUARY

FEBRUARY 2012

S	M	T	W	T	F	S
				1	2	3
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29			

CategoriesOperatorsAssociativity

1. Highest Category

() [] →

Left to Right

function subscript points member @
call operator to array
number access.

(field)

2. Char Category

or is complement!

not Unary Unary preincrement
minus plus decrement

Right to Left

& * sizeof typecast
address of value at
address

3. Arithmetic

* / %
modulus

Left to Right

3.

+ -

L to R

4. Bitwise
Category

<< (Leftshift)

L to R

4. REPLY TO

>> (Rightshift)

L to R

5. Relational
Category

< <= > >=

L to R

5. 4

Equality (==) !=
Comparison Not equal to

L to R

20

MARCH 2012

S	M	T	W	T	F	S
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

20 12

18

SATURDAY

FEBRUARY

049-317 • Week 7

→ Bitwise
operators
category.

&
Bitwise And

L to R.

→ Bitwise
operators

Λ

(Bitwise exclusive OR)

L to R.

→ "

! (Bitwise OR)

L to R.

→ Logical
category.

&& (Logical And)

L to R

!! (logical OR)

L to R

→ "

→ Ternary (or)
Conditional
operator.

? :

R to L

→ Assignment
operator.

= (Simple assignment)

*=, /=, %=, +=, -=, <=, >=, & R+=.

REPLY TO
multiplication assignment *=, /=, +=

→ Comma operator.

,

SUNDAY 19

→ — — —

++ --

postincrement/postdecrement

MONDAY 20 2012

05i-315 - Week 8 FEBRUARY

FEBRUARY 2012

SUN	M	T	W	T	F	S
			1	2	3	4
	5	6	7	8	9	10
	12	13	14	15	16	17
	19	20	21	22	23	24
	26	27	28	29		

Arithmetic Operators:-

op in int op in float

Expression

y.d

y.f

$5/2$

2

2.000000

$2/5$

0

0.000000

$5/2.0$

2

2.5

double

\downarrow

if represented
as if it is 2.0/5

0

0.4

~~2.0f only~~
~~float~~

$2.0/5.0$

0

0.4

Expression

op%

$47 \% 5$

2

Denominator will be the
only effect numerator
will be zero

$47 \% -5$

2

$-47 \% 5$

-2

$-47 \% -5$

-2

if it is modulus not %
so it depends on
numerator value.

$5 \% 7$

5

$7 \% 7$

0

$7.0 \% 2.0$

X invalid.

Note:-

Floating point no's will not work with
% operator

MARCH 2012

S	M	T	W	T	F	S
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

20 12

21

TUESDAY

FEBRUARY

052-314 • Week 8

But C language supports one predefined function to work with floating point nos. & to get the o/p of a remainder - the function is called

`fmod()` — "math.h"

Ex:- `#include <math.h>`
`main()`

```

{ 
    float a=5.0, b=2.0;
    float s;
    s = fmod(a,b);
    printf ("%f", s);
    getch();
}
```

O/P: 1.00000.

→ Write the prog accept a 4 digit no display the no in reverse order?

$$\begin{array}{r} 10 \\ \overline{) 7548} \end{array}$$

70

50

40

40

```

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

{

`int num, x;` $\text{14d} \Rightarrow 4$ specify the 1^{st} four digits are taken & reverse
`printf("enter the 4digit Number: ");`
`scanf("%4d", &num); // 7548`
`x = num % 10; // 8` remainder.
`printf("%d", x); // 8` last no '8' will be printed here.

WEDNESDAY

22

2012

053-313 - Week 8

FEBRUARY

% — Takes remainder
 / — Takes coefficient

FEBRUARY 2012

S	M	T	W	T	F	S
				1	2	3
				4		
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29			

num = num / 10; // 754

x = num % 10; // 4

printf("%d", x); // 4

num = num / 10; // 75;

x = num % 10; // 5

printf("%d", x); // 5 O/P: Enter 4 digit no:

num = num / 10; // 7

7548

printf("%d", num);

8454

getch();

→ Write a program to accept a 4 digit no & find out sum of 4 digits.

#include <stdio.h>

#include "conio.h"

main()

{

int num, x, y, z, sum;

clrscr();

printf("Enter the 4 digit no: ");

x = num % 10; // 8 scanf("%d", &x);

num = num / 10; // 7

printf("%d",

num = num / 10; // 754

y = num % 10; // 4

num = num / 10 // 75

z = num % 10 // 5

num = num / 10 // 7

sum = x + y + z + num;

REPLY TO

MARCH 2012

S	M	T	W	T	F	S
			1	2	3	
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

20 12

23

22

THURSDAY

FEBRUARY

054-312 • Week 8

printf("%d", sum);
 getch();

5 (or)

main()

{

int num, x, sum = 0;

clrscr();

printf("Enter the 4 digit number: ");

scanf("%4d", &num); // 9561

X = num % 10;

sum = sum + X; // sum = sum + X

num = num / 10;

X = num % 10;

sum = sum + X;

num = num / 10;

X = num % 10;

sum = sum + X;

num = num / 10;

sum = sum + num;

printf("%d", sum);

getch();

{}

REPLY TO

24

2012

055-311 • Week 8

FEBRUARY

FEBRUARY 2012

S	M	T	W	T	F	S
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29			

→ Write a program accept a 5 digit no & find out the sum for 2nd & last digit? $56437 = 13$

A

#include <stdio.h>

#include <Conio.h>

main()

{

long

int num, x, y, sum;

clrscr();

if i.1d.

printf(" Enter the 5 digit number : ");

scanf("%d", &num); // 56437

x = num % 10; // 7

num = num / 100; // 5643

num = num / 100; // 56

y = num % 10; // 6

Sum = x + y; // 7 + 6 = 13

printf("%d", sum); // 13

getch();

}

No Need of
Variable if

sum = num % 10;

sum = x + sum;

sum =

sum +

MARCH 2012

S	M	T	W	T	F	S
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

20.12

25

MONDAY

FEBRUARY

056-310 - Week 8

→ Write a program accept & amount in Rupees & find out the denominations for that amount in 100's, 50's, & 10's & display the denominations.

```
#include < stdio.h >
#include < conio.h >
main()
{
    int num,x,y,z;
    clrscr();
    printf("Enter the amount in Rupees:");
    scanf("%d", &num); // 780
    x = num/100; // 7
    printf("%d", No.of 100's are : x); // 7
    num = num % 100; // 80
    y = num/50; // 1
    printf("%d", No.of 50's are : y); // 1
    num = num % 50; // 30
    z = num/10; // 3
    printf("%d", No.of 10's are : z); // 3
    getch();
}
```

REPLY TO

SUNDAY 26

MONDAY

27

2012

058-308 - Week 9

FEBRUARY

FEBRUARY 2012

S	M	T	W	T	F	S
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29			

22/4/12

#include < stdio.h >

#include " conio.h "

main()

int range = -65536

{

int a;

90000

clrscr();

24464 81

a = 300 * 300 / 300; // 90000

24464

printf(" %d ", a);

800

getch();

O/P: 81

}

→ We have a 16-bit editors like turbo-c &
32 bit editors like Borland-c, Linux editors etc.When a program is executing There are
2 types of memories will be taken place.

1. Compile time memory

2 Runtime memory.

For every bit editors there is a
maximum runtime memory will be taken place.
for suppose a 16-bit editor (1) O.S will have
a maximum space of 65536 (total length)
(0-65535)a 32-bit editors (2) O.S a maximum of
space of millions of space will be taken place
as in the above program, the expression
 $a = 300 * 300 / 300$

REPLY TO

MARCH 2012

S	M	T	W	T	F	S
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

20 12

28

24

FEBRUARY

059-307 • Week 9

it takes 90000, 90,000 out of lumspace.
 in order to get the value within one range
 we have subtract from the lumspace.

 -65536 90000 $\underline{24464} \quad 24464/300$ $= 81$

After subtracting we get a
 value of $24464/300$

Some times it is out of your datatype range
 but within the lumspace to get the value.

Once again you have to subtract from the
 lumspace.

 \rightarrow

Main()

{

int a, b;

 $a = 200 * 200 / 200;$ $b = 200 / 200 * 200;$

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

{

~~40000 is not in int range, -65536~~
~~so we got total strength~~
~~is in -65536~~

 40000 $\underline{-65536}$ $-32768/32767$ 40000 $\underline{-65536}$ $-25536/200$ 40000 $\underline{-65536}$ $= -127$ O/P:- a = -127

b = 200.

~~3~~
~~-65536~~
~~-160000~~
~~-5538~~
~~-55~~
~~100~~
~~100~~

REPLY TO

 \rightarrow

Main()

{

printf("%d", 300 * 200 / 100);

{

z0(p) - 55

WEDNESDAY

29

20 12

060-306 • Week 9

FEBRUARY

FEBRUARY 2012

S	M	T	W	T	F	S
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29			

3.

main()

32000

65536

+ 8536

33536

~~33536~~~~32000~~~~32768~~O/P:- 32000~~33536~~~~32768~~~~32768~~

4. main()

int + int = int

long a = 32767 + 1; → it is in inf range

printf("%d", a);

O/P:- -32768

5. main()

int int

65536

9

long a = 65536 + 300;

printf("%d", a);

3Max int
range: 65536

X 300

out of range

O/P: 300

fall from

180 * 5 / 9

180 / 5

180 / 9

180 / 5

180 / 9

180 / 5

180 / 9

180 / 5

180 / 9

180 / 5

180 / 9

180 / 5

180 / 9

180 / 5

180 / 9

180 / 5

180 / 9

180 / 5

180 / 9

180 / 5

180 / 9

180 / 5

180 / 9

180 / 5

180 / 9

180 / 5

180 / 9

180 / 5

180 / 9

180 / 5

180 / 9

180 / 5

180 / 9

180 / 5

180 / 9

180 / 5

180 / 9

180 / 5

180 / 9

180 / 5

180 / 9

180 / 5

180 / 9

180 / 5

180 / 9

180 / 5

180 / 9

180 / 5

180 / 9

180 / 5

180 / 9

180 / 5

180 / 9

180 / 5

180 / 9

180 / 5

180 / 9

180 / 5

180 / 9

180 / 5

180 / 9

180 / 5

180 / 9

180 / 5

180 / 9

180 / 5

180 / 9

180 / 5

180 / 9

180 / 5

180 / 9

180 / 5

180 / 9

180 / 5

180 / 9

180 / 5

180 / 9

180 / 5

180 / 9

180 / 5

180 / 9

180 / 5

180 / 9

180 / 5

180 / 9

180 / 5

180 / 9

180 / 5

180 / 9

180 / 5

180 / 9

180 / 5

180 / 9

180 / 5

180 / 9

180 / 5

180 / 9

180 / 5

180 / 9

180 / 5

180 / 9

180 / 5

180 / 9

180 / 5

180 / 9

180 / 5

180 / 9

180 / 5

180 / 9

180 / 5

180 / 9

180 / 5

180 / 9

180 / 5

180 / 9

180 / 5

180 / 9

180 / 5

180 / 9

180 / 5

180 / 9

180 / 5

180 / 9

180 / 5

180 / 9

180 / 5

180 / 9

180 / 5

180 / 9

180 / 5

180 / 9

180 / 5

180 / 9

180 / 5

180 / 9

180 / 5

180 / 9

180 / 5

180 / 9

180 / 5

180 / 9

180 / 5

180 / 9

180 / 5

180 / 9

180 / 5

180 / 9

180 / 5

180 / 9

180 / 5

180 / 9

180 / 5

180 / 9

180 / 5

180 / 9

180 / 5

180 / 9

180 / 5

180 / 9

180 / 5

180 / 9

180 / 5

APRIL 2012

SUN	M	T	W	T	F	S
	2	3	4	5	6	7
	9	10	11	12	13	14
	15	16	17	18	19	20
	22	23	24	25	26	27
	29	30				

2012

01

25

MARCH 061-305-Week 9

→ Write a program accept Two Variable Values Swap the Variable Values (interchange).

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

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

```
main()
```

```
{
```

```
    int a, b
```

```
    clrscr();
```

```
    printf(" Enter two values: ");
```

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

```
    swap(a, b);
```

```
    printf("%d %d", a, b);
```

```
    getch();
```

```
}
```

 $a = 5$
 $b = 7$
 $a = 7$
 $b = 5$
 $a = 7 = a;$
 $y = b;$
 $a = 7 = b;$
 $b = t;$
 $a = 7 = t;$
 $b = 5;$
 $a = 7 = 5;$
 $b = 5 = 7;$
 $a = 7 = 7;$
 $b = 5 = 5;$

$$b = (a+b) - (a=6)$$

$$7 - 6 = 1$$

$$\begin{array}{r} a \\ \boxed{7} \\ + b \\ \hline \boxed{2} \end{array}$$

$$b = 7 - 6$$

$$b = 1$$

$$b = 1$$

$$\text{Logic } ③$$

$$a = a * b;$$

$$b = a / b;$$

$$a = a / b;$$

REPLY TO

02 2012
062-304 - Week 9 MARCH

MARCH 2012

S	M	T	W	T	F	S
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

- Write a program accept radius, find out area & perimeter of circle
 → Write a programs accept the centigrade find out the farenheit value?

Relational operators:-

a Relational Expression is taking place by a valid combination of numeric Variables, Numeric Constants & Relational (~~Op~~) equality operators.

either 1 (or) 0.

Whenever a NonZero value occurs it returns 1. (~~True~~) (True)
 if zero value occurs it returns 0.

	T	F
1		
0.1		0
-5.6		
'A'		
-7		

main()

{ int a;

a = 5 > 3;

printf("%d", a);

getch();

}

— o/p: 1

printf("%d"; 8 == 8); — 1

APRIL 2012

SUN	MON	TUE	WED	THU	FRI	SAT
15	16	17	18	19	20	21
16	17	18	19	20	21	22
17	18	19	20	21	22	23
18	19	20	21	22	23	24
19	20	21	22	23	24	25
20	21	22	23	24	25	26
21	22	23	24	25	26	27
22	23	24	25	26	27	28
23	24	25	26	27	28	29
24	25	26	27	28	29	30

MARCH

063-303 - Week 9

2012

03

26

2. main()

{ int a;

a = 5 > printf("%d", a); — 1

{}

3. main()

{ int a;

a = 4 > printf("%d", a); — 0

{}

4. main()

{ int a; Left to Right —

a = 10 > 9 > 8;

printf("%d", a); — 0

{}

5. a = 10 > ; // invalid eng Binary operator needs 2 operand.

REPLY TO 6. a = 6 / = 8 — 1

7. a = 6 / = 6 — 0

SUNDAY 04

8. 5 > 4 > 3 > 2 > 1 > 0 > -1 — 1

9. a = 5 > 4 > 3 > 2 > 1 — 1

5 > 4 ⊕ 5 > 4

5 > 3 > 4
High preference

MARCH 2012

SUN	M	T	W	T	F	SAT
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

10. $a = \underline{(5+4)} + \underline{(5+4)};$
 1 + 1
 = 2

11. main()

{

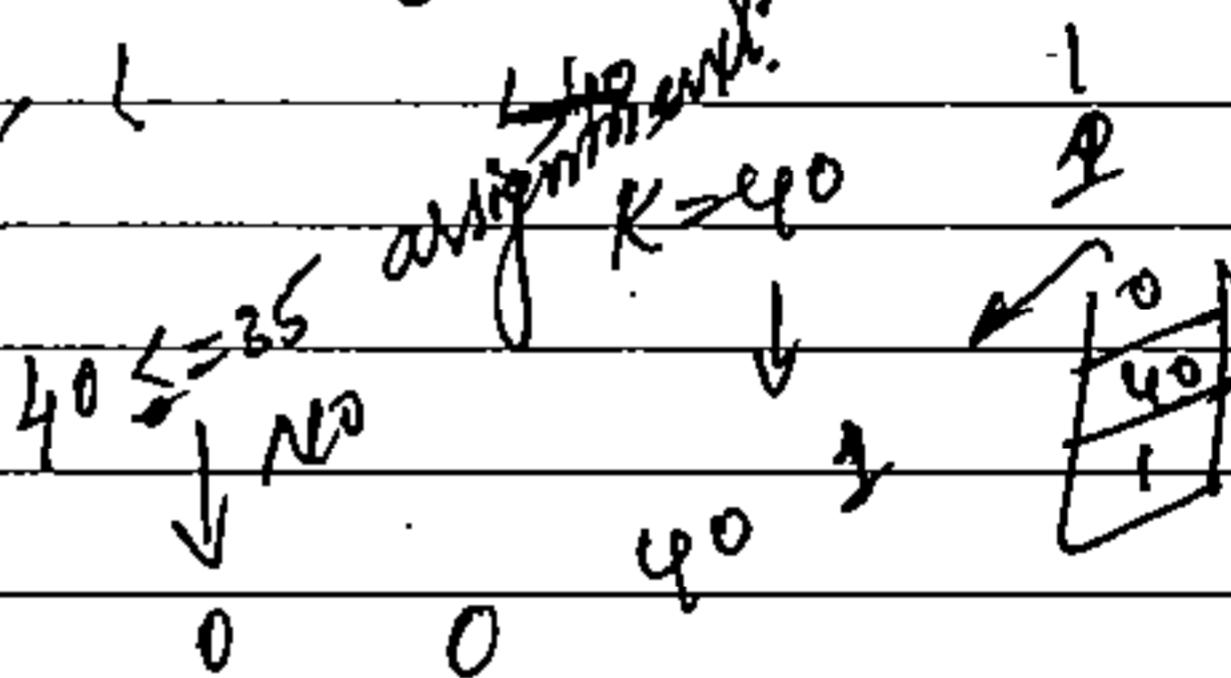
int K=35;

printf("%d %d %d", K<=35, K=40, K==35);

getch();

}

← Right to left workflow.

O/P: 0, 40, 1Logical Categories:-

a Logical Expression is a valid combination of logical values, variables & operators.

The logical values can be combined with relational Expressions.

REPLY TO

The logical operators also returns either 1 or 0

Logical NOT

Operands	A	R	Result
	0	1	
	1	0	

APRIL 2012

SUN	M	T	W	T	F	S
	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30					

2012

06

TUESDAY

MARCH

066-300 • Week 10

Logical And &&

Logical OR |||)

A	B	R
0	0	0
0	1	0
1	0	0
1	1	1

A	B	R
0	0	0
0	1	1
1	0	1
1	1	1

1. $a = 5 > 4 \& \& 4 > 3;$
 1 && 1
 =

2. $a = 7 \& \& 8;$

7 && 8 define NonZero

1

i.e. 1

3. $a = 5 > 4 \& \& 5 > 10; — 0$

4. $a = 7 || 0; — 1$

(0)

1 || 0

=

REPLY TO

$5 \cdot a = 5 > 4 || 5 > 10; — 1$

6. $a = 1 + 2 \& \& 2 + 3; — 1$

7. $a = 1 + (\underline{2 \& \& 2}) + 3 — 5$

1

1 + 3 = 5

WEDNESDAY

07

2012

067-299 • Week 10

MARCH

MARCH 2012

	M	T	W	T	F	S
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

8. $a = 6 > 6 \text{ || } 5 > 1$

$$a = \frac{6 > 6 \text{ || } 5 > 1}{0 \text{ || } 0} = 0$$

9. $a = !1 = 0$ (! Logical Negation operator)

10. $a = !-67 = 0$ — The Result of the type is int.

11. $a = !-6.45 = 0$ // it returns 0 when the operand is NonZero.

12. $a = !-0.57 = 0$ // it returns 1 when the operand is Zero.

13. $a = !0 = 1$

1st preference

14. $a = \underline{\underline{!-3 > 3}}$ — The Expression !E is evaluated to (0 == E) if it's true returns 1 otherwise 0

15. $a = !(-3 > 3)$

$$!0 = \underline{\underline{1}}$$

Unary Category:-

REPLY TO

sizeof() — The sizeof operator is used to estimate the size of a datatype (as) Variable, function etc. according to the os

APRIL 2012

SUN	M	T	W	T	F	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30					

20 12

08

THURSDAY

MARCH

068-298 • Week 10

8.00 →

#include <limits.h>

#include <float.h>

main()

{

double s;

printf("Size of integer: %d\n", sizeof(int));

printf("Size of float: %d\n", sizeof(float));

printf("Size of character: %d\n", sizeof(char));

printf("Size of S: %d\n", sizeof(s));

printf("Max of integer: %d\n", INT_MAX);

printf("Max of float: %f\n", FLT_MAX);

getch();

{

O/P:

Size of integer: 2

Size of float: 4

Size of Char: 1

Size of S: 8

Max of integer: 32767

Max of float: 3.40282e+38

→

REPLY TO

main()

{

char k;

printf("%d %d %d %d %d %d", sizeof(45), sizeof(A),
sizeof(1.0), sizeof(32768), sizeof(C), sizeof(0x23))

{

double

O/P: 2 2 8 4 1 2

A
1Hekade
malis tape:
end.

09

2012

069-297 - Week 10

MARCH

MARCH 2012

SUN	M	T	W	T	F	S
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31



main()

{

char a, b;

float c, d;

printf("%u.%u%u", sizeof(a+b), sizeof(b+c),
sizeof(c+d));

}

O/P: 2 4 4

main()

{

printf("%u.%u", sizeof(3.14+67),
sizeof(3.14+67));

{

O/P: 8 751st preference sizeof 3.14

Type Casting:

8+67=75

because

8

double takes 8 byte

Converting from one datatype to another
datatype.

REPLY TO

There are 2 types of Castings

1. Implicit Type Casting
2. Explicit Type Casting

APRIL 2012

SUN	M	T	W	T	F	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30					

20 12

10

SATURDAY

MARCH

070-296 • Week 10

1. Implicit Type Casting:

If - The Compiler itself Convert from One type to Another type is Called implicit type casting.

Ex:-

main()

{

int s = 65;

printf("%c", s);

getch();

}

s is integer

s is char

but s.c is ASCII

Compiler convert ASCII

65 ASCII is A

O/P = A

2. Explicit Type Casting:

If the Compiler Unable to Convert from One type to Another type then the User Explicitly has to Convert to Another type.

Ex:-

long K;

K = 32767 + 11; // explicit

int m1, m2, m3, tot;

float Avg;

tot = m1 + m2 + m3;

Avg = tot / 3.0; // explicit

(as)

Avg = tot / 3f; // explicit

SUNDAY 11

REPLY TO

MONDAY
12

20 12

072-294 • Week 11

MARCH

MARCH 2012

S	M	T	W	T	F	S
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

Syntax:-

Var - name = (datatype) expression;

8.00 ~~int~~ main ()

{

10.00 int ~~x, y~~; x=5, y=2;float quo; \rightarrow explicit type casting.11.00 quo = (float) x / y; ~~return quo~~12.00 printf ("%.2f", quo); \rightarrow to get a proper output.

getchar();

1.00 Updation Operators:-

2.00 The value of the variable will be updated in the memory locations.

3.00 num = num / 10; // assignment-

4.00 Here Two operators

(Q8)

/, =

5.00 num /= 10; // updation.

6.00 \rightarrow Here only one operator /=~~num~~

REPLY TO Sum = sum + num % 10;

(Q8)

Sum += num % 10;

T - 1st preference

2nd preference

APRIL 2012

M	T	W	T	F	S
2	3	4	5	6	7
9	10	11	12	13	14
16	17	18	19	20	21
23	24	25	26	27	28
29	30				

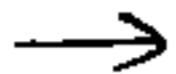
20 12

MARCH

13

30

073-293 • Week 11



main()

{

```

int a=10;
printf("x.d\n", a);
a=a+1;           // Updatable Stmt
printf("x.d\n", a);
a+=1;           // Updatable Stmt
printf("x.d\n", a);
a+1;           // Non Updatable Stmt
printf("x.d\n", a);
printf("x.d\n", a+1); // Non Updatable
printf("x.d\n", a); // Here printf only
                    // Updatable data displayed
                    // but actually it will not change
                    // the memory location.

```

/* Memory locations */

1. a=a+1;

2. a+=1;

3. a++;

4. ++a;

5. --a;

6. a--;

int x=10;

UPDATES

x+=5;

x*=5;

x/=5;

x%5; x-=5;

Assignments

x+=x+5;

x=x-5;

x=x*5;

x=x/5;

x=x%5;

WEDNESDAY

14

2012

074-292 • Week 11

MARCH

MARCH 2012

S	M	T	W	T	F	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

InClement Decrement Operator:

(imp)

Note:-

01P
a++ ; The result of the Expression is
the Original value of a.

11.00
++a ; The result of the Expression is The
Incremented value of a.

1.00
a-- ; The result of expression is the
Original value of a.

2.00
--a ; The result of Expression is the
Decremented value of a.

4.00
Type 1:

5.00
#include <stdio.h>

main()

{

6.00
int x=5;

REPLY TO
printf(" value of x=%d\n", x++);

printf(" value of x=%d\n", --x);

printf(" value of x=%d\n", *++x--);

printf(" value of x=%d\n", ++x);

printf(" value of x=%d\n", x++);

} getchar;

O/P:- 5 5 5 5 5

APRIL 2012

S	M	T	W	T	F	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30					

20/12

15

THURSDAY

MARCH

075-291-Week 11

Type 2:

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

{

int x=6;

Right to Left



printf("%d %d %d", x++, ++x, x--);

{

}

. x- 6 7 8 9Solving procedure:

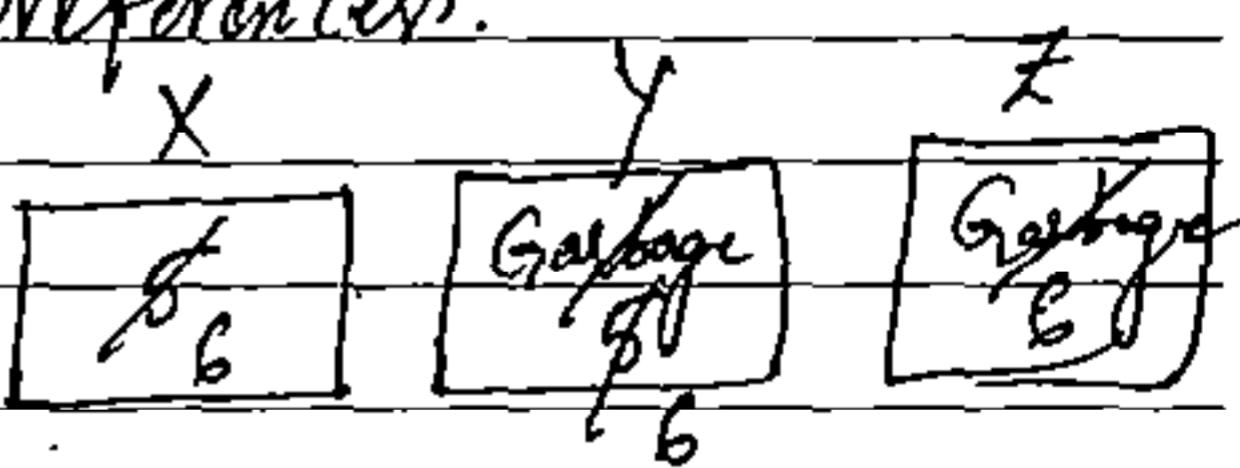
3, 6, 6

- Evaluation Starts from Right-most Expression.
- push the result of that expression onto stack
- use the New value of x to find the result of next Expression.

Type 3:- based on operator preferences.

main()

{



int x=5, y, z;

Here or pre increment

printf("x:%d y:%d", x, y); 6, 5

value of z = ++y; → pre increment is preference

printf("y:%d z:%d", y, z); 6, 6

REPLY TO
assignment
garbage

{}

When we evaluating the Expression with post increment operators, I simplify the expression, (remove the post-increment, once again write the stmt)

FRIDAY

16

2012

076-290 • Week 11

MARCH

MARCH 2012

	M	T	W	T	F	S
1	2	3				
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

If the Expression is present within the function Only if Can be Type 1, Type 2, Type 4.

printf("d\n", x);

7	y	12
6	6	6

printf("d\n", ++x);

8 ← ++x; // if is Type 3 because it not given in

printf("d\n", x); // a printf() function.

Op: 9 ← x++ ; // Here Type 3 ; is Executed Stmt is

printf("d\n", x); Complete

printf("d\n", x++);

printf("d\n", ++x); → 11

printf("d Y.d d\n", ++x, x--); 11, 11

printf("d Y.d Y.d Y.d", X++, X; ++x, x--);

X	X	X	X
10	11	11	12

6+

a

b

fp
100

Garbage
100

Type 3 →

main()

{

int a=6, b;

b = a++ + a++;

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

{ } ②

③

5

remove

b = a + a

10 = 5 + 5

APRIL 2012

SUN	M	T	W	T	F	SAT
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30					

20 12

17



077-289 • Week 11

MARCH
6

a

[Garbage]
[14]

→ main()

{ }
int a=5; b;

$$b = \cancel{+} a + \cancel{+} a + \cancel{+} a;$$

printf("a: %d b: %d", a, b);

$$b = a + a;$$

$$b = 7 + 7 = 14$$

{ }
}

12 a b

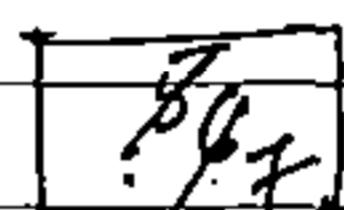
→

main()

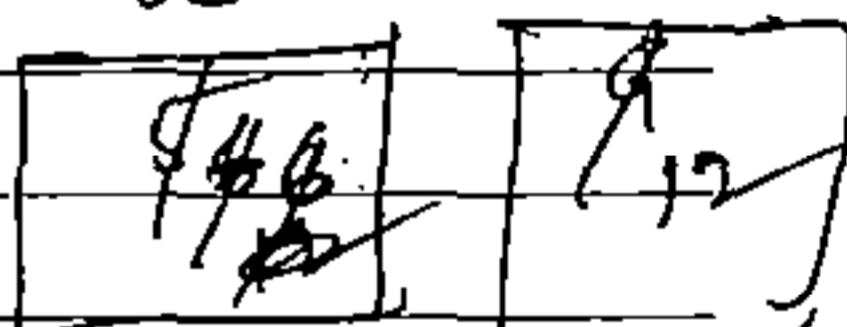
{ }
int a=5, b;

$$b = + a + a + +;$$

printf("a: %d b: %d", a, b);

{ }
}[garbage]
[6]

$$a = 7, b = 12$$



1. left preference to pre-increment.

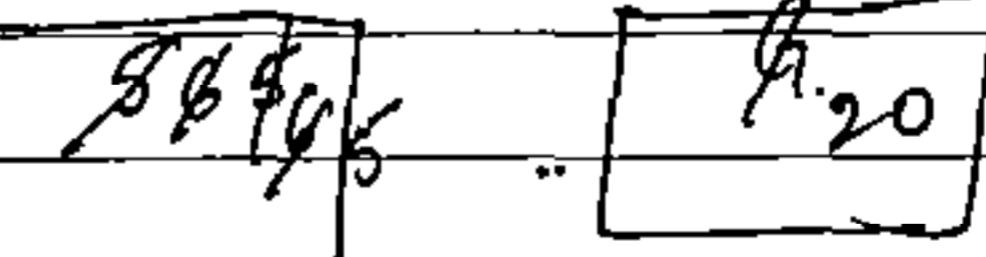
2 - then +

7

3 =

4. post-increment

a b



main()

REPLY TO

{ }
int a=5, b;

$$b = a + f + --a + a + a - - + + a;$$

printf("a: %d b: %d", a, b);

{ }
}

$$a - - a + a + + a;$$

SUNDAY 18

19

2012

079-287 • Week 12 MARCH

MARCH 2012

SUN	M	T	W	T	F	S
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

a

6

16 X
24

main()

{

int a=10, b;

class();

a + ++a;
b, a, a, ++a.

b = a + + + a;

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

} getch();

0/p: 22, 13, 13, 13

→ Main()

{

int x=5, y=4, z=3;

z = y + + + x --;

printf ("x=%d y=%d z=%d\n", x, y, z);

4, 5, 2

x = -4; y += 5;

z = y -- + x ++;

printf ("x=%d y=%d z=%d\n", x, y, z);

-3, 9, 6.

{}

REPLY TO

①

x
-4y
*5z
8 9 6

printf

x

y

z

printf ②

x
-4-3y
9z
6

APRIL 2012

S	M	T	W	T	F	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30					

20 12

20

TUESDAY

MARCH

080-286 - Week 12

 $\rightarrow \text{main}()$

{

int $Z = X = 5, Y = -10, a = 4, b = 2;$

Class(80): ① ② ③

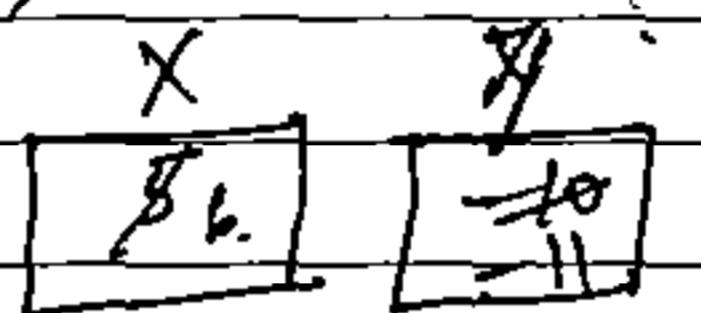
 $Z = X + + - - Y * b / a;$

printf("Y.d", Z);



{

O/P: 10



$$\begin{aligned}
 & -11/2 \quad X - - Y \\
 & = 5 * 2 / 4 * 2 \quad X \cdot 5 - 6 \\
 & = 5 \quad - 10 \\
 & = 10
 \end{aligned}$$

Solving procedure:-

1. Check the prefix expression.

2. Substitute the Value to every expression.

3. Check the post-fix expression.

Type 4:- #include < stdio.h >

[4]

$$\begin{aligned}
 & \{ \quad 4 \rightarrow 86 \quad c = 4 \times 6 \times 6 \\
 & \text{int } x = 4; \quad \cancel{4} \quad \cancel{6} \quad \cancel{6} \quad = 144 \\
 & \text{printf("Y.d", } x + + * + + x * x --);
 \end{aligned}$$

{ Starts from left most expression &
 Execute each expression & every expression.

WEDNESDAY

21

20 12

081-285 • Week 12 MARCH

MARCH 2012

S	M	T	W	T	F	S
1			1	2	3	
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

Note:-

As an Type 4 is based on the Linux operating system the off will varies.

Increment & Decrement Operators by Using Relational, logical operators (Type 5)

Note:-

According to the || (OR) operator Before Condition is Satisfy it doesn't check the remaining Condition.

According to the && (And) operator if before Condition is dissatisfy it doesn't check the remaining Condition.

Ex:-

Main()

{

int a, b, c;

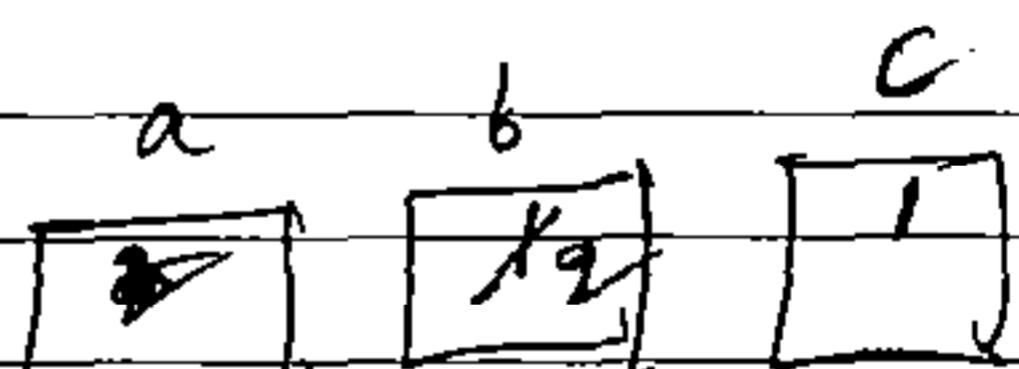
b = c = 1;

Class();

a = f + b > 1 || ++c > 1;

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

{



a=1, b=2, c=1

here a check

++b is set and then assigned to 'a'.

Condition compiler will assign this to 'a'.

Condition compiler this to 'a'.

APRIL 2012

S	M	T	W	T	F	S
5	1	2	3	4	5	6
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30					

2012
04 22

MARCH

082-284 • week 12

22

34

$$\rightarrow b = c = 1 \geq 1 \quad a = 1 + c \geq 1 \quad c$$

$$b = c = 1 \geq 1 \quad a = 1 + c \geq 1 \quad c$$

$$a = 1, \quad b = 2, \quad c = 2.$$

$$a = 1, \quad b = 2, \quad c = 2$$

$$a = 1, \quad b = 2, \quad c = 2$$

$$a = 1, \quad b = 2, \quad c = 2$$

$$a = 1, \quad b = 2, \quad c = 2$$

$$a = 1 + b \geq 2 \quad a = 1 + c \geq 2$$

$$a = 1 + b \geq 2 \quad a = 1 + c \geq 2$$

$$a = 1, \quad b = 2, \quad c = 1, \quad d = 1$$

REPLY TO

083-283 • Week 12

2012

23

MARCH

MARCH 2012

S	M	T	W	T	F	S
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

→ $b = c = d = 1;$

$a = (++b > 1) || (c++c > 1 - \cancel{--} \cancel{d} > 1);$

21
1

a b c d
1 $\cancel{\frac{1}{2}}$ 1 1

→ $b = c = d = 1;$

$a = (++b > 1) \cancel{||} (c++c > 1) \cancel{--} \cancel{d} > 1;$

21

2

a b c d

initially if $++b > 1$ returns $\cancel{+a}$ of $+8^4$ [1] [2] [1] [2]

check if $a = 1$ satisfies $a = 1$. so, $a = 1, b = 2, c = 1, d = 2$

if $++b > 1$ satisfies -8^4

if $++b > 1$ after x^8 x^8 x^9
check $a = 2$ $\cancel{a = 2}$

5.00 preference

→ main()

8
int a;

class();

a = 10;

a* = 10 + 2;

printf("x.d", a);

}

a = 120

a
[10
120]

~~a = 10 + 2;~~
~~a* = 12~~

APRIL 2012

	M	T	W	T	F	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30					

20 12

SATURDAY

24

MARCH

084-282 • Week 12

 $\rightarrow \text{main}()$

{

int a;

a = (3, 8+9, 5);

printf(" %d ", a);

{

o/p: a = 5

a

3 17 5

1st preference is =

assignment
after , (left to right assignm
it will assign the values
one after another.

initially a = 3

for example

a = 3, 8+9, 5; also out. It is replaced with 5

a = 3. & 3 also out. So a = 5.

generate out effect so a = 5.

code has no effect.
So based on base value.

So ()

3 is replaced with 8+9=17.

17 is replaced with 5

** Note:-

Every operator after performing some operations it returns some value.

Similarly a function also returns a value.
Receiving the value of the function is optional.The function printf() always returns the 1st argument value. i.e. No. of characters.

SUNDAY 25

Similarly the scanf() also returns the first argument i.e. No. of format specifications.

MONDAY

26

2012

086-280-Week 13

MARCH

MARCH 2012

	M	T	W	T	F	S
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

Ex:-

main()

{

int x;

x = printf("Hello");

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

getchar();

{

O/P: Hello

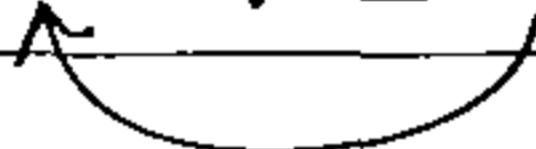
5

→ main()

{

printf("%d", printf("Hello"));

{



O/P: Hello 5

inner printf() will be performed
 & it will print Hello
 printf() function also returns
 no. of characters so

Hello is 5 characters

5 is return to %d

because %d is format specification

REPLY TO

1. → main()

36

{ printf("%d", printf("%d %d", 10, 20));
}

return

space

printf("%d", printf("10 20hi"));

→ printf returns no. of characters also.

10 20hi
1234567
space. i.e = 7

2. main()

{ int a;

clrscr();

a = printf("Hai %d Kisan", printf("SQL"));

printf("%d", a);

3

1. 1st argument printf is evaluated.

outer printf returns print SQL on screen.

Output : value 2. Then it returns a value 3

& stores to a.

i.e passed to %d

a = Hai3Kisan. print - SQLHai3Kisan.

Count it again it i.e 9.

SQLHai3Kisan9.

3. #include <stdio.h>

main()

{ int a, b, c;

return 14 to a.
 $a=14$

a = printf("Welcome Naresh"); O/P: Welcome Naresh14

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

$a=14$.

// Enter two values 20 30

$b=2$ (because Two %d
specification)

printf("In %d %d %d", a, b, c);

$c=20$

i.e 1st entered value

```

4. #include <stdio.h>
main()
{
    int a,b,c;
    b=c=100;
    printf("In %d %d %d", a,b,c);
    a = printf("welcome Naresh %d", printf("Hello Kiran"));
}

```

printed from left to right

a b c

16 Garbage 100 100

Hello Kiran welcome Naresh $\frac{a=16}{16}$

// at last because in " " %d

at last.

```

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

```

// Entered Values ~~are 20 30~~

scanf has many

no. of specifications that no. will defer.

a = In 16 100 100 ~~100 100~~

$a=13$

```

printf("In %d %d %d", a, b, c);

```

$a=13 \quad b=100 \quad c=100$

}

Control Structures

where implementing the programming upto eight time by executing every stmt without ignoring any stmt.

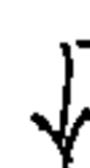
Such type of programming is called as Sequence-

But always this sequence programming

Can't been implemented then we have to take the decisions we will provide options to choose then based on the user decisions programming stmts executed.

In This programming Some Stmt will execute & Some Stmt will ignore.

Control Structures



Decision Making

(conditional stmts)

Ex:- Conditional Operator (Ternary)

if

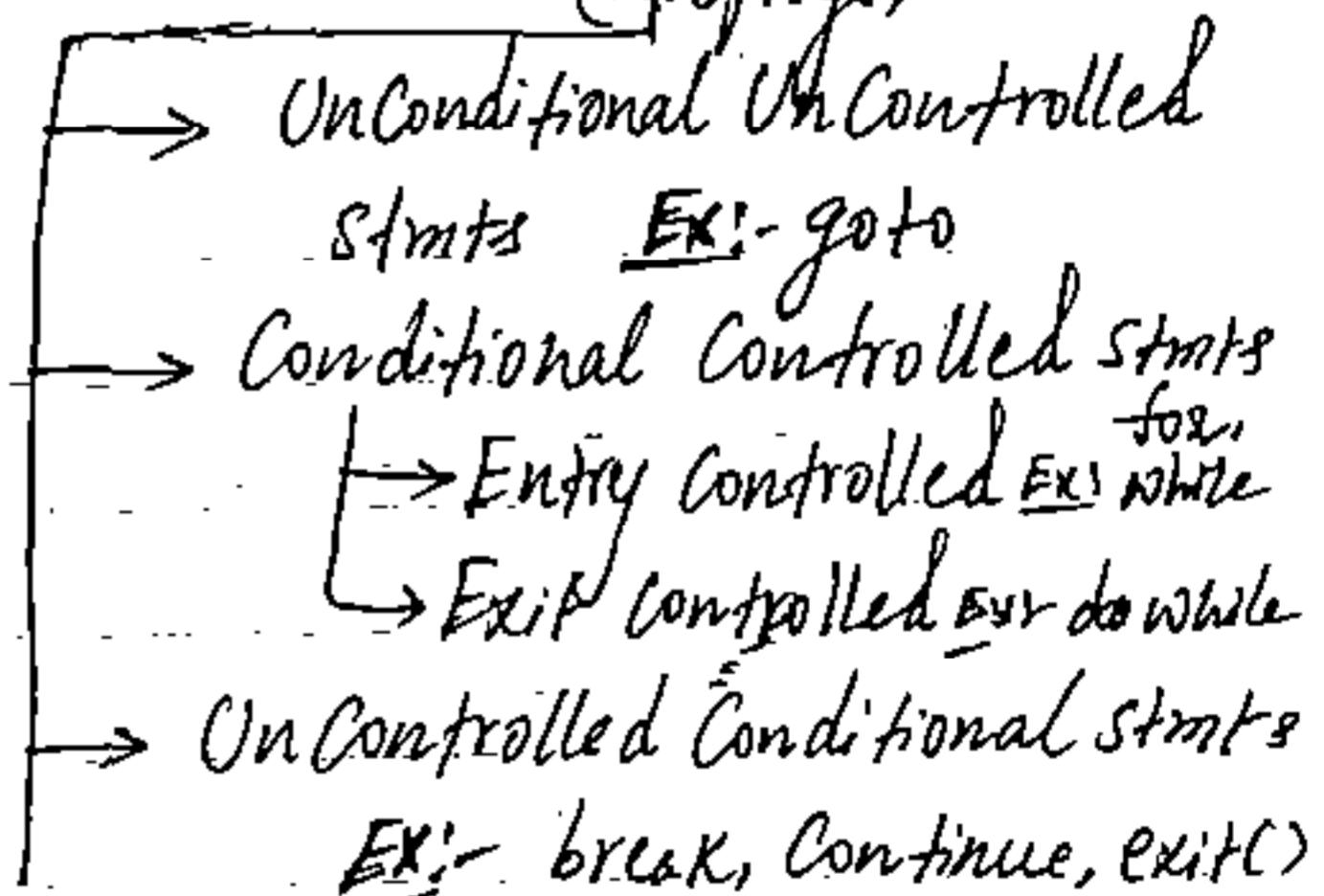
if else

Nested if's

Switch

Iterations.

(Loopings)



→ A Ternary operator is also called

Conditional operator it uses Two Special Symbols.

? And :

Either it is an operable it acts as Conditional
stmts.

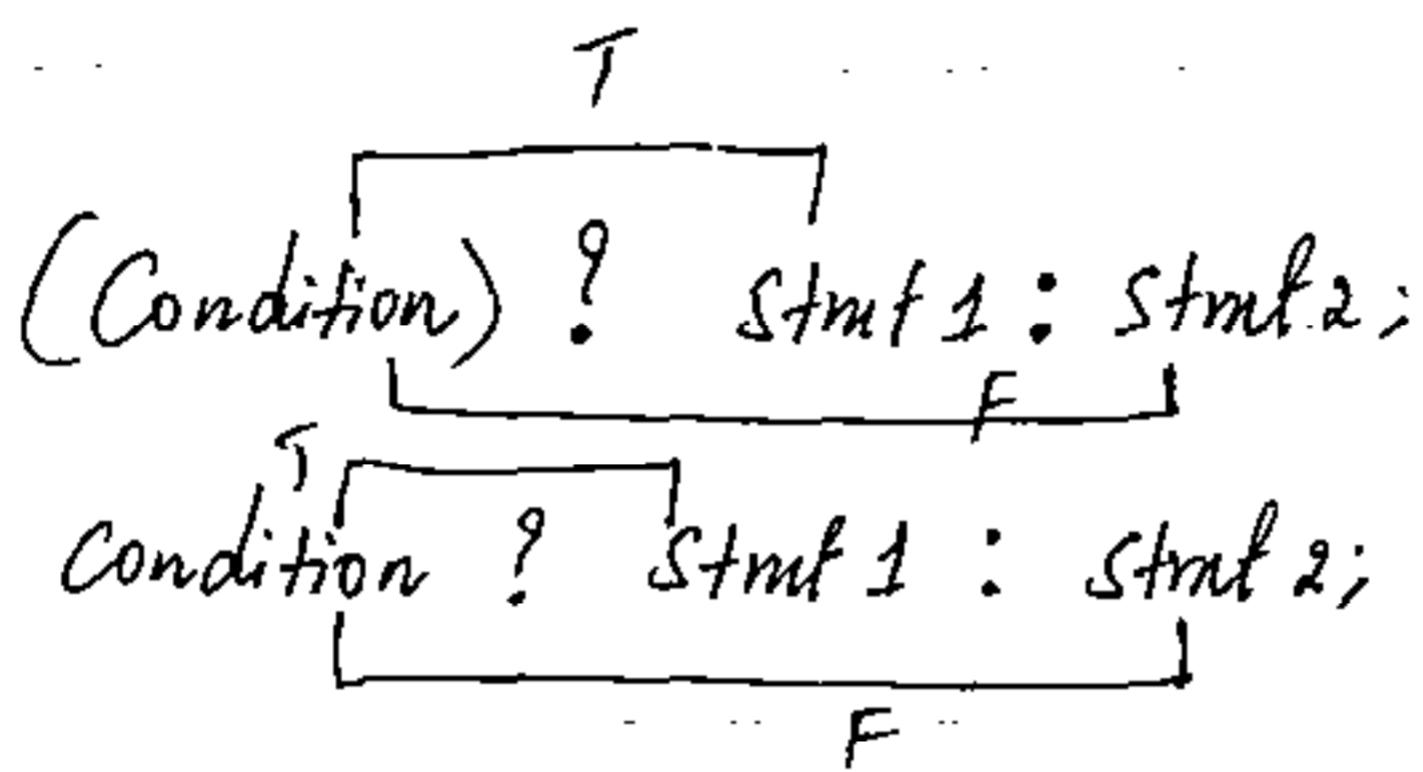
Based on the Condition the stmts will be Executed.

If the condition is Satisfied, if execute the Statement after the ?

If Condition is Not Satisfied if execute the Stmt's after the :

In either of the case only One Stmt is Executed.

Syntax:-



prog: Write a program accept a No. Findout the given no is Even or odd.

Main()

{

int num;

printf("Enter the Num:");

scanf("%d", &num);

(num % 2 == 0) ? printf("Even") : printf("Odd");

getch();

}

prog Write a program accept the age of the person & indent the person age vote (or) not.
 #include <stdio.h>

main()

{

int age

printf("Enter the age of the person:");

scanf("%d", &age);

if (age ≥ 18)

{

printf("vote is valid");

else

printf("Vote is not valid");

}

getchar();

}

→ main()

{

int a;

a = 324910:20;

printf("%d", a);

getchar();

O/P: a=2

}

→

a = 1923;

O/P a=2.

// sign Singt value itself is Nonnegative
 return 1. (true)

$\rightarrow a = 3 > 2 > 1 ? 10 : 20; \quad - \text{O/P: } 20.$

$\rightarrow a = 1 + 2 ? 2 + 3 : 3 + 4; \quad - \text{O/P: } 5$

$\rightarrow a = 7 ? 8; \quad - \text{Invalid } (:) \text{ missing.}$

$\rightarrow a = 1 : 2 ? 3 \quad - \text{Invalid (Not follow order) } (? : ;)$

$\rightarrow a = 3 > 2 \& 2 > 1 ? 10 : 20; \quad - \text{O/P: } 10$

$\rightarrow a = ! - 4 > 4 ? 1 : 2; \quad - \text{O/P: } 2$

Nested Ternary Expressions:-

A Nested Ternary Expression is evaluated from right to left.

In N.T. Expression - the evaluated value is substituted in the place of ternary expression & the evaluation continues until all the ternary expressions evaluated.

Ex:- $a = 3 > 4 ? 5 * 6 ? 1 : 2 : 10 > 8 ? 3 : 4$

$$a = \underline{\underline{3 > 4}} \quad ? \quad \underline{\underline{5 * 6}} \quad ? \quad \underline{\underline{1 : 2}} \quad : \quad \underline{\underline{10 > 8}} \quad ? \quad \underline{\underline{3 : 4}}$$

$$a = \underline{3}$$

Always Nearest colon(:) goes to nearest question mark(?)

$\rightarrow a = 5 > 6 ? 10 : 6 > 4 ? 20 : 30;$

↓
flare : 320

$$a = 20$$

$\rightarrow a = 5 > 4 ? \underbrace{6 > 7 ?}_{20} \underbrace{10 : 20 : 30 ;}_{20}$

$\rightarrow a = 3 > 4 ? \underbrace{2 > 1 ?}_{3 > 4 ?} \underbrace{10 : 20 : 5 > 4 ?}_{10} \underbrace{30 : 40 ;}_{30}$

$a = 30$

$\rightarrow a = 3 > 4 ? \underbrace{2 > 1 ?}_{8 > 7 ?} \underbrace{10 : 20 : 5 > 6 ?}_{8 > 7 ?} \underbrace{30 : 40 : 50 : 60 ;}_{60 : 60}$

$5 > 6 ? 30 : 50 ;$

$3 > 4 ? 10 : 50$

$a = 50$

$\rightarrow a = 2 ? 1 ? 0 ? \underbrace{10 : 20 : 30 : 40 ;}_{1 ? -20 : 30}$

$2 ? \quad 20 : 40$

20

$a = 20$

-6

$\Rightarrow num * 1 =$

$num * -1 = -num ;$

$num * -1 = num ;$

Qn: Write a program accept a number find out the given no is +ve, -ve, (or) Zero.

main()

{ int num

printf("Enter the num: ");

scanf("%d", &num);

(Right to left)

(num>0) ? printf("+ve") : (num<0) ? printf("-ve") :
printf("Zero");

getch();

→ Write a prog. accept 4 numbers find out the minimum value
Using Ternary operator 40

→ Write a prog accept 3 numbers, display that ascending (0x)
descending order $b = (a + b) - (a \geq b)$

→
 main()
{
 int K, num = 30;
 K = (num > 5 ? (num <= 10 ? 100 : 200) : 500);
 printf("%d", num);
 } 200. K > 200.
 num = 30

→ accept 3 no's display that in ascending (0x)
descending order.

~~ascending order~~ main()
{
 int a, b, c;
 clrscr();
 printf("Enter Three numbers :");
 scanf("%d %d %d", &a, &b, &c); // 45 78 34
 (a > b) ? b = (a + b) - (a = b) : ;
 (b > c) ? c = (b + c) - (b = c) : ;
 (b > a) ? b = (a + b) - (a = b) : ;
 printf("%d %d %d", a, b, c);
 getch();
 }

Advantages: Ternary operator:

Advantage of the Ternary Operator we can finish the program in a single line of Stmt. It is more flexible for writing the programs.

Drawback:-

In Ternary Operator after the ? (as) : we will find only Single Stmt. it is never possible to write multiple stmt's in Ternary Operator.

To overcome this problem only if we implemented if Condition.

Interview pattern

Main() ✓

$$K = 12 \quad n = 30$$

int K=12, n=30;

K = (K > 5 & & n = 4) ? 100 : 200; $K > 5 \& \& 200$

printf ("K=%d", K); Error.

} Since L value required.

L Value:

An L Value is any variable whose value can be changed (assigning a new value) an L Value is a ~~value~~ Variable, ~~for~~ whose value can't be change.

$$\begin{aligned} & K > 5 \& \& n = 4 ? 100 : 200 \\ & \text{next preference} \quad 1 \& \& n = 4 \\ & 1 \& \& 30 \quad 1 \& \& 30 = 4 \\ & \text{Non-(rc)} \quad 1 = 4 \quad \underline{\text{error}} \end{aligned}$$

41

As the precedence to the operators, rather than assignment operator is higher priority, & and the Condition will be defined as $i=4$
 the left side can't be a Constant Number. So there is no L value. Hence 4 can't been assigned.

8/05/11

if

```
if ( condition )
  Stmt 1;
  Stmt 2;
```

Condition	Stmt 1	Stmt 2
T	✓	✓
F	✗	✓

The 'if' is decision making Stmt, based on the Condition, the Stmt's will be evaluated.

```
main()
{
```

```
  if(5<6)
    printf("right");
    printf(" wrong");
    getch();
}
```

O/P: right
wrong

```
main
{
```

```
  if(5>6)
    printf("right");
    printf(" wrong");
    getch();
}
```

O/P: wrong

In the 1st case both the Stmt's are executing, it is not the propolity of O/P.

In 2nd case Only one Stmt will executing, but the Condition is satisfied both Stmt's executing, to execute single Stmt we have to use if else

if else:

```
if (condition)
    Stmt1;
else
    Stmt2;
```

Condition	stmt1	stmt2
T	✓	✗
F	✗	✓

→ else is Not Compulsory ~~with~~ when if is present.

→ if (condition)

 Stmt1;

else

 Stmt2;

 Stmt3;

 ⋮

 Stmtn;

Condition	stmt1	stmt2	stmt3	⋮	stmtn
T	✓	✗	✓	✓	✓
F	✗	✓	✓	✓	✓

→ Write a program to accept basic salary of emp if the salary greater than 20,000, deduct the tax 10% by 20%, else 5%, display the total salary & tax.

main()

{

 float bs, tax;

 printf("Enter the Basic Sal: ");

 scanf("%f", &bs);

 if (bs > 20000)

 tax = 20 * bs / 100;

 else

 tax = 5 * bs / 100;

```

printf(" Tax Deduction amount: %.2f\n"
      " On the basic sal: %.2f ; tax, %s");
getch();
}

```

How the program is executing when we defining with if & if else

Case 1:

```

main()
{
    if (S<6)
        printf(" A");
        printf(" B");
        printf(" C");
        printf(" D");
}

```

O/P: ABCD

Case 2:

```

main()
{
    if (S>6)
        printf(" A");
        printf(" B");
        printf(" C");
        printf(" D");
}

```

O/P: BCD

When we executing both the cases the program will not executed as similar. the Compiler make a translations.

```

if( S<6)
{
    printf(" A");
}
printf(" B");
printf(" C");
printf(" D");

```

```

if( S>6)
{
    printf(" A");
}
printf(" B");
printf(" C");
printf(" D");

```

The compiler is executing after the conditional stmt by placing the body within the body One Stmt will be taken place.

for single Stmt from the user side
it is optional to place the body.

(a) If you want to execute multiple stmt's ignore multiple stmt's explicitly. Place the body by the user.

```
if( s>6)
{
    pf("A");
    pf("B");
}
pf("C");
pf("D");
```

→ When else is present if you consider multiple Stmt's after the if (Condition) the user explicitly has to place the body.

Multiple if Stmt's:

```
if(Condition)
{
    Stmt 1;
    Stmt 2;
}
else
```

Stmt 3;

Stmt 4;

Stmt n;

Condition	Stmt 1	Stmt 2	Stmt 3	Stmt 4	n
T	✓	✓	✗	✓	✓
F	✗	✗	✓	✓	✓

```

if( Condition )
{
    Stmt1;
    Stmt2;
}
else
{
    Stmt3;
    Stmt4;
}
}
StmtN;

```

Condition	stmt1	stmt2	stmt3	stmt4	stmtn
T	✓	✓	✗	✗	✓
F	✗	✗	✓	✓	✓

→ main()

```

{
    if( 5 < 6 )
        printf( "A" );
    printf( "B" );
}
else
    printf( "C" );
printf( "D" );
}

```

Because if(Condition) is true only one stmt is executed, more than one stmt is in {}

Solution: if(5 < 6)

```

{
    printf( "A" );
    printf( "B" );
}

```

Write a program accept some basic salary if the salary > 10000
HRA - 25%. DA - 15%. TA - 10%. else
HRA - 20%. DA - 10%. TA - 5%. On basic salary.
Calculate HRA, DA, TA, & Net-Salary & display it

main()

{

float bs, HRA, DA, TA, NetSal;
printf("Enter the basic salary : ");
scanf("%f", &bs);
if (bs > 10000)

{

$$HRA = 25 * \frac{bs}{10000} / 100;$$
$$DA = 15 * \frac{bs}{10000} / 100;$$
$$TA = 10 * \frac{bs}{10000} / 100;$$

else

{

$$HRA = 20 * \frac{bs}{10000} / 100;$$
$$DA = 10 * \frac{bs}{10000} / 100;$$
$$TA = 5 * \frac{bs}{10000} / 100;$$

{

NetSal:

printf("Net Salary : %f", HRA + DA + TA);

~~getchar();~~ NetSal = bs + HRA + DA + TA;

{

printf("%f", bs);

printf("%f", HRA);

printf("%f", DA);

printf("%f", TA);

printf("%f", NetSal);

}

* Do's and Don't with if and else */

44

```
main()
{
    if('A' < 'a')
        printf("A");
    else
        printf("a");
}
```

O/P: a

```
main()
{
    if(0)
        printf(" A");
    else
        printf(" a");
}
```

O/P: a

```
main()
{
    if(0);
    printf(" A");
    printf(" a");
}
```

// The compiler always checks the syntax, as per the syntax there is no error, but it is not treating as conditional stmt.

finally there is no if(condition).

```
→ main()
{
    if(1);
    printf("A");
    else
        printf("a");
}
```

else not executed with odd if

O/P: Error misplaced else

```
→ main()
{
    if(1);
    /* Null Stmt */
    else
        printf("a");
}
```

main()

{

```
float me = 1.1;           // me = 5.375 }  
double you = 1.1;         } me = 5.375 } = OK  
if (me == you)  
    printf("OK");          // me = 1.5 }  
else  
    printf("YES");         you = 1.5 } O/P: OK  
} getch();
```

O/P: YES. // because 1.1 is a recurring no.
recurring no is something but
if never stop.

→ In this program the o/p is executing Not only float & double types, if is executing on the value of the Number of Recurring & Non-recurring.

~~Recurring no~~ ↗ ↘
O/P: 3.0 = 1.333
→ If the user has define a Non-recurring Number
At the Number ~~Never~~ ends)

Non-recurring No's means if Ends.

Recurring No's means if Never-Ends.

→ When Recurring no's occurs O/P will be after else.
When Non-Recurring no's occurs O/P will be before else

→ main()

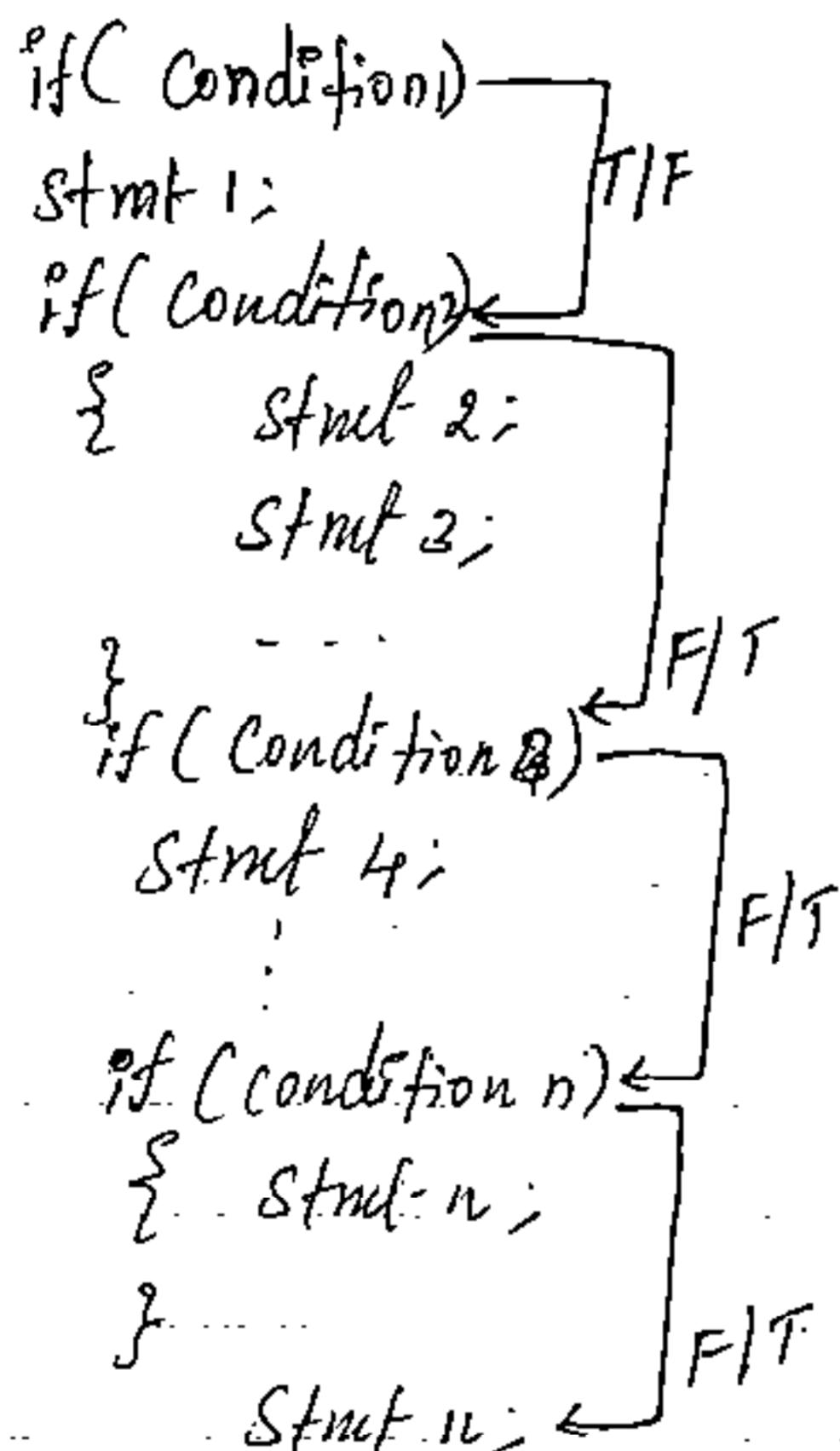
```
{ int a=0; int b=20; char x='1';  
char y='0';  
if(y, b, x, a)      First if has y value, it will  
    printf("Hello");  replaced by 'b' value like that  
else  
    printf("hi");      last value is a = 0  
} O/P: hi
```

```

→ main()
{
    int x,y=2,z,a;
    if(x==y & z) z=2;
    a=2;
    printf("x.d.y.d",z,x);
}
    
```

$x=2 \& z = 0$
 $\text{if}(x=0)$
 $\underline{\text{O/P: } z = \text{Garbage. } x=0.}$

Multiple if Conditions:



```

big = a;
if( b > big)
    big = b;
if( c > big)
    big = c;
if( d > big)
    big = d;
printf("Bigger value: %d", big);
}
getch();

```

O/P: Bigger value: 67.

// if else if ladder

```

if( condition ) T
{
    Stmt 1;
    .
    .
}
else F
if( condition 2 ) ←
    Stmt 2;
else
    Stmt
if( condition 3 )
{
    Stmt n;
}
else
    if( condition n )
    {
        Stmt n;
    }
    .
    .
}
else
    Stmt n; ←
}

```

→ Write a program accept a character & find out the character is uppercase (or) lowercase, digit, (or) any special symbol

46

26 - A-Z (65-90)

26 - a-z (97-122)

10 - Digits (0 to 9) (48 to 57)

32 - Special symbols (+, -, ;, ...)

34 - Non-printable characters or Control keys
or White space)

Ex: ESC, Space, enda, etc

128 - Graphic characters.

256 Totally 256 characters 0-255

- Every key in the keyboard acts as a character.
→ \n indicates moving next line.

Prog:

Main()

{

char ch;

printf("Enter the character:");

scanf("%c" &ch);

if(ch >= 65 && ch <= 90)

printf("The character is uppercase of ASCII %d", ch, ch);

else if(ch == 'a'
(or)

if(ch >= 97 && ch <= 122)

printf("The character is lowercase of ASCII %d", ch, ch);

else

if(ch >= '0' && ch <= '9')

printf("The character is digit of ASCII %d", ch, ch);

else

printf("The character is a special character of ASCII %d", ch, ch);

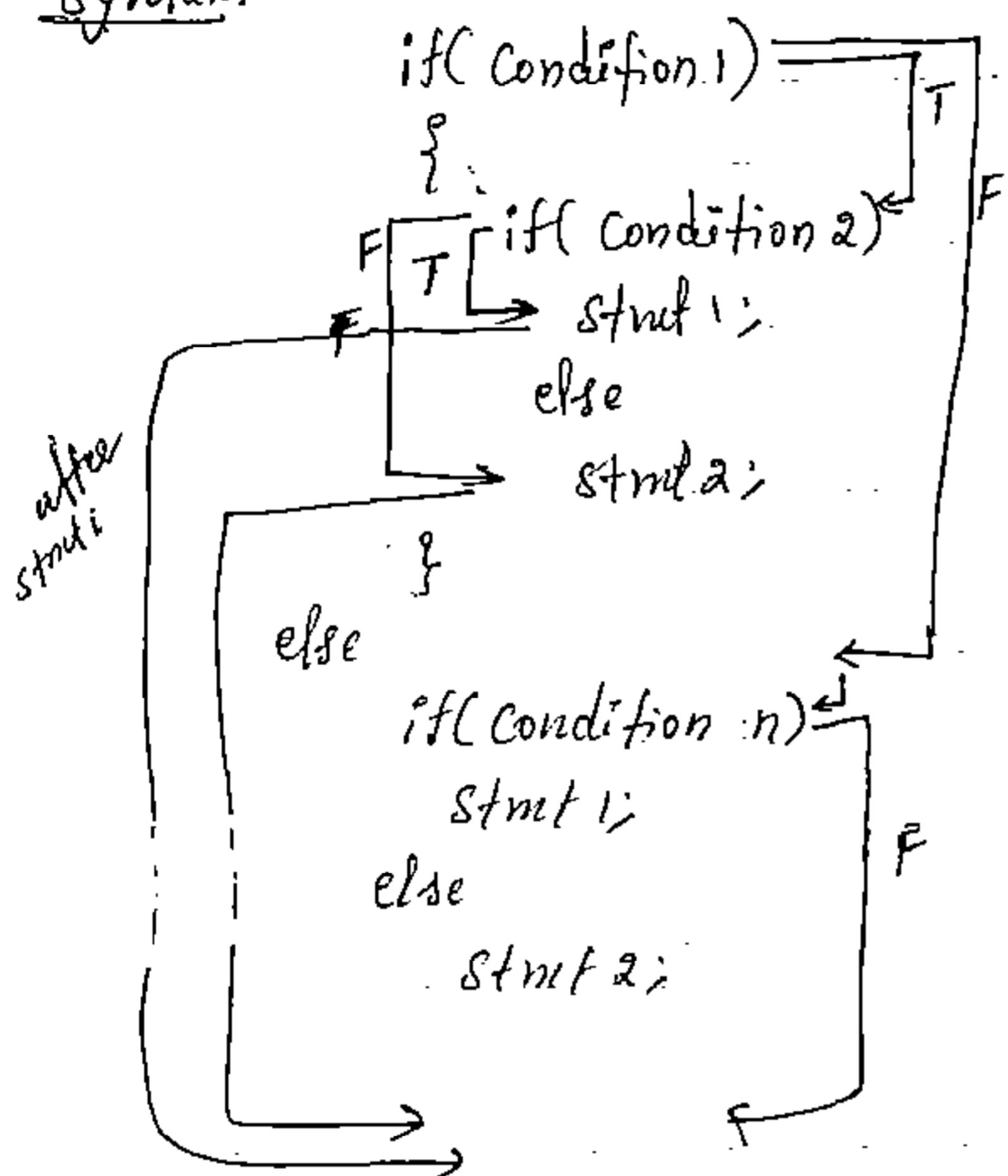
ASCII
Enter - 10
Space - 32

getch();

}

Nested if Condition:

Syntax:-



Ques. write a prog. accept marks in Three Subjects, find out Total & Avg. display the Grade.

1. To display the Grade the student has to pass in all the subjects with min marks of 35.
2. If any one of the Subject has failed display a message "Failed".
3. if $\text{Avg} > 60$ - Grade-A
 $\text{Avg} > 50$ - Grade-B
else Ordinary pass.

Don't use any logical operators.

main()

{

int M₁, M₂, M₃, total, Avg;

printf("Enter three Subjects Marks:");

Scanf("%d %d %d", &M₁, &M₂, &M₃);

if(M₁ >= 35)

{

if(M₂ >= 35)

{

if(M₃ >= 35)

total = M₁ + M₂ + M₃;

Avg = total / 3;

if(Avg >= 60)

{

printf("Grade A");

}

if(Avg >= 50 && Avg <= 60)

{

printf("Grade B");

}

else

printf("ordinary pass");

}

else

printf("Fail");

getch();

}

wrong

✓ Main()

{

int m₁, m₂, m₃, total;

float Avg;

printf("Enter the Marks:");

scanf("%d%d%d", &m₁, &m₂, &m₃);

total = m₁ + m₂ + m₃;

Avg = (float) total / 3;

printf("Tot : %d Avg : %f", total, Avg);

if (M₁ >= 35)

{ if (M₂ >= 35)

{

if (M₃ >= 35)

{

if (Avg > 60)

printf("GRADE A");

else

if (Avg >= 50)

printf("GRADE B");

else pass

printf("GRADE C");

{

else {

{

printf("Fail");

else printf

{ printf("Fail");

else

printf("Fail");

getch();

{

Finding big no. among 4 no's using ifelse. Nested if

main()

{

int a,b,c,d;

printf(" Enter the values:");

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

if(a>b)

{ if(a>c)

{ if(a>d)

printf(" A");

else

printf(" D");

{

else {

~~printf(" C");~~

{

if(b>c)

{

if(b>d)

{

~~if(b>d)~~

printf(" B");

else

printf(" D");

}

else

if(c>d)

printf(" C");

else

printf(" D");

{ getch();

Notes

When we compare multiple if biggest no prog & Nested if biggest no.prog i.e this prog.

In this prog the no.of conditions is more, but when evaluating only 3 comparisons is taking place for any type of no's.

So in the before prog & This prog no.of comparisons are same,

So the performance wise both are same

prog write a prog. a shop keeper allows a debate as part

Item no price Discount-

101 >10,000 25%

{ 102 (or)

{ 103 < 10,000 20%

or > 8000

104(108) 105 < 8000 &

(or) 106 > 5000 15%

Any of the Any of %5
item. price

display

Accept itemno, price, as an input's findout the
amount howmuch the customer has to pay.

display itemno, price, Discoun..

main()

{

int item no, Discoun-

float price,

printf("Enter the item no:");

Scanf("%d", &item no);

if (itemno == 101)

{ printf("%d", itemno);

if (price > 10000)

{

printf(" price");

Discount = 25 * price / 100; \rightarrow printf("%d", Discount);

ApAri = price - Discount;

printf("%f", ApAri);

}

else printf(" price not match");

if(item no == 104 || item no == 105)

49

{ printf(" ~~Item~~, item no: "); }

if(price < 8000 && > 5000)

main()

{ int item, iNo;

float pr, dis,amt;

printf(" Enter the item no: ");

scanf("%d", &iNo);

printf(" Enter the price: ");

scanf("%f", &pr);

if(iNo == 101)

{

if(pr > 10000)

{

discont = pr * 25 / 100;

{ }

else

if(iNo == 102 || iNo == 103)

{

if(pr < 10000 && pr > 8000)

{ dis = pr * 20 / 100;

else

if(iNo == 104 && iNo <= 106)

{ if(pr < 8000)

{ if(pr > 8000)

dis = pr * 15 / 100;

}

else

dis = pr * 5 / 100;

```
    pamt = price - dis;  
printf ("Item no.: %d\n", ino);  
printf ("Original price: %.2f\n", pr);  
printf ("discount : %.2f\n", dis);  
printf ("pamt : %.2f\n", pamt);
```

```
getch();
```

O/P: Enter item no.: 104
enter the price: 6000.

item : 104
Original price: 6000.0
discount : 900
pamt : 4500.0

→ Write a program accept present reading & previous reading.
findout no. of units, based on the Units Generate
Electricity building System.

<u>Units</u>	<u>price</u>	75 50 + 25
0 - 50	1.50	0 - 50. — 1.25
51 - 100	2.25	51 - 100. — 2.25
101 - 200	3.75	
From 201 - 201	6.00	

→ Write a program a Shopkeeper allows a Commission for the
Sales person's

<u>Items No.</u>	<u>Sale Amount-</u>	<u>Rate of Commission</u>
CPU	< 10000	Nill
	$\geq 10000 \text{ & } < 20000$	8%
	≥ 25000	Rs. 2000 + 10% on salamount in excess of 25000.

Monitor	< 10000	5%
	≥ 10000	5% upto 10000 + 8% above

Main)

{

int units, pr,

~~price~~,

float price,

prinff" Enter the present reading");

scnff" Y.d", & pr);

plnff" Enter the previous reading:");

```
main()
{
```

```
    int a;
```

```
    printf("Enter the value:");
```

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

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

```
    getch();
```

```
}
```

O/P: Enter the values: 10 20 30 40 ↵

10

All these values first stored in a buffer
after the 10 is accepted by the variable 'a'.

Remaining space 20 space 30 space 40 are
leftout in buffer.

, fflush(stdin);

Whenever if am accepting some i/p's
with mixed mode of (Integers) & Numbers & characters
always the characters (or) numbers will accept
the data from the buffer, When we
accepting no's from the buffer, there is No problem
but when accepting characters may be read of
garbage characters.

Since every key in the keyboard acts
as a character.

So Not to get any garbage characters
use fflush(stdin) to clean the buffer. & then
accept the i/p's

How many no. of diff char items are used that
many times flush(stdin) will be used

```
#include <stdio.h>
main()
{
    char item;
    float sale_amount, comm;
    clrscr();
    printf("Enter the amount:");
    scanf("%f", &sale_amount);
    printf("Enter the Item Type");
    fflush(stdin);
    scanf("%c", &item);
    if(item == 'C' || item == 'c')
    {
        if(sale_amount < 10000)
            comm = 0;
        else
            if(sale_amount >= 10000 && sale_amount < 25000)
                comm = sale_amount * 0.08;
            else
                if(sale_amount >= 25000)
                    comm = 2000 + (sale_amount - 25000) * 0.1;
    }
    else
        if(item == 'M' || item == 'm')
        {
            if(sale_amount < 10000)
                comm = sale_amount * 0.05;
            else
                comm = 10000 * 0.05 + (sale_amount - 10000) * 0.08;
        }
}
```

```

printf("In Item If It Sale Amount If If Commission\n");
if(item == 'C' || item == 'c')
    printf("In CPU");
else
    if(item == 'M' || item == 'm')
        printf("In Monitor");
printf("If If If If If", sale_amount, comm);
}

```

→ Write a program accept a company ensure the job as follows.

- ~~Prog~~ (10)
- if the person age > 25, Marital Status = Married & Gender is Male
 - person age > 20, Marital Status = UnMarried
Gender - Male
 - person age > 20, Marital Status = UnMarried
Gender = Female.

~~Note:~~ if any one of the Criteria is satisfied he/she eligible for job

~~Note:~~ Final in 'if' Conditions we have to check every condition only if any one of the condition is satisfied
for ex: There are 16 Conditions are present
the 14 Condition is satisfied condition,
then it has to check 13 conditions, due to this in performance wise System will degrade

Ans. ASCII supports 16 if conditions only

Switch()

Syntax:-

```
Switch (integer Variable (or) integer Expression  
       (or) character Variable)  
{
```

case integer (or) character.

constant-1 : statement(s);

break;

case integer (or) character.

constant-2 : statement(s);

break;

case integer (or) character.

constant-n : statement(s);

break;

default : statement(s);

}

Switch() is one more decision making stmt based on the Switch() choice the cases will be executed.

The Switch() choice can be only integer (or) character type. That must be equal to Case type.

If it is not equal then default stmt will be executed.

Whenever the Compiler leads the break stmt it will Come out of the Switch(). Once it is Come out of Switch() there is No reentry.

Since Switch() is a decision making
Decision making only execute only one time

If any of the Case is Not match
default will be executed.

Note:- break & default are optional

// Valid Switch and Cases

Case : 101: Case 101;

Case : 100: Case 100;

Case 1;

Case 0;

Case -64;

Case 6+3*4;

Case 'A';

Case '+';

Case '=';

Switch(5+3-4)

int a;

int a=4;

Switch(a) :

{

 Case 'A' // Not works

 Case 4;

}

Note:

Switch Can't accept
floating point no.,
duplicate, relational
operators, variables,
' ' etc

prog: 10
Ans:

```
#include <stdio.h>
main()
{
    int age;
    char MS, G;
    printf("Enter the person Age\n");
    scanf("%d", &age);
    printf("Enter the Marital Status\n");
    fflush(stdin);
    scanf("%c", &MS);
    printf("Enter the Gender\n");
    fflush(stdin);
    scanf("%c", &G);

    if (age > 20)
    {
        if (MS == 'U' || MS == 'M')
        {
            if (G == 'M' || G == 'm')
                printf("he is eligible for this job");
            else
                printf("you are not eligible for this job");
        }
        else if (G == 'F' || G == 'f')
            printf("She is eligible for this job");
        else
            printf("U are not eligible for this job");
    }
    printf(" U are not eligible for this job");
}
```

```

if( age > 25)
{
    if( MS == 'M' || MS == 'm')
    {
        if( G == 'M' || G == 'm')
            printf(" he is eligible for this job");
        else
            printf(" he is not eligible for this job");
    }
    else
        printf(" he is not eligible for this job");
}

```

```

printf(" In Age It MaritalStatus It Gender \n");
if( MS == 'M' || MS == 'm')
    printf(" Marid");
if( G == 'M' || G == 'm')
    printf(" Male");
if( G == 'F' || G == 'f')
    printf(" Female");
if( MS == 'U' || MS == 'u')
    printf(" Unmarid");
printf(" If you want to exit press C, Age, MaritalStatus, Gender);
getch();
}

```

```

main()
{
    int choice;
    printf("It *** MAIN ***\n");
    printf("It *** 100 police ***\n");
    printf("It *** 108 EMERGENCY ***\n");
    printf("It *** 103 ENQUIRY ***\n");
    scanf("%d", &choice);
    switch(choice)
    {
        case 100: printf("POLICE");
                    break;
        case 108: printf("Ambulance");
                    break;
        case 103:
                    printf("ENQUIRY");
                    break;
        default:
                    printf("wrong choice");
    }
    getch();
}

```

→ Write a prog accept 2 No's and implement the arithmetic operations, subtraction, addition, etc by using switch prog.

```

main()
{
    int choice;
    printf("It *** MAIN ***\n");

```

~~printf("t")~~ AND Arithmetic operation

~~printf("t + Addition\n");~~

~~printf("t - Subtraction\n");~~

~~printf("t * Multiplication\n");~~

~~scanf("%c", &choice);~~

~~Switch(choice)~~

{

~~Case '+': printf("addition:"~~

Main()

{

~~int choice, a, b; char choice;~~

~~printf("MAIN**\n");~~

~~printf("+ + Addition\n");~~

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

~~printf("* * Multiplication\n");~~

~~printf("Enter 'a' value:");~~

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

~~printf("Enter 'b' value:");~~

~~fflush(stdin); scanf("%d", &b);~~

~~scanf("%c", &choice);~~

~~Switch(choice)~~

{

~~case 't': printf("Addition of a & b is: %d", a+b);~~

~~break;~~

~~case '-': printf("Subtraction of a & b is: %d", a-b);~~

~~break;~~

~~case '*': printf("Multiplication of a & b is: %d", a*b);~~

~~break;~~

```
default : printf("Wrong choice");
}
getch();
}
```

Modify the Electricity program by adding some more option & implement by switch.

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

```
<conio.h>
```

```
Void main()
```

```
{
```

```
int no, ph;
```

```
char type;
```

```
float amt, pt, sc, tamt;
```

```
disc();
```

```
printf("1. Domestic (D/d)");
```

```
printf("2. Commercial (C/c)");
```

```
printf("Enter Option:");
```

```
scanf("%c", &type);
```

```
if(type != 'D' & type != 'd' & type != 'C' & type != 'c')
```

```
{
```

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

```
getch();
```

```
exit(0);
```

```
}
```

```
disc();
```

```
printf("Enter No of Units:");
```

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

```
printf("Enter phase (1 or 3):");
scanf("%d", &ph);
```

```
if(ph != 1 & & ph != 3)
```

```
{
```

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

```
    getch();
```

```
    exit(0);
```

```
}
```

```
switch(type)
```

```
{
```

```
    Case 'd':
```

```
    Case 'D':
```

```
        if(nu <= 50)
```

```
            amt = nu * 1.45;
```

```
        else
```

```
            if(nu <= 100)
```

```
                amt = (50 * 1.45) + ((nu - 50) * 2.25);
```

```
    SC = 10.00;
```

```
    pt = nu * 0.06;
```

```
    if(ph == 1)
```

```
{
```

```
        if(pt < 20)
```

```
            pt = 20;
```

```
        else
```

```
            if(pt < 50)
```

```
                pt = 50;
```

```
}
```

```
    break;
```

$$tarif = amt + pt + sc;$$

printf("In PURPOSE (C - COMMERCIAL; D - DOMESTIC) : x.c", TgP);

printf("In PHASE TYPE")

TOTAL NO OF UNITS : x.d", nu);

printf("In PHASE TYPE : x.d", ph);

printf("In BILL AMOUNT : x.2f", amt);

SERVICE CHARGE : x.2f", sc);

POWER TAX : x.2f", pt)

TOTAL BILL AMOUNT : x.2f", tarif)

→ Write a prog

accept Day, month, year find out the big week day?

1986 * 386

$$dd = 5$$

$$dp = yy - 1 *$$

$$mm = 6$$

$$dp = (yy - 1) + 365.l + (yy - 1)/4$$

$$\text{Case 1: } dp = dp + 30$$

$$yy = 1987$$

$$7,25,386$$

31

30

31

28

31

$$\text{Case 2: } dp = dp + 30$$

$$\text{Case 3: } dp = dp + 31$$

$$\text{Case 4: } dp = dp + 31$$

$$3: \quad \quad \quad + 28$$

$$2: \quad \quad \quad + 31$$

$$1: \quad dp = dp + dd$$

$$\begin{array}{r} 25543 \\ + 1 \\ \hline 25543 \end{array}$$

Main()

{
 int dd, mm, yy;
 long dp = 0;
 clrscr();

 printf("Enter day no. : ");
 scanf("%d", &dd);
 printf("Enter Month no. : ");
 scanf("%d", &mm);
 printf("Enter year no. : ");
 scanf("%d", &yy);

 dp = (yy - 1) * 365l + (yy - 1) / 4; // or
 switch(mm)

{

 Case 12 : dp += 30;
 Case 11 : dp += 31;
 Case 10 : dp += 30;
 Case 9 : dp += 31;
 Case 8 : dp += 30;
 Case 7 : dp += 30;
 Case 6 : dp += 31;
 Case 5 : dp += 30;
 Case 4 : dp += 31;
 Case 3 : dp += 28;
 Case 2 : dp += 31;
 Case 1 : dp += dd;

}

if (yy % 4 == 0 & mm > 2)
 dp +=;

1. (yy - 1) - means it can't
 calc. the present year.
2. remove present year
 from the date.
3. 365l : 365 is no. of days
 on a ~~leap~~ year, & l' is leap
 so it can't hold at this value.
4. (yy - 1) / 4 → is leap year
 checking.

if ((yy / 100 == 0 & yy % 400 == 0)
 || (yy % 4 == 0))

Switch (def Y, f)

{

Case 1: printf("Sunday"); break;

Case 2: printf("Monday"); break;

Case 3: printf("Tuesday"); break;

Case 4: printf("Wednesday"); break;

Case 5: printf("Thursday"); break;

Case 6: printf("Friday"); break;

Case 0: printf("Saturday"); break;

{

getchar;

{

prog

→ Write a prog accept a character find out the given character
is vowel (a) or not.

→ " display the o/p of the nos 1, 2, 3, 4, 5

inf- a=1, b=2, c=3, d=4, e=5;

printf("H~~i~~d~~a~~d

y.d)f y.d)k y.d)f y.d)k y.d)f", &a, &b, &c, &d, &e,

Notes This type of Examples can't easily implemented with the decision making Statement if it check the Condition Only once time, & writing without decision making It takes no. of steps to overcome this problem we use Iterations (or) looping-

On Conditional On Controlled Stmts:-

Ex:- goto

* Syntax:

forward declaration:

goto < lablename >

Stmt 1;

Stmt 2;

< lablename >

Stmt 3;

Stmt 4;

backward declaration:

< lablename >

Stmt 1;

Stmt 2;

goto < labname >

→ the goto is an On condition, On Controlled Stmt if it is not a act actual looping Stmt, basically it is called as Jumping Stmt- it can be use in forward & backward directions.

→ main()

{

xyz:

printf(" goto ");

goto xyz;

}

→ In This example the process of Jumping Unconditionally will be takes place, it doesn't have any controlling power to make this unconditional as a Conditional Stmt use default ternary operator.

main()

{

i=1;

xyz:

if(i<=100)

{

printf("%d", i);

if:

goto xyz;

}

Program

→ Write a prog accept a no & find out sum of series for

1 + n.

1+2+3+...+n

+10
+20...etc.

→ Write a prog accept no & display the no. in reverse order.

Program

O/P: num = 451

num = num * 10 + n % 10;

(Q8)

O/P: 154.

num % 10;

num = num / 10;

#include < stdio.h >

#include < conio.h >

main()

{

int i, num;

clrscr();

printf("Enter the number");

scanf("%d", &num);

i:

if(num > 0)

{

int rev = 0, num;

if(num)

{

rev = rev * 10 + (num % 10);

num = num / 10;

goto reverse;

{

556.

```
    printf("%d", num%10); // 556 - 6, 5, 5  
    num = num /10; // 55, 5  
    goto i;  
}  
getch();
```

Prog 11

```

#include < stdio.h>
#include < conio.h>
int i=0, num, sum=0;
clrscr();
printf(" Enter the number:");
scanf("% d", &num);
i:
if(i<=num)
{
    sum = sum + i;
    i++;
    goto i;
}
printf(" Sum of the Series is: % d"
getch();

```

~~o/p:~~ Enter the Number: 10 ↴
Sum of the Series is: 55

~~prog9~~overall not

```

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

main()
{
    char j;
    clrscr();
    printf("Enter the character:");
    scanf("%c", &j);
    if(j == 'A' || j == 'a' || j == 'E' || j == 'e' || j == 'I' || j == 'i'
        || j == 'O' || j == 'o' || j == 'U' || j == 'u')
        printf("given character is over");
    else
        printf("given character is not a over");
}

```

O/P:-

```

1   10
2   9
3   8
4   7
5   6
6   5
7   4
8   3
9   2
10  1

```

```

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

main()
{
    int i=1, n=10, t;
    clrscr();
    t=n;
    while(i<=n)
    {
        printf("%d\t%d\n", i, t);
        i=i+1;
        t=t-1;
    }
}

```

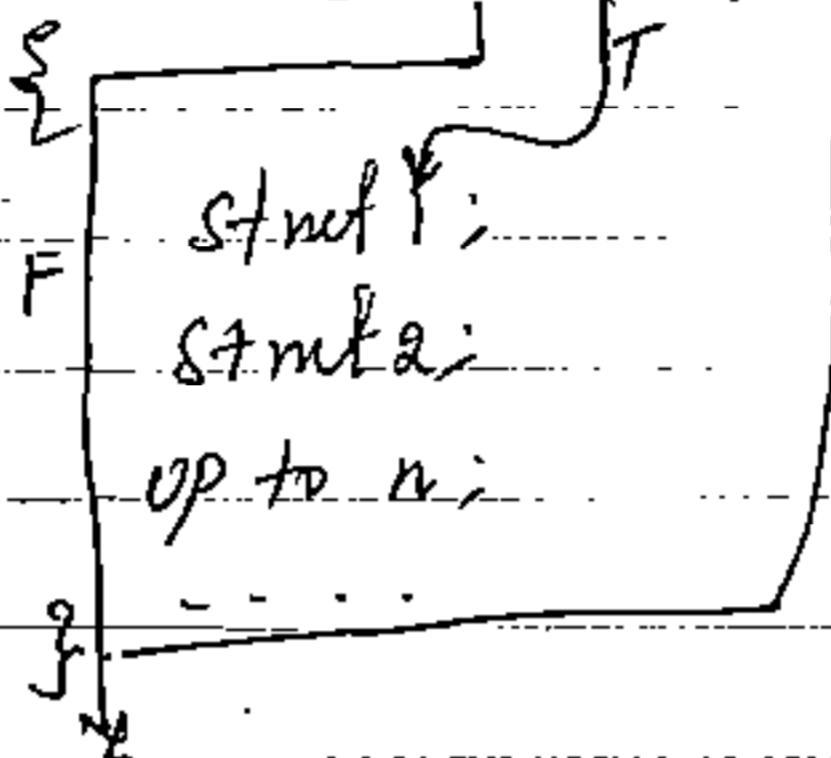
Conditional Control Stmt's:-

While ()

Syntax

Initialization;

while (condition) {



This is actual looping Concepts which can be used for repetition.

The while is only Control loop - the Condition is evaluated if it is Non-zero, & the Stmt's are executed & the condition is evaluated, this cycle continues until the condition become zero & it stops the evaluation at zero if it is still.

prog :- write a program display 1 to 100 no's.

main()

{

int i=1;

while (i<=100)

{

printf("%d", i)

i++;

{

getchar();

2

Write a program. Find out the Squares & Cubes for 1 to 10.

```

main()
{
    int i=1;
    printf("1. Number its square & cube\n");
    while (i<=10)
    {
        i = i*i;
        printf("%d", i);
        i = i*i;
        printf(" cubes %d", i);
    }
    getch();
}

```

Write a prog accept no print the multiplication table for that no.

```

main()
{
    int num=7, i=1; printf("%d num it * 1 to mul. )n");
    while (i<=10)
    {
        mul = num*i; if;
        printf("%d %d %d", num, *, mul);
    }
    getch();
}

```

(08)

```

int num, i=1;
printf("Enter number:");
scanf("%d", &num);
while(i<=10)
{
    printf("%d * %d = %d\n", num, i, num*i);
    i++;
}

```

Write a prog accept 2 no's find the product of 2 no's
with out using multiplication operator.

$1 \cdot 2 = 2$ $(3, 4) = 12$

```

main()
{
    int n1, n2, prod=0;
    printf("Enter the Numbers:");
    scanf("%d%d", &n1, &n2);
    while(n2)
    {
        prod = prod + n1;
        n2--;
    }
    printf("Product of Two numbers %d * %d = %d", n1, n2, prod);
}

```

→ Write a prog accept 2 no's b/w the 2 no's (including)
display the multiplication table

main()

```

{
    int n1, n2,

```

Printff "Enter two no's : ");

Scanff "%d %d", &n₁, &n₂);

while (n₁ <= n₂)

{

~~for~~ i=1;

while (i <= 10)

{

p.f("x.d * x.d = x.d \ln", n₁, i, n₁+i)

i++;

}

n₁+i;

}

n₁
5 + 2 * 6
n₂
2 → ← 7

{ i / (n₁ <= n₂)

p.f(n₁, i, n₁+i)

n₁+i

}

121

i++

- Write a prog accept no findout the given no. is palindrome or not.
- Write a prog accept a no findout the given no. is Armstrong or not.

$$\begin{array}{r} n \\ 125 \\ 27 \\ \hline 153 \end{array}$$

153

$$1^3 + 5^3 + 3^3 = \boxed{153}$$

Palindrome (Q8) Not:

```
#include < stdio.h>
#include < conio.h>
main()
{
    int num, rev=0, temp;
    printf("Enter the number:");
    scanf("%d", &num);
    temp = num;
    while(num) // 121
    {
        rev = rev * 10 + (num % 10);
        num = num / 10;
    }
    if(temp == rev)
        printf(" Number is Palindrome");
    else
        printf(" Number is Not a palindrome.");
    getch();
}
```

num%10 = 121%10
= 1
num = 121
rev = 0 * 10 + 1
rev = 1
num/10 = 121/10
= 12

Armstrong Number (Q8) Not:

```
#include < stdio.h>
#include < conio.h>
main()
{
    int num, i, mul, res=0, temp;
    clrscr();
    printf("Enter the Number:");
    scanf("%d", &num);
```

```
temp = num;
```

```
while( num)
```

```
{
```

```
i = num % 10;
```

```
mul = (i * i * i);
```

```
res = res + mul;
```

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

```
}
```

```
if( temp == res)
```

```
printf(" Number is amstrong number");
```

```
else
```

```
printf(" Number is Not an amstrong number");
```

```
getch();
```

```
}
```

~~11/5/12~~

→ Any looping programming Can be implemented in
3 ways.

1. with body (default)

2. without body

3. with Semicolon ;

1. With body: Whenever after the body multiple stmt's are present then we can implement the programming with body. The compiler will execute the program without any replacement.

Ex:- main()

```
{ int i=1;
```

`while(i <= 10)`

{

`p.f("x.d", i);`

`i++;`

}

No replacement at the time
of execution by the compiler

2. without body:

Another way we can also write the
looping programming without body.

When we define without body at the
time of execution the compiler replace the program
with body and only one stmt will be
takes place within the body.

Ex:- main()

{ int i=1;

`while(i <= 10)`

`p.f("x.d", i);`

`i++;`

}

Replacement

`while(i <= 10)`

{

`p.f("x.d", i);`

}

`i++;`

3. with a Semicolon

We can also implement the
program of looping with Semicolon.

When we implemented with Semicolon, at the
time of execution the loop will be replaced with the body
of with in the body there is no stmt's.
i.e. empty body.

Compiler will replace
like this.

Ex:- main()

```
{ int i=1;
```

```
while(i<=10);
```

```
p-f("x.d", i);
```

```
i++;
```

```
}
```

replacement

```
while(i<=10)
```

```
{ } [ ]
```

```
{ }
```

```
p-f("x.d", i);
```

```
i++;
```

Note:- Any type of looping procedure

the prog will execute with body only

nothing will be printed

①

main()

```
{ int a=5;
```

```
while(a)
```

5, 4, 3, 2, 1

```
{ }
```

```
p-f("x.d", a)
```

5, 4, 3, 2, 1

```
a=a-1;
```

```
{ }
```

```
p-f("x.d", a+10);
```

10

```
}
```

②

main()

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

```
clrscr();
```

```
a=b=1;
```

```
while(a)
```

p-f("m x.d x.d", a+10, b+10);

```
{ }
```

~~del, del~~

a=b <= 3

1 = 1 <= 3

satisfied
then return

If is Not a
if condition.

{ 1, 2 < 3 } \rightarrow logical operation

2: a=b <= 3;

b=b+1;

```
p-f("x.d x.d", a, b);
```

```
}
```

(3)

```
main()
{
    int a, b = 1;
    while (a)
    {

```

$a \leq b + f \leq 3$:

p-f("y,d,y,d", a, b)

$a > 1 \leq 3$, $1, 2 \geq 3$ also $4 \leq 3$ X
 $1, 2, 3, 0$

1, 3, 4, 4, 0, 5

}

p-f("In y,d,y,d", a+10, b+10); 10, 15

(4)

```
Main()
{
    int a;
```

classer;

$a \leq 3$;

while (a--)

p-f("y,d", a);

p-f("y,d", a+10);

}

depth control contains only single symb.

while (a--) when condition fail it returns

{

p-f("y,d", a); 2, 1, 0

{

p-f("y,d", a+10); 10

-1 + 10 = 9

(5)

main()

{ int a;

classer(); a=1;

while (a+f <= 1)

while (a+f <= 2);

p-f("y,d", a);

}

replacement fail $a \leq$

4 + f <= 1 1 + f <= 1

while (a+f <= 1) \leftarrow 4

while ($a^2 + f \leq 2$) \leftarrow 3 ≤ 2 fail

$a = 3 \quad 3 + f = 4$

~~1 + f = 2~~

p-f("y,d", a);

5

Main()

```
{
    int a=1;
    class();
    while(a+f <=3)
        p-f("Y.d", a);
        p-f("Y.d", a+10);
}
```

replacement-

condition fail
 $\text{while}(a+f \leq 3) \rightarrow$ also value
 increment

```
{
    p-f("Y.d", a); 2,3,4,
    p-f("Y.d", a+10); 15
}
```

⑥

Main()

```
{
    int a=1;
    class();
    while(a+f <=1)
        while(a+f <=2)
            p-f("Y.d", a);
}
```

replacement-

$\text{while}(a+f \leq 1) \leftarrow^5$

```
{
    while(a+f <=2) 4
    p-f("Y.d", a); 3,4
}
```

⑦

Main()

```
{
    int a;
    class();
    a=1;
    while(a-->=1)
        while(a-->=0)
            p-f("Y.d", a);
}
```

replacement-

$\text{while}(a-- \geq 1) \leftarrow -3$

$\text{while}(a-- \geq 0) \leftarrow -2$

$p-f("Y.d", a); \leftarrow -3$

void Main()

```
{  
    int a;  
    clrscr(); a=1;  
    while(a<=1)  
        if(a>2)  
            p.f("Y.d", a++);  
        else  
            p.f("Y.d", ++a);  
    p.f("Y.d", a+10);  
}
```

replacement -

```
while(a<=1) {  
    if(a>2)  
        p.f("Y.d", a++);  
    else  
        p.f("Y.d", ++a);  
} p.f("Y.d", a+10);  
2+10 = 12
```

⑧

void Main()

```
{  
    int i=1;  
    while(a+i>=1);  
    p.f("Y.d", a);  
}
```

opt^{cheat}
sub range
while(a+i>=1) {
} 2,3,4, ..., 32767
} - 32768
p.f("Y.d", a);
→ -32767

⑨

void Main()

```
{  
    int a=-1;  
    clrscr();  
    while(a--);  
    p.f("Y.d", a);  
}
```

while(a--) {
} -2, -32768, Nat^o
p.f("Y.d", a); 0

break:

The break is an UnConditional Control Stmt. When the Compiler sees the break Stmt, it will come out of the loop.

break Can be used only in loop & switch.
i.e. inside the loop.

break Can't be used in 'if' condition.

A Example: on break Stmt :-

main()

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

while (i<=10)

{

p·ff(" Enter the Number >d:", i);

scanf(" >d", &num);

if (num < 0)

break;

sum = num;

i++;

}

p·f(" Count: >d Sum: >d ", i, sum);

getch();

}

main()

{

while (Condition)

{

if (Condition)

break;

stmt-1;

stmt-2;

{

stmt-3;

}

→ Write a prog. accept no. find out the given no. is in financial series or not

Main()

{

int ~~t~~ t₁=0, t₂=1, t, n; CK=0;

clrscr();

p-f(" Enter the Number : ");

scanf("%d", &n); //8

if(n==0)

{

CK=1;

else

{

while((t=t₁+t₂)<=n)

{

if(f==n)

{

CK=1;

break;

}

t₁=t₂;

t₂=t;

} // while close

} else close

if(CK==1)

p-f(" Given Number is present in fiboacci series ");

else

p-f(" given Number Not present in fiboacci series ");

}

Continue:

The Continue is another unconditional controlled Stmt whenever the Compiler leav the Continue Stmt it will once again leav inside the loop, if is exactly opposite to break Stmt:

The Continue must be there only inside the loop.

Main()

{
=

while (condition) {

{
=

if (Condition) {

 Continue; }
 T

} //end while

} //end Main

→ write a program accept 10 no's & find out the biggest no the Condition all the 10 No's must be (+ve) No's

Main()

{

int num, i=1, big=0;

While (i <= 10) {

{

 printf("Enter the Number %d : ", i);

 Scanf("%d", &num);

```
if (num <= 0)
```

```
{
```

```
    printf("Native no's are not allowed... In");
```

```
    getch();
```

```
    Continue;
```

```
}
```

```
if (num > big)
```

```
    big = num;
```

```
i++;
```

```
{
```

```
p-f("Biggest: %d", big);
```

```
getch();
```

```
}
```

①

```
Main()
```

```
{
```

```
int i=1;
```

```
while (i <= 10) {
```

```
{
```

```
    if (i > 4)
```

```
        break;
```

```
    p-f("%d", i);
```

```
i++;
```

```
{
```

```
p-f("%d", i+10);
```

15

```
}
```

②

```
Main()
```

```
{
```

```
int i=0;
```

```
while (i < 100) {
```

```
{
```

```
i = i+2; /2,
```

```
if (i > 40 && i < 60)
```

```
    Continue;
```

```
    p-f("%d", i);
```

1, 2, 3, 4

5, 6, 7, 8

9, 10, 11, 12

13, 14, 15, 16

17, 18, 19, 20

21, 22, 23, 24

25, 26, 27, 28

29, 30, 31, 32

33, 34, 35, 36

37, 38, 39, 40

41, 42, 43, 44

45, 46, 47, 48

50, 51, 52, 53

55, 56, 57, 58

60, 61, 62, 63

65, 66, 67, 68

70, 71, 72, 73

75, 76, 77, 78

80, 81, 82, 83

85, 86, 87, 88

90, 91, 92, 93

95, 96, 97, 98

100, 101, 102, 103

105, 106, 107, 108

110, 111, 112, 113

115, 116, 117, 118

120, 121, 122, 123

125, 126, 127, 128

130, 131, 132, 133

135, 136, 137, 138

140, 141, 142, 143

145, 146, 147, 148

150, 151, 152, 153

155, 156, 157, 158

160, 161, 162, 163

165, 166, 167, 168

170, 171, 172, 173

175, 176, 177, 178

180, 181, 182, 183

185, 186, 187, 188

190, 191, 192, 193

195, 196, 197, 198

200, 201, 202, 203

205, 206, 207, 208

210, 211, 212, 213

215, 216, 217, 218

220, 221, 222, 223

225, 226, 227, 228

230, 231, 232, 233

235, 236, 237, 238

240, 241, 242, 243

245, 246, 247, 248

250, 251, 252, 253

255, 256, 257, 258

260, 261, 262, 263

265, 266, 267, 268

270, 271, 272, 273

275, 276, 277, 278

280, 281, 282, 283

285, 286, 287, 288

290, 291, 292, 293

295, 296, 297, 298

300, 301, 302, 303

305, 306, 307, 308

310, 311, 312, 313

315, 316, 317, 318

320, 321, 322, 323

325, 326, 327, 328

330, 331, 332, 333

335, 336, 337, 338

340, 341, 342, 343

345, 346, 347, 348

350, 351, 352, 353

355, 356, 357, 358

360, 361, 362, 363

365, 366, 367, 368

370, 371, 372, 373

375, 376, 377, 378

380, 381, 382, 383

385, 386, 387, 388

390, 391, 392, 393

395, 396, 397, 398

400, 401, 402, 403

405, 406, 407, 408

410, 411, 412, 413

415, 416, 417, 418

420, 421, 422, 423

425, 426, 427, 428

430, 431, 432, 433

435, 436, 437, 438

440, 441, 442, 443

445, 446, 447, 448

450, 451, 452, 453

455, 456, 457, 458

460, 461, 462, 463

465, 466, 467, 468

470, 471, 472, 473

475, 476, 477, 478

480, 481, 482, 483

485, 486, 487, 488

490, 491, 492, 493

495, 496, 497, 498

500, 501, 502, 503

505, 506, 507, 508

510, 511, 512, 513

515, 516, 517, 518

520, 521, 522, 523

525, 526, 527, 528

530, 531, 532, 533

535, 536, 537, 538

540, 541, 542, 543

545, 546, 547, 548

550, 551, 552, 553

555, 556, 557, 558

560, 561, 562, 563

565, 566, 567, 568

570, 571, 572, 573

575, 576, 577, 578

580, 581, 582, 583

585, 586, 587, 588

590, 591, 592, 593

<

(3)

/ 68 /

Main()

{ int i=1;

while (i<=20) {

{

if (i>4 & & i<=17)

continue;

printf(" %d ", i); } 1, 2, 3, 4, 18, 20... infinit

} i++; 2, 3, 4, 5

}

exit()

The exit() makes you to terminate from the program.

→ the exit() can be used, only were in

→ the program but it is mostly used in decision making
i.e ifMain()

if indicator → { =

```

if (num == 0 || num == 1)
    {
        printf("1");
        exit(0);
    }
}
return 0;
}

```

if indicator → { =

program successfully. {

Commonly successful. {

W. i.e. {

→ Write a prog accept 2 no's as a base & exponent find the base value?

→ write a prog accept a decimal no Convert into Binary.

→ write a prog accept no find out for for the given no the factors for that no, & also display the given no is prime (or) Composit no.

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

```
Void Main()
```

```
{
```

```
int n, i=2, c=0;
```

```
clrscr();
```

```
p.f("nEnter the Number:");
```

```
Scanf("%d", &n); //10
```

```
while(i<=n && c<=1)
```

```
{
```

```
i)(i==2)
```

```
p.f("Even Number from 1 to %d:", n); 2,4,6,8
```

```
if (i==1)
```

```
p.f("Odd Number from 1 to %d:", n); 1,3,5,7,9
```

```
-p.f("%d\n", i);
```

```
i=i+2;
```

```
if (i>n)
```

```
{
```

```
i=1;
```

```
c++;
```

```
}
```

```
getch();
```

```
}
```

18/8/19 69

→ Write a C prog. To Compute the sum of series $1+x+x^2+x^3+x^4+\dots+x^n$

Main()
{

int x, count, term, sum, i=1;

printf("In Give the value of x:");

scanf("%d", &x);

printf("In Upto How many numbers sum is needed?");

scanf("%d", &count);

term = sum = 1;

while(i < count)

{

term = term * x;

sum = sum + term;

i++;

}

printf("The Sum up to %d terms of the series", count);

printf("%d", 1 + 1 * x + 1 * x * x + 1 * x * x * x + ... + 1 * x * x * x * ... * x * x);

{

→ Write a C prog. to compute sum of series

$(1 + 1/x + 1/x^2 + 1/x^3 + 1/x^4 + \dots + 1/x^n)$

$1 + 1/x + 1/x^2 + 1/x^3 + 1/x^4 + \dots + 1/x^n$.

→ Convert decimal to Binary.

$$0 + ((12 \times 2) * \text{pow}(10, 0))$$

Main()

$$0 + 0 * 0$$

{

int n;

$$\stackrel{>0}{=}$$

$$0 + (6 \times 2) * \text{pow}(10, 1)$$

Unsigned long bin = 0, p = 0;

$$0 + 0 * 100$$

printf("Enter the No:");

$$\stackrel{>0}{=}$$

scanf("%d", &n);

while(n)

$$0 + (3 \times 2) * p(10, 2)$$

{

$$\text{bin} = \text{bin} + ((n \times 2) * \text{pow}(10, p)); \quad \underline{= 100.}$$

$$n / 2;$$

$$p++;$$

$$100 + (1 \times 2) * p(10, 3)$$

}

$$100 + (1 * 1000)$$

printf("Binary No: %lu", bin);

$$\underline{\underline{1100}}$$

}

decimal to Binary:

$$\text{bin} = \text{bin} + ((n \times 2) * \text{pow}(10, p))$$

Binary to decimal:

$$\text{bin} = \text{bin} + ((n \times 0) * \text{pow}(2, p))$$

Hexadecimal to decimal

$$\text{bin} = \text{bin} + ((n \times 16) * \text{pow}(8, p))$$

decimal to Hexa

$$\text{bin} = \text{bin} + ((n \times 8) * \text{pow}(16, p))$$

O/P:

00 01 02

10 11 12

20 21 22

When we implementing this type of patterns first consider No. of rows & columns.

8th & final values

The initial Values of Row's & Column's.

170

Main()

{

int i, j;

i = 0;

→ while (i < 3)

{

j = 0;

while (j < 3) ←

{

printf(" %3d %d ", i, j);
j++;

}

i++;

printf("\n");

}

getch();

}

Main()

O/P

1 1 1

{ int i, j;

2 2 2 2

i = 1;

3 3 3 3

while (i < 5)

4 4 4 4

{

printf("%

j ≥ 1;

→ while (j < 5)

{

printf("%3d ", i);

j++;

}

i++;

{ printf("\n");

getch();

O/P:

1

Main()

1 2

{

1 2 3

int i, j;

1 2 3 4.

i=1;

while (i <= 4)

{

i=4

3 <= 4

4 <= 4

j=1;

i <= j =

while (~~i > j~~) j <= i)

2 <= 1

2 <= 2

3 <= 3

4 <= 4

printf("%d", j);

j++;

2 1 2

}

1 2 3

if;

1 2 3 4

printf("\n");

}

1 2 3 4

4 3 2 1

5 5 5 5 5

1 2 3

3 2 1

5 5 5 5 5

1 2

2 1

5 5 5 5 5

1

1

1

printed info s;

Main()

i=1;

{

while (i <= 15)

{

p.f("5");

{

if (i % 5 == 0)

{

p.f("\n");

{

i++;

{

}

5 columns
5 rows
15 prints

}

{

{

{

{

printf("%d", i);

j++;

{

{

{

if;

1 2 3 4

1 78 1

1 2 3

4 3 2 1

1 2

3 2 1

1

2 1

1

Main()

Main()

{

{

int i,j;

int i,j; int i=1;

i=1;

while (i<=4)

{

{

j=1;

j=5-i;

while (j<=(5-i))

while (j+i>i) ~~if i>1~~

{

p.f("y.d.",j);

~~if i>1~~
421,321,221 121

j+=;

32,1
2,1

{

i+=;

{

p.f("n");

{

getch();

{

getch();

{

Note:- in the program condition
have 'i' it contains ~~++ i~~

1 2

3 = 4 -- i

1 2 3

1 2 3 4

1 2 3 4 5

Main()

{

int i, j, n;

printf("Enter the rowsize:");

scanf("%d", &n); n=5

i=1;

while(i<=n) /* row */

{

j=n;

while(j>i) /* spaces */

{

printf(" ");

j=j-1;

}

j=1;

while(j<=i) /* column */

{

printf("%d", j);

j++;

i++;

printf("\n");

{

getch();

}

F2

1.2.1

1 2 3 2 1

1 2 3 4 3 2 1

1 2 3 4 5 4 3 2 1.

Matrix:

{

int i, j, n;

printf("Enter the row size:");

&cauf("%d", &n);

i=1;

while (i <= n)

{

j=n;

while (j > i)

{

printf(" ");

j--;

}

j=1;

while (j <= i)

{

printf("%d ", j);

j++;

}

j=i;

while (j > 1)

{ printf("%d ", j-1)

j--;

}

j++;

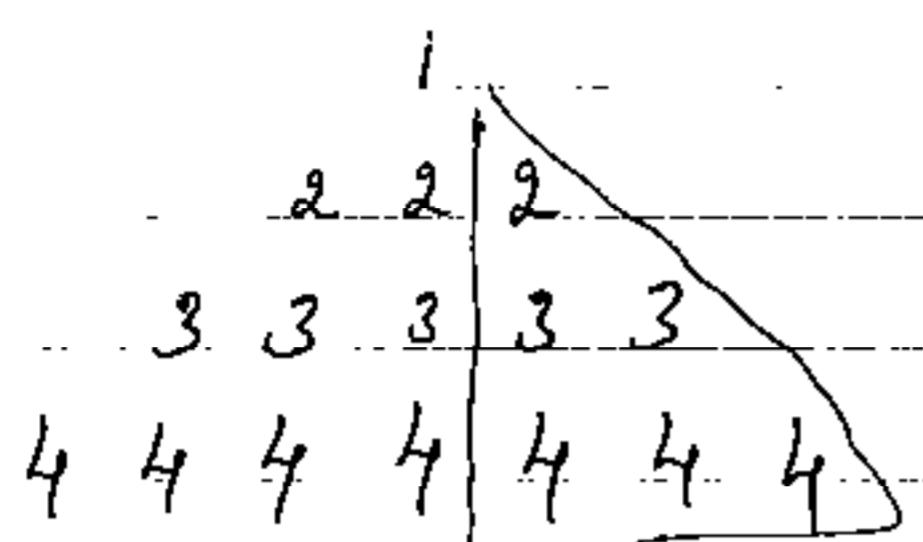
printf("\n");

} getch;

1 2 3 2 1

1 2 1

1



4 4 4 4 4 4 4
3 3 3 3 3 3
2 2 2

main()

{

int i, j, n;
printf("Enter the row size: ");
~~scanf("%d", &n);~~

i = 1;

while (i <= n)

{

j = n;

while (j > i)

{

printf(" ");

j = j - 1;

}

j = 1;

while (j <= i)

{

printf(" %d ", j);

j = j + 1;

}

j = i;

while (j > i)

{

printf(" %d ", j);

j = j - 1;

i++;
printf(" \n");

getchar();

}

}

* * *

 * * *

 * * * *

* main()

{

int i, j, l, s;

char ch = 'A';

printf("Enter the row size:");

scanf("%d", &s);

i = 1;

while (i <= s)

{

s = 1;

while (s <= 38 - i + 2) /* spaces */

{

~~printf("%c", ch);~~

stf;

}

j = 1;

while (j <= i)

{

textcolor(15);

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

j++;

}

if:

printf("m");

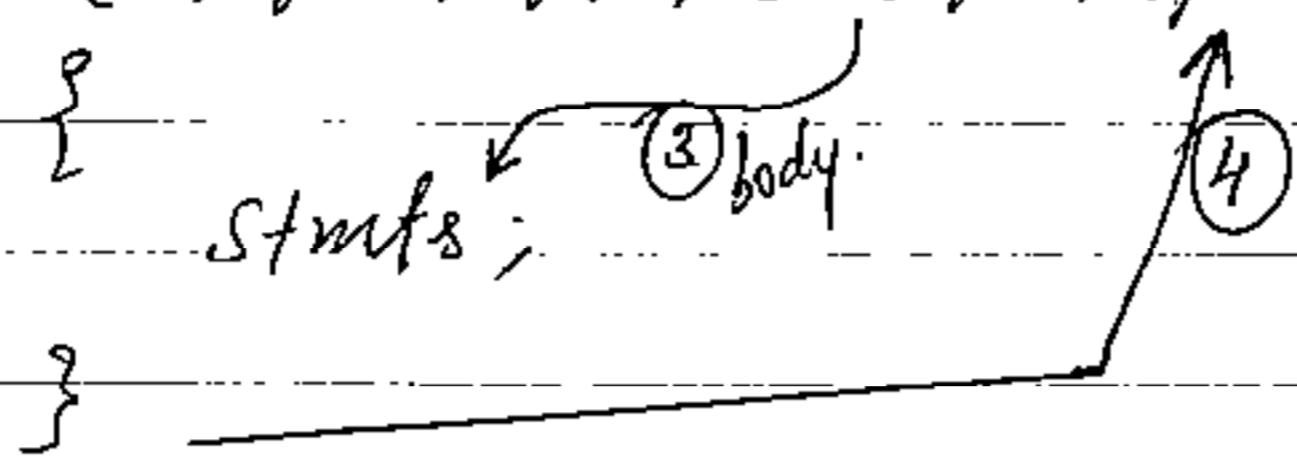
}

getch();

}

for :-

for(initialization, condition, updation)



The for is also a entry control looping Stmts.
In for everything is optional.

for(; ;) // valid

while() // invalid.

The workflow of the for will be in A clockwise direction.

The for is more flexible rather than all the looping Concepts. We can make any no. of initializations, any no. of Conditions & Updations in for loop & by separating with (,) operator

Ex:- for(i=1, j=1; i<=5, j<=6; i++, j--)

Main()

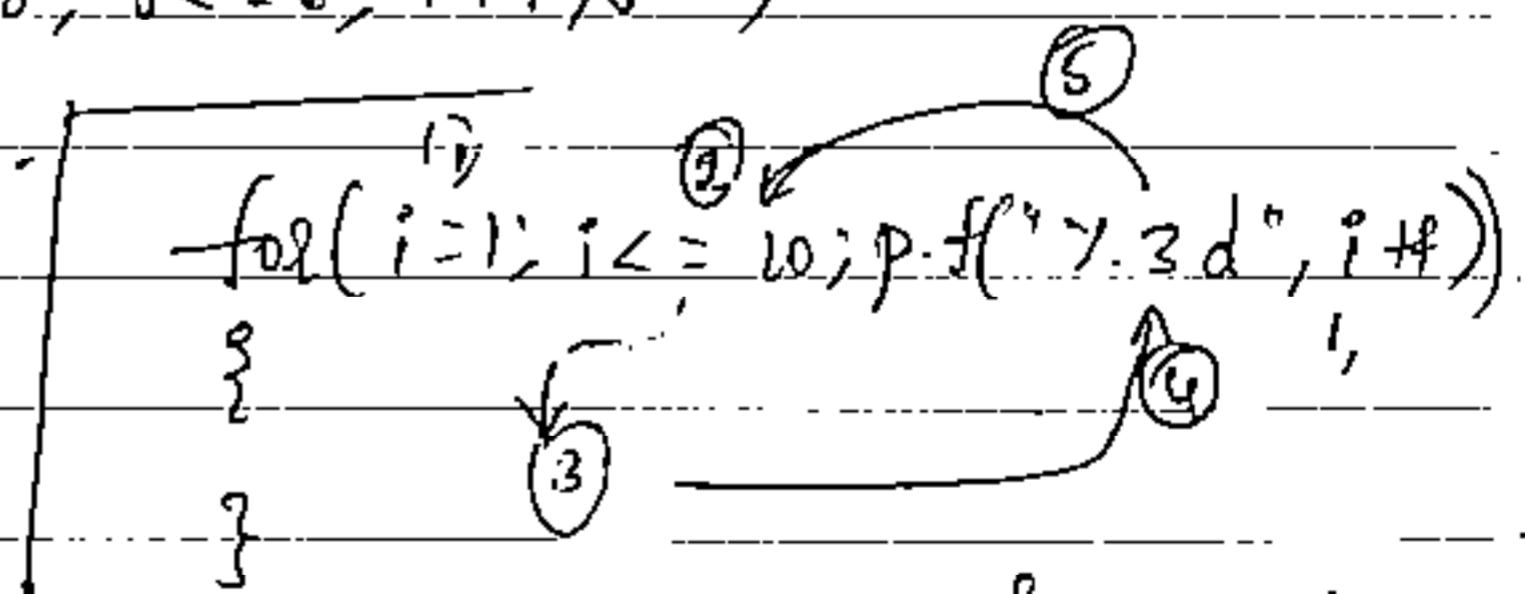
{

int i;

for(i=1; i<=10; printf("%d", i++)) ;

* getch();

{



With Semicolumn

① main()

{

int a, b;

for(a=b=1; a; printf("%d %d", a, b)) {

a = b + f <= 3;

printf("%d %d", a+10, b+10);

{

int a, b; \rightarrow condition is NonZero

for(a=b=1; a; printf("%d %d", a, b))

1, 2

P, 3

a = b + f <= 3;

14

j \downarrow condition is satisfied
it returns 1

printf("%d %d", a+10, b+10);

10, 15

✓

② Main()

{

int a=0, j, x;

6 3
4 4
2 2
 $x=0, j=5$

0 < 5 5 > 0

x++;

for(k=0, j=5; k<5, j>0; k=k+2, j--)

{

x++;

printf("The values of k, j, x is %d %d %d", k, j, x);

{

0, 5, 6

{

2, 4, 2

{

4, 3, 3

This total output is executing

6, 2, 4

based on the condition j>0

8, 1, 5

it never goes depending on the k<5

it always depends on last condition.

for(k=0, j=5; k<5 && j>0; k=k+2, j--)

for(k=0, j=5; k<5 || j>0; k=k+2, j--)

(3)

Main()

```

{
    int i=0;
    for(;;)
    {
        printf("%d", i); 0, 1, 2, 3, 4, 5
        i++;
        if(i>5)
            break;
    }
}

```

(4)

Main()

```

{
    int a=1;
    for(a++; a++ <= 2; a++)
    for(a++; a++ <= 7; a++)
        a++;
    printf("%d", a);
}

```

Stepmout

$a = \text{int } a = 1;$

for($a++$; $a++ \leq 2$; $a++$)

{ 3 → 4 → 7 → 10
 for($a++$; $a++ \leq 7$; $a++$)
 { 5 → 8 → 6 → 9
 a++; }

} printf("%d", a); 13

1 1 1 1 1

1 2 2 2 2

1 2 3 3 3

1 2 3 4 4

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

1 2 3 4 5

{

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

{ if(j >= i)

p.f("%d", j);

else

p.f("%d", i);

p.f("%n");

1 1 1 1 1

75

1 0 0 0 1

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

1 0 0 0 1

{

1 0 0 0 1

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

1 1 1 1 1

if(i==1 || j==1 || i==n || j==n)

1 0 0 0 1

1 1

else

0 1 0 1 0

2 3 5

0

0 0 1 0 0

4 5 6 15

1 0 1

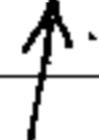
0 1 0 1 0

7 8 9 10 34

2 1 0 1 2

1 0 0 0 1

11 12 13 14 15 65



Main()

{

int i, j, n, b=1, s;

printf("Enter the 'n' value:");

scanf("%d", &n);

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

{

s=0;

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

{

printf("%d", b);

b+=s; //s=0+1=1

b++;

}

printf("%d", s);

printf("\n");

}

}

→ Write a prog accept a no. the given no. is prime (or) not.

Algorithm
Main()

```
{  
    int n, i, count = 2;  
    printf("Enter the Number:");  
    scanf("%d", &n);  
    for(i=2; i<=n/2; i++)  
    {  
        if(n/i == 0)  
        {  
            count++;  
        }  
    }
```

#include <math.h> /* Display the prime nos in b/w the Two nos */
void main()

{

long int n1, n2, n, cnt = 0, t, flag;

clrscr();

printf(" Enter the Two Numbers:");

scanf("%d %d", &n1, &n2);

for(n=n1; n<=n2; n++)

{

// for(flag = 1, t=2; t<n; t++)

for(flag = 1, t=2; t<=sqrt(n); t++)

{

if(n % t == 0)

{

flag = 0;

break;

}

if (flag == 1)

printf("In %ld PRIME = %ld", ++cnt, n);

}

getch();

}

* Write a program on Square Root logic

#include < stdio.h >

#include < conio.h >

Void main()

{

int n, i;

float j;

clrscr();

textmode(0);

printf("Enter any Number: ");

scanf("%d", &n);

for (i=1; i*i < n; i++); // with semicolon has no body

if (i*i == n)

{

p-f("Square root of %d is %d\n", n, i);

p-f(" %c %d = %c %f ", 251, n, 241, i);

}

else

{ for (j=i-1; j*j < n; j=j+0.0001);

p-f(" square of %d is %f \n", n, j);

p-f(" %c %d = %c %f ", 251, n, 241, j);

getch();

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

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

A B C D E E D C B A

A B C D D C B A

A B C C B A

A B B A

A A

A B B A

A B C C B A

A B C D D C B A

A B C D E E D C B A

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

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

#include < stdio.h >

#include < Conio.h >

Void main()

{

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

clrscr();

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

{

p=65;

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

{

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

printf("%c", 32); // 32 ASCII for space + /

else

printf("%c", p);

if(j<x)

p++;

else

p--;

```
if(i>1) //spacers logic
{
```

```
    if(i<#) /* Top spacers */
{
```

x--;

```
y++;
```

```
else /* bottom spacers */
{
```

x++;

y--;

}

} // if

printf("%u");

} // outer for

getchar();

}

①

1 1

1 1 1

4

4

2 2

2 2

3 3

3 3

3 3 3

3

1 1 1 1 1 1

1 1 1 1 1 1

0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0

③

1 1 1 1 1 1

1 1 1 1 1 1

1 2 2 1

2 2 2 2 2

1 3 3 1

3 3 3

1 4 1

4

1 3 3 1

2 1

1 1 1 1 1 1

④

4	4
3	3
2	2
1	1
0	0
1	1
2	2
3	3

⑤

1	1	1	1
0	0	0	0
1	1	1	1
2	2	2	2
3	3	3	3
4	4	4	4
5	5	5	5

⑥

4	4
3	3
2	2
1	1
0	0
1	1
2	2
3	3
4	4

→ Write a program display all the ASCII codes with their ASCII Numbers.

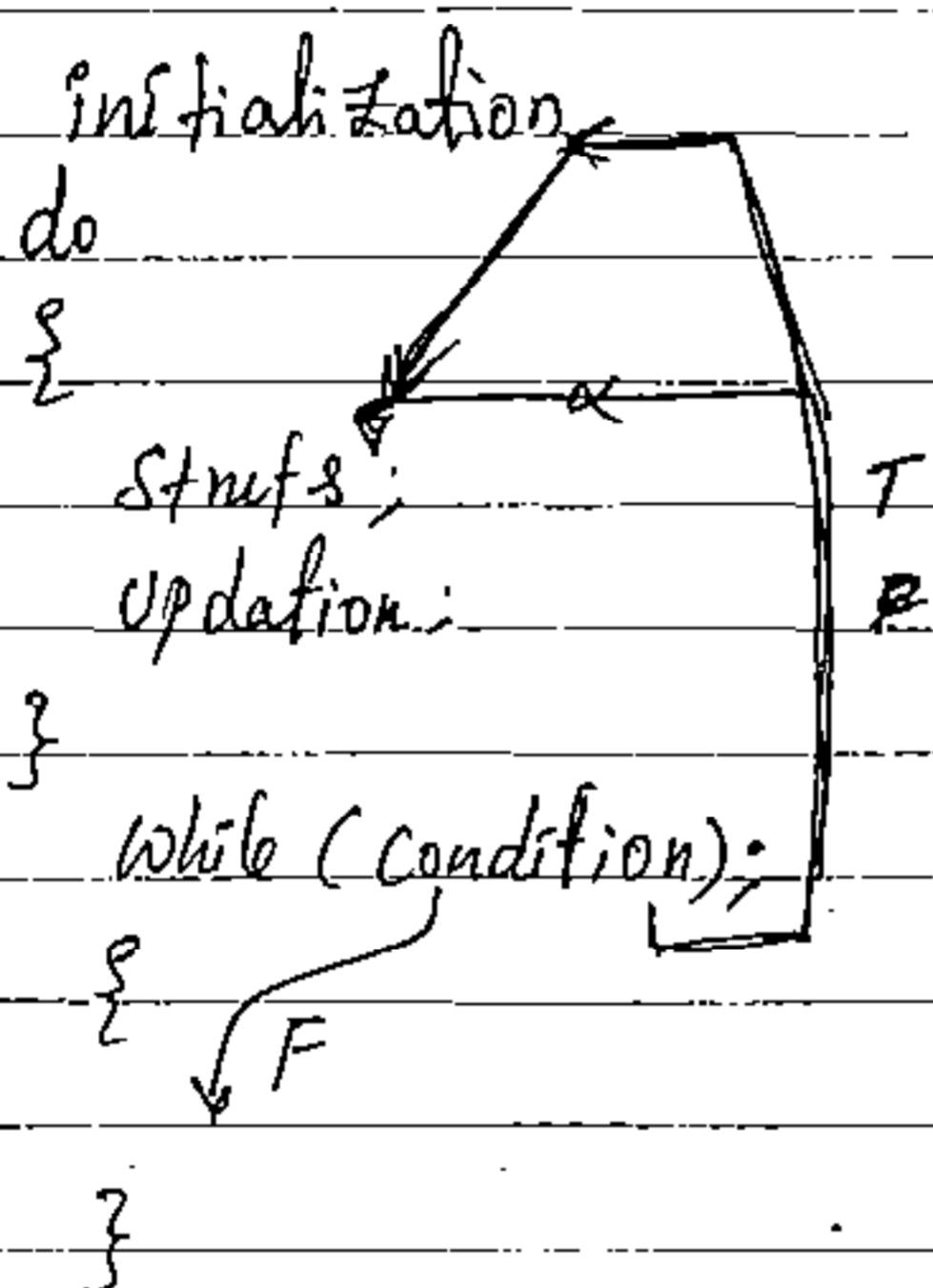
→ Write a program accept a Number & also accepts 2 digits replace the 1st digit with the 2nd digit & display the Number.

Ex:- 54647 $d_1 = 4$ rev = 74645
 $d_2 = 9$ 59697

→ Write a prog display all the amstrong numbers below 1000.

The do while is exit control looping stnkt.
first- if enter into the body & then atleast execute
one stnkt & then check the condition.

if the condition is NonZero, Once again
if reenter inside the loop & this process of checking
the condition will be taken place until the
Condition is Zero.



In do-while the while by default-
will have the ; . it is Mandatory.

do while also can be takes place

1. with body.
2. without body.
3. with Semicolon.

Whenever the do loop is closed
immediately while has to present with ; (semicolon).

After do loop closed immediately while only has present. No other stmts ad takes place.

Within the body only one stmnt

In side the 'do'

while with ;

There can't be present

replacement -

①

main()

```
{  
    int i=10;  
    do  
        printf("%d", --i);  
    while(--i);  
}
```

main()

```
{  
    int i=10;  
    do  
        {  
            printf("%d", --i); 9753  
        }  
    while(--i);  
}
```

②

main()

```
{  
    int i=1;  
    do  
        while(i++ < 1);  
        while(i++ <= 2);  
        printf("%d", i);  
    }  
}
```

main()

```
{  
    int i=1;  
    do  
        {  
            while(i++ < 1)  
            {  
                while(i++ <= 2);  
                printf("%d", i); 5  
            }  
        }  
    }  
}
```

main()

{

int i=1;

do

while (i++ <= 1);

while (i++ <= 3);

while (i++ <= 5);

printf("y.d", i);

}

int i=1;

do

{

while (i++ <= 1);

}

while (i++ <= 3);

}, while (i++ <= 5)

}, while (i++ <= 5);

printf("y.d", i); p.f("y.d", i);

7

Low level programming (Number system)

main()

{

int a = 12;

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

printf("%o\n", a);

printf("%x\n", a);

printf("%X\n", a);

printf("%#x\n", a);

printf("%b", a);

getch();

}

O/P: 12

14

014

C

C

0xc, %d

decimal = base 10 : (0-9)

Octal = base 8 (0-7)

Hexadecimal = base: 16 (0-15) 10-A, 11-B, 12-C, 13-D,

14-E, 15-F, 16-G, 17-H, 18-I, 19-J, 1A-K, 1B-L, 1C-M, 1D-N, 1E-O

Convert to Hexa and Octal

100

%o (Octal)

%d

. X (Hexa)

30

8 | 100

100

16 | 100

20

8 | 12 - 4

1 - 4

6 - 4

main()

1 - 4

0x6.4

{ int a = 100;

$\frac{8 \times 4 = 32}{8} + 4 = 100$

100

b = 30;

$\frac{8 \times 3 = 24}{8} + 4 = 30$

96 + 4 = 100

c = 20;

$64 + 32 + 4 = 100$

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

$\frac{16 \times 30 + 1}{1} = 14$

printf("%o\n", a);

8 | 30

1E

printf("%x\n", a);

3 - 6

0x36

$\frac{8 \times 3 = 24}{8} + 6 = 30$

24 + 6 = 30

Add

$$3+4+5 = 12$$

1 80 1

0743

12 Not present in octal. That way

0124

Convert ~~12~~ 8/12

0635

1-4

01714

represent it like this
4 at sum side & 1 as added to
next addition.

Subtract

In octal subtraction ~~borrow~~

0476

borrow

- 0237

0237

Haxa

0x174

0x9d6

0x732

0x92d

0x6e5

0x2ae

0x936

0xa6c

0x486

0x14d6

0x167c

0238 (In ~~oct~~ octal 8 Not present)

- 0124

(0-7)

~~0135~~

No. - Binary code

$1 \rightarrow 1$

$2 \rightarrow 10$

$3 \rightarrow 11$

$4 \rightarrow 100$

$5 \rightarrow 101$

$6 \rightarrow 110$

$7 \rightarrow 111$

$8 \rightarrow 1000$

$9 \rightarrow 1001$

$10 \rightarrow 1010$

$97 \rightarrow 1100001$

Note:- Any Number Converted to
binary divisible by 2
Octal = div 8

decimal div - 10

Hexa div - 16.

$$2^1 - 1 \rightarrow 1 \rightarrow 1$$

$$2^2 - 1 \rightarrow 11 \rightarrow 3$$

$$2^3 - 1 \rightarrow 111 \rightarrow 7$$

$$2^4 - 1 \rightarrow 1111 \rightarrow 15$$

$$2^5 - 1 \rightarrow 11111 \rightarrow 31$$

$$2^6 - 1 \rightarrow 111111 \rightarrow 63$$

$$2^7 - 1 \rightarrow 1111111 \rightarrow 127$$

$$2^8 - 1 \rightarrow 11111111 \rightarrow 256$$

Add

$$\begin{array}{r} 1 \ 1 \ 0 \ 1 1 \\ + 1 \ 1 \ 1 \ 0 \\ \hline 1 \ 0 \ 0 \ 1 1 \end{array} \quad 1+1+1=3 \text{ binary } = 11$$

$$\begin{array}{r} 1 \ 0 \ 1 \ 0 1 \\ + 1 \ 1 \ 1 \ 1 \\ \hline 1 \ 0 \ 0 \ 1 0 \end{array} \quad 1+1=2 \text{ binary } = 10$$

$$\begin{array}{r} 1 \ 1 \ 0 \ 0 1 \\ + 1 \ 1 \ 1 \ 1 \\ \hline 1 \ 0 \ 1 \ 1 0 \end{array} \quad 1+1+1+1=4 \text{ binary } = 100$$

$$\begin{array}{r} 1 \ 1 \ 0 \ 1 1 \\ + 1 \ 0 \ 1 \ 1 0 \\ \hline 0 \ 0 \ 1 \ 0 1 \end{array}$$

$$\begin{array}{r} 1 \ 1 \ 0 \ 1 1 \\ + 1 \ 0 \ 1 \ 1 0 \\ \hline 0 \ 0 \ 1 \ 0 1 \end{array}$$

Whenever a variable is declared a physical memory is created as per the range of bytes
for example: char ch (1 byte = 8 bits)

$$11111111 \rightarrow 2^8 \rightarrow 255$$

By this way possible values will be accepted

$$\underline{0 \text{ to } 127} \text{ and } \underline{-1 \text{ to } -128} \quad (\underline{256})$$

Every Number in friendly binary code is present they will be converted into the binary format.

A Number can be represented in either in 8 bit (08) 16 bit.

1111 1111

$$2^7 + 2^6 + 2^5 + 2^4 + 2^3 + 2^2 + 2^1 + 2^0$$

$0 = +ve$

$$0000\ 0000\ 0000 \rightarrow 0$$

This is reverse combination

'0' reverse is -1

32767 reverse is -32768

first indicate sign
 $+ve$

$$0111\ 1111\ 1111 \rightarrow 32767$$

$$1000\ 0000\ 0000 \rightarrow -32768$$

$$1111\ 1111\ 1111 \rightarrow -1$$

Reverse Condition

$$0 \dots -1$$

$$5 \dots -6$$

$$10 \dots -11$$

$$100 \dots -101$$

$$1 \dots -2$$

$$-100 \dots 99$$

$$111111111011 \dots -5 \quad 4 \dots 0.000\ 0000\ 0000\ 0100$$

-100

99 = 0000 0000 0110 0011

-100 = 1111 1111 1001 1100

32767

0111 1111 1111

+

1

0000 0000 0001

1111 1111 1111
1000 0000 0 0⁰ = -32768

borrow 2 from last present 1

-32768

1000 0000 0000

the way of getting

0 000 0 000 0 001

borrow remaining

0111 1111 1111

0 values contain 1
because it travel

2 to one place to last.

-32768

1000 0000 0000

+

0.000 0000 0000

-1

000 0000 0000 1

1000 0 000 0 000

0111 1111 1111

0111 1111 1111

= 32767

Main()

{

int i;

for(i=0; i<255; pf("THE ASCII OF: %c is %d\n", i, iff))
 getch();

{

Replace one digit with another:Main()

{

long n, rev=0, R>0, t;

int d1, d2;

pff("Enter Num");

Sf(" %d", &n);

Pf(" select two digits from above no to replace
one digit with another");

Pf(" enter first digit, second digit");

Sf("%d %d", &d1, &d2);

fall(n);

rev = rev + t * 10;

n = n / 10;

{

fall(rev);

t = rev % 10;

if(d1 == t)

d = d * 10 + d2;

else

d = d * 10 + t;

rev = rev / 10;

3

psl' after replace: $x.d^6x$:

{ gelehr,

III III III III — 65535

III III III III

1 7 7 7

III III III III III = 15

F F E F

0cfal = F

main()

{

44
7710

unsigned u;

for(u=0; u<=0; u--)

printf("xd--%d--%d... %X\n", u, u, u);

5

0ff' 0---0 --- 0

FFFF = III III III III I — 1 - 65535 --- 177777 - ffff

ffe = III III III III I I 0 " 2 -- 65534 --- 177776 -- fffe

ffd = III III III I I I 0 3 -- 65533 --- 177775 -- fffd

4 - - 65532 -- 177774 -- fffc

The program tells you which are the numbers are equivalent with their binary codes.

Unsigned int

%d	y. u
-1	1111 1111 1111 1111
0	0000 0000 0000 0000
-32768	1000 0000 0000 0000
32767	0111 1111 1111 1111

Bit wise operators:

1's complement (\sim) is Nothing but reverse.

Left shift

Right shift

Bit wise and

Bit wise exclusive or

Bit wise or

(1) Main()

{

p.f("%d", ~-32767);

getch();

}

O/P = -5

(2) Main()

{

p.f("%d", ~-32767);

getch();

}

= 22

O/P: -32767

(3) Main()

{

p.f("%d", ~-32767);

getch();

}

P:

$\text{printf}("%d", \sim 32768); \quad \text{o/p: } -32768$

Note: 1's Complement is Known as bit Reverse Combination.

5×2^1	$5 \ll 1$	0000 0000 0000 0101	10
5×2^2	$5 \ll 2$	0000 0000 0000 1010	20
5×2^3	$5 \ll 3$	0000 0000 0010 1000	40
5×2^4	$5 \ll 4$	0000 0000 0101 0000	80
5×2^5	$5 \ll 5$	0000 0000 1010 0000	160
11×2^2	$11 \ll 2$	0000 0000 0010 1100	44

System
o/p:
 $20/2^1$ $20 >> 1$
 $20/2^2$ $20 >> 2$
 $20/2^3$ $20 >> 3$
 $20/2^4$ $20 >> 4$
 $20/2^5$ $20 >> 5$
 $0111 = 7 >> 2$
 $20/2^2$

0000 0000 0000 0000	$20 = 1.0100$
0000 0000 0000 1010	$\rightarrow 10$
0000 0000 0000 0100	$\rightarrow 25$
0000 0000 0000 0010	$\rightarrow 2$
0000 0000 0000 0001	$\rightarrow 1$
0000 0000 0000 0000	$\rightarrow 0$
0000 0000 0000 0001	$\rightarrow 1$

Note: Any Bit wise operator will not work with floating point

$\text{printf}("%d", 8 \ll 2); \quad \text{o/p: } 24 \quad (8 \times 2^2)$

$\text{printf}("%d", 12 \gg 2); \quad \text{o/p: }$

Note: Any Number 16 times left shift leads to zero (0)
Ex: $7984 \ll 16 \rightarrow \text{o/p: } 0$

→ If it is Even Number 15 times leftshift
leads to Zero.

→ If it is an odd Number 15 times leftshift
leads to -32768

`printf("%d", -10 >> 1);`

-10

reverse

9

0000 0000 0000 1001

-10

1111 1111 1111 0110

thus one bit right shift

1111 1111 1111 0111 → 0111 = 5

+5 = ~~10000 0000 0000 0100~~ Here '0' is in the ~~last~~ place of
Replace it get -5 5 so, ~~0~~ → -5

`printf("%d", -8 >> 2);`

0110 -2

Note: Any \rightarrow re Integer Right-shift above 15 times
always leads to -1

Ex: $-9845 \gg 15 \rightarrow 0110$

Bitwise And (&)

Bitwise Exclusive OR (^)

Bitwise OR (|)

A	B	R
0	0	0
0	1	0
1	0	0
1	1	1

A	B	R
0	0	0
0	1	1
1	0	1
1	1	0

A	B	R
0	0	0
0	1	1
1	0	1
1	1	1

8

0000 1000

10

0000 1010

8810

1000 = 8 8 = 11 = 1 remain = 0

8 | 10

1010 = 10 00 = 0 remain = 1

8 | 10

0010 = 2 + 10, 01 = 1 remain = 0



→ Write a program accept 2 No's. find the product of 2 No's
using bitwise operators.

→ If $x = 7$ $y = 9$

$x^1 = y^1 = z^1 = y$ After then $x = ?$ $y = ?$

$$x^1 = y \Rightarrow x = x^1 y = \begin{array}{r} 0111 \\ 1001 \end{array}$$

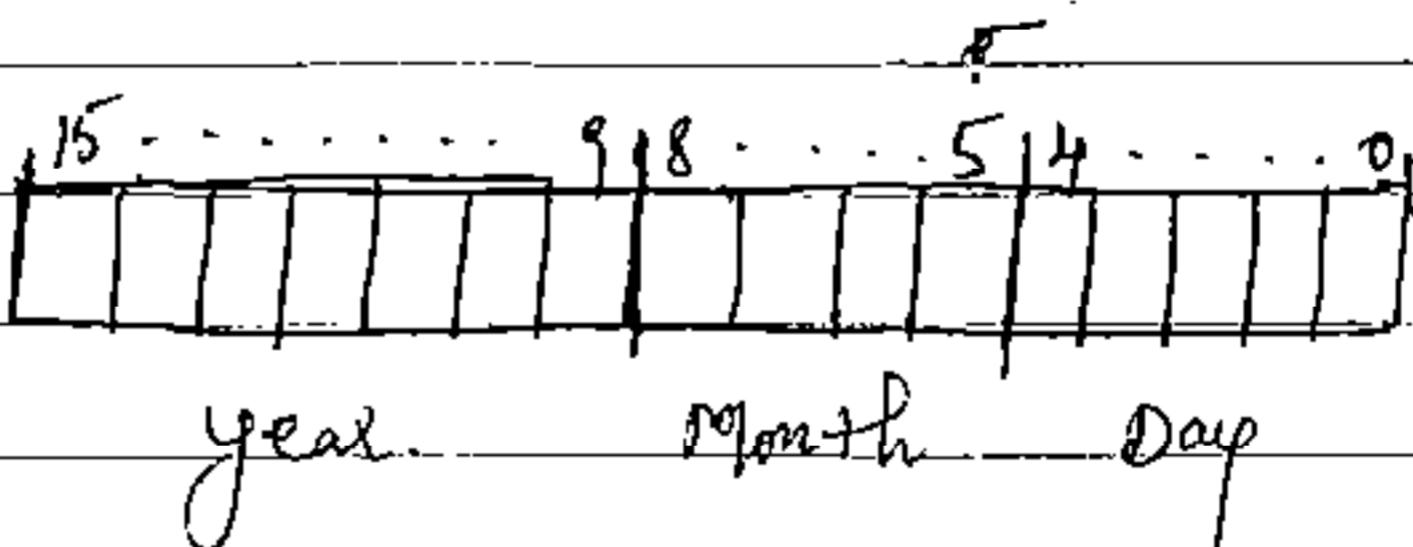
$$y = y^1 x = \begin{array}{r} 1001 \\ 1110 \end{array} \quad x = \underline{\underline{1110}} \\ y = \underline{\underline{0110}}$$

$$x = x^1 y = \begin{array}{r} 1110 \\ 0110 \end{array}$$

$$x = \underline{\underline{1001}} = 8 \text{ } \therefore x = 9, y = 7$$

```
#include <stdio.h>
Main()
{
    int a, b, result;
    printf("Enter the Two Numbers:");
    scanf("%d%d", &a, &b);
    result = 0;
    while(b != 0) // iterate the loop
        till b == 0
    {
        if(b & 1) // logical ANDing of the value of b with 01
            result = result + a; // update result with new value
        a = a << 1; // left shift value contained in 'a' by 1
        b = b >> 1; // Right shift value contained in 'b' by 1
    }
    printf("Result: %d", result);
}
```

Where Bitwise Operators Can Utilised:



In Dos & Windows O.S convert the actual data into 2 bytes, by implementing a formula.

$$\text{date} = 512 * (\text{year} - 1980) + 32 * \text{month} + \text{day}$$

Year starts from 9th position. month is starts from 5th position.

Suppose 09/03/1999 then the conversion

$$512 * (1999 - 1980) + 32 * 3 + 9$$

$$512 * 10 + 96 + 09$$

$$5120 + 105$$

$$\underline{5225}$$

$$5225 = 0001010001101001$$

The binary equivalent of 5225 is

0001010001101001 this binary

Value is stored in the date field in

the directory.

0001010001101001
year mm dd

In this that the year month & date
existed as bunch of bit's in continuous memory
locations. To get an year we have to separate
from the 2 bytes entry by using << & left shift
& Right shift.

Extracting an year:

1) To Extract an year from the date
shift-left of date 9 times & then add 1980

date = 0001010001101001

Year = 1980 + (date >> 9)

$$0000\ 0000\ 0000\ \cancel{0101010} - 10 + 1980 = 1990.$$

Extract Month:

To Extract a month leftshift the date by
7 times & then right-shift by 12 times

month = ((date << 7) >> 12)

$$0001010001101001 << 7$$

$$0000100100000000$$

$$0011010010000000 >> 12 \quad 0000000000000011 \rightarrow 3$$

Extract day:

To Extract a date 1st left shift by 11 times & Then Right shift by 11 times.

date: 0001010 001101001 <<11

0100 1000 0000 0000 >>11

0000 0000 0000 1001 → 9

Void main()

}

unsigned int d = 9, m = 3, y = 1980,
year, month, day, date;

date = (y - 1980) * 512 + m * 32 + d;

printf("In Date = %u", date);

year = 1980 + (date >> 9);

month = ((date << 7) >> 12);

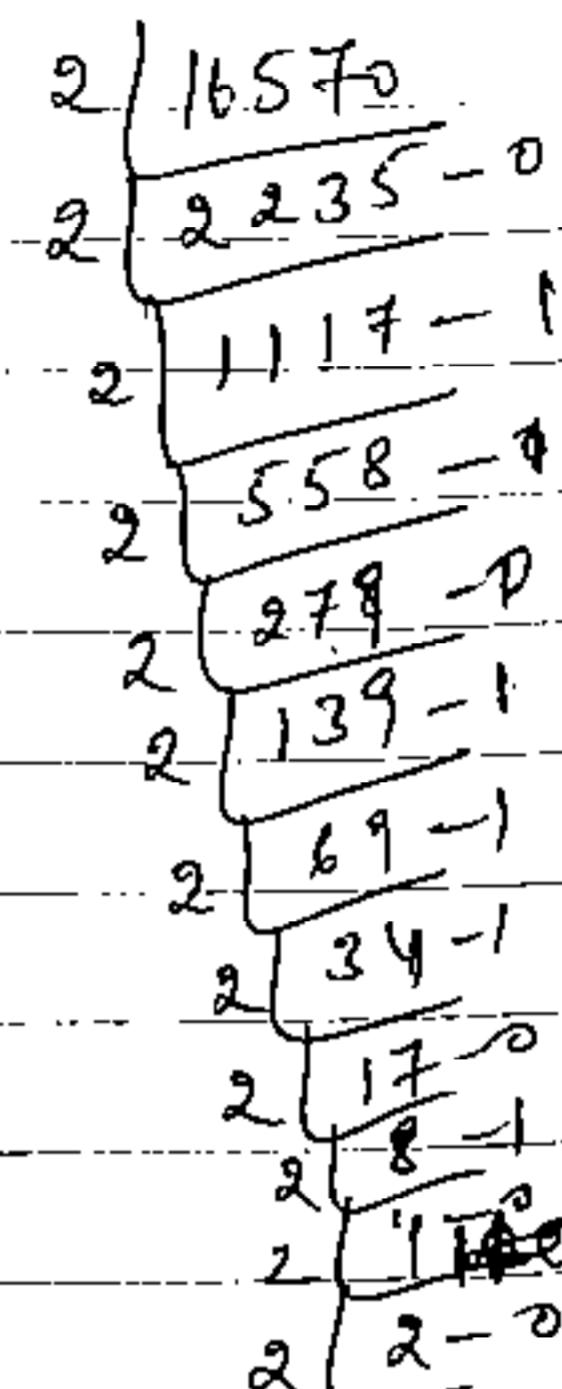
day = ((date << 11) >> 11);

printf("In Year = %u", year);

printf("Month = %u", month);

printf("Day = %u", day);

}



- Write a program accept a number display the binary code for that Number in 16 bit format. (Bitwise operators using)
- write a prog accept a Number find out the given no is even (e) odd

Preprocessor Directives:

The preprocessor directive is the second stage of your C program, all the preprocessor directives are defined with #. The job of the preprocessor directive whichever the Macros are present in your program converted into Highlevel language. This preprocessor job will be taken place before compilation.

There are 4 types of preprocessor directives.

1. Macro Expansion.
2. File Inclusion.
3. Conditional Compilation.
4. Miscellaneous directives

A predefined preprocessor or Compiler directive which evaluates the related code with high priority can be called as preprocessor directives.

Macro Expansion:

This Macro Expansion can be used in three ways.

- i. To Generate Constants. EX: #define A 10

The left-side will be replaced by the right-side

```
#define A 10
```

```
void main()
```

```
{ int i; // left-side will be replaced by the right-side  
i = A;
```

```
p.f("x d x d", i, A);
```

```
}
```

→ #define ISEMPTY top NULL

```
→ #define size 10  
int a[size];
```

ii) alias Name:

187

(2) It provides alias names for existed data types & functions.

Ex: `#define L long int
#define PF printf`

(3) iii) code replacement:

`#define MUL(a,b) a*b`

void main()

{ int i;

$i = MUL(2,3);$ it replaced as $2 * 3 = 6$

`printf(" %d", i); // 6`

$i = MUL(2+3, 2+3);$ $2+3 * 2+3 =$

`printf(" %d", i); // 11` $2 + 6^{\textcircled{1}} + 3 = 11$

}

`#define MUL(a,b) (a)*(b)`
we get Expected op
par main function ()'s

The macros will replace total struc without any changes.

Testing of the Macro's (preprocessor directive)

a preprocessor directive can be tested how the macro is expanding by saving the file in the s/w directive.

the Macro expansion can be observed i.e. how the implemented code is generated. as the compiler not understand the macro's they will be expanded, the expanded program can be send to the compiler.

→ Save the program with .c extension in the s/w directive
`C:\tc\bin>filename.c` → s/w directive.

After saving the file open the DOS prompt from the editor

`D:\TC\BIN>`

→ My Prog in 'C' Drive Turbo C2

C:\TURBOC2\CPP <fibName.c>

After this Command

An intermediate code is generated without any Macro's.

C:\TURBOC2\ type m.i → Total program will converted without Macro's.

~~prog~~ Write a program Create a Macro Just Similar to an algorithm by replacing the Structs.

```
#include <stdio.h>
#include <conio.h>
#define begin main()
#define print(a) printf("%d");
#define end getch();
#define number int
#define charfor char
#define printint(a) printf("%d", a);
#define readint(a) scanf("%d", &a);
#define printchar(a) printf("%c", a);
```

Save → Save the file with .c (or) .h Extensions.

c:\alg.c Open a New file call this program by using file inclusion Macro (#include) & execute the program by placing the programming contents.

```
#include "c:\alg.c"
```

begin

number n;

Character c;

print(Enter Any ^{value} Number);

readint(n)

c = n;

print(Given value:)

print(c)

printchar(c)

end

O/P:

Enter Any Value 69

Given value: 69 E

→ ASCII

Example:

#define AND &&

#define OR ||

Macros with arguments:-

#define AREA(x) (3.14 * x * x)

a Macro's can also be present with more than one line. if a macro is present with more than one line then the macro will be ended by using backslash that indicates the macro is still continued.

#define PRINT(n,c){\

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

\}

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

 printf(" ");\

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

 printf("%c", c);\

 printf("\n");\

\}

K.b

o/p

~~*~~ *

~~*~~ **

* *木

木木木木

*木木木木

main()

{

 i,j;

#

#

 clrscr();

#

 PRINT(6,'*');

#

 PRINT(5,'#');

 getch();

/ In ~~and~~ Dos by using edit-m, i
we can show simplified prog-

```
#define size 100
```

```
void main()
```

```
{ int a;
```

~~int~~

```
a = #size; // Here in place of size 100 will come
```

```
printf("%d %d", size, a); // 100 is constant can't improve. (as)  
                                O/P: Error(L value required) - Devenuel
```

→

```
#define size a
```

```
void main()
```

```
{
```

```
int a;
```

```
a = 100;
```

```
a = #size;
```

```
printf("%d %d", size, a); // 101, 101.
```

```
}
```

Macros and Functions:

```
#define sum(x,y) x+y
```

```
/* int sum(int x, int y)
```

```
{ return x+y; } // 30 → In this program the
```

```
} */
```

macro's & functions defining
same type of output. That means

Macro's & functions - Not the same.

```
void main()
```

```
{
```

```
int s;
```

but - macro's run faster but

```
s = sum(10,20); increase the prog size, were as
```

```
printf("%d", s); 30 functions make the prog
```

```
}
```

small & compact.

if we use macro 100 times in the prog

the macro expansion takes place into our

source code 100 diff places thus increase the size of
the prog. On the other hand - a function if it is used

& if it call 100 times from diff places. it also can except

same amount of space in the prog.

but however functions will take some time to store the value in the parameters & returning back to the caller.

② File inclusion:

This is one type of macro which included to files in your prog. This file inclusion macro is used in 2 cases.

1. If the prog is very large the code is divided into several diff files each containing set of related functions, then this file inclusion preprocessor is used.

These files are included into at the main prog file.

2. Functions or Macro definitions are needed in many prog's in such situations. The commonly used functions & macro definitions are stored in a file & that file can be included in every prog & this programming stmts are included in one program as a .h file ② any other extension.

There are two ways to write
#include statement.

They are

#include "filename"

#include <filename>

The < > symbols will indicate to check the header files only in the S/W directives (TC / TURBO C2).
If does not check any local directives (C :)

(double quotes)

→ " " indicates, first checks in S/W directory.
if it is NOT available, then it check the
Local directory.

If it is user defined Header files (.h)
any other extension files use the " " (double
quotes)

Example: Macn3

define SQUARE(x) x*x

main()

{

int i;

i=64/SQUARE(4);

printf("%d", i);

}

$$64 / 4 * 4$$

$$16 * 4 = 64$$

② # define < stdio.h >

define a 10

main()

{

define a 50 // within {}

printf("%d", a); // 50

}

④ # define CUBE(x)(x*x*x)

main()

{

int a;

a=27/CUBE(3);

printf("%d", a); 1

}

③ # define MESS(m) printf("m"),
main()

{

MESS("Hai"); m

MESS("Hello"); m

}

⑤ # define CUBE(x),

(x*x*x)

main()

int a, b;

b=3; 14

27/3 = 9

a=CUBE(b++) / (b++ -

p.f("a=%d", b=a, a, b);

}

97

⑥ #define CUBE(x) (x*x*x)

main()

```
{ int a, b = 3;
    a = CUBE(1+b)/++b;
    p.f("a = %d b = %d", a, b);
}
```

1 90 1

⑦ #define CLSCL(100)

main()

```
{ clscl();
    p.f("%d\n", clscl());
}
```

100

⑧ #define LP2

#define HP10

#define AP HP+LP

main()

```
{ int i;
    i = AP*AP;
    p.f("%d", i);
}
```

⑨ #define FALSE -1

#define TRUE 1

#define NULL 0

main()

```
{ if (NULL)
    puts("NULL");
else if (-1)
    puts("TRUE"); or TRUE
else
    puts("FALSE");
}
```

⑩ #define PROD(a,b) a*b

main()

```
{ int x=3, y=4;
    p.f("%d", prod(x+2, y-1));
}
```

10

⑪ #define DIM(array, type) sizeof(array)/sizeof(type)

main()

```
{ int arr[10]; // array 10 block each block 2 bytes = 20
    p.f("The dimension of the array is %d", DIM(arr, int));
}
```

10

```
#define int char
```

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

```
{ int i=65;
```

```
printf("size of(i)=%d", sizeof(i)); //byte
```

```
}
```

③ Conditional Compilation Macro's:

This Conditional compilation macro's will be used whenever the compiler skipping some of the stmts of the source code by inserting processing commands, then we use:
Conditional compilation macro's.

This Condition-C macro's are diff types are available.

#if

#ifndef

#ifndef They are similar to if-else condition.

#else etc

Syntax:

```
#ifdef macroname
```

```
stmt1
```

```
stmt2
```

```
stmt3
```

```
#endif
```

When we working with this preprocessor (conditional compilation) depends on the condition it decide that code need to be compile or not.

Generally by using preprocessor we can reduce the size of the .exe.

① #define pf printf

Void main()

{

 pf("A");

 #if (7>11)

 pf("B");

 pf("C");

 #endif

 pf("Hello"); O/P: A Hello

}

• i file

void main()

{

 printf("A");

 printf("Hello");

}

② #define pf printf

Void main()

{

 pf("Hello");

 #if 5!=552

 pf("A");

 s!=1

 pf("B");

 #else

 pf("X");

 pf("Y");

 #endif

 pf(" welcome"); Hello A.B.Welcome

}

→ #define pf printf
 #define TEST
 void main()
 {
 pf("Hello");
 #ifdef TEST
 pf("A");
 pf("B");
 #endif
 pf("welcome");
 }
 {
 #define pf printf
 void main()
 {
 pf("Hello");
 #ifdef TEST
 pf("A");
 pf("B");
 #endif
 pf("welcome");
 }
 }
O/P: Hello A B welcome

→ The Conditional Compilation is used when the Compiler ~~is~~ skip some part of the Source code by inserting the processing commands. This processing command ~~#ifdef & #endif~~ if the macro has to be define the block of code will ~~not~~ be proceeded. Otherwise the block of code will not be provided.

#Undef :

by using #Undef we can close the scope of existing define Macro.

#define pf printf

#define A 10

void main()

{ pf("Y.d", A); }

#Undef A

#define A 20

pf("Y.d", A);

#Undef A

O/P : 10, 20, 30

#define A 30

pf("Y.d", A);

}

void main()

{ pf("Y.d", A); }

#Undef A

pf("Y.d", A); → Error

#define A 20

pf("Y.d", A);

Miscellaneous Directive:

192

#Error : Error Messages can Create

#Pragma : Directive

This directive is one special directive that is used to turn on or off certain features.

programs vary from one compiler to another.

0 // Supress Warnings → desirable workings

#pragma warn -ev1 /* return values */

#pragma warn -par /* parameter not used */

#pragma warn -reh /* unreachable code */

int f1();

+reh is enabled.

{ int a=5;

+ means enabled.

}

- means disabled.

void f2(int x)

{ p.f() "inside f2"(); }

}

int f3()

{ int x=6;

return x;

x++;

}

void main()

{ f1();

f2();

f3();

}

FUNCTIONS

A function is a self defined block ~~with void~~
which performs a predefined task.

Functions are of Two types

1. predefine function (library functions)

2. defined by the programmer

ex:- clrscr(); printf(); scanf(); etc

2. User define functions:

defined by the user

ex:- main()

→ Why main() is Call User define function?

1. The functionality what we provided
inside the main() function is depends on user
requirement. Depends on the user requirement
the functionality can be changes. And every time
we generate diff. executable files.

i.e (in main() function we write addition prog!
& in same main() another time subtraction like
diff. exec files).

A.

B.

→ Every operator returns a value similarly
function also returns a value. for deceiving the value is
optional.

→ The default nature of any function is integer

→ The return type of the main() can possible to
change

Char main()

{ printf("Y.d", sizeof(main())); }

}

O/p: 1

The return type of the main() function can be change as per the user requirement.

If the user want to fill the Nature of the function we can place void as a return type. void is a datatype which the size is Nothing.

```
Void main()
{
    No return-type
}
```

```
main()
{
    by default return-type is
    integer
}
```

Note → In program void present - return Not present - & vice versa.

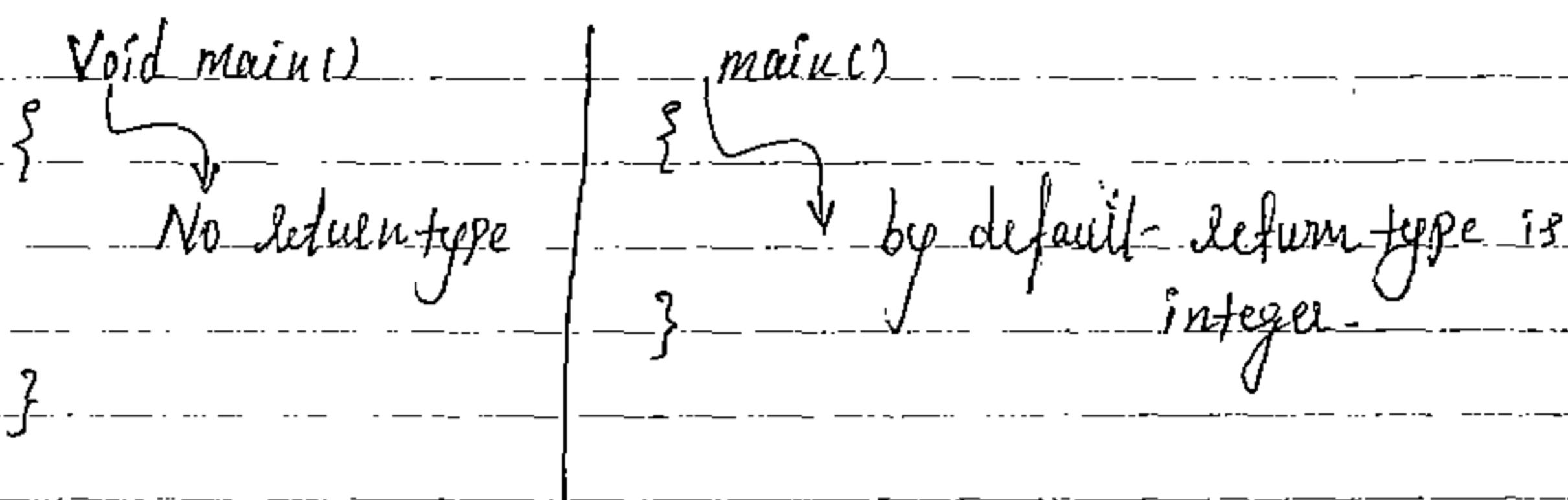
How to write a programming with functions.

To write a programming we have to know some Component Names of the function.

1. Function declaration (Function prototype)
2. Signature of a function.
3. Function Caller.
4. Actual arguments.
5. Function Calling (declarator).
6. formal parameters.
7. Function definition.
8. Return Stmts.

The return type of the main() function can be change as per the user requirement.

If the user want to fill the Nature of the function we can place void as a return type. void is a datatype which the size is nothing.



Note → In program void present - return Not present & viceversa.

How to write a programming with functions.

To write a programming we have to know some component names of the function.

1. Function declaration (Function prototype)
2. Signature of a function.
3. Function Caller.
4. Actual arguments.
5. Function Calling (declarator).
6. formal parameters.
7. Function definition.
8. Return Stmts.

1. Function declaration (F. prototype)

return-type function-name (arg₁, arg₂, ..., arg_n);

↓ return-type
based on o/p

Signature of function.

↓ arguments based on i/p.

return-type function Signature-

name ↓

int

sum (int, int);

function declaration

(def)

Void

nature (int);

prototype

Void

pattern (int);

long

fast (int);

The Function declaration is the first component of the functions in can also be called as prototype of the function.

The function prototype tells to the compiler the return type of the function, no. of arguments of the function, order of arguments of the function & type of arguments.

The declaration will be available for the predefined functions also in their header files.

The declaration contains the arguments, they are called as Signature of the function.

2. Function caller:-

The Function caller is one more component that can be present anywhere & any no. of times.

At the time of function caller we can pass the arguments can be Variable type, Constant Numbers & expressions.

Whenever function call taken place it jumps to the function callee (SubModule)

3. Actual arguments:-

At the time of function caller the arguments are called as actual arguments. This arguments contains the data that data will pass to the partial parameter.

At the time of function caller don't mention
the type of the Variables.

actual arg's are Variables, Constant's, expression

4. Function calling:

The function callee is Separate Module
always the function callee Separately will present.

One Submodule Can't Contain Another Submodule
The function callee will be present only one time,
but it can be used any no. of times.

5 Formal parameters:

At the time of function callee which ever receiving the values of actual arguments they are called as formal parameters.

This formal parameters must be Variable type.
The function callee contains the body of the program it's called as function definition.

The function callee don't have any Semicolon.

6. Return Statement

We are Making the Communication from the Caller to the Callee by arguments.

To send back the Communication back to the Caller from the Callee we need of Return Statement.

A Functions programming can be written 4 ways.

1. Function No argument & No return value closure
2. " " With argument & No return value ~~return~~ ~~value~~
3. " " with arguments & return value printf
4. " " No argument & return value getch

1. Function with No argument & No return value:

```
Void drawline(void); // declaration  
Void xyz(); // declaration  
int main()  
{  
    clrscr();  
    drawline(); // Caller  
    printf("Modular programming\n");  
    xyz(); // Caller  
    printf("Back to Main\n");  
    drawline(); // Caller  
    getch();  
    return 0;  
}
```

O/P - - - - -
Modular programming
It is Submodule -
Back to Main

void drawline(void) // callee

```
{  
    int i;  
    for(i=1; i<=40; i++, printf("-"));  
    printf("\n");  
}
```

void xyz() // callee

```
{  
    printf("It is Submodule...\n");  
}
```

Note: Always the compilation takes from top to bottom and execution from Main to Main.

→ Without declaration the program raises an Error but as per the C programs it can execute without declaration.

Void drawline(); // declaration

drawline(); // ^{int} by default
it is error return type not matching.
{
}
{

→ When you provide the definition first, & then ^{write} the main() program, then there is no necessary to provide the prototype. Since Compiler already taken place before execution.

Void drawline()

{

Write a prog Sum of 2 no's with
No arg's No return values

}

Void sum(Void); // declaration

Void main()

{

inf main()

{ inf a,b,c;

}

inf p.f("Enter 2 Values:");

sum(); // Caller P.f("Sum = ", &a, &b);

} // end main P.f("Sum = ", &c);

Void sum() // callee

{

C = a+b;

}

Write a prog accept a number find out Number is even @ odd

void even(); // declaration

(void odd(); // declaration)

int main()

{

even(); // caller

}

void even() // callee

{ p.f("Enter a Number"); int n;

if (n % 2 == 0); p.f("Yd", n % 2 == 0)

p.f("No. is even");

}

else

void odd() // callee p.f("No. is odd");

{ int n;

p.f("Yd", n % 2)

(a)

void even()

{ int n;

p.f("Enter n value");

s.f("Yd", &n);

if ((n % 2) == 0)

p.f("Yd even", n);

else

p.f("Yd odd");

getch();

}

2. Function with Argument & No return value.

! 96 !

In this category there will be a communication from Caller to callee but there is No communication from callee to the caller.

Write a program to print the following Series.

```
1.2
void series(int); // declaration
1.2.3
void main()
1.2.3.4 {
    int n;
    printf("Enter the row value:");
    scanf("%d", &n);
    series(n); // caller
```

```
}
```

Void series(int); // callee

```
{
```

int i, j, k = 1;

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

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

printf("%d", k);
 if(j != n)
 {

if(j != 1)
 {

K++;
else

K--;

```
{
```

K = K + n; printf(" ");

} // solution

→ Write a program to find the factorial for that Number.

Void fact (int); // declaration

Void main ()

{ int num;

p.flt

$$3! = 3 \times 2 \times 1$$

$$n(n-1) \dots 1$$

f=1

fact (); // caller

f*f

Void fact () // callee

{ int n;

| printf ("Enter a value: ");

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

for (i=n; i>=1; f=f*i-1)

{

f=f*n;

Void fact (int); // declaration

Void main ()

{ int num;

| p.flt ("Enter the Number: ");

| s.flt ("%d", &num);

fact (num); // caller

{

Void fact (int num) // callee

{ long f=1;

if (n==0 || n==1)

{

| p.flt ("Factorial is 1");

| getch();

| exit(0);

for (i=n; i>=2; f=f*i-1)

{

| p.flt ("Factorial of num is %d", f);

| getch();

{

Write a prog accept a number display the Number in words
 #include <stdio.h>

Void main()

{

int txt(long); // declaration

long int num, t;

clrscr();

p.f("Enter the num:");

s.f("%ld", &num);

if (num == 0)

p.f("Zero");

if (num >= 10000000)

{ t = num / 10000000;

if (t <= 20)

txt(t); // caller 1

else

{ txt(t / 10 * 10); // caller 2

txt(t % 10); // caller 3

}

txt(10000000); // caller 4

}

num = num % 10000000;

if (num >= 10000)

{ t = num / 10000;

if (t <= 20)

txt(t); // caller 5

else

{ txt(t / 10 * 10); // caller 6

txt(t % 10); // caller 7

}

txt(10000); // caller 8

num = num % 10000;

if (num >= 1000) 6925

{

t = num / 1000; // caller 9

if (t <= 20)

txt(t); // caller 10

else

{ txt(t / 10 * 10); // caller 11

txt(t % 10); // caller 12

}

txt(1000); // caller 13

{ num = num % 1000;

if (num >= 100)

{

txt(num / 100); // caller 14

txt(t % 100); // caller 15

}

num = num % 100;

if (num <= 20)

txt(num); // caller 16

else

{ txt(num / 10 * 10); // caller 17

txt(num % 10); // caller 18

}

getch();

}

txt (long sum) // function declaration

{

switch (num);

{

Case 1: p.f("One");
break;

Case 2: p.f("Two"); break;

Case 3: p.f("Three"); break;

Case 4: p.f("Four"); break;

Case 5: p.f("Five"); break;

Case 6: p.f("Six"); break;

Case 7: p.f("Seven"); break;

Case 8: p.f("Eight"); break;

Case 9: p.f("Nine"); break;

Case 10: p.f("TEN"); break;

Case 11: p.f("Eleven"); break;

Case 12: p.f("Twelve"); break;

Case 13: p.f("Thirteen"); break;

Case 14: p.f("Fourteen"); break;

15:

16:

17:

18:

20:

30: p.f("Thirty"); break;

40: p.f("FouLty"); break;

50: p.f("FIFTy"); break;

60: p.f("Sixty"); break;

70: p.f("Seventy"); break;

80: p.f("Eighty"); break;

90: p.f("Ninety"); break;

100: p.f("Hundred"); break.

Case 1000: p.f(" thousand"); break;

case 10000: p.f(" ten thousand"); break;

case 100000: p.f(" Lakh"); break;

case 1000000: p.f(" Crore"); break;

}

}

Advantages of Functions:

→ The functions provide Modularity So it is easy to code & debug.

→ Function Reusability i.e Once a function is defined it can be called from any other module without having to rewrite the same. This saves time in rewriting the same code.

Reusability Nothing but: Same code Reusability.

3. Function with Argument & return value:

In this category — The Communication b/w the sides

→ Write a prog accept a Number Count- No. of 0's in that Number in the binary format:

Memory Map

Instruction	Address	Value
countone	65510	
	65511	
num	65512	9
	65513	
A	65514	
K	65515	

int countone(int); // declaration

void main()

{

int num, K;

p.f(" Enter number: ");

s.f("%d", &num);

K = countone(num); // call

p.f(" Number of ones %d in %d", K, num);

}

int countOne(int num) // callee

```
{  
    int count = 0;  
    while (num) {  
        if (num > 2)  
            count++;  
        num /= 2;  
    }  
    return count;
```

(2)

int countOne(int num) // callee

```
{  
    int cnt = 0;  
    for (cnt = 0; num != 0; num >>= 1)  
        if (num & '0' == 1)  
            cnt++;  
    return cnt;
```

num	65516	X4280
	65517	
count	65518	X47
	65519	

→ factorial with Arg's & return value.

long fact(); // declaration

int main()

```
{  
    int num; long k;  
    p.f1° Enter the Num;  
    s.f1° Y.d; & num);
```

K = fact(num); // callee

p.f1° fact of num is Y.d; K, num);

}

long fact(int num) // callee

{
 long f = 1;

if (n == 0 || n == 1)

{

p.f1° factorial is 1);

R.getch();

} exit(0);

fact: n >= 2; f = f * n - 1

p.f1° factorial of num is:

Y.d, f);

getch(); return f;

}

→ Write a program accept 2 No's as a base & exponent
find the power value of the base.

1 / 99 /

Main()

abs = absolute

if convol. -ve no to +ve no

{
float a;

int n;

float power(float, int); // declaration

p.f("Enter base");

s.f("y.f", &a);

p.f("Enter EXPONENT:");

s.f("y.d", &n);

p.f("y.f raised to power y.d is y.f^n", a, n, power(a, n));

↳ callee

{
float power(float a, int n); // callee

{
int i;

float p=1;

if (n==0)

return 1;

else

for(i=1; i<abs(n); i++) abs = convert -ve values to +ve values

p = p*a;

return p;

}

→ Main()

{ p.f("Genius y.d"; fun(123));

{

fun(m-n)

O/P: 123 Genius 3

{ return(p.f("y.d", n));

{

① print 123

→ print return const no., characters = ?

char c₁, c₂, c₃, c₄;

int i₁, i₂, i₃, i₄;

float f₁, f₂, f₃, f₄;

Declaration

int sum(int, int);

Calling / caller

i₁ = sum(i₂, i₃);

Void sum(int, int);

sum(i₂, i₃);

float sum(float, float);

f₁ = sum(f₂, f₃);

char sum(char, int, float);

c₁ = sum(c₂, i₁, f₁);

Void Kilan();

← Kiran();

int Kilan(Void);

← i₁ = Kiran();

Void abc(int, char);

abc(i₁, c₁);

float Kilan(char, int,

float, float);

→ Write a program Display - the numbers in binary formate
of 16 bit.

int main()

{

unsigned int num;

Single Number can display in
16bit binary.

int i;

S. sf("u", &num);

for (i = 0; i < 16; i++)

{

p. sf("1.d", (num << i & 1 << 15) ? 1 : 0);

}

between 0

}

Number Control Display 16 bit Binary Number from 100

#include <stdio.h>

void main() { void showbits(int); // declaration

{

int j;

for (j=0; j<=32767; j++)

{

if (j == -32768)

break;

printf("In Decimal %d is Same as Binary %b, j);

showbits(j); // caller

}

}

Void showbits(int n) // callee

{

int i, k, andmask;

for (i=15; i>=0; i--)

{

andmask = 1<<i;

K = n & andmask;

K == 0 ? printf("0") : printf("%b", 1);

}

}

→ Write a program that returns sum of squares of all odd nos
from 1 to 25.

→ Write a program implement fiboacci series with functions.

→ void fun(int*)

{
 p-f("y.d"); } ; \therefore underscore itself is available.

}{
main()
{

 fun(23); OP: = 23

 return 0;

}

→ #include <stdio.h>

void main()

{
 int a=15;

 void cdecl fun1(); // declaration

 void pascal fun2(); // declaration

 fun1(a); // caller

 fun2(a); // callee

 getch();

}

void cdecl fun1(int a, int b) // Callee

{

 p-f("y.d x.d", a, b);

 void pascal fun2(int a, int b) // callee

{

 p-f("In y.d x.d", a, b);

}

As per the CTF This program raises an Err.
Since it is strong type checking. as per the C language
it will execute, since it is loosely type checking.

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

```
Void cdecl fun1(int, int);
```

```
Void pascal fun2(int, int);
```

```
Void main()
```

```
{
```

```
int a=5, b=5;
```

```
clscl();
```

```
fun1(a+a, a+a); 7 5
```

```
fun2(b+b, b+b); 6 6
```

```
getch();
```

```
}
```

```
Void cdecl fun1(int p, int q)
```

```
{
```

```
p.f1 edeel: yd yd (n, p,q); 7,5
```

```
}
```

```
void pascal fun2(int p, int q) pascal workflow from
```

```
{
```

Left to Right

```
p.f1° Pascal: yd yd, p,q); 5,7 5,7 predefined
```

```
{
```

Nested Functions

If More than One function fall in a Single line Then the principle is called Nested Function.

If is mostly useful when a Series of arithmetic transactions fall in a Single formula it will be implemented. The result of the particular function need to be dependent on the operation carried out by another function.

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

```
Void main()
```

```
{
```

```
int add(int, int);
```

```
int mult(int, int);
```

```
int sub(int, int);
```

```
int div(int, int);
```

```
int a, b, c, d, f, g, h, R;
```

```
clear();
```

```
a = 3; b = 2; c = 4; d = 5; f = 8; g = 2; h = 6;
```

```
R = sub(mult(add(a, b), add(c, d)), add(div(f, g), h));
```

P.S. The Result of Expression is $y.d$; R;

```
}
```

```
float div(int x, int y)
```

```
{
```

```
return((float)x / (float)y);
```

```
}
```

```
int mult(int x, int y)
```

```
{
```

```
return(x * y);
```

```
}
```

```
int add(int x, int y)
```

```
{
```

```
return(x + y);
```

```
int sub(int x, int y)
```

```
{
```

```
return(x - y);
```

```
}
```

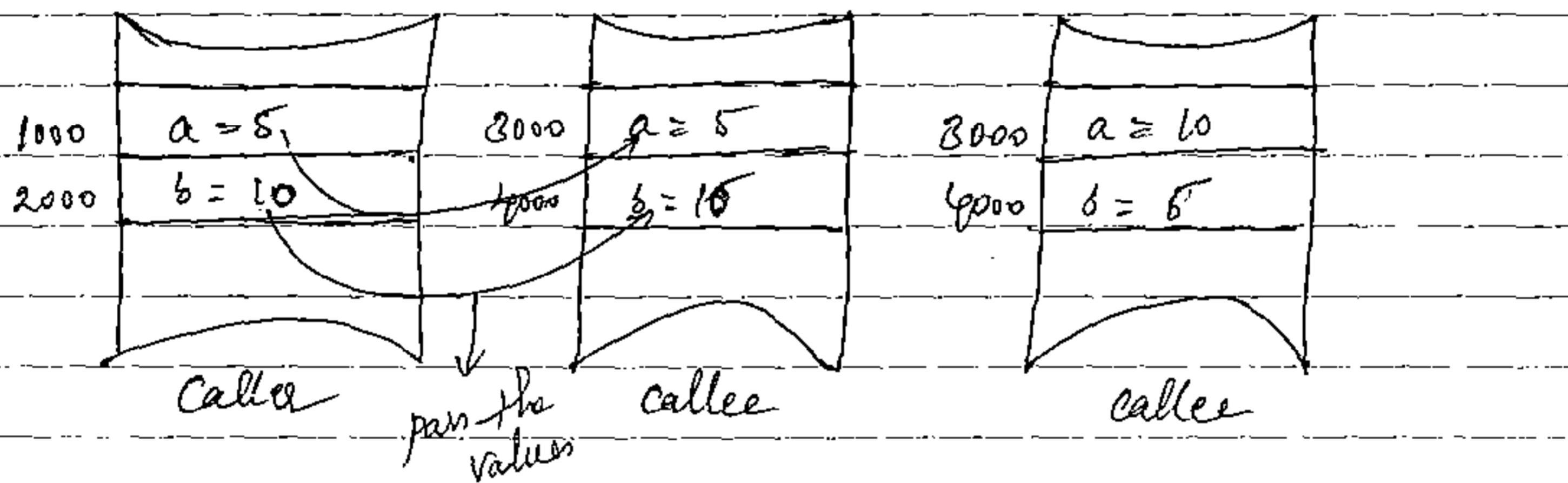
a function call's can be taken place in Two ways.

1. Call by value.
2. Call by Address.

1. Call by value:

If is the default designation when a function is calling. Sending the value of a Variable from the Caller to the callee is called as call by value.

Any changes made in the subfunction it will not effect to the Caller function Variable Values.



Void swap(int, int); // declaration

Void main()

{ int a, b;

P. fl' Enter Two Numbers: ";

S. fl' >d% d", &a, &b);

swap(~~a~~^{ap}_b, ~~b~~^{afte}_a); // caller

P. fl' after Swapping is >d >d; a, b);

Void swap(int a, int b) // callee

{

$b = (a+b) - (a=5); \quad || \quad a^1 = b^1 = a^1 = b;$

{ P. fl' after swap: >d >d, a, b);

→ Write a prog accept decimal Number Convert into octal.

→ Write a prog implement the pascal triangle using functions.

Recitation:

How to call the functions in diff ways:-

return; // return only control.

1. void main()

{

return; // Not raises any Errors

} we are returning only Control Not Value

2. // returning control & value

return(ss); These are returns control & value

return(60);

return(x+y+z);

3. // Are these calls OK

calsum(a, 25, d); // these are also possible

calsum(10, 25, 3, d);

calsum(a, calsum(25, 10, 4), d);

calsum(a, 25, d) * calsum(a, 25, d) + 23;

// returning more than one value

main()

{ int a=10, b=21, c=30;

int s, p;

s, p = sumprod(a, b, c); // invalid

Right side only one variable

p.s{ x.d y.d, s, p);

sumprod(int x, int y, int z)

{ int ss, pp;

ss = x+y+z;

pp = x*y*z;

return(ss, pp); // not possible

}

// One more try to
main()

103

{ int a=10, b=20, c=30;

int s,p;

s = sumprod(a,b,c);

p = sumprod(a,b,c);

p.f1° Y.d Y.d", s,p);

}

sumprod(int x, int y, int z)

{

int ss, pp;

ss = x+y+z; This way also not possible. in this way

pp = x*y*z;

to solve

return(ss); // Secondary

return(pp);

}

// Only way out

main()

{ int a=10, b=20, c=30, s,p;

s = sumprod(a,b,c,1);

p = sumprod(a,b,c,2);

p.f1° Y.d Y.d", a,b);

}

sumprod(int x, int y, int z, ~~Code~~ int code)

{

int ss, pp;

ss = x+y+z; pp = x*y*z;

if (Code==1)

{ return ss;

else

return pp;

}

→ a Better way of implementation using Ternary operators.

Best way:

```
return (code == 1 ? x+y+z : x*y*z);
```

code == 1 ? return (x+y+z) : return (x*y*z); // Invalid
We can't use more than one return stnts.

→ return stnt can't accept the floating point values.
float variables are allowed

Recursion:

The function which calls by itself is called as Recursion.
Recursion is one of the process similar to Looping but whenever we don't know some particular conditions capable to answer, then Recursion will be taken place.

Recursion is nothing but Same function calling by itself any no. of times.

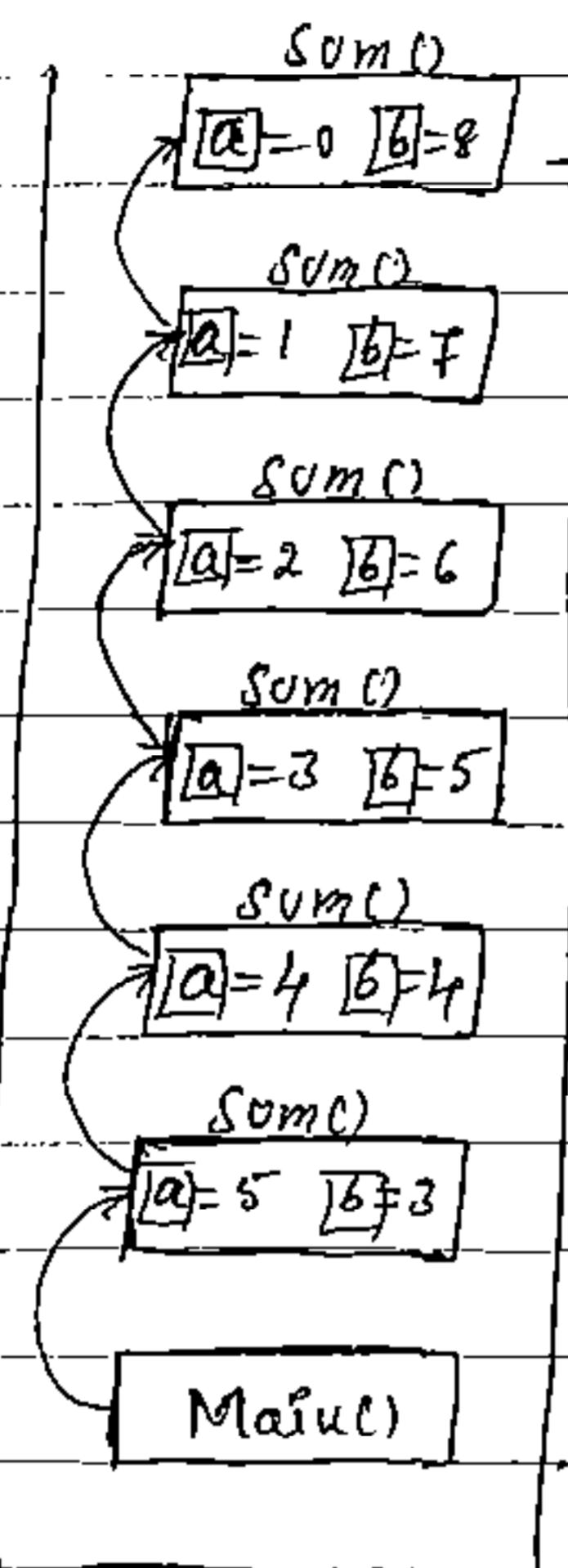
When Recursion is implementing there is no need to use any looping process. For conditional checking use the 'if' condition.

Main()

```
{ int sum(int, int); // declaration  
int a, b;  
P.f("Enter the values:");  
S.f(" x.d x.d", &a, &b);  
P.f(" x.d", sum(a,b));  
getch(); }  
else  
sum(-a, +b);  
}
```

int sum(int a, int b) // callee

```
{ if (a == 0)  
return b;
```



At this position my condition is satisfied after no. of function calls and the return value of '6' will be send from the function $\text{sum}(0, 8)$ to $\text{sum}(1, 7)$ - this value will go back from one function to another function - the flow will be continued on file from where we call.

The total depletion is taking place on stack (Last in first out).

In our program the i/p's 5 & 3 the stack is taking place nearly 7 times with my i/p's are increasing, the

↑ stack. Runtime stack growth will increase.

(This concept is known) indirectly performance of the system will be degraded.

The application working on recursion
Start → Search → Files & Folders.

When function is working it takes the performance on CPU.

How many times the stack is incremented that many of times the stack has to decrement until come back to the original call.

(1) void lee(int i)

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

if(i<=2)

return (i+1)

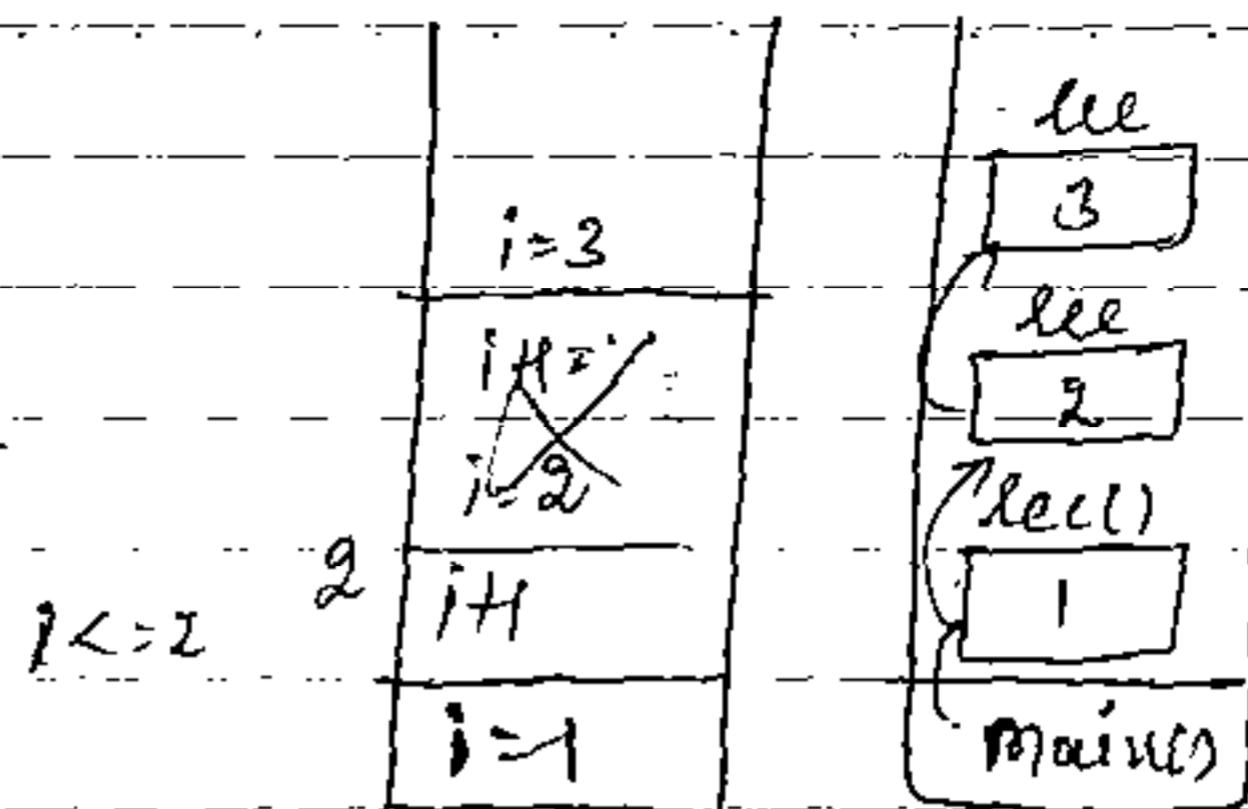
}

void main()

{

lee();

}



(2)

void lee(int i)

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

if(i<=2)

lee(i+1);

printf("%d", i);

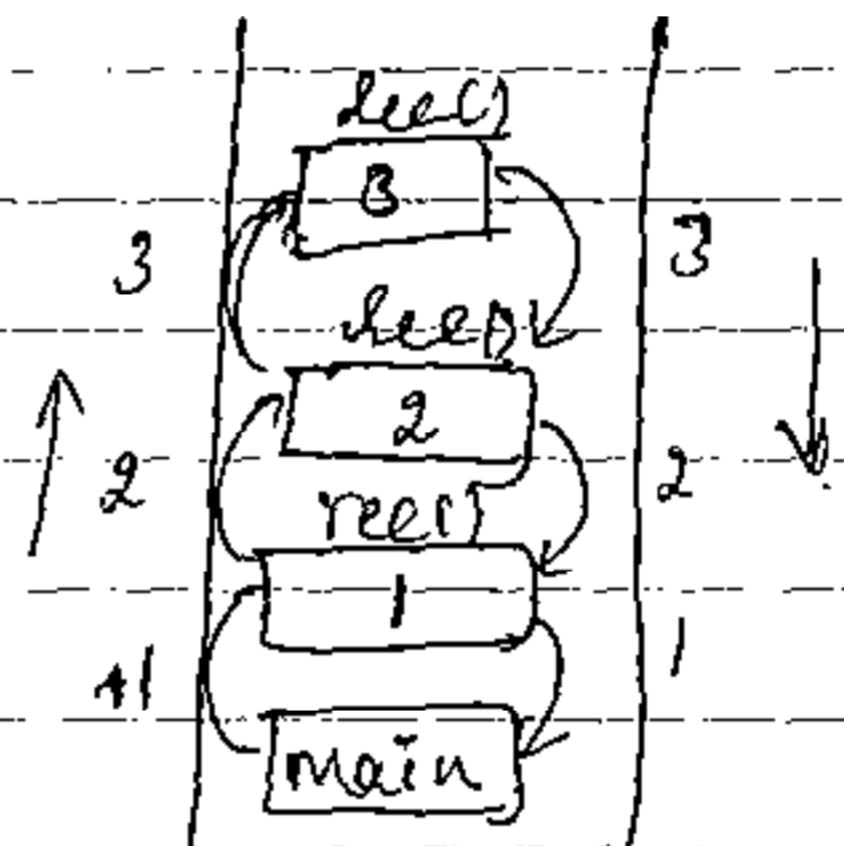
}

void main()

{

lee();

0/P!



}

33

2

t

→ Write a program how the system performance will be degraded when function calls are more

long fib(int);

void main()

{

int i;

printf("Enter the Rowvalue:");

scanf("%d", &i);

fib(n=0, n<2);

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

getch();

{

long fib(long n)

{ long fib;

if(n==0)

fib=0;

else if(n==1)

fib=1;

else

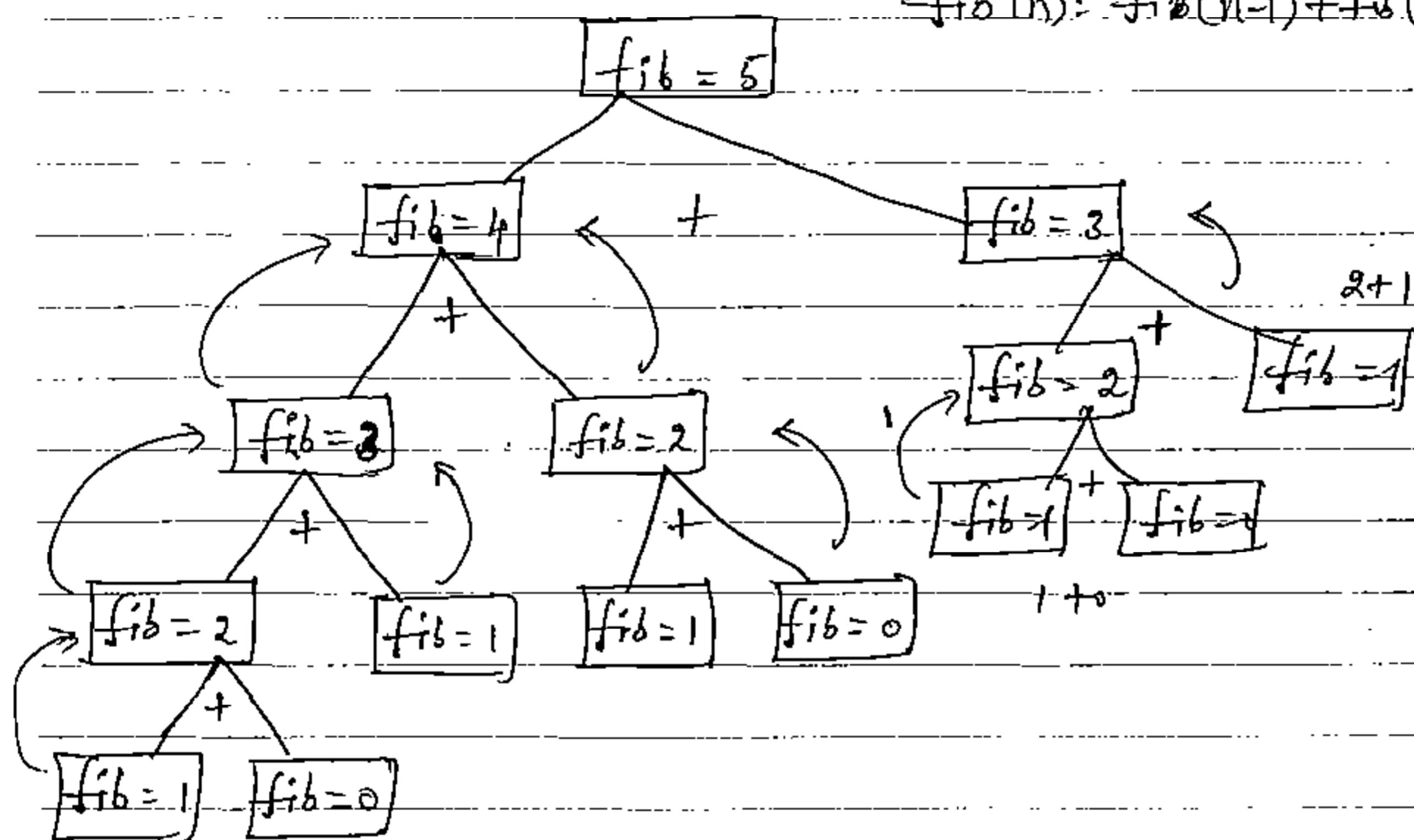
106

~~fib = fib(n-1) + fib(n-2);~~

return fib;

}

$$\text{fib}(n) = \text{fib}(n-1) + \text{fib}(n-2)$$



If my no is increasing - the winding (Top-down) of Recursion will be implemented how many times winding process taken place - that many of times unwinding (down-top) process will taken place!

Advantages & Disadvantages

1. Use of Recursion make the code more compact & elegant.

it simplifies the logic & makes the program easier to understand but the code written is less efficient. Since

Recursion is slow process because of No. of functions involved in it.

STORAGE CLASSES

Defining the Storage classes diff types of ~~Scope's~~ & lifes etc. can be taken place for the variables, the Scopes & lifes can be changes by the Storage classes.

Scope of Variables:

The Scope of Variables refers to the portion of the program, where the variables can be accessed. They are accessible in some portion of the program & in the other they are not accessible.

Scope of variables refers to in which set of variables can be manipulated.

Example of Scope:

Void Sample()

- a cov^x var { +a;
- a declare };

Void main() → Since 'a' variable is defined as
- a is with ^{main()} scope. a=10; local scope for the module main() by default it is an automatic storage
Sample(); class.

Sample(); Eller Raisen

Sample();

p.f1("d", a);

}

→ There are 2 types of scope's basically present if can be declared as Local Variables & Global Variables

1. Local Variables
2. Global Variables

The local variables that are declared inside a function.

their scope is only within the function in which they are declared.

This variable can't be accessed outside the function.

Life:

Lifetime refers to how long a variable will exist. OR defines its values.

Example: for life:

Whenever the scope is defined - the 1st block is called declaration block.

Every time of scope we can declare the variables.

void main()

{ int a;

a = 10;

p. f1 "%d", a); //1.0

{ int b;

b = 20;

a++;

b++;

printf ("%d", a); //11

p. f1 "%d", b); //21

}

p. f1 "%d", a); //11

p. f1 "%d", b); //Error (Scope already finished)

}

The scope & life will be changes depends on the storage class.

There are 4 types of Storage classes.

1. Automatic storage class
2. Static .. "
3. External (global) .. "
4. Register storage class

There are 5 types of Storage class specifiers.

1. Auto
2. Static
3. Extern
4. Register
5. type def

Storage class	specification	Initial value	Memory location	Scope	Life
---------------	---------------	---------------	-----------------	-------	------

automatic auto Garbagevalue Stack Within Entire body area the body

static static Zero PMA Within the entire function program

External extern Zero PMA all the entire functions program

register register Garbagevalue Register Within the entire body body

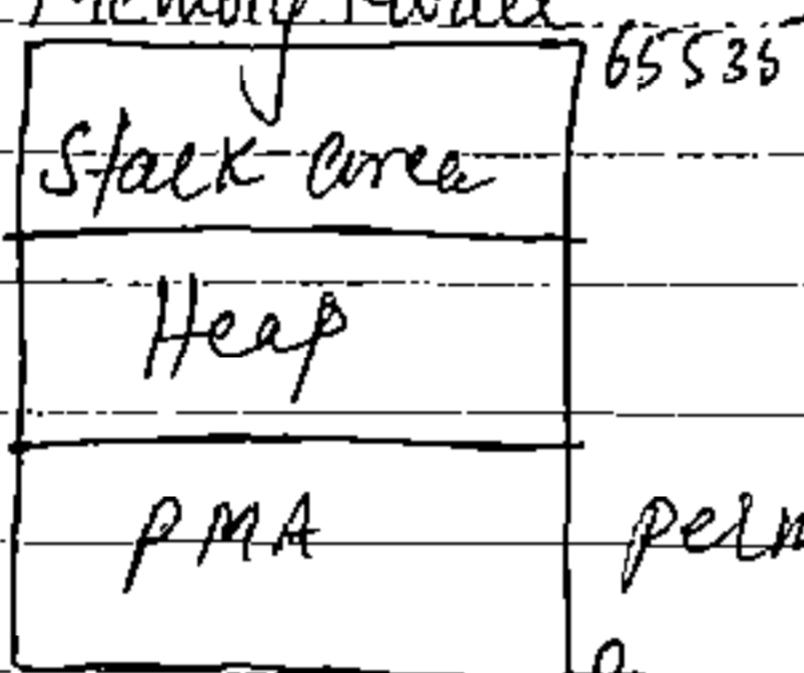
As per our system there are 3 types temporary areas like RAM, register, & Cache Memory are available.

Basically one data will be stored in RAM area
RAM contains 16 Segments (Large chunk of memory)
Each segment size nearly 64 KB

Our data will be stored in data Segment. Some other segments like Video, text, segment, Graphic segment, etc.

The data Segment contains 3 parts:

Memory Model



permitted Memory area.

The local variables of a program will store in Stack area, the static & global variables are stored in Permitted memory area. (PMA)

The Dynamic memory Concept will allocate the memory from the heap area.

→ ① Main()

{ auto int i;

p.f("Y.d", i); // due to "Automatic storage class"

}

the Variable Value is Garbage

② main()

{ int i;

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

p.f("Y.d", i, fun(i));

getch();

fun(i)=s) off 1, 10⁶

{ int X; 2, 10²

X=100; 3, 10³

X+=5;

defun(X); 10, 10⁴

}

Note:

automatic variables every time enter into the body will be initialized. (Once block will be closed it lost its previous values).

Static

1. Main()

{ static int i;

p.f("y.d", i); O/P: 0 (by default initial value of stack is 0)

}

Note: Static variables only one time will be initialised, every time of entry there is no initialisation.

(2) Main()

{ jump();

jump();

jump();

}

O/P: 0 404

1 404

{ static int i;

2 404

p.f("y.d%0.u", i, &i);

if;

}

(3) Main()

{ int i;

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

p.f("y.d%u", i, func(i));

getch(); // 1, 2, 3, 4 ... 10

}

func(int s)

{ static int x=100;

x+=s; // 101, 103, 105 ...

return(x);

}

(4)

Main()

{ int i; , , ,

for(i=0; i<5; i++)

{ static int a=0;

int b=0;

a+=i;

b+=i;

p.f("na=y.d", a);

p.f("nb=y.d", b);

}

pmA auto
a= 0 b= 91

3 X 5

a =

0/p: 1 1 1

2 1

3 1

4 1

5 1

6 1

7 1

8 1

9 1

10 1

11 1

12 1

13 1

14 1

15 1

16 1

17 1

18 1

19 1

20 1

21 1

22 1

23 1

24 1

25 1

26 1

27 1

28 1

29 1

30 1

31 1

32 1

33 1

34 1

35 1

36 1

37 1

38 1

39 1

40 1

41 1

42 1

43 1

44 1

45 1

46 1

47 1

48 1

49 1

50 1

51 1

52 1

53 1

54 1

55 1

56 1

57 1

58 1

59 1

60 1

61 1

62 1

63 1

64 1

65 1

66 1

67 1

68 1

69 1

70 1

71 1

72 1

73 1

74 1

75 1

76 1

77 1

78 1

79 1

80 1

81 1

82 1

83 1

84 1

85 1

86 1

87 1

88 1

89 1

90 1

91 1

92 1

93 1

94 1

95 1

96 1

97 1

98 1

99 1

100 1

101 1

102 1

103 1

104 1

105 1

106 1

107 1

108 1

109 1

110 1

111 1

112 1

113 1

114 1

115 1

116 1

117 1

118 1

119 1

120 1

121 1

122 1

123 1

124 1

125 1

126 1

127 1

128 1

129 1

130 1

131 1

132 1

133 1

134 1

135 1

136 1

137 1

138 1

139 1

140 1

141 1

142 1

143 1

144 1

145 1

146 1

147 1

148 1

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

```
int d();
```

```
int main()
```

```
{
```

↑
updation

```
for(d(); d(); d());
```

```
{
```

```
p.f("Y.d", d()); 5, 2
```

```
}
```

```
return 0;
```

```
}
```

PMA

defines first

```
int d()
```

num

~~def~~ between f (because num =)

```
int static num = 7;
```

~~px~~ num defines first

```
def num --;
```

f function calls.

```
}
```

External Storage classes: (Global)

```
int i; /* global variable */
```

```
main()
```

```
{ int i=10; /* local variable */
```

```
p.f("Y.d", i); 10
```

```
}
```

When local variable & global variable

present with the same name always preference

takes place for local variable. When local variable not

(2) Main()

present. It will be takes place for that

```
{ extern int i;
```

Global Variable

```
p.f("Y.d", i); // runtime error
```

```
}
```

when a local variable has been defined

```
int i; // definition with the keyword of extern it's
```

```
Main()
```

only the declaration, it

```
{ extern int i; // declaration doesn't provide any definition
```

```
p.f("Y.d", i); 0 whenever we define extern keyword
```

```
}
```

We have to once again define it outside the

Main().

diff b/w declaration & definition

declaration: only gives the type, status and nature of variable ~~without leaving~~ ~~reserving~~ any space for the variable.

definition: actual space is reserved for the variable and some initial value is given.

int i; // definition

main()

{ extern int i = 78; // declaration

p-f("x.d", i);

} // declaration Syntax error in function
because ~~at~~ ~~at~~ declaration time there is no space for String variables so we can't assign value.

① int i; // definition

main()

{ extern int i; // declaration

i = 78;

fun();

p-f("x.d", i);

} O/P: 78

fun()

{ p-f("x.d", i); } 78

}

② int g;

void xyz()

{ fg;

void main()

g = 2;

{ fg;

xyz();

}

Variable
Escaping from function

③ void xyz()

{ fg;

{ fg; // Error

→ Execution from main

{ int g; }

Void main() → compilation top to bottom

{ abc();

xyz();

}

Void Kilau() // declaration on NO

110

{ extern int g; // physical memory
+fg;

}

int g; // definition with physical memory

Void xyz()

{

+fg;

}

Void abc()

{ +fg;

}

Void Main()

{ +fg;

O/P: 4

xyz();

abc();

Kilau();

p.f("xyz", "g");

}

→ Void abc()

{ int i; s=0; i=1; } G

Static int s; i=1; s=0;

i = +fs;

p.f("xyz", "g", "d"); i, l, s;

if(i <= 2)

2, 2

3, 3

3, 3

abc();

p.f("xyz", "g", "d", "i", "s"); 2, 3

}

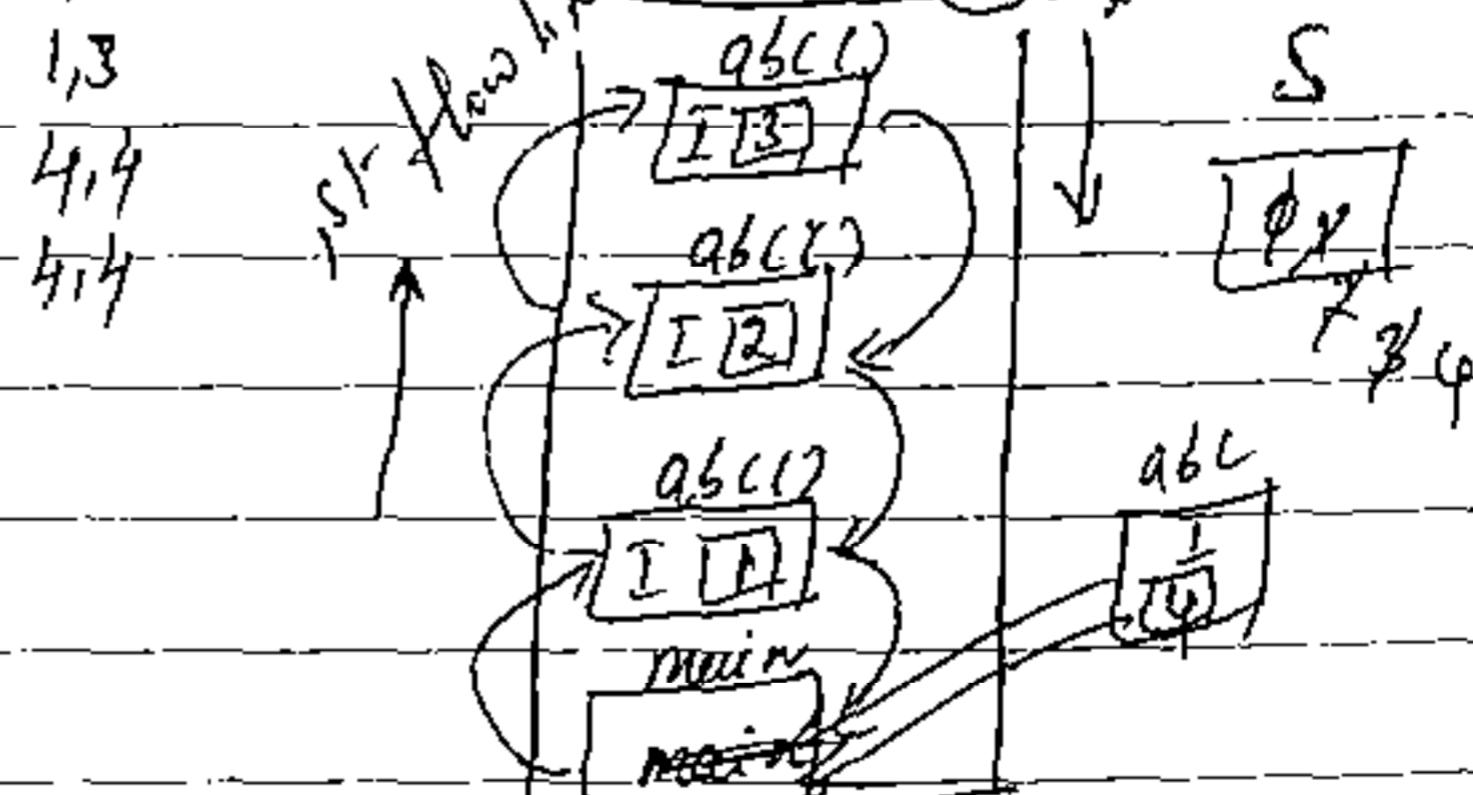
Void main()

{

abc();

abc();

}



→ Inf-g:

Void abc()

{ Inf-ab;

Static int as;

al = +fas;

+fg;

p.f("Y.d Y.d Y.d", al, as, g); l, l, 2

if(al <= 2)

2, 2, 3

abc();

3, 3, 4

3,

p.f("In Y.d Y.d Y.d", al, as, g);

}

Void Kiran()

off 1 1 1

1 1 2

2 2 3

3 3 4

3 3 4

+fg;

p.f("In Y.d Y.d Y.d", l, s, g); 1 1 1

2 3 4

1 3 4

abc();

2, 2, 4

2 2 5

if(l <= 2)

4 4 6

Kiran();

4 4 6

p.f("In Y.d Y.d Y.d", l, s, g); abc()

3 3 7

5 5 8

}

Void main()

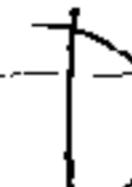
5 5 8

3 3 8

2 2 8

Kiran();

al=2



al=2

abc()

al=3

abc()

al=4

abc()

al=5

Kiran()

l=1

main

l=2

l=3

→ Main()

{ static int i=5;

if(c-i)

{ main();

}

p.f("Y.d", i); 00000

Void Kiran(int a, int b)

1 111

{
 printf("%d %d", a, b);

}

Void main()
{ int i = 1; auto i

 Kiran(i + i, i + i);

 Kiran(i + i, i + i);

 printf("%d", i);

}

Utilises the registers for accessing integers & char
Registers :- rather than float

Registers are the separate memory which are
part of the CPU register.

String in Registers we can access the data very fastly
rather than External RAM.

Registers has less no. of capacity basically we have
7 registers like Program Counter(PC), Stack pointer, Accumulators
etc.

Basically the size of Register is 2 bytes (Not only
Registers all storage class specifier has 2 bytes)

When to use register variables ?

main()

{ auto i;

 for (i = 1; i <= 255; i++)

 { i++;

}

}

255 times
memory
accessing

256 times
memory
accessing

main()

{ register i;

 for (i = 1; i <= 255; i++)

 { i++;

}

}

i

How many times 'i' Memory will be accessing?

In This programming nearly 766 times the
memory is accessing.

So instead of defining automatic storage class use the register to access the data very fastly.

→ Main()

```
{ register a,b;           // we can't trace it  
    printf("Enter the values:");  
    scanf("%d %d", &a, &b); Error  
    printf("%d", a+b);  
    getch(); o/p: Error Note: we can't track out the register  
    } (cp) addresses, because it is part of CPU.  
        like &a, &b.
```

→ Main()

{ Void fun(auto int)

```
{ p-f("%d", i);           A function work on CPU place a storage  
    } classes can't define for the parameters.  
    Storage class at RAM place
```

main() Error

{ fun(23);

}

→ void fun(register int i)

```
{ p-f("%d", i);           register part of CPU  
    }
```

Main()

{ fun(23); o/p: 23

}

#include<stdio.h>

p-f("%d", sizeof a); 2 bytes

main()

p-f("%d", sizeof s); 2

{ auto a;

p-f("%d", sizeof s); 2

register b;

p-f("%d", sizeof c); 2

static s;

p-f("%d", sizeof t); Error

extern e;

defun o;

typedef t;

}

't' is not acting as variable name, it is an alias name.

to define its size p.f("y.d", sizeof(t)); 2 bytes.

By default every storage class takes 2 bytes.

Note:

extern void func()

{

}

main()

{

func();

}

~~Static is with file scope~~

Static

file1.c

static void func()

{

} void func()

{

} void main()

{

func(); // Valid

func(); // Invalid

}

By default - a function ^{is} Global Extent
we can define extern keyword.

For a function we can also define static, but the function becomes to file scope. i.e. from locally function can access it is not possible to access from another program.

#include <c:\fib1.c>

void main()

{

func(); // Valid

func(); // Invalid

}

→ Register is also part of CPU.

allows Xyz(Register int -) → We can't specify storage

Xyz (auto int -) (more than input)

{ p.f("y.d",); } ;

→ xyz (int -)

{ p.f("y.d", -); 10; }

}

main()

main() - 'Underscore' is a Variable.

{ xyz(10);

}

xyz(10);

}

The Storage Class can't be defined for the

Parameters because function
Execution based on CPU. Storage Class part of a PGM.

POINTERS

→ System always use physical locations.
address locations.

$\&$ → address of operator

$*$ → value at address

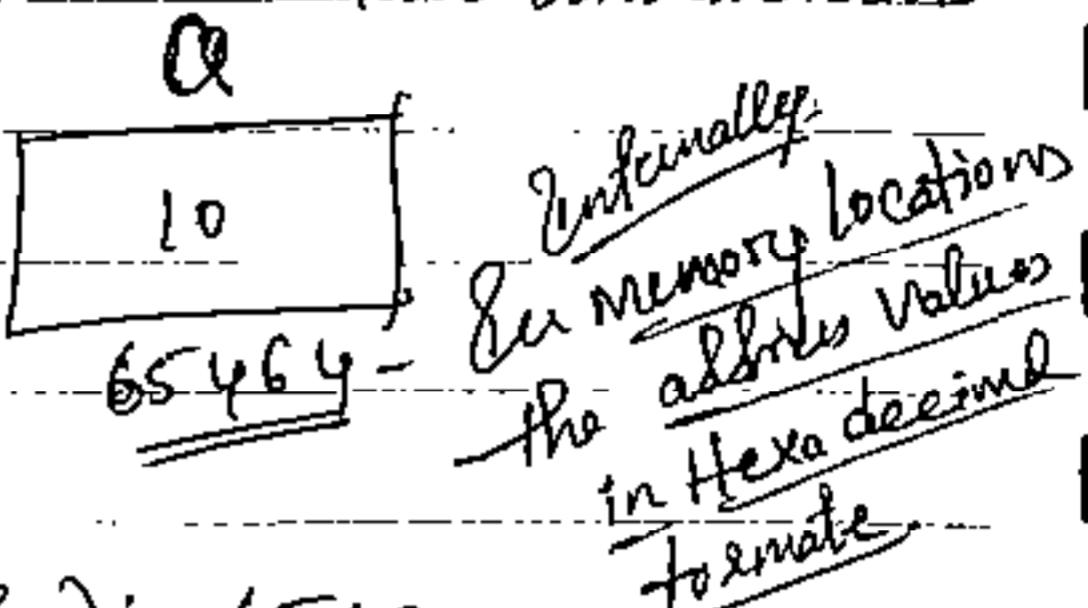
(indirection operator)

(Object at that location)

(dereference)

p-ff[°] Y.d.ln[°], *(int*)654949
 $\&$ &a

Above both are same.



→ Main()

{ int a=10;

p-ff[°] Value of a: Y.d.ln[°], a); 10

p-ff[°] Value Address of a: Y.d.ln[°], &a); 654949

p-ff[°] address of a: Y.d.ln[°], &a); FFD6 Hexadecimal format

p-ff[°] address of a: Y.d.ln[°], &a); Ffd6 specification

p-ff[°] Value of a: Y.d.ln[°], * &a); 10

p-ff(Y.d.ln[°], *a); Invalid, a is not address variable → Value at that address

Note: The symbol '*' works only for address indirectly (or) directly.

definition of pointer

a pointer is a derived datatype which can hold the address of another variable if can be Ordinary Variable or array Variable, Pointer Variable, user defined datatype variables.

If can also called the address of a function

Syntax:

hold

datatype * Variable name = <initialization>

Ex:

int * s; → The way of defining this

float * k; 's' is a variable of type

char * i; integer pointer.

{ main()

int a=10; 1 → p1 is a variable of type integer pointer

int * p1; 2 → * p1 gives integer

3 → Int. p1 we can store address integer variable

4 → pte is called as pointer variable of type integer
 (a) integer pointer (b) Pointer to integer

→

5 * This operator gives the value at the address pointed by the pointer.

Main() {

int i = 10;

int *ptr;

ptr = &i;

printf("Value of i: %d\n", i); // 10

printf("Address of i: %u\n", &i); // 500

printf("value of i.d: %d\n", *i); // 10

printf("Value of pte: %d\n", ptr); // 500

printf("Address of pte: %d\n", &ptr); // 600

printf("Value of i: %d\n", *ptr); // 10

* gives the value pointed at that address

printf("Value of pte: %u\n", *ptr); // 500

* pte is replace with the object-(i) at that location

In the above program * pte is replace with object-(i) at that location.

printf("%d\n", *(int *)65494);

→ In 44 operator's the 43 operator's returns a value.

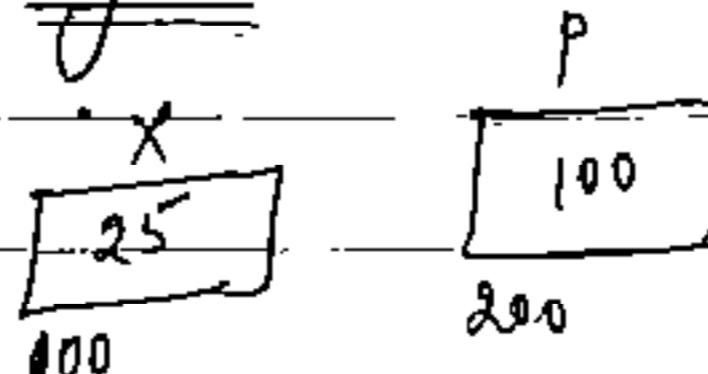
Only * returns object

Main()

int x = 25;

int *p;

p = &x;



printf("Value of x: %d %d\n", x, *p); 25, 25

x = 67;

x
25 67

p
100
200

printf("Value of x: %d %d\n", x, *p); 10, 10 100

*p = 67

printf("Value of x: %d %d\n", x, *p); 67, 67

→ *ptr is acting as an alias to 'x'
alias to the object of the specific location.

Main()

{ int a=100; int b;

a
1000

p1
1000

p2
1000

int *p1, *p2;

p1 = &a;

1000

2000

3000

p2 = &a;

b
50

4000

printf("a: %d p1: %d p2: %d\n", a, p1, p2); 100, 1000, 1000

*p2 = 50;

printf("a: %d p1: %d p2: %d\n", a, *p1, *p2); 50, 1000, 500

b = *p1;

printf("b: %d\n", b); 50

/* b = p2; */ Invalid. P2 contains address 'b' is ordinary variable.
ordinary variables don't store address.

/* p2 = 150; */ Invalid. P2 is pointer variable - don't store values.

} if holds only addresses

→ Main()

{ int i, a=0, b=1, c, *pval, *pval1, *pval2;

a
10
1000

b
1
2000

c
1000
2000
3000

pval1 = &a;

pval1
1000

pval2
2000

pval3
3000

pval2 = &b;

4000

5000

6000

for(i=1; i<6; i++)

{ *pval = *pval1 + *pval2; i = 0+1 2 3 5 8

*pval1 = *pval1;

1 1 2 3 5

*pval2 = *pval2;

2 3 5 8

Levels of pointers

114

int *P; We can place any no. of levels of pointers.

int **S; There is no limitation for the levels of
int ***K; pointers.

Way of reading 'S' is a variable of type ^{integer} pointer to pointer
'K' is a variable of type integer pointer to pointer to pointer

Example:

Main()

{ int a=10; *P, **P2P, ***P2P2P;

P = &a;

(*P) = (*P) + 10;

P2P = &P;

(**P) = (**P) + 5; (**P2P) = *(P2P) (Value at ^{address} P2P) (Value at ^{address} P2P)

P2P2P = &P2P;

printf("%d",

a, *P, **P2P, ***P2P2P); 20, 20, 25, 25

a = 25

&a = 1000

*P2P = 1000

P = 1000

**P2P = 25

&P = 2000

P2P2P = 3000

*P = 25

***P2P2P = 1000

P2P = 2000

*P2P *P = 2000

&P2P = 3000

***P2P2P = 25

a	P	P2P	P2P2P
1000 25	1000	2000 25	3000 4000

i	j	K
10	1044	1040

404 1040 1042

Main()

{ int i=10, *j, **K;

K = &j;

p.f("Y.d|u", i); 10,

p.f("Y.d|u", K, &K, *K, **K);

p.f("Y.d|u", j); 404

p.f("Y.d|u", i, *j, **j);

p.f("Y.d|u", j); 8i;

K, *K);

p.f("Y.d|u", &j); 1040

getch();

p.f("Y.d|u", j); 404

}, 1040, 1042, 404, 1040.

p.f("Y.d|u", *j); 1040

, 10, 10, 10, 10, 10.

Properties of Pointers:-

it is not taking w.r.t. datatype.

1. The size of a pointer variable is independent of the data type w.r.t. to operating system.

```
main()
{
    float * s;
    printf("%d %d: %d %d", sizeof(s), sizeof(*s));
    or *s; // printf("%d %d: %d %d", sizeof(s), sizeof(*s));
}
```

→ A pointer variable at the time of declaration we can initialize with the NULL (a) Zero (b) with any variable

int * p = NULL;

(a)

int * p = 0; (b) int k;

int * s = &k;

→ What is null pointer? whether it is same as uninitialized pointer?

A Null pointer has a reserved value, often but not necessarily the value zero, indicating that it refers to no object. Null pointers are used ~~mostly~~ particularly, particularly in C & C++ where compile-time constant NULL is used. Null pointer means it is not referring any ~~value~~ object but uninitialized pointer can't contain any value (just like uninitialized variable) which we call Garbage.

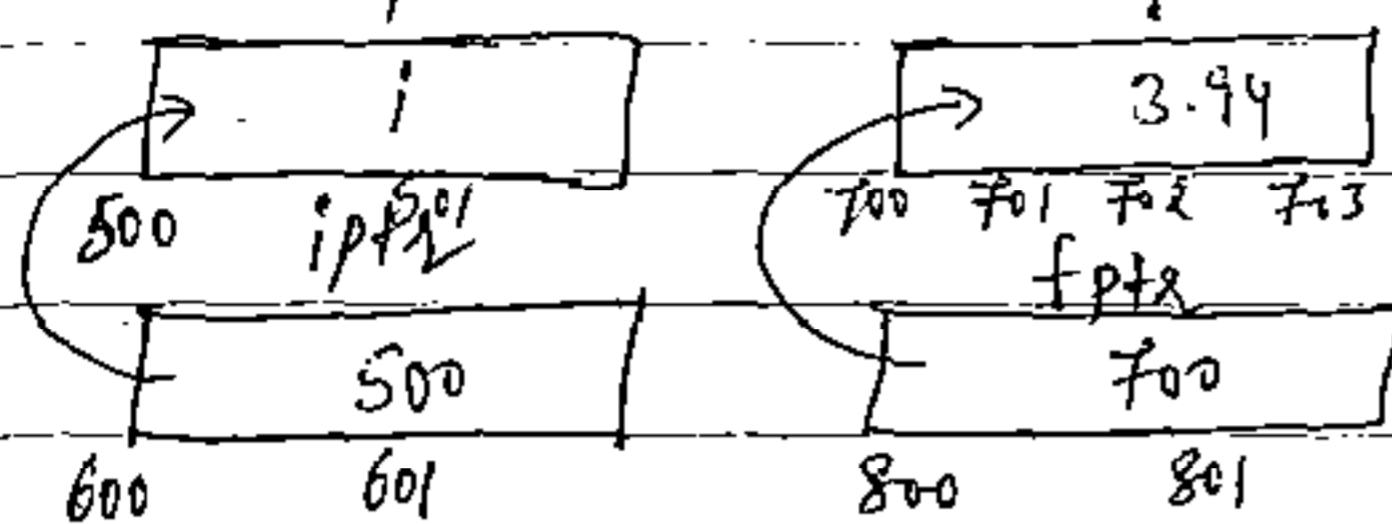
→ An integer pointer can point out an integer variable, if can't point out any other types. it is similar to the other types also.

int i=10; float s=3.94; char c='A'; 115

int *iptr; float *fptr;

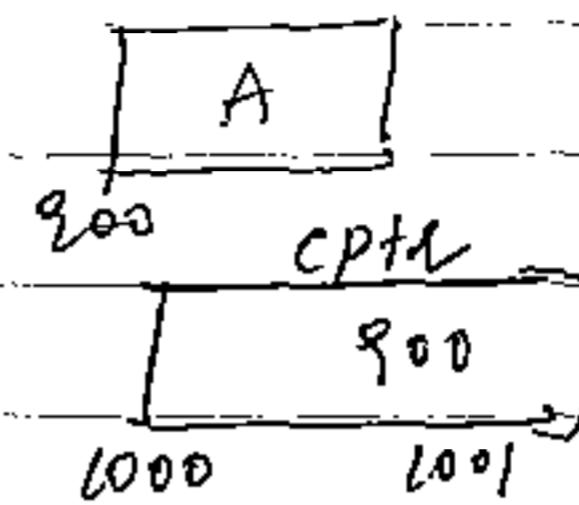
iptr=&i; fptr=&s;

f



cptr=&c;

c



→ Void * (Generic Pointer)

In order to hold any type of addresses we can define Generic pointers. This Generic pointers is a common type of pointer called as void pointers.

Size of pointer is w.r.t to operating system

ex: void *s;

If you want to hold any type of addresses we can define void pointers.

Main()

{ int i=10;

float K=67.78675;

Void *ptr;

Vptr=&i;

printf "%d", *(int *) vptr);

Vptr=&K;

printf "%f", *(float *) vptr);

→ Type Casting

~~Note: But int *p → int *p = 500;~~

Basically if cast be done in either C & C++ in C language doesn't raises an error, Simple a warning will rise i.e NonPortable Conversion. In C++ it raises an Error Can't convert from int to integer pointer (i.e int *)

$\text{int } *p_1 = (\text{int } *) 500;$

This stat is valid in 'C' & C++. Since explicit typecasting is implementing then performing the operations.

Main()

```
{ int a=5;
    int *p_1 = &a;
    int *p_2;
    p_2 = p_1; // Valid
    p_2 = &p_1; // Invalid.
```

→ The stat. $p_2 = \&p_1$ raises a Warning in C language i.e. Suspicious pointer Conversion. In C++ it raises an Error. Since we are assigning double pointer for a single pointer.

p_1	p_2	$*p_1$	$*p_2$
500	600		

We implement arithmetic Operations On pointers if it is called as pointer arithmetic (addresses are adding).

1. Addition of Two pointer Variables is Not Possible

$p_1 + p_2 \rightarrow$ Invalid.

Main()

```
{ int *p_1 = (int *) 500;
    int *p_2 = (int *) 600;
    printf("%u", p_1 + p_2); // Error.
```

getchar(); Invalid pointer addition in function.

→ Any where the 8-bit Microprocessor will not support of the pointer arithmetic of increasing the memory blocks.

→ a pointer & the Constant number addition is possible.

Main()

```
{ int a; // Here "in"
    int *p_1 = &a; // address + Number in float.
    p_1 = (float) 1.2;
    p_1 = (float) 1.2 + 2.3;
    getch();
}
```

→ a pointer & the Constant number addition is possible.

Each Number takes based on its datatype. If $a=2$; it takes 4 bytes. Number is float, #if float $a=2$; it takes 8 bytes.

for adding adding each No. takes it's datatype value.

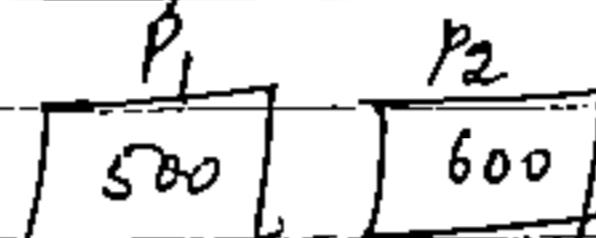
→ Subtraction of pointer & constant number is possible:

$$p_1 + 1 \rightarrow 502$$

$$p_1 - 1 \rightarrow 498$$

In the above struct we are implementing diff's of pointers.

Subtraction of 2 pointers is also possible:



$$\begin{aligned} p_2 - p_1 &= 600 - 500 \\ &= 100/2 = 50 \end{aligned}$$

→ remaining arithmetic operators will not work's with pointers.

→ ~~p1 * p2~~ invalid

→ ~~p1 / p2~~

→ - p1 % p2

→ Relational operators takes place the off based on their addresses what is present.

$p_1 > p_2$ valid

$p_1 < p_2$ valid

$p_1 == p_2$ (valid) Comparing of two memory locations.

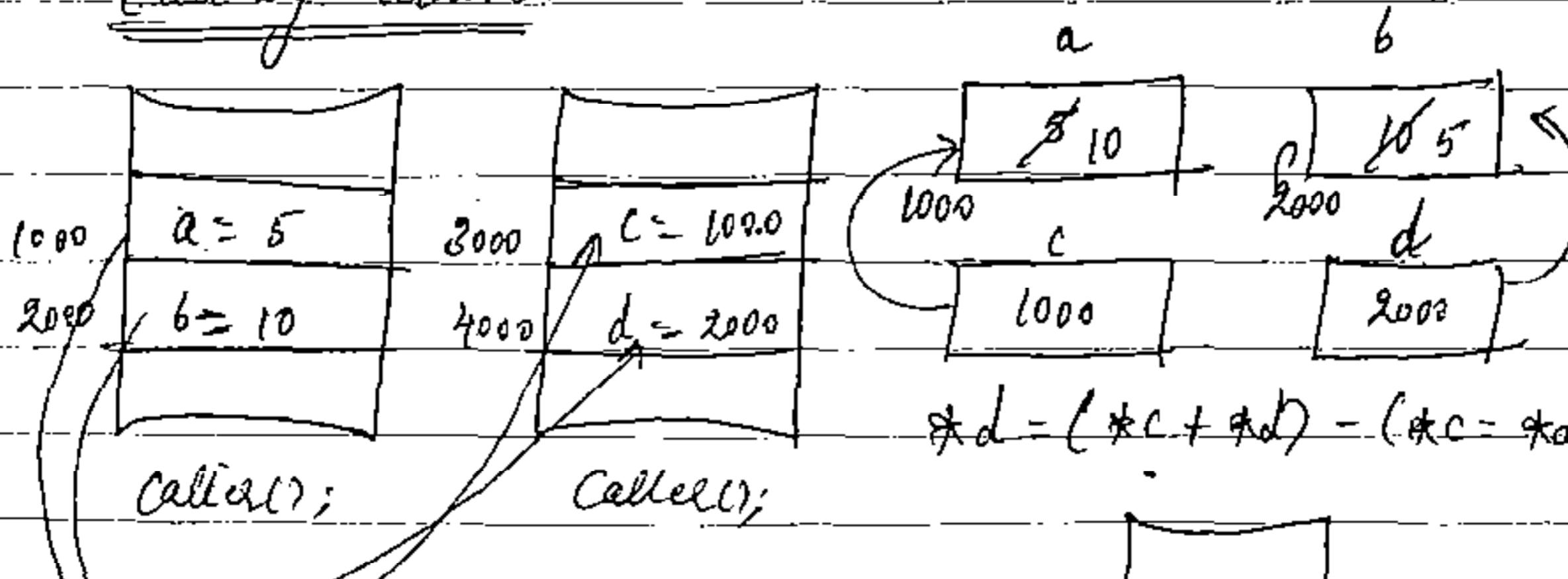
$p_1 != p_2$ (valid)

$p_1 + f;$

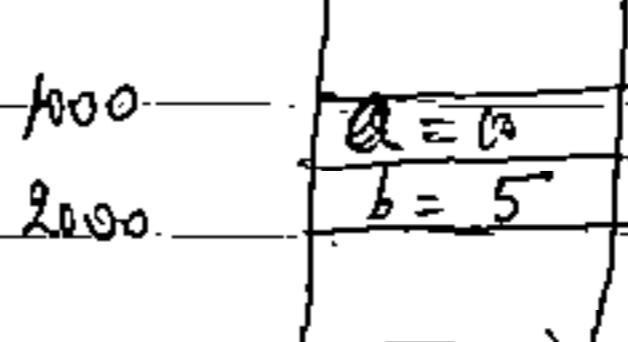
$p_1 - =;$

Passing a pointer variable as a function parameter.

Call by address:-



$$*d = (*c + *d) - (*c - *d) \rightarrow \text{Swapping.}$$



Sending the address of the variables from the caller to
callee function is called as Call-by-address
If any changes made in the callee function it will
affect to the caller function variable values.

Void Swap (int *, int *)

{ int a, b;

p.f("Enter the values:");

S.f("d x.d", &a, &b);

Swap (&a, &b); // Caller.

p.f("After Swapping in Caller(a: x.d b:y.d", a, b);

getch();

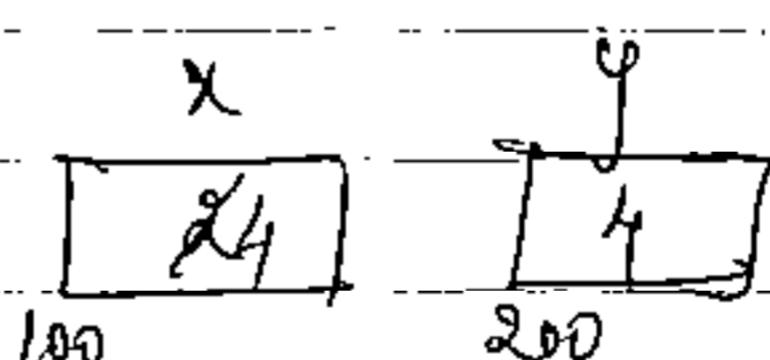
Void Swap (int *c, int *d)

*d = (*c + *d) - (*c = *d);

p.f("After Swapping in swap(*c: x.d *d: y.d", c, d);

→ Void Main()

{ int x = 2, y = 4;



Jump(&x, &y);

p.f("x: x.d y: y.d", x, y);

{ Jump(int *x, int *y)

{ *x = *x + *x;

200 4

= 2 * 2

y = y * y;

16 (Value Not affected in main memory).

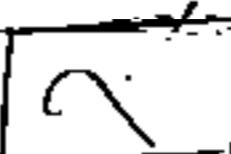
Call-by-value- Not effect the value.

Call-by-address will be effected.

Void Main()

' 11F '

{ long int l;



char *ptr;

clrscr();

l = 0X12345678;

00010010	00110100	01010110	01111000
----------	----------	----------	----------

503

502

501

500

ptr = &l;

+ptr; // This char pointer incre-

+ptr;

moved its address 501 502

printf("%x", *ptr);

by l

ptr

500

Value at 502

0x34

}

Advantages of pointers:-

Main()

{ printf(" Enter the values: "); // Here without declaring variable directly
scanf("%d %d", &i1, &i2); // resolve the address later
printf("%d\n", *(&i1) + *(&i2));
getch();

1. Accepting the data for Neutrino's location is not the proper way,
as our operating system is multitasking if the next location
not free then the problem is started.

2. O.S. itself is a program anything happened security
setting's will be effected.

In windows 3.5 the same program will damage the
System In windows 95 the system will be hang, present
systems abnormally terminated (Core-out-of process)

2. Main()

{ S. f1("%d", ptr);

{ S. f1("%d %d", a, *ptr);

int *ptr;

{ // If ptr. It self contains address No need of

ptr = &a;

{ }

{ S. f1(" Enter the value of a: ");

3. Pointers allows us to pass values to functions using call-by-reference. This is useful when large sized arrays are passed as arguments to functions. A function can return more than one value by using call-by-reference.

4. Dynamic allocation of memory is possible with the help of pointers.

5. We can resize data structures.

For instance, if an array's memory is fixed, it can't be resized. But in case of an array whose memory is created out of malloc can be resized.

6. Pointers points to physical memory and allow quicker access to data.

→ Main()

```
{ int i=20; int *j=&i;
    f1(j);
    *j+=10;
    f2(j);
    printf("%d %d", i, j); }
```

f1(int *k)

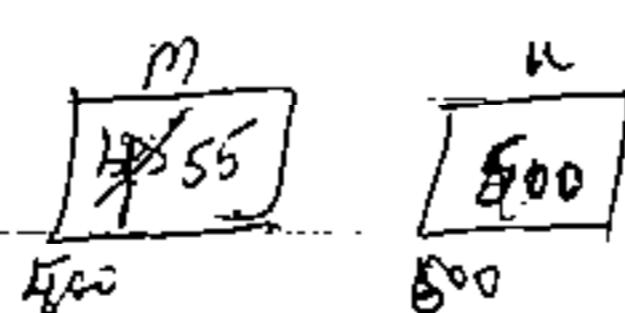
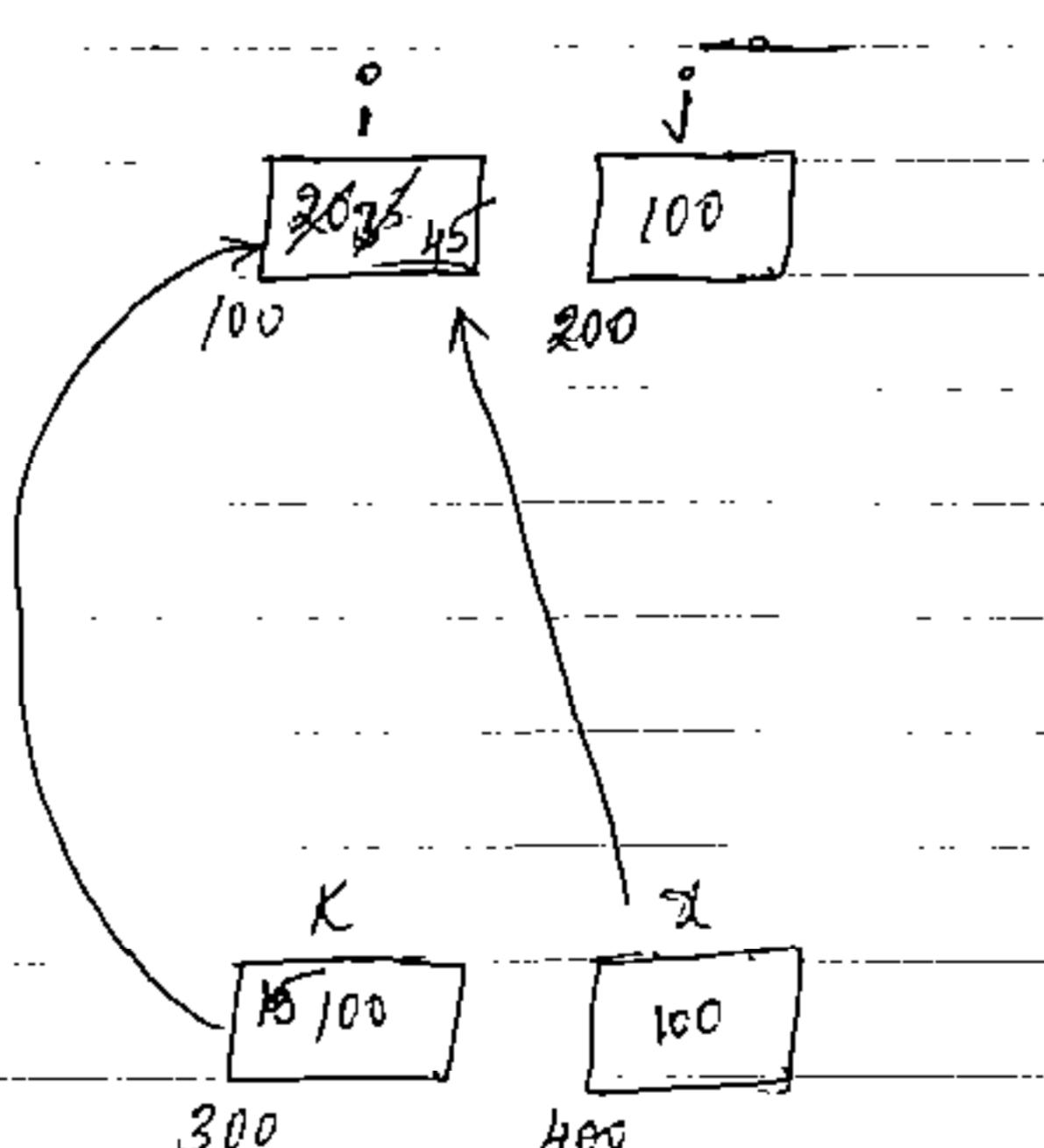
```
{ *k+=15; (K=K+15) }
```

}

f2(int *x)

```
{ int m=*x, n=&m; }
```

```
*n+=10;
```



a &b

void abc (int p₁, int *p₂)

{ ++p₁;

++*p₂;

printf ("n%d%d", p₁, *p₂);

if (p₁ <= 2)

abc(p₁, p₂)

printf ("n%d%d", p₁, p₂);

}

Void Main()

{

int a=1, b=1;

abc(a, &b);

printf ("n%d%d", a, b);

}

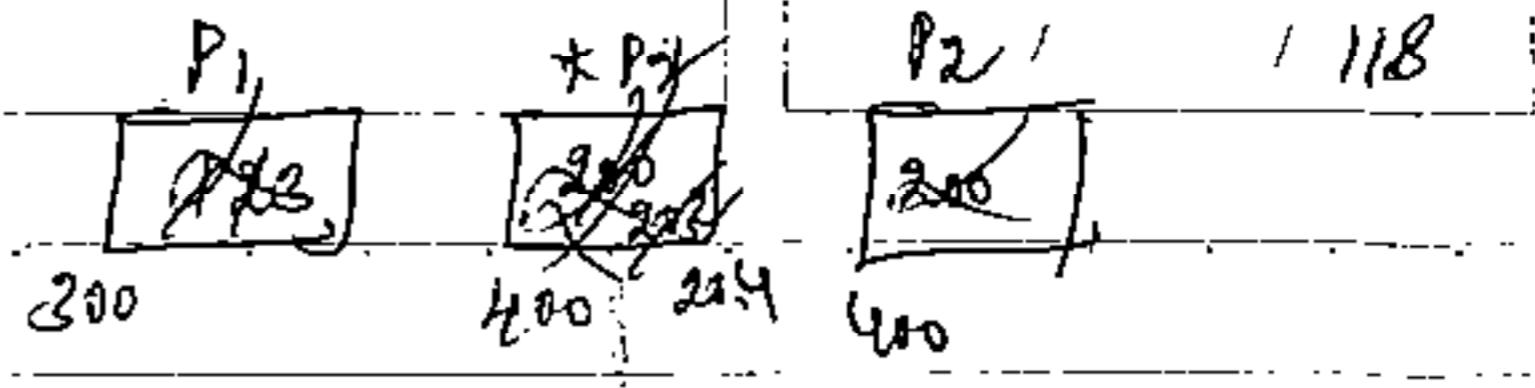
Off 22

33

33

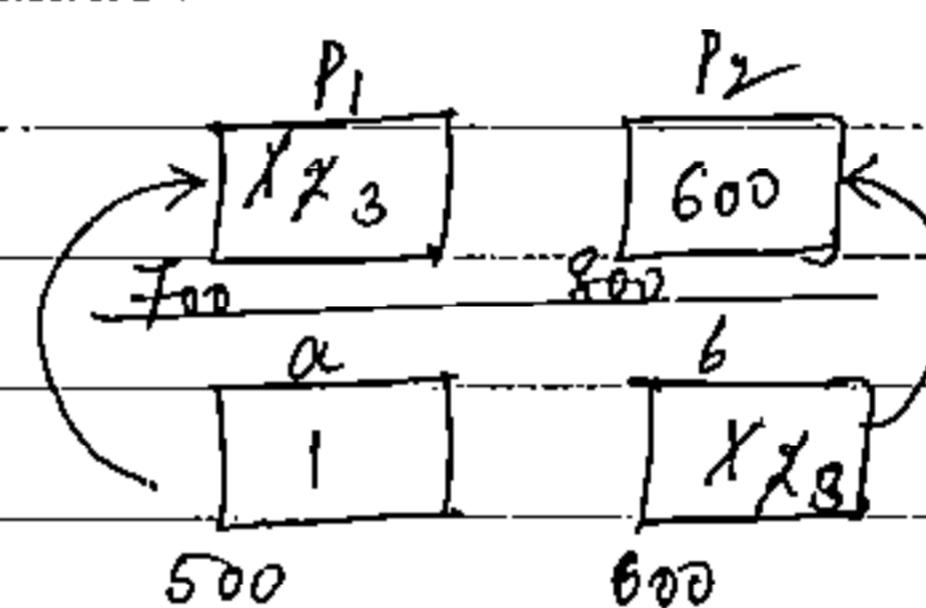
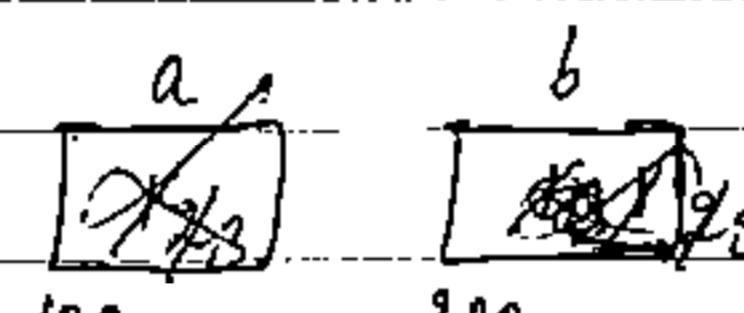
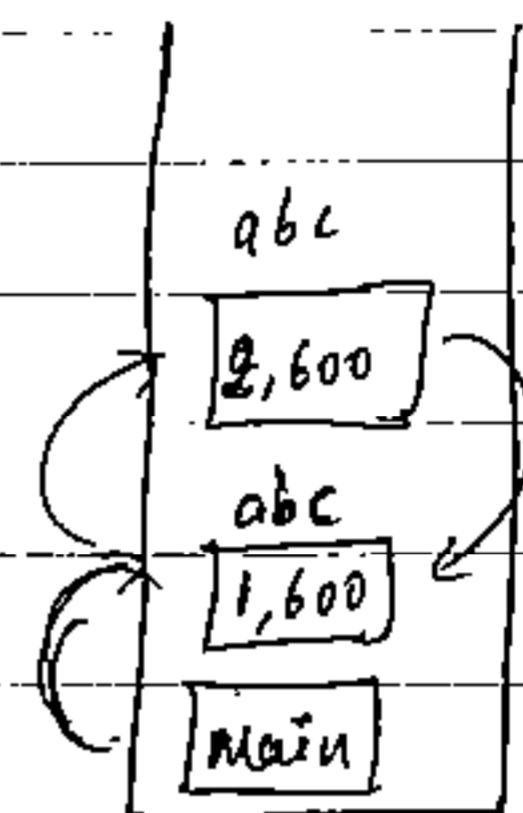
23

18



2, 222, 212
3, 204

3, 204



Pointers to Functions:-

Functions have addresses just like data items.

A pointer to a function can be defined as the address of the code executed when the function is called.

A Function's address is the starting address of the Machine language code of the function stored in the memory.

Pointers to functions are used in writing Memory resident programs writers, viruses, vaccines to remove the viruses.

// Declaration of a pointer to a function.

The declaration of a pointer to functions requires to the function's return type & function argument-list to be specified along with the pointer variable.

Syntax:-

defun-type (* pointer-variable) (function argument-list);

Ex:-

Thus, the declaration int(*fp)(int, int);

declare fp to be a variable of type "pointer to a function" that takes two integer arguments & return an integer as its value.

Example on pointer to a function:

int fun1(int a)

{ return a;

}

float fun2(float b)

{

return b;

}

void main()

{ int (*p)(int); // declaration of pointer to a function

float (*q)(float);

int i = 10;

float s = 56.678;

p = fun1;

q = fun2;

printf("%d %f", p(i), q(s));

getch(); ← p.f function will flow R to L.

→ implementing factorial of a no by using pointer to functions.

#include <stdio.h>

void main(void)

{ unsigned int fact(int); // fun() declaration

unsigned int ft, (*ptr)(int); // pointer to function with prototype of fact(*)

int n;

ptr = fact; // assigning address of fact to ptr

printf("Enter integer whose factorial
is to be found: ");

119

ptx = fact;

ff = scanf("%d", &n);

ft = ptx(n); // call to function fact Using pointer ptx.

p.ft("Factorial of %d is %u", n, ft);

}

unsigned int fact(int m)

{

if (m == 0)

return 1;

else

{ for (i=m; ans=1; i>1; ans *= i-- ans *= i--);
return ans;

}

}

Function defining pointers:-

We have already learnt that a function can return an int, a double or any other datatype.

Similarly it can return a pointer. However, to make a function return a pointer it has to be explicitly mentioned in the calling function as well as in the function declaration.

While defining pointers, define the pointer to global variables or static (as) dynamically allocated address.

Don't return any address of local variables because Stop to exit after function call.

Ex:- int * ptx (void);

declares 'ptx' to be function with no parameters
that returns 'pointer to an int'

```

/* Example of function defining pointer */

int * Kilan();
int * Kiran();

main()
{
    int * ptr;
    ptr = Kilan();
    printf("%d", *ptr); O/P Garbage value
}
    
```

D/F: 25.

Note: As a Non Static variables will be dies when the program body is finished.

Defining Localvariables when the Control is there all the localvariables are created, when the functionality Stop's. Then the Control Came back the life will be dead. Pointing to a dead location is called as 'Dangling pointers'

For this reason define the Variables with 'Static' Keyword

Ex:2: A prog accept 2 No's find greater no.

#include <stdio.h>

Void Main(void)

{ int a, b, *c;

int * check(int *, int *); // Check function takes 2 integers as arguments & returning pointer to an integer.

p.f("Enter Two No.:");

s.f("x1 x2", &a, &b);

c = check(&a, &b);

p.f("In Greater Number is: %d", *c);

}

int * cheer(int *p, int *q)

120

{

if (*p >= *q)

return(p);

else

return(q);

}

/* Here we define static.

because function parameters are by default global. That's why we can't declare 'static' keyword.

Const pointers:

We can define constant pointers just similar like Constant address.

Diff ways are present to define pointers as Constant

const int *p;

(or)

int const *p;

If means 'p' points to integer that is

constant.

Main()

{ int i=10, j=6;

const int *p = &i;

p=f("x.d", *p);

(*p = &j); Valid *

/* *p = 45; error since p is a constant pointer */

}

int * const p; //

it means 'p' is a constant pointer to an integer.

Main()

{ int i=10, j=6;

int * const p = &i;

p=f("x.d", *p); 10

(*p = &j); Invalid since 'p' contains constant address. */

*p = 45;

printf("x.d", *p); 45

→ Main()

{ inf * p;

Const Inf a = 5;

p = &a;

*p = 6;

p.f("x.d", a);

return;

}

>Main()

inf const * p = (inf *) 100;

p++; \rightarrow p points to an integer that

p.f("x.d", p); is constant.

return; // 102

}

\rightarrow inf * const - p = (inf *) 100; // Error.

Arrays

1121

As we know that an ordinary variable can hold only one value at a time. it is not possible to hold more than one value.

To store no. of elements there is no need to declare 'n' no. of variables. We can define an array variable.

An array is an single subscripted variable which can hold, n no. of values.

Syntax:- $\text{data-type Var-name [size];}$ → No. of elements.

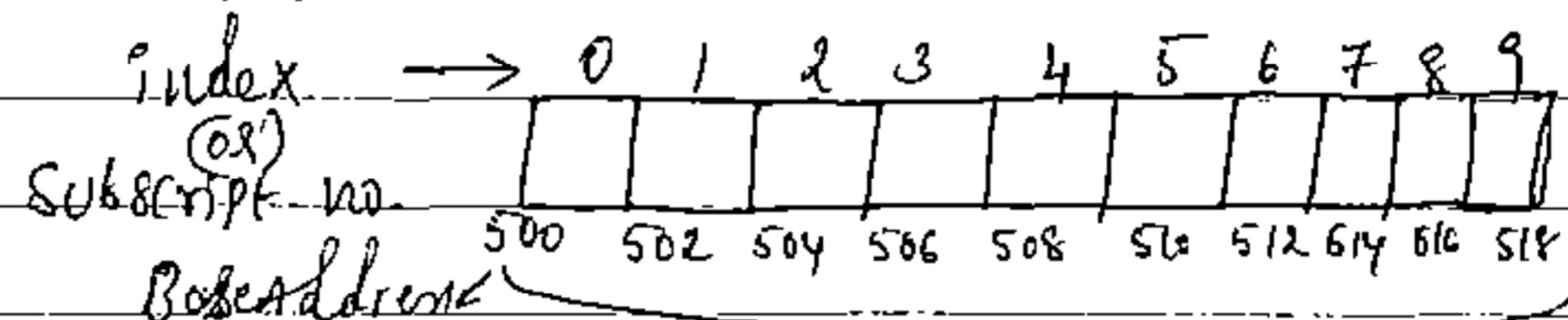
$\text{data-type Var-name [size];}$

dimension

Declaration & initialisation of array variables:

An array variable indirectly or directly the size of the variable must be initialised at the time of declaration only.

Eg:- int arr[10];



→ Once we define the size of the array → Address of each block that many block's are created depend on the size.

Each block is uniquely identified by the Subscript no. User will access the element by the Subscript but internally the compiler will access by the address.

→ Physically only the address is present - there is no subscript no's.

The first subscript will be indicated by 0. That address is called as base address.

In an array all the addresses will be sequential.

An array is called as homogenous type all the elements blocks contains same type of elements.

The Compiler always access by the addresses of each-block.

declaration of array: int- a[5];

float- arr[8];

char name[25]; // we can store one name

int- a[5] = {10, -2, 9, 15, 7};

float s[7] = {45.92, -100.09, 92.78, 68}; // only 4 values are using remaining

char name[25] = {"Kiran"} empty. (08)

char name[25] = {"K", "i", "g", "a", "n"}

→ int- a[1]; invalid (with defining the size of array it is invalid)

→ int- a[-5]; invalid

→ float- arr[5-2]; invalid

→ int- a=10; } Error.

int- arr[a]; constant expression only required

→ const int- a=10; } because variable change its value

int- arr[a]; } possible (valid)

→ int- arr[1] = {10, -12, 9}; // indirectly size is present (valid)

→ #define M 10 }

int- arr[M]; } (valid)

→ int- a[3] = {14, 19, 17, -2}; (invalid)

→ int- a[2000]; (valid)

→ long a[2000]; // Not possible in 16 bit, possible in 32 bit

→ int- arr[0]; // invalid. it can't hold any value.

→ int- a[5];

0	1	2	3	4
Guru	Guru	Guru	Guru	Guru

initially

0	1	2	3	4
10	20	30	0	0

{10, 20, 30}

→ Only at the time of declaration if initialize

An array at least with one element, remaining block's contain zeros.

int a[5] = { }; (invalid)

122

Arrays are classified in 3 categories.

1. Single Dimensional array

2. Double Dimensional array. (2 dimension array)

3. Multi Dimensional array

→ Write a program to accept some elements in an array and display the elements.

Main()

{ int a[3]; Note: This way of implementation of accepting &

a[0] = 10;

displaying is not the proper way.

a[1] = -7

a[2] = 45;

printf("%d %d %d", a[0], a[1], a[2]); 10, -7, 45

}

→ Main()

{ int a[10] = {0}; / out of bounds is (and below is an error

int i, n;

printf("Enter the size of n: ");

scanf("%d", &n); a[i] → element of the array

/* Accepting */

i = 0;

i = indicates index of the array

(or) subscript

while (i < n)

{ printf("Enter the element at a[%d]: ", i);

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

i++;

}

/* displaying */

i = 0;

while (i < n)

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

i++;

getch();

→ Write a prog accept Some elements in an array find the sum of all the elements & display the sum.

Main()

```
{ int a[10];
```

```
int i, sum=0;
```

```
/* Accepting */
```

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

```
{
```

```
printf(" Enter the Element at a[>d]> ");
```

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

```
}
```

```
/* Display */
```

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

```
{ sum = sum + a[i];
```

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

```
{
```

```
getch();
```

```
}
```

Write a prog accept Some elements in an array display the elements in reverse order.

Main()

```
{ int a[10];
```

```
int i, sum=0;
```

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

```
{ printf(" Enter the element at a[>d]> ");
```

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

```
{ if
```

```
for(i=n-1; i>0; i--)
```

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

```
{
```

```
getch();
```

```
}
```

Write a prog accept an array of Elements & accept a number
find out that the how many times it is existed
in an array display the count.

Main()

```
{ int a[7]; // Here 'n' value we taken Runtime also s.f(7.d, &n);
    int i, n, ele, count=0;
    check=0;
    for(i=0; i<n; i++)
    {
        printf(" Enter the value of a[%d]", i);
        scanf("%d", &a[i]);
    }
    printf(" Enter the Element to Check:");
    scanf("%d", &ele);
    /* logic */
    for(i=0; i<n; i++)
    {
        if(a[i]==ele)
        {
            count++;
            check=1;
        }
    }
    if(check==1) // means element is present
        printf(" The %d element is found %d times", ele, count);
    else
        printf(" Element is not found");
}
```

Write a prog accept Some elements on array find out the 2nd maximum
element from the array & display it.

Main()

```
{ int a[5];
    int i, n, fd, sm;
    printf(" Enter the n value:");
    scanf("%d", &n);
```

```

for(i=0; i<n; i++)
{
    printf("Enter the Element at a[%d]", i);
    scanf("%d", &a[i]);
}
fm=a[0];
id=0;
for(i=0; i<n; i++)
{
    if(fm<a[i])           (Another way is arrange the elements
    { fm=a[i];             in decreasing order & return
        id=i;               2 elements.
    }
}
if(id!=0)
    sm=a[id];
else
    sm=a[0];
for(i=0; i<n; i++)
{
    if(fm>a[i] && sm<a[i])
        sm=a[i];
}
printf("Second Maximum value: %d", sm);
getchar();
}

```

Write a program to accept some elements in an array & insert one element
in any ~~position~~^{position} of the array.

```

Main()
{
    int a[10];
    int i, n, ele, pos;
    printf("Enter the value of n: ");
    scanf("%d", &n);
    for(i=0; i<n; i++)
    {
        printf("Enter the element at a[%d]", i);
        scanf("%d", &a[i]);
    }
}

```

0	1	2	3	4	5	6	7	8	9
10	20	30	40	50	60	70	80	90	100
15	20	30	40	50	55	65	75	85	95

1 124 1

printf("Enter the Element to insert: ");

scanf("%d", &ele);

printf("Enter the position of element: ");

scanf("%d", &pos);

pos--;

for(i=n; i>=pos; i--) i=5; 5>=1; i--;

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

a[pos] = ele;

n++;

printf("The elements after insertion: ");

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

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

3

Write a program to delete an element from array.

#include <stdio.h>

#include <conio.h>

Void main()

{ int a[20], n, i, j, de, ck=0;

clrscr();

printf("Enter the no. of elements: ");

scanf("%d", &n);

printf("Enter Array elements: ");

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

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

}

printf("Array Elements before deletion: ");

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

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

}

p-8 Enter Element to be deleted: ");

S-81 Y.d = de)

1* Deletion * /

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

{ if(a[i]==de;

CK=1;

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

{

a[j]=a[j+1];

}

n--;

i--;

{ }

if(CK==0)

printf("Y.d = not found", de);

else

{

p-8 In Array elements after deletion:");

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

{ p-8 If Y.d", a[i]);

{ }

getch();

{ }

⑥

→ Write a program accepts some elements in an array. In that elements
split the even no's in even array & odd no's odd array
& display them. (3 array variable, 1 for accepting all, 1 for even,
1 for odd.)

⑦

→ Write a program accept an array of elements & also accept a lower limit &
Upper limit with in that limit display all the array elements. If it = 20
Upper limit with in that limit display all the array elements. If it = 20

accept some array elements 10 20 30 35 & 40 $\frac{Upper}{Lower} = \frac{40}{20-30,35}$

⑧

accept some array of elements from that elements display
the fibonaccis.

Main()

{ int a[5] = { 5, 1, 15, 20, 25 };

int i, j, k=1, m;

i = ++a[i];

j = a[i]+f;

m = a[i+f];

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

}

0	1	(2)	(3)	4	/ 125
5	12	15	20	25	

2,3
i = ++a[i];

j = a[i]+f;

a[i] = i=2

m = a[i+f] a[2]=15

3, 2, 15, 20

Main()

{ int a[7] = { 10, 20, 30, 40, 50, 30, 20 };

j=32
~~32~~
~~32~~
~~32~~
~~32~~
~~32~~
~~32~~

int i=3, x;

x = 1 * a[i-1] + 2 * a[-i] + 3 * a[-i];

printf("%d ", a); - 60.

}

Write a program accept a no & its base. find out the greatest no. of
the base value.

#include <stdio.h>

<conio.h>

Void main()

{ int j, num, base, a[10], s, i=0;

clrscr();

p. ft " Enter UR number & required base. ";

f. ft "%d %d ", &num, &base);

while(n != 0)

{ s = num % base;

if;

}

i--;

for(j=i; j >= 0; j--)

{ if(a[j] <= 9)

p. ft("%d ", a[j]);

```
else  
if(a[j]==10)  
p.f("A");  
else  
if(a[j]==11)  
p.f("B");  
else  
if(a[j]==12)  
p.f("C");  
else  
if(a[j]==13)  
p.f("D");  
else  
if(a[j]==14)  
p.f("E");  
else  
if(a[j]==15)  
p.f("F");  
getch();
```

6) Ans

```
#include <stdio.h>  
#include <conio.h>  
Void main()  
{  
int a[20], c[20], O[20], i, j, CK=0, n,  
clsclc();  
P.f("Enter the no. of elements: ");  
S.f("%d", &n);  
P.f("Enter array elements: ");  
for(i=0; i<n; i++)  
{ S.f("%d", &a[i]);  
}
```

// Passing Single Dimension array as function parameters

When we passing a Single Dimension array as a function parameter it accept's minimum 2 arguments.
the 1st argument is the array name. 2nd argument is the size of the array.

The first argument can be any type, but the 2nd argument must be integer.

→ Write a program accept's some elements in array & display them by passing as an function parameters.

```
void read(float arr[], int)
```

```
void write(float arr[], int)
```

```
void main()
```

```
{ float arr[10];
```

```
int r;
```

```
clrscr();
```

```
p.f(" Enter the Row size:");
```

```
s.f("%d", &r);
```

```
p.f(" Calling the read function ... ");
```

```
read(arr, r);
```

```
p.f(" Calling the write function ... ");
```

```
write(arr, r);
```

```
getch();
```

```
}
```

```
void read(float arr[], int r)
```

```
{ int i;
```

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

```
{
```

```
s.f("%f", &arr[i]);
```

```
}
```

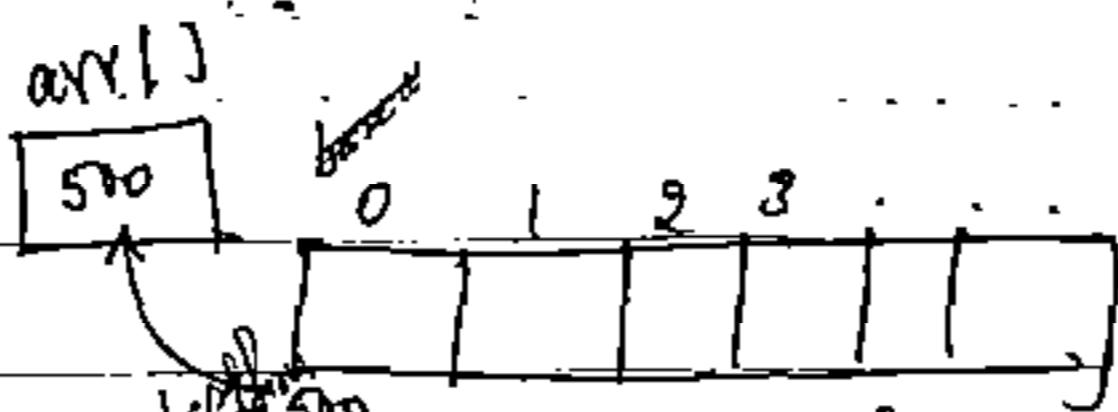
```
}
```

Optional
size optional

```
void write(float arr[], int r) // write(&arr[0], r);
```

```
{ int i;
```

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



127

When array variable is passing as a function parameter by default if passes base address.

~~When an array is passing an address the concept is called~~
Call-by-address.

→ In this definition the array address lecoring by Subscript operator, (Not a pointer).

→ The subscript also lecoring address instead of Subscript (i) if a i use pointer simbole, it also works.

Since, actually the pointer only lecere the address

→ So we can replace the definition with pointer Simbole.

Void leak (float *a, int)

{

}

→ As the Subscript is replaced with pointer, we can say that array is a pointer.

Each Subscript will be replaced by pointer.

[] = *

[] [] = **

[] [] [] = ***

If the user explicitly not replace also, internally the Compiler will replace by the pointer Simbols.

Why Can't we say array is a pointer?

YES, but the difference is

a = 25

(Integer variable) (Integer Constant)

a+f; // Valid

25+f; // invalid 25 is constant. w.

Similarly arr- is a Constant-implicit pointer

p1- is integer pointer

Note: Array name always gives the base address of the array (arr or arr[0]) both are same.
It is a constant pointer.

→ So finally every subscript internally replace by the pointer symbol. We are working with subscript, that subscript replaced by pointer.

Void Main()

{ int arr[5] = {3, 13, 23, 33, 43};

+arr; // Invalid.

-- *arr;

-- *arr;

printf("%d", *arr); Error.

0	1	2	3	4
3	13	23	33	43
500	502	504	506	508

array is constant pointer

We can't increment it.

}

How to read a pointer notation with increment, Decrement operators.

$++ptl$ → pre increment pointer (address)

$ptl++$ → post increment pointer

$--ptl$ → pre Decrement pointer

$ptl--$ → post Decrement pointer

→ ptl is a pointer, * address. ($*ptl$) → value at that Address

$++*ptl$ → Pre increment of object

$--*ptl$ → Pre Decrement of object

$(*ptl)++$ → Post increment of object

$(*ptl)--$ → Post decrement of object

$*+ptl$ → Pre incrementation of pointer & accessing the data.

$*ptl+$ → Accessing the data & Post-incrementation of pointer.

$*--ptl$ → Pre decrementation of a pointer & accessing the data.

$*ptl--$ → Accessing the data & post-decrementation of a pointer.

→ Main()

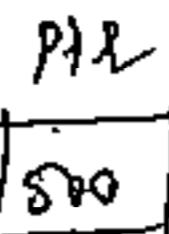
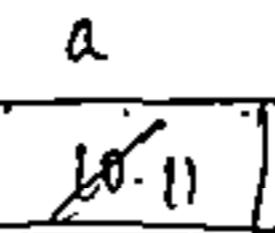
{ int a=10;

int *ptr;

ptr = &a;

(*ptr)++; // post-increment of object / ++(*ptr);

printf("%d %d %d", a, *a, *ptr); 11, 11, 11.



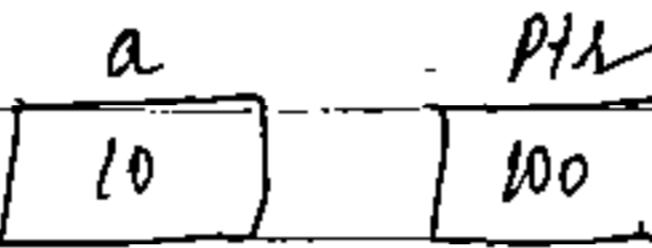
500

600

O/P 11, 11, 11.

→ Main()

{ int a=10;



int *ptr;

500

200

ptr = &a;

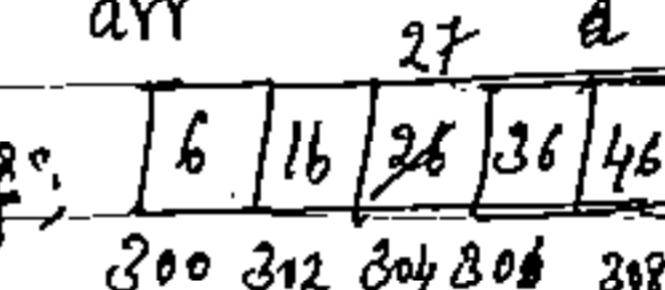
*ptr++; // Accessing the data & post-incrementation of pointer

printf("%d %d %d", a, *a, *ptr); 10, 10, G.V.

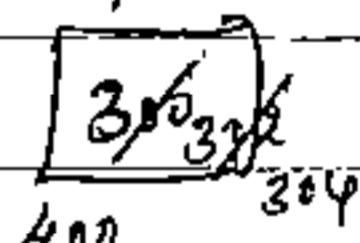
→ Void Main()

{ int arr[5] = {6, 16, 26, 36, 46};

arr



ptr



int *ptr;

300

400

304

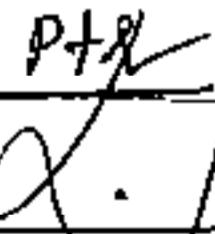
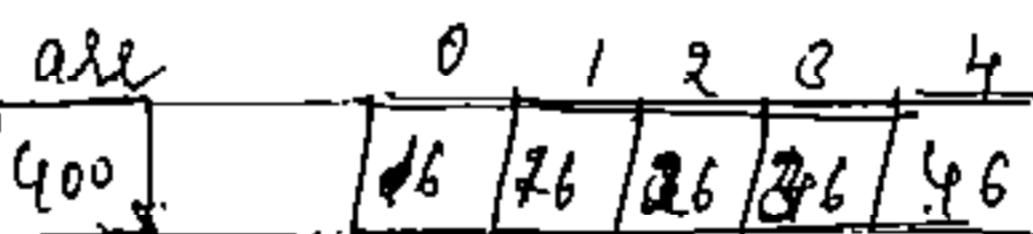
++ptr; // pre incrementation of pointer

++ptr; //

+*ptr; // pre incrementation of object

printf("y.d %d y.d %d", *ptr);

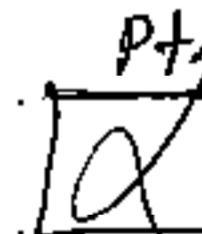
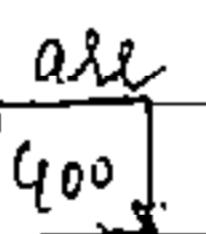
arr



}

→ Void Main()

{ int arr[5] = {6, 16, 26, 36, 46};



printf("y.d %d y.d %d", arr[2], *arr+2, *arr+2+arr);

}

26

26

26

26

Externally

Internally

arr[2] → *(arr+2)

*(&arr+2)

*(&arr+2) = 26

→ *(2+arr)

*(&arr+4)

*(&arr+4) = 26

Note: When we construct an array, where continuous space is deserved, when a constant pointer is generated with the name "arr", or a base address with the index operator comfortable we can locate all locations.

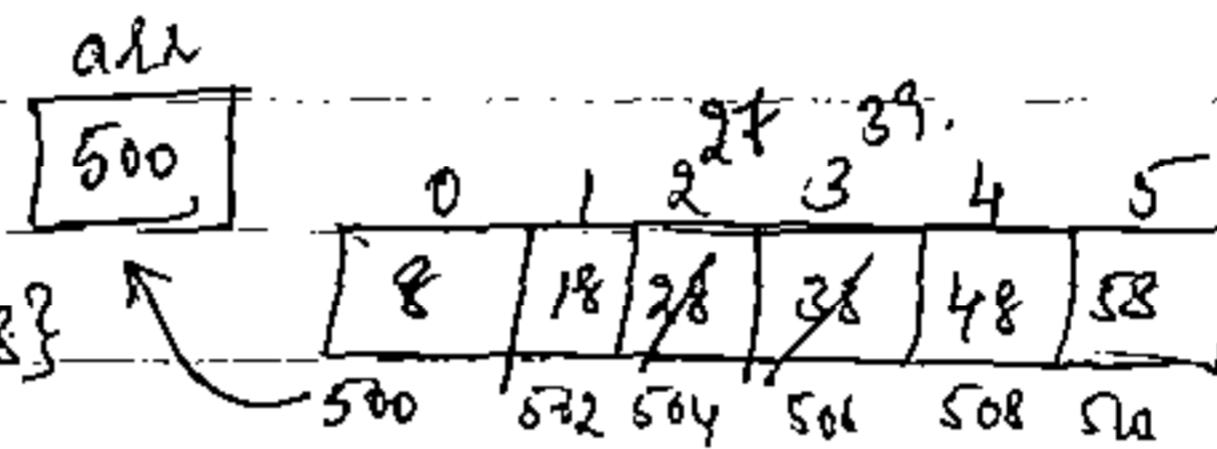
Why the index starting with zero (0) :-

because $\text{arr}[0] \rightarrow *[\text{arr}+0]$

$\text{arr}[1] \rightarrow *[\text{arr}+1]$

this is result array starts with zero, physically first location.

```
void main()
{
    int arr[7] = {8, 18, 28, 38, 48, 58};
    int *ptr;
    ptr = arr;
}
```



```
+4 ptr; ptr = arr + 1;
```

```
-- *ptr; // predecrement of object -
```

```
+4 ptr;
```

```
++ *ptr; // preincrement of object - (preincrement of value at that address)
```

```
printf("%d %d %d", ptr[-1], arr[0], ptr[1]); // arr[1] = *(ptr+1)
```

```
{ 27, 39, 48 } ← R to L      ptr[0]      *(506+1)
```

```
ptr[-1]      *(ptr+0)      *(508)
```

```
* (ptr-1)      *(506)      48
```

```
* (506-1)      39
```

```
* (504)
```

```
27
```

Main()

```
static int arr[5] = {2, 4, 6, 8, 10};
```

```
int i, b = 5;
```

```
for (i = 0; i < 5; i++)
```

```
{
```

```
    if (arr[i] == 6); // caller
```

```
    printf("%d %d", arr[i], b); // printf(4, 100/6, 400/8, 400/4, 400/2)
```

```
}
```

$f(x,y)$ // callee

int $x, *y;$

{ $x = *y + 2;$

}

$$x = (\ast y) + 2$$

$$x = b + 2$$

$$5 + 2 = 7$$

$$x = 7$$

Main()

{ int a[5] = {9, 88, 54, 23, 78};

int *p;

$p = a + 2;$

printf("%d %d %d", ~~*p~~, ~~*p + 1~~, ~~*p + 2~~); 23, 54, 54, 24

}

\uparrow \leftarrow R. to L
++ *p : pre increment - of (value) object - (at that address)

*p++ : Accessing the data is post-increment

*p + 1 : pre increment - & Accessing - data

[*** printf('') work flow always Right to left.]

Main()

{ int a[5] = {6, 18, 26};

++ *a;

++ *{a + 1};

++ a[1];

printf("%d %d %d %d", a[0], a[1], a[2], a[3]); 7, 18, 26, Q0

}

array contains only 0 value. remain value 0's after

a

600

0 1 2 3 4 129

2 4 6 8 10

600 602 604 606 608

b

f()

x

y

400

700

800

900

1000

1100

1200

1300

1400

1500

1600

1700

1800

1900

2000

2100

2200

2300

2400

2500

2600

2700

2800

2900

3000

3100

3200

3300

3400

3500

3600

3700

3800

3900

4000

4100

4200

4300

4400

4500

4600

4700

4800

4900

5000

5100

5200

5300

5400

5500

5600

5700

5800

5900

6000

6100

6200

6300

6400

6500

6600

6700

6800

6900

7000

7100

7200

7300

7400

7500

7600

7700

7800

7900

8000

8100

8200

8300

8400

8500

8600

8700

8800

8900

9000

9100

9200

9300

9400

9500

9600

9700

9800

9900

10000

10100

10200

10300

10400

10500

10600

10700

10800

10900

11000

11100

11200

11300

11400

11500

11600

11700

11800

11900

12000

12100

12200

12300

12400

12500

12600

12700

12800

12900

13000

13100

13200

13300

13400

13500

13600

13700

13800

13900

14000

14100

14200

14300

14400

14500

14600

14700

14800

14900

15000

15100

15200

15300

15400

15500

15600

15700

15800

15900

16000

16100

16200

16300

16400

16500

→ Main()

{ int a[5] = { 11, 22, 33, 44, 55 };

int b[5] = { 66, 77 };

b = a; Error

printf("%d", b[1]);

}

a

100

0 1 2 3 4

66 77 0 0 0

100 102 104 106 108

200

6

200

66

77

0

0

0

0 1 2 3 4

200 202 204 206 208

Ques

→ Array is Constant - Implicit pointer €

Array is Constant - implicit pointer (we can't change the

Address of ~~array~~ Array).
it is constant

Main()

{ int a[2];

int i;

for (i=0; i<10; i++)

a[i] = i*i;

printf("%d %d", a[3], a[4]); 9, 16, abnormal termination.

}

a

100

↑

0

1

0

1

100

102

Drawbacks of arrays:

1. In the above program the array has only 2 block's, but the loop is satisfying in 'i & i+1'; it doesn't raises any error, since there is no 'bounds checking'.

2. As an array is homogenous it can accept only similar type of elements.

3. As an array is a static memory allocation, the size of the array must be defined before execution of the program.

4. Due to size has been defined before execution there is a wastage of Memory blocks.

Array of pointers:

130

int *A[5]; → This array of pointer variable can hold
5 block's of addresses.

→ void Main()

{ int A[] = { 4, 14, 24, 34 };

int B[] = { 6, 16, 26, 36 };

int C[] = { 9, 19, 29, 39 };

int *ptr[3];

int *pptr;

int i;

ptr[0] = A;

ptr[1] = B;

ptr[2] = C;

pptr = ptr;

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

{ pptr + = i;

*pptr + = i;

+f pptr;

}

- = pptr;

p.f("In %d", *pptr);

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

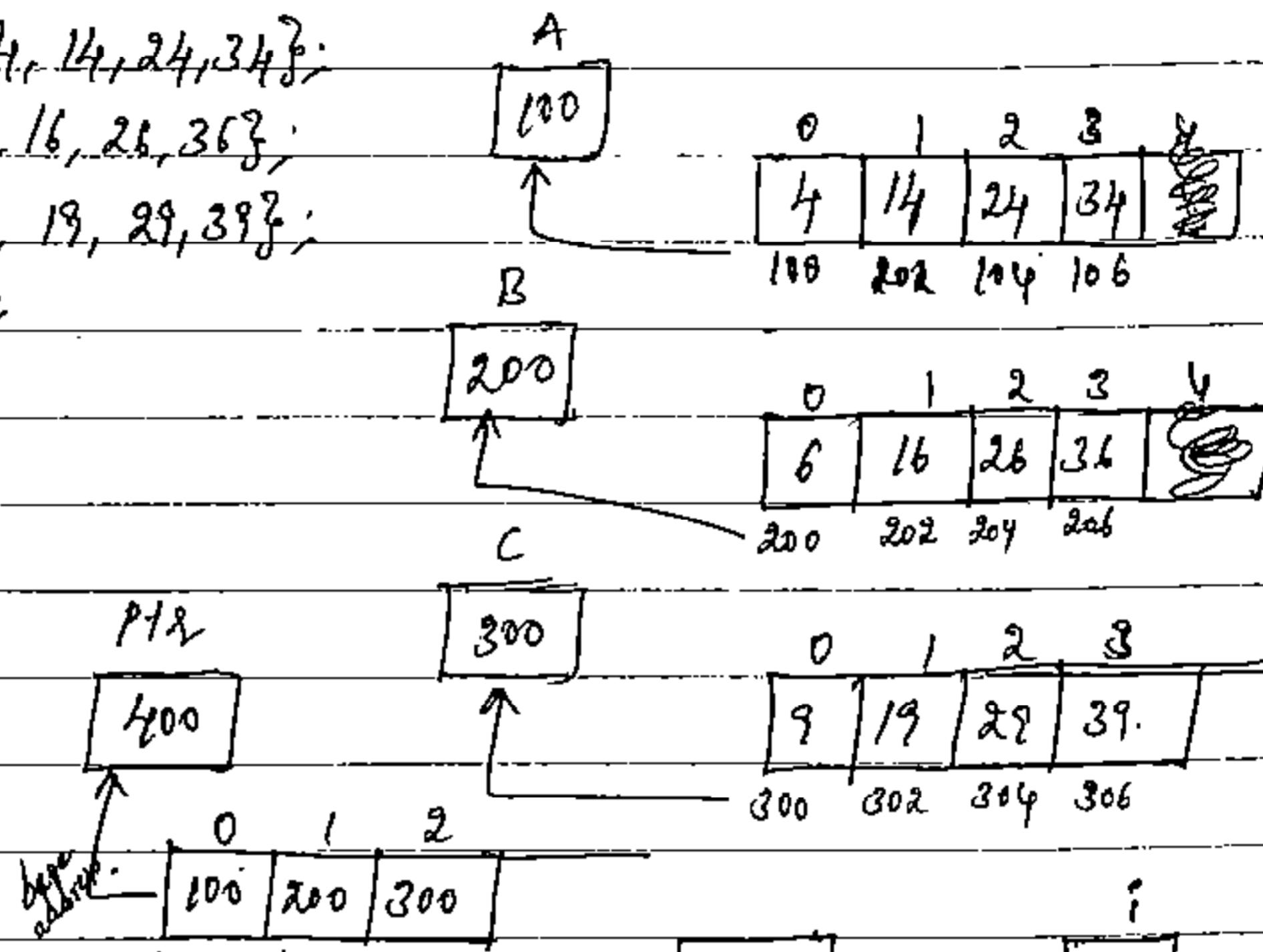
{ p.f("%d", *ptr[i]);

+f (i=0; i<3; i++)

p.f("In %d %d %d", A[i], B[i], C[i]);

}

}



Pointed to an array :-

// program to understand diff b/w pointer to an array &
pointer to an array of integer

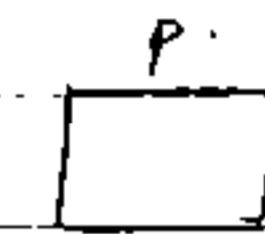
Main()

```

{ int *p;
int (*ptr)[5]; // pointer to an array (pointer to an integer)
int arr[5]; // array of integers.
p = arr; /* points to 0th element of arr */
ptr = arr; /* points to whole array arr */
printf("p=%u, ptr=%u\n", p, ptr);
p++;

```

65516 65516

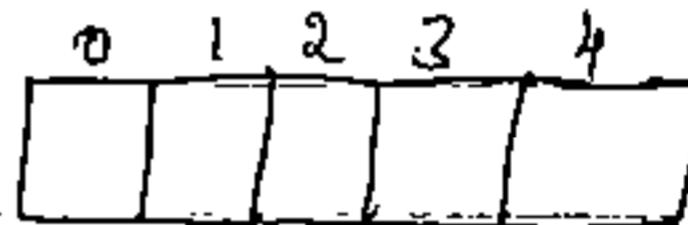


```

printf("p=%u, ptr=%u\n", p, ptr);
}

```

65518, 65526



ptr = arr;

→ ptr has an array variable, So it store total array of arr.

Double dimensional array:

More than one dimension, generally we can call double(08) multidimension.

Any type of dimension internally the data will be on rowbased programming.

'C' is a Rowbased programming

'pascal' is column based programming

```

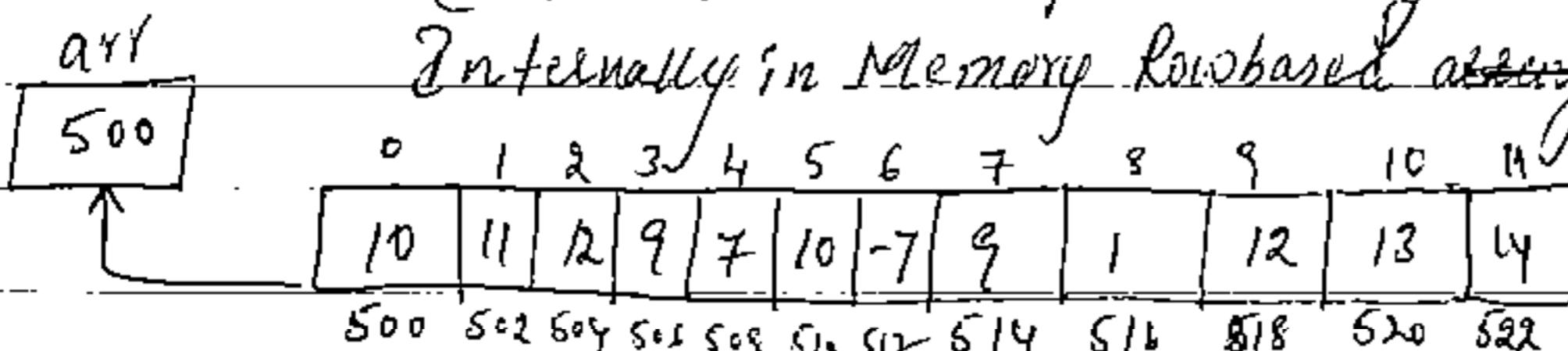
int arr[4][3] = { {10, 11, 12}, {9, 7, 10}, {-7, 9, 13}, {12, 13, 14} };

```

↑
optional

(Leftmost dimension optional when you initialize)

Internally in Memory Rowbased arrangement's programming.



The size of a double dimensional array

datatype * ColmnSize * RowSize

`int arr[3][3] = {`

`{ 10, 11, 12 }, → optional (braces)
 { } → Mandatory ()
 { 92, -5, 27 }`

`};`

(08)

`int arr[3][3] = { 11, 12, 15, -7, 2, 1, 4 };`

3×3

depends on the column size, based on elements decide the no. of rows (In above ex: 2 ten element's are empty).

How the double Dimensional array elements arranged in Memory.

Ordinary Variable.

Main() `a`
`int a=5; [5]`

`p.f("%d", a); // 5`

`p.f("%d", &a); // 1000`

`p.f("%d", *a); Error`

Since pointer variable

always play the role

with address variable only

→ Since pointer variable always

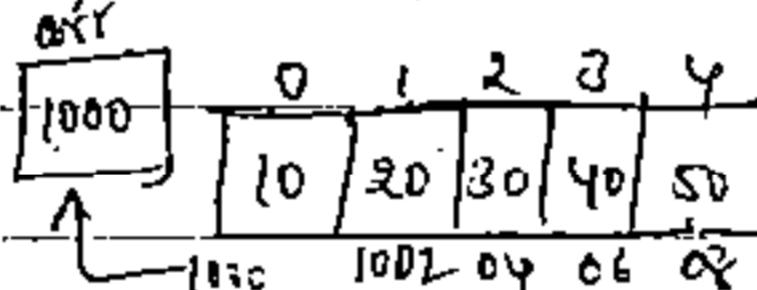
play the role with address

variable only.

Single Dimensional array.

Main()
`{ int arr = {10, 20, 30, 40, 50} };`

`p.f("%d", arr);`



`p.f("%d", &arr); // 1000`

`p.f("%d", *arr); // 10`

`p.f("%d", &arr[0]); 1000`

`p.f("%d", arr[0]); 10`

Double Dimensional array

Main()
`{ int arr[3][3] = { {10, 20, 30}, {40, 50, 60} } };`

1000	500	600	502
1001	514	614	516
1002	504	604	506
1003	508	608	510

Main Row Column's
 Memory Dimension.

`p.f("%d", arr); 500`

`p.f("%d", &arr); 500`

`p.f("%d", *arr); 500`

`p.f("%d", **arr); 10`

An array of arrays. So here 'arr' is an array of 3 elements where each element is a 1-D array of 4 integers.

We know that the name of the array is a "constant pointer" that points to the 0th element of the array. In case of 2-D arrays, 0th element is a 1-D array so the name of 2-D array represents a pointer to a 1-D array.

For example : In the above case, 'array' is a pointer to 0th One-Dimensional array contains address 404, since array is a "POINTER TO AN ARRAY OF 4 INTEGERS". So according to pointer arithmetic, the expression (array) will represent address 412

→ Void Main()

```

? static int a[3][4] = {{1, 2, 3, 4}, {5, 6, 7, 8}, {9, 10, 11, 12}};
printf("%d\n", a); 500
printf("%d\n", *a); 500
printf("%d %d %d\n", a[0], a[1], a[2]); 500, 508, 516
printf("%d %d %d\n", *(a+0), *(a+1), *(a+2)); 500, 508, 516
printf("%d %d %d\n", a[0], a[1], a[2]); 500, 508, 516
printf("%d %d %d\n", a[0]+1, a[1]+2, a[2]+3); 502, 510, 522
printf("%d %d %d\n", *(a[0]+1), *(a[1]+2), *(a[2]+3)); 502, 510, 522
printf("%d %d %d\n", a[0][1], a[1][2], a[2][3]);

```

}

$$a[0] = *(a+0)$$

$$a[1] = *(a+1)$$

$$(1000+1)$$

$$*(1002)$$

608

a	500
0	500
1	508
2	516

500	502	504	506
1	2	3	4
5	6	7	8
10	12	14	16

→ Void Main()

{ Static int arr[3][3] = {{ {1, 11, 21}, {2, 12, 22}, {3, 13, 23} } };

int *ptr[3];

int **pptr;

clrscr();

ptr[0] = &arr[0][0];

ptr[1] = &arr[1][0];

ptr[2] = &arr[2][0];

pptr = ptr;

+f ptr[0];

+f *pptr;

+f *ptr[0]; 21

+f **pptr;

+f pptr;

++ptr[1]; 608

+f *pptr; 610

+f *ptr[1];

-- *pptr;

+f pptr;

+f ptr[2];

-- *ptr;

-- *ptr[2];

+f **pptr;

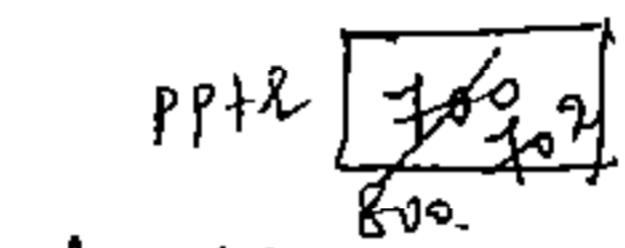
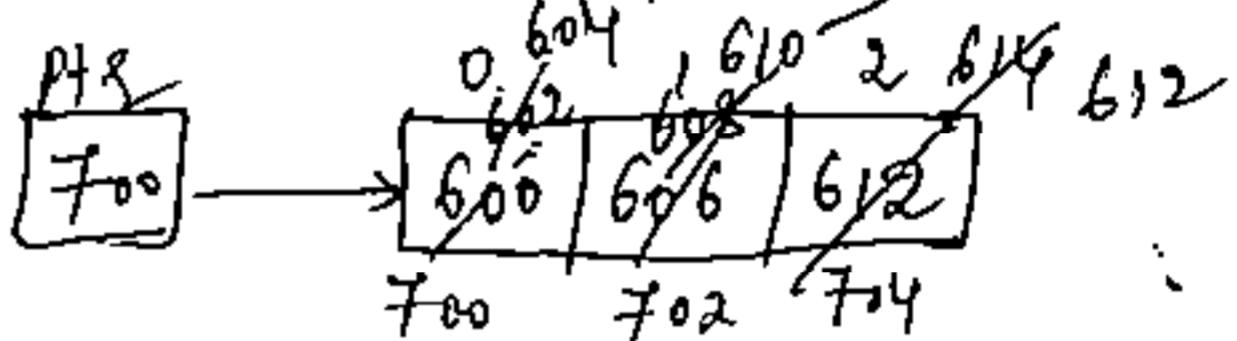
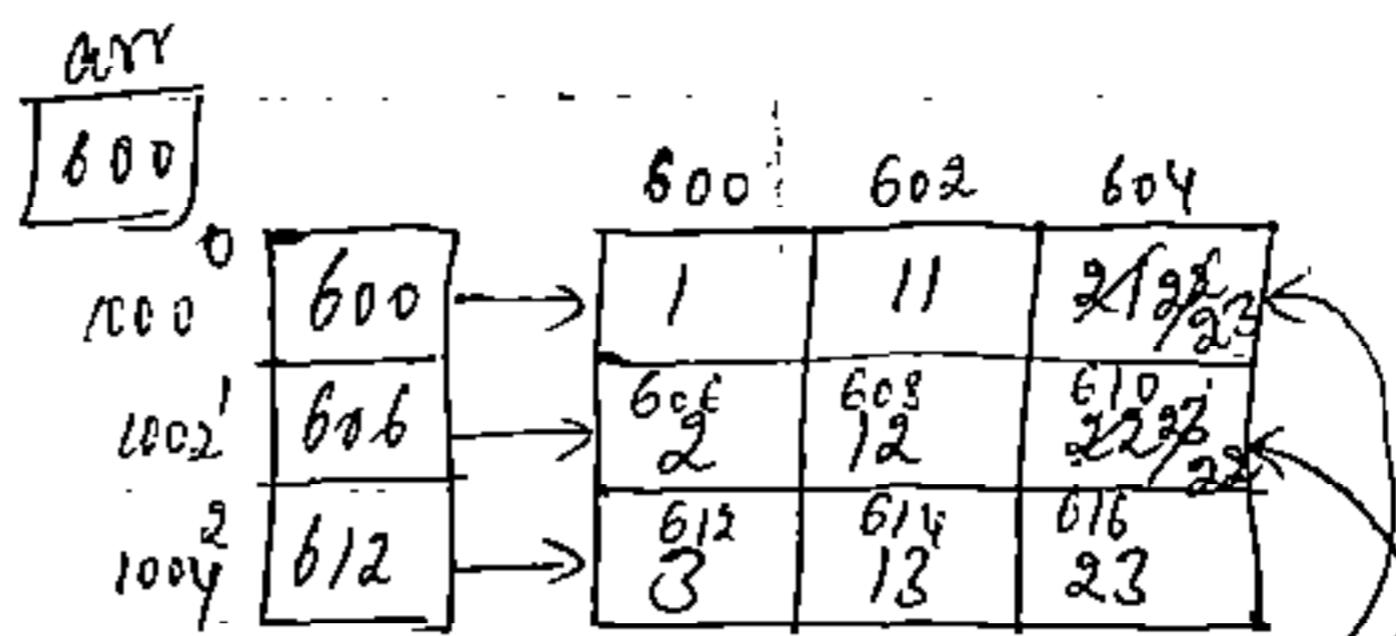
printf("u %d %d %d", arr[0][2], arr[1][2], arr[2][0]); 23, 22, 3

printf("u %d %d %d", *(*(arr+1)+2), *(*(arr+2)+0), *(*(arr+0)+2));

printf("u %d %d %d", *ptr[0], *ptr[1], *ptr[2]); 22, 12, 3

printf("u %d %d %d", **(ptr+0), **(ptr+1), **(ptr+2));

9



*(ptr+0)

Void Main()

{ Shall num[3][2] = {3, 6, 9, 12, 15, 18};

printf("%d %d", *(num+1), *(num+2));

15 15

num

500	502
504	506
508	510

500 502

504 506

508 510

*(num+2)

*(num+1)[1]

loop 2

(1000+2)

(1000+1)

*(1002)

*(1002)[1]

*508 = 15

((1002)+1)

*(1004)

*508 = 15

⇒ Void Main()

{ long arr[2][4] = {0L, 1L, 2L, 3L, 4L, 5L, 6L, 7L};

printf("%d %d", arr[1][2]), 6L;

printf("%d %d %d", *arr[1]+3, 3[arr[1]]); 7L

printf("%d %d %d %d", *(arr+1)+2), *(3[arr]+2), 3[1][arr]);

}

arr[1][2] = *(*(arr+1)+2)

(1000+1) 1000 600

*(1002) 1002 608

(608+2)

600 602 604 606

0L 1L 2L 3L

4L 5L 6L 7L

608 610 612 614

*(612) = 6L

*arr[1]+3

→ #include <stdio.h>

((arr+1)+3)

Void main()

11a+1

(1000+1)

{ long double a; 10

0+1 = 1

*(1002)

signed char b; 1,

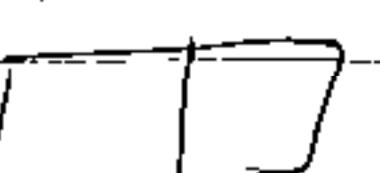
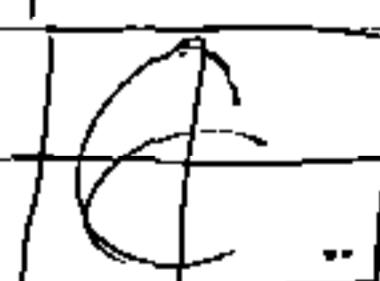
*(608+3)

int arr[Sizeof(10a+b)];

*614 = 7L

printf("%d", sizeof(arr)); 4

}



$A[2][3] \rightarrow *(*a + 3) \rightarrow *(*(*a + *(*a + 3))$

Array

Subscript

Pointer

1D	$a[i]$	$*(*a + i)$
2D	$a[i][j]$	$*(*(*a + i) + j)$
3D	$a[i][j][k]$	$*(*(*(*a + i) + j) + k)$
4D	$a[i][j][k][l]$	$*(*(*(*(*a + i) + j) + k) + l)$

→ Always prefer pointers they will run very fast.

→ Array is very to implement.

Write a prog accept some elements in array & display them.

Manual

```
{ int a[3][3];
int i, j;
for (i=0; i<3; i++)
{
    for (j=0; j<3; j++)
        printf("Enter the elements at a[%d][%d]: ", i, j);
    scanf("%d", &a[i]);
}
```

for (i=0; i<3; i++)
{

for (j=0; j<3; j++)
 printf("%d", a[i][j]);
}

O/P: Enter the Element at a[0][0]= 10

getch();

}

10 20 30

40 50 60

70 80 90

a[0][0]= 10

a[0][1]= 20

a[0][2]= 30

a[1][0]= 40

a[1][1]= 50

a[1][2]= 60

a[2][0]= 70

a[2][1]= 80

a[2][2]= 90

→ 28 is stored in the position of 27 because it starts from '0' position.

5 So. 28 place is $28 - 1 = 27$ place $\frac{8}{\text{Column}} \frac{27}{24} [3]$ all $[3] [3]$:

→ int arr[7][8]; find the positions of array values. 134

1 → arr[0][0]

28 → arr[1][1] Not using array representation table we can

16 → arr[1][7] easily find out it-

45 → arr[5][4] 28. position will divided by Column

36 → arr[4][3] Column = 8 Row = 7

52 → arr[6][3] $\frac{8}{\text{Row}} \frac{16}{8} [1] \rightarrow \text{Row}$ $\frac{8}{\text{Column}} \frac{45}{5} [5]$
 $\frac{8}{8}$ $\frac{40}{5}$
7 → Column

→ Main()

{ int arr[5][8]; one row one column
total array \uparrow \uparrow

printf(" %d %d %d ", sizeof(arr) sizeof(arr[0]) sizeof(arr[0][0]));

80 16 2

$5 \times 8 \times 2 \rightarrow$ datatype contains 8 columns.

$8 \times 2 \rightarrow$ single column
 2×4

→ Main()

{ int arr[4][8] = {5, 55, 555};

printf("%m %d", &arr[3][4] - &arr[1][2]); 18

}

S. cut: $8 \times 3 + 4 = 24 + 4 = 28$

$8 \times 1 + 2 = 8 + 2 = 10$

18

→ Write a program to implement the Matrix addition by passing as a function parameter.

#include <stdio.h>

<Conio.h>

Void main()

{ Void accept(int a[2][2], int, int);

Void display(int l[2][2], int, int);

Void addMatrix(int l[2][2], int m[2][2], int c[2][2], int, int);

int a[2][2], b[2][2], c[2][2], i1, i2, j1, j2, C1, C2;

```
clrscr();
```

```
printf("Enter the size of Row & columns of First Matrix:");
```

```
printf("Enter the size of Row & columns of Second Matrix:");
```

```
scanf("%d %d", &r1, &c1);
```

```
scanf("%d %d", &r2, &c2);
```

```
if(r1 == r2 & c1 == c2)
```

```
{
```

```
p-f1 "Enter the 1st matrix elements [n^2];
```

```
accept(a1, r1, c1);
```

```
p-f1 "Enter the 2nd matrix elements [n^2];
```

```
accept(a2, r2, c2);
```

```
void display(int l1[20], int m, int n)
```

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

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

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

```
    p-f1 "%d", a1[i][j];
```

```
    p-f1 "\n";
```

```
}
```

```
void addMatrix(int a1[20], int b1[20], int c1[20], int m, int n)
```

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

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

```
{
```

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

135

{

$$C[i][j] = a[i][j] + b[i][j];$$

}

}

Multiplication:-

Matrix Multiplication is possible when no. of columns in 1st Matrix is equal's to No. of rows in 2nd Matrix.

[A]_{2x2} [B]_{2x3}

3x2 2x3

Resultant Matrix = 3x3.

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}_{3 \times 2} \times \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}_{2 \times 3}$$

$$R_1 = 3 \quad C_1 = 2 \quad R_2 = 2 \quad C_2 = 3$$

Matrix

? Int Mat1[30][30] Mat2[30][30] Mat3[30][30], i,j,x,sum;

Int R1, R2, C1, C2;

ClearScreen;

P-FT Enter the size of the mat1(n^o);

S-FT "x.d x.d", &R1, &R2);

P-FT Enter the 2nd matrix Mat2(n^o);

S-FT "x.d x.d", &C1, &C2);

If(C1 == R2)

? P-FT Enter the 1st matrix(n^o);

for(i=0; i<R1; i++)

? for(j=0; j<C1; j++)

? S-FT "x.d", &Mat1[i][j]);

{

? P-FT Enter the 2nd matrix(n^o);

? for(i=0; i<R2; i++)

{

```

{
    for(j=0; j < c2; j++)
        sum += mat2[i][j];
}

for(i=0; i < r1; i++)
{
    for(j=0; j < c2; j++)
    {
        sum = 0; mat1[i][j] = 0;
        for(k=0; k < c1; k++)
            sum = sum + mat1[i][k] * mat2[k][j];
        mat3[i][j] = sum;
    }
}

/* display in matrix form */
for(i=0; i < r1; i++)
{
    for(j=0; j < c2; j++)
        pff(" %d ", mat3[i][j]);
    pff("\n");
}
else
    pff(" multiplication not possible.");
getchar();
}

```

→* Write a program to find out the sum of Principle diagonal elements. $(r_1 = c_1)$
 " accept an array of matrix, display the transpose of matrix.

MultiDimension arrays:

The Multidimensional can be used for the efficiency of the programming. we can define any no. of dimensions just similar like pointer.

int arr[5][3][4];

\downarrow \downarrow \downarrow
Block's Row Column

int arr[5][3][4];

\downarrow Left most dimension is optional, when you initialise an array

int arr[5][6][2][4];

\downarrow \downarrow Row Column

set Block

int arr[3][4][5];

See one become start's form

1 → arr[0][0][0];

Column ← 5) 36 (7. Row ← 4 | 7 | 1 → Block

27 → arr[1][3][1];

$\frac{35}{3}$ 4

14 → arr[0][2][3];

Coloorn ③ → Row

59 → arr[1][3][3];

5) 18 (2 0th block.

5) 58 (11 4) 11 / 2 → Block.

$\frac{10}{3}$

$\frac{55}{3}$ 8
3 → Coloumn 3 → Row

2nd Row

3rd Coloumn

Adding set :- int arr[5][3][4][5];

5) 146 / 29 4) 29 / 7 3) 7 / 2 → set

10 → arr[0][0][0][0];

$\frac{145}{2}$

147 → arr[1][2][1][1][1];

① ②

Row

1 - 8 block

234 → arr[3][2][2][3];

$\frac{23}{3}$

96 → arr[1][1][3][0];

$\frac{20}{3}$

33

4 →

2 →

1 →

Block

5) 96 / 19 4) 19 / 4

$\frac{45}{3}$ 3

$\frac{45}{45}$ 0

30

2

1

Representation of pointers for multidimensional arrays:

Main()

→ 2 blocks.

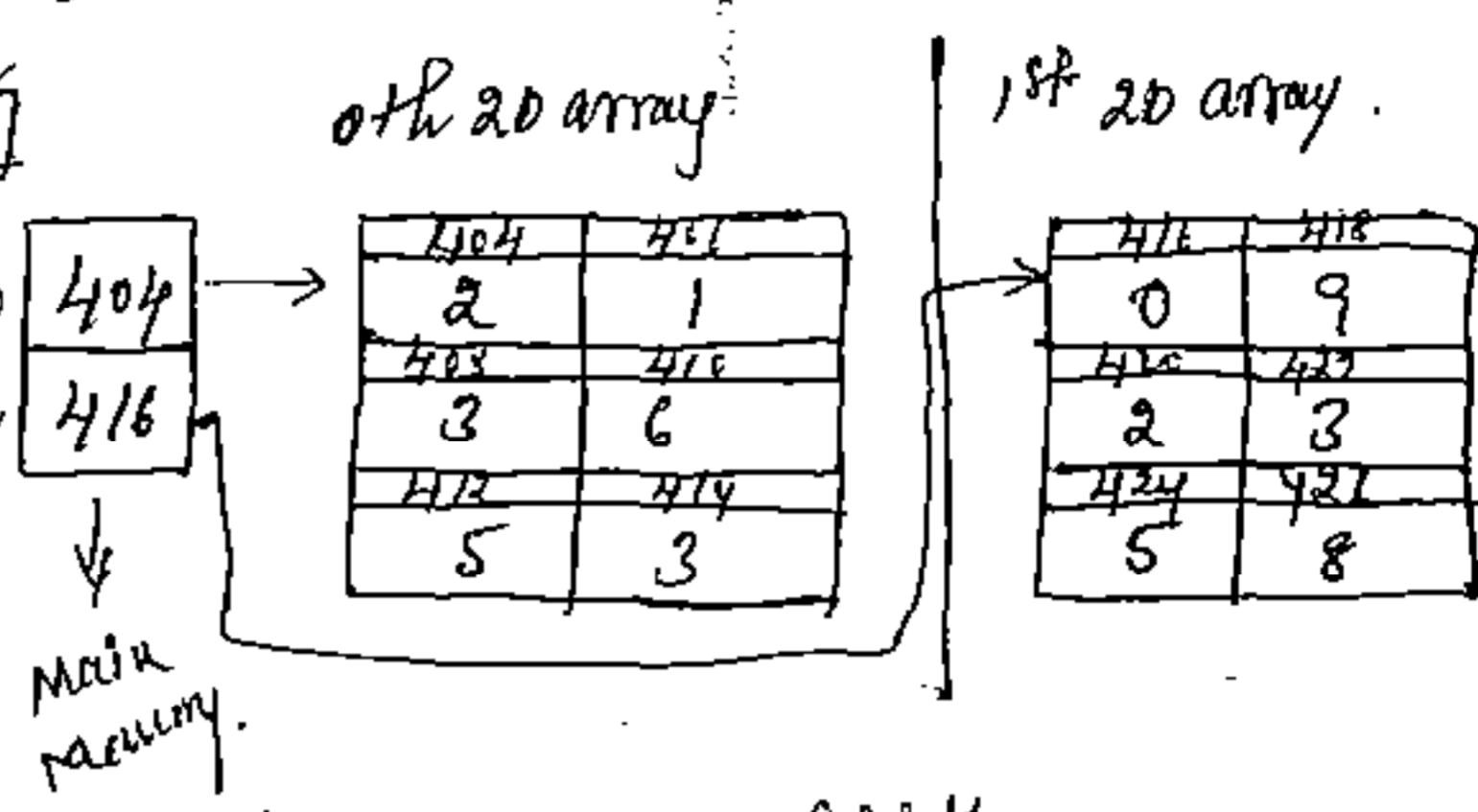
{ Static sub-arr [2][3][2] = {{ {2,1}, {3,6}, {5,3} }, { {10,9}, {2,3}, {5,8} } }

int i, j, k;

int *arr[2];
arr[0] = 404; arr[1] = 416;

p-f("Y.dlu", arr); 404
 p-f("Y.dlu", *arr); 404 1000
 p-f("Y.dlu", **arr); 1004 1002
 p-f("Y.dlu", ***arr); 2
 p-f("Y.dlu", arr+1); 416
 p-f("Y.dlu", *arr+1); 408
 p-f("Y.dlu", *arr+2); 412
 p-f("Y.dlu", ***arr+1); 3
 p-f("xdlu", *(***arr+1)); 1
 p-f("Y.dlu", *(***arr+2)); 3
 p-f("Y.dlu", *(***arr+6)); 0

for(i=0; i<2; i++)
 { for(j=0; j<3; j++)
 { for(k=0; k<2; k++)
 p-f("Y.d", *(*(arr+i)+j)+k));
 p-f("Iu");
 }
 }
 }
 printf("Iu");



Main
memory

arr

1000 + 1 = 1001

(1000 + 1) = 1001

*(**arr+1) 1002

*1000 → Block 416

*404 + 1

*404 + 1

+ 406 = 1

Here 1st preference goes to

pointers. *arr = *(arr+0)

*arr 1000 + 0

*1000 = 404.

→ 1000 is MainMemory. *(arr+0)

address *1000 *(1000+0)

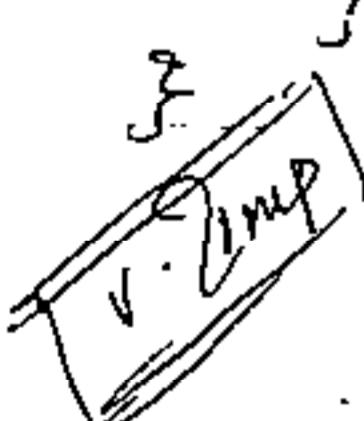
Value at 1000 = 404 *(1000) =

→ 404 is Value in MainMemory. *(404) = 404

but it is also a address of memory.

block 0. So Main value points to Block address

i.e. 406 *404 → it points to Block address



→ Write a prog on Multidimensions & store some Marks.

: 137

#include <stdio.h>

#include <conio.h>

This is multidimension for Student -

void main() { } Marks in all 3 years for one student -

{ int Marks[2][3][5][2], i, j, k, l; to find more student add
int subsum, semsum, yearsum, grandtotal = 0; no. of students
class[]; size at leftmost one.

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

{ yearsum = 0;

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

{ semsum = 0;

for(k=0; k<5; k++)

{ subsum = 0;

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

{

P : f1. enter 1. d year x d sem x d sub x d marks: i+1, j+1,
k+1, l+1;

S : f1 " 1. d ", & Marks[i][j][k][l]);

subsum = subsum + Marks[i][j][k][l];

}

semsum = semsum + subsum;

}

yearsum = yearsum + semsum;

{ grandtotal = grandtotal + yearsum;

}

p : f1 " GRANDTOTAL = 1d ", grandtotal);

getch();

}

Dynamic Memory allocation:-

- As in - the Static memory allocation, like array's there is a wastage of Memory Block's. Since a static memory allocation must be define the size before execution of the program, due to this either we can't increase the size (or) decrease the size at the time of declaration. Either we want to increase (or) decrease the size if is not possible.

To overcome this drawback we can implement Dynamic memory allocation.

The D.M.A allocate the memory dynamically at the time of execution.

STACK V/S HEAP ALLOCATION:-

→ The memory allocated into stack has both name and address, therefore memory location can be accessed by name or through address.

→ The memory allocated into heap don't contain a name instead contains only the starting address of the memory allocated, therefore accessed through this address.

→ The memory allocated through heap can be local or global respective to it's functional definition.

→ If memory allocation is global then that memory can be accessed through a different function by locating it's address.

→ In static memory allocation the memory is allocated and freed by system.

→ In dynamic memory allocation it is task of the programmer.

Either in Static (or) Dynamic the Memory allocation will be takes place at Execution time, Only for static we define the size at the time of Compilation.

To implement the dynamic Memory Concept 'C' lang Support's the dynamic memory functions like malloc(), calloc() and realloc() but deallocation we have free functions.

All the above functions by default return type is either
Char * (As per K & R C) }
Void * (ANSI C) } return type of malloc(), calloc(), realloc()

All these functions allocate the Memory from the heap area.

malloc():

This malloc() function is used to allocate required no of bytes in Memory at runtime. It takes One argument: i.e. Size in bytes.

Syntax:-

Void * malloc (Size - t Size);

(Q8)

Pointer - var = (type cast - *) malloc (size of (size));

Example:-

int *a;

a = (int *) malloc(4);

'4' is the size (in bytes) of Memory to be allocated.

/* Example of allocating memory dynamically */

#include < stdio.h >

#include < malloc.h >

Void main()

{ int *P, n, i;

clrscr();

```
printf("Enter the size of n:");
```

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

```
p = (int *)malloc(sizeof(n));
```

```
if(p == NULL)
```

```
{
```

```
p-> Memory allocation failed - In);
```

```
getch();
```

```
exit(0);
```

```
}
```

```
p-> Enter the Elements... In);
```

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

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

```
printf("The Elements are - In);
```

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

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

```
getch(); free(p);
```

```
}
```

In the above program the 'p' doesn't know what type of element's is pointing, & if doesn't know how many elements it is pointing, you access the dynamic memory only by address (ie only base address).

calloc():

Used to allocate required number of bytes in memory at run time. It needs Two arguments viz -

1. Total no. of data and

2. Size of each data.

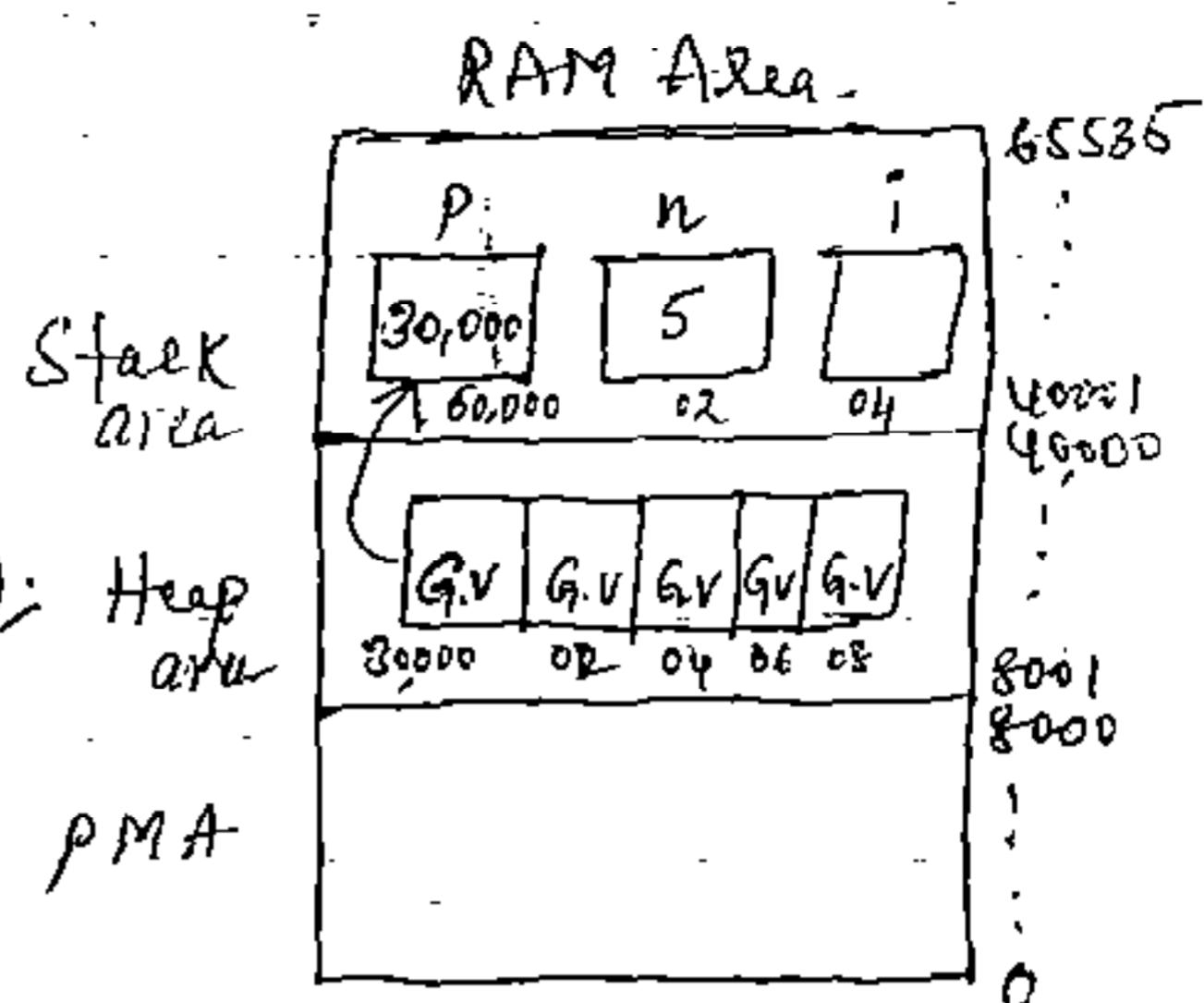
Syntax:-

```
Void * calloc(Size-t nmemb, Size-t size);
```

Example:-

```
a = (int *) calloc(8, sizeof(int));
```

Here sizeof indicates the size of the data-type and 8 indicates that we want to reserve space for storing 8 integer.



→ The diff b/w malloc() & calloc(). The malloc() accept's one argument & calloc() accepts 2 arguments.

After allocating the memory by the malloc() function it stores Garbage value.

The memory allocated by calloc() it will assign's zero's(0).

→ The above program can be called as dynamic array single-dimensional array.

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

Main()

```
{ int *a, r, c, i, j;
```

printf("Enter the no. of Row's &

cols:");

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

a = (int **) malloc(r * sizeof(int)); PMA

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

{

*(&a[i]) = (int *) malloc(c * sizeof(int));

}

printf("Enter the values: ");

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

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

{ scanf("%d", &*(a+i)+j);

}

}

P-f1 * fd; *(*(a+i)+j));

} p-f1("1u");

}

→ Write a prog of ~~depth~~. a Multidimensional Example - by the dynamic array without any wastage of blocks. (previous MCA Example).

In the previous MCA student example 3 year's each year 2 semesters & each semester 6 subjects, ~~but~~ based on this we can allocate the array size & used it - but in last semester's only 3 subjects no. of subjects are decrease In this case Only 3 subjects utilises memory, remain are wastage based on dynamic memory we allocate ~~no of~~ & how many columns we require for 1 row's. for diff rows diff col no's column size is also possible.

#include < stdio.h >

< conio.h >

< alloc.h >

Void main()

{ int *KA, Y, *C, I, J, X, Y, K;

clrscr();

p. ff " Enter No. of Rows : " ;

S. ff " .d " , & R);

C = (int *) malloc (R * sizeof (int));

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

{

p. ff " Enter No. of Coloum's for a row : .d " , & I);

S. ff " .d " , & C + i);

}

A = (int **) malloc (R * sizeof (int));

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

{

* (A + i) = (int *) malloc (* (C + i) * sizeof (int));

{

p. ff " Enter the values : |u ");

Y = wherey();

K = whereX () + 4;

Here * is

→ Multiplication Operator

```
for(i=0; i<l; i++)
{
```

$x = k;$

```
for(j=0; j<*(c+i); j++)
{
```

gotosxy(x,y);

pst "y.d, *(a+i)+j);

$x = x + 4;$

}

$y +=$

pst "1u\n Given value: 1u");

$y = wher(y);$

$k = wher(x) + 4;$

```
for(i=0; i<8; i++)
{
```

$x = k;$

```
for(j=0; j<*(c+i); j++)
{
```

gotosxy(x,y);

pst "y.d, *(a+i)+j));

$x = x + 4;$

$y +=$

off Enter the no. of Rows: 3

? Enter the no. of columns for row 0: 3

getch();

" " " for row 1: 2

? Enter the no. of columns for row 2: 1

Enter the Values:

56 47 89

27 34

19

Given Value: 56 47 89

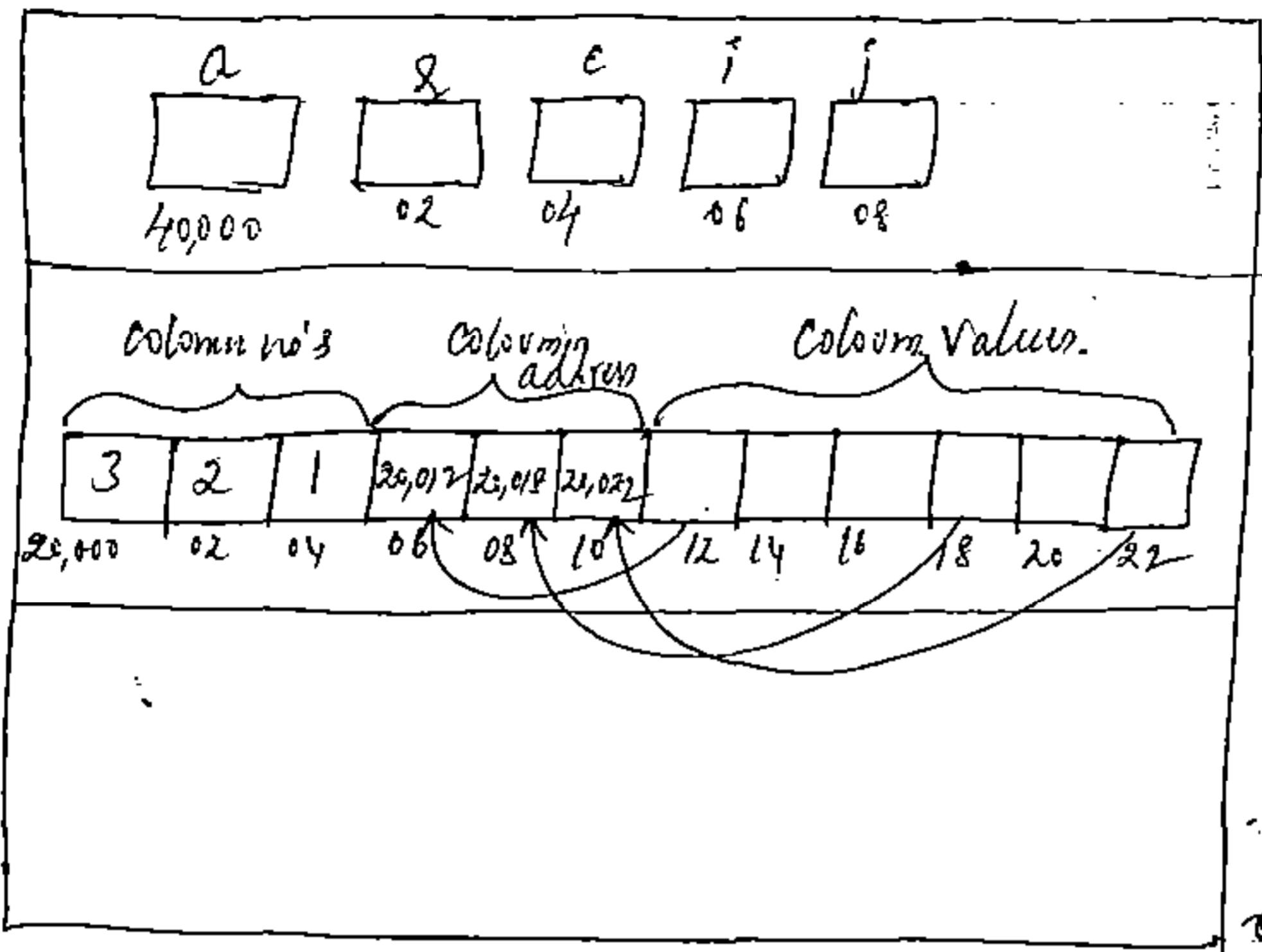
27 34

63, 535

Stack area

Heap area

PMAT



Realloc()

The realloc() function makes you increase (or) decrease the ~~#~~ size of any dynamic memory, if it is allocated by using malloc() (or) calloc().

Syntax:

```
void * realloc(void *ptr, size_t newsize);
```

- The first argument 'ptr' is pointer to the memory previously allocated by the malloc (or) calloc() functions.
- Second argument 'newsize' is the size in bytes, of a new memory region to be allocated by realloc. This value can be larger (or) smaller than the previously allocated memory. The realloc function adjusts the old memory region, if newsize is smaller than the size of old memory.

If the newsize is larger than the existing ~~size~~ memory size, it increases the size by copying the content's of old memory region to new memory region. The function then deallocates the old memory region. Realloc function is helpful in managing a dynamic array whose size may change during execution.

free() :-

whatever the memory is allocated by the dynamic-memory allocation, they explicitly has to deallocate by using free() function.

If the user not deallocated until the user shutdown the system the memory deallocation will not takes place.

Once we implemented free() function it automatically deallocated & given back the memory area to the heap area.

free(pointer variable);

Unformatted i/p o/p :-

The unformatted character input functions will accept only one byte of character & display only one byte of character.

There is no i/p, o/p functions for integers & float we have String i/p, o/p functions.

ASCII Value

(Enter key) \n = 10

Given as 0/1's

\r = 13

\t = 9

\b = 8

' ' = 32 ~~space~~ (Space)

'0' = 0

EOF = 26 (end of file)

getchar() :-

The getchar() is similar to scanf() function. both this functions can capture only data ^{keys}: The getchar & scanf function of the last is the key is the Enter key.

The getch() function will accept the i/p, until you press enter key.

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

```
main()
```

```
{ char ch;
```

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

```
ch = getch();
```

```
p-f("x.c", ch);
```

```
getch();
```

```
}
```

→ The getch() will accept the character until the user press enter key. Until then all the characters will store in buffer, from the buffer if accept only one character. If is defined under header file of stdio.h.

→ The scanf() & getch() functions, will accept the Enter key, as an \n:-10.

getche() & getch() :-

This two functions are similar, both this functions can Capture data keys, & also Non data keys. (uparrow, downarrow)

getche() :-

The getche() function will also accept Only one character as an I/P. The getche() function doesn't wait for any key after given an I/P.

getche() & getch() will be defined from Console i/p.h (conio.h) header file.

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

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

```
main()
```

```
{ char ch;
```

```
p-f(" Enter the character!");
```

```
ch = getche();
```

```
p-f("x.c", ch, ch);
```

```
getch();
```

```
}
```

getch() :-

The getch() function don't show the character's whatever the user type at the time of i/p.

#include <conio.h>
Main()

by using getch() function we implement's
the password logic.

{ char ch;

p.f("Enter the character:");

ch = getch();

printf("%c", ch);

getch();

} → By this function we can write a password logic.

→ #include <conio.h>

Main()

→ The getch() function & getche()
will capture the enfekey as a 'Y' - 13

{ char ch;

p.f("enter the character:"); based on situation it also as 'Y'
ch = getch(); accu - 13 will display.

p.f("%c %d", ch, ch);

getch();

The ASCII code of every Non-datakey Staff's

} with zero & followed by the Number.)

Ex:- uparrow → 0x70 Non datakeys

#include <conio.h>

Main()

UP arrow → 0x72

{ char ch1, ch2;

down arrow → 0x80

p.f("enter the character:");

left → 0x75

ch1 = getch();

right → 0x7F

ch2 = getch();

pg up → 0x73

p.f("%d %d", ch1, ch2);

page down → 0x81

getch();

Generally every open, close, moving by using up, down

} arrow key's moving the C edit screen if possible,
become already that logic implemented on prog logic, like that

in 1 - 90

getchar and puts :-

The gets() function will capture any no. of characters as an I/O. It is an String I/O function.

It can capture the I/O's until the user press Enter key.

The puts() function will display the I/O's. These both are Only for String I/O I/O.

#include <conio.h>

Main()

{ char ch[100];

puts("Enter the String:"); This is a class

/* scanf(" %s", ch); */ This

gets(ch); I/O - This is a class

puts(ch);

getch();

}

→ The drawback of scanf() if can't read the string's when whitespace is occurs, by using gets() function we can overcome it.

gotoxy() :-

Syntax :- gotoxy(col, row);

The gotoxy() makes you to move the function in any were of the ~~position~~ rows & column's.

position.

#include <conio.h>

& Main() I/O : If I/O 'A' is present at 40 column, 12 row.

{ clrscr();

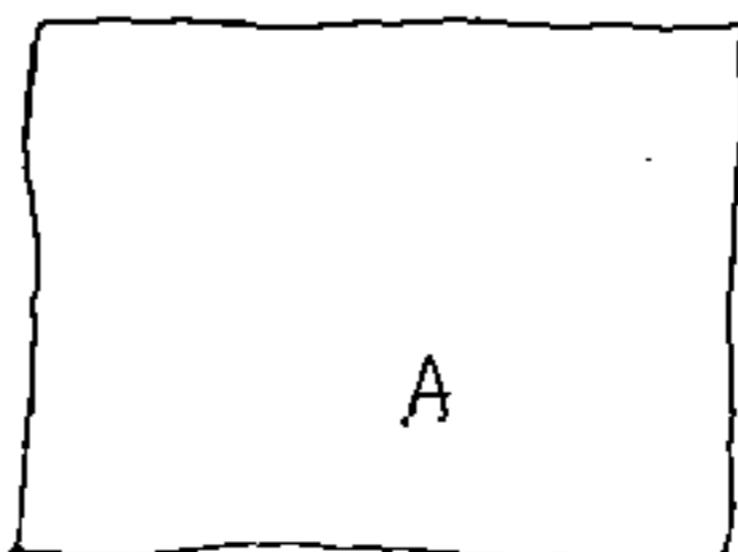
gotoxy(40, 12);

p.f(" A");

getch();

}

I/O
Screen



Where x():

143

where x() gives the current cursor horizontal position,
if returns an integer value, within the range of 1 to 80.

Where y() This function gives current vertically cursor position.
if returns an integer value, in the range of 1 to 25,
1-43 & 1 to 50.

→ write a prog to move the cursor on the editor

Void Main()

{ char ch1, ch2;

int x, y;

ch1 = getch();

while (ch1 == 0)

{

ch2 = getch();

x = Where x();

y = Where y();

If (ch2 == F2) → update the above program by substituting
goto xy(x, y-1); in the prime no logic.

if (ch2 == FF)

goto xy(x+1, y);

if (ch2 == 80)

goto xy(x, y+1);

if (ch2 == 75)

goto xy(x-1, y);

ch1 = getch();

}

→ Write a program to display the cursor positions whenever the cursor is moving.

Example on ScreenSaver programming Using gotoxy()

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

void main()
{
    char sf[180];
    int i, j, c;
    clrscr();
    p.f("Enter the String:"); 3 types of
    gets(sf);                Custom style as you wish.
    — setcursorstyle(-SOLIDCURSOR);
    while(!kbhit())
    {
        c = (80 - strlen(sf)) / 2;
        clrscr();
        gotoxy(c, 1);
        p.f("Y.S", sf);
        // Top to bottom
        for(i=0; i<strlen(sf); i++)
        {
            if(sf[i] != 32)
            {
                for(j=2; j<=25; j+f)
                {
                    gotoxy(c, j);
                    p.f("Y.C", sf[i]);
                    gotoxy(c, j-1);
                    p.f("Y.C", 32); spacebar
                    delay(20);
                }
            }
        }
        // Bottom to top
        for(i= strlen(sf)-1, i>=0; i--)
        {
            c--;
            if(sf[i] != 32)
            {
                for(j=2; j<=25; j+f)
                {
                    gotoxy(c, j);
                    p.f("Y.C", sf[i]);
                    gotoxy(c, j-1);
                    p.f("Y.C", 32);
                    delay(20);
                }
            }
        }
    }
}
```

Example 2:

loop(2)

K

(3)

K

(4)

K

144

loop(1) K I R A N (5) o/p terminate.
A A A

(6) N

N

N

(7)

(6)

Example 3: Scrolling

← Ban

Kiran

o/p: terminate.

#include <stdio.h>

<conio.h>

<string.h>

<dos.h>

void main()

{ char st[40], ch;

int i, j, k, c;

clrscr();

textmode(1);

printf("Enter the string:");

gets(st);

stupr(st);

strcat(st, " ");

setcursestype(NORMALCURSOR);

clrscr();

textcolor(RED);

textbackground(WHITE);

c = (40 - strlen(st))/2;

gotoxy(c, 12);

cprintf("Y.S", st);

while(!kbhit());

{

ch = st[0];

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

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

st[i-1] = ch;

st[i] = 'O';

gotoxy(c, 12);

printf("Y.S", st);

delay(200);

} getch();

<Clipper.h>:

Strings

This <ctype.h> functions will accept a character as it's ASCII code, then find out if it is what type of Character & returns an integer value either true(1), (or) false(0).

int isalnum(int c);
int islower(int c);
int isalpha(int c);
int isdigit(int c);
int isasciil(int c);
int isupper(int c); ... etc.
All these are basically called as
Predefined functions.

`toUpper()` If Convert the lower case to Uppercase (only one character)
`toLower()` If Convert upper to lower.

Generally we convert lower to upper without these functions.

$$Ch = 'a' \quad Ch = Ch - \underline{3x} \quad Ch = A \quad \left. \begin{array}{l} A = 65 \\ a = 97 \end{array} \right\}$$

difference.

$$ch = \gamma_1' \quad ch = ch + 3\alpha; \quad ch = \alpha.$$

→ Main()

{ char style] = "Klein";

$\inf i = 0$;

```
while (i <= 5)
```

```
{     printf("c",str[i]);
```

14

}	0	1	2	3	4	5
	K	i	g	a	n	10

Null-English
0 - ASCII value

74

\Rightarrow The will display if the size of the String either increase (or) decrease the condition will not work's.

→ We need a perfect Condition
either String increase (or) decrease.

„NULL - English word String will be ended one special character

θ -Assume

Called as Null character.

10' The Null Character always Stores at the end of the String
So When the Strings are implemented Use Null Character as a Condition.

```
{ char str[] = "Kiran";
int i=0;
```

```
while(str[i]!='\0')
```

```
{
```

```
p.f("%c", str[i]);
```

```
{ if(
```

```
{
```

→ In 'C' language if you not assign any space the the NULL character won't occupy but there is a chance of displaying garbage character's.

In C++ The NULL character forcedly will occupy one byte of Space.

<u>C</u>	<u>C++</u>
<code>char str[5] = {"Kiran"};</code>	<code>char str[5] = {"Kiran"};</code>

Kiran g.v.g.r //so please assign 'i' through
given an empty space in array
may size

Kira

forcedly last bit occupied by '\0'

Diff b/w `char[]` and `char *` :-

<code>int Main()</code>	<code>/*impi = impf + 2; invalid */</code> <code>puts(in);</code>
<code>{ char impf[10] = "Kiran";</code> <code>char *expf = " Dennis Ritchie";</code> <code>char dupim[10];</code> <code>char *dupex;</code> <code>clrscr();</code> <code>puts(ar); printf(impi); printf("%u");</code> <code>puts(ar); printf(expf);</code>	<code>expf = expf + 7;</code> <code>puts(expf);</code> <code>/* dupim = impi; *is valid */</code> <code>strcpy(dupim, impi);</code> <code>puts(dupim);</code> <code>deepeX = expf;</code> <code>printf(deupeX);</code> <code>getch();</code> <code>return 0;</code>

* We are displaying the strings by using `printf()`.

→ We not specify any format specifications.

Since for the first argument of `printf()` is String type. So there is no necessity to explicitly specify any format specifications.

③ All above are invalid for
array is a constant pointer.
If it's contain base address
we can't change (or) assign to
another.

→ it accepted static type of String & dynamic type.

O/P: Kiran

Dennis Rechie

→ Void Main()

{ Char str[10] = "abcdef";

printf("%s", str+3); def

str[3] = 'b';

printf("%s", str); abcdef

str[3] = '0';

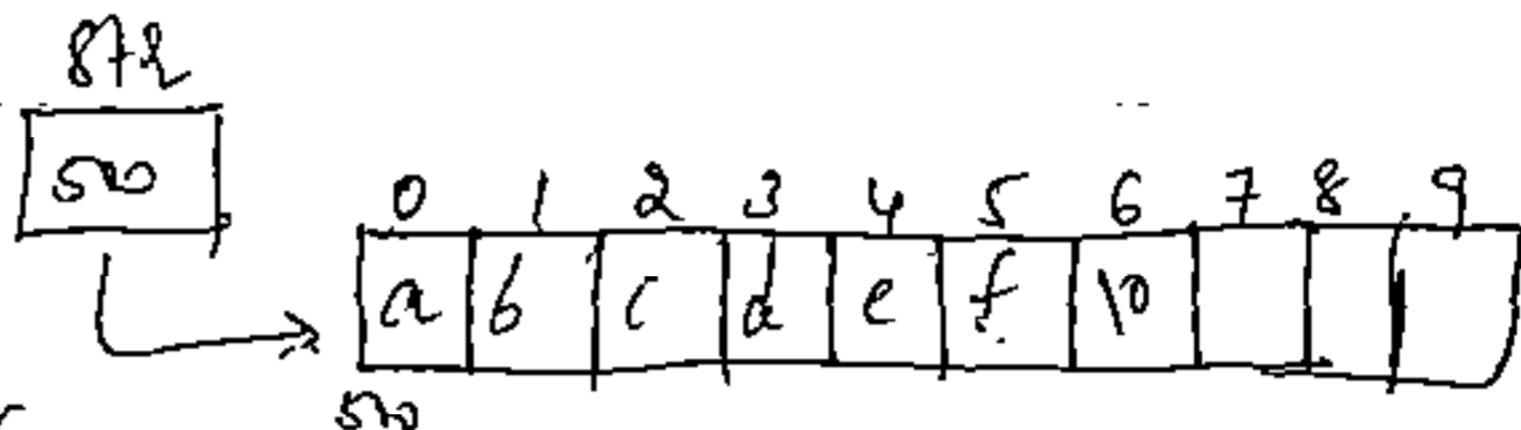
printf("%s", str); abcdef

str[3] = '0';

printf("%s"); abc

str[3] = 'd';

printf(str); abcdef



}

→ Void display(char *pt[2])

{ printf("%s", pt[1]);

if (*pt[2])

display(pt[1]);

printf("%s", pt[2]);

}

Void main()

{ Char str[10] = "Hello";

display(str);

}

Hello

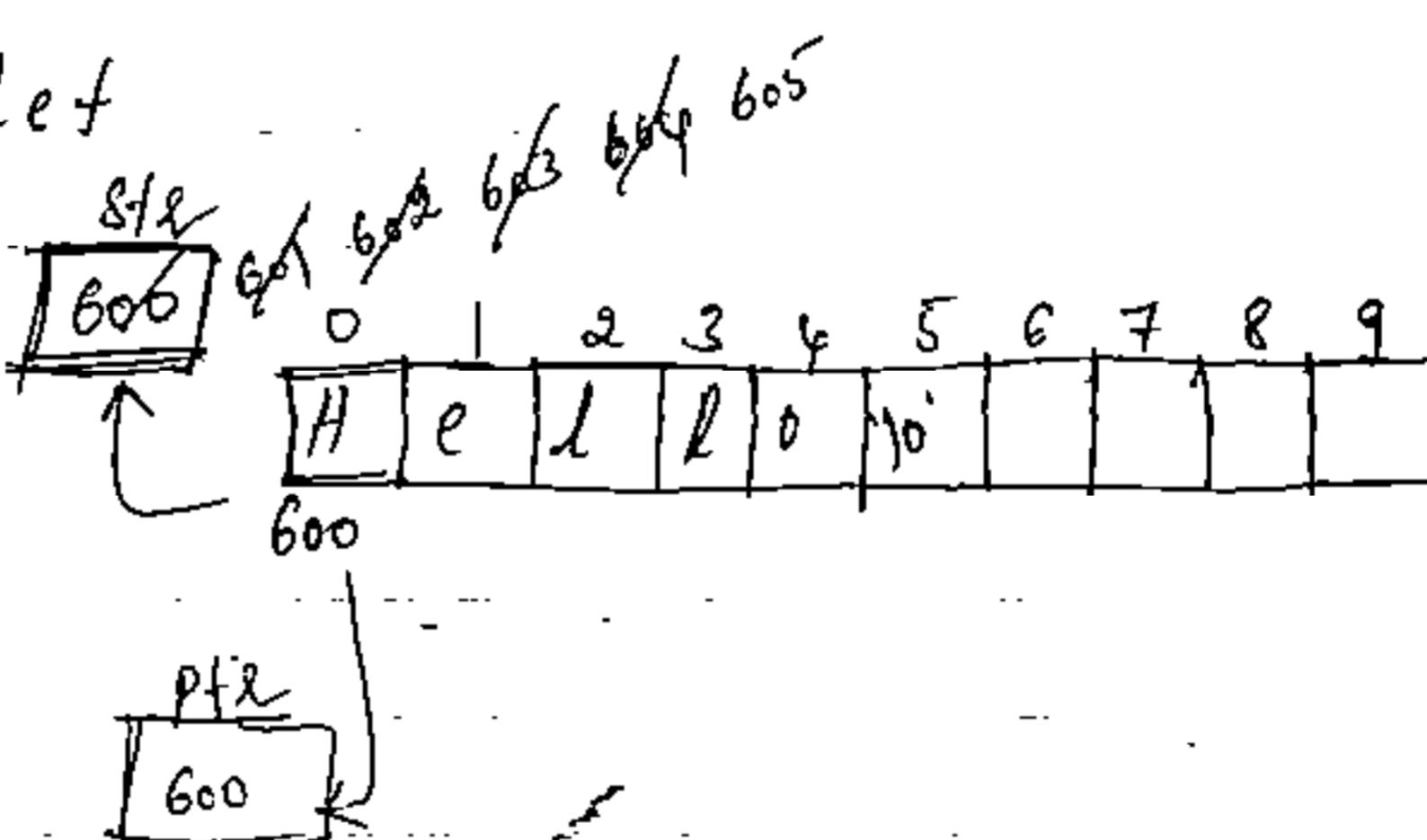
ello

llo

lo

o

L



O/P: Hello

ello

llo

lo

o

L

l

l

o

ello

ello

146

void main()

{ char str[10];

str[0] = 'a';

str[1] = 'b';

str[2] = 'c';

puts(str); abc G.C.G.C.

}

↳ Garbage characters.

for removing G.C.
Chrs.

char str[10] = " ";

str[0] = 'a';

str[1] = 'b';

str[2] = 'c';

puts(str); abc

→ void main()

{ char s[150] = "Hello";

char s2[50] = "Cobal";

s1 = s2; // Error:

puts(s1);

puts(s2);

}

< string.h >

1. strcpy(T, S); T = target S = source

It copy the source string to target string.

2 strcat(T, S);

It concatenate (or) Combining 2 strings

into one string

3 strrev(T);

It reverse the string

4.strupr(T);

If convert the string into uppercase.

5. strlwr(T);

If convert to lowercase

→ All the above string functions returns pointer to char. (i.e String type)

6. l = strlen(str);

If find the length of the string including with spaces.

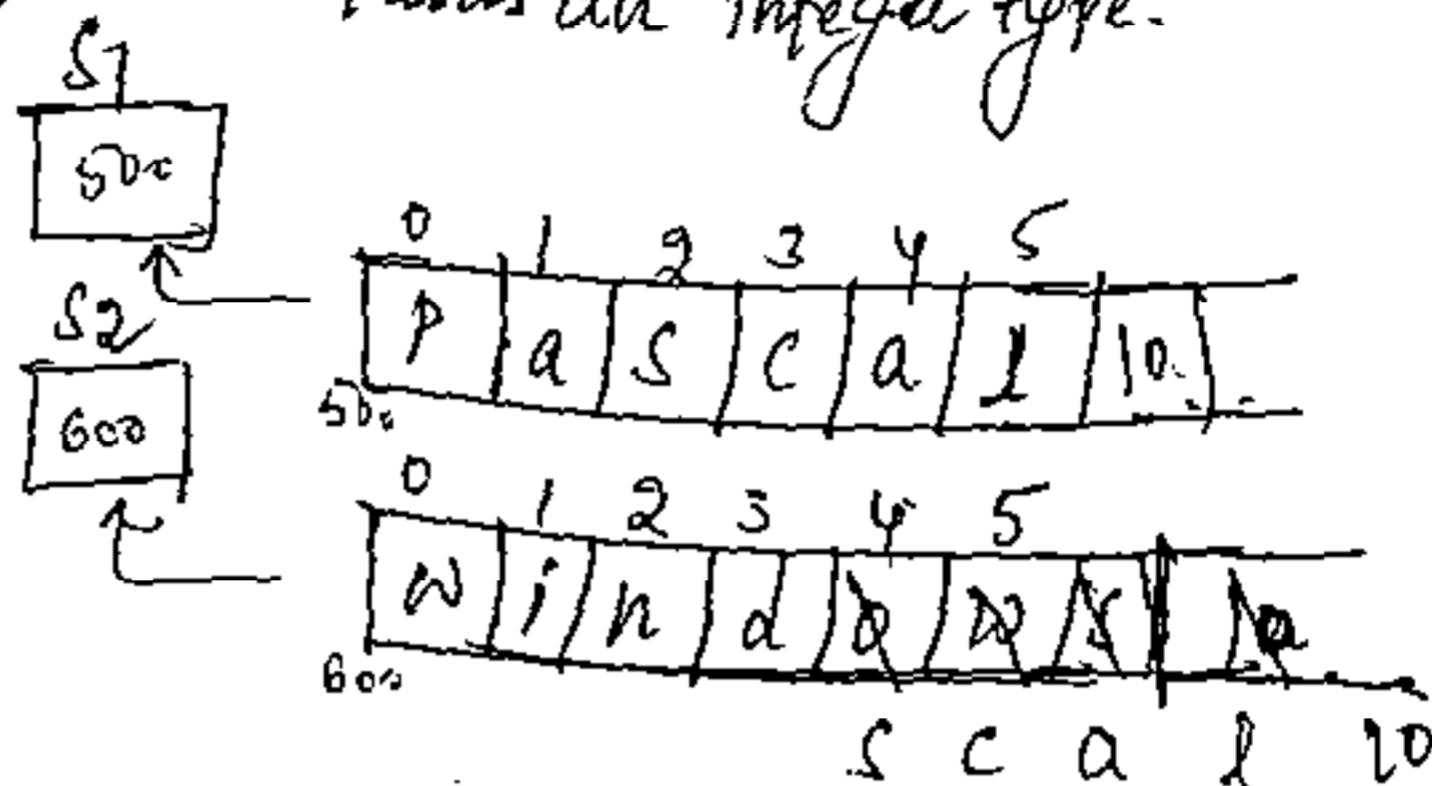
7. $l = \text{strcmp}(s_1, s_2)$; If Compare 2 String's based on the ASCII codes.

8. $l = \text{strcasecmp}(s_1, s_2)$; If Compare 2 strings by ignoring case sensitive.

→ All the ~~all~~ above functions returns an integer type.

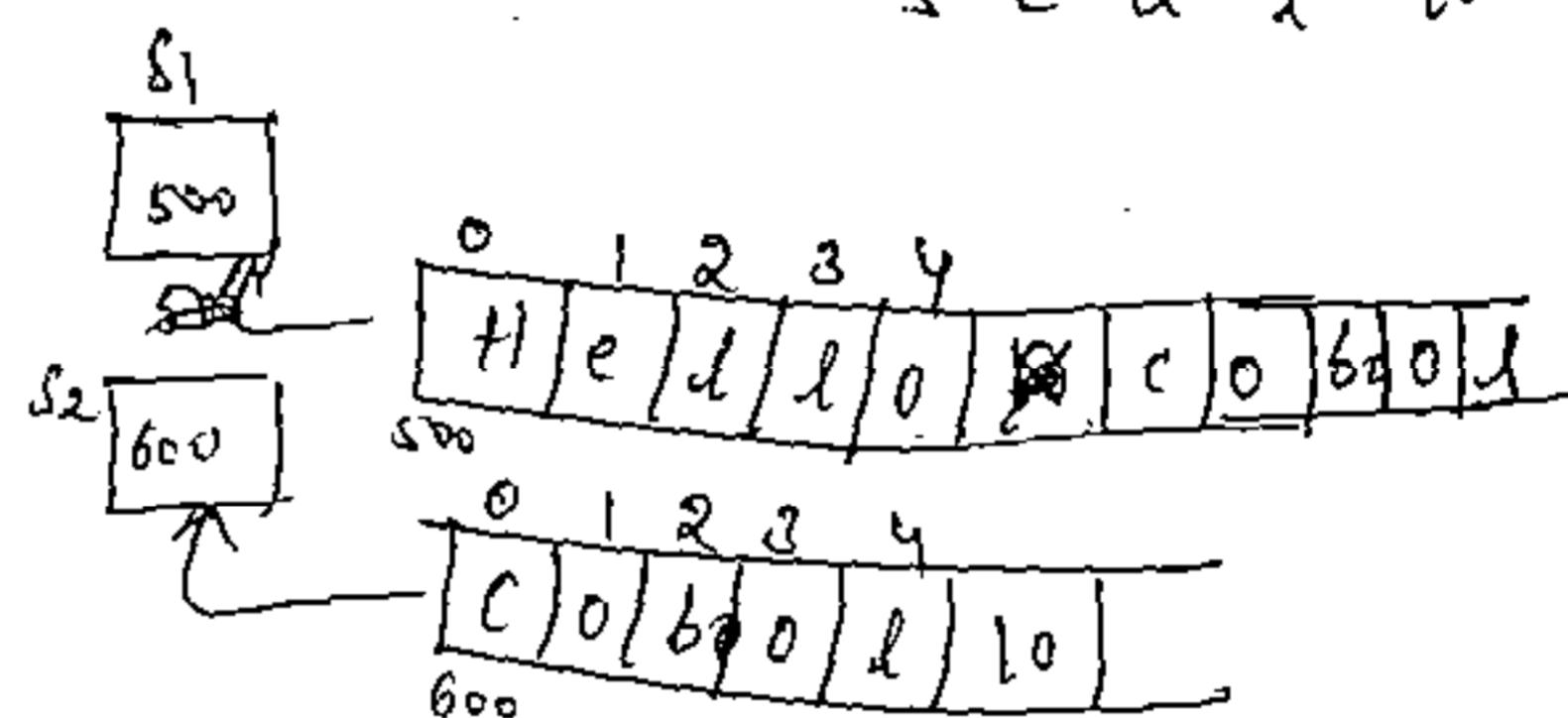
Void main()

```
{ char s1[10] = "Pascal";
char s2[10] = "Windows";
strcpy(s2+4, s1+2);
puts(s1); pascal
puts(s2); Windows
}
```



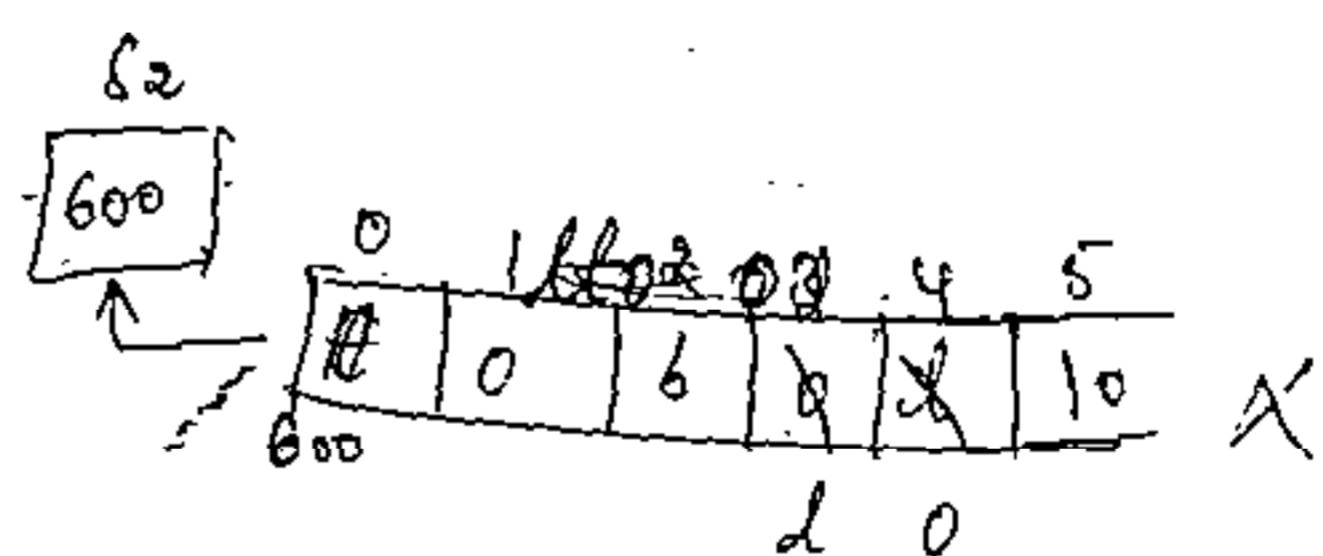
→ Void main()

```
{ char s1[5] = "Hello";
char s2[5] = "cobel";
strcat(s1, s2);
puts(s1); HelloCobel
puts(s2); cobel.
}
```



→ Void main()

```
{ strncat(s2+3, s1+3);
puts(s1); Hello
puts(s2); Coblo.
}
```



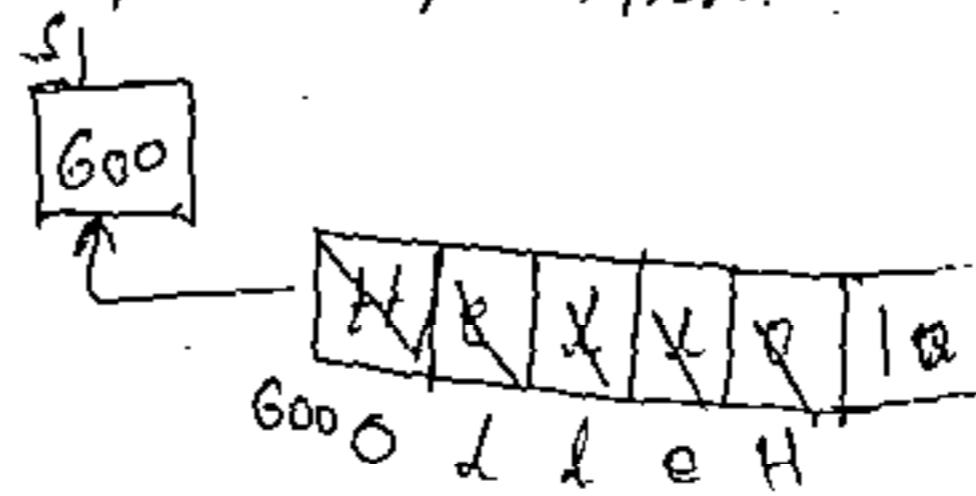
~~Note:~~

String Copy the data from the given position.

→ String Concatenate always copies from a Null character, either you define (or) not define the position.

→ Void main()

```
{ char s1[10] = "Hello";
strrev(s1);
puts(s1); olleH
}
```



→ String reverse doesn't reverse the NULL character.

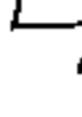
→ void main()

```
{ char s[150] = "Pascal";
  Stlrev(s, t2);
  Puts(s1);
```

}

s1

500



0 1 2 3 4 5

P a x x x x \0

500 1 a c s

→ char s[150] = "Hello 8910111213";

```
Stlrev(s, t5);
Puts(s1); Hello 3121110198
```

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

H e l l o 8 9 1 0 1 1 1 2 1 3

3 1 2 1 1 1 0 1 9 8

Stlent:-

Main()

```
{ char str[100];
```

int i;

Puts("Enter the String:"); Taf India

gets(str);

i = Stlent(str); q (including space)

printf("%d", i);

P-f("%d", Stlent(str)); q

getch();

}

→ The String length Never Count the null character. But

→ Main()

Sizeof will Count the null character

{ int i;

char str[50] = "abcdefg";

length of operator

P-f("%d %d %d", Stlent(str), sizeof(str), sizeof("abc"));

}

8

50

4

but sizeof will

500

0 1 2 3 4 5 6 7 8 9 10

R e n d e z v o u s \0

→ Char s[1] = "Rendezvous";

P-f("%d %d", *(s+Stlent(s)));

500 f. 11

= 500 - 0

10 = 0

strcmp :-

If compares 2 strings based on the ASCII Codes
if String 1 > String 2. It returns a positive value.

if String 1 < String 2 it returns (-ve) value

if both the strings are same it return zero.

→ The value will be differences of 2 characters.

Void Main()

{ Char S₁[50] = "abcd"; 1st string 1st character ASCII

Char S₂[50] = "abed"; Compares with the

int i; 2nd string 1st character like

i = strcmp(S₁, S₂); 2nd - 2nd ...

P.f("d", i); 0

}

→ abcdef; 0/1: (+ve) value "abcdef"

abcaaa; 100 - 97 = 3.

0/1: (-ve) value

→ "Helloabc"

"Kilabc"

i = strcmp(S₁+5, S₂+3);

P.f("d", i); 0.

Strcmp:-

It ignores the Case sensitive.

main()

{ Char S₁[50] = "KiranSrinivas";

Char S₂[50] = " KiranSrinivas";

int i;

i = strcmp(S₁, S₂)

P.f("d", i); 0.

}

→ Write a prog accept a String of Paragraph Count no. of words present in the paragraph.

Void Main()

{ Char Str[100]; Int i, Cnt=1;

puts("Enter the String:");

gets(Str);

for(i=0; Str[i]!='\0'; i++)

{ if (Str[i]==32)

Cnt++;

}

printf(" Number of words: %d", Cnt);

}

→ Write a Prog accept a String of Paragraph display the String in the format I/P: This is a data

I/P: This

Void Main()

{ Char Str[100]; ~~This is a data~~; a

int i, Cnt=0;

is

data

puts(" Enter the String:");

gets(Str);

for(i=0; Str[i]!='\0'; i++)

{ if (Str[i]==32)

Str[i] = '\n';

pf("%u");

}

→ " String of Paragraph & display the String in reverse Order

Void Main()

{ Char Str[100];

gets(Str);

Str = ~~for(i=0; Str[i]!='\0';)~~

strrev(Str);

puts(" Enter the String:");

→ gets(str)

l = strlen(str);

for(i = l - 1; i >= 0;

/* program for reverse the total string */.

void main()

{ char name[26];

int i, j, s;

clrscr();

p-f("Enter the Name:");

fflush(stdin);

gets(name);

j = strlen(name) - 1;

/* s = strlen(name)/2; */

for(i = 0; i < (strlen(name))/2; i++, j--)

{

name[i] += name[j];

name[j] = name[i] - name[j];

name[i] -= name[j];

}

p-f("The reverse value is: %s", name);

}

0	1	2	3	4	5
A	B	C	D	E	l.

E D B A

name[i] = name[i] + name[j]

j = 5 - 2

i = 0

X 2

65 + 69

= 134

name[j] = name[i] - name[i]

= 134 - 69

65

name[i] = name[i] - name[j];

134 - 65

69.

/* write a prog for reverse the strings of paragraph word by word. */

#include <stdio.h>

#include <conio.h>

#include <string.h>

main()

{ char str[25];

int i, j, col = 0; l;

puts("Enter the string");

fflush(stdin);

gets(str);

```

    l = strlen(st);
    for(i=0; i<l; i++)
    {
        if(st[i] == ' ' || st[i] == '\0')
        {
            for(j=i-1; j>cut-1; j--)
                printf("%c", st[j]);
            printf(" ");
            cut = i;
        }
    }
    getch();
}

```

* Program for convert integer number decimal no to Binary, octal, Hexa
 * itoa() - (integer to string) format by using these functions.

atoi() (string to integer).

This 'i to a' function will accept any
integer decimal number & store into string type

#include <stdio.h>

#include <conio.h>

#include <stdlib.h>

void main()

{ int n;

char st[40];

clrscr();

textmode(0);

p.f(" Enter Any decimal Number: ");

s.f(" %d ", &n);

itoa(n, st, 2);

p.f(" Binary number for given number : %s ", st);

iota(n, sf, 8);

p-f() In octal number for a given number: (%s, sf);

iota(n, sf, 16);

p-f() In Hexadecimal number for a given number: (%s, sf);

getch();

}

→ Write a Program accept a String of Paragraph Count each character how many time occur in the paragraph.

Format of o/p:

Enter the String : This is a123\$;'. taj

Char — Fre.

:

4

,

1

.

1

1

1

3

1

;

1

a

2

h

1

t

2

j

1

s

2

t

2

0 1 2 - 65 100 - - - 150 - - - 255

9 0 0 0 6 1 0 0 0 0 0 0 0 0 0

#include <stdio.h>

#include <conio.h>

void main()

{ char st[80];

static int a[256], i;

clrscr();

printf("Enter the String: "); AA is

gets(st);

for (i=0; st[i]!='\0'; i++)

{

a[st[i]]+=1;

p-f(" Char (%c) - Fre. %d");

for (i=0; i<256; i++)

{

if(a[i]==0)

p-f(" (%c) (%d) ");

getch();

defn static int a[256];

a(st[i]) = a(st[i]) + 1

= a(st[0]) + 1

= a(65) + 1

= a(65)

o+(=1

→ Write a program accept a string of Paragraph delete a word from a given String.

```

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

void main()
{
    char sf[80], dw[20], wd[20];
    int i, j, K, id=0, CK=0;
    clrscr();
    p: ftl° Enter the string: ";
    gets(sf);
    p: ftl° Enter the word to delete: ";
    gets(dw);
    for(i=0; i< strlen(sf); i++)
    {
        if (sf[i] == 32 || sf[i] == '\0')
        {
            word[fd]:
            wd[i] = '\0';
            if (strcmp(dw, wd) == 0)
            {
                CK=1;
                if (sf[i] == '\0')
                    sf[i - strlen(wd)] = '\0';
                else
                    sf[i - strlen(wd)] = sf[i];
            }
            else
                sf[i] = sf[K];
            K++;
        }
    }
    sf[i] = '\0';
    id = 0;
}

```

} else

{

wd[i][j] = sf[i][j];

} id++;

{

if (c[K] == 0)

p->st" word not found");

else

p->st* Given String after deletion: y. s", sf);
getch();

{

O/P: → A class is a Common relationship & behaviour
Enter the deleted word: Common
Given String after deletion: a class is a relationship & behaviour.

 Write a program accept a string find out the given string is Palindrome or not.
by passing the for the function as a parameter.

#include <stdio.h>

<Conio.h>

int palindrome(stc[Char sf])

{ int i, j, len=0;

while (sf[i][len] != '0')

{ len++;

}

for (i=0; i<len-1; i++, j--)

{ if (sf[i][j] != sf[j][i])
return 0;

}

return 1;

}

Void main()

{

True
False
0 = True
1 = False

```

    char &ft[20];
    int K;
    clrscr();
    printf("Enter the String:");
    gets(sta);
    K = Palindrome - sta[ft];
    if(K == 0)
        printf("Given String is not a palindrome");
    else
        printf("Given String is a palindrome");
    getch();
}

```

→ Write a prog accept a String of Paragraph & also accept 2 characters
Replace the first character with the Second character. implement
by passing as a function parameter & write the logic with pointer.

#include<stdio.h>

void strreplace(char ft, char, char);

void main()

{ char sta[100];

char ch1, ch2;

printf("Enter the String:");

fflush(stdin);

gets(sta);

printf("Enter the character1:");

fflush(stdin);

gets(ch1);

ch1 = getch();

printf("Enter the second character:");

fflush(stdin);

ch2 = getch();

```
8. strreplace (str, ch1, ch2);  
puts ("After replacement...lu");  
puts (str);
```

? void strreplace (char *pt1, char p1, char p2)
{
 for (; *pt1; pt1++)
 {
 if (*pt1 == p1)
 *pt1 = p2;
 }
}

→ Write a program accept StringCopy functionality. Working with pointers.

```
#include <stdio.h>  
void strkopy (char1, char2);  
void main()  
{  
    char s1[100], s2[100];  
    puts ("Enter the strings:");  
    fflush (stdin);  
    gets (s1);  
    strkopy (&s2) (s2, s1);  
    puts ("After copying ... lu");  
    puts (s2);  
}
```

internally convert -

while (*T++ = *S++)
{
 // compiler convert
 // with body of for
 // check it

```
void strkopy (char *T, char *S)  
{  
    for (*S; S++, T++)  
    {  
        *T = *S;  
        *T = '\0';  
    }  
}
```

'\0' = null character not copied

while (*T++ = *S++) → S will be like a

→ void strcat(char *t, char *s) // String Concatenation logic.
 {

 for(; *t; t++);

 while (*t++ = *s++)

}

→ Write a program accept a string replace the string of characters from lower case to upper case, & uppercase to lowercase using functions, arrays, pointers.

void struplw(char []);

void main()

{ char str[100];

gets(str);

struplw(str);

puts(str);

}

voidstruplw(char *ptr)

{

 for(; *ptr;) ptr++;

 if(*ptr >= 65 & & *ptr <= 90)

 *ptr += 32;

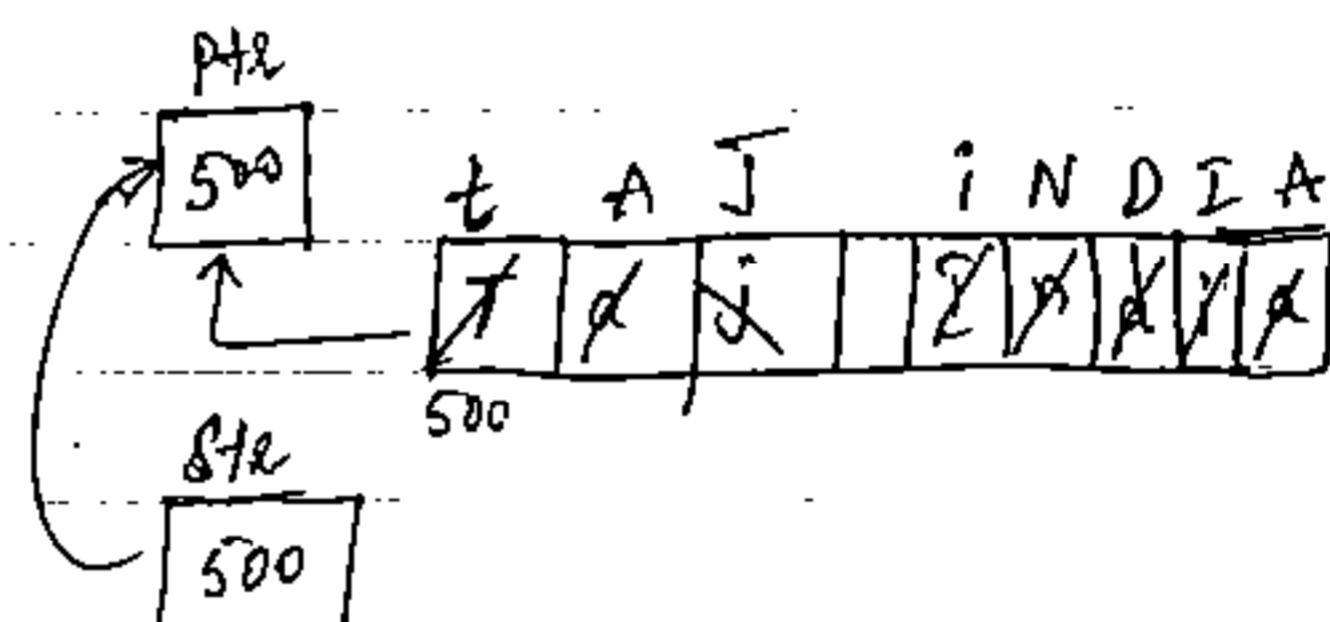
 else

 if(*ptr >= 97 & & *ptr <= 122)

 *ptr -= 32;

}

→ ~~white~~



** Write a program on Password logic. (without backspace)

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

```
#include <string.h>
```

```
{
```

```
void password(char *);
```

```
char str[50];
```

```
int a, b;
```

```
clrscr();
```

```
printf("Enter the password:");
```

```
password(str); /* caller */
```

```
if (strcmp(str, "Kiran") != 0)
```

```
{
```

```
printf("The password is not correct");
```

```
getch();
```

```
exit(1);
```

```
}
```

```
/* Any other logic execute the logic is correct */
```

```
a=5, b=10;
```

```
printf("n %d; a+b);
```

```
getch();
```

```
}
```

```
void password(char * ptr) /* callee */
```

```
{ while(1)
```

```
{ *ptr = getch();
```

```
if (*ptr == '\r')
```

```
{ *ptr = '\n';
```

```
break;
```

```
}
```

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

```
ptr;
```

```
}
```

```
}
```


THE UNIVERSITY OF TORONTO LIBRARIES
UNIVERSITY OF TORONTO LIBRARY

Program on String Comparison: (Strcmp)

154

After Main()

```
{ char a[40], b[40];  
    int ans;
```

```
int Stringcmp( char *a, char *b);  
char(a);
```

printf("Enter the Two strings:"); Here comparison is done
gets(a); through ASCII values.

gets(b);

ans = Stringcmp(a,b);

Convert the strings into uppercase

if (ans > 0)

& Compare 2 strings ASCII value

```
printf("1st > 2nd string\n");
```

1st string 1st ASCII value >

else

2nd string 1st ASCII value - . . .

```
printf("1st < 2nd string\n");
```

else

```
printf("String's are Same\n");
```

getchar();

}

```
int Stringcmp( char *p1, char *p2)
```

{

```
while (toupper(*p1) == toupper(*p2))
```

```
{ if (*p1 == '\0')
```

break;

p1++;

p2++;

}

```
return (toupper(*p1) - toupper(*p2));
```

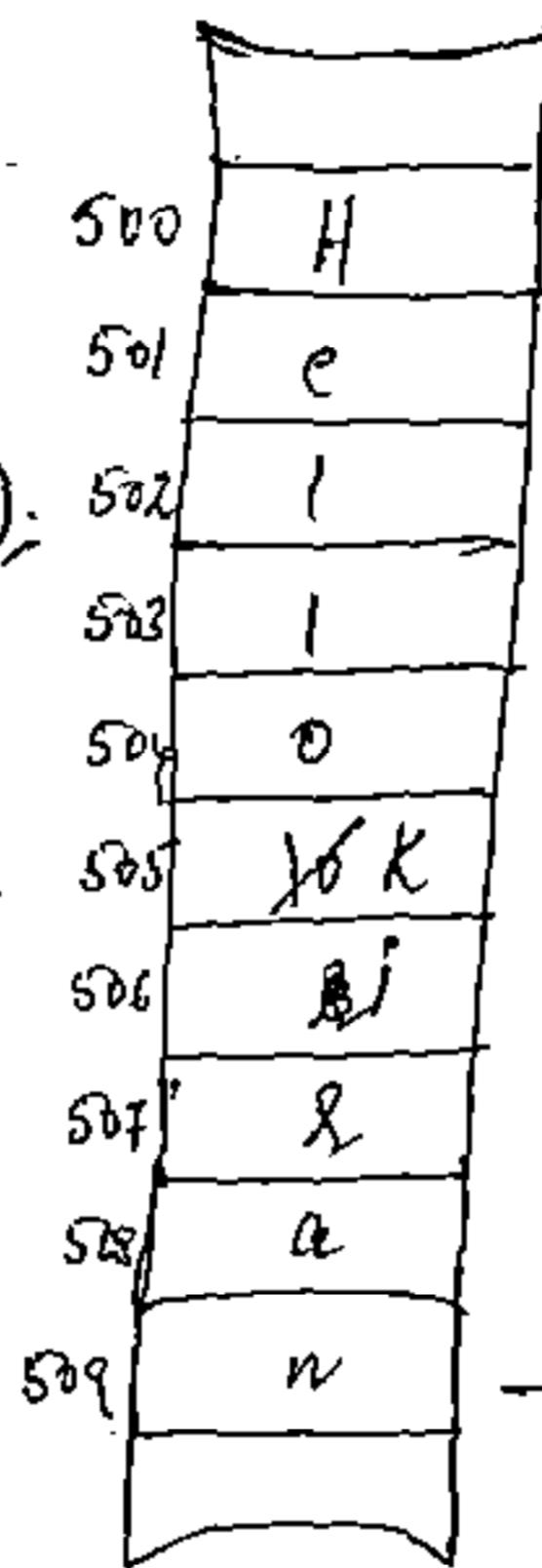
}

String with double dimensions.

To store some collection of names we have to accept an double dimensional array. So suppose if we want to store 10 names, each name of 25 characters, we have to define
 Char name[10][25];

→ void Main()

```
{
    char str[2][15];
    puts(str[0]); G.v's
    puts(str[1]); G.v's
    strcpy(str[0], "Hello");
    puts(str[0]); Hello
    puts(str[1]); Nothing.
    strcpy(str[1], "Kiran");
    puts(str[0]); Hello
    puts(str[1]); Kiran
}
```



2 names each name contains 5 characters so totally 10 characters.

before copy 1st & 2nd prints contains Garbage value. but after 1st print copy it ends with \0 after \0 Garbage values are removed & nothing is displayed

→ Increase the row, names will be successive.

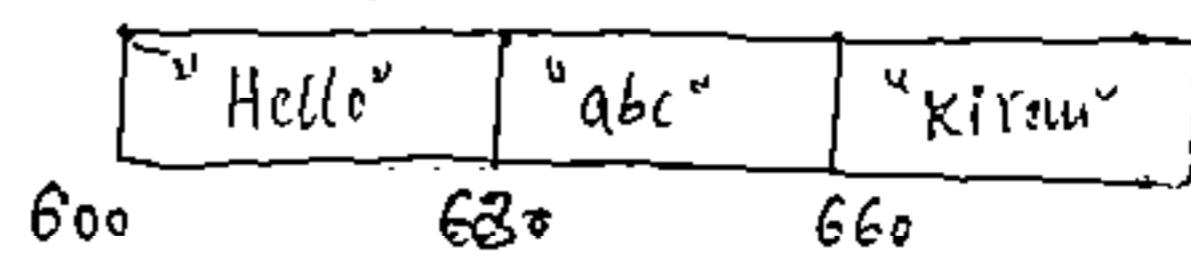
→ In order to store the Null characters & to display the O/P properly, define large size of arrays are implemented by pointers.

→ void Main()

```
{
    char s1[3][30] = {"Hello", "abc", "Kiran"};
    char s2[3] = {"Cobol", "Pascal", "Java"};
```

→ Size specification stored sequentially,

3 rows each name 30 characters



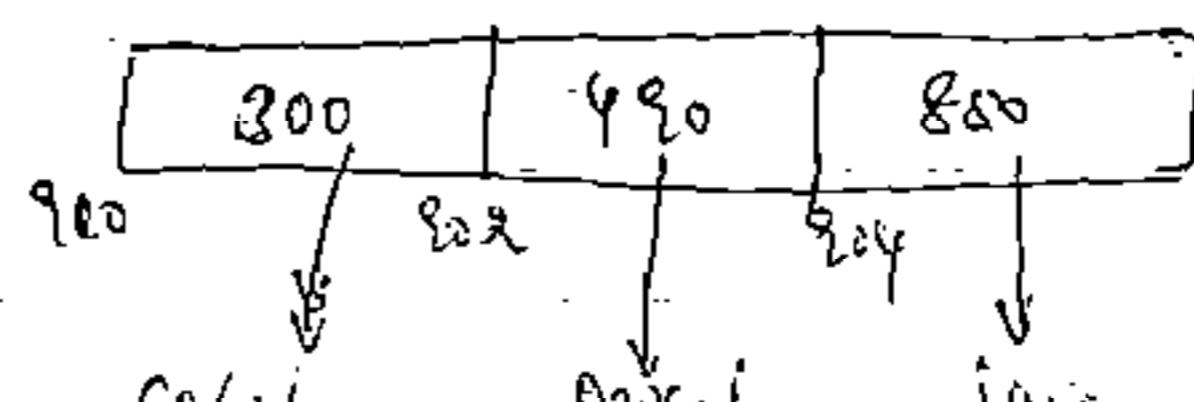
for(i=0; i<3; i++)

puts(s1[i]);

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

puts(s2[i]);

}



Cobol stored at
--- location

and it will be randomly filled.

Note: Array will store the data in sequential memory locations.
Pointer will store in the random locations.

→ Main()
{
 char

 Static char *s[] = {"Ice", "Green", "Cone", "Please"};

 Static char * *ptr[] = {s+3, s+2, s+1, s};

 char *p = p+2;

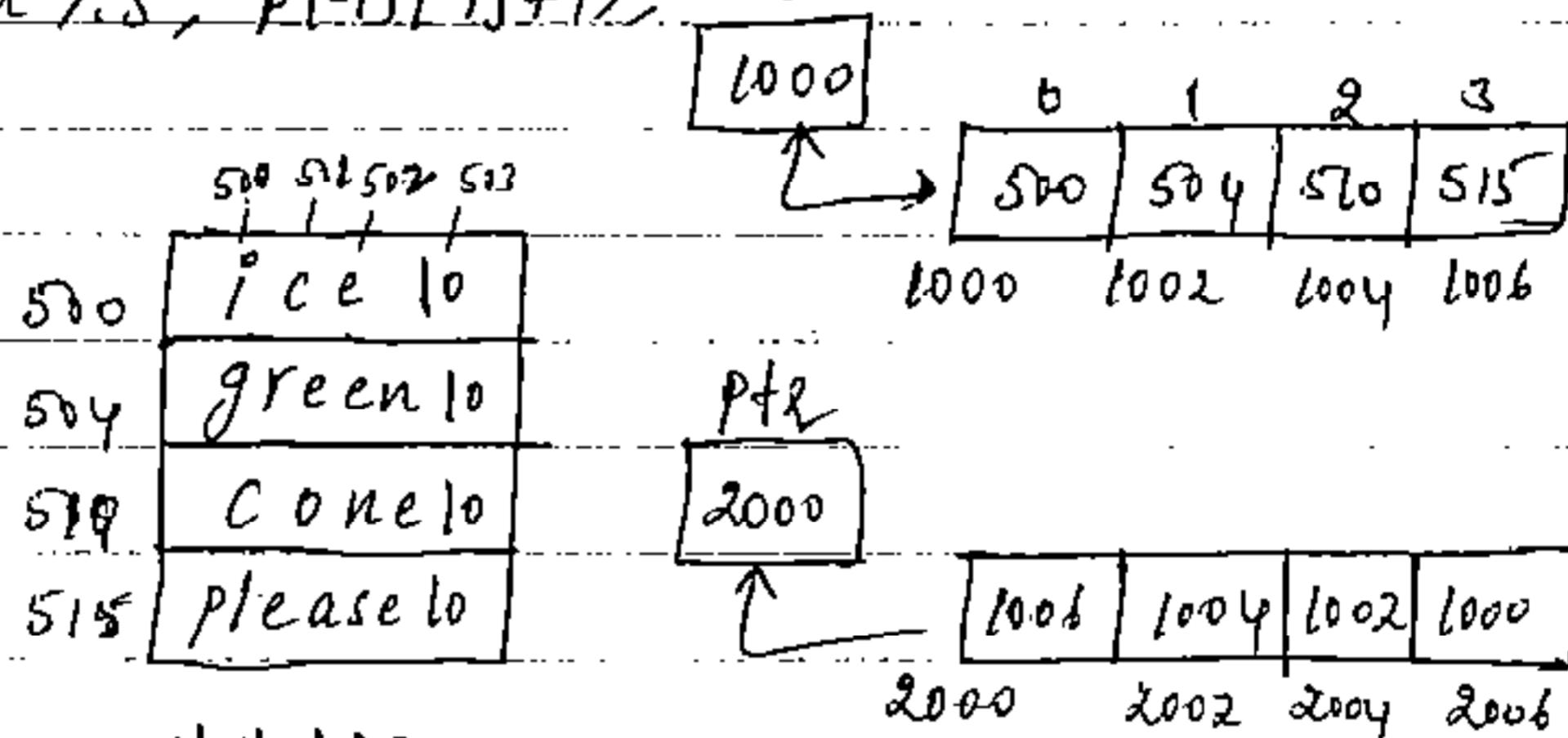
 printf("%u\n", *p);

 p = f("u\n", *p+1);

 p = f("n\n", *p-2)+3;

 p = f("u\n", p[-1]-1)+1);

}



* * + p

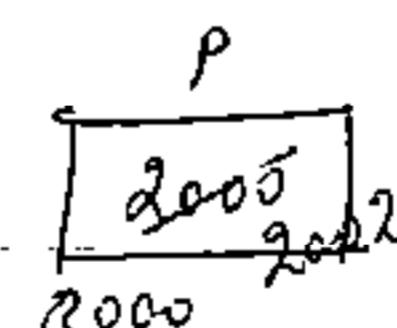
p = 2000

p = 2002

* (2002)

* (2004)

* 520 = CONE



#include <stdio.h>

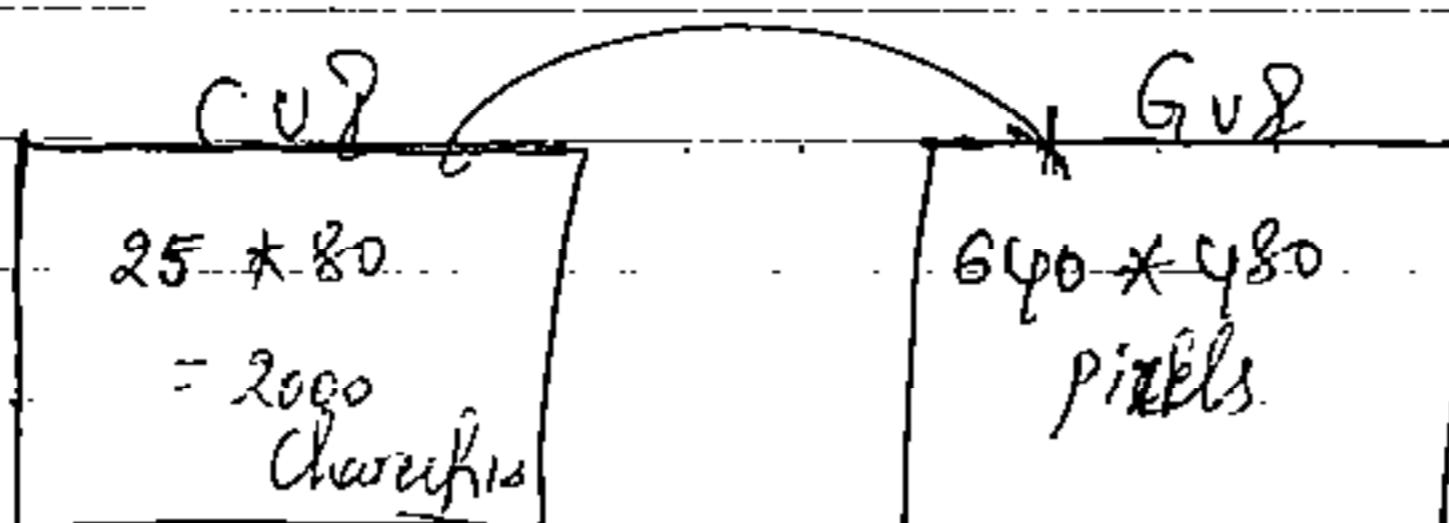
-void main()

{

cout <Color> BLACK, WHITE.

GRAPHICS

156



To convert into graphics environment first initiate the graph by graphic drivers. That graphic drivers will show you what type of drivers is present in your system.

- the Graphic mode will convert from CUI to GUI
- the double quotes " indicates the path of the file
- there is no need to mention, by default it will be taken as place.

```

#include <graphics.h>
void main()
{
    int gd = DETECT, gm;
    initgraph(&gd, &gm);
    setgraph(&gd, &gm, " ");
    printf(" %d %d ", getmaxx(), getmaxy());
    getch();
}
  
```

~~gm~~: graphic mode
~~gd~~: graphic driver
This program used for finding our system supported graphic driver & respected pixel size.

O/P: 639 479

```

→ #include <graphics.h>
Void main()
{
    int gd = DETECT, gm;
    initgraph(&gd, &gm, " ");
    Setbkcolor(2);
    SetColor(4);
    line(0, 0, 639, 479);
    Leeangle(15, 175, 175, 15);
}
  
```

~~Setbkcolor(2);~~
~~SetColor(4);~~
ClearDevice();
Circle(300, 300, 50);
getch();
}

→ { setTextStyle(GOTHIC) \downarrow f_{out} → Direction
setTextStyle(GOTHIC-FONT, HORIZ-DIR, &) ; site

outTextXY(150, 100, "Kiran");
closegraph(); // it will come out from G02 to C02
getchar;

→ #include <graphics.h>
#include <sfalib.h>
Main()
{
int gd = DETECT, gm = 0, i;
initgraph(&gd, &gm, "");
setTextColor(4);
while (!kbhit())
{
for (i = 0; i <= 100; i++)
{
setcolor(random(16));
Circle(300, 300, i);
delay(10);
}
}

FILES

157

Many applications wants to implement to store the data on the auxiliary storage devices (like hard disk, pendrives, etc).

To store the data for implementing to lead (or) written of information is stored on memory devices in a form of data files.

To implement this operations we have so many I/O library functions. There are divide in 3 categories.

1. Console I/O functions.

Functions to receive I/O from Keyboard & write O/P to VDU (visual display unit, or) monitor.

2. Disk I/O functions.

Functions to perform I/O operations on a hard disk, pendrives, etc.

3. Port I/O functions.

→ Console oriented applications always use the terminal as a Keyboard & a screen as the target places.

→ This works fine as long as the data is small.

→ however many real life problems has large amount of data, In that situations, the console I/O applications mainly has two providers.

1. If becomes cumbersome & time consuming to handle large volumes of data through terminals.

2. The entire data is lost, when either the program is terminated (or) the computer turned off.

It is therefore necessary to have a more flexible approach where data can be stored on disks & lead whenever necessary, without destroying the data. This method employs the concept of FILE to store data.

def: file:

- A file is a place on the disk, where a group of related data is stored.

(Q8)

- A file is a collection of related data stored in a particular area on disk.

The file operations of disk I/O is implemented in 2 ways to perform the file operations in 'C' language.

1. Low level I/O and uses UNIX system calls.
(C system-oriented)
2. High-level I/O (Stream-oriented)

High-level I/O more commonly used in 'C' programs.
Since they are easier to use than low-level disk I/O functions.

→ There is one diff b/w low level & high level disk I/O functions. High level disk I/O functions do their own buffer management.

where as in low level disk I/O functions buffer management is done explicitly by the programmers.

→ Stream oriented I/O are divided in to 2 types.

1. Text files.

2. Unformatted data files.

→ The Text files contains consecutive characters.

→ The Unformatted data files will be define block of data
they will be interpreted by using predefined library
functions.

→ System oriented data files are closely related to the operating system.

→ They are much completed but more efficient, for implementation of the programming.

→ Any type of operation should perform also it is compulsory to opening & closing of the files.

Opening & Closing of data files:

The 1st step to opening a file of a Stream oriented data files is to establish a buffer area, the buffer area is a temporary storage which is used while the information is being transferred b/w the computer memory & the data files.

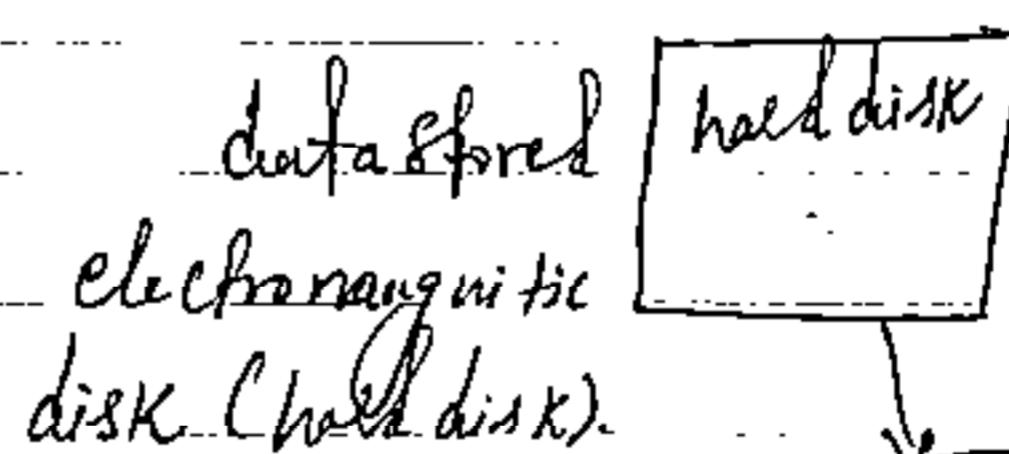
The buffer area allows information to be read @ written to the data files.

Establishing the buffer area:-

FILE *file - pointer

Ex:-

FILE *fp;



We are not directly retrieve the data from that it takes lot of time.

So by using file concept 1st that data stored in on buffer area that buffer are contains data will be used.

by using FILE Concept we can refine info buffer.

If is built-in structure which can include the variables, which can refer the attributes of the file.

If is used to represents file pointer object.

→ file is a special structure type that establish buffer area.

A data file must be opened before it can be created (or) processed this associates the filename with the buffer area.

If specifies how data file will utilized.

fp = fopen(filename, filetype);

Fopen is used to open a file, filename, of the data file and the way in which data file will be utilized.

FILE

```
typedef struct
{
    short      level;
    unsigned   flags;
    char       fd;
    unsigned char hold;
    short      bsize;
    unsigned char *buffer, *cusp;
    unsigned   istemp;
    short      token;
}
```

FILE;

They are 9 member elements & 16 bytes is the size.

→ When we defining a file to open we have to specify the modes of the files, that indicate what type of operations.

We are performing on a file. There are different types of File modes are available, some of the file modes

1. 'r' mode:

'r' Searches file. If the file exists loaded it into memory & sets up a pointer which points to first character init. If file doesn't exist, it returns null. Operations possible - Reading from a file.

2. 'w' mode:

'w' Searches file, if the file exists if the contents are overwritten. If the file doesn't exist new file is created. Returns null if unable to open a file. Operations possible - Writing to a file.

3. 'a' mode: (append)

'a' Searches file, if file exists loaded into memory & sets up a pointer to points the first character. If file doesn't exist new file is created. Returns null, if unable to open a file. Operations possible - Appending new contents at end of the file.

4. 'r+' mode:

'r+' Searches a file, if it exists loaded into memory. Sets up a pointer which points to first character init. If file doesn't exist it returns null. Operations possible - reading, writing, modifying existing contents.

→ Some other modes are w+, a+, a+b, a+b+, etc

'b' stands for binary

Functions:

When we implementing file there are predefined functions are available.

fclose(), fgets(), fread(), fwrite(), feof(), fopen(),
fscanf(), remove(), getc(), printf(),
fseek(), rename(), getch(), putc(), tell().

// Constants Data types & Global Variable

- 1. EOF
- 2. SEEK_CUR
- 3. FILE
- 4. SEEK-END
- 5. NULL
- 6. SEEK_SET

→ Write a program accept a file Read the contents of a file, display the contents.

#include <stdio.h>

Void main()

{

FILE *fp;

Char fname[20], ch;

puts(" Enter the filename");

gets(fname);

fp = fopen(fname, "r");

if (fp == NULL)

{

printf(" file Not found ");

getch();

exit(0);

}

while (1)

{

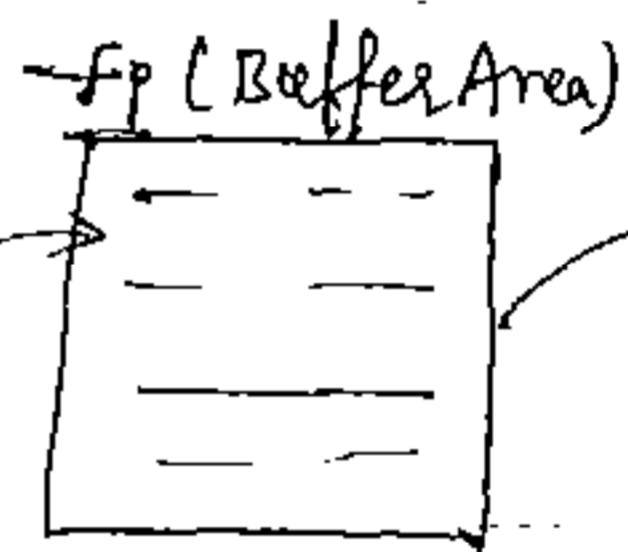
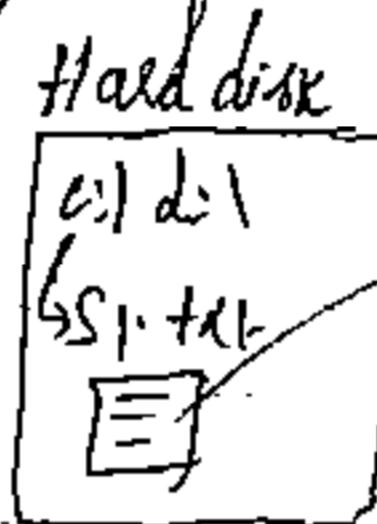
ch = getc(fp);

if (ch == EOF)

break;

putchar(ch);

}



fclose(fp);

printf(" In File opensuccessfully ");

getch();

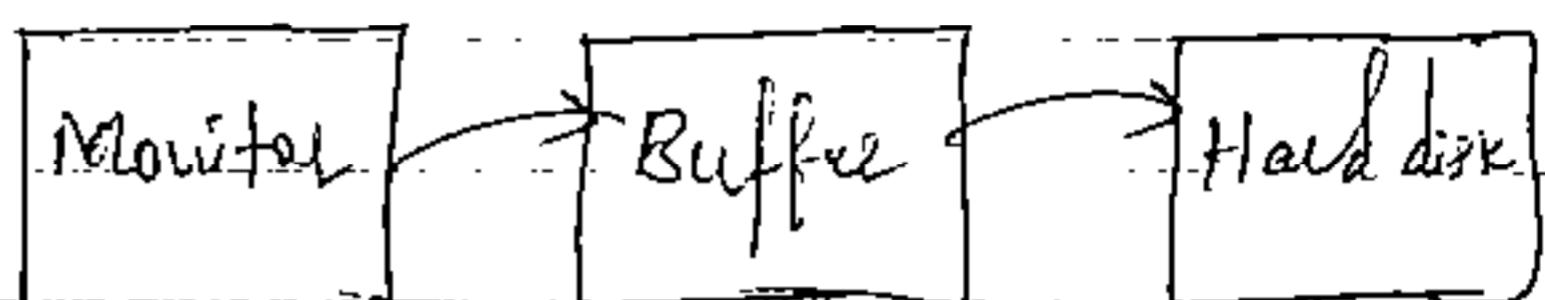
}

getc() function returns
at a time one character
from buffer.

→ Every time of loop iteration in the above program every character is changing by the `getchar` function.

→ Write a program to ~~read~~ write the contents onto the file.

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



```
FILE *fp;
```

```
char fname[20], ch;
```

```
pals("Enter the file Name:");
```

```
gets(fname);
```

```
fp = fopen(fname, "w");
```

printf("Enter the contents onto the file (%s) file will use
Press Ctr+Z...ln");

```
while(1)
```

```
{ ch = getchar(); // Initially the data is copied into
    if (ch == EOF) // buffer & after completion of buffer
        break; // whenever fclose() function executed
    fputc(ch, fp); // file will be closed & total data will be
} // updated into the file.
```

```
fclose(fp);
```

printf("File Created Successfully...");

```
getchar();
```

fputc() also loading one, one character
onto buffer.

fputc(): If redirects a ^{single} character to a standard file
fputc(file, char*);

fgetc()

Reads a single character into a target variable from a specified file

<char> = fgetc(file #);

Appending the data on to the file:

To append the data onto the file we can use append mode. always the append mode will add the data from end to the end of the file.

fp = fopen(fname, "a");
 |
 | Append

Deleting a file:

To delete a file use the function
remove(fname); It makes to delete the file.

main()

{

FILE *fp;

Char fname[20];

clrscr();

puts("Enter the file to Delete:");

if (fp == NULL)

gets(fname);

fp = fopen(fname, "r");

if (fp == NULL)

{ puts("File Not found");

getch();

exit(1);

fclose(fp);

```
curlink(&fname);
fp = fopen(fname, "r");
if fp == NULL
    puts("File Deleted Successfully...!!!");
fclose(fp);
getch();
```

→ Write a program to copy the contents of one file into another file.

→ Write a prog accept a file rename the file name
↳ Using onclick

~~=printf()~~

`int -fprinf(-fp, format, s)`

The call `sprintf()` places output on the named output to which the file pointer `fp` points. `S` represents the arguments whose values are printed.

format is the female Specific String - the formal
conventions of print() works exactly sprintf().

Scanf()

scanf: It is used to redirect C to accept the data from the Standard file into given Variables.

```
fscanf(file, "%d", &Variable);
```

Random accessing of Files:

A file may be accessed Sequential (or) Randomly. In a Sequential access all the preceding data is accessed before accessing a specific portion of a file.

Random access permits direct access to a specific portion of a file.

fseek(), ftell(), and rewind()

fseek():

flag = fseek(file-pointer, offset, from-where);

↓ ↓ ↓
address starting Ending

- int flag is value returned by the function fseek(). 0 if successfully and 1 if unsuccessfully.
- fseek() sets the position of the next I/O (or) output operation in file to which the file pointer fp points to.
- The new position is at the signed distance offset bytes.
- The third parameter moves from the beginning from the current position (or) from the end of the file depending upon the value of the ptename
- If the 3rd parameter is + the 2nd parameter specifies an increment (or) decrement to the current position of the file.

fseek(fp, fl, 0);

position of the file associated with 'fp' of the eight character of the file (remember the first character is at position 0)

(0) SEEK_SET → beginning position

(1) SEEK_CUR → Current position

(2) SEEK_END → Ending position.

fseek(fp, 3L, 1);

162

→ Skips a head 3 characters in the file.

fseek(fp, -3L, 1);

→ move back Three characters (-' higher means move back)

→ In emp table we need to define 2nd record of this
One we use fseek() function.

Struct Emp

{ int eno;

char ename[20];

float esal;

}; // 26

fseek(fp, (1d-1)*sizeof(Emp), SEEK_SET);

1d = 3

fp = 1st record

→ If in emp table there are 'n' records, then to define
the last record.

fseek(fp, - sizeof(Emp), SEEK_END);

'-' Move back.

→ long tell(FILE *fp);

The function tell() returns the current
value of the file position indicator associated with fp!

~~READ~~

fread() and fwrite():

The drawback of fprintf() & fscanf()

Emp;

sprintf(fp, "%d.%s.%d", e.id, e.name, sal); // OK

We can write this buf if it is not recommandable.

→

In This situation if you want to make any modification

Then we need to Create a temporary file & copy this one & make

typedef:

~~typedef provides alias names for the existing data types~~

changes & once again copied into this one.

fwrite()

If is used to redirect the complete object of a structure into the binary file.

`fwrite(<&object>, size, 1, file *);`

→ fwrite(&e, sizeof(e), 1, fp);

1 indicates no. of records

e - source, fp - target.

→ To achieving a uniqueness of a memory allocation
then we go for fwrite() function.

fread()

If is used to read the data from the binary file into a structure object with a known size.

`fread(<&obj>, size, 1, file *);`

COMMAND LINE ARGUMENTS

163

a command is something but it performs a predefined task in which the .exe file is converted into commands, by passing some parameters to the main().

Def: CLA:

It is a parameter supplied to a program when the program is invoked. These are the arguments passed to the at Command prompt.

Explanation:

Main() function in 'C' takes two arguments called argc, argv. The information contained in the command arguments when main() is called up by the Stream.

argc: (argument count)

The variable argc is an argument count that counts the no. of arguments on the Command line.

argv: (argument refer)

The argv is an argument refer and represents an array of character pointers that point to the Command line arguments. The size of the array will be equal to argc.

Inf: main(int argc, char argv[])

(or)

main(int argc, char **av)

→ Write a Prog create the logic of calculator by the Command line args

#include <stdio.h>

inf: main(int argc, char *argv[])

{

inf: n1, n2;

char op;

if(argc < 4)

{

p-f("Syntax of the Command is incorrect, pass 4 arguments");

exit(0);

}

'arg 0' always contains .exe file

op = argv[2][0];

n1 = atoi(argv[1]);

n2 = atoi(argv[2]);

switch(op)

{

case '+': printf("n %d", n1 + n2); break;

case '-': printf("n %d", n1 - n2); break;

case '*': printf("n %d", n1 * n2); break;

case '/': printf("n %d", n1 / n2); break;

case '%': printf("n %d", n1 % n2); break;

default: printf("n Invalid operator");

}

}

Like this every program is converted

info .exe &

→ Save the file with .c extension & generate the .exe file.

go to the Command prompt, Using dos shell (a) cmd.

E:\Kalc. T.18

15 → Like this every program converted info

From the Editor also we run the .exe & generate a command.

.exe file. go to Turbo C → Run → Arguments. So we want to generate

One our own DOS operating system.

typedef:

typedef provides the alias name for the existing datatype.

Syntax:

typedef existing datatype-name aliasname;

Main()

{

typedef int identity;

typedef float decimal;

typedef char* string;

identity a = 0;

decimal k;

String S1, S2, S3 = "abc";

printf("%d %d %d", a, sizeof(k)); 10 4

printf("%s\n", S3); abc

printf(S3); abc

printf("%u %d %d", sizeof(S1), sizeof(S3)); 2 2

}

→ #define CP1 char*

typedef char* String

void main()

{ CP1, P1, P2, P3; // it is replaced with char*

String P4, P5, P6; // it is alias names to char*

}

char*

char

#define P1, P4, P5, P6

P2, P3

but #define only left side
convert the whole

In #define macro only left side

convert the whole replaced by regular size. Not convert all

CP1 replaced by char*

char*, P1, P2, P3;

char* P1. only P1 is replaced with String.

because Macro will over replacement. In typedef.

STRUCTURES

As we know that array is an homogenous type. That is used to represent group of data items, that belongs to the same type. However if you want to represent collection of data items of diff types using a single name, then we can't use arrays. at this time we use structures.

Structure used to represent set of attributes of such that is mixe of diff types.

Ex:- the attributes are member elements like train no, train Name, PNR No, etc are defined as member elements for the structure called as railway ticket.

The structure member elements are implemented by integers, float, pointers, arrays, another structures as a member elements.

For all these member elements will define one name that Name is called Structure name (or) user defined datatype.

By using structures we can create user defined datatype.

The userdefined datatype can be also called as tag name.

Struct <Structure name>

{ → tag name (or) userdefined datatype.

Member Element 1

member element 2

=====

} ; member element n

Structure always enclosed with Semicolon(;)

#define N3 → datatype (3) tag-name
 typedef struct employee
 {

union bit delbit; // bit is enum

int no;

char name[15];

department d; // enum

float sal;

date dob; // Structure

} Employee; → alias-name
 use defined type

Void main() → Variable Name.

{ employee emp[N];

Employee ← employee accept data();

void display-data(Employee);

int findEmployee(Employee[], int)

}

The Structure member elements can be access by using
 Member access operator: → (dot) & → pointer to member access
 operator: →

Syntax for accessing variables:

use defined datatype Val-name1, Val-name2, ...;

Ex: Employee e1, e2, e3;

datatype ↑ Variables

To access the Member elements:

Val-name · mem-ele = <initialization>

Write a prog accept a structure & also accept a member element data
 display all the data.

{ Struct-Structure

int sid;

char * Sname;

float fees;

} Member Elements

Main()

```
{  
    struct student s;  
    printf("Size of Structure is %d\n", sizeof(s));  
    printf("Enter the elements... %u");  
    scanf("%d %s %f", &s.sid, s.sname, s.fees);  
    printf("The Elements are... %u");  
    printf("%d %s %f", s.sid, s.sname, s.fees);  
    getch();  
}
```

Note:- The size of the Structure is the total member elements present in the Structure.

→ a Structure can be defined in diff ways of representations -

Pattern 1

```
struct emp  
{  
    int id;  
    char name[30];  
    int sal;  
};
```

```
void Main()  
{  
    emp e1, e2, e3; // invalid inc (Valid in C++)  
    struct emp e1, e2, e3; // valid  
    struct emp e4 = {4, "xyz", 4500}; // valid  
    e1.id = 1;  
    e1ENAME = "abc";  
    e1.name = abc; // invalid
```

e2 = e1; // e1 data copies into e2. (valid)

e1.Sal > e2.Sal;
// e1 > e2; invalid

pattern 2

Struct employee

166

int id;

char name[20];

int sal;

{ Struct alias name.

typedef struct employee emp;

void main()

{ struct employee e1, e2, e3; // yes

{ emp e4, e5, e6; // yes. without using struct also possible.

pattern 3

// Defining global variables.

Struct emp

{ int id;

char name[20];

int sal;

{ e1, e2 = {2, "xyz", 1200}, e3; // yes

// e1, e2, e3 are become global variables.

Void main()

{ Struct emp e4, e5; // yes // e4, e5 are local variables.

{

// Nameless Structures.

Struct

Void main()

{ int id;

{

char name[20];

struct e4, e5; // no

int sal;

{ // Not possible to create local Variables

{ e1, e2, e3; // yes

because No Name of Structure.

→ `typedef struct`
{
 int id;
 char name[20];
 float sal;
} EMP; // yes // Emp becomes alias name to the Structure.

`void main()`
{
 EMP e1, e2, e3; // yes.
}

// define inside main()

`void main()`
{
 typedef struct
 {
 int id;
 char name[20];
 float sal;
 } EMP; // yes.
}

`void abc()`
{

 EMP e1, e2, e3; // no - Since Structure become local to Main().

} // Array of Structures.

`typedef struct employee`
{
 int empno;
 char ename[20];
 float sal;
} emp;

↳ alias name

```
void main()
```

```
{    emp e[5] ; int i;
```

```
printf ("Size of Structure %d, Size of c[%d], Size of Cmp), Size of (e);
```

// Accepting

```
for (i=0; i<3; i++)
```

```
{
```

```
    printf ("Enter the employee details of %d\n", i+1);
```

```
    scanf ("%d %s %f", &e[i].eno, e[i].ename, &e[i].esal);
```

// Display

```
for (i=0; i<3; i++)
```

```
{
```

```
    printf ("The employee details are ", i+1);
```

```
    printf ("%d %s %f\n", e[i].eno, e[i].ename, e[i].esal);
```

```
}
```

```
link_float()
```

```
{    float a, *b;
```

```
    b = &a;
```

```
    a = *b;
```

```
}
```

→ There are some cases in which the reference to the float is a bit obscure and the compiler doesn't detect the need for emulator.

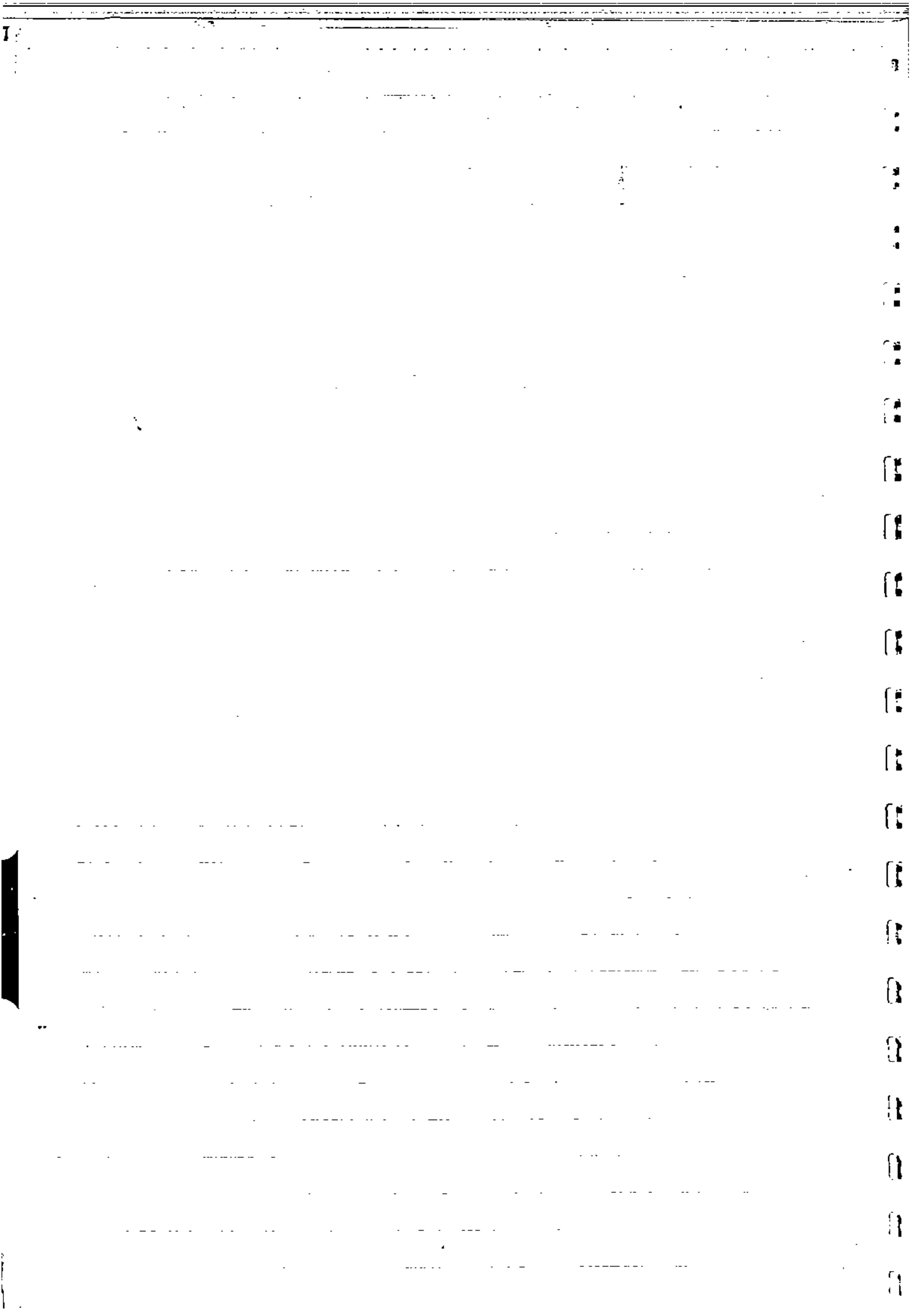
→ The most common is using scanf() to read a float in an array of structure as shown in our program.

→ We can force the format to be linked by the using link_float() function. If forces taking of the floating-point-emulator into an appn. There is no need to call this function just include wherever necessary in our program.

Note: The above program is implemented by array of structures.

There's a wastage of memory block's in order to overcome drawback

We implemented dynamic memory concepts.



Structures within structures

struct X

{ int a;
 int b;
 struct X x1;
};

→ a structures within structures Contains one Structure variable as a member element to another structures.

struct Y

{ int b;

struct X x1;

};

struct Z

{ int c;

struct Y y1;

};

int Main()

{ struct Z z1;

z1.c=30;

z1.y1.b=20;

z1.y1.x1.a=10;

printf(" %d %d %d ", z1.y1.x1.a, z1.y1.b, z1.c);

}

Unions:

Syntax: Union <Union-name>

{ Member-ele1;

Member-ele2;

:

";

Unions are Similar to Structures Concept

Unions also can create user defined datatypes. The Unions Contains diff types of member elements. Every Union application Can be possible to implemented by Structures Concept also.

but Structures applications Can't possible to implement by Unions.

→ The Unions share a Common memory location by 2 or more diff types of Variables.

The basic differences b/w Union's & Go-Structures in Conserving of Memory allocation.

Struct s;

{ int a;

float b;

}

→ The size of the structure is the total member elements.

→ The size of Union is highest Member element size.

→ In one Member element Memory only the element's will store.

In Unions it is not possible to initialise all the member elements at a time, we can initialise one member element-

main()

{ Struct ss1;

union u1;

printf("Size of structure %d lu", sizeof(ss1)); 6

printf ("Size of Union %d lu", sizeof(u1)); 4

}

/* Struct sss,

Struct sss, = { 10, 56, 78 }; // Valid */

/* Union u1, u1 = { 34, 99, 5643 }; // invalid */

union u1;

u1, a = 45;

p.f("Y.d lu", u1, a);

u1, b = 78.786;

p.f("X.f lu", u1, b);

}

How Unions & Structures implementing in Memory.

Struct a,

{ int i;

Char ch[2];

}

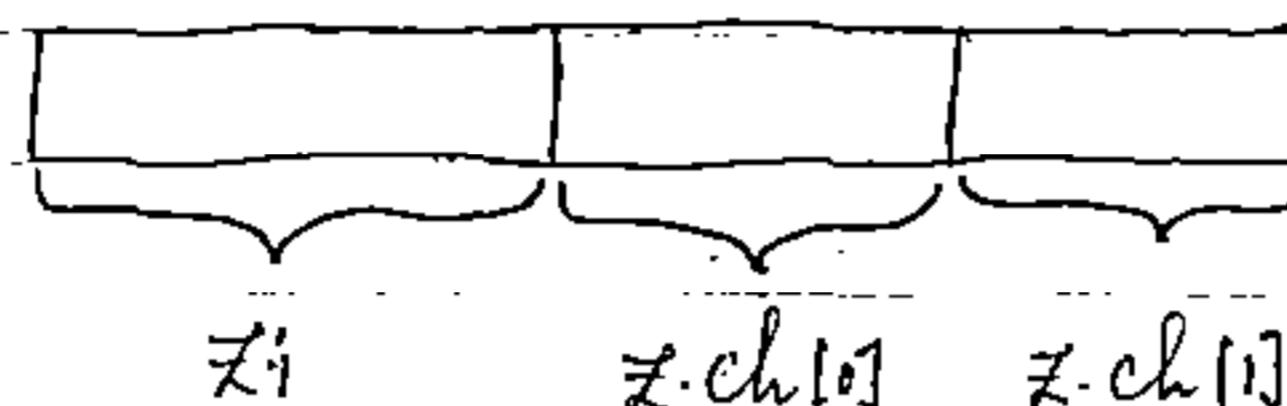
Struct a Z:

Z.i = 512;

printf("%d", sizeof(Z)); 4

printf("%d %d %d", Z.i, Z.ch[0], Z.ch[1]); 512 Garbage value Garbage value.

}



Structure Memory location

Z.i Z.ch[0] Z.ch[1]

Main()

{ union a

{

int i;

char ch[2];

}

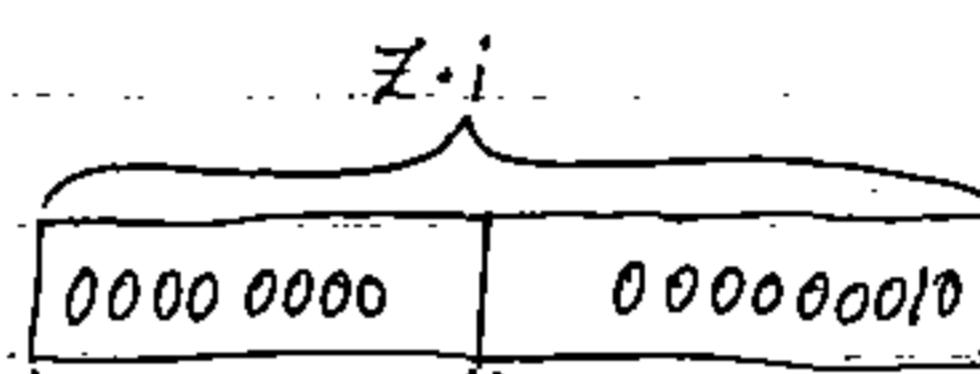
Union a Z;

Z.i = 512;

p.f("%d", sizeof(Z)); 8 2

p.f("%d %d %d", Z.i, Z.ch[0], Z.ch[1]); 512 0 2

}



Union Memory location

Z.ch[0] Z.ch[1] - high

512 = 100000000

→ In this low byte store first and then higher byte then this above architecture called "Little Endian Concept"

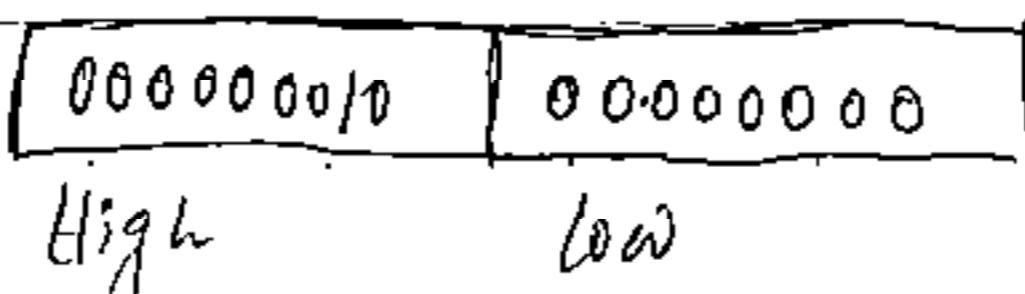
→ This is not for only Union but for every thing like int, float, etc

→ "Same bytes in Multiway we use Union" (the Union are used)

→ Little Endian Maintained by Intel processors.

If high bytes & low bytes it is called "BIG ENDIAN".

Concept is maintained by Motorola processor.



For which type of applications we can implement Union
& it is better than Structure Concept

→ If we store employee information Highly scale HSK
semi scale SSK.

One way to store Structures. Structures wasting
two slots either any two used but not both then
we can store by Unions.

// UTILITY.

Name

Grade

Age

if Grade = HSK

Hobby name

Credit card no

if Grade = SSK

Vehicle no

Distance from Company.

→ Struct info // 12 bytes Struct. emp //

{ char hobby[10]; { char n[10];
 int creditno; char Grade[4];
}; int age;
Struct info 2 // 12 bytes union info f;
{ char VehNo[10]; };
int dist;
};
union info
{ struct info a;
struct info b; // 12 bytes
};

By using Union. Saving memory of 12 bytes
in the above program.

enum (enumeration)

170

enumeration is also similar to Unions & Structures
it is also user defined datatype through enumerations
we can create some user defined datatypes & make the
program more readable, it allows to create your
own datatype with predefined values.

The values i.e. its members are all constants.

enum identifier

{ ele1,

ele2

ele3 = <values> --- ele_n,

}

The Elements what we define they internally
acting as integer type. Through enum we can generate
series of integer constant's. It can be called as alias to
integer

Size of enum is 2 bytes.

Ex:-

enum Bool

{ FALSE,

TRUE,

}

#include < stdio.h >

enum tiffins {

deadlyidly = 5,

dosa = -2,

poor,

rada

}

void main()

{ enum tiffins a = 0;

switch (a)

{

case deadlyidly : printf("Deadlyidly: RS: 12");
break;

case dosa : printf("dosa: RS 15");

break;

case poor : printf("poor: RS 20"); break;

case rada : printf("rada: RS 20"); break;

enum MONTH { JAN=1, FEB, MARCH ... DEC};
enum DAY { nof-day - Jan = 31, no.of-day - feb = 28, nof-day -

// int dd;

enum DAY dd;

enum MONTH mm;

// int mm

switch(mm)

}

Case DEC: dd = nof-days - nov;

// Case 12: dd + 30;

Examples of enum:

#include <stdio.h>
<string.h>

Note:

By Using Enumerations we can
save lot of memory.

void main()

{

enum Color { BLACK, WHITE, GREEN};

enum Color x;

printf("In x.d BLACK", BLACK); 0/1: 0. BLACK

printf("In x.d WHITE", WHITE); 1. WHITE

printf("In x.d GREEN", GREEN); 2. GREEN.

printf("Select a Color! "); Select a Color!

scanf("xd", &x); WHITE selected.

switch(x)

Case BLACK: printf("In x.d BLACK is selected"); break;

Case WHITE: printf("In WHITE is selected"); break;

Case GREEN: printf("In Green is selected"); break;

}

Passing Structure Variables as a function parameter

We can pass Structure Variables also as a function parameters
by default it pass the value of the Variables.

- By using Enumerations we can save lot of memory.
- Write a program accept & display the structure data by passing
 - function parameter.

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

```
#define N 2
```

```
typedef struct
```

```
{
```

```
    int id;
```

```
    char name[15];
```

```
    float sal;
```

```
} Employee;
```

```
Void main()
```

```
{
```

```
    Employee emp[N];
```

```
    Employee emp accept-data();
```

```
    Void display-data(Employee);
```

P.S/ In enter employee details:

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

```
{
```

```
    emp[i] = accept-data();
```

/* Jumps to accept-data() function definition */

```
}
```

P.S/ In displaying the data:

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

```
{
```

```
    display-data(emp[i]); /* Jumps to display-data() function  
definition */
```

```
}
```

/* end-of main */

~~Employee~~

```
Employee accept-data()
```

```
{
```

```
employee emp;
printf("Enter Employee no.:");
scanf("%d", &emp.no);
printf("Enter Employee Name:");
scanf("%s", &emp.name);
printf("Enter employee salary:");
scanf("%f", &emp.sal);
return emp;
```

```
}
```

```
void display-data(employee emp)
```

```
{
```

```
    printf("no: %d", emp.no);
    printf("name: %s", emp.name);
    printf("Salary: %f", emp.sal);
}
```

```
return;
```

```
}
```

Modification:-

```
#include <stdio.h>
typedef struct
{
    int d, m, y;
} date;
typedef struct
{
    int no; char name[15]; float sal;
    date dob;
} employee;
#define N 2
void main()
{
```

```

employee emp[N];
employee accept-name();
void display-data(employee);
int find-employee(employee el[], int)
    int i, n, pos;
printf("Enter the employee details:");
for(i=0; i<N; i++)
{
    emp[i] = accept-data();
}
printf("In display the data:");
for(i=0; i<N; i++)
{
    printf("In employee[%d]: ", i+1);
    display-data(emp[i]);
}
printf("In Enter the employee %s to modify: ");
scanf("%d", &n);
pos = find-employee(emp, n);
if(pos > -1)
{
    printf("Employee found: displaying the data: ");
    display(
        display-data(emp[pos]));
    printf("In Enter new details: ");
    emp[pos] = accept-data();
}
else
{
    printf("Not found"); return;
}
printf("In display the data after modification: ");

```

```

for( i=0; i < N; i++)
{
    p.f("Employee [y,d]:", i+1);
    display-data( emp[i] );
}

/* end of main */

int find-employee(employee e[], int n)
{
    /* definition of find employee */
    int i;
    for( i=0; i < N; i++)
    {
        if( e[i].no == n )
            return i;
        else
            return -1;
    }
}

employee accept-data()
{
    employee e;
    p.f("Enter no: ");
    s.f("%d", &e.no);
    p.f("Enter Name: ");
    s.f("%s", &e.name);
    p.f("Enter Salary: ");
    s.f("%f", &e.salary);
    p.f("Enter date of birth (dd-mm-yy): ");
    s.f("%d-%d-%d", &e.dob.d, &e.dob.m, &e.dob.y);
    return e;
}

void display-data( employee e )
{
}

```

P-f(°) In Employee no: x.d⁴, emp);

P-f(°) In Employee name : Y.S⁴, c.name);

P-f(°) In Employee salary : x.f⁴, e.sal);

P-f(°) In date of birth : Y.d-Y.d-Y.d⁴, e.dob.d, e.dob.m, e.dob.y);
return n;

{

Deleting:

enum department { finance, Marketing, Personnel};

typedef struct date { int d, m, y};

union bit { int FLAG; int TRUE};

typedef struct

union bit delbit;

int no;

char name [15];

enum department d;

float sal;

struct date dob;

} employee;

#include <stdio.h>

#define N 2

void main()

{

employee emp[N];

int i, n, pos;

char display;

employee accept_data();

void display_data(employee);

int find_employee(employee e[1], int),

P-f(°) In Employee details:"");

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

{

emp[i] = accept_data();

{

P-f(°) In display the data:"");

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

{

P-f(°) employee (%d,% , i+1);

display_data(emp[i]);

{

P-f(°) Enter employee no to
delete:"");

S.f("Y.d", 8n);

pos = find_employee(emp, n);

if (pos > -1)

{

```
p-f("In Employee found: display the data : ");
display-data(emp[pos]);
f-f("In delete record (Y/N) : ? ");
flush(stdin);
s-f("Y/C", &replay);
if (replay == 'Y' || replay == 'y')
    emp[pos].delbit.FALSE = 1;
}
else
{
    p-f("In not found : ");
    return;
}
```

```

p-f("In display the data after deletion : ");
for (i=0; i<N; i++)
{
    if (emp[i].delbit.FALSE == 0)
    {
        p-f("In employee[%d] : " i+1);
        display-data(emp[i]);
    }
}
```

```

p-f("In employee - accept : ");
employee accept-data()
```

```

employee e;
e.delbit.FALSE = 0;
p-f("In no : ");
s-f("Y.d", &e.no);
p-f("In name : ");
s-f("Y.s", &e.name);
p-f("In salary : ");
s-f("Y.f", &e.sal);
```

p.f1("In department Y.d. finance Y.d. marketing & personnel";
finance, marketing, personnel);

s.f1("Y.d.", & e.d);

p.f1("In date of birth (dd-mm-yyyy):");

s.f1("Y.d.-Y.d.-Y.d-", & e.dob.d, & e.dob.m, & e.dob.y);
scanf(e);

}

void display_data(employee e)

{

p.f1("In no: Y.d.", e.no);

p.f1("In name: Y.s", e.name); //name Salary : Y.f, e.name, e.sal);

p.f1("In date of birth: Y.d-Y.d-Y.d", e.dob.d, e.dob.m, e.dob.y);

p.f1("In department: ");

switch (e.d)

{

case finance: p.f1("finance"); break;

case marketing: p.f1("Marketing"); break;

case personnel: p.f1("Personnel"); break;

}

scanf(

inf->find_employee(employee l), inf);

{

inf;

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

if ((e[i].no == n) && (e[i].debit > FACS & = 0))

scanf(-1);

}

Drawbacks of structures:

```

Main()
{
    struct s;
    {
        int a, b, c;
    };
    struct s s1, s2, s3;
    s3 = s1 + s2;
}
// If raised an error, since the pre-defined
// Operators can't understand the user-defined datatype
// Variables.

Not Only Structures concept, also is also not possible.
→ For this reason only we have to pass as a function
parameter. C++ provides an operator overloading concept.
We have to implement manual interpretation, we implement
any operations with operators.

```

```

struct s s1, s2, s3;
s3 = s1 + s2; // Invalid
s3.a = s1.a + s2.a; // Valid.

```

Applications:

→ Database management — In real world dissimilar elements are tasks place.

like Railway Reservation System

— Name, source, destination, distance, etc

Bank, DBMS, RDBMS written in C. Using Structures

extensively used.

Ques / Points out structures.

→ Can we create the structure without any member variable?

A) (No in C) (Yes in C++)

Ex: Main()

```

    struct s;
    {
    };
}

```

→ least size of Structure is 1 byte. [YES].

→ Giving the name [tag] to Structure is optional? [YES].

→ If is a Nameless Structure we can generate variable of this Structure only at end part of Structure body.

→ Can we define a structure within a function? [YES].

→ We can define the function inside the Structure? [NO].

→ Can we Create one existed Structure as a member to the Structure? [YES] [Container Shop]

→ Can we define a Structure inside a structure? Yes. (Nested).

→ Can we Create same Structure as member to the Structure? [NO]. logically size will be infinit.

ex: Struct s;

{ int a; // If is 2 bytes we can estimate ✓ }

Struct s s; // But it is we can't estimate ✗ .

};

};

→ Can we Create same Structure pointer as member to the Structure is called self referential Structure. Struct s;

{ int a;

Struct s *s;

}

→ For implementing list related applications
we can go for self referential Structure.

Bitfield:

The advantage of bitfield concept lot of memory can save, by using bitfield.

Bitfields allows to access a single bit based on structures.

→ Bitfields of length should be declared as unsigned because a single bit can't have sign.

Syntax:-

Struct Struct

Struct Struct-type-name &

{ typename

type name1: length;

type name2: length;

type name n; length;

} variable-list;

↳

→ A bitfield must be declared as integral (or) enumeration type-

→ The : indicates allocates the memory in bits not on byte bytes.

→ Many applications of the O.S is implemented on Bitfield Concept like time & date.

Struct time

{ unsigned ft-sec : 5; /* Two seconds */

unsigned ft-min : 6; /* Minutes */

unsigned ft-hour : 5; /* hours */

unsigned ft-day : 5; /* Days */

unsigned ft-month : 4; /* Months */

unsigned ft-year : 7; /* Year - 1980 */

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

```
#define MALE 0
```

```
#define FEMALE 1
```

```
#define SINGLE 0
```

```
#define MARRIED 1
```

```
#define DIVORCED 2
```

```
#define WIDOWED 3
```

```
void Main()
```

```
{
```

```
Struct employee
```

```
{
```

```
    unsigned gender : 1;
```

```
    unsigned mal-sat : 2;
```

```
    unsigned hobby : 4;
```

```
    unsigned Sch-Scheme : 1;
```

```
}
```

```
Struct employee e;
```

```
e.gender = MALE;
```

```
e.mal-sat = DIVORCED;
```

```
e.hobby = 5;
```

```
e.Sch-Scheme = 9;
```

```
p-f("In Gender = %d", e.gender);
```

```
p-f("In Marital Status = %d", e.mal-sat);
```

```
p-f("In Bytes occupied by e = %d", sizeof(e));
```

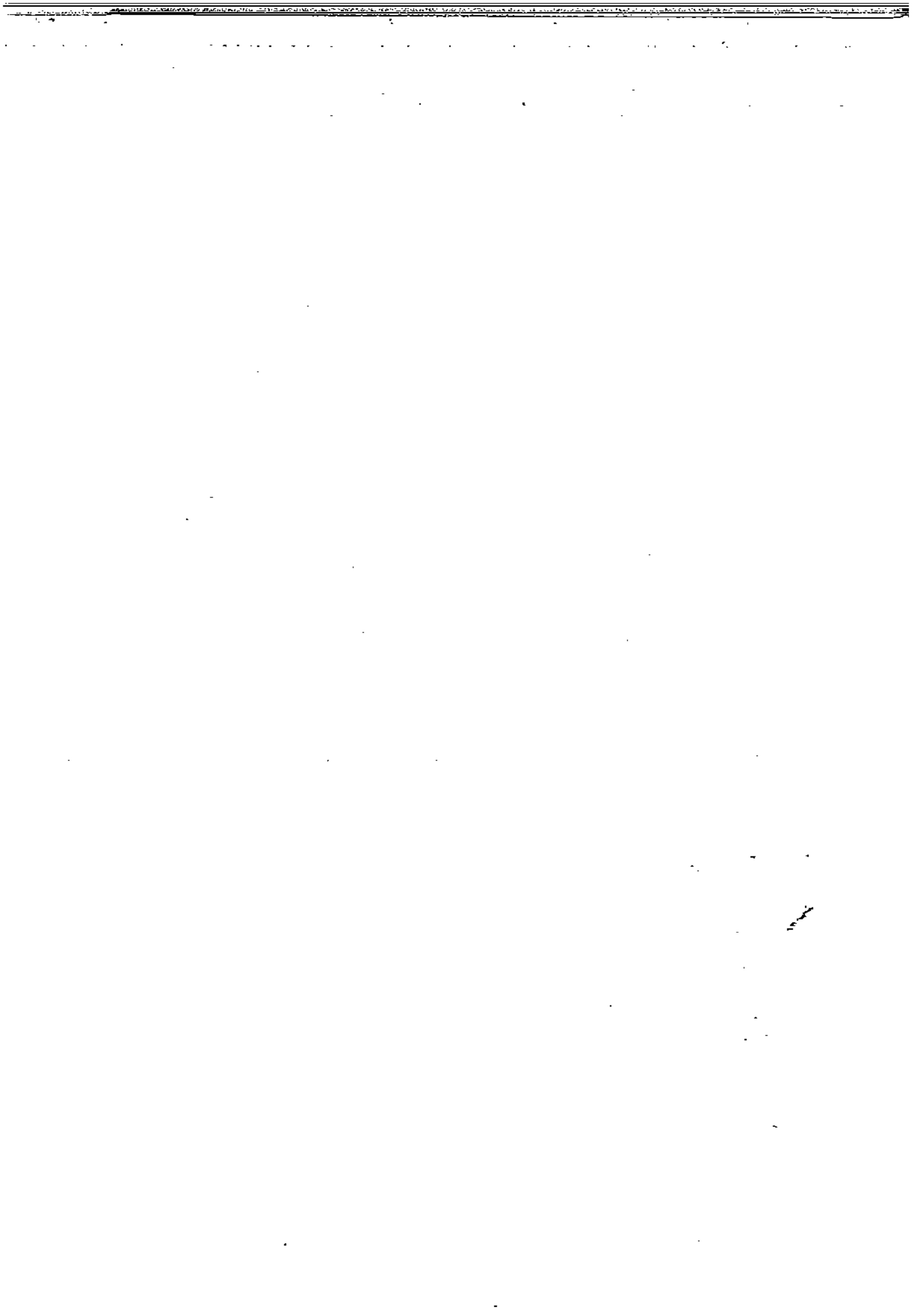
```
g
```

O/P:

Gender = 0

Marital Status = 2

Bytes occupied by e = 1.



MONDAY

31

20 12

By pressing F5
-c' prog window will be
minimized.

366-000 • Week 53

DECEMBER

DECEMBER 2012

S	M	T	W	T	F	S
30	31					1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29

1. Function errors are not displayed at compile time
i.e. at pressing of Alt + F9.
Ex:- printf().

we give it as `printf()`; the compiler
will not give any error at compilation time.
because function errors are
shown on Execution time

2. To Remove Add watch Alt + F3

3. Alt + no. like Alt + 1 - shows the first program &
i.e. program on 1st ~~window~~ window
like that we press

Alt + 1, Alt + 2, Alt + 3 ...

We can show the previous programs.

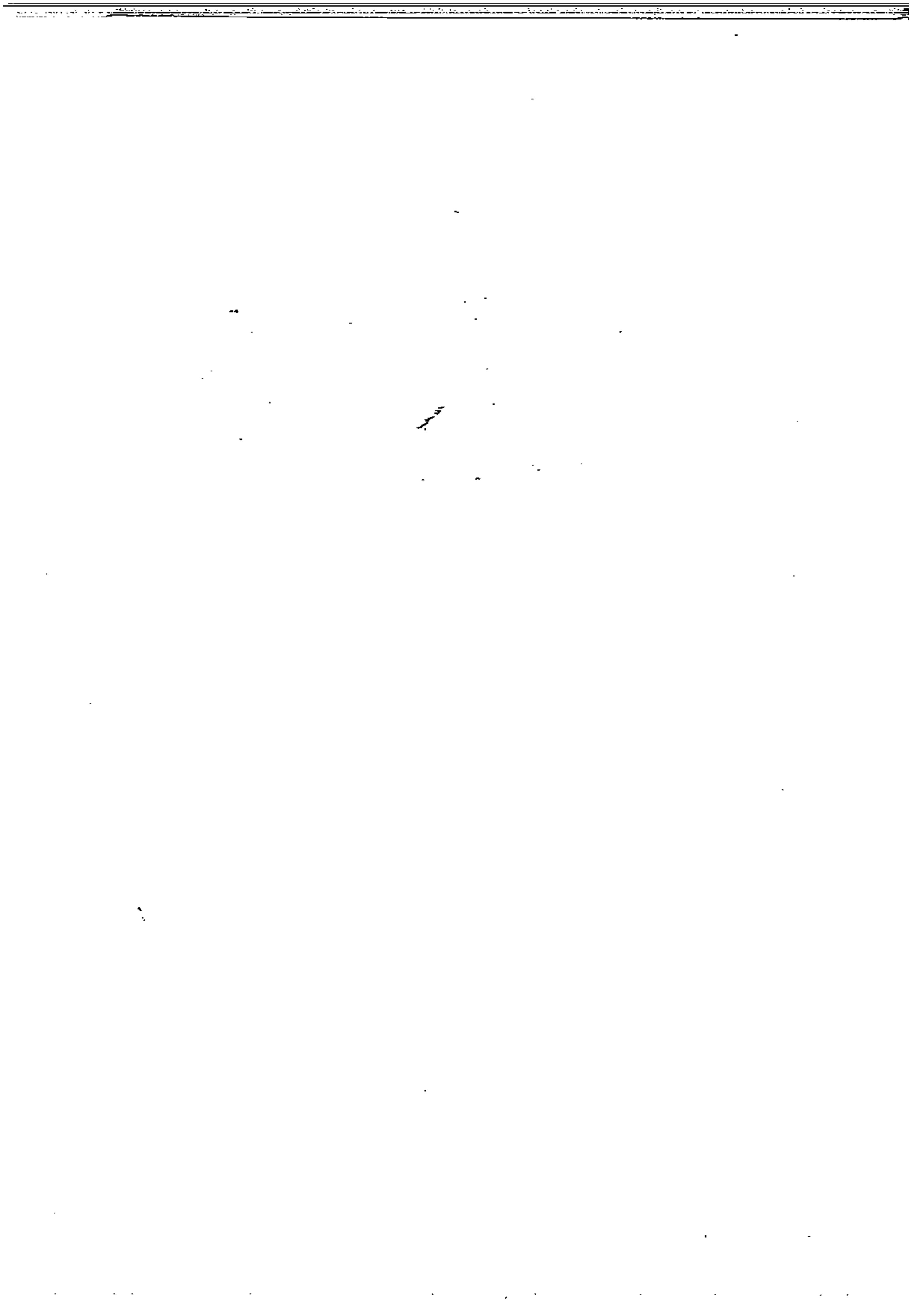
4. The Floating point Result will show upto 6 digits
only after .

5. Every Headerfile has some predefined functions &
variables. To show that we press Ctrl + F1

6. Importance of getch(); function.

REPLY TO

We are not using `getch()` in your program.
your program is compiled & executed also, but
the output screen will be going to background.
in that time you pressing Alt + A5
if you are using `getch()` in your program
when pressing Ctrl + F1 - the off is displayed
screen



35
38
Papers
(concept)

1.

```

#include<stdio.h>
#include<conio.h>
void main()
{
FILE *fp;
int eno;
char ename[20];
float sal;
clrscr();
fp=fopen("c:\\emp.dat","a+b");
printf("Enter Eno :");
scanf("%d",&eno);
printf("Enter Ename :");
fflush(stdin);
gets(ename);
printf("Enter Salary :");
scanf("%f",&sal);
fprintf(fp,"%d %s %.2f",eno,ename,sal);
printf("1 Record Saved");
fclose(fp);
getch();
}

void main()
{
FILE *fp;
int eno;
char ename[20];
float sal;
clrscr();
fp=fopen("emp.dat","r");
fscanf(fp,"%d %s
%f",&eno,ename,&sal);
printf("\nEno : %d",eno);
printf("\nEname : %s",ename);
printf("\n Salary :%.2f",sal);
fclose(fp);
getch();
}

```

2.

```

#include<process.h>
void main()
{
FILE *fp;
char fname[20],ch;
int nc=0,ns=0,nt=0,nal=0;
int nl=1,nw=1,flag=0;
clrscr();

puts("Enter the File Name ");
gets(fname);
fp=fopen(fname,"r");
if(fp==NULL)
{
printf("File Not Created");
getch();
exit(1);
}
while(1)

```

(4) →

```

ch=fgetc(fp);
if(feof(fp)) /* if(ch==EOF) */
break;
nc++;
if(ch>='A' && ch<='Z' || ch>='a' &&
ch<='z')
nal++;
if(ch==' ')
ns++;
if(ch=='\t')
nt++;
if(ch=='\n')
nl++;
if(ch==' ' || ch=='\t' || ch=='\n')
flag=1;
if(flag==1 && ch!='\t' && ch!='\n' && ch!='
')
{
nw++;
flag=0;
}
}
printf("\n No of chars = %d",nc);
printf("\n No of nal. = %d",nal);
printf("\n No of spaces = %d",ns);
printf("\n No of Tabs = %d",nt);
printf("\n No of Lines = %d",nl);
printf("\n No of words = %d",nw);
fclose(fp);
getch();
}

void main()
{
FILE *fp;
char ch,ch1,ch2;
char fname[20];
clrscr();
printf("Enter the File Name :");
gets(fname);
fp=fopen(fname,"r+");
if(fp==NULL)
{
printf("File Not FOUND");
getch();
exit(1);
}
printf("Enter Character1:");
flushall();
ch1=getchar();
printf("Enter Character2:");
flushall();
ch2=getchar();
while(1)
{
ch=fgetc(fp);
if(feof(fp))
break;
}

```

```

if(ch==ch1)
{
fseek(fp,-1,SEEK_CUR);
fprintf(fp,"%c",ch2);
fseek(fp,0,SEEK_CUR);
}
}
fclose(fp);
getch();
}

/* To Convert given file into caps and viceversa letters */
void main()
{
FILE *fp;
char fname[20],ch;
clrscr();
printf("Enter the File Name :");
gets(fname);
fp=fopen(fname,"r+");
if(fp==NULL)
{
printf("File Not FOUND");
getch();
exit(1);
}

while(1)
{
ch=fgetc(fp);
iffeof(fp))
break;
if(ch>='A' && ch<='Z')
{
fseek(fp,-1,SEEK_CUR);
fprintf(fp,"%c",ch+32);
fseek(fp,0,SEEK_CUR);
}
else if(ch>='a' && ch<='z')
{
fseek(fp,-1,SEEK_CUR);
fprintf(fp,"%c",ch-32);
fseek(fp,0,SEEK_CUR);
}
fclose(fp);
getch();
}

```

→ /* Encoding and Decoding the Program */

```

void main()
{
FILE *fp;
char fname[20],ch;
int code=0,flag;
clrscr();
printf("Enter the File Name :");
gets(fname);
fp=fopen(fname,"r+");

```

```

if(fp==NULL)
{
printf("File Not FOUND");
getch();
exit(1);
}
printf("For Encode Enter 1 ");
printf("\n For Decode Enter 2");
printf("\n Enter the Choice :");
scanf("%d",&flag);

if(flag==1) code is based on code
code=40; value it will encrypt
else if(flag==2) any value give.
code=-40; based on that it will
convert. if is increment
decrement also.

while(1)
{
ch=fgetc(fp);
iffeof(fp))
break;
if(ch!='\n' && ch!='\r')
{
fseek(fp,-1,SEEK_CUR);
fprintf(fp,"%c",ch+code);
fseek(fp,0,SEEK_CUR);
}
fclose(fp);
getch();
}

```

→ /* To reverse a the file whatever data we entered in a file */

```

#include<process.h>
#include<string.h>
#include<alloc.h>
void main()
{
FILE *fp;
char fname[20],ch;
char *str;
long int fsize,i=0;
clrscr();
printf("Enter the File Name :");
gets(fname);
fp=fopen(fname,"r+");
if(fp==NULL)
{
printf("File Not FOUND");
getch();
exit(1);
}
fseek(fp,0,SEEK_END);
fsize=ftell(fp); /* ftell is a function to get the total no of bytes of a file */
str=(char*)malloc(fsize+1);
fseek(fp,0,SEEK_SET);
while(1)
{
ch=fgetc(fp);

```

```

if(feof(fp))
break;
str[i++]=ch;
}
str[i]='\0';
strrev(str);
fseek(fp,0,SEEK_CUR);
fprintf(fp,"%s",str);
fclose(fp);
free(str);
getch();
}

```

→ /* Pgm to split a source into 4 parts and sent each part into 4 target files */

```

#include<stdio.h>
#include<process.h>
#include<conio.h>
#define f1 "c:\\kiran1.txt"
#define f2 "c:\\kiran2.doc"
#define f3 "d:\\kiran3.txt"
#define f4 "c:\\kiran4.txt"
int main()
{
FILE *sfp,*tfp;
char fname[100],ch;
char tfnames[4][100]={f1,f2,f3,f4};
long int fsize,i=0,nb=0;      nb = no of bytes
clrscr();
printf("Enter the File Name :");
gets(fname);
sfp=fopen(fname,"r");
if(sfp==NULL)
{
printf("File Not FOUND");
getch();
exit(1);
}
fseek(sfp,0,SEEK_END); // moving to end
fsize=f.tell(sfp); /* tell is a function to get
the total no of bytes of a file */
fseek(sfp,0,SEEK_SET); // Coming to initial position
tfp=fopen(tfnames[0],"w"); // tfp target file '0'
if(tfp==NULL)
{
printf("\n Unable ....");
exit(2);
}
while(1)
{
ch=fgetc(sfp); // Read the data from file
if(feof(sfp))      store in ch.
break;
nb++;
if(i!=3 && nb==fsize/4) // no of bytes initially = 1000/4 = 250
{
fclose(tfp);
tfp=fopen(tfnames[++i],"w");
}
}

```

```

nb=0;
}
}
fclose(sfp);
fclose(tfp);
getch();
} // Example for fread(), fwrite()
→ #include<stdio.h>
#include<conio.h>
struct emp
{
int eno;
char ename[20];
float sal;
};
void main()
{
FILE *fp;
struct emp e;
char ch;
clrscr();
fp=fopen("emp1.dat","a+b");
do
{
printf("Enter Eno :");
scanf("%d",&e.eno);
printf("Enter Ename :");
fflush(stdin);
gets(e.ename);
printf("Enter Salary :");
scanf("%f",&e.sal);
fwrite(&e,sizeof(e),1,fp);
printf(" Record Saved successfully");
printf("\nEnter another record (y/n)");
fflush(stdin);
scanf("%c",&ch);
}while(ch!='n');
fseek(fp,0,0);
printf("%-10s%-15s%-15s%-15s","ENO","ENAME","SALARY");
printf("\n-----");
fread(&e,sizeof(e),1,fp);
while(!feof(fp))
{
printf("\n%-10d%-15.2f",e.eno,e.ename,e.sal);
fread(&e,sizeof(e),1,fp);
}
fclose(fp);
getch();
}

```

1. fprintf
2. fscanf
3. accept the file Count no. of characters alphabets, Spacers file
4. write a prog accept a file & also accept 2 characters
replace the 1st character with the 2nd character

```

#include<stdio.h>
#include<conio.h>
struct student
{
    int sno;
    char sname[20],course[20];
    float fee;
};
typedef struct student stud;
stud input();
void addrecord();
int check_rec();
void header();
void display();
int search(int);
void search_rec();
void delete_rec();
void insert_before();
void insert_after();
void update_rec();
void main()
{
    int opt;
    while(1)
    {
        clrscr();
        printf("\n\n\n\t\t MENU");
        printf("\n\t-----");
        printf("\n\t\t1.Add Record");
        printf("\n\t\t2.Display All");
        printf("\n\t\t3.Search Record");
        printf("\n\t\t4.Delete Record");
        printf("\n\t\t5.Insert Before");
        printf("\n\t\t6.Insert After");
        printf("\n\t\t7.Update Record");
        printf("\n\t\t8.Exit");
        printf("\n\t-----");
        printf("\nEnter Your Option :");
        scanf("%d",&opt);
        clrscr();
        switch(opt)
        {
            case 1:
                addrecord();
                break;
            case 2:
                display();
                break;
            case 3:
                search_rec();
                break;
            case 4:
                delete_rec();
                break;
            case 5:
                insert_before();
                break;
            case 6:
                insert_after();
                break;
            case 7:
                update_rec();
                break;
            case 8:
                exit(0);
        }
        getch();
    }
}
stud input()
{
    stud s;
    printf("Enter Student Number :");
    scanf("%d",&s.sno);
    printf("Enter Student Name :");
    fflush(stdin);
    gets(s.sname);
    printf("Enter Course :");
    gets(s.course);
    printf("Enter Fee :");
    scanf("%f",&s.fee);
    return s;
}
void addrecord()
{
    char ch;
    stud s;
    FILE *fp;
    fp=fopen("stud.dat","ab");
    do
    {
        s=input();
        fwrite(&s,sizeof(s),1,fp);
        printf("\n1.Record Saved");
        printf("\nEnter Another Record(y/n)");
        ch=getchar();
        if(ch=='y')
            fflush(stdin);
        else
            break;
    }while(ch!='n');
    fclose(fp);
}
int check_rec()
{
    int c=0;
    FILE *fp;
    stud s;
    fp=fopen("stud.dat","rb");
    while(fread(&s,sizeof(s),1,fp)==1)
    {
        c++;
    }
    fclose(fp);
    if(c==0)
        return 0;
    else
        return 1;
}
void header()
{
}

```

```

{
    printf("%-10s%-15s%-\n"
15s%","SNO","SNAME","COURSE","FEE");
    printf("\n-----");
}
void display()
{
    FILE *fp;
    stud s;
    int c=0;
    if(check_rec()==0)
    {
        printf("No Records");
        return;
    }
    fp=fopen("stud.dat","rb");
    header();
    while(fread(&s,sizeof(s),1,fp)==1)
    {
        printf("\n%-10d%-15s%-\n"
15s%.2f",s.sno,s.sname,s.course,s.fee);
        c++;
        if(c%20==0)
        {
            printf("\nPress any key to
Continue.....");
            getch();
            clrscr();
            header();
        }
    }
    fclose(fp);
}
int search(int n)
{
    FILE *fp;
    stud s;
    fp=fopen("stud.dat","rb");
    while(fread(&s,sizeof(s),1,fp)==1)
    {
        if(s.sno==n)
        {
            fclose(fp);
            return 1;
        }
    }
    fclose(fp);
    return 0;
}
void search_rec()
{
    FILE *fp;
    stud s;
    int tsno;
    if(check_rec()==0)
    {
        printf("\nNo Records");
        return;
    }
}

}
printf("Enter Student no to search :");
scanf("%d",&tsno);
if(search(tsno)==0)
{
    printf("\nRecord Not Found");
    return;
}
fp=fopen("stud.dat","rb");
while(fread(&s,sizeof(s),1,fp)==1)
{
    if(s.sno==tsno)
    {
        printf("Student Name :%s",s.sname);
        printf("\nCourse :%s",s.course);
        printf("\nFee :%.2f",s.fee);
        fclose(fp);
        return;
    }
}
void delete_rec()
{
    FILE *fp,*fp1;
    stud s;
    int tsno;
    if(check_rec()==0)
    {
        printf("\nNo Records");
        return;
    }
    printf("Enter Student no to delete :");
    scanf("%d",&tsno);
    if(search(tsno)==0)
    {
        printf("\nRecord Not Found");
        return;
    }
    fp=fopen("stud.dat","rb");
    fp1=fopen("temp.dat","wb");
    while(fread(&s,sizeof(s),1,fp)==1)
    {
        if(s.sno!=tsno)
        {
            fwrite(&s,sizeof(s),1,fp1);
        }
    }
    fcloseall();
    remove("stud.dat");
    rename("temp.dat","stud.dat");
    printf("Record Deleted");
}

void insert_before()
{
    FILE *fp,*fp1;
    stud s,s1;
    int tsno;
    if(check_rec()==0)
    {
}

```

```
.printf("\nNo Records");
return;
}
printf("Before which sno do you want to
insert :");
scanf("%d",&tsno);
if(search(tsno)==0)
{
    printf("\nRecord Not Found");
    return;
}
fp=fopen("stud.dat","rb");
fp1=fopen("temp.dat","wb");
while(fread(&s,sizeof(s),1,fp)==1)
{
    if(s.sno==tsno)
    {
        printf("Enter new record for insertion\n");
        s1=input();
        fwrite(&s1,sizeof(s1),1,fp1);
    }
    fwrite(&s,sizeof(s),1,fp1);
}
fcloseall();
remove("stud.dat");
rename("temp.dat","stud.dat");
printf("Record Inserted");
}
void insert_after()
{
FILE *fp,*fp1;
stud s,s1;
int tsno;
if(check_rec()==0)
{
    printf("\nNo Records");
    return;
}
printf("After which sno do you want to Insert
:");
scanf("%d",&tsno);
if(search(tsno)==0)
{
    printf("\nRecord Not Found");
    return;
}
fp=fopen("stud.dat","rb");
fp1=fopen("temp.dat","wb");
while(fread(&s,sizeof(s),1,fp)==1)
{
    fwrite(&s,sizeof(s),1,fp1);
    if(s.sno==tsno)
    {
        printf("Enter new record for insertion\n");
        s1=input();
        fwrite(&s1,sizeof(s1),1,fp1);
    }
}
fcloseall();
remove("stud.dat");
rename("temp.dat","stud.dat");
printf("Record Inserted");
}
void update_rec()
{
FILE *fp,*fp1;
stud s;
int tsno;
if(check_rec()==0)
{
    printf("\nNo Records");
    return;
}
printf("Enter sno to update :");
scanf("%d",&tsno);
if(search(tsno)==0)
{
    printf("\nRecord Not Found");
    return;
}
fp=fopen("stud.dat","rb");
fp1=fopen("temp.dat","wb");
while(fread(&s,sizeof(s),1,fp)==1)
{
    if(s.sno==tsno)
    {
        printf("Enter new data for update\n");
        s=input();
    }
    fwrite(&s,sizeof(s),1,fp1);
}
fcloseall();
remove("stud.dat");
rename("temp.dat","stud.dat");
printf("Record updated");
}
```

For More Details Check the Website
[http:// kiransrinivas.wordpress.com](http://kiransrinivas.wordpress.com)

For doubts mail to
[vKiransrinivas@yahoo.co.in](mailto:vkiransrinivas@yahoo.co.in)

@@@ALL THE BEST@@@
By
KIRAN SIR (NARESH IT)

C-project

```

1 int main()
{
    float arr[] = {12.4, 2.3, 4.5, 6.7};
    printf("%d\n",
    sizeof(arr)/sizeof(arr[0]));
    return 0;
}

2 int main()
{
    int arr[5], i=0;
    while(i<5)
        arr[i] = ++i;
    for(i=0; i<5; i++)
        printf("%d ", arr[i]);
    return 0;
}

3 main()
{
    static int a[5] = {2,4,6,8,10};
    int l, b=5;
    for(l=0; l<5; l++){
        if(a[l]&b);
        printf("%d %d\n", a[l], b);
    }
}

f(x,y)
int x,*y;
{
    x = *y += 2;
}

4 main()
{
    int l = 5, j=10;
    abc(&l, &j);
    printf("%d %d\n", l, j);
}
abc(int *l, int *j)
{
    *l = *l + *j;
    *j = *l - *j;
    *l = *l * *j;
}

5 void test(int, int *);
main()
{
    int *iptr, j, k = 2;
    iptr = &j;
    j = k;
    printf("%d %d ", k, j);
    test(j, iptr);
    printf("%d %d\n", k, j);
}
void test(int l, int *p)
{
    l++;
    (*p)++;
}

6 #define INTPTR int *
main()
{
    INTPTR pl, p;
    int L;
    L=10; j=20;
    pl = &L;
    p = &j;
    j++;
    i = *pl;
    printf("%d\n", i);
    j++;
    i = *p;
}

7 main()
{
    unsigned int m[] = {0x01, 0x02, 0x04,
    0x08, 0x10, 0x20, 0x40, 0x80};
    unsigned int n;
    scanf("%d", &n); // Input n = 3
    for(l=0; l<7; l++)
    {
        if((n&m[l]))
            printf("\nyes");
        else
            printf("\nnos");
    }
}

8 What does the following
declaration mean?
int (*ptr)[10];

9 int main()
{
    static int a[2][2] = {1, 2, 3, 4};
    int f, j;
    static int *p[] = ((int*)a, (int*)a+1,
    (int*)a+2);
    for(l=0; l<2; l++)
    {
        for(j=0; j<2; j++)
        {
            printf("%d %d %d %d\n",
            *(p+l)+j, *(j+p)+l,
            *(l+p)+j, *(p+j)+l);
        }
    }
    return 0;
}

10 #include<stdio.h>
int main()
{
    void fun(int, int[]);
    int arr[] = {1, 2, 3, 4}, i;
    int l;
    fun(4, arr);
    for(l=0; l<4; l++)
        printf("%d ", arr[l]);
    return 0;
}

void fun(int n, int arr[])
{
    int *p=0;
    int l=0;
    while(l++ < n)
        p = &arr[l];
    *p=0;
}

11 #include<stdio.h>
void fun(int **p);
int main()
{
    int a[3][4] = {1, 2, 3, 4, 4, 3, 2, 8, 7, 8,
    9, 0};
    int *ptr;
    ptr = &a[0][0];
    fun(&ptr);
    return 0;
}

void fun(int **p)
{
    printf("%d\n", **p);
}

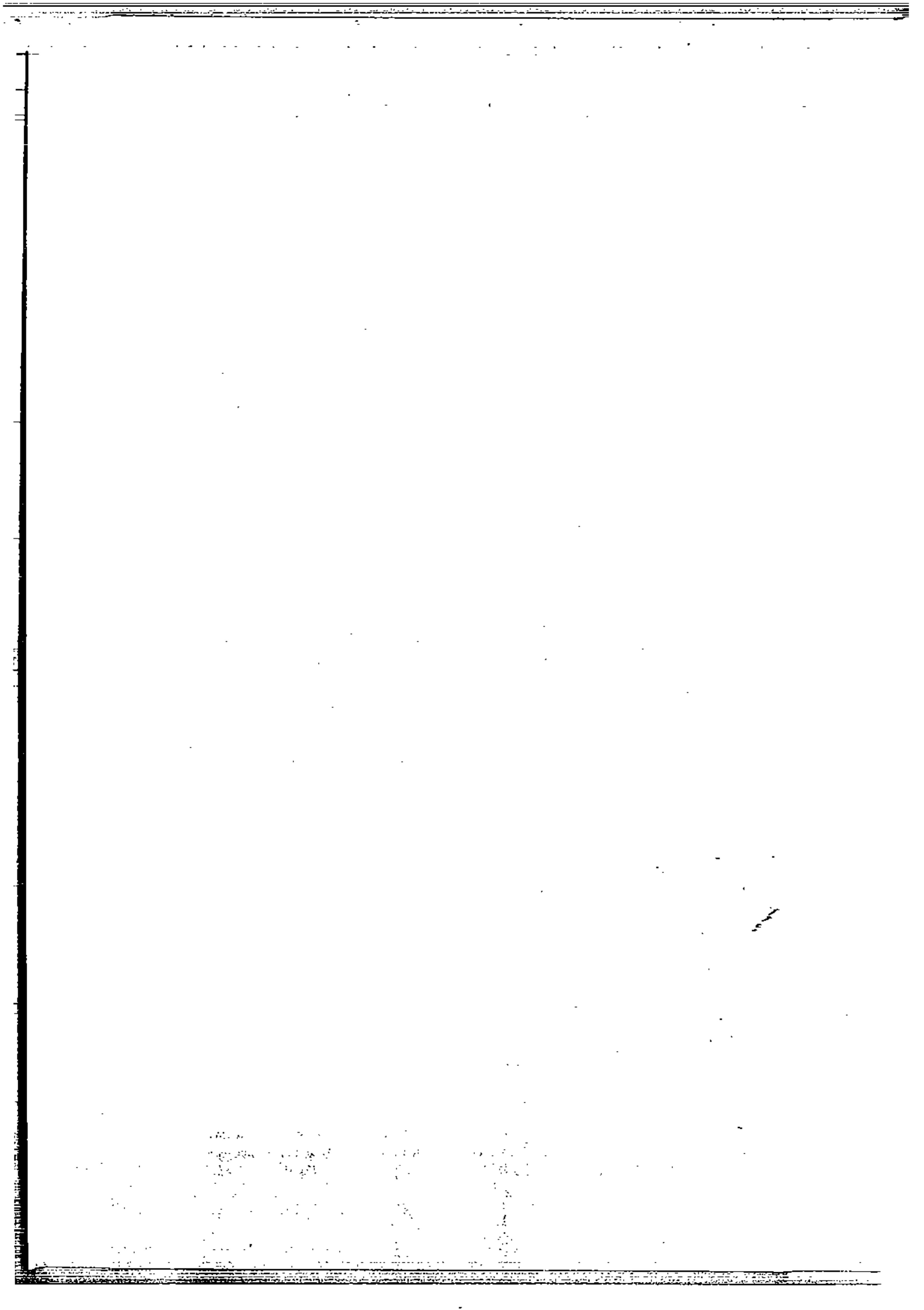
12 int main()
{
    static int arr[] = {0, 1, 2, 3, 4};
    int *p[] = {arr, arr+1, arr+2, arr+3,
    arr+4};
    int **ptr=p;
    ptr++;
    printf("%d, %d, %d\n", ptr-p, *ptr-
    arr, **ptr);
    *ptr++;
    printf("%d, %d, %d\n", ptr-p, *ptr-
    arr, **ptr);
    **ptr;
    printf("%d, %d, %d\n", ptr-p, *ptr-
    arr, **ptr);
    ++ptr;
    printf("%d, %d, %d\n", ptr-p, *ptr-
    arr, **ptr);
    return 0;
}

13 main()
{
    int l, m, n, b, x[25];
    int f1(int, int, int)(25));
    for(l=0; l<25; l++) x[l] = l;
    l=0; m = 24;
    b=f1(l, m, x);
    printf("res %d\n", b);
}

int f1( int p, int q, int a[25])
{
    int m1, m2;
    if(q==0)
        return(a[p]);
    else
    {
        m1 = f1(p, q/2, a);
        m2 = f1(p+q/2+1, q/2, a);
        if(m1 < m2)
            return(m2);
        else
            return(m1);
    }
}

@@@ALL THE BEST@@@
BY
KIRAN SIR

```




```

a[10];
int b[10];
}u;
void main()
{
int i;
printf("%d", sizeof(u));
printf("%d", sizeof(u.a));
printf("%d", sizeof(u.a[0].i));
}
a. 4, 4, 0 b. 0, 0, 0 c. 100, 4, 0 d. 20, 20, 2,

```

15. What does the following program print?

```

#include <stdio.h> [WIPRO]
struct my_struct
{
int p:1;
int q:1;
int r:6;
int s:2;
};
struct my_struct bigstruct;
struct my_struct1
{
char m:1;
};
struct my_struct1 smallstruct;
void main (void)
{
printf ("%d %d\n", sizeof (bigstruct), sizeof
(smallstruct));
}
a. 10 1 b. 2 2 c. 2 1 d. 1 1 e. Not sure

```

16. struct a{ [SAMSUNG]

```

int x;
float y;
char c[10];
}
union b{
int x;
float y;
char c[10];
}
which is true?
a) sizeof(a)!=sizeof(b);
b) sizeof(a)=sizeof(b);
c) sizeof(a)>sizeof(b);
d) sizeof(a)<sizeof(b);

```

17.union sample

```

{
char x;
int y;
char s[10];

```

}sam;
a) 16 bytes b) 13 bytes c) 10 bytes d) none

18. What is main returning in the following program

```

struct transaction
{
int sno;
char desc[30];
char dc;
float amount;
}
main(int argc, char *argv[])
{
struct transaction t;
scanf("%d %s %c %f", &t.sno, t.desc, &t.dc,
&t.amount);
printf("%d %s %c %f", t.sno, t.desc, t.dc, t.amount);
}

```

19.main()

```

{
struct a
{
category : 5;
scheme : 4;
};
printf("size= %d", sizeof(struct a));
}

```

20. What would be the output of following program?

```

struct syntax
{
int i;
float g;
char c;
}
main()
{
printf( "I won't give you any error" );
}

```

21.what is the difference b/w arrays and structures?

22.what is the difference b/w structures and unions?

23.what is the bit-field?

24.what are the features of a structures?

25.WAP to define the structure containing the details of employee.the structur may contain first,middle,last name,city and pincode.use typedef to create data type.display the records of 2 employees.

1 Which of the following function sets first n characters of a string to a given character?

- A. strinit() B. striset()
C. strset() D. strcset()

2 Which of the following function is used to find the first occurrence of a given string in another string?

- A.strchr() B.strrchr()
C.strstr() D.strnset()

3 int main()
{
char p[] = "%d\n";
p[1] = 'c';
printf(p, 65);
return 0;
}

4 int main()
{
char str[] = "Kiran\0\Nar\0";
printf("%s\n", str);
return 0;
}

5 int main()
{
void fun();
fun();
printf("%d");
return 0;
}
void fun()
{
char c;
if((c = getchar()) != '\n')
 fun();
printf("%c", c);
}

6 int main()
{
static char str1[] = "dills";
static char str2[20];
static char str3[] = "Daffo";
int i;
i = strcmp(strcat(str3, strcpy(str2,
str1)), "Daffodills");
printf("%d\n", i);
return 0;
}

7 int main()
{
int i;
char a[] = "\0";

```
if (printf("%s", a))  
    printf("The string is empty\n");  
else  
    printf("The string is not  
empty\n");  
return 0;
```

```
8 int main()  
{  
static char mess[6][30] = {"Don't walk  
in front of me...", "I may not  
follow", "Don't walk behind  
me...", "Just walk beside me...", "And be  
my friend."};  
printf("%c,%c\n", *(mess[2]+9),  
*(mess[2]+9));  
return 0; }
```

9 Which of the following statements are correct about the below declarations?

char *p = "Sanjay";
char a[] = "Sanjay";

```
10 int main()  
{  
char *str;  
str = "%d\n";  
str++;  
str++;  
printf(str-2, 300);  
return 0;
```

11 What shall be the output of the following code?

```
main()  
{  
char str1[] = "Hello";  
char str2[] = "Hello";  
if (str1 == str2 && (*(str1+6) == *(str2+6)))  
    printf("\n Equal");  
else  
    printf("\n unequal");  
}
```

- a. Equal
- b. Unequal
- c. Compilation error.
- d. Runtime error.
- None of the above.

```
12 main()  
{  
char as[] = "\0\0";  
int i = 0;  
do{  
switch(as[i++]) {  
case '\\': printf("A");
```

```
        break;
    case 0 : printf("B");
        break;
    default : printf("C");
        break;
    }
}
while(i<3);
}

13 void main()
{
char a[5] = "abcd";
int b = 3;
printf("%c\n",a[b]);
printf("%c\n",*((char *) b)[(int) a]);
}

14 main()
{
char *arr = "This is to test";
printf("\n%c %c ",*(arr), *(arr+1));
}

15 main()
{
print_in_reverse( "char *str");
}
print_in_reverse( char *str)
{
    if( *str == '\0')
        return;
    print_in_reverse(str+1);
    printf("%c", *str);
}

16 #include <string.h>
main()
{
char strp[] = "Never ever say no";
char *chp, c='e';
int i,j;
chp = strrchr(strp, c);
i = chp-strp;
for(j=0;j<=i;j++)printf("%c",strp[j]);
}

17 main()
{
char input[] = "SSSWILTECH1\1\1";
int i, c;
for ( i=2; (c=input[i])!='\0'; i++){
    switch(c){
case 'a': putchar ('i'); continue;
case '1': break;
case 1: while(( c = input[++i]) != '\1'
&& c!= '\0');
case 9: putchar('S');
case 'E': case 'L': continue;
default: putchar(c);continue;
}
putchar(' ');
}
putchar('\n');
}

18 main(){
int a,b,c;
a=strcmp("a","A");
b=strcmp("A","a");
c=strcmp("A","A");
printf("\n%d",a);
printf("\n%d",b);
printf("\n%d",c);
return 0;
}

19 void main(){
char array[]="Kiran \0 Srinivas";
char *str="Kiran \0 Srinivas";
printf("%s %c\n",array,array[2]);
printf("%s %c\n",str,str[2]);
printf("%d %d\n",sizeof(array),
sizeof(str));
}
```

ALL THE BEST

@@@By Kiran Sir@@@

For more information download from
<http://kiransrinivas.wordpress.com>

1 What's the I value?

```
main()
{
    int I=5;
    If(I>5);
    {
        I=100;
        printf("%d",I); } }
```

2 void main()

```
{
    int Var = 90;
    If(Var += Var == ++Var == 89)
        printf("%d",Var); }
```

3 void main(void)

```
{
    int I;
    Intk=0;
    If(k=='0')
        printf("one");
    else If(k=='48')
        printf("two");
    else
        printf("three"); }
```

4 void main()

```
{
    int i=2;
    int a=4;
    If((i+3)>a)
        { printf("TRUE\n"); }
    else
        { printf("FALSE\n"); }}
```

5 What are Interrelated statements?

- a) If-else replace with operator
- b) Nested If-else statements allowed
- c) Multiple If statements allowed
- d) Multiple statements allowed
- e) We write If-else and switch-else

6 main()

```
{
    int I=3,J=1;
    If(I==J)
        printf("%d %d",I,J); }
```

7 void main()

```
{
    int x=-1,y=-1;
    If(++x==++y)
        printf("Naresh");
    else
        printf("Kiran"); }
```

8 void main()

```
If(sizeof(void))
    printf("Kiran");
else
    printf("Srinivas"); }
```

9 void main()

```
If(strcmp("Yes"))
    If(strcmp("No"));
```

10 void main()

```
If(0xA)
    If(0x2)
        If(\xeb)
            If(\012)
                printf("Dennis");
            else;
        else;
    else;
```

else;

```
11 void main()
{
    int a=5,b=10,c=1;
    If(a&&b>c){
        printf("Right");
    }
    else{
        break; }}
```

```
12 #include<stdio.h>
void main()
{
    If('0');
    else If(NULL)
        printf("Wrong");
    else;
```

```
13 void main()
{
    int a=2;
    If(a==a)
        printf("NareshIT");
    else
        printf("Kiran C"); }
```

```
14 void main()
{
    unsigned int b=5u;
    If(b>5)
        printf("Right");
    else
        printf("Wrong"); }
```

```
15 void main()
{
    int x,y;
    If(x+10u+10u>x)
        printf("%d %d",x,y);
    else if(x=y)
        printf("%d %d",x,y);
    else
        printf("%d %d",x,y); }
```

```
16 void main()
{
    If(strcmp("%c",59)); }
```

17 main()

```
{
    switch(S)
    {
        case 5: printf(" 5 ");
        default: printf(" 10 ");
        case 6: printf(" 6 "); }}
```

18 main()

```
{
    int I=5,J=0;
    switch(I&&J||!J) {
        case 1:
            printf("1");
        case 2:
            printf("2");
        default:
            printf("default"); }}
```

19 main()

```
{
    float I=1.5;
    switch(I)
    {
        case 1.0: printf("%f,I);
        break;
```

```
case 1.5: printf("%f,I);
break;
case 2.5: printf("%f,I);
break; } }
```

```
20 main()
{
    unsigned short a=-1;
    unsigned char b=a;
    printf("%d %d",a,b); }
```

```
21 main()
{
    int c=0,d=5,e=10,a,
    a=c>17d>1||e>1?100:200:300;
    printf("a=%d",a); }
```

```
22 main()
{
    int i=7;
    printf("%d",i++*i++); } 49
```

```
23 main()
{
    int i=4;
    If(i=1)
        printf("statement 1");
    else
        printf("statement 2"); }
```

```
24 main()
{
    int x=10,y=20;
    If(!x&&y)
        printf("x=%d",x);
    else
        printf("y=%d",y); }
```

```
25 main()
{
    int k=4,j=0;
    switch(k)
    {
        case 3: j=300;
        case 4: j=400;
        case 5: j=500;
    }
    printf("%d\n",j); }
```

```
26 main()
{
    int x=10,y=20;
    If(!x&&y)
        printf("x=%d",x);
    else
        printf("y=%d",y); }
```

```
27 main()
{
    char c=-34;
    int l=-32;
    unsigned int u=-16;
    If(c>l)
        printf("Pass1");
    If(c<u)
        printf("Pass2");
    else
        printf("Fail1");
    If(u>l)
        printf("Pass3");
    else
        printf("Fail2"); }
```

```
28 main()
{
    int l=0;
```

```

11.
switch_printf("K"),printf("ku"))
{
    case 1: printf("%d",i);
              break;
    case 2: printf("%d",++i);
              break;
}

29 main()
{
    100;
    printf("%d\n",100);
}

30 main()
{
    int a = 10, b = 5, c = 3, d = 3;
    if ((a < b) && (c == d++)) printf("%d %d %d %d", a, b, c, d);
    else printf("%d %d %d %d", a, b, c, d);
}

31 main()
{
    int i=4,j=2,k=0;
    char c1='a',c2='b';
    if(i==0)printf("k is zero\n");
    else if(i==2)printf("j is 2\n");
    else if(i==4)printf("i is 4\n");
    if(c1=='a')printf("c1 is not a\n");
    else if(c2=='b')printf("c2 is b");
    else printf("Hello\n");
}

32 void main()
{
    unsigned int x,y,z;
    int y=0;
    if(y>x) printf("y is greater than x\n");
    if(y == x) printf("y is equal to x\n");
    true\n";
    if((int)x>0) printf("x is positive\n");
}

33 void main()
{
    int i,j,k;
    i=2;
    j=4;
    k=i+j\&2;
    printf("%d\n",k);
    if(++k && ++i<-j) i++;
    j=++k;
    printf("%d %d %d",i,j,k);
    getch();
}

34 #include<stdio.h>
void main(){
    switch(0X0){
        case NULL:printf("Dennis
Ritchie");
                    break;
        case 'A':printf("Grady Booch");
                    break;
        case 'B':printf("Ken Thompson");
                    break;
        default:printf("Christopher");
    }
}

35 #include<stdio.h>
void main(){
    switch(2){
        case 1:L:printf("No");
        case 2:L:printf("%f",L);
        goto Love;
        case 3:L:printf("Please");
        case 4:L:Love:printf("Hi");
    }
}

```

For more information download from
<http://kransri@vias.wordpress.com>
 For doubts mail to
vkransri@yahoo.co.in

1 main()

```
{
    int a=0,b=0;
    a = (b = 75) + 9;
    printf("\n%d, %d",a, b);
}
```

2 $x \% y$ is equal to

- a) $(x - (x/y))$ b) $(x - (x/y)*y)$ ✓
 c) $(y - (x/y))$ d) $(y - (x/y)*y)$

3 main()

```
{
    int i=-1;
    +i;
    printf("i = %d, +i = %d \n",i,+i);
}
```

4 main()

```
{
    int i;
    printf("%d\n", (2+3,6,2+5));
    i = (2>3,6,5%3);
    printf("%d\n",i);
}
```

5 main()

```
{
    int i=-1,j=-1,k=0,l=2,m;
    m=i++&&j++&&k++||l++;
    printf("%d %d %d %d %d",i,j,k,l,m);
}
```

6 main()

```
{
    int x=10,y=5,p,q;
    p=x>9;
    q=x>3&&y!=3;
    printf("p=%d q=%d",p,q); }
```

7 main()

```
{
    float f1=0.0001;
    float f2=0.0003;
    double lf2=f2,lf1=f1;
    f2=f1+f2;
    f2=f2/100;
    printf("%f\n",lf2);
    printf("%f\n",lf1);
    lf1=lf1/100;
    lf2=lf2/100;
    printf("%f\n",f1);
    printf("%f\n",f2);
    printf("%f\n",lf2);
    printf("%f\n",lf1);
}
```

8 main ()

```
{
    int x=5,y=6,z=2;
```

```
z/=y/z==3?y/z:x*y;
printf("%d",z); }
```

9 main()

```
{
    int ans=12,m=10,k;
    k=((ans<2) && (m>2));
    printf("\n%d",k);
}
```

10 main()

```
{
    int x,y,z;
    y=2; x=2;
    x= 2*(y++);
    y=2*(++y);
    printf("\nx = %d y=%d z=%d",x,y,z);
}
```

11 main()

```
{
    int i=2;
    int j=i+(1,2,3,4,5);
    printf("%d",j);
}
```

12 main()

```
{
    int a=500,b=100,c=30,d=40,e=19;
    a+=b-=c*=d/=e%5;
    printf("%d %d %d %d %d",a,b,c,d,e);
}
```

13 main()

```
float i=12.5,j=2.2;
printf("%f",i%j);
```

14 main(){

```

int i=10;
i=i>14;
printf("i=%d",i);
}
```

15 main()

```
{
    int i=400,j=300;
    printf("%d..%d");
}
```

16 main()

```
{
    int x,y,z;
    x=2>5<=0;
    y=2>3<=1;
    z=3>2>1>0;
    printf("%d %d %d",x,y,z);
    x=-1<=1>=2<=1;
    y=2>=2<=3>=4<=0;
    z=5>=8<=3>=0;
```

Faculty Mr Kiran

```

16 printf("%d %d %d",x,y,z);
x=6<8>=1<=0>-1;
y=8>=6<=2>=5<=-1;
printf("%d %d",x,y);
}

17 main()
{
int a;
a=10;
a*10;
printf("\n%d %d %d",a,a*10,a);
a+=20;
printf("\n%d %d %d",a,a=50,a);
a++*a++;
printf("\n%d %d %d",++a,a++,a);
a=100;
printf("\n %d %d %d",a++,++a,a++);
printf("\n%d",++a);
}

18 main(){
int i=5;
printf("%d",i+++++i);
}

19 main(){
char not;
not=!2;
printf("%d",not);
}

20 main()
{
int a,b,c;
a=3>2&&5<=8;
b=3>(2&&5)<=8;
c=3<(2&&5>=8);
printf("%d %d %d ",a,b,c);
a=5>2!=0||2!=2;
b=5<(2!=0)||2==2!=1;
c=(5==5)!=0||2!=5;
printf("%d %d %d",a,b,c);
}

21 main(){
const int i=4;
float j;
j = ++i;
printf("%d %f", i,++j);
}

22 main(){
int i=5;
printf("%d",++i++);
}

23 main()
{
int a=1,b;

```

Naresh i Technologies

```

b=a,++a,++a;
printf("\n%d %d",a,b);    ↗ 3,3   a=1   b=1
a=1;
b=a++,++a,++a;           ↗ 2,3   a=2   b=3
printf("\n%d %d",a,b);    ↗ 2,3   a=2   b=3
a=1;
b=++a,++a,++a;           ↗ 4,4   a=4   b=4
printf("\n%d %d",a,b);    ↗ 4,4   a=4   b=4
a=1;
b=(++a,++a,++a);         ↗ 3,3   a=3   b=3
printf("\n%d %d",a,b);
}

24 main()
{
    int a=4.5;
    printf("%d",a);
}

25 main()
{
    printf("%d",-9&&9);
}

26 main(){
int i=5;
printf("%d",i==++i==6);
}

27 main(){
int i=10,j=20;
j=i, j?(i,j)?i:j:j;
printf("%d %d",i,j);
}

28 main(){
int c=-2;
printf("c=%d",c);
}

29 main(){
int i=5,j=10;
i=i&~j&&10;
printf("%d %d",i,j);
}

30 main(){
int i=4,j=7;
j=j||i++ && printf("YOU CAN");
printf("%d %d", i, j);
}

31 main(){
unsigned giveit=-1;
int gotit;
printf("%u ",++giveit);
printf("%u \n",gotit--giveit);
}

32 main() {

```

```
int a=10,b=20,c=a---b;
printf("%d %d %d",a,b,c); }

33 main() {
    int a =10,b=20,c=a----b;
    printf("%d %d %d",a,b,c); }

34 main() {
float value=10.00;
printf("%og %0.2g %0.4g
%of",value,value,value,value); }

35 main()
{
    int a,b,c;
    a=!(2>5)&&!6!=0;
    b=!(2<=2)||!(!(6!=0));
    c=2!=12&&0==!2>0;
    printf("\n%d %d %d",a,b,c);
    a=(-1)<0||!0>0;
    b=!(5<=0)&&!(5!=5<=0);
    c=2!=1>=18!=1||(17!=5!=12);
    printf("\n%d %d %d",a,b,c);
}

36 main()
{
    int a;
    a=printf("\nThree") + printf("\nfour")
* printf("\nfive") - printf("\nsix");
    printf("%d",a); }

37 main()
{
    printf("hai")||printf("bye"); }

38 main()
{
printf("\n%d,%d,%d,%d",printf("\na"
),printf("\nab"),printf("\nabc"),printf(
"\nabcd")); }

39 main()
{
    (printf("\na"),printf("\nab"),printf("\n
abc"),printf("\nabcd")); }

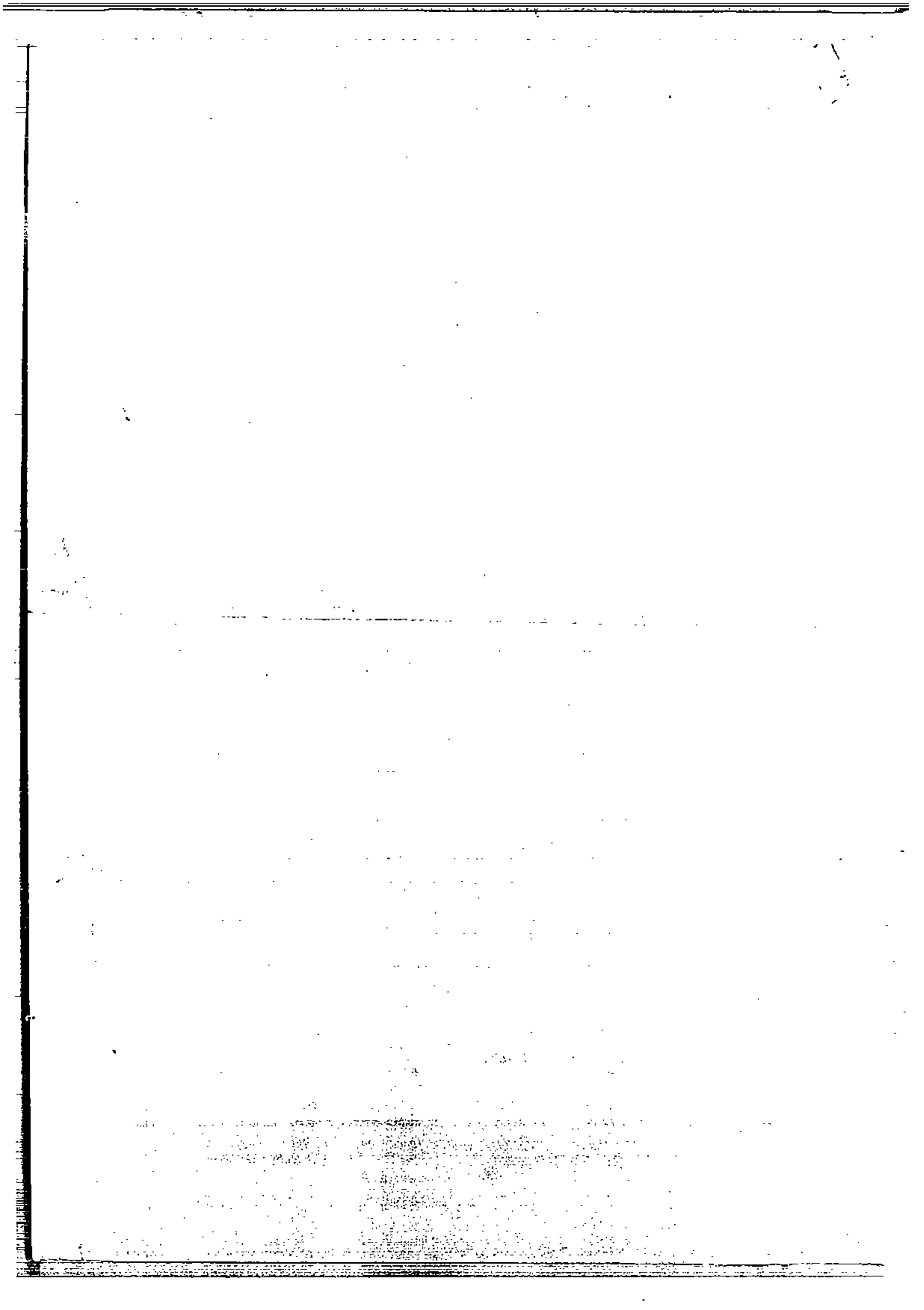
40 main()
{
    int x=5;
x==5?x<5?printf("one"):printf("two"):
printf("three"); }
```

```
41 main()
{
printf("0!=0)?printf("true"):printf
("false"); }

42 main()
{
float a1,b1,a2,b2,a3,b3;
a1=2;b1=6.8;a2=4.2;b2=3.57;a3=9.82
;b3=85.673;
printf("%3.1f,%4.2f\n",a1,b1);
printf("%5.1f,%6.2f\n",a2,b2);
printf("%7.1f,%8.2f\n",a3,b3);
}
```

For more downloads visit the website
<http://kiransrinivas.wordpress.com>

for doubts mail to
vkitansrinivas@yahoo.co.in



1 void main()
{ char a=127,b=7;
while(b>=0) ~~7>=0~~ ~~O/O~~
printf("%d",a & 1 << b--?1:0); } ~~0~~ ~~0/1/1/1/1~~

2 int main(void){
 while(!printf("Hello\n")) { }
 return 0; } ~~Hello~~

3 main() {
 int a=1;
 do {
 if(a%2==0)break;
 printf("%d\n",a++);
 } while(a<=10); }

4 main() {
 int a=1;
 do {
 if(a%2==0) continue;
 printf("%d\n",a++);
 } while(a<=10); } *going to infinite loop*

*This & line will internally
converted as
while(0)*

5 main() {
 int a=1;
 for(; ;) *infinite loop for this*
 printf("%d\n",a++); } ~~0 1 2 3 4 5~~

6 main() {
 for(; 0 ;) *internally converted as*
 printf("Hello\n"); } ~~Hello~~

7 main() {
 unsigned char a;
 for(a=1;a<=255;a++)
 printf("%d\n",a); } ~~1 2 3 ... 255~~

8 #include <stdio.h>
main()
{ int i=0;
char ch='A';
do
 putchar(ch); AAAAAAA B C D E F
 while(i++ < 5 || ++ch <='F'); } ~~i<6, i++ || 2<=ch <=F~~

9 main() {
 while(prlntf("Hai"),printf("")) D C = P
 printf("hello"); } ~~Hai~~ *E < P, F > P*

10 main() { int a=1;
 while(a<=10); {
 printf("%d",a--);
 if(a<5) break; *Break*
 } printf("%d",a+10); } ~~12~~

11 main() { int i;
 for(; scanf("%d",&i);printf("%d",i));
}
 //Entered Value is 30

12 Find the output for the following

i=20,k=0,j;
for(j=1;j<i;j=1+4*(i/j)) ~~8~~
{ k+=j<10?4:3; }
printf("%d",k); } ~~4~~

13 Which of the following is not an infinite loop

- (a) while(1) { } (b) for(; ;){...}
- (c) x=0; do { /* x unaltered within the loop */.... } while(x==0);
- (d) #define TRUE 0
 while(TRUE){...}

14 int i=10;

main()

```
{
{
int i=20,n;
for(n=0;n<=i;)
{
int i=10;
i++; } int i=10;
i++; } loop output
```

15 For the following C program

```
int d=0;
for(int i=0;i<31;i++)
for(int j=0;j<31;j++)
for(int k=0;k<31;k++)
    if(((i+j+k)%3)==0) d=d+1;
Find value of d
```

16 for(i=0,j=0;i<5,j<25;i++,j++)

- (a) i=4,j= 24 (b) i=24,j= 24 ~~(c) i=25,j= 25 (d) i=5,j=25~~

17 main()

Int x=5; i=0 condition fails.
for(;x==0;x--) {
 printf("x=%d\n", x--); }
a. 4, 3, 2, 1, 0
b. 1, 2, 3, 4, 5
c. 0, 1, 2, 3, 4
d. none of the above

18 main () {

```
int a=010,sum=0,tracker;
for(tracker=0;tracker<=a;
tracker++)
sum+=tracker;
printf(" %d\n",sum);}
A. 55     B. 36
C. 28     D. n
```

```
19 void main()
{
    int i;
    for(i=1;i++;i<100)  ↗ 32767
    printf("%d",i); } How many times
does the loop executes
```

```
20 main(){
    int i=0,n=6;
    while(n>0);
    i+=n;
    printf("%d\n",i);  ↗ 0
```

```
21 int main()
{
    unsigned int num;
    int c=0;
    scanf("%u",&num); //input 3
for(;num;num-1)
    { 3  ↗ 3-1
if(num&1)
{
    c++;
}
printf("%d",c);
getch(); }
```

```
22 main()
{
    char c='A';
    int i,mask=01;
    for(i=1;i<=5; i++)
    {
        printf("%c",c|mask);
        mask=mask<<1;  ↗ 11000
    }
    getch(); }
```

```
23 main()
{int i=0;
for(i=0;i<20;i++)
{
switch(i) {
case 0:i+=5;
case 1:i+=2;
case 5:i+=5;
default : i+=4;
break;}
printf("%d,",i);
}}
```

```
24 void main()
{
    int a,b;
    a=b=1;
    while(++a<=3 || ++b<=3)
    printf("%d%d",a,b);
    printf("%d%d",a+10,b+10);
}
```

```
25 void main()
{
    int i;
    for(i=1;i++<=1;i++)
    for(i++;i++<=6;i++)
    i++;
    printf("%d",i); } ↗ 12
```

```
26 void main()
{
    int i=65,j=0;
    for(j<26; i++,j++) {
        printf("%c\n", i); } ↗ STUVWXYZ
```

```
27 void main()
{
    int i;
    clrscr();
    for(i=1;i<6;++)
    switch(i)
    {
    case 1:
    case 2: printf("%d",i); break;
    case 3: continue;
    case 4: printf("%d",i);
    }
    printf("%d",i);
    getch(); }
```

```
28 main()
{
    signed int bit=512, i=5;
    for(;i>0;)
    {
        printf("%d\n", bit = (bit >> (i - (i
-1))));
```

```
29 int count = 11;
while(--count>0)
    printf("count down is %d\n",count)
```

```
30 main()
{
    char ch = 'A';
    while(ch <='F'){
        switch(ch){
            case 'A':case 'B':case 'C': case 'D':
            ch++; continue;
            case 'E': case 'F': ch++;
        }
        putchar(ch);
    }}
```

```
31 main()
{
    int i=6,j=4;
    printf("NO\n");
```

```

switch(i){                                }
do{
case 1: printf("yes\n");
case 2:
case 3:
case 4:
case 5:
case 6:   j--;
}while (j);    }

32 int main()
{
unsigned int i=4294967296;
while(i++!=0)
{
printf("%d",++i);
}
printf("\n");
return 0;}

33 #include<stdio.h>
int main()
{
short int i=0;
for(i<=5;i>=-1; ++i) i>0)
printf("%d", i);
return 0;
}

34 main()
{
int i=32,j=0x20,k,l,m;
k=i|j;
l=i&j;
m=k^l;
printf("%d%d%d%d",i,j,k,l,m);
}

35 main()
{
int i=4,j=8;
printf("%d %d
%d",i|j&j|i,i|j&&j|i,i^j);
}

36) main()
{
unsigned int m=32;
printf("%ox",~m);
}

37) main()
{
unsigned int a=0Xffff;
~a;
printf("%ox",a);
}

38) main()
{
unsigned int a,b,c,d,e,f;
a=b=c=d=e=f=32;
a<<2;
b>>2;
c^=2;
d|=2;
e&=2;
f~=2;
printf("%ox%cx%ox%ox%ox",a,b,c,d,e,f
);
}

39) main()
{
printf("%d>>%d%d>>%d",4>>1,8>>
1);
}

40)main()
{
printf("%ox",5<<3);
}

41) main()
{
unsigned int res;
res=(64>>(2+1-2))&(~(1<<2));
printf("%d",res);
}

42) main()
{
printf("%d %d ",32<<1,32>>0);
printf("%d %d ", 32<<-1,32<<-0);
printf("%d %d ", 32>>1,32>>0);
printf("%d %d ", 32>>-1,32>>-0);
}

43) main()
{
unsigned char i;
printf("%d",i<<1);
}

44 What will be output of the following
program?
main(){
  int a;
  a=015 + 0x71 +5;
  printf("%d",a);
}

45 main()
{
  int a= 256, b = -1;
  int c;
}

```

```

c = a & b;
printf("\n%od", c);
}

46.main() {
    printf("%o %o %o", 10, 010, 0x10);}

47 main() {
    printf("%x %x %x", 10, 010, 0x10);}

48.main()
{
    unsigned int bit=256;
    printf("%od", bit); }

49. main() {
signed int bit=512, mBit;
{
mBit = ~bit;
bit = bit & ~bit ;
printf("%od %od", bit, mBit);
} }

50 Where do you set the action iterations for a specified action?
a. Action Settings
b. Action Properties
c. Action Rule Properties
d. Action Call Properties

```

51 What is the correct statement to print A-Z alphabet

(a) for(a='A';a<'Z';a=a+1)
(b) for(a=a;a< div=""><>
(c)for(a='A';a<='Z';a=a+1)
(d)for(a=a;a<=z;a=a+1)

52 What does the hex number E78 in radix 7

A 12455 B 14153 C 14256
D 13541 E 131112

53 #include<stdio.h>

```

int main()
{
    int i = 5;
    while(i-- >= 0)
        printf("%od", i);
    i = 5;
    printf("\n");
    while(i-- >= 0)
        printf("%oi", i);
    while(i-- >= 0)
        printf("%od", i);
    return 0;}

```

54 int main()

```

unsigned int num;
int i;
scanf("%u", &num); //assume 5
for(i=0; i<16; i++)
{
    printf("%od",
    (num<<i & 1<<15)?1:0);
}
return 0;
}

Number can be displayed in
either 16-bit binary format

```

For more information download from
<http://kiransrinivas.wordpress.com>

For doubts mail to :
vkiransrinivas@yahoo.co.in

NAME OF THE STUDENT :
TOPIC : Loops and Bitwise Operators

For doubts Mail To vkiransrinivas@yahoo.co.in visit website <http://kiransrinivas.wordpress.com>

```
1 #include<stdio.h>
int i;
int fun();
int main()
{
```

```
    while(i)
    {
        fun();
        main();
    }
    printf("Hello\n");
    return 0;
}
```

```
int fun()
```

```
{   printf("Hi");   // Hi
}
```

```
2 int sumdig(int);
```

```
int main()
```

```
{
    int a, b;
    a = sumdig(123);
    b = sumdig(123);
    printf("%d, %d\n", a, b);
    return 0;
}
```

```
int sumdig(int n)
```

```
{
    int s, d;
    if(n==0)
    {
        d = n%10;
        n = n/10;
        s = d+sumdig(n);
    }
    else
        return 0;
    return s;
}
```

```
3 int main()
```

```
{
    int fun(int);
    int i = fun(10);
    printf("%d\n", --i);
    return 0;
}
```

```
int fun(int i)
```

```
{
    return (i++);
}
```

4 What is the error in the program

```
int main()
{
    int a;
    a = f(10, 3.14);
    printf("%d\n", a);
    return 0;
}
```

```
}
float f(int aa, float bb)
{   return ((float)aa + bb);}
```

5 Will the following functions work?

```
int f1(int a, int b)
```

```
{   return (f2(20));}
```

```
int f2(int a)
```

```
{   return (a*a);}
```

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

```
int n, R;
main()
{
    R = 0;
    scanf("%d", &n); // assume n=5
    printf("\n %d, %d", fun(n), R);
}
```

```
int fun(int n)
```

```
{
    if (n>3) return
    R = 5;
    R = 6;
    return(1);
}
```

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

```
main()
{
    int x;
    x = 3;
    f(x);
    printf("MAIN");
}
```

```
f(int n)
```

```
{
    printf("F");
    if (n!=0)
        f(n-1);
}
```

8 how many times the printf is executed

```
fun(n);
```

```
int fun( int n)
```

```
{
    int i;
    for(i=0;i<=n;i++)
        fun(n-i);
    printf(" well done");
}
```

9 Whether all recursive pgm can be written iteratively?

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

```
void main(void);
```

For doubts Mail To vkiransrinivas@yahoo.co.in visit website <http://kiransrinivas.wordpress.com>

```
double dbl=20.4530,d=4.5710,dblvar3;
void main(void)
{
    double dblIn(void);
    dblvar3=dblIn();
    printf("%.2f\t%.2f\t%.2f\n",dbl
,d,dblvar3);
}
double dblIn(void)
{
    double dblvar3;
    dbl=dblvar3=4.5;

    return(dbl+d+dblvar3);}
11 output?
initially n = -24;
printf (int n)
{
    if (n < 0)
    {
        printf ("");
        n = -n;
    }
    if (n% 10)
        printf ("%d", n);
    else
        printf ("%d", n/10);
    printf ("%d", n);
}
```

```
12 int fun(int);
void main()
{
    float k=3;
    fun(k=fun(fun(k)));
    printf("%f",k);
}
fun(int i)
{
    i++;
    return i;
}
```

13 how to define a function that can receive any number of arguments?

```
14 int main()
{
    float n=1.54;
    printf("%f, %f\n", ceil(n), floor(n));
    return 0;
}
15 int get();
int main()
{
    const int x = get();
    printf("%d", x);
    return 0;
}
int get()
{
```

```
    return 20;
}
```

16 What error would the following function give on compilation?
~~f(int a, int b)~~

```
{
    int a;
    a=20;
    return a;
}
```

17 In C all functions except main() can be called recursively. TRUE / FALSE

18 Point out the error, if any in the following program.

```
main()
{
    int b;
    b=f(20);
    printf("%d",b);
}
int f(int a)
{
    a>20? return(10):return(20);
}
```

19 Is this Program Raises Error or not
 Point out the error

```
main()
{
    int p=23, f=24; packman(p,f);
    printf("p=%d f=%d",p,f);
}
packman(q,h)
int q, h;
{
    q=q+q;
    h=h+h;
    return(q); return(h); }
```

20 int f(int a, float b)

```
{
    /* Some code */
}
```

```
int f(a, b)
int a; float b;
```

```
{
    /* Some code */
}
```

- (a) ANSI Notation
- KR Notation
- (b) KR Notation
- ANSI Notation
- (c) Pre ANSI C Notation
- KR Notation

```

1 main()
{
FILE *fp;
fp=fopen("TRAIL.c","r");
fclose(fp);
}

2 #include<stdio.h>
main()
{
char str[20];
FILE *fp;
fp=fopen(strcpy(str,"ENGINE.c"),"w");
close(fp);
}

3 #include<stdio.h>
main()
{
FILE *fp;
char str[80];
/* TRIAL.c contains only one line Its a round a
round , round world*/
fp=fopen("TRIAL.C","r");
while(fgets(str,80,fp)!=EOF)
puts(str);
}

4 #include<stdio.h>
main()
{
FILE *fp;
char c;
fp=fopen("TRY.c","r");
if(fp==NULL)
{
puts("cannot open the file");
exit(1);
}
while((c=getchar(fp))!=EOF)
putchar(c);
fclose(fp);
}

a)Normal Execution
b) Error message, NULL pointer assignment.
c)Cannot open the file.
d)None

5 #include<stdio.h>
main()
{
FILE *fp,*fs,*ft;
fp=fopen("A.c","r");
fs=fopen("B.c","r");
ft=fopen("C.c","r");
fclose(fp,ft,fs);
}
a)Error
b)No Output
c)Filename is printed
d)None

```

```

6 #include<stdio.h>
main()
{
char name[20],name1[20];
int age,age1;
printf("Enter name and Age\n");
scanf("%s %d",name,&age);
printf(" Enter name and age\n");
fscanf(stdin,"%s %d",name,&age1);
sprintf(stdout,"%s %d",name1,age1);
}

7 #include<stdio.h>
main()
{
static char str[]="triplet";
char *s;
s=str;
while(*s)
{
putc(*s,stdout);
sputchar(*s)
printf("%c\n",*s);
s++;
}
}

/*this program is stored as a file called prob.c*/
8 main(argc,argv)
int argc;
char *argv[];
{
printf("%d\n",argc);
printf("%s",argv[0]);
}

9 main()
{
char ch='z';
static char str[]="zebra";
putc(ch,stdout);
sprintf(stdout,"%s",str);
fwrite(str,5,1,stdout);
fputs(str,stdout);
}
(a)zzzz (b)zzzz (c)zZebraZebraZebra (d)none.

10 #include<stdio.h>
main()
{
struct a
{
char city[10];
int pin;
};
static struct a b={"udaipur",20};
static char c[]="bangalore";
FILE *fp;
fp=fopen("trail","wb");
fwrite(&b,sizeof(b),1,fp);
fwrite(c,9,1,fp);
}

```

)

- 11 which of the following functions is more versatile for positioning the file pointer in a file?
- rewind()
 - fseek()
 - ftell()

- 12 which of the following functions are ideally suitable for reading the contents of file record by record?
- getc()
 - gets()
 - fread()
 - fgets()

13 # include "stdio.h"

```
main()
{ FILE *fp;
  fp=fopen("trial", "r");
}
```

fp points to..

- The first character in the file
- A structure which contains a char pointer which points to first character in the file.
- The name of the file.
- None of the above.

- 14 If a file contains the line "I am a boy\r\n" then on reading this line into the array str using fgets(). What would str contain?

- "I am a boy\r\n\0"
- "I am a boy\0\0"
- "I am a boy\n\0"
- "I am a boy"

15 # include "stdio.h"

```
main()
{FILE *fp;
  fp=fopen("trial",r);
  fseek(fp,20,SEEK_SET);
  fclose(fp);
}
```

16 # include "stdio.h"

```
main()
{ FILE *fp;
  char str[80];
  fp= fopen("ieg","r");
  while(!feof(fp))
  { fgets(str,80,fp);
    puts(str)
  }
  fclose(fp)
}
```

- 17 What would be the output of the following program?

```
main()
{
  printf("\n%%%%");
}
```

- 18 point out the error if any ,in the following program ?

```
# include "stdio.h"
main()
{ unsigned char;
  FILE *fp;
  fp=fopen("c:\tc\ieg","w");
  if(!fp)
    exit();
  fclose(fp);
}
```

- 19 What would be the output of the following program :

```
main()
{ int n=5;
  printf("\nn=%d",n,n);
}
```

- Runtime error
- compile time error
- 5
- some garbage value,

- 20 According to Ansic specification which is the correct way of declaring main() when it receives command line arguments:

- main(int argc,char *argv[])
- main(argc,argv)
 - (int argc,
 - char *argv[])
- main()
 - (int argc;
 - char *argv[]);
- None

- 21 What would be the output of the following program?

```
/*sample.c*/
main( int argc, char * argv)
{ argc= argc-(argc-1);
  printf("%s",argv[argc-1]);
}
```

- 22 In the following program (myprog) is run from command

line as myprog 1 2 3 what would be the output ?

```
/*consider less than symbolically*/
main( int argc,char *argv[])
{ int i;
  for(i=0; i less than argc;i++)
    printf("%s",argv[i]); }
```

- 23 In the following program (myprog) is run from command

line as myprog 1 2 3 what would be the output ?

```
main( int argc,char *argv[])
{ int i;
  i= argv[1]+argv[2]+argv[3];
  printf("%d",i); }
```

24. In the following program (myprog) is run from command line as myprog 1 2 3 what would be the output ?

```
main( int argc,char *argv[ ] )
{ int i,j=0;
  for(i=0; i < argc;i++)
    j=j+atoi(argv[i]);
  printf("%d",j);
}
```

25. Which I/O function is used to receive input from keyboard and write output to VDU

- a) Disk I/O b) port I/O c) Console I/O d) None

26. Which I/O function have been categorised into formatted and unformatted I/O functions

- a) Disk I/O b) port I/O c) Console I/O d) None

27. Which of the following is formatted I/O functions used for input

- a) gets() b) scanf() c) both a and b. d) None

28. Which of the following is formatted I/O functions used for input

- a) getch() b) getche() c) both a and b. d) None

29. Which of the following functions is used to output a single character

- a) puts() b) putch() c) putchar() d) b and c

30. Which of the following is used as unsigned character format specifier

- a) %s b) %c c) %uc d) %d

31. Identify the following is used as unsigned character format specifier

- a) %d b) %I c) both a and b. d) none

32. %o

a) Organised format specifier

b) signed octal format specifier

c) Unsigned Octal format specifier

d) b and c

33. Identify the long signed format specifier

- a) %lu b) %ls c) %ld d) None

34. %10d would print the output exactly as:

- a) print the variable as a decimal integer in field of 10 columns.
b) print the output.
c) print the variable as a decimal integer in field of 10 columns as right justified.
d) print the variable as a decimal integer in field of 10 columns as left justified.

35. %-5d will print output with

- a) output padded with blanks in right.
b) output padded with blanks in between.
c) output padded with blanks in left.

d) None

36. main()

```
{  
char s1[ ]="ieg";  
char s2[ ]="scio";  
printf("%5s",s1);  
printf("%-5s",s2);  
}
```

Here the output in first and second printf statements would start printed from which column of output?

- a) 4,2 b) 2,0 c) 4,1 d) 3,2

37. Which of the following are escape sequences

- a) \n b) \b c) \f d) all the above.

44. To print \ (back slash) at the output ,which of the following is used as an escape sequence in .

```
printf
```

- a) \' b) \" c) \\ d) none

38. Which of the following used as alert escape sequences

- a) \a b) \A c) \a d) /a

39. main()

```
{  
char ch='Z';  
int i=125;  
printf("\n %c %d",ch,ch);  
}
```

what is the output of following program

- a) error b) Z 122 c) 122 122 d) Z,Z

40. Low-level disk I/O uses:

- a) UNIX system calls b) c's standard I/O library
c) both a and b d) None

41. What is that sprintf() used for

- a) reads a set of data values from file
b) Writes a set of data values from file
c) Writes only a character to a file
d) none

42. putc() is used for:

- a) reads a character from memory cache
b) reads a character from file.
c) both a and b
d) None

43. write the syntax for opening a file:

- a) fp=fopen("mode","filename");
b) fp=fopen("filename","mode");
c) fp=fopen("mode,filename");
d) fp=fopen("filename,mode");

44. Which mode of file opening is used to open a file for adding content to a file with current contents being safe:

- a) a b) a+ c) w d) both a and b

45. A file must be closed as soon as all operations on it have been completed because:

- a) So that all outstanding information associated with files is flushed out from buffers, all links to file are broken.
- b) We want to open file in different mode.
- c) To prevent accidental misuse of file.
- d) All the above

46. When `getc(file-pointer)` is used for reading data from file, reading should be done until:

- a) file-pointer points to null
- b) Using some Null values
- c) When the End-of-file mark EOF has been encountered
- d) All the above

47. What is syntax of `putw()` function in files:

- a) `putw(fp)`
- b) `putw(fp,integer)`
- c) `putw(integer,fp)`
- d) none

48. Syntax of `sprintf()`

- a) `sprintf(fp,"control string",list);`
- b) `sprintf("control string",fp,list);`
- c) `sprintf(list,fp,"control string");`
- d) none

49. Which of the following are error handling conditions during I/O

- a) Device overflow
- b) writing to a write protected file
- c) both a and b
- d) None

50. Typical error situations during I/O:

- a) Trying to read beyond End-of-file mark
- b) Trying to use a file that has not opened
- c) opening a file with invalid filename
- d) All the above

51. `error()` function

- a) Takes no arguments
- b) Takes pointer as an argument
- c) Takes filepointer as an argument
- d) Takes structure as an argument

52. Whenever a file cannot be opened using `fopen`

- a) Value 0 is returned
- b) Value 1 is returned
- c) Non-zero value is returned
- d) Null pointer is returned

53. `error` function returns

- a) Non zero integer if an error has been detected
- b) positive integer if an error has been detected
- c) Negative integer if an error has been detected
- d) None

54. `ftell` function takes file pointer as an argument returns

- a) Number of type long that corresponds to current position

b) Number of type int that corresponds to current position

- c) Number of type float that corresponds to current position
- d) None

55. `fseek(fp,m,0)`

- a) Move to $(m+1)$ th byte in the file
- b) Go forward by m bytes
- c) Go backwards by m bytes
- d) Move to the mth byte in the file.

56. `fseek(fp,0L,0)`

- a) Go to the End-of-file
- b) Go to the beginning
- c) both a and b
- d) None

57. `fseek(fp,-m,2)`

- a) Go backwards by $(m+1)$ bytes
- b) Go to the $(m+1)$ th byte
- c) Go to the mth byte
- d) Go backwards by m bytes from the end

58. _____ counts the no. of arguments on the command line

- a) argv
- b) argc
- c) both a and b
- d) None

59. Command line arguments in C are

- a) Arguments on C editor
- b) Parameters supplied to program when the program is invoked
- c) These can be passed as arguments to main()
- d) Both b and c

60. `argv` represents

- a) An array of integer pointers that point to the commandline arguments
- b) An array of long pointers that point to the command line arguments
- c) An array of character pointers that point to the commandline arguments
- d) None

61. _____ takes a file pointer and resets the positions to start of file

- a) `fseek(fp,0L,0)`
- b) `rewind(fp)`
- c) both a and b
- d) None

62. If statement on the command line is program x-file y-file the value in `argv[1]` is

- a) program
- b) x-file
- c) y-file
- d) none