Fundamentals of C Language

Structure of a C program:

C language does not actually follow any specific method of writing a program. It is a case sensitive language. All the statements must be written in lower case letters (small latters). The structure of a C program is as follows:

```
< Header files >
< Global declaration of variables >
Main ( )
{
        < Local declaration of variables >
            -  -  -  -  -  -  -  -  -  -  -
        < Statements >
            -  -  -  -  -  -  -  -  -  -  -
}
< sub programs - function blocks >
```

Q. What are the basic steps involved in writing a computer program?

Solution:

Writing of programs may involved the following steps:

1. Understanding the problem.
2. Planning the method of solution.
3. Development of the methods by using suitable algorithms, flowcharts, etc.
4. Coding the instructions in a programming language.
5. Transforming the instructions into machine readable form (using an input medium)
6. Program testing and debugging.
7. Documentation of the works involved in the production of the program.

Notes:

a) **Pseudo code:** Pseudo code consists of English-like statements describing an algorithm. It is written using simple phrases and avoids cryptic symbols. It is independent of high level languages and is a very good means of expressing an algorithm. It is written in a structured manner and indentation is used to increase clarity. As an example, the following algorithm finds and returns the maximum of n given numbers:

```
1  Algorithm Max(A,n)
2  /* A is an array of size n with index starts from 1 */
3  {
4          Result = A[1];
5          for i= 2 to n do
6                  if A[i] > Result then Result := A[i];
7          return Result
8  }
```

b) **C-preprocessor:** The C-preprocessor is a collection of special statements, called directives, which are executed at the beginning of the program compilation. The commands for the preprocessor are inserted in C source-code that is (.c) files and called compiler directives. Each compiler directive is prefixed by a hash sign (#).

Examples of preprocessor directives are #include, #define, #if, #elif, #endif, #undef etc.

#define (Macro Directive): 'C' allows defining an identifier having constant value using #define directive. This is called a preprocessor directive as if it is not part of a C program. This directive is placed at the beginning of a C program. The symbol # occurs in the first column and no semicolon is allowed at the end.

For example,

```
#define PI 3.14
#define MAXIMUM 100
```

#include (File Directive): #include is also a processor directive. This directive causes one file to be included in another for example #include<stdio.h> which appears at the start of the program.

c) **Library function:** Library functions are prewritten routines that carry out various commonly used operations or calculations. They are contained within one or more library files that accompany each C compiler.

Some library functions are described below:

fclose( ): The function fclose( ) closes a stream. The fclose( ) function returns 0 if the close operation is successful, otherwise it returns -1.

printf( ): The printf( ) function writes formatted output to stdout i.e. standard output device.

strcpy( ): strcpy( ) function copies one string to another string.

C-preprocessor C-preprocessor

Ex: What is a header file in C? List any two header files.

Solution: A C program is written using many functions, which are available under many header files. The required header file has to be included whenever such functions are used in the program.

Example:

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

Ex: Explain the need for the following:

#include<stdio.h>

#include<math.h>

Solution: The header file <stdio.h> is used to include and link standard input/output functions in a C program.

The header file <math.h> is used to include and link mathematical functions like sqrt( ), fabs( ) and so on in a C program.

Q. What is Preprocessor?

Solution: The preprocessor is used to modify your program according to the preprocessor directives in your source code. Preprocessor directives (such as #define) give the preprocessor specific instructions on how to modify your source code. The preprocessor reads in all of your include files and the source code you are compiling and creates a preprocessed version of your source code. This preprocessed version has all of its macros and constant symbols replaced by their corresponding code and value assignments. If your source code contains any conditional preprocessor directives (such as #if), the preprocessor evaluates the condition and modifies your source code accordingly. The preprocessor contains many features that are powerful to use, such as creating macros, performing conditional compilation, inserting predefined environment variables into your code, and turning compiler features on and off. For the professional programmer, in-depth knowledge of the features of the preprocessor can be one of the keys to creating fast, efficient programs.

Q. What is a computer program?

A set of instructions written in one of the programming languages to solve a problem is called a computer program.

Q. What is an assembler?

An assembler is a computer program or translator which translates an assembly language program into a machine language program.

Q. What is a source program ?

Solution:

The language in which a programmer writes programs is called source language. It may be a high-level language or an assembely language. A program written in a source language is called a source program. When a source program is converted into machine code by an assembler or compiler it is known as an object program.

Q. What is an object program ?

The translated or binary form of a source program is called an object program.

Q. What is an executable program ?

A program generated from object program by linking the input/output devices in order to execute the instructions given in a source program is called an executable program.

Q. What is a text editor ?

A text editor is a program which is used to type and edit your computer program or document. Commonly used editors are Turbo, NE(Norton Editor)Vi editor(Used in UNIX system)

Q. What is an operating system ?

An operating system is a collection of programs used to connect the user with the electronic hardware. The OS programs actuate and control the activities of a computer.

Q.: What are assemblers, compilers and interpreters?

Solution:
Assembler: An assembler is a computer program or translator which translates an assembly language program into a machine language program.

Compiler: Compiler translates a source program, written in high-level language into machine language. Compiler takes the whole program at the input and check for errors. So, it is fast in the speed. Most of the high level languages are compiled language.

One of the feature of the compiler that, it will convert the source program in the object program. So, every time no need of source program.

Interpreter: Interpreter translates each statement of the source program into a sequence of machine level instructions. Interpreters takes the input line by line and check for the errors. So, it is slower than compiler. Interpreter always need the source program every time.

Q. : What do you understand by compilation and execution of a program?

Solution: To translate the entire source code of a program from a high-level language into object code prior to execution of the program is known as compilation. A program that performs this task is known as compiler. When the object code made after compilation, the object code is linked with other library code, which are needed for execution of the program. The resulting code is known as executable code. If some error(s) occur during linking, debug them and compile the program again. After successful compilation we get file with **.obj** extension and after linking a file with **.exe** will be created. After executing the **.exe** file the result is obtained.

a. Function and Subroutine:

Solution:

Function: Function is the set of statements which can perform particular operation and called in the program whenever needed.

- Function always return value
- Function return value therefore it can be assign to the variable at the right side.

Subroutine: Subroutine is also same as function in sense set of statements to perform particular task.

- It always perform task, never return any value.
- The output of the subroutine will never assigned to the variable.

Q.8: State the differences between compiler and interpreter. State the advantages and disadvantages of machine level language, assembly language and high level language.

Solution:

The following table lists the differences between a Compiler and an Interpreter.

	Compiler	**Interpreter**
1	Scans the entire program first and then translates it into machine code	Translates the program line by line.
2	Converts the entire program to machine code; when all the syntax errors are removed execution takes place.	Each time the program is executed, every line is checked for syntax error and then converted to equivalent machine code.
3	Slow for debugging	Good for fast debugging
4	Execution time is less	Execution time is more

Advantage of Machine Language:

It is faster in execution since the computer directly executing it.

Disadvantage of Machine Language:

It is difficult to understand and develop a program using machine language. Anybody going through this program for checking will have a difficult task understanding what will be achieved when this program is executed. Nevertheless, the computer hardware recognizes only this type of instruction code.

Advantage of Assembly Language:

Writing a program in assembly language is more convenient than in machine language. Instead of binary sequence, as in machine language, it is written in the form of symbolic instructions. Therefore, it gives a little more readability.

Disadvantage of Assembly Language:

Assembly language (program) is specific to a particular machine architecture. Assembly languages are designed for specific make and model of a microprocessor. It means that assembly language programs written for one processor will not work on a different processor if it architecturally different. That is why the assembly language program is not portable.

Assembly language is not as fast as machine language. It has to be first translated into machine (binary) language code.

Advantage of High-level Programming Language:

There are four main advantages of high-level programming languages. These are:

(i) **Readability:** Programs written in these languages are more readable than assembly and machine language.

(ii) **Portability:** Programs could be run on different machines with little or no change. We can, therefore, exchange software leading to creation of program libraries.

(iii) **Easy debugging:** Errors could be removed (debugged).

(iv) **Easy Software development:** Software could easily be developed. Commands of programming language are similar to natural language (ENGLISH).

Q. Give the difference between testing and debugging of a program.

Solution:

Testing and debugging are two separate tasks. The differences between these two processes are outlined in Table 1:

Table 1: Differences between Testing and Debugging

Testing	Debugging
1. Testing is the process in which a program is validated.	Debugging is a process in which program errors are removed.
2. Testing is complete when all desired verifications in terms of the specifications have been performed	Debugging is a process that ends only temporarily, because subsequent execution of a program may uncover other errors thereby restarting the debugging process.
3. Testing can and should be planned. It is a definable task in which the how and what to test can be specified. Testing can be scheduled to take place at a specific time in the development cycle.	Debugging is a reactive procedure, which stems from testing. It cannot be planned ahead of time. The best that can be done is to establish guidelines of how to debug and develop a list of "what to look for."
4. Testing can begin in the early stages of the development effort. Of course, the test themselves must be run near the end of a project, but the decisions of what to test, how to test, with what kind of data can and should be completed before the coding is started.	Debugging, on the other hand, cannot begin until the end of the development cycle, because it requires an executable program.

Q. : Differentiate between syntax errors and run-time error.

Solution: When the compiler detects an error, the computer will display an error message.

(a) Syntax error messages: Syntax error or compilation errors are detected and displayed by the compiler as it attempts to translate our program. If a statement has a syntax error, it cannot be translated and our program will not executed.

(b) Runtime error messages: Run time errors are detected by the computer and are displayed during execution of a program. A run-time error occurs when the program directs the computer to perform an illegal operation, such as dividing a number by zero. When run-time error occurs, the computer will stop executing our program and will print a diagnostic our program and will print a diagnostic message that indicates the line where the error was detected.

Q. What is an lvalue?

Solution:

An lvalue is an expression to which a value can be assigned. The lvalue expression is located on the left side of an assignment statement, whereas an rvalue is located on the right side of an assignment statement. Each assignment statement must have an lvalue and an rvalue. The lvalue expression must reference a storable variable in memory. It cannot be a constant.

Q. Distinguish between L-value and R-value giving two examples of each.

OR

What do you understand by R-value and L-value? Give suitable examples.

Solution: The address associated with a program variable is in C called its **lvalue**; the contents of that location is its **rvalue**, the quantity which we think of as the value of the variable. The rvalue of a variable may change as program execution proceeds; its lvalue, never. The distinction between lvalue and rvalues becomes sharper if we consider the assignment operation with variables **alpha** and **beta**:

Alpha= beta;

beta, on the right hand side of the assignment operator, is the quantity to be found at the address associated with **beta,** i.e., is an rvalue, the contents of the variable **beta**. **alpha** , on the left hand side, is the address at which the contents are altered as a result of the assignment. **alpha** is

an lvalue. The assignment operation deposits **beta's** rvalue at **alpha's** lvalue. Think of an lvalue as something to which an assignment can be made.

Q.: What's the difference between #include <> and #include "" ?

Solution: The <> syntax is typically used with Standard or system-supplied headers, while "" is typically used for a program's own header files.

Q. Differentiate between compile-time and run-time errors giving proper examples of each.

OR

Give one example each of run-time and compile-time errors.

Solution:

A compile-time error is an error that occurs when a program is being compiled. Examples: syntax errors such as omitting a required semicolon, using an undeclared variable, using a keyword for the name of a variable.

A run-time error is an error that occurs when a program is running. Numeric overflow and division by zero are examples of run-time errors.

Q. Mention any two operating systems commonly used.

Solution: Commonly used two operating systems are:

- Windows-XP or Windows 2003 server and
- Linux or Unix

Q. Which translator reads an entire program written in a high-level language and converts it into machine language?

Soution: ***Compiler*** is a translator which reads an entire program written in a high-level language and converts it into machine language.

Q. Explain the differences between:

Third generation and Fourth generation languages.

Solution:

3GL's supply a few small operations which can be combined with great flexibility to accomplish almost anything. The cost, of course, is in time-consuming detail in which each and every bit of code must be cast.

3GL or third-generation language is a “high-level” programming language, such as PL/I, C or Java. A compiler converts the statements of a specific high-level programming language into machine language.

3GL language requires a considerable amount of programming knowledge. Conversely, 4GL application generators include meta-operations which provide much more functionality for a gi How does the type ven amount of code.

4GL or fourth-generation language is designed to be closer to natural language than a 3GL language. Languages for accessing database are often described as 4GLs.

Q. What is top-down design? Why should a program be documented? What do you meant by bug and debugging?

Solution:

Top-down design is the technique of breaking down a problem into various major tasks needed to be performed. Each of these tasks is further broken down into separate subtasks, and so on till each subtask is sufficiently simple to be written as a self-contained or procedure module. The entire solution of the problem will then consist of a series of simple modules.

Top-down approach is made use of in the system analysis and design process.

The top-down approach, starting at the general levels to gain an understanding of the system and gradually moving down to levels of greater details is done in the analysis stage. In the process of moving from top to bottom, each component is exploded into more and more details.

Thus, the problem at hand is analyzed or broken down into major components, each of which is again broken down if necessary.

A program should be documented as it provides the information that one needs in order to use a program. This information includes a number of different items. These items are:

a. A description of the application area of the program or what will this program do?
b. A description of the input data that the program requires.
c. A description of the output produced by the program.
d. A description of the commands needed to start the program.
e. A description of the kinds of interactions that are possible with the program.
f. An explanation of all the messages that the program can produce.
g. A discussion (in non-technical terms) of the performance capabilities and limitations of the program.

So long as computers are programmed by human beings, computer programs will be subject to errors. Programming errors are known as bugs and the process of detecting and correcting these errors is called debugging. In general, testing is the process of making sure that the program performs the intended task, and debugging is the process of locating and eliminating program errors.

Q. What is top-down program design? Explain the needs of documentation.

Solution:

Top-down design is the technique of breaking down a problem into various major tasks needed to be performed. Each of these tasks is further broken down into separate subtasks, and so on till each subtask is sufficiently simple to be written as a self contained or procedure module. The entire solution of the problem will then consist of a series of simple modules.

Top-down approach is made use of in the system analysis and design process.

The top-down approach, starting at the general levels to gain an understanding of the system and gradually moving down to levels of greater details is done in the analysis stage. In the process of moving from top to bottom, each component is exploded into more and more details.

Thus, the problem at hand is analyzed or broken down into major components, each of which is again broken down if necessary.

The needs of documentation are:

a. To help in the design of the system by working to a set of standards and having available a clear description of the work done so far. The documentation needs to be kept up-to-date throughout the project.
b. Good documentation ensures that everyone involved in the system (system designers, programmers, and users) fully understand how their aspect of the system will work. For example, what data will be inputted and how, and what information will be available from the system. This allows any misunderstandings or disagreements to surface before they become deeply entrenched in the system.
c. To ensure that the system can be maintained after completion even through the programmers involved in the original design or programming of the system may not be available. It is essential that proper documentation is kept to enable a newcomer to make necessary corrections, alterations or enhancements.

&&&&&&&&&&