• Chapter 08 : Getting Started With C

8.1 Overview

Q : 08-01-01 : Define Computer Program, High Level Programming Language and Low Level Programming Language ?

Answer :

Computer Program : A computer is a device that follows the instructions given to it. [A well-defined set of instructions given to the computer is called a computer program.

High Level Programming Language : A computer program is written in a programming language. [A computer programming language that describes the set of statements / commands / instructions nearest / similar to written English language]. The examples are Pascal, Ada, Small Talk, C, C++, Java etc.

Low Level Programming Language : In starting years (1940s and 1950s) computer programs were written in machine language, this was very difficult and time taking for programmers. Later on, Assembly language was introduced that used pneumonics (understandable names for various instructions), and was thus comparatively easier and time saving for programmers.

Q : 08-01-02 : Describe C Language and give its short History ?

Answer :

C Language : Since the emergence of computer, many programming languages have been developed but the effect of C on the computer world is everlasting. The C programming language was developed by Dennis Ritchie in 1972 at AT&T Bell Laboratories.

History : The C programming language was developed by Dennis Ritchie in 1972 at AT&T Bell Laboratories. It was derived from an earlier programming language named B. The B was developed by Ken Thompson in 1969-70 and provided the basis for the development of C. The C was originally designed to write system programs under UNIX® Operating System. Over the years its power and flexibility have made it popular in industry for a vide range of applications. The earlier version of C was known as K&R (Kernighan and Ritchie) C. As the language further developed, the ANSI (American National Standards Institute) developed a standard version of the language known as

ANSI C.

Writing Program in C : Writing a program in C is not too difficult; however it requires a good understanding of the development environment of C language. The programmer should also have the knowledge of steps required to prepare a C program for execution. As a first step, install a compiler for the C language on the computer so that the source program can be compiled, and executed. Many compilers for C language are available from number of vendors. Any of them can be used, but most commonly used is Turbo C++.

8.2 Developing A C Program (A Stepwise Approach)

Q : 08-02-01 : Explain C Program Development Process with Diagram ?

Answer :

C Program Development Process :

Turbo C++ (A Compiler for the C language) : Turbo C++ is a Borland International's implementation of a for C language. In addition to a compiler, TC provides a complete (Integrated Development Environment) to create, edit and save programs is called TC editor. It also provides a powerful debugger that helps in detecting and removing errors in the program. Once the TC (Turbo C) has been installed, it is very easy to write C programs in its editor. The IDE can be invoked by typing tc on the prompt or by double clicking the TC shortcut. The menu bar of the IDE contains menus to create, edit, compile, execute (Run) and debug program. A menu can be opened by either clicking the mouse on it or pressing the first highlighted character of the name of the menu in conjunction with the Alt key.

Creating and Editing a C Program : Open the edit window of the Turbo C++ IDE, select File/New option from menu bar. This window has a double-lined border, and a cursor inside the window represents the starting point to write a program. We can expand this window by clicking the arrow in the upper right corner, or by selecting Windows / Zoom from the menu bar. We can also navigate through the program by using the vertical and horizontal scroll bars or by using arrow keys.

Saving a C Program : After writing the C program, we should save it on the disk. This can be done by selecting File / Save command from the menu bar or pressing the F2 key. Type the name of the file in dialog box and press the Enter key. The default path for saving the file is BIN folder. The TC assigns a default name NONAME00.CPP to the file. To save the file in a specific folder / location with a different file name, one has to specify the absolute path.

Note : Turbo C++ is a compiler for C++ programming language - an extension to C. Therefore it can compile programs of both C and C++. When we save a program with .cpp extension, it can use many additional features that are not supported in ANSI C. When a program is saved with .c extension, the Turbo C++ compiler restricts it to only use standard features of C.

Compiling a C Program : The computer does not understand source program because instructions in the program are meaningless to the microprocessor, as it understands only the machine language. A program that is to be executed must be in the form of machine language. C compiler translates the source program into an object program with .obj extension. To invoke Turbo C++ compiler, select Compile / Compile from the menu bar or press Alt + F9 key. If there is no error in the source program, the program will be translated to object program successfully otherwise, the compiler will report errors in the program.

Source Program : The program written in any high level programming language, such as C, is called source program.

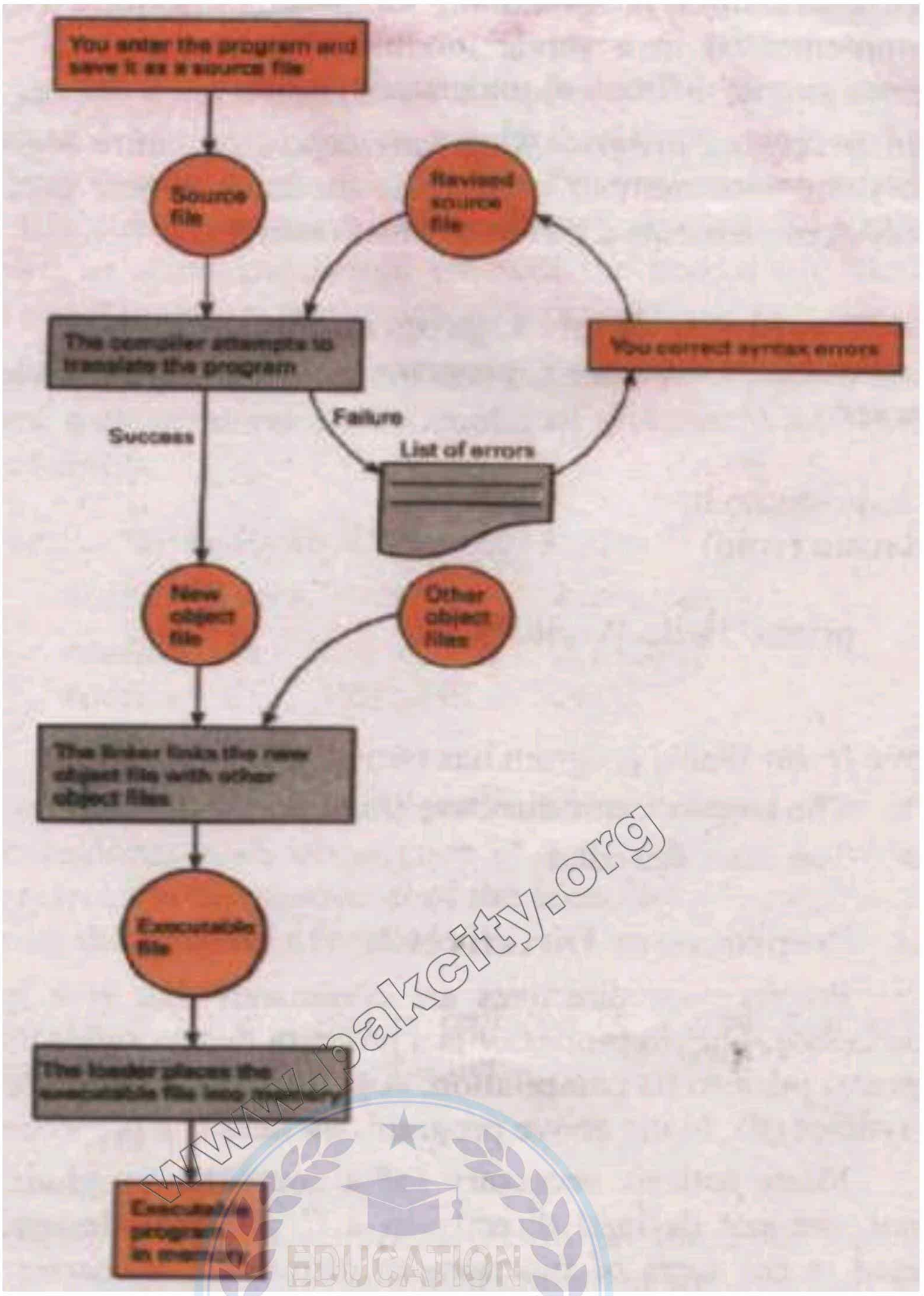
Object Program : The compiler produces an object program from the source program.

Linking a C Program : While writing a C program, the programmer may refer to many files to accomplish various tasks such as input / output etc. In case of C language, a lot of functionality is available in the form of library files. Rather than reinventing the wheel, most of the times we prefer to use the built-in functionality of

the language. Such files are needed to be linked with the object file, produced by the compiler, before execution of the program. Linking is the process in which the object file produced by the compiler is linked to many other library files by the linker. The linker is a program that combines the object program with additional object files that may be needed for the program to execute and save the final machine language program as an executable file on disk. In Turbo C++, the linker can be invoked by selecting Compile / Link from the menu bar. The Linker combines different library files to the object file and produces an executable file with .exe extension.

Executing a C program : After successfully compiling and linking the program, we are now ready to execute it. For execution the program must be loaded into memory. This is done by the loader. Loader is a program that places executable file in memory. In Turbo C++, this is done by selecting Run / Run from the menu bar or pressing Ctrl+F9 key. When the program is run, the screen flickers for a moment and the output screen will disappear in a flash. To see the program's output select Window / User Screen or press Alt+F5. The normal DOS output screen will appear. Flowchart 8.3 describes the steps required to prepare a C program for execution.

Setting the Output and Source Directories : By default, Turbo C++ places the object and executable files in the BIN subdirectory of the TC directory. This is not the right place to put these files. These files should be placed in the same directory where the source file (with .c extension) was created. To do so, select the Option / Directories from the menu bar. A window appears with four fields captioned Include Directories, Library Directories, Output Directories and Source Directories. The Include Directories field should already be set to drive:\TC\INCLUDE and Library Directories should be set to drive:\TC\LIB, where the drive: is the drive in which the directory TC is placed. It can be C, D, or E etc. We need to set the output directory field to source file directory e.g., D:\MyPrograms etc. this is where the compiler will put .obj file and the linker will put .exe file.



8.3 Basic Structure Of A C Program

Q : 08-03-01 : Describe Unstructured and Structured Programming Languages ? **Answer :**

Unstructured Programming Languages : The entire logic of the program is implemented in a single module (function), which causes the program error prone, difficult to understand, modify and debug.

Structured Programming Languages : The entire logic of the program is divided into number of smaller modules, where each module (piece of code) implements a different functionality.

Q : 08-03-02 : Briefly Explain C Program Structure with Example ?

Answer :

C Program Structure : The structure of a C program is very flexible which increases the power of the language. C is a structured programming language; therefore it provides a well- defined way of writing programs. A C program is combined with many other files before execution. The linker does this job. But we have to specify these files to be linked. We understand the basic structure of the C program with the help of example :

Hello World - A simple C program : A simple C program that displays the phrase Hello World ! on the screen.

```
#include<stdio.h>
void main (void)
{
        printf(”Hello World!”) ;
}
```

The above Hello World program has two parts :

The preprocessor directive : #include<stdio.h>

The main function.

Preprocessor Directives : Preprocessor directives are commands that give instructions to the C preprocessor. The preprocessor is a program that modifies the C program (source program) prior to its compilation. A preprocessor directive always begins with the symbol (#). In the above programs include is a preprocessor directive. Many actions necessary for a computer program, such as input and output, are not defined directly in a C program. Instead, these actions are defined in the form of functions in different C libraries. Each library has a standard header file, which is referred to with .h extension. In the above program, the stdio.h refers to the header file containing the definition of standard input / output functions. The include directive gives a program access to a library. This directive causes the preprocessor to insert definitions from a standard header file into a program before compilation. Hence, the statement #include<stdio.h> gives the program access to standard input and output functions.

 # include Directive for Defining Identifiers from Standard Libraries :

SYNTAX : #include<standard header file>

EXAMPLE : #include<stdio.h>
#include <math.h>

The include directive tells the compiler where to find the meanings of standard identifiers (e.g., printf in the Hello World program) used in the program. These meanings are described in files called standard header files. The header file stdio.h contains information about standard input and output functions such as scanf and printf, whereas the header file math.h contains information about common mathematical functions.

#define directive : Another important preprocessor directive is #define directive. It is used to define a constant macro.

#define Directive for Defining Constant Macros :

```
#define      Macro_Name          expression
#define      PI                  3.142857
#deflne      SECTOR_PER_HOUR     3600
```

The expression may be constant, arithmetic expression or a string. C preprocessor replaces each occurrence of the identifier Macro_Name with value of expression. The expression of the identifier Macro_Name can not be changed during the program execution.

Constant Macro : It is a name that is replaced by a particular constant value before the program is sent to the compiler.

FUNCTION main : The main is the function where the execution of the C program begins. Every C program has a main function. The rest of the lines of program forms the body of the main function, the body is enclosed in braces { and }. C programs are divided into units called functions. This division is usually done on the basis of functionality, where every function carries out a single task. However, it is not necessary to divide every program into functions. The same functionality may be achieved through a single function. But, every C program must have the function main as the execution of the program starts from there. The main function is actually the entry point of the C programs.

main Function Definition :

SYNTAX :

```
void main (void)
{
    body of main function.
}
```

In algebra, every function returns a single value and may accept one or more arguments (parameters). There is some resemblance between an algebraic function and the main function. The definition of the function main starts with a reserved word void. This void represents the data type of the value that is returned by the function, which means the function main returns nothing. The second void enclosed in parenthesis describes that the function main does not accept any argument. However arguments can be passed to the main function and it can also return a value. Body of the function (enclosed in braces) consists of C language statements, which are used to implement the program logic. There are many types of C statements that help programmers to write C programs.

Delimiters : Next to the function definition are braces, which indicate the beginning and end of the function body. These braces are called delimiters. The opening brace { indicates the beginning of a block of code (set of statements) while the closing brace } represents the end of a block of code.

Statement Terminator : Every statement in a C program terminates with a semicolon (;). If any of the statement is missing the statement terminator, the compiler will report it. Always be careful about the semicolon while writing C program statement.

Function printf : The last statement in the Hello World program is printf function. It is used to display the output of the program on the screen.

8.4 Common Program Errors

Q : 08-04-01 : Describe Common Programming Errors ? Briefly explain various types of Programming Errors ?

Answer :

Common Programming Errors : The programmer may come across errors while writing a computer program. In programming languages, these errors are called "bugs", and the processing of finding and removing these bugs is called debugging. When the C compiler detects an error, it displays an error message describing the cause of the error. There are three types of programming errors :

Syntax Errors : A syntax error occurs when the program violates one or more grammar rules of C language. The compiler detects these errors as it attempts to translate the program. If a C statement has syntax error, it can not be translated and the program could not be executed.

There can be many causes of syntax errors, i.e., missing statement terminator (semicolon), using a variable without declaration, missing any of the delimiters.

Runtime Errors : A runtime error occurs when the program directs the computer to perform an illegal operation, such as dividing a number by zero. Runtime errors are detected and displayed by the computer during the execution of a program. When a runtime error occurs, the computer stops executing the program and displays a diagnostic message.

Logical Errors : Logical errors occur when a program follows a faulty algorithm. The compiler can not detect logical errors; therefore no error message is reported from the compiler. Moreover, these errors don't cause the program to be crashed, that's why these are very difficult to detect. One can recognize logical errors by just looking at the wrong output of the program. Logical errors can only be detected by thorough testing of the program.

8.5 Programming Languages

Q : 08-05-01 : Describe the two broad categories of Programming Languages ?

Answer :

Programming Languages : These are used to write computer programs. There are two broad categories of programming languages :

Low Level Languages : Low level languages are divided into two broad categories i.e., machine language and assembly language. Machine language is the native language of the computer. The computer does not need any translator to understand this language. Programs written in any other language must be converted to machine language so that the computer can understand them. Every machine language instruction consists of strings of binary 0s and 1s. As it is very difficult for human beings to remember long sequences of 0s and 1s, therefore writing programs in machine language are very difficult and error prone. So, it was thought to replace the long sequences of 0s and 1s in machine language with English like words. This idea provided the basis for the development of assembly language. In assembly language, machine language

instructions (long sequences of 0s and 1s) are replaced with English like words known as mnemonics (pronounced as Ne-Monics). An assembler (language translator for assemb1y language programs) is used to translate an assembly language programs into machine language.

High Level Languages : Programming languages whose instructions resemble the English language are called high level languages. Every high level language defines a set of rules for writing programs called syntax of the language. Every instruction in the high level language must confirm to its syntax. If there is a syntax error in the program, it is reported by the language translator (compiler or interpreter). The program does not translate into machine language unless the error is removed.

Common high-level languages include C, C++, Java, Pascal, FORTRAN, BASIC, and COBOL etc. Although each of these languages was designed for a specific purpose; all are used to write variety of application software. Some of these languages such as C and C++ are used to write system software as well.

Q : 08-05-02 : Explain Common Characteristics of High Level Programming Languages ?

Answer :

Common Characteristics of High Level Programming Languages

Each of these languages has some advantages and disadvantages over the other e.g., FORTRAN has very powerful mathematical capabilities while the COBOL is ideal for writing business applications, C and C++ are very handy for writing system software while Java is equipped with strong network programming features. Besides having different features, all high level programming languages have some common characteristics are :

English Like Languages : These are English like languages, hence are close to human languages and far from the machine language and are very easy to learn.

Easy to Modify and Debug : Programs written in high level languages are easy to modify and debug, and more readable.

Concentrate on Problem : These languages let the Programmers concentrate on problem being solved rather than human-machine interaction.

Well Defined : These describe a well defined way of writing programs.

Understanding The Machine Architecture : These do not require a deep understanding of the machine architecture.

Machine Independence : High level languages provide machine independence. It means programs written in a high level language can be executed on many different types of computers with a little modification. For example, programs written in C can be executed on Intel® processors as well as Motorola processors with a little modification.