Date. No.

Basic Structure of a C- program:

(i): Documentation section/ Comment section:
- It consists of author, date and description of program.
- It is optional
- It is used for understanding / future use.

// → single line
/* ... */ => multiple line.

(ii) Link section:
- It includes header files
eg: #include <stdio.h> => standard input-output function.
#include <conio.h> => console input-output.
#include <math.h> => mathematical operation
#include <string.h> => string function.

- This section is necessary
- It links pre-defined code in the program.

(iii) Definition section:
- It includes all symbolic constant.
- This is optional.
eg: #define PI 3.14

Date No.

(iv) Global Decleration Section:
→ It is used to define variables and functions globally.

(v): Main section:
- It consists of a main function and the program execution starts from main function
- This is compulsory.
- It is divided into two parts: decleration and execution part
- Decleration part is not necessary but execution part is important

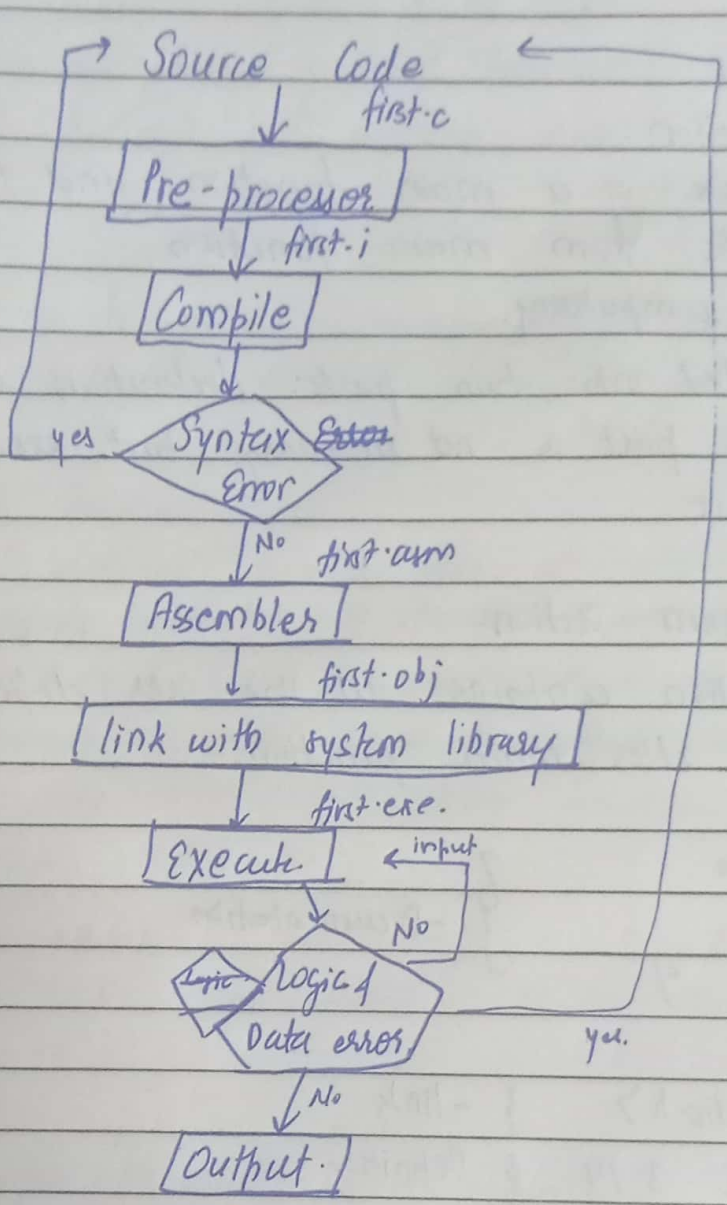
(vi) Sub- program section:
- This section contains all the user defined functions and comes after main function:

Eg:
```
// aaaa
/*   .....
     ...:  */        } - Documentation

# include < stdio.h >    } - link
# define PI 3.14         } Definition.
int a = 5                } global decleration.
void display();
void main ()
{ printf ("Hello world");    } main section
   display(); }
void display ()
{ printf ("Sun"); }      } - sub-program section
```

Date. No.

Execution process of a C-program:



Execution process is divided into three steps:

i) Translation
ii) Linking
iii) Loading

Date. No.

i) Firstly the program is written.
ii) The preprocessor replaces the files starting with '#' with the declaration of function included in this file. The source code is expanded
iii) The compiles compiles the program.
iv) Assembles converts the code into object code.
v) The object code is linked with system libraries
vi) Then, the file is localized to main memory in a single file and then the execution is done.

The execution of program starts from main function.

#Formatted Input Function in C:

→ scanf() is formatted input function.
→ It takes input and assigns a value.

Syntax: scanf ("control string", arguments)

control string = contains the format specifies
arguments = contains the address of variable.

Eg: scanf ("%d%d", &a, &b)

Date. No.

- It returns the numbes of input that has been taken from the user.
- It takes integer, float and character values or mixed.

Eg: scanf ("%4d %d", &a, &b);

This means ~~with is~~ width is 4.

Cases: i) %6d → takes 6 space → right justified

(ii) %-6d → takes 6 spaces → left justified.

(iii) %06d → right justified → fills empty space with 0

(iv) %-06d → left justified → fills empty space with 0.

Formatted Output Function in C:

→ printf() function is formatted output function.

Syntax: printf ("control string", argument);

i) putting a message
ii) putting format specifier.

Date. No.

Unformatted Input Function:

Here, the input is not arranged in particular format.

- It works on character datatype.

The functions are: getchar(), getch(), getche(), gets().

i) getchar():

→ it reads single character from standard input device

Syntax: variable = getchar()

→ Many characters can be included but it reads only one character from left.

ii) getch():

→ It is used to hold output screen. ~~and it doesn~~

→ It doesn't show what is written in the screen and we get directly output.

Syntax: ch=getch()

- It is used for passwords.

iii) getche():

→ It takes single character from console, at the character is ecode on the screen and immediately, next line is executed.

Syntax: ch = getche()

Date. No.

(iv) gets()
→ It takes complete string as input.
→ Character array is used to declare it.
Format specifier: %s.
Syntax: char ch[10]
gets(ch);

Unformatted output Function:

Here, the output is not arranged in particular format.
The functions are: putchar(), putch(), puts().

(i): putchar():
- It gives single character as output. → Under stdio.h
→ It doesn't move cursor to next line
Syntax: putchar(variable name);

(ii) putch():
- It gives single character as output.
→ falls under conio.h.
Syntax: putch(variable name);

(iii) puts():
Syntax
i) puts("string");
ii) char ch[10]
puts(ch);

→ It has newline operator build in so cursor move to next line after execution.