INTRODUCTION TO C AND BASIC STRUCTURE OF C PROGRAM

CHAPTER - 7

OBJECTIVES OF THIS CHAPTER

7.1 INTRODUCTION AND HISTORY OF C

C is a general-purpose programming language. C can be used to develop any type of application programs. We can develop Business applications, Scientific applications etc. using it. It can also be used to develop System programs such as Operating Systems, Language Translators etc. Thus, we can say that it is useful for developing both types of software, i.e. System Software and Application Software.

In 1960, many computer programming languages were emerged like FORTRAN, COBOL etc. But, these languages were used for specific applications. For example: FORTRAN (FORmula TRANslation) was used to develop Scientific Applications only, while COBOL (COmmon Business Oriented Language) was used to develop Business Applications only. Later an international committee was set up to develop a general purpose programming language. As a result, in 1963, Combined Programming Language (CPL) was developed at Cambridge University. It was hard to learn and difficult to implement. So, later in 1967, BCPL (Basic Combined Programming Language) was developed by Martin Richards at Cambridge University. Similarly, B language was developed by Ken Thompson at AT & T's Bell Laboratory in 1970.

But both of these languages, BCPL and B, are type-less languages and too specific. Finally, in 1972, using many important ideas of BCPL and B languages, Language C was developed by Dennis Ritchie at AT & T's Bell Laboratories of USA and it became a general-purpose programming language.

7.2 WHY C-LANGUAGE IS CALLED MIDDLE LEVEL LANGUAGE?

All the programming languages can be divided into two types: Low Level Programming Languages and High Level Programming Languages.

Low Level Languages are known as Machine-Oriented Languages because programmers have to concentrate more on the architecture of underlying hardware (machine) rather than on the logic of the program to be solved. It is difficult to learn & use these programming languages. These languages are used to write machine dependent system programs such as operating system, translators and so-on. Examples of these languages are Assembly Language and Machine Language.

High Level Languages are known as Problem-Oriented Languages because programmers have to concentrate more on the logic of the program rather than the hardware architecture of the machine. It is easy to learn and use these programming languages. These languages are used to write any type of application programs, such as business applications, scientific applications etc. Examples of these languages are FORTRAN, COBOL, Pascal etc.

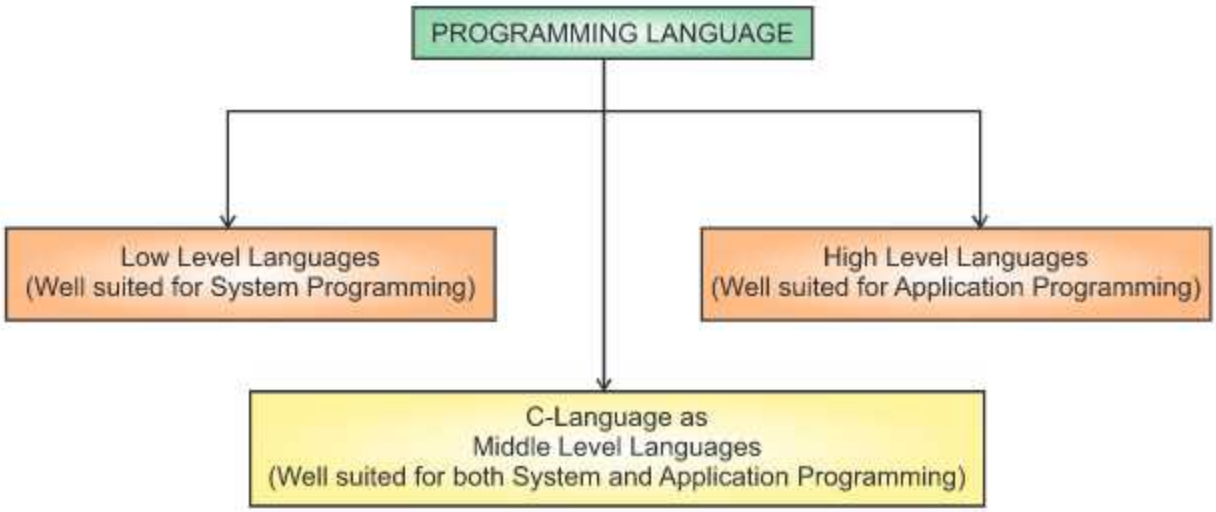


Fig. 7.1 Why C is considered specially as a Middle Level Lanaguage

C Language has the capabilities of these both types of programming languages, i.e. Low Level and High Level programming languages. It means C language is well suited for writing both system programs and application programs. Hence, C is a programming language which stands in between the above two types of programming languages. So C is called Middle Level Language.

Though, there is no special category of Middle level Programming language, it is only due to the special features of C language why it is designated as Middle Level Programming Language.

7.3 INTRODUCTION TO C EDITORS & IDEs

Editors are the programs that are useful for writing source codes of languages. There are many programming languages which have their own editors to write programs, such as C, C++ etc. But some programming languages do not have their own editor for writing programs, such as Java. Java Programming can be written in any simple text editor program. After writing source programs in programming languages, they must be compiled in order to execute by the computer systems. These source programs must be compiled by their respective Compilers.

Many programming languages also support the IDEs. IDE stands for Integrated Development Environment. **Integrated Development Environment (IDE)** can be defined as software that gives its users an environment for writing programs (Editor), along with tools for compiling, executing, testing and debugging the programs. Usually,Modern IDE Software are very user-friendly. They provide an easy-to-use interface. These IDE interfaces provide suggestions for syntax to programmers, the graphical user interface having buttons and menus to interact with, editors and plugins and many other features.

There are many IDEs that are available for programming in C. Some of these common IDEs are given below:

- Turbo C
- Code Blocks
- Eclipse
- Code Lite
- Net Beans
- Dev C++ etc....

Turbo C is one of the oldest IDE for programming in the C language. It was developed by Borland and first introduced in 1987. At the time, Turbo C was known for its compact size, comprehensive manual, fast compile speed and low price. But now a days, Turbo C was getting out-dated due to the advancements in the newer Operating Systems, such as Windows 7,

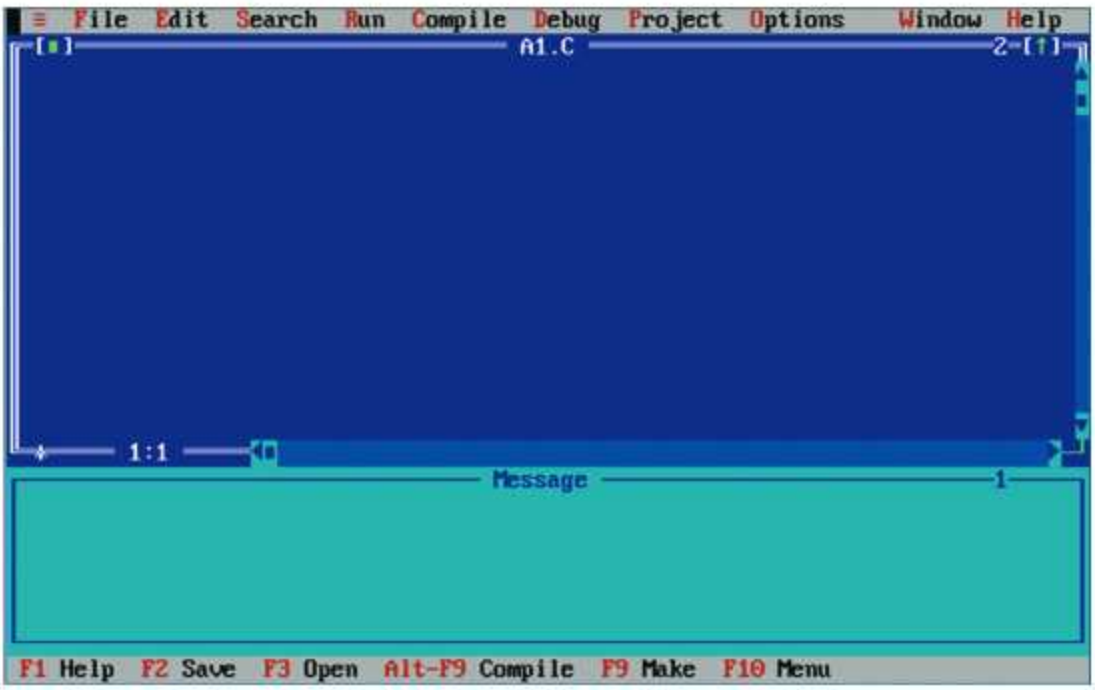


Fig: 7.2 Turbo C/C++ IDE Interface

Windows 8 and Windows 10. In these operating systems, programmers have been facing many issues while they make C programs in Turbo C due to compatibility issues. Many modern IDEs have been developed for programming in C which works well in modern operating systems.

Code::Blocks is also a popular modern IDE for programming in C/C++. It is a free (Open Source), highly extensible and configurable, cross-platform C and C++ IDE. It offers programmers the most demanded and ideal features. It delivers a consistent user interface and feel. Most importantly, we can extend its functionality by using plugins developed by users.

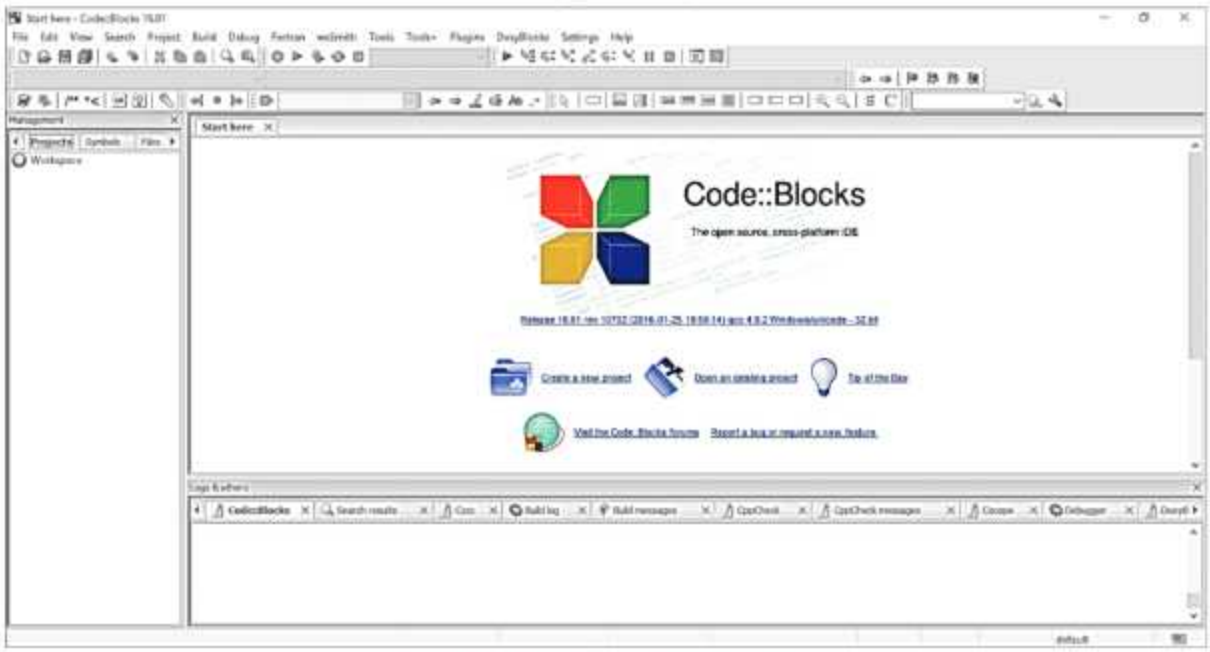


Fig: 7.3 Code::Blocks IDE interface

Code::Blocks provides many features to programmers which are given below:

- It works on Windows, Linux and Mac OS X as well.
- It supports Compiling, Debugging, Code Coverage, Profiling, Auto-completion of code
- It provides support for multiple compilers including GCC, mingw, clang, Borland C++ 5.5, etc.
- It is very fast, no need for makefiles
- It provide facilities of debugging by supporting full breakpoints including code breakpoints, data breakpoints, breakpoint conditions plus many moredisplay local functions symbols and arguments
- It is designed to be fully configurable and extensible with its plugins.
- It provides workspace that supports combining of projects
- It supports the feature of Custom memory dump and syntax highlighting
- It also provides support for code analysis, etc.

Code::Blocks IDE can be downloaded from the following link:

http://www.codeblocks.org/downloads

Any one of the C/C++ IDE, that are available in market including modern or old ones, can be used for writing C programs of this book.

7.4 CREATING AND EXECUTING C PROGRAMS

Any C program development involves the following steps:

1. Design the program
2. Write the program using any Text Editor or IDE supporting C Language
3. Save the program by giving filename with extension .c
4. Compile the program
5. It there are any errors in the program, then correct it and repeat the step 4
6. Execute the program.
7. View the Output Window

7.5 GETTING STARTED WITH C

For communication with human beings, we use natural languages like English, Hindi, and Punjabi etc. But to communicate with computers, we can use only those languages which a computer system can understand directly or indirectly. These languages are called Programming Languages. C is a programming language. As we have to learn any natural language before using it, similarly we must also have to learn programming languages like C for communication with the computers. Learning programing languages are very similar to learn any natural language like Hindi, Punjabi etc.

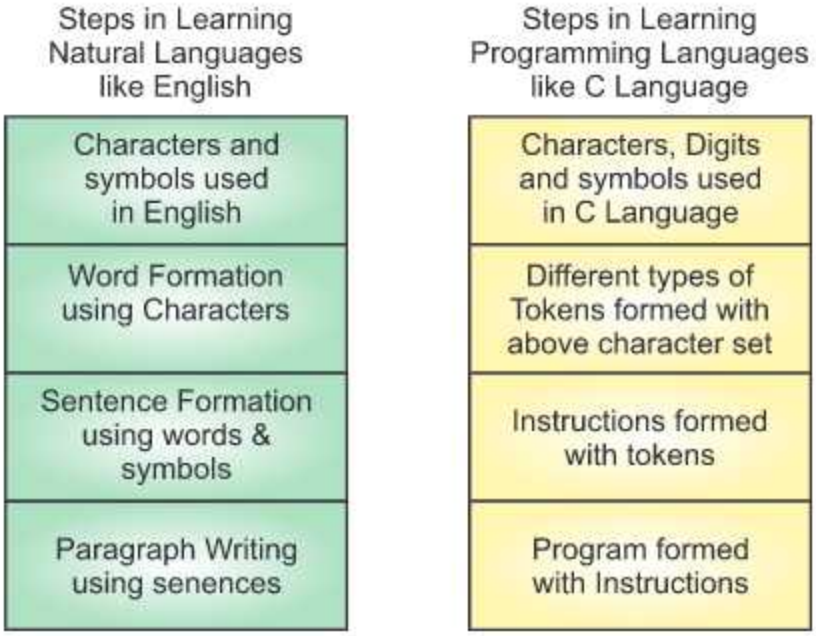


Fig. 7.4 Analogy between learning Natural and Computer Languages

There is a very close analogy between learning natural languages (like English, Hindi, Punjabi, etc.) and computer programming languages (like C, C++, JAVA, etc.). The classical method of learning any natural language (for example English) is to learn the alphabets or characters used in the language, then learn to combine these characters to form words, which in turn are combined to form sentences and sentences are combined to form paragraphs. Learning any computer programming language, for example C, is similar and much easier.

Therefore, instead of straight-away learning how to write programs, we must first know what characters, numbers, symbols are used in C, then using these how different tokens are constructed, finally how these tokens are combined to form instructions. A group of instructions would be combined later on to form a program. This analogy of learning natural and computer programming languages can be represented using figure 7.4.

7.6 CHARACTER SET

It is the first step for learning any language whether it is Natural or a Computer Programming Language. We can learn any language only if we know which characters and symbols are allowed in that language. So, before learning C, we must be familiar with the characters and symbols used in the C language. C language supports ASCII (American Standard Code for Information Interchange) character set. The ASCII set of characters includes the following characters and symbols:

- Upper and Lower case Alphabets (A to Z, a to z)
- Digits (0 to 9)
- Special Symbols (all the printable symbols present on the keyboards, For Example: ! @ # $ % ^ & * ( ) - _ + = { } [ ] ; : ' " < , > . ? / | \ etc.
- Some Non-printable characters, For example: new-line, horizontal-tab etc.

The set of these above characters and symbols is called the Character Set of C Language.

7.7 TOKENS

Tokens are like words and punctuation-marks in English language. In any programming language, like C, a program is made up of tokens. Tokens are the smallest individual units in a program. A C-program can have five types of tokens as shown below:

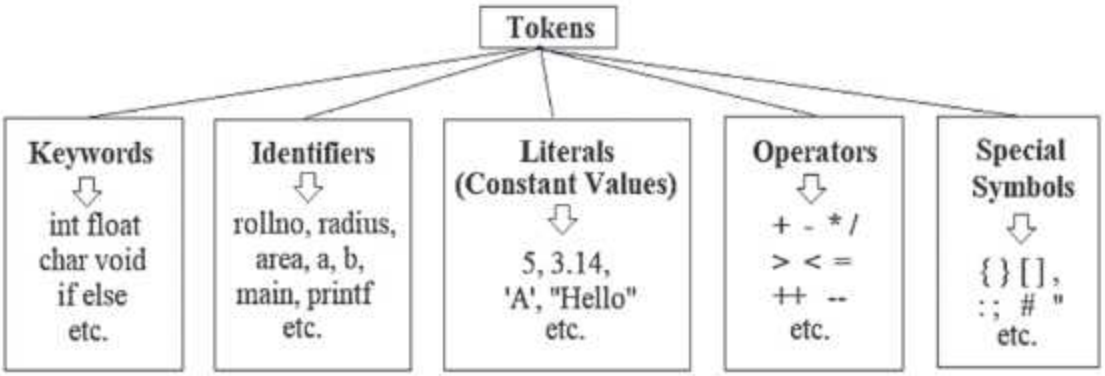


Fig. 7.5 C Tokens with Examples

Introduction of these tokens with suitable examples is given below:

7.7.1 Keywords

Keywords are also called the Reserve Words. These words are predefined in the Compiler of C Language. Meaning of these words is predefined. They are used for special purposes for which they had been defined. We cannot change their meaning. In Turbo C, these words are shown in white colour while in code::blocks these words are shown in blue colour. C language has 32 keywords but some new compilers introduce some more keywords to C Language. Following is the list of 32 keywords that are supported by all the compilers of C language:

Table : 7.1- List of C Keywords

auto	const	double	float	int	short	struct	unsigned
break	continue	else	for	long	signed	switch	void
case	default	enum	goto	register	sizeof	typedef	volatile
char	do	extern	if	return	static	union	while

These keywords can be used wherever they are required in the program. All the keywords in C programs must be written in lower-case only. As C is case-sensitive language, so if we write these keywords in upper-case in a program, it will display compile errors. (A case sensitive language is a language which considers lower case and upper case alphabets as different elements.)

7.7.2 Identifiers

Identifiers are the name given to elements of program such as variables, constants, arrays, functions, structures etc. Every program element must be named to distinguish it from other elements. The name assigned to the elements should be meaningful because it facilitates easy understanding of the program elements. After naming the program elements, they can be identified by their name. For defining names of program elements, some naming rules must be followed during writing of C programs. These naming rules are given below:

- Identifier name must begin with an alphabet or underscore (_) symbol. It must not begin with a digit. For example: Identifier name 5star will be wrong because it begins with a digit 5 which is not allowed.
- No special character, except underscore (_), is allowed for defining name of program element. For example: if we define a name: roll#, it will be considered wrong and compiler will show an error because we are using # in the name which is a special character.
- Two consecutive underscores are not allowed in the identifier name. For example: Identifier name roll__no will be wrong because we use two consecutive underscores in the name which is not allowed in c programs.
- In some C language compilers (Turbo C), length of identifiers is restricted to 31 characters. It means identifier name can have maximum 31 characters and minimum

1 character. If we use more than 31 characters in the name than it will not show any error, instead the compiler considers only first 31 characters as identifier name and ignore the remaining characters.

- Keywords cannot be used as identifier names. For example: int cannot be used for defining identifier name because it is a keyword and has a special meaning.
- Identifier names are case-sensitive. It means identifier names in lower-case and upper-case are considered distinct. So, we should take care of lower and upper cases while defining names. For example: roll and ROLL will be considered two different identifiers in C programs.
- Blank spaces in the identifier names are not allowed. For example: identifier name roll no will be wrong because it contains a blank space between the words roll and no.

7.7.3 Literals

Literals are also called constant values. These are the fixed values that are normally assigned to variables in the C-programs. These fixed or constant values can be categorized into two categories: Numeric and Character constant values as shown in the following diagram with suitable examples:

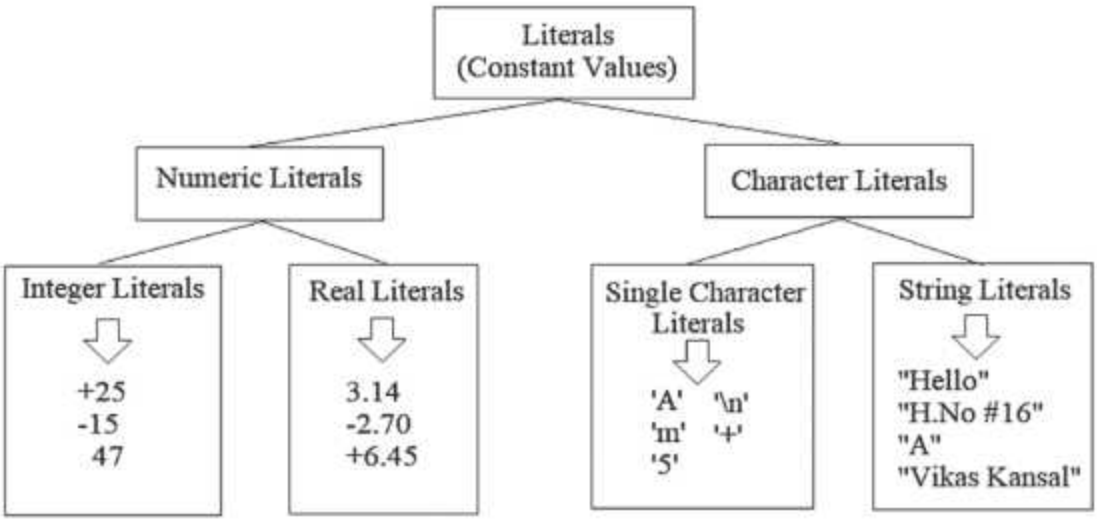


Fig. 7.6 Different Types of Literals (Constant Values)

7.7.3.1 Numeric Literals : These are the numeric values, which can be used for numeric calculations. There are two types of numeric literals:

- **Integer Literals :** These are literals without fractional(decimal) part. It consists of digits from 0 to 9 along with positive (+) or negative (-) sign. If number does not have any sign, it is considered the positive integer literal. For example: 56, +26, -96 etc. are the integer literals.
- **Real Literals :** These are literals with fractional(decimal) part. It consists of digits from 0 to 9 along with positive (+) or negative (-) sign. It also has a decimal point (.) which separates the integral and fractional part of the real number. If integral part

does not have any sign, it is considered the positive real literal. For example: 3.14, +256.5896, -96.14. 36.00 etc. are the real literals.

7.7.3.2 Character Literals : These are the character values, which usually do not involve in the calculations. There are two types of character literals:

- **Single Character Literals :** These are the literals which have a single character enclosed in the single quotes. More than one printable character is not allowed in these literals. Non-printable characters also come in this category of constant literals though two symbols are used in these characters, for example: new line character (\n - backslash and a letter n), but yet they are considered to be a single character. Examples of single character literals are: 'A', 'g', '7', '+', '$', '\n', '\t' etc. Values 'AB', '45' are the invalid examples of single character literals because more than one character is enclosed in single quotes which are not allowed in the single character constant.
- **String Literals :** These are the literals which have one or more characters enclosed in the double quotes. These literals may have the combination of letters, digits, special symbols, and blank space. Examples of these literals are: "V. Kansal", "A", "House#196", "1829" etc.

7.7.4 Operators

Operators are the symbols which are used to perform some mathematical or logical operation. Operators are used to manipulate values/variables in the program. These values/variables are called operands. C supports a rich set of built-in operators. All these operators can be classified into three broad categories: unary, binary, and ternary. A detailed explanation has been given on operators in the next chapter of this book.

7.7.5 Special Symbols

These are the symbols used as punctuation marks. Each symbol has its different speciality in program. Each symbol is used to denote something special in the program. For example: semicolon (;) is used to terminate the statement, comma (,) is used as a separator, parenthesis ( ) are used to represent the functions, square brackets [ ] are used to denote arrays, Braces { } are used for grouping the statements etc.

7.8 VARIABLES AND CONSTANTS

These both are the important program elements which are used to store values in the program. Both are given a name and type of value to be stored in them. But there is a little difference between them. Variables allow us to change their values during execution time while constant do not. It means constants have fixed values while variables can have changeable values during program execution. C is a **strictly typed language**. It means we must declare variables and constants before using them in the program. If we use variables or constants without declaring them, compiler will generate a syntax error: "Variable not declared". So, each variable or constant must be declared in the program. Following are the syntax rules for declaring variables and constants in the program:

Syntax of variable declaration:

data_type variable_name;

Here, **data_type** tells the compiler what type of value is going to be stored in the variable, and **variable_ name** is the valid identifier which tells the compiler about the name of the variable which will be used to refer to the value stored in the variable. Consider the following example of variable declaration:

int roll_no;

Here, int represents the integer data type while roll_no is the identifier name which is used to refer to the variable. It means the variable roll_no can hold only the integer values. It is not given any value, so a garbage value will be stored in it by default. If we want to assign it a value during its declaration, then it will be called variable initialization.

For example:

int roll_no=5;

This assigned value can be changed later at any time because a variable allows us to change its values at any time during execution.

Syntax of constant declaration:

A variable can be made constant if we put a keyword const before the variable declaration and assign it a fixed value at the time of declaration. Consider the following syntax rule:

const data_type constant_name = value;

Here, const is a keyword which tells the compiler that the given value cannot be changed during program execution. Consider the following example of defining constant:

const float pi=3.14;

In this example, we define a constant value 3.14 which is of real type. It is given a name pi. The keyword float tells that the value will be of real type and const make it a fixed value that cannot be changed during execution time. If we try to change its value, compiler will generate a compile error.

7.9 DATA TYPES

Data type defines which type of data will be stored in the program elements, such as variables, constant, arrays etc. Data types define a specific type or range of values for the variables or other program elements. C is a strongly typed language therefore data type of all the variables must be declared during declaration time.

C supports many different types of data, each of which may be represented differently within the computer's memory. Storage representation of data in memory varies from machine to machine and compiler to compiler. For example: in Turbo C, int data type takes 2 bytes of memory while in Code::Blocks it takes 4 bytes of memory. Following table shows the list of primitive or basic data types available in the Standard C language:

Table 7.2: Primitive Data types available in C

Keyword	Description	Memory Requirement	Range of values	Format
char	Used to store single byte/character data	1 byte	-128 to 127	%c
int	Used to store integer type data	2 byte	-32768 to +32767	%d
float	Used to store single precision floating values	4 byte	3.4×10^{-38} to $3.4\times10^{+38}$	%f
double	Used to store double precision floating values	8 byte	1.7×10^{-308} to $1.7\times10^{+308}$	%lf
void	Used with functions which do not return any value	-	-	-

7.10 HEADER FILES IN C

C provides a huge library of predefined functions to perform various types of tasks in the programs. All these functions are called library functions. To organize these functions, all the functions are logically grouped into separate files. These files are termed as header files. All the header files have **.h** extension. To use any of these functions in our program, we have to include these header files in our program. This inclusion is done by using the **pre-processor directive #include.** In C, pre-processors begin with the # symbol. Consider the following example:

#include<stdio.h>

In this example, stdio.h is a header file and #include is a pre-processor directive. Using this example, we can use any of the function defined in the stdio.h header file in our program.

There are many header files in C. The header files that will be used in a program, depends solely on our requirement in the program. Following are some of the common header files that are used in the C programs:

- **The header file stdio.h :** The full name of this header file is standard input output header file. This file contains the functions that can be used for input and output from the standard input/output devices. For Example: scanf( ) and printf( ) functions
- **The header file conio.h :** The full name of this header file is console input output header file. Console is screen where our program executes. This file contains the functions that are used for console during input/output. For example: clrscr( ) and getch( ) functions
- **The header file math.h :** This file contains mathematical and trigonometric functions that we can use in our programs for various mathematical operations. For example: sqrt( ), pow( ), sin( ), cos() etc.

- **The header file string.h :** This file contains functions that can be used for string manipulation operations. For example: strlen( ), strcpy( ), strupr( ), strlwr( ), strcmp( ) etc.

To use any library function, we must use respective header file with pre-processor directive #include in our program.

7.11 INPUT AND OUTPUT STATEMENTS IN C

Input and Output statements provide interaction between program and users. Using input statements, user provides input to program and using output statements output is displayed by the program to user.

All the input and output operations in C program are carried out using pre-defined functions. Although, there are many formatted and unformatted input/output functions provided by the C library, but normally, this input/output operation is carried out using scanf( ) and printf( ) formatted functions in C programs. Consider the following diagram which shows the purpose of input and output statements in the programs:

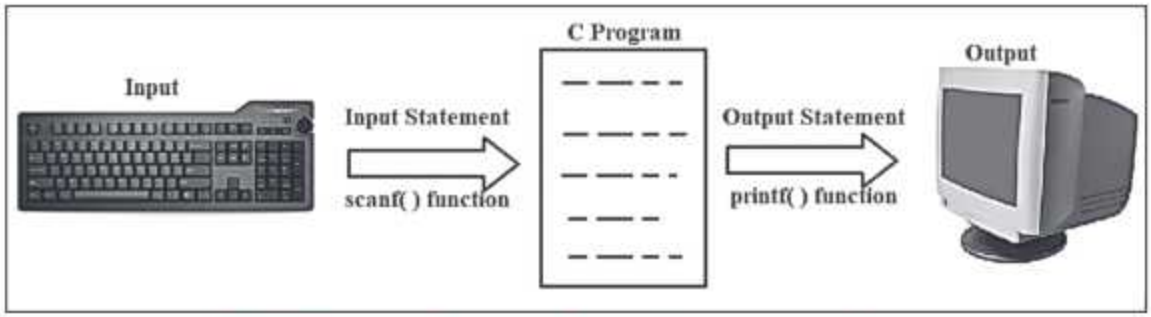


Fig. 7.7 Purpose of Input and Output Statements in a Program

Now, we shall discuss these two commonly used input and output library functions in c programs, i.e. scanf( ) and printf( ) function:

The scanf( ) function:

Input data can be entered into the program from a standard input device by using C library function scanf( ). This function can be used to enter any combination of numerical values, single characters and strings. The general syntax for using functions is as follows:

scanf("format string", &arg1, &arg2, ……., &argn);

Here, format string refers to a string that contains certain format codes depending on the data type of arguments, and arg1, arg2, ……., argn are the arguments that represent the individual input data items. Normally, arguments are the variables preceded by address operator &. This address operator & specify the address of the variable where the data will be stored. Format string and all the arguments must be separated by the comma operator. To have better understanding of the concept, consider the following example:

```
int a;
float b;
scanf("%d%f",&a,&b);
```

Here, "%d%f" is a format string which specify the type of value to receive from the keyboard. The format string %d is used to input the integer value for variable a, and %f is used to input float value for variable b. Here, a, and b are the two arguments representing variables which are getting values from user. The symbol & before these variables a, and b specifies that the integer and float values will be stored at the memory addresses allotted to a, and b.

The printf( ) function : The printf( ) function is used to display information on the monitor in C programs. This function is normally used to display simple text messages on the monitor (output) screen. The general syntax of this function for displaying simple text is as follows:

```
printf("simple text message");
```

For example:

```
printf("Hello from C Language");
```

This function can also be used to show any numerical, single character and string values stored in variables on the monitor (output) screen. The general syntax for displaying values stored in the variables is as follows:

```
printf("format string", arg1, arg2, ……., argn);
```

Here, format string refers to a string that contains certain format codes depending on the data type of arguments, and arg1, arg2, ……., argn are the arguments that represent the individual output data items. Here, the printf( ) functions read values of arguments from memory and display them on the monitor screen. Format string and all the arguments must be separated by the comma operator. To have better understanding of the concept, consider the following example:

```
int a=56;
float b=3.14;
printf("%d%f",a,b);
```

Here, "%d%f" is a format string which specify the type of value to display at the monitor screen. The format string %d is used to display the integer value stored in variable a, and %f is

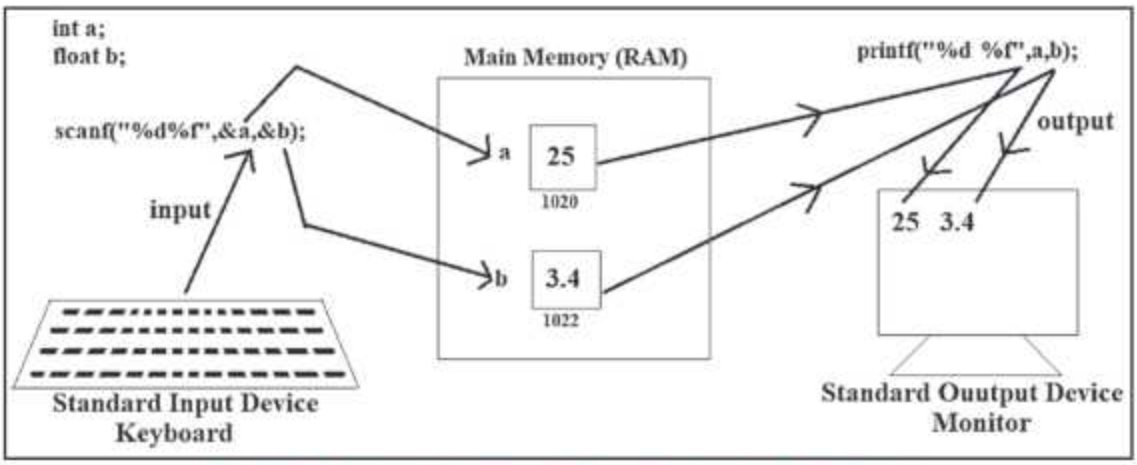


Fig. 7.8 Concept of working with scanf( ) and printf( ) function

used to display float value stored in variable b. Here, a, and b are the two arguments representing variables which are displaying values to user at monitor screen.

Figure 7.8 tries to illustrates the concept of working with scanf( ) and printf( ) functions in C programs:

Functions scanf( ) and printf( ) are the most important and useful functions that are widely used in any C program. These functions are part of the stdio.h header file. So, the header file stdio.h should be used in each c program.

7.12 STRUCTURE OF C PROGRAM

As we know that a program is a set of instructions written for specific task. Logical grouping of instructions in a program is called a function or block. Each C program is a collection of one or more functions and one of them should be named as main. It is because the function main is the entry point of execution for a C program. It means a computer system starts execution of c program from the main function.

Till now, we have discussed many concepts of learning any programming languages. Now, we have a clear understanding about what is the character-set, different types of tokens, different types of data, input and output statements, and header files. All these concepts are essential to begin programming using c language. Now, we need to know the basic structure of c program where we can put all these stuffs to make simple program. In general, a simple executable C-program can have the structure as shown in figure 7.9.

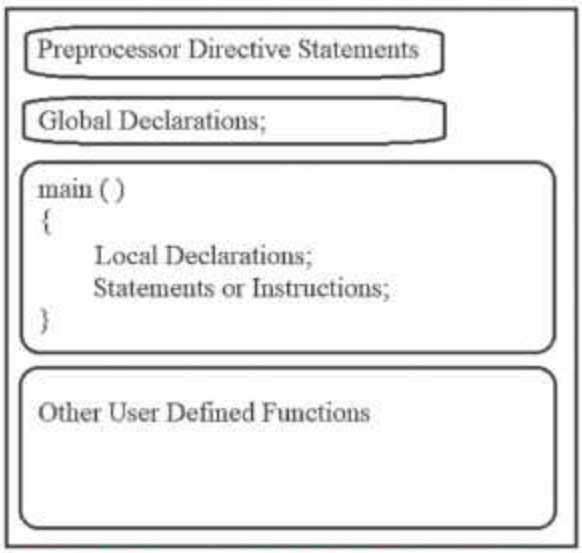


Fig. 7.9 Structure of a Simple Executable C Program

Although we have discussed many of concepts used in the above structure, yet again we are going to have a brief discussion of all element of above program-structure:

- **Pre-processor Directive Statements :** The pre-processor statement begins with # symbol. These statements instruct the compiler to perform some operations before compilation. Common use of these directive statements is to include header files or to define symbolic constants etc. Some of the common pre-processor examples are:

  #include<stdio.h>

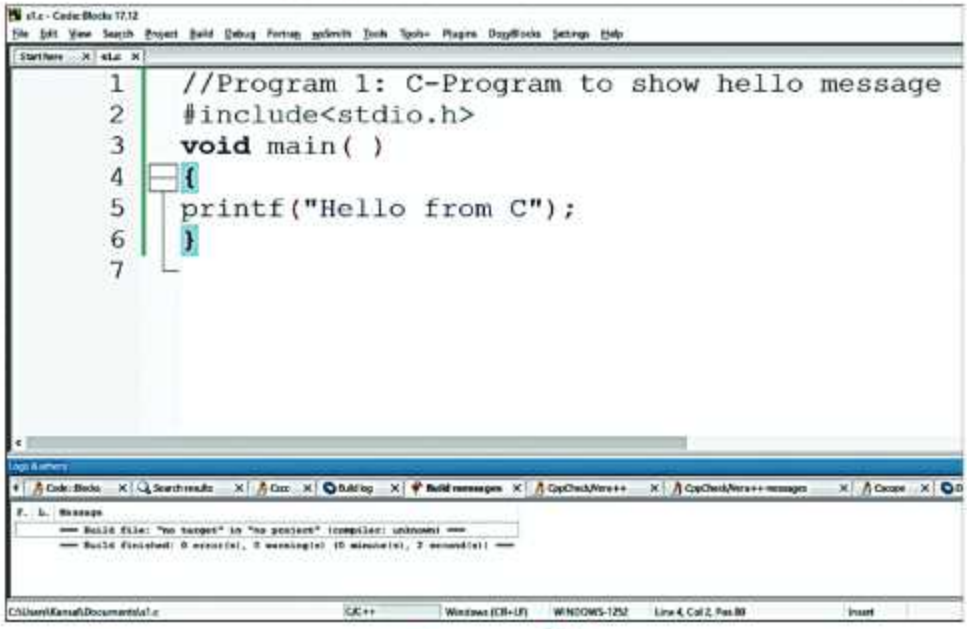
  #define PI 3.14

- **Global Declarations :** Program elements in these declarations can be used throughout the whole program. These declarations can be variables, functions or any other program element. These declarations are written outside the body of the functions.

- **The main ( ) function :** Execution of C program starts with main ( ) function. C program cannot run without the main function. Every executable C program must contain one main ( ) function.
- **Braces { } :** Braces are used to define the block of statements. The left braces after main( ) indicates the beginning of the main function and the right braces indicates the end of the main function.
- **Local Declarations :** Declarations inside any function are called local declarations. These declarations can be used only within the function in which they are declared.
- **Program statements :** These statements are the instructions of program. They are used to perform a specific task. Each executable statement should be terminated with semicolon.
- **User defined functions :** A function is a logical grouping of statements to perform some specific task. Like main( ) function, we can define more functions according to our requirements. Because these functions are written by the user, therefore these are called user defined functions.

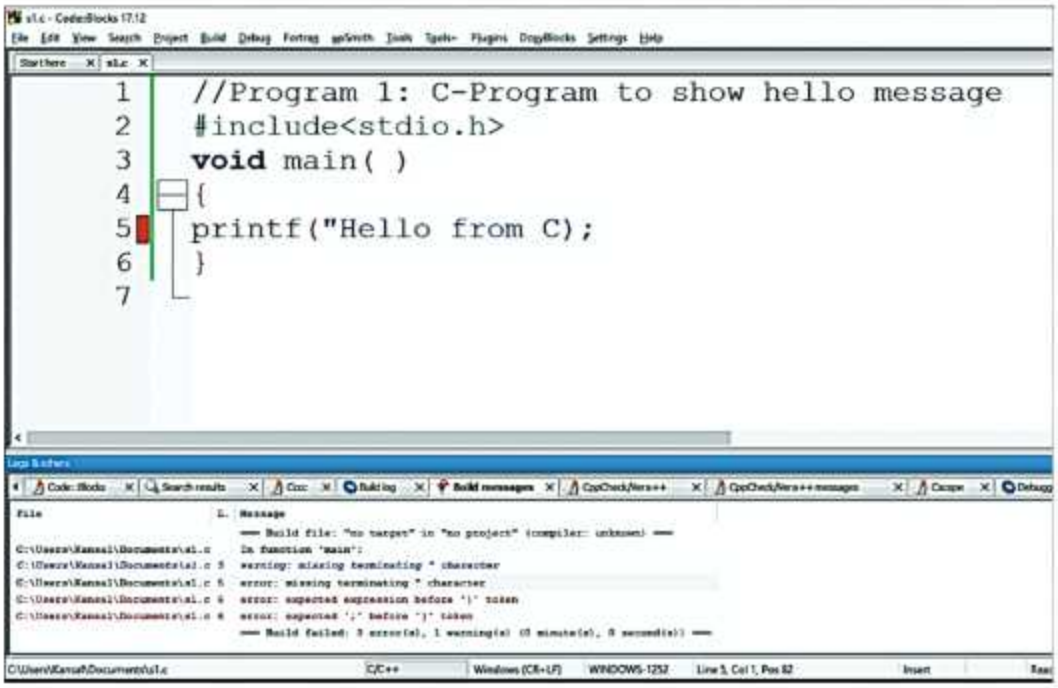
Now let us start C programming by making small C programs using whatever we have learnt in this chapter. Before proceeding, we are assuming that you have downloaded and installed the Turbo C or Code::Blocks for C program. Now, we will proceed with how to develop and execute C programs using Code::Blocks:

1. Open Code::Blocks by double clicking on its icon on the Desktop.
2. Create a New file by Click on **File → New → Empty** File or Using shortcut key **Ctrl+Shift+N**
3. Now type the following program in the file, as shown in the following figure:



Program: 7.1

4. After typing the source code of the program, save it by clicking the **File → Save File** or by pressing shortcut key **Ctrl+S**. While saving the file, don't forget to write the file extension .c with the file name, for example: test1.c, where test1 is the file name and .c is the extension name for the C programs.
5. Now, we have to compile and execute this program by clicking on **Build → Build and Run** or by pressing shortcut key **F9** from the keyboard.
6. If the program contains errors, they will be shown in the Logs and Others window which is present at the bottom of this window. If this window is not showing in the interface, we can show it by clicking on **View → Logs** or by pressing shortcut key **F2** from the keyboard. All the errors must be corrected to compile and execute the program. Consider the following program having error at line 5:



In the above figure, red box in the 5th line shows that there is some error in this line. Details of the errors can be seen in the "Build Messages" tab of "Logs and Others" window. All these errors must be removed for proceeding with compilation and execution.

7. When no errors left in the program, it will display the output of the program as shown below:

This output window will also display the time taken to execute the program. Now, to come back to the source code, press any key from the keyboard.

By following the above mentioned steps, we can develop, compile and execute C programs in the Code::Blocks. Following are some more examples of C programs. These programs show the usage of various concepts that has been explained in this chapter.

Program 7.2: C-Program to show multi-line message

```
1  #include<stdio.h>
2  void main( )
3  {
4  printf("Hello from C");
5  printf("\nC is Middle Level Language");
6  }
7
```

Program 7.2

```
Hello from C
C is Middle Level Language
Process returned 27 (0x1B)
Press any key to continue.
```

Output of Program 7.2

Program 7.3: C-Program to show Variables Declaration and showing their value

```
1  #include<stdio.h>
2  void main( )
3  {
4  int a=47;   //integer variable initialization
5  float pi=3.14;  //float variable initialization
6  printf("\n%d",a);
7  printf("\n%f",pi);
8  }
9
```

Program: 7.3

```
47
3.140000
Process returned 9 (0x9)   ex
Press any key to continue.
```

Output of Program: 7.3

In this program (Program 7.3), we have used many types of tokens. Let us discuss these tokens to illustrate whatever we have read in this chapter.

All Tokens in program	:	# include <> stdio.h void main ( ) { int a = 47 ; float pi 3.14 printf " " %d %f pi }
Keywords	:	void, int, float (representing data types)
Identifiers	:	stdio (name of header file), main, printf (name of functions), a, pi (name of variables)
Literals	:	47, 3.14 (fixed values)
Operators	:	= (assignment operator for storing values)
Special Symbols	:	# <> . ( ) { ; " " % }

Now we are going to explain the whole program (Program 7.3) line by line so that we have a better understanding of programming:

- **In line 1**of the program, we used pre-processor directive to include header file stdio.h in our program so that we can use printf( ) function in our program.

- **In line 2,** we started the main() function. Our program starts execution from this main( ) function. Any executable program must have the main function. We cannot define more than one main function in a c program.
- **In line 3,** we used opening brace { which shows the beginning of main( ) function.
- **In line 4 and 5,** we declared integer and float type variables: int a; and float pi; and assigned them literal values. These declaration statements are terminated with special symbol semicolon (;).
- **In line 6 and 7,** we used output statement printf( ) function to show the values of integer and float variables a and pi, respective format strings %d and %f are used to represent the values.
- **In line 8,** we use closing brace } which represents the end of the main function.

In the line 4 and 5 of above program, we use a special symbol // to explain the code in the program. The lines beginning with // are called Comments. Comments are used to describe our code in the program. They are ignored by compiler during compilation process. The symbol // is used for single line comment. For multiline comments, we can use /* and */ symbols in our program.

Program 7.4: C-Program to show how to use input and output statements with integer variable

```
1  #include<stdio.h>
2  void main( )
3  {
4  int a;  //integer variable declaration
5  printf("Input Value of a ");
6  scanf("%d",&a);
7  printf("Value of a is %d",a);
8  }
9
```

Program 7.4

```
Input Value of a 56
Value of a is 56
Process returned 16 (0x10)
Press any key to continue.
```

Output of Program 7.4

Program 7.5: C-Program to show how to use input and output statements with float variable

```
1  #include<stdio.h>
2  void main( )
3  {
4  float radius;   //float variable declaration
5  printf("Input Value of radius ");
6  scanf("%f",&radius);
7  printf("Value of radius is %f",radius);
8  }
9
```

Program: 7.5

```
Input Value of radius 1.25
Value of radius is 1.250000
Process returned 27 (0x18)
Press any key to continue.
```

Output of Program: 7.5

Program 7.6: C-Program to show how to use constants

```
1  #include<stdio.h>
2  void main( )
3  {
4  const float pi=3.14;     //float constant
5  printf("Value of pi is %f", pi);
6  }
7
```

Program: 7.6

```
Value of pi is 3.140000
Process returned 23 (0x17)
Press any key to continue.
```

Output of Program: 7.6

If we use Turbo C for programming, then output window will not be display directly after compiling and executing the program. To view the output window, press Alt+F5 or open **Windows** menu and then click **User Screen** to view the output window. But if we use code::blocks for c programming, then output window will automatically appears on our screen after successful execution of the program.

Points To Remember

1. C is a general purpose programming language which is developed by Dennis Ritchie.
2. C is specially designated as a Middle Level language because it has the capabilities of both low and high level languages.
3. IDE is an Integrated Development Environment that provides an environment for writing programs along with tools for compiling, executing, testing and debugging programs.
4. Character set is a set of all allowed characters for making a program.
5. Tokens are smallest individual units in a program. They are like words and punctuation marks in English.
6. Keywords are the reserve words whose meaning are predefined in the C compiler.
7. Identifiers are the names given to program elements such as variables, arrays, functions etc.
8. Operators are the symbols which are used to perform some mathematical or logical operation.
9. Constants do not allow us to change their value during execution time while the value of variables can be changed.
10. Data types define a specific type or range of values for the variables or other program elements that can hold values in the memory.
11. The functions scanf( ) and printf( ) are formatted Input/output functions which can be used to deal with any combination of numerical values, single characters and strings.
12. Comments are used to describe our code in the program.

EXERCISE

Part–A

1. Multiple Choice Questions

I. C is a _____ purpose programming language.

a. special b. general

c. objective d. None of these

II. Which of the following is invalid example of identifier?

a. roll_no b. %age_marks

c. rollno d. main

III. Which of the followings are the tokens?

a. keywords b. special symbols

c. Literals d. All of these

IV. Which of the following keywords do not represent a data type?

a. int b. float

c. const d. char

V. _____ are used to describe a code in the program?

a. Compiler b. Comments

c. Literals d. Identifiers

2. Fill in the Blanks:

I. _____ are the smallest individual units of a program.

II. The names given to program elements, such as variables, constants, arrays, functions etc. are called _____

III. Those program elements which do not allow changing their value during execution are called _____

IV. To work with single precision values, we use _____ data type.

V. File extension of header files is _____

3. Write the Full form of following:

I. FORTRAN

II. BCPL

III. IDE

IV. stdio.h

V. conio.h

VI. ASCII

Part-B

4. Short Answer Type Questions. (Write the answers in 4-5 lines)

I. Why C is called Middle Level Programming Language?

II. What is a character set?

III. What are keywords?

IV. What should be the steps for creating and executing C program?

V. Write the difference between variables and constants.

VI. What are Pre-processor directives?

Part-C

5. Long Answer Type Questions. (Write the answers in 10-15 lines)

I. What are Identifiers? Write the naming rules of identifiers.

II. What are Tokens? What are the different categories of tokens that can be used in a program?

III. What are the data types? Which primitive data types are supported by C language?

Lab Activity

- Draw a chart which represents different types of Tokens in C Language with suitable examples
- Write a C Program to Show Your School Name with complete address, each address line must be shown in a separate line