UNIT – II

6

CHAPTER

INTRODUCTION TO C

6.1 HISTORY OF C

C is a powerful, portable and elegantly structured programming language. It is an extremely popular language because it is simple, efficient, and reliable. C is considered to be a middle level language since it has the features of both low-level language and high-level language.

Combined Programming Language (CPL) was developed at Cambridge University in 1963, to solve different types of problems on various hardware platforms. However, it was too complex, hard to learn and difficult to implement. In 1967, Basic CPL, a subset of CPL, was developed by Martin Richards for writing system software. It incorporated only the essential features of CPL, but it was also not found to be sufficiently powerful. In 1970, a language using many features of BCPL was created by Ken Thompson and it was simply called as B. B was to create early versions of UNIX operating systems at Bell Laboratories. Incorporating the best features of both BCPL and B languages, Dennis Ritchie developed C language in 1972, at AT & T Bell Laboratories. It was named as C to present it as the successor of B language.

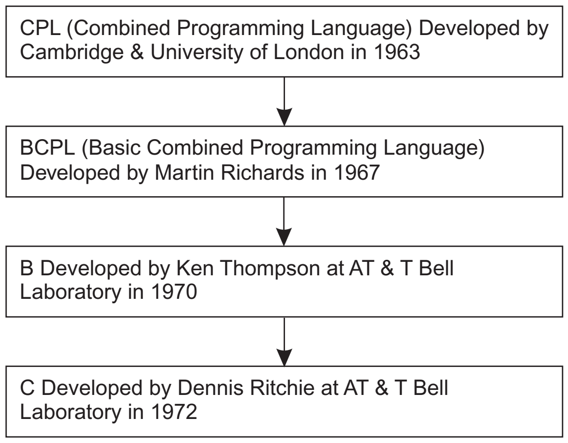

CPL (Combined Programming Language) Developed by Cambridge & University of London in 1963
BCPL (Basic Combined Programming Language) Developed by Martin Richards in 1967
B Developed by Ken Thompson at AT & T Bell Laboratory in 1970
C Developed by Dennis Ritchie at AT & T Bell Laboratory in 1972

Fig. 6.1 *Various stages in Evolution of C language*

6.2 FEATURES OF C LANGUAGE

C language has been developed for implementing various systems such as Operating Systems, Compilers, Linkers, Word Processors and Utility Packages. C language is faster than BASIC language.

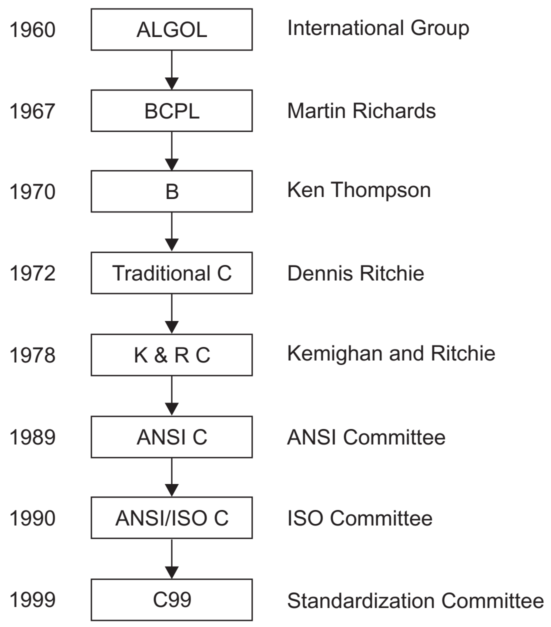


Fig. 6.2 *History of ANSI C*

For example, a program to increment a variable from 0 to 15000 takes one second in C while it takes more than 50 seconds in an interpreter of BASIC. There are only 32 keywords in ANSI C and its strength lies in its built-in function.

The various other features of C include the following:

Portable*:* Portable feature of C states that the C program written for one computer can be executed on another computer with either little or no modification.

Structured: In C, the problem has to be represented in terms of function modules and blocks. Program debugging, testing, and maintenance become easier because of this modular structure.

Extensible: In C, own functions can also be added to the existing C library which makes it extensible.

Flexible: C language can be used for system programming like UNIX OS, C compiler as well as for application programming.

Robust: The rich set of built-in functions and operators that are available in C helps the programmer to write any complex program.

C compiler combines the capabilities of an assembly language along with high-level language features. Therefore both system software and business packages can be developed using C.

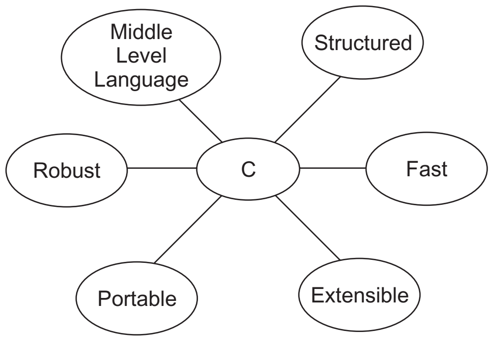


Fig. 6.3 *Features of C Language*

6.3 CHARACTERISTICS OF C

1. It is a highly structured language.
2. It uses features of high level language.
3. It can handle bit-level operations.
4. It is highly portable as it is a machine independent language.
5. A variety of data types and a powerful set of operators are supported by C.
6. Using the concept of pointers, it supports dynamic memory management.
7. It enables implementation of hierarchical and modular programming with the help of functions.
8. It can extend itself by addition of functions to its library continuously.

Sample Program 1:

```
main()
{
/*….print……*/
printf("hai");
/*…..print ends….*/
}
```

In the above program, the first line starts with **main( )** which is a special function and it states that the execution begins at this line. Every program must have exactly one main function, else the compiler can't understand the beginning of the program.

The empty pair of parentheses ( ) indicates that the main function has no arguments or parameters. The opening braces '{' mark the beginning of main function and closing braces '}' mark the end of main function. The function body is defined by the statements between these two braces. The set of instructions to perform the task is defined inside the function body.

In the above example, there are three statements inside the function body out of which only the printf line is an executable statement. The readability and understanding of the program is enhanced using the comment line /*…….*/. Comment lines are not executable and therefore anything written between it is ignored by the compiler. It can be inserted at the beginning, middle or end of line wherever blank spaces can occur but never in the middle of the word. Also comment lines can never be nested.

The comment line

/*……./*……*/………..*/

is invalid and therefore results in an error.

The comment line serves as an aid in debugging and testing and helps in understanding the various functions and operations of a program.

In the above example, **printf** is the only executable statement of the program which is a predefined function for printing output. Predefined function is defined as a function that is already been written and compiled and linked together with the program at the time of linking. The **printf** statement prints out everything that is given between the starting and ending quotation. On executing the above program, the following output will be obtained:

hai

Every statement in C should end with a semi colon. To print in two lines, it should be given as

```
printf("hai\n");
printf("How are you?");
```

The information contained between the parenthesis is called the arguments of the function. These arguments are simply strings of characters to be printed out.

Newline operator '\n' is used to print in two lines. It instructs the computer to go to the next new line. No space is allowed between \ and n.

Types of main function:

C permits different forms of main statement.

1. **main( )** main( ) indicates that the function has no argument. This may be explicitly indicated by using keyword 'void' inside the parenthesis, i.e., main (void).
2. **int main ( )** In this, function returns an integer value, i.e., the last statement in the program must be "return 0".
3. **void main ( )** The keyword void means that the function does not return any information to the operating system.

6.4 BASIC STRUCTURE OF C PROGRAM

C programs can be viewed as a group of building blocks called as functions. A function is defined as a subroutine which may include one or more statements that are designed to perform a specific task. C program may contain one or more sections as shown in figure below:

Documentation Section The name of the program, author and other details are defined in this section as a set of comment lines

Link Section This section provides instructions to the compiler to link functions from the system library.

Definition Section This section defines all symbolic constants.

Global Declaration Section Some variables are used in more than one function. Such variables are called as global variables and are declared in the global declaration that is outside of all functions. All user defined functions are also declared in this section.

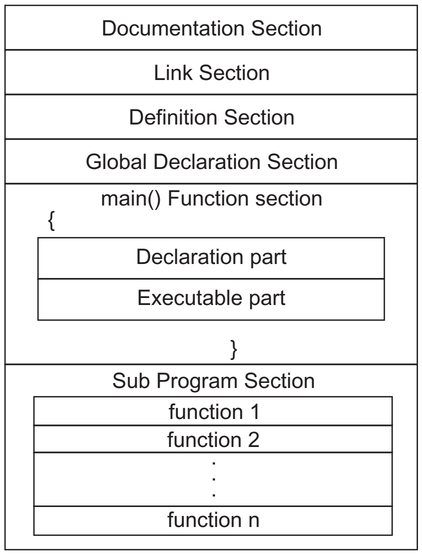


Fig. 6.4 *An overview of C Program*

main ( ) Every C program must have one main( ) function section. This section contains two parts, namely declaration part and executable part. All the variables to be used in the executable part are declared in the declaration part. Executable part contains at least one executable statement. Declaration and executable parts must appear between the opening and closing braces. The opening brace indicates the beginning of the program execution and the closing brace denotes the end of the program. The logical end of the program is denoted by closing braces. All statements in the declaration and executable part must end with semicolon.

Subprogram Section All the user defined functions that are called in the main function are defined in this section. Immediately after the main function, user defined functions are generally placed although they may appear in any order. All sections except the main function section may be absent, when they are not required.

Programming Style C programs have to be written in lowercase letters. Uppercase letters are used only for symbolic constants. Braces are used to group the statements together and mark the beginning and end of the function. A program is made easier to read and debug by using proper indentation of braces and statements.

Since C is a free-form language, the statements can be grouped together on one line. For example,

```
x=y;
x=x+2;
z=x+1;
```

can be written in one line as

```
x=y;x=x+2;z=x+1;
```

However, this style makes the program more difficult to understand and should not be used.

Programming Rules:

1. All statements must be in lower case letters.
2. Blank spaces may be inserted between the words.
3. The opening and closing braces must be balanced.
4. The statements can be written anywhere between the opening and closing braces.
5. Every statement must be terminated by a semicolon (;).

6.5 EXECUTING A ‘C’ PROGRAM

C program is executed in a series of steps which involves the following:

1. Program creation.
2. Program compilation.
3. Linking the program with functions that are needed from the C library and
4. Program execution.

The process of creating, compiling and executing a program is shown in the following figure. The steps for compiling the program remain the same irrespective of the operating system though the system commands for implementing the steps and the convention followed for naming the files may vary on different systems.

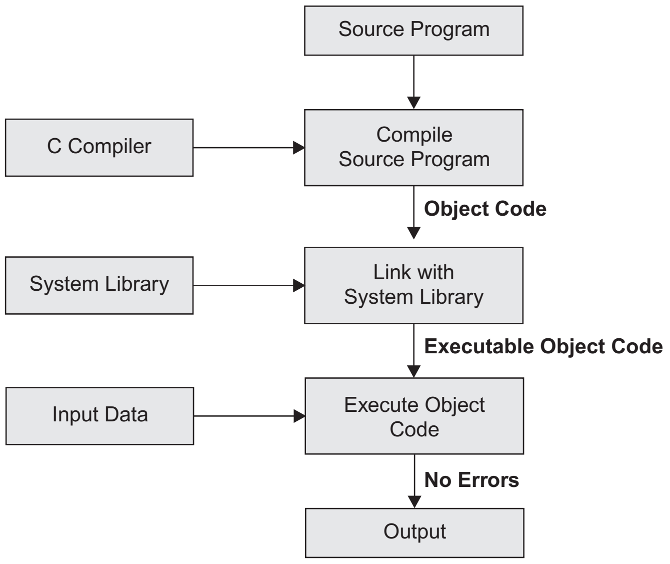
1. Program Creation: The program that is to be created is entered into a file. The file name can consist of letters, digits and special characters, followed by the extension **.c**. Examples of valid file names are:

```
a.c
xyz.c
```

2. Program Compilation: During the compilation process, the source program instructions are translated into a form that is suitable for execution by the computer. The translation process checks each and every instruction for correctness and if no errors are reported then it generates the object code. During the linking process, the other program files and functions that are required by the program are put together with it.

During this process, if any mistake in the syntax and semantics of the language is discovered, then the errors will be listed out and the compilation process ends here. The errors should be corrected in the source program with the help on an editor and the program is compiled again.

3. Program Execution: During execution, the executable object code is loaded in the computer memory and the instructions will be executed. During this process, the program may request some data through the keyboard.


Source Program
C Compiler
Compile Source Program
Object Code
System Library
Link with System Library
Executable Object Code
Input Data
Execute Object Code
No Errors
Output