Beginner’s Guide to C

FIRST EDITION December 2014

Copyright Information

Author Soni

Dedicated to my Parents, my Family, my Lecturers and Teachers

About the Author

Soni a former teacher and is now a technical writer. She holds a degree in commerce and a post graduate degree in computer application. She has hands on experience in programming language and worked as a programmer too. She also has written books on Java that are used by school teachers. She has designed curriculum for online computer courses on programming languages such as Java, C, and C++.

Foreword

This book contains information which will help you to start working with C. Structures, Union and Pointers that are part of C programming language will be covered in the next edition of the book.

You can mail your feedback to gayathrimanoharlal@yahoo.co.in; soni0618@gmail.com

Preface

This book provides information on Programming with C. It has all the information from the basics that will help a beginner to start working with C programming language. As you all are aware that C is the basic programming language that will enhance and build your capability before learning and working with other programming languages. In addition the book gives you simple examples that will help you to work with the programming language.

Contents

Introduction to Programming Languages

A computer requires instructions to perform a task. Such instructions are included in a programming language. Collection of such program is software. The procedure of developing software is programming. There are different programming languages. A Computer can be given instructions in any programming language. All computer languages follow a set of rules called syntax. Each programming language uses its own set of syntax.

The 3 main categories in programming languages are, machine language, assembly language and high level language.

Machine Language

Sequence of instructions written in binary form is a Machine Language. Machine languages use 0 and 1 to write instructions. The program written in a machine language execute faster than program written in any other language.

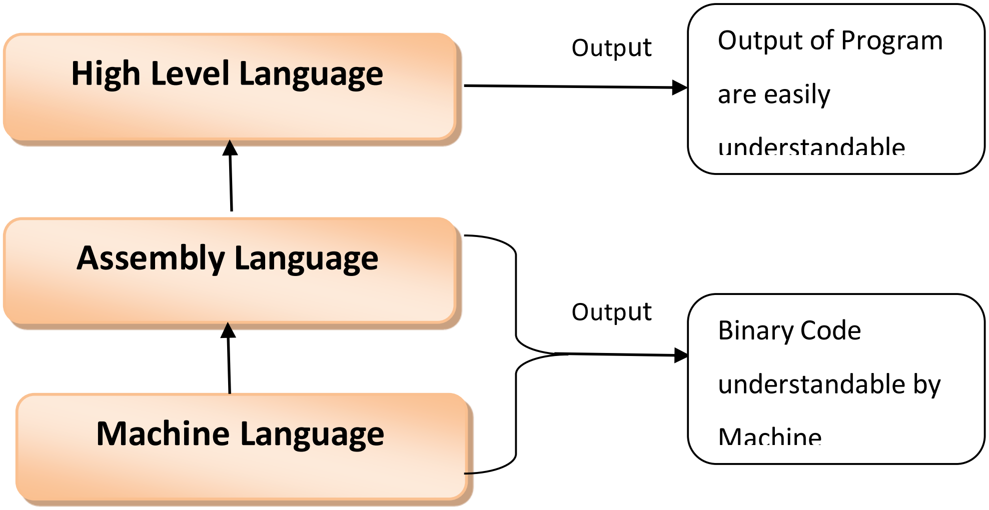
Assembly Language

It is an easy language, as special symbols like special characters, letters, or digits are used to write the program. Computers do not understand the Assembly Language hence the code written in Assembly Language has to be translated to a Machine Language. The Assembler performs the task of translation. After translation of a Assembly Language to a Machine Language the resultant output is an object code in the form of 0 and 1.

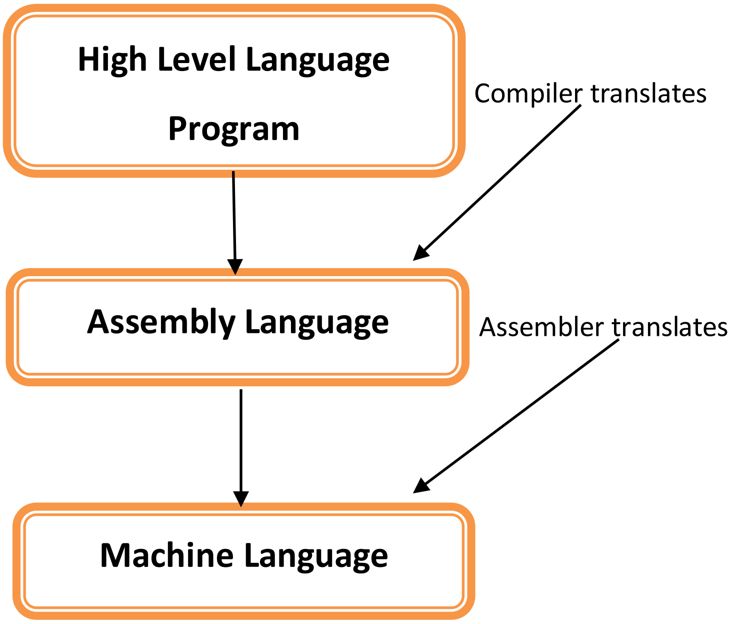
High Level Language

High level languages are easier to understand as English language is used to specify commands. Examples of High Level Language are: C, C++, Java, COBOL, and PASCAL. The High Level Language has to be translated to a machine readable language. A compiler translates the High Level Language. The resultant output is an object code.

The Hierarchy of programming language is:


High Level Language
Output
Output of Program are easily understandable
Assembly Language
Output
Binary Code understandable by Machine
Machine Language

A programming Language conversion to machine understandable language happens the way it is illustrated below:



Introduction to C Language

C is a general purpose, high level programming language and is a widely used language. This language has facilities for structured programming and allows to efficiently provide machine instructions. Hence, the language is used for coding assembly language too. The system software like Unix operating system has been coded using C. C is a procedural language. It was designed to be compiled using a relatively straightforward compiler, to provide low-level access to memory, to provide language constructs that map efficiently to machine instructions, and to have a have need of minimal run-time support. C is useful for many applications that had formerly been coded in assembly language, such as in system programming.

Software programs required for a C program

As no computer can understand C, you need several programs that help the computer understand the instructions. The programs are:

- Editor
- Compiler
- Linker
- Loader

Here Turbo C editor is used for C programming.

The table below explains the purpose of the various programs.

Program	Purpose
Editor	Used to write and alter the content of the program (source code).
Compiler	Used to convert the program to machine language.
Linker	Used to link the header file with the main program before execution.
Loader	Used to load the program to the computer memory for execution.

The basics of a C programming language needs an understanding of the basic building blocks of the programming language, the bare minimum C program contains the following parts:

Note

Compiler is used to check for errors in the program. The complier stores the translated machine language code with .obj extension and is called as object code.

- Preprocessor Commands
- Functions
- Variables
- Statements & Expressions
- Comments

Every C program consists of one or more functions, one of which must be main().

Let us look at a simple code that would print the words "My First Program in C" followed by an explanation of the various parts of the C program:

```
#include<stdio.h>

void main()

{

/* My first program in C */

printf("My Frist Program in C \n");

}
```

Structure of a C program

The various parts of the program given above are explained below:

Preprocessor Commands

1. ***#include<stdio.h>***

   Start writing a C program with a preprocessor command ***#include<stdio.h>.***

   This informs a C compiler to include a header file stdio.h before compiling the program.

   - # is a preprocessor command.
   - Include instructs the C compiler to include the header file library function. The header file contains the instruction for standard input and output.
   - <stdio.h> - All the header file library functions are written within open and close angular brackets, <>.
   - The expansion of stdio.h is “standard input and output header file.
   - .h is the extension of the library function file identified by the compiler with .h extension.
   - All the library functions included in a C program are written as #include<header file name.h>

Refer Library Functions to know about the list of other library functions that can be included in a C program.

2. **main() function**

The next line int main() is the main function where program execution begins.

Note

In C, semicolon (;) indicates the end of statement.
#include statement does not end with a semicolon.

All programs in C should have a main() function. There are two variations of main(). They are void main() and int main(). void main() denotes a function does not return a value, int main() denotes a function that is of a numeric return type.

3. **Starting Brace**

All C program start with a left brace {.and the program code always ends with a closing brace - }.

4. **Statements**

Comment Statement

The next line /*...*/ will be ignored by the compiler and it has been put to add additional comments in the program. So such lines are called comments in the program. You can give a brief description about the program within the /*…..*/ comment line statements.

Input and Output statements

Output statement

printf(...)

The next line *printf(...)* is another function available in C which causes the message "My First Program in C” to be displayed on the screen. All the statements that you want to display on the screen should start with the printf( ) function.

Input Statement

scanf(“%d”, &a);

The data type of the input is specified within double quotes, if it is of integer type then it is % d and &a represents a variable data item.

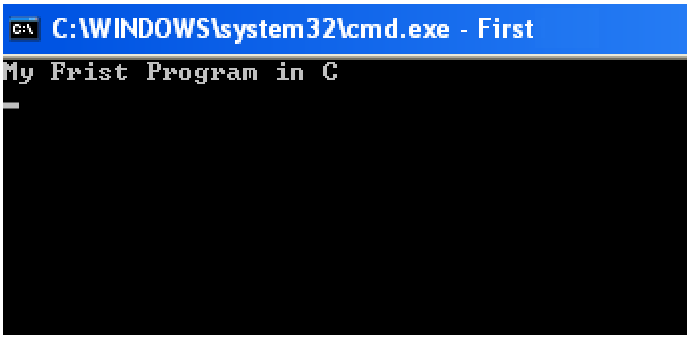
scanf(…..) function can be used to enter any combination of numerical values, single characters and strings.

Compile & Execute C Program

Let’s look at how to save the source code in a file, and how to compile and run it. Following are the simple steps:

1. Open a text editor and add the above-mentioned code.
2. Save the file as First*.c in the TC\Bin directory. The directory where you have the C compiler.*
3. Open a command prompt and go to the directory where you saved the file.
4. Type TC First*.c* and press enter.
5. This opens up the First.c program in the Turbo C window.
6. To compile your code press Alt + F9 key simultaneously.
7. If there are no errors there will be no error message.
8. You have to execute the file to view the output. To execute the file press Ctrl + F9 keys simultaneously.
9. You will be able to see *"My First Program in C"* printed on the screen.

Output screen of First.c



```
My Frist Program in C
_
```

C uses structured programming with variables and recursive statements. In C, all executable code is contained within "functions" All C program use the semicolon as a statement terminator and curly braces({ }) for grouping blocks of statements.

The following are the key points of C language:

- Uses preset keywords
- Includes flow of control structures for, if/else, while, switch, and do/while.
- A large number of arithmetical and logical operators are used.
- Statements are used to specify actions. Common statements are input, output and an expression statement. The expression statement consists of an expression to be evaluated, followed by a semicolon;.

- Functions and procedures may be called and
- Variables may be assigned new values.

The basic characters that are included in a program are:

- Lowercase and uppercase letters: a–z A–Z
- Decimal digits: 0–9
- Graphic characters: ! " # % & ' ( ) * + , - . / : ; < = > ? [\ ] ^ _ { | } ~
- Newline indicates the end of a text line; it need not correspond to an actual single character, although for convenience C treats it as one.

Data types

C supports different types of data, the basic C data types are as listed below:

S.No.	Data Type	Description	Storage Size	Value Range
1.	int	The numeric integer data	2 or 4 bytes	-32,768 to 32,767 or -2,147,483,648 to 2,147,483,647
2.	Char	Single character	1 byte	-128 to 127 or 0 to 255
3.	Float	Floating point number	4 byte	1.2E-38 to 3.4E+38 (Precision up to 6 decimal places)
4.	Double	Double-precision floating point number	8 byte	2.3E-308 to 1.7E+308 (Precision up to 15 decimal places)

The array types and structure types are referred to collectively as the aggregate types. The type of a function specifies the type of the function's return value. We will see basic types in the following section.

The void Type

The void type specifies that no value is available. It is used in the following kinds of situations:

1. When a function does not return a value.
2. For empty set of values.

Variables

Variables store values and they are used to perform calculations and evaluate conditions. A variable refers to values that are not fixed, they keep changing. C programming language also allows defining various other types of variables, other than those defined in the following type:

Variable Type	Description	Example
Integer	A variable, which can have only integers, is called integer variable. Integer variables are used to store whole numbers. A fractional value cannot be stored in an integer variable.	int v=10; int is the data type v is the variable 10 is the value
Character	A variable, which can store only characters, is called character variable. Character constants stored in these variables should be enclosed within single quotes.	char str;; char str= 'A'; where, char is the data type str is the variable , 'A' is value. The char data type values should be provided within single quotes.
Float	Variables, which are used to store decimal values, are called float variables. They can store both integer part (to the left of decimal point) and fractional part (to the right of decimal point).	float marks = 66.78; where, float is the data type marks is the variable 66.78 is the value assigned to the variable.

Variable Definition in C

A variable definition means to tell the compiler where and how much to create the storage for the variable. A variable definition specifies a data type and contains a list of one or more variables of a specific data type.

Variable Initialization

Assigning value to the variable is called Variable Initialization. A variable can be initialized at the place where the variable is declared.

Syntax

data_type variable_name = value;

int x = 15;

float p = 17.1;

char z=’ p’;

In the following example variables have been declared, but they have been defined and initialized inside the main function:

Example

CODE

```
#include<stdio.h>
int main ()
{
/* variable definition: */
int a, b,c;
float f=3.2, g;
/* actual initialization */
  a =15;
  b =20;
  c = a + b;
  g= f * (b/a);
printf("Value of c : %d \n", c);
printf("Value of g:\ %f", g);
}
```