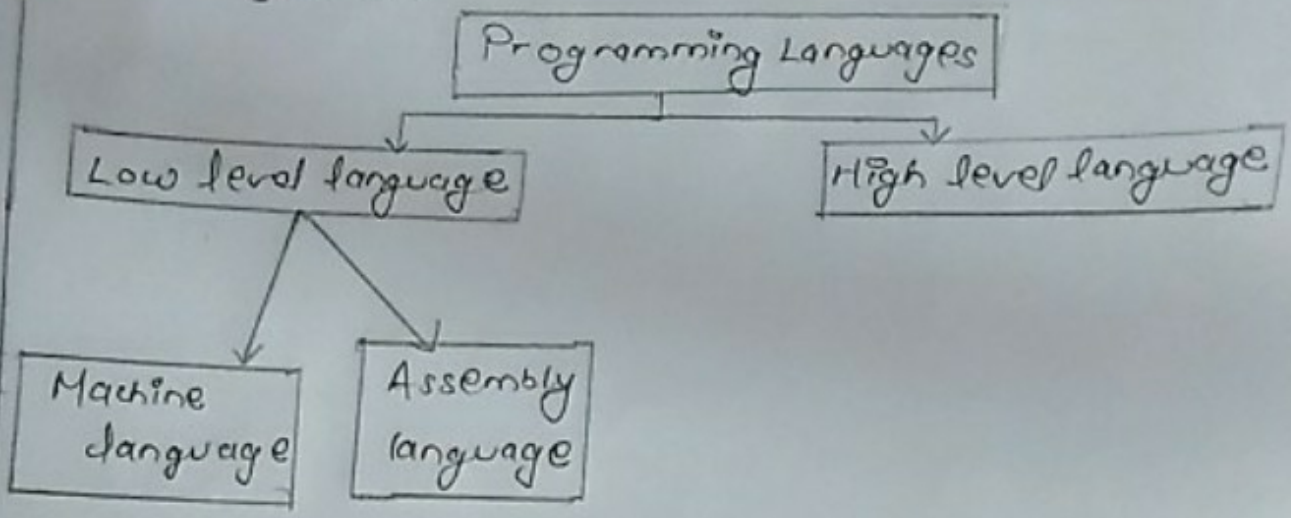
1 Define Programming Languages and explain its types.

⇒ The languages that are used to write a program or set of instructions are called "Programming languages". It is a set of grammatical rules for instructing computing device to perform specific task.

Its types are :-



A) Low level language :-

It is a computer programming language that is closer to the machine architecture. To write program in this language, one required detailed information about internal architecture of computer.

(i) Machine language or 1st generation language :-

The lowest level language in which every program statement is written using series of 1's and 0's which is directly understandable by CPU. It is very close to hardware.

An instruction in machine language tells the computer about

Location of data in main computer memory (RAM)

- Operation to perform, such as adding the two numbers together.
- Location address of result of this simple operation.
- Where to find the next instruction to perform

(ii) Assembly level language or 2nd generation language:-

It uses symbolic notations for writing a program. The instruction in assembly language are represented as alphanumeric symbolic codes called mnemonics.

- The negative part of this language is that the assembly program must be translated to machine code to execute as it cant be understood by CPU directly.
- The translation is done by language translator program called assembler.

3. High level language:-

High level languages uses format or languages that is most familiar to users so also called programmer friendly programming language.

- The instruction in this language are called codes
- Now for the performance of computer, this language is converted to machine level language with the help of language translators: Compiler & Interpreter.
- Compiler translates the whole set of lines of a program at once but interpreter does the translation line by line.

2. Explain Program errors:

⇒ Errors refers to violating the rules and program errors means violating the rules of programming.

Types of errors:-

A. Syntax error.
B. Logical/semantic error
C. Run time Error.

A. Syntax error :- It occurs due to violating the programming rules that are in syntax form and shows error message at the time of compiling so also called compile time error.

B. Logical/semantic error:- Errors that appear due to fault of programmer which leads to undesired or incorrect output.

This type of error is not detected during compile & run time and becomes difficult to detect the errors.

C. Run time Error :-

The errors that appears during program execution phase and generally occur due to some illegal operation performed in the program is called run time error.

For Eg:- Dividing a number by zero, trying to open a file which is not created, Lack of free memory space, etc.

3. What are the different features of good program:-

⇒ Every computer requires appropriate instruction set (Programs) to perform the required task and good program is that which ensures proper functionality of the computer.

Features of good Program:-

(i) **Portability:-**
Portability refers to the ability of an application to run on different platforms with or without minimal changes.

(ii) **General purpose:-**
If a program is developed for a particular task, then it should also be used for all similar tasks of the same domain. i.e able to solve the problem related to several fields.

(iii) **Efficient:-**
The efficiency is measured in the terms of machine efficiency which is also execution speed i.e requires less time for execution and memory efficiency which refers to the high work in least memory space.

(iv) **Readability:-**
The program should be written in such way that it makes other programmers or users to follow the logic of the program without much effort. If a program is written structurally, it helps the programmers to understand it in better way.

(v) Documentation :-

Documentation is one of the most important components of an application development. A well-documented applications is also useful for other programmers because even in the absence of the author, they can understand it.

(vi) Cost effectiveness :-

However, the good programe has been developed, it should not be cost more, for this developers can develop it in less time & no decrease in work performance.

4. Define C programming, also mention its features.

⇒ C programming is a general purpose, high level & structured computer programming language. Here, the programes are written in high level languages and are converted to object code with the help of language translator compiler.

Features of C language :-

(i) Simple :-

C language is simple and easy programming language for both learning and programming. Most of the computer programming languages follow its syntax and concepts like C++, Java, C#, etc.

(ii) Structured :- The program in C language follows the certain structure & also a big program is divided into a several different blocks structured

and manage the program easily.

(iii) Middle level :-
C is also called middle level programming language, because it contains features of both low level and high level programming languages.
-It refers to low level language as its program execution speed is faster, in comparison to other programming languages like c++, Java. and also very close to machine.

(iv) Rich set library :-
C language provides lots of libraries like math.h, string.h, graphics.h etc. These libraries helps to create programs or applications easily & faster.

(v) Memory Management :-
C language provides better memory management using pointer. Therefore, most of the memory related software are written in c language like DBMS, Compilers, etc.
we can allocate dynamic memory at runtime in c using pointer functions like malloc(), calloc(), alloc() etc.

(vi) Portable :-
C language is a portable programming languge due to which, we can run c language programs on different platforms with little modifications or without modifications.

and manage the program easily.

(iii) Middle level :-
C is also called middle level programming language, because it contains features of both low level and high level programming languages.
-It refers to low level language as its program execution speed is faster, in comparison to other programming languages like c++, Java. and also very close to machine.

(iv) Rich set library :-
C language provides lots of libraries like math.h, string.h, graphics.h etc. These libraries helps to create programs or applications easily & faster.

(v) Memory Management :-
C language provides better memory management using pointer. Therefore, most of the memory related software are written in c language like DBMS, Compilers, etc.
we can allocate dynamic memory at runtime in c using pointer functions like malloc(), calloc(), alloc() etc.

(vi) Portable :-
C language is a portable programming languge due to which, we can run c language programs on different platforms with little modifications or without modifications.

5 Write down the functions of
(a) include (b) printf (c) scanf (d) getch.

(a) **Include:-** It is preprocessor command & supports the content of header files for both predefined functions & user defined functions. This directive is read by the preprocessor and orders it to insert the content of a user defined or system header file into the following program.

(b) **Printf :-**
It is a predefined function that prints the text as it is and prints the value of expression or variable and these variables are defined in the header file <stdio.h>.

(c) **Scanf:**
It is also a predefined function whose definations are defined in header file <stdio.h>
It takes the data from the keyboard and ~~store~~ display on the screen and also store that data to variable.

(d) **getch :-**
The getch function basically stands for get character from the user. It is unformatted input function & usually used to hold the output screen until the user presses on the keyboard & its definations are included under header file <conio.h>

6.

Explain different programming paradigms.

Paradigm can be termed as method to solve some problem or do some task using some programming languages, tools and techniques.

There are lots of programming language that are known but all of them need to follow some strategy when they are implemented & called paradigms & different programming paradigms are:-

Imperative Programming Paradigm:-

It is one of the oldest programming paradigms of computer programming in which the program describes a sequence of steps that changes the state of computer.

It explicitly tells the computer "how" to accomplish the task.

It is further divided into:

(i) Procedural Programming Paradigm:-

It uses linear or top down approach. It relies on procedures or subroutines to perform computations. It breaks down the programming task into a collection of variables, data structures & subroutines.

(ii) Object Oriented Programming:-

The program is written as a collection of classes and objects which are meant for communication. The smallest & basic entity is object and all the computation is performed on the objects only.

- More emphasis is on data rather procedure
- It can handle almost all kind of real life problems which are today in scenario.

(3) Declarative Programming Paradigm :-

It is divided into Logic, Functional & Database. In computer science the declarative programming is a style of building programs that expresses logic of computation without talking about its control flow.

- It often considers programs as theories of some logic.
- It just declares that result we want rather how it has beenproduced.
- Hence, imperative means (how to do) and declarative means (what todo). programming paradigms.

Its types are:-

(i) Logic programming Paradigms:

It can be termed as abstract model of computation. It would solve logical problems like puzzles, series, etc. The main emphasize is on knowledge base and the problem. The execution of the program is very much like proof of mathematical statement, e.g., Prolog.

(ii) Functional Programming Paradigms:

The functional programming paradigm has its roots in mathematics and it is language independent.

- The key principal of this's paradigm is the execution of series of mathematical functions.
- Function can be replaced with their values without changing the meaning of the program.
- Some of the languages like perl, javascript mostly uses this paradigm.

(iii) Database/Data driven Programming Approach:-

This programming methodology is based on data & its movement. - A database program is a part of a business information system and provides file creation, data entry, update, query, reporting functions.
- Programming languages developed for database mostly is SQL.

7. Explain different compilation process.

⇒ The process of translation of source code into computer understandable form i.e object code is called compilation process.

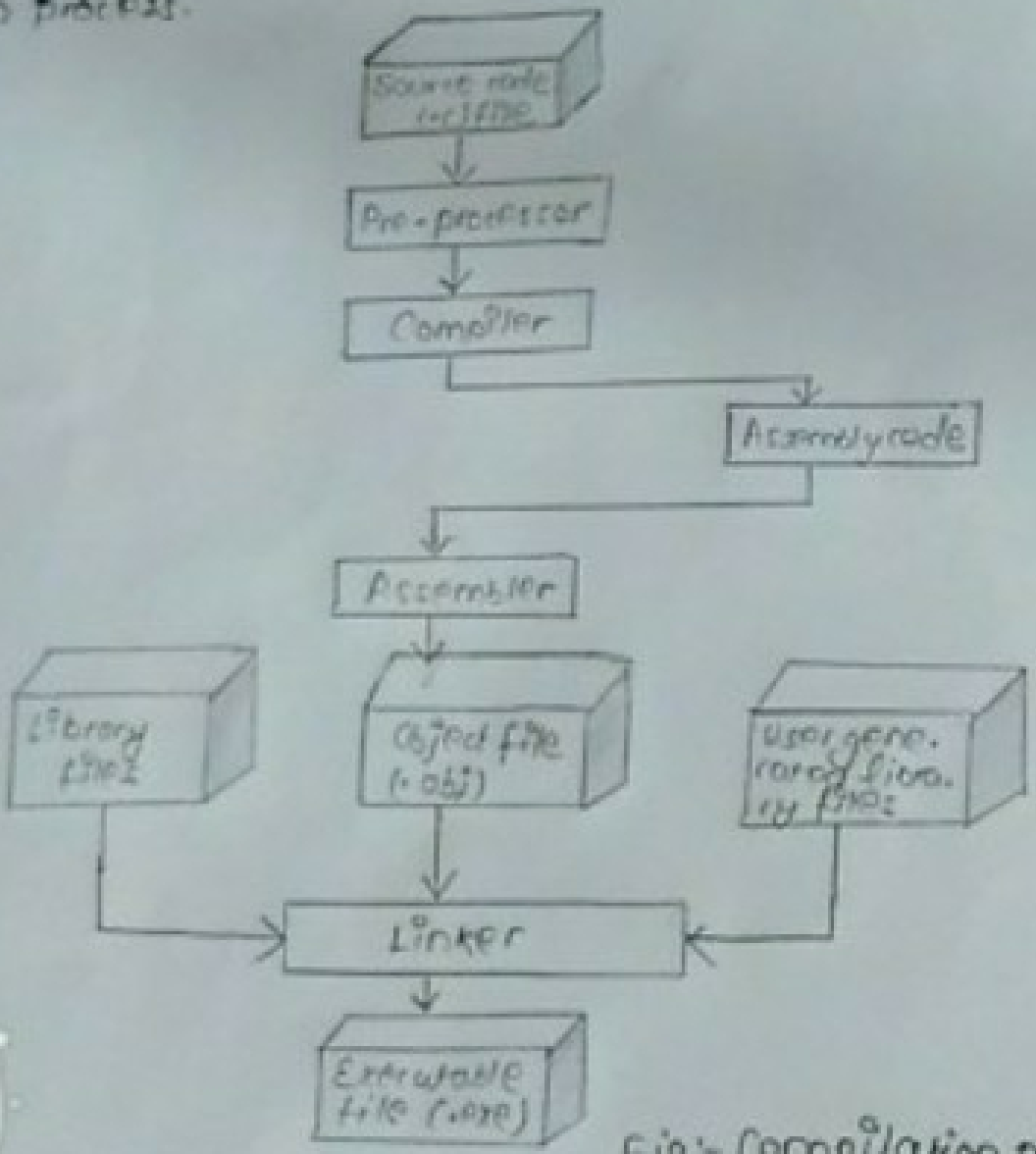


Fig:- Compilation process

Compilation process includes four steps:

1. Pre-processing
2. Compilation
3. Assembling & Linking.

1. Pre-processing :-

Pre-processing is a small software that accepts source file and performs following tasks

- Remove comments from the source code.
- Macro expansion
- Expansion of included header files

After pre-processing it generates a temporary file with .i extension. Since, it inserts contents of header files in our source code file.

- Preprocessor generated file is larger than the original source file.

2. Compilation of pre-processed file :-

In the next phase of C compilation the compiler comes in action. It accepts temporary pre-processed <file_name.i> file generated by preprocessor & performs following tasks.

- Check C program for syntax error.
- Translate the file into intermediate code i.e in assembly language. as <file name.s>
- Optionally optimize the translated code for better performance.

3. Assembling of compiled source code :-

Moving on to the next phase of compilation. Assembler accepts the intermediate source code (compilation.s) and translates to low level machine code.
After successful assembling it generates <file-name.o> (in Linux) or <file-name.obj> (in windows) file known as object file. In C compilation process it generates the compilation.o file.
This file is encoded in low level machine language.

4. Linking of object files :-

Finally, the linker comes in action & performs the final task of compilation process. It accepts the object file <file-name.o> generated by the assembler and links with built-in library files and possible other user generated object files.
It means the function printf() gets linked to its original defination. & generates the final executable file (.exe in windows)

- The job of Loader is to load the .exe file (binary program) into the memory for execution.

8. Define symbolic constant. What is significance? Illustrate with an example.

⇒ A symbolic constant is a name that is used in place of a sequence of characters that may represent numeric constant, character constant or string constant.

When a program is compiled each occurance of a symbolic constant is replaced by its corresponding character sequence.

Its significance are:-

(i) Modifiability:- If we have to change value of constant used in different places then we can modify its value simply by changing value of symbolic constant at the place where it is defined.

(ii) Understandability:- When a numeric value appears in a program, it used is not clear For eg:- number 10 represents no. of students & size of array in another place of same program. So Assignment of symbolic name instead of actual va numeric value in such case removes such problems. i.e NO_OF_STUDENTS to define number of students & SIZE to define size of array.

Examples:-

```
#define PI 3.1428
#define SIZE 80
```

Constant

Identifier

Preprocessor directive

9. Why is it required to translate a high-level source code? Differentiate between a compiler and an interpreter.

⇒ High level source code is in the human readable form that uses alphabets, numerics, constants symbols & machine understands the machine level object code i.e 0 & 1 and performs all the operations on the series of 0 & 1 so for the performance of computer system it is required to translate high level source code to machine code and it is done by the language translators softwares compiler & interpreter.

Differences between compiler & interpreter are as follows:-

Compiler	Interpreter.
(i) Compiler scans the entire line of the program before translating it into machine code.	(i) Interpreter translates & executes the program line by line.
(ii) Syntax errors are found only after the compilation of complete program.	(ii) Syntax errors can be trapped after translation of every line.
(iii) It takes less execution time.	(iii) It takes more execution time.
(iv) It requires more memory storage.	(iv) It requires less memory space.
(v) Generates the intermediate object code.	(v) No intermediate object code is generated.
Eg:- C, C++, Java.	Eg:- Python, perl