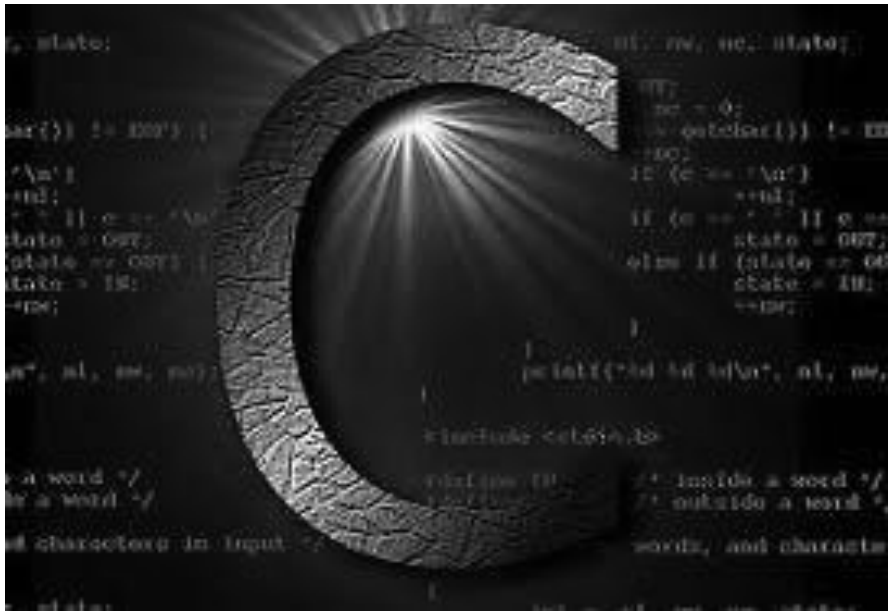
Computer Programming Using C



DAVID LIVINGSTON J, M.E.

Copyright © DAVID LIVINGSTON J 2018
All Rights Reserved.

Written and Published By:

DAVID LIVINGSTON J, M.E.
CFY Literature Service,
Coimbatore—28

Mobile:
99942 86194

Our Websites:
http://ffy.vpweb.in
http://jdlcse.blogspot.com

TABLE OF CONTENT!

SECTION I

Basics of Computer Programming

This section explains some of the basic concepts involved in Programming using Computers, which include the following:

Introduction to Programming Languages

Structured Vs. Object Oriented Programming

Software Development Life Cycle (SDLC)

Planning a Program using Flowchart and Pseudo Code

Introduction to C Language

Fundamental Elements of C

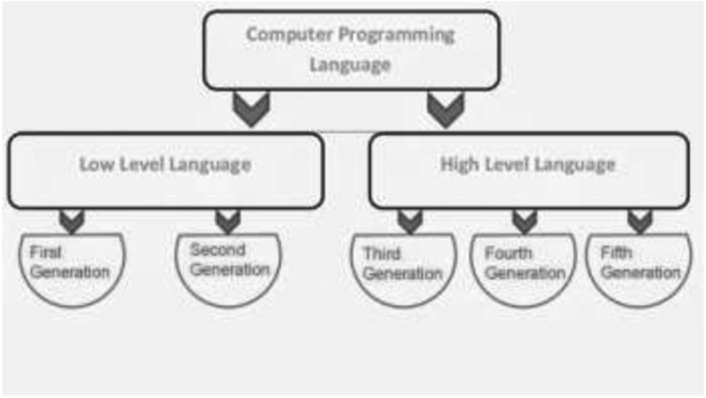
Primitive Data Types in C

Using Enumerated Data Type in C

Chapter 1

INTRODUCTION TO PROGRAMMING LANGUAGES

Programming languages are coding schemes that are used for providing instructions to a computer system in order to perform a processing activity by it. They are used by IS professionals to develop both Operating Systems and Application Software. They were developed to help solve particular problems.



Generations of Programming Languages

Programming languages provide a programmer with a set of keywords, symbols and a system of rules for constructing statements that can be executed by a computer. The symbols and the keywords are having special meaning in the language. The set of rules (called syntax) dictate how the symbols should be combined into statements capable of conveying meaningful instructions to the CPU.

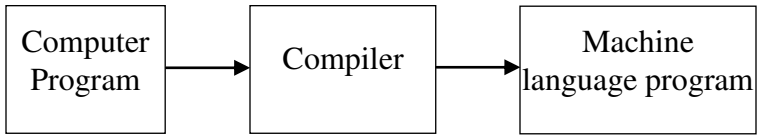
Generation of Programming Languages

The **First Generation** programming language is machine language, which requires the use of binary symbols (0s and 1s). As machine language is the language of the CPU, programs written using machine language can be understood and executed directly by the CPU.

Second Generation languages overcome some of the difficulties inherent in machine language by replacing the binary digits with symbols. Thus, the programs written using second generation language are in more readable form than that of the machine language. Second generation languages are also called as Assembly language, since assembly language programs are converted into binary language coding before their execution with the help of a language translator named Assembler.

Third Generation languages are high level languages that use English-like statements and commands. Some of the third generation languages are: BASIC, COBOL, C, and FORTRAN. 3GL languages are easier to learn and use than machine and assembly languages because they more closely resembles everyday human communication and understanding.

With third-generation programming languages, each statement in the language translates into several instructions in the machine language. A special software program called compiler converts the programmer's source code into the machine language instructions consisting of binary digits, as shown in below figure:



The role of a Compiler

Fourth Generation languages are languages that support Client/Server technology. They have the following characteristics which distinguish them from the third generation languages:

1. Provide features for storing and accessing information from a database.
2. Emphasize what output results are desired rather than how programming statements are to be written.
3. Provide the tools for designing the User Interface screens easily.

4GL languages are of two types: Front-ends and Back-ends. Front-ends are tools that include visual development tools like PowerBuilder, Delphi, Essbase, Focus, Powerhouse and SAS. Back-ends are Database Management Systems (DBMS) such as MS Access, SQL Server, or Oracle. From the front-end, we can access the back-end using a database language called SQL (Structured

Query Language) for performing database queries and manipulations.

Fifth Generation programming languages are Visual Programming languages such as Visual Basic, Visual C++ and PC COBOL. They provide an environment called Integrated Development Environment (IDE), which includes all the necessary tools for program development. Some of the tools available in an IDE are: Editor, Screen Designer, Code Generator, Compiler and a Debugger.

Moreover, fifth generation languages use a visual or graphical development interface to create source language code that is usually compiled with a 3GL or 4GL language compiler. Microsoft Visual Studio 7, now known as Visual Studio.Net allows 20-some programming languages such as COBOL, C++, Perl, SmallTalk, C#, Jscript, Visual Basic, Visual Foxpro and even Java to share a single GUI.

Object-Oriented Programming (OOP) Languages

The third generation languages separate data elements from the procedures or actions that will be performed on them. They give more importance to actions (called procedures) than the data handled by them. In this approach, language programs are divided into smaller programs known as functions. Most of the functions share the data globally.

But, Object Oriented Programming languages tie both data and functions (actions) into a single unit called an **Object**. An object consists of data and the functions that can be performed on the data. Programming languages that are based on the object oriented concepts such as objects,

encapsulation, data hiding, polymorphism and inheritance are called Object Oriented languages.

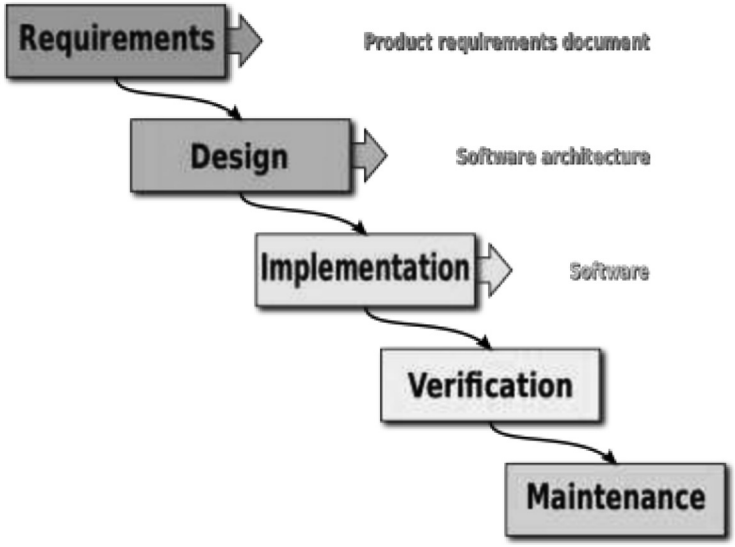
In this type of programming, any real world entity can be modeled as object. The whole software is considered as a group of objects that work together to accomplish a particular task. During execution, objects interact with each other by sending messages and receiving responses. For instance, in a program that performs withdrawal from an account, a customer object may send a withdraw message to a bank account object in order to perform withdrawal. Any object that communicates with another object need not be aware of its internal workings but only its function signatures.

Thus, OOP is defined as a method of programming in which programs are organized as co-operative collections of objects, each of which represents a real world entity. The entire programming is centered on concepts such as objects, classes, polymorphism and inheritance.

Chapter 2

STRUCTURED PROGRAMMING

Software development is a process of creating new software or modifying existing software for meeting the current requirements of its users. This process consists of various stages or phases in it.



Software Development Life Cycle (SDLC)

Steps involved in Software Development process include the following:

1. Problem Definition (Analysis)
2. Program Design
3. Coding / Implementation
4. Testing and
5. Maintenance

A complete set of all these activities involved in developing software is known as Software Development Life Cycle (SDLC). This is because the same sequence of steps are to be followed whenever we develop new software from scratch or modifying existing software for up gradation.

Some small programs like text editor (e.g., Notepad) can be coded directly without following all the steps involved in SDLC. But, large programs like MS Word or MS Excel involve complexity in areas like understanding the problem domain, meeting the customer needs and delivering a good quality product in time.

To overcome the complexities involved in software development, many methodologies, tools and languages were introduced. Following are two major methodologies introduced for simplifying software development process:

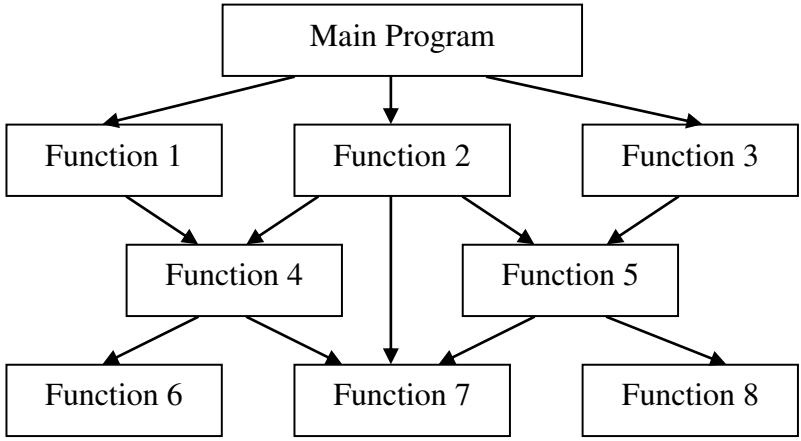
- Structured (Procedural) Programming
- Object Oriented Programming (OOP)

Structured Programming:

In structured programming model, software designers tend to use Top-Down approach, in which the overall objective of the system is defined first. Then the system is divided into various sub tasks or sub modules. With this methodology, software development is done by writing a set of sub programs, called **functions** that can be integrated together to form a complex system.

In Structured programming, the primary focus is on functions. A function is a sub program that performs a specific task using the values given to it through input variables (called parameters) and then returns the result to its calling program. Each function consists of a set of program statements and some local variables.

A function when invoked behaves as though its code is inserted at the point of its call. The communication between the caller (calling function) and the callee (called function) takes place through parameters. A typical program structure for structured approach is shown below:



At the time of function call, the control is transferred from the caller to the first statement of the callee. All the statements in the function body are executed and then the control is transferred back to the caller to resume the execution of other statements.

Some characteristics exhibited by Structured or Procedural-oriented approach are:

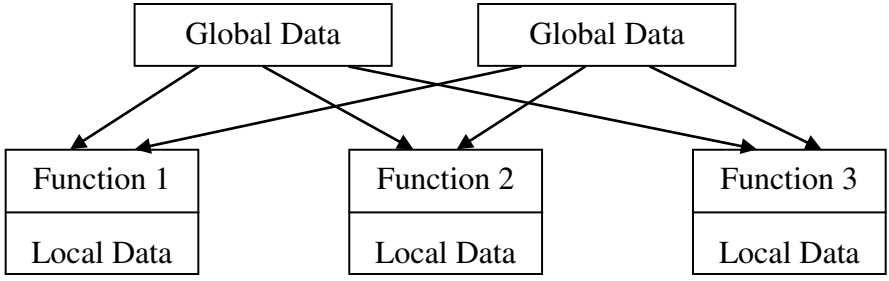
- ➔ Emphasis is on doing things (algorithms)
- ➔ Large programs are divided into smaller programs called functions.
- ➔ Most of the functions share global data.
- ➔ Data move openly around the system from function to function and
- ➔ Employs Top-down approach in program design

Limitations of Structured Programming:

Structured programming was a powerful tool that enabled programmers to write moderately complex programs fairly easily. However, as the programs grew larger, this approach failed to show the desired results in terms of bug-free, easy-to-maintain and reusability of programs.

In this approach, very little attention is given to data used by the function. And, in a multi-function program, many important data items are placed in the global scope, so that they may be accessed by all functions. But, this leads to the problem of accidental modification of data due to its access from various functions of the program. Hence, in a large program it is difficult to keep track of the data items having global scope.

The following picture depicts the relationship of data and function in structured (or) procedural programming:



Relationship of data and functions in Structured programming

Another series drawback with the procedural approach is that it does not model the real world entities to the elements in a program in a one-to-one manner. This is because the functions are action-oriented and they do not really correspond to the elements of the problem.

Object Oriented Programming:

Object Oriented Programming is centered on new concepts such as objects, classes, polymorphism, and inheritance. OOP is defined as follows: *It is a method of programming in which programs are organized as co-operative collections of objects, each of which represents an instance of some class and whose classes are all members of a hierarchy of classes united through the property called inheritance*.

In this approach, any real world entity can be modeled as an object. The whole software is considered as

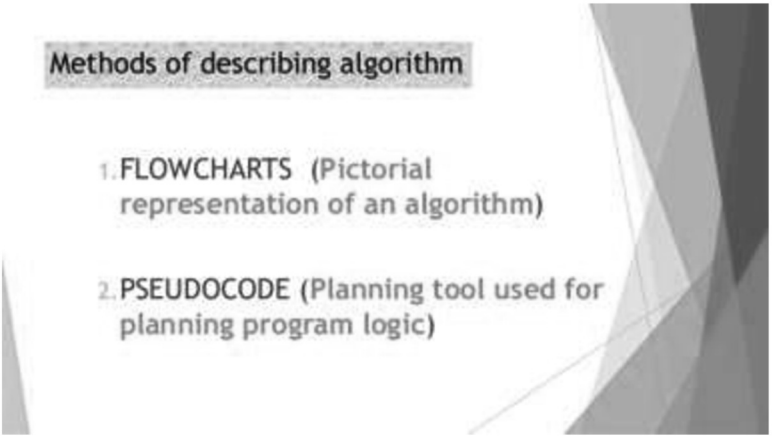
a group of objects that work together to accomplish a particular task. During execution, objects interact with each other by sending messages and receiving responses.

For instance, in a program that performs withdrawal from an account, a customer object may send a *withdraw* message to a bank account object in order to perform a withdrawal operation. Any object that communicates with another object need not be aware of its internal workings but only its function signatures.

Chapter 3

PLANNING A COMPUTER PROGRAM (Part I)

Planning a computer program is nothing but planning the logic of the program. In order to produce a correct and effective computer program, the logic of the program has to be planned first. Without having the logic, a programmer can't write the program well.



Methods of Planning a Computer Program

In a computer program, all the instructions must be written in a proper sequence. When the order is not correct or some of the instructions are left out, the computer will calculate a wrong answer. To ensure the correct order and the appropriateness of the computer instructions, a program must be planned first. Planning a computer program is done with the help of planning tools and techniques, which include Algorithm, Flowchart and Pseudo code.

A sequence of instructions is called an **algorithm**. Algorithms are a fundamental part of computing. There are two commonly used tools to help to document program logic (the algorithm). They are **flowcharts** and **Pseudo code**. Generally, flowcharts work well for small problems but Pseudo code is used for larger problems.

Using Algorithm

The term algorithm refers to the logic of the program. An algorithm is defined as a step-by-step description of how to arrive at the solution of a given problem. An algorithm contains a set of instructions that must be executed in a specified sequence to produce a desired result. The characteristics of a good algorithm are listed below:

1. Each and every instruction should be precise and unambiguous.
2. Each instruction should be designed in such a way that it can be performed in a finite time.
3. Not a single instruction should be repeated infinitely, i.e., there should be an end for an algorithm both logically and physically.

4. After the termination of an execution, the user must be able to get the desired output.

The following are three ways in which an algorithm can be represented:

- ❖ As Programs
- ❖ As Flowcharts
- ❖ As Pseudo codes

The first one is the language representation of an algorithm that can be compiled and executed by a computer to produce an expected output. When a high-level language is used for representing an algorithm, it becomes a computer program. The syntax and semantics of that particular programming language must be followed to write the program in it.

Normally an algorithm is written in a simple and plain English. No rules and regulations are formed for writing an algorithm except some characteristics, which qualify a set of instructions to be an algorithm. To represent an algorithm pictorially a flowchart is used.

Using Flowchart

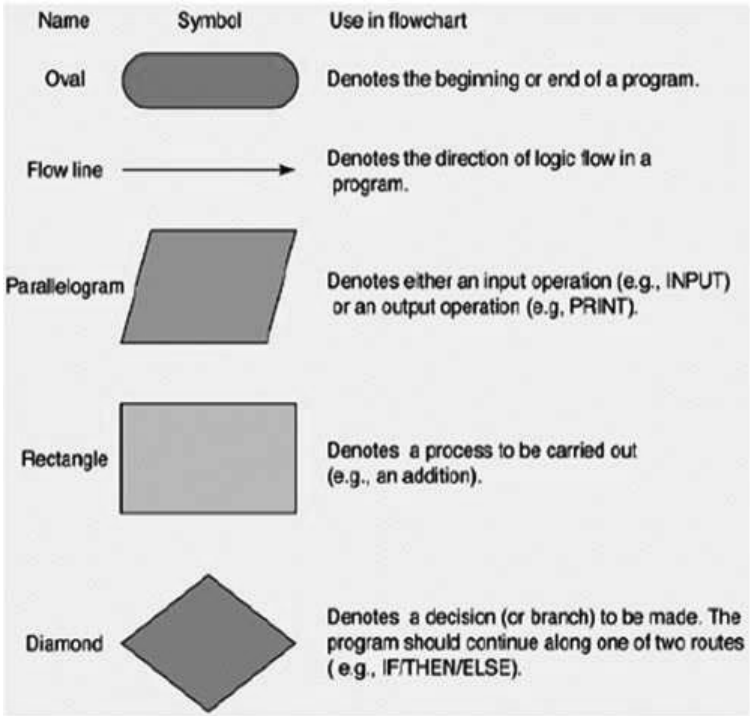
A flowchart is a pictorial representation of an algorithm. Programmers often use it as a visual tool for organizing the sequence of steps necessary to solve a problem. The process of drawing a flowchart for an algorithm is often referred to as **flowcharting**.

A set of symbols is provided for drawing a flowchart and to represent different operations to be executed by a computer. The symbols used in a flowchart are connected together using arrow headed solid lines to

indicate the sequence in which the instructions must be evaluated. Flowcharting is a task that must be done after writing the algorithm for a computer program. It is a pictorial representation of a program.

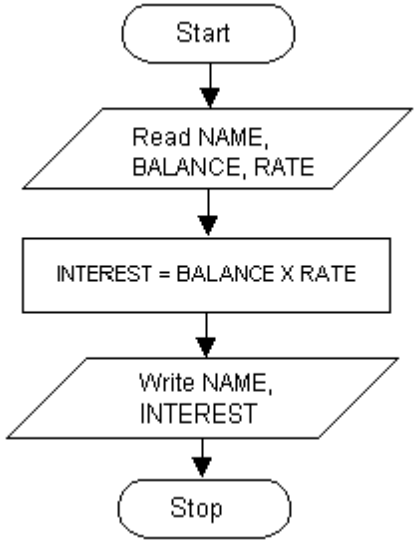
With flowcharting, essential steps of an algorithm are shown using the shapes above. The flow of data between steps is indicated by arrows, or flow lines.

Some of the common symbols used in flowcharts are shown below. It is followed by a flowchart (and equivalent Pseudo code) drawn for calculating the interest for a given loan amount. The second flowchart and the equivalent Pseudo code given above is for computing sum, average and product of three numbers.



Symbols used in a Flowchart

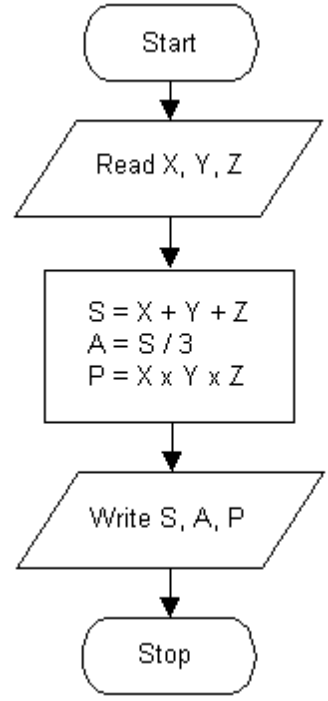
Flowchart & Its Corresponding Pseudo Code



Read NAME, BALANCE, RATE

Compute INTEREST as BALANCE x RATE

Write (Display) NAME and INTEREST



Read Read X, Y, Z

Compute Sum (S) as X + Y + Z

Compute Average (A) as S / 3

Compute Product (P) as X x Y x Z

Write (Display) the Sum, Average and Product