Unit-1

Introduction of C Programming

#What is programming? Describe the types of programming language with appropriate example

Explain different types programming language with their merits and demerits

Ans: The process of developing and implementing various sets of instructions to enable a computer to do a certain task. These instructions are considered computer programs and help the computer to operate smoothly. The language used to program computers is not understood by an untrained eye. Computer programming continues to be a necessary process as the Internet continues to expand.

Two Basic Types of Computer Language:

- Low-Level Languages: A language that corresponds directly to a specific machine
- High-Level Languages: Any language that is independent of the machine

Low-Level Languages:

Low-level computer languages are either machine codes or are very close them. A computer cannot understand instructions given to it in high-level languages or in English. It can only understand and execute instructions given in the form of machine language i.e. binary. There are two types of low-level languages:

- Machine Language: a language that is directly interpreted into the hardware
- Assembly Language: a slightly more user-friendly language that directly corresponds to machine language

Machine Language (1GL):

Machine language is the lowest and most elementary level of programming language and was the first type of programming language to be developed. Machine language is basically the only language that a computer can understand and it is usually written in binary digits (bits) 0 and 1. Since a computer is capable of recognizing electric signals, it understands machine language.

Advantages (merits) of Machine Language.

- Machine oriented Language.
- Machine language makes fast and efficient use of the computer.
- It requires no translator to translate the code. It is directly understood by the computer.

Disadvantages (demerits) Machine Language.

- All operation codes have to be remembered
- All memory addresses have to be remembered.
- It is hard to amend or find errors in a program written in the machine language.

Assembly Language (2GL):

Assembly language was developed to overcome some of the many inconveniences of machine language. This is another low-level but very important language in which operation codes and operands are given in the form of alphanumeric symbols instead of 0's and l's. These alphanumeric symbols are known as mnemonic codes and can combine in a maximum of five-letter combinations e.g. ADD for addition, SUB for subtraction, START, LABEL etc. Because of this feature, assembly language is also known as 'Symbolic Programming Language.' It is not very user friendly but execution is fast.

Advantages (merits) of Assembly Language:

- Assembly language is easier to understand and use as compared to machine language.
- It is easy to locate and correct errors.
- Program execution is faster than high level language.
- It is efficient in program execution. Hence, ALL is still used is developing firmware, device driver and operating system kernel.

Disadvantages (demerits) of Assembly Language:

- It is machine dependent; the programmer also needs to understand the hardware.
- the good knowledge of machine architectures is required.
- Program development and debugging is more difficult and time consuming than is High level language.

High-Level Languages (3GL):

High-level computer languages use formats that are similar to English. The purpose of developing high-level languages was to enable people to write programs easily, in their own native language environment (English).High-level languages are basically symbolic languages that use English words and/or mathematical symbols rather than mnemonic codes. Each instruction in the high-level language is translated into many machine language instructions that the computer can understand.

Advantages (merits): of high level Language.

- High-level languages are user-friendly
- They are similar to English and use English vocabulary and well-known symbols
- They are easier to learn and maintain
- They are problem-oriented rather than 'machine'-based.
- A program written in a high-level language can be translated into many machine languages and can run on any computer for which there exists an appropriate translator

Disadvantages (demerits): of high level Language:

- A high-level language has to be translated into the machine language by a translator, which takes up time
- Every programming language must have its own translator because high level language can't directly generate executable code.
- The object code generated by a translator might be inefficient compared to an equivalent assembly language program
- Computer does not understand HLL directly, so the program needs conversion before execution.

Programming Approach:

The Top-Down Approach

In the top-down approach, a complex algorithm is broken down into smaller fragments, better known as ‘modules.’ These modules are then further broken down into smaller fragments until they can no longer be fragmented. This process is called ‘modularization.’ However, during the modularization process, you must always maintain the integrity and originality of the algorithm. Moreover, a top-down approach is more suitable when the software needs to be designed from scratch and very specific details are unknown.

By breaking a bigger problem into smaller fragments, the top-down approach minimizes the complications usually incurred while designing algorithms. Furthermore, in this approach, each function in a code is unique and works independently of other functions. The top-down approach is heavily used in the C programming language.

Advantages:-

- Each module of code is to be tested separately.
- Breaking a problem down into smaller chunks makes it far easier to understand, solve and manage.
- Testing and debugging are efficient and easier.
- Project implementation is smoother and shorter.

Drawbacks:-

- Specification tends to change over time and in a top-down approach, all decisions made from the beginning of the project depend directly or indirectly on the high-level specification.
- In Dynamic Programming, the top-down approach is slow as compared to the bottom-up approach, as it involves recursion.

The Bottom-Up Approach

Contrary to the top-down approach, bottom-up programming focuses on designing an algorithm by beginning at the very basic level and building up as it goes. In this approach, the modules are designed individually and are then integrated together to form a complete algorithmic design. Moreover, the bottom-up approach is more suitable when a system needs to be created from

So, in this method, each and every module is built and tested at an individual level (unit testing) prior to integrating them to build a concrete solution. The unit testing is performed by leveraging specific low-level functions.

Advantages:-

- Test conditions are easier to create.
- Observation of test results is easier.
- Contains less redundancy due to the presence of data encapsulation and data-hiding.
- Reusability of the code.

Drawbacks:-

- In the Bottom-Up approach, we solve all sub-problems (even though some of the solutions of the subproblems aren't needed to solve), which requires additional calculations.
- In the Bottom-Up approach, sometimes it is difficult to identify the overall functionality of the system in the initial stages.

Top-down vs Bottom-up Programming

Top-Down Approach	**Bottom-Up Approach**
Top-Down Approach is Theory-driven.	Bottom-Up Approach is Data-Driven.
Emphasis is on doing things (algorithms).	Emphasis is on data rather than procedure.
Large programs are divided into smaller programs which is known as decomposition.	Programs are divided into what are known as objects is called Composition.
Communication is less among the modules.	Communication is a key among the modules.
Widely used in debugging, module documentation, etc.	Widely used in testing.
The top-down approach is mainly used by Structured programming languages like C, Fortran, etc.	The bottom-up approach is used by Object-Oriented programming languages like C++, C#, Java, etc.
May contains redundancy as we break up the problem into smaller fragments, then build that section separately.	This approach contains less redundancy if the data encapsulation and data hiding are being used.

Frequently Asked Questions

Why is bottom-up better than top-down?

The bottom up approach first identifies the small chunks of the problem and solves it moving its way to the top while the top down approach divides the bigger problem into smaller parts and solves it. Bottom up approach is better as it focuses on the fundamentals first and then moves on to the original problem as a whole.

History of C Language

History of C language is interesting to know. Here we are going to discuss a brief history of the c language.

C programming language was developed in 1972 by Dennis Ritchie at bell laboratories of AT&T (American Telephone & Telegraph), located in the U.S.A.

Dennis Ritchie is known as the **founder of the c language**.

It was developed to overcome the problems of previous languages such as B, BCPL, etc.

Initially, C language was developed to be used in **UNIX operating system**. It inherits many features of previous languages such as B and BCPL.

Let's see the programming languages that were developed before C language.

Language	Year	Developed By
Algol	1960	International Group
BCPL	1967	Martin Richard
B	1970	Ken Thompson
Traditional C	1972	Dennis Ritchie
K & R C	1978	Kernighan & Dennis Ritchie
ANSI C	1989	ANSI Committee
ANSI/ISO C	1990	ISO Committee
C99	1999	Standardization Committee

4. Define flowchart? Write some advantages and Disadvantages of flowcharts.

Ans: It is a graphical representation of an algorithm. It is used in programming to diagram the path in which data and instruction are processed to obtain desired output. It helps the programmer to understand the sequence of steps necessary to solve a given problem. It gives logical flow of the solution in a diagrammatic form, and provides a plan from which the computer program can be written.

Advantages/Merits of flowcharts

- A flowchart is a graphical representation of flow of an operation, hence any error in the logic can be easily detected.
- They are brief and to the point.
- They express clearly the logic of a given procedure
- It is very useful in case of modification of program in future.
- They are unambiguous as there can be only one direction of logic at any one time.
- They show at once whether all actions are covered.

Disadvantages/Limitation of flowcharts

- Complex and detailed charts can be laborious to plan and draw.
- Drawing a flowchart is time consuming.
- When there are complex branches and loops, flowcharts become very complicated to manages.
- The actions to be taken in specified situations can be difficult when many decision paths are involved.
- Flow chart cannot represent modular programming approach.

 # What is Algorithm? Write some features and advantages of algorithm.(2064)

Ans: Algorithm is a sequences of well define instruction that come step by step to solve the problems in a finite number of times .it can be written in any language the selection of language depends on the programmer. The simple common language makes it easy to the developers. The number of steps in an algorithm should be reduced to a minimum so as to increase the speed of the program. Algorithm should have following properties.

Features of Algorithm:

- Simple and easy language
- There should be finite number of steps in an algorithm.
- Common for every programming language
- Accurate result processing way.
- Each statement brings new step.
- The steps are in finite number of times.

Advantages of an Algorithm

- Algorithm is easy to understand and provide step by step solution of a problem.
- Logical error can be easily detected while checking the every step of the algorithm.
- Algorithm does not depend on any of the programming language.
- Algorithm is compatible to any programming language.

Coding

Coding is the translation of an algorithm or flowchart into a suitable computer language like c, c++, java etc. Coding is the real job of programmer. The algorithm to solve a problem which is described by pseudo-code or flow chart is converted into actual programming language code. The code written by programmer by using any programming language like C, C++ etc. is called the source code or source program.

Compilation and Execution

The source code written in any programming language is not directly executed by the computer. It should be translated into to the machine readable format i.e. actual machine language. The process of translation of source code into the target code is called the compilation. Each programming language has its own compiler program that translates the source code into its target code. The converted program in actual machine language is then executed by the computer which is known as program execution.

Debugging and Testing

A written program may have errors, some errors can be detected by the language compilers and some errors cannot be identified by the compiler and occurred during the program run. Common types of errors are:

Syntax Errors: Identified by compiler at the program compilation time.

Logical Errors: Not identified by the compiler at compile time and identified at the execution time. E.g. misuse of operators

So testing is the process of checking the program for its correct functionality by executing the program with some input data set and observing the output of the program.

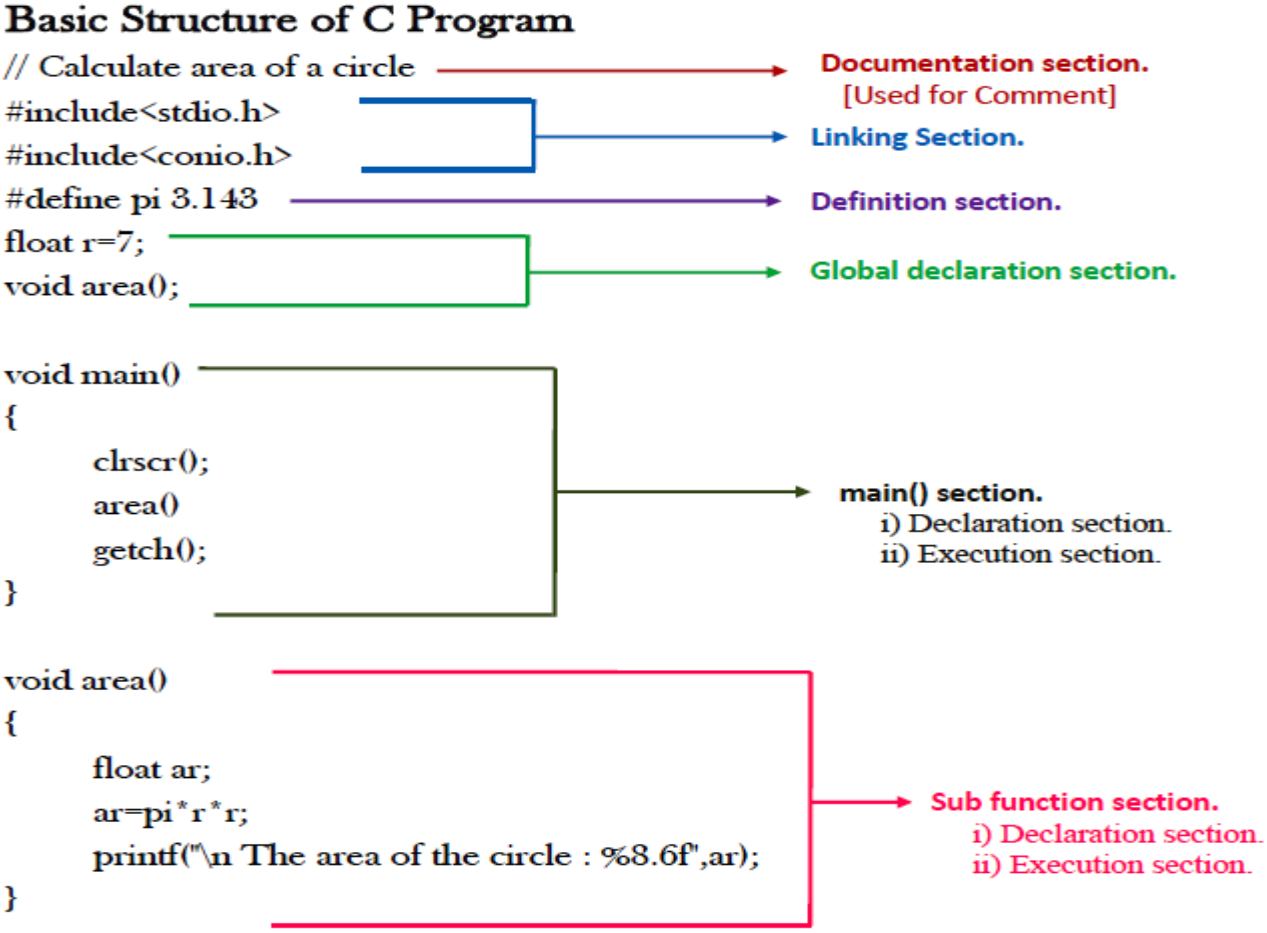
Documentation

From the start of the problem solving to the end of the implementation of the program, all the tasks should be documented i.e. kept for future reference. It is also the important part of the problem solving or program development. Documentation may be of two types:

a. Technical Documentation known as programmer's documentations which includes the problem analysis to implementation details for that program. It is needed for future reference for any modification, update of the program.

User manual is the documentation prepared for the end-user of the program that guides theuser how to operate the program

Basic Structure of C Program



or

Basic Structure of C Program

Whenever we create a program in **C language**, we can divide that program into six different sections. This section is as follows:

1. Documentation (Documentation Section)
2. Preprocessor Statements (Link Section)
3. Definition Section
4. Global Declarations Section
5. Main functions section
6. User-Defined Functions or Sub Program Section

In C language, all these six sections together make up the Basic Structure of C Program. Now let's learn in detail about all these sections -:

1. Documentation (Documentation Section)

Programmers write comments in the Documentation section to describe the program. The compiler ignores the comments and does not print them on the screen. Comments are used only to describe that program.

In the comments, programmer writes the name of the program, the author name which is making the program, and other information like – the date of the program, its purpose, etc. all These are written under the Documentation Section.

To know more about the comments in C language, see this -: Comments In C Language

Preprocessor Statements (Link Section)

Within the Link Section, we declare all the Header Files that are used in our program. From the link section, we instruct the compiler to link those header files from the system libraries, which we have declared in the link section in our program.

Example -:

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

In addition to all these Header Files in the Link Section, there are a lot of Header Files which we can link in our program if needed.

3. Definition Section

The definition of Symbolic Constant is defined in this section, so this section is called Definition Section. Macros are used in this section.

Example -:

```
#define PI 3.14
```

4. Global Declarations Section

Within the Global Declarations Section section, we declare such variables which we can use anywhere in our program, and that variable is called Global Variables, we can use these variables in any function.

In the Global Declaration section, we also declare functions that we want to use anywhere in our program, and such functions are called Global Function.

Example -:

```
int area (int x);  //global function
int n;          // global Variable
```

5. Main functions section

Whenever we create a program in C language, there is one main() function in that program. The main () function starts with curly brackets and also ends with curly brackets. In the main () function, we write our statements.

The code we write inside the main() function consists of two parts, one Declaration Part and the other Execution Part. In the Declaration Part, we declare the variables that we have to use in the Execution Part, let's understand this with an example.

Example -:

```
int main (void)
{
 int n = 15; // Declaration Part
 printf ("n = %d", n); // Execution Part
 return (0);
}
```

*****Thank you*****