Machine Language

Machine Language is the language written as strings of binary 1’s and 0’s. it is the only language which a computer understands without using a translation program.
A machine language instruction has two parts. The first part is the operation code which tells the computer what function to perform and the second part is the operand which tells the computer where to find or store the data which is to be manipulated. A programmer needs to write numeric codes for the instruction and storage location of data.

Disadvantages

- It is machine dependant i.e. it differs from computer to computer
- It is difficult to program and write
- It is prone to errors
- It is difficult to modify

Assembly Language

It is a low level programming language that allows a user to write a program using alphanumeric mnemonic codes, instead of numeric codes for a set of instruction.
It requires a translator known as assembler to convert assembly language into machine language so that it can be understood by the computer. It is easier to remember and write than machine language.

Assembler

It is a computer program which converts or translates assembly language into machine language. It assembles the machine language program in the main memory of the computer and makes it ready for execution.

Advantages

- It is easy to understand and use
- It is easy to locate and correct errors
- It is easier to modify

Disadvantages

- It is machine dependant

High level language

It is a machine independent language. It enables a user to write programs in a language which resembles English words and familiar mathematical symbols. COBOL was the first high level language developed for business.
Each statement in a high level language is a micro instruction which is translated into several machine language instruction.

Compiler: A compiler is a translator program which translates a high level programming language into equivalent machine language programs. It compiles a set of machine language instruction for every high level language program.

Source code: It is the input or the programming instructor of a procedural language.
The compiler translates the source code into machine level language which in known as object code. Object code can be saved and executed as and when desired by the user.
Source code -> Language Translator Program -> Object code
High level language -> Machine language

Linker: A program used with a compiler to provide links to the libraries needed for an executable program. It takes one or more object code generated by a compiler and combines them into a single executable program.

Interpreter: It is a translator used for translating high level language into the desired output. It takes one statement, translates it into machine language instruction and then immediately executes the result. Its output is the result of program execution.

Advantages of High level Language

- It is machine independent
- It is easier to learn and use
- It is easier to maintain and gives few errors

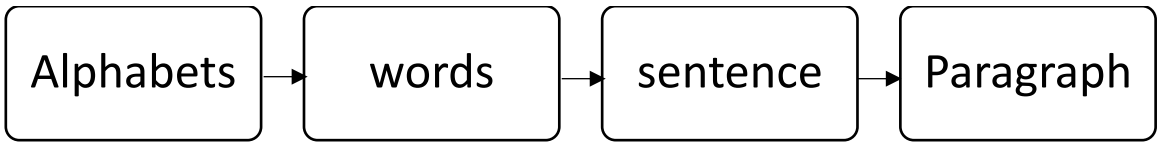
Disadvantages

- It lowers efficiency
- It is less flexible

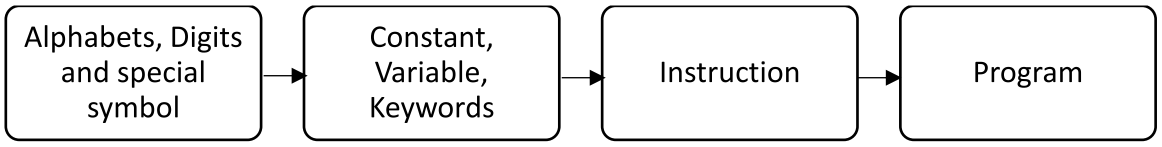
Introduction to C

C is a programming language developed at AT & T’s Bell Laboratories of USA in 1972. It was designed and written by a man named Dennis Ritchie. In the late seventies C began to replace the more familiar languages of that time like PL/I, ALGOL, etc.
It was initially designed for programming UNIX operating system. Now the software tool as well as the C compiler is written in C. Major parts of popular operating systems like Windows, UNIX, Linux is still written in C. This is because even today when it comes to performance (speed of execution) nothing beats C. Moreover, if one is to extend the operating system to work with new devices one needs to write device driver programs. These programs are exclusively written in C. C seems so popular is because it is reliable, simple and easy to use. often heard today is – “C has been already superceded by languages like C++, C# and Java.
There is a close analogy between learning English language and learning C language. The classical method of learning English is to first learn the alphabets used in the language, then learn to combine these alphabets to form words, which in turn are combined to form sentences and sentences are combined to form paragraphs. Learning C is similar and easier. Instead of straight-away learning how to write programs, we must first know what alphabets, numbers and special symbols are used in C, then how using them constants, variables and keywords are constructed, and finally how are these combined to form an instruction. A group of instructions would be combined later on to form a program.

Steps in learning English Language



Steps in learning C Language


Alphabets, Digits and special symbol
Constant, Variable, Keywords
Instruction
Program

Character set

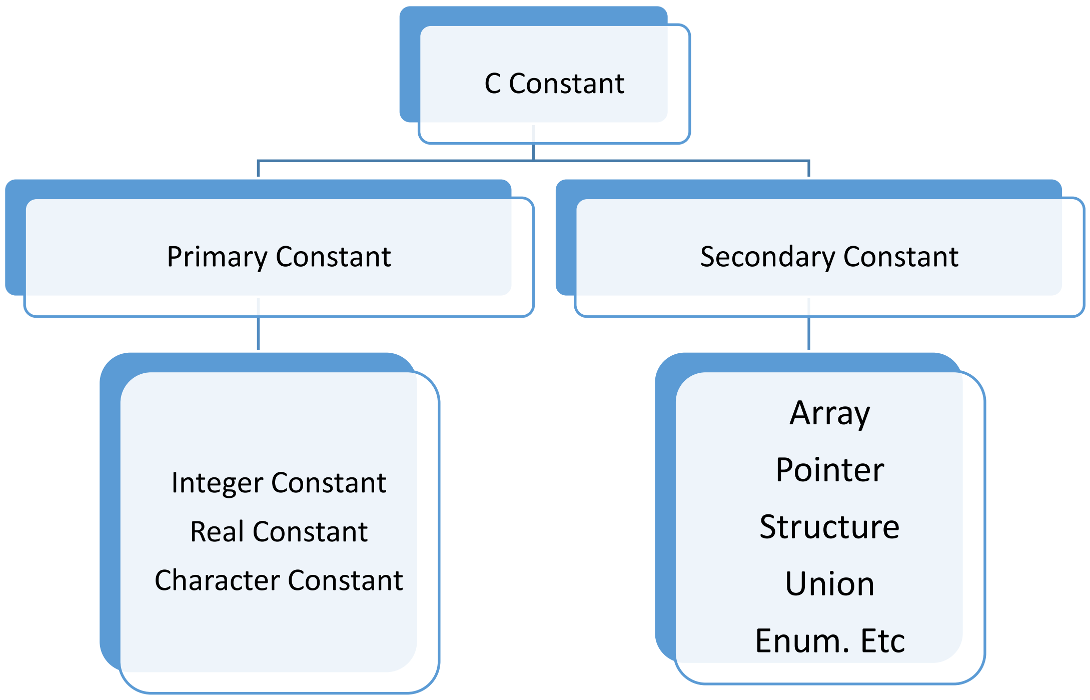
A character denotes any alphabet, digit or special symbol used to represent information. In C there are 256 character are used to form an instruction. Valid alphabets, numbers and special symbols allowed in C are

Alphabets	A, B, ……………….,Y,Z a,b, ………………., y,z
Digits	0,1,2,3,4,5,6,7,8,9
Special Symbol	~,`,!,@,#,$,%,^,&,*,(),_,- ,+,=,\,|,},{,[],’,”,:,;,<>,?,/

The alphabets, numbers and special symbols when properly combined form constants, variables and keywords.

Constants

Constant is a any value that cannot be changed during program execution. In C, any number, single character, or character string is known as a constant. A constant is an entity that doesn't change whereas a variable is an entity that may change. For example, the number 50 represents a constant integer value. The character string "Programming in C is fun.\n" is an example of a constant character string. C constants can be divided into two major categories: Primary Constants Secondary Constants These constants are further categorized as



Primary Constant

Integer Constant: Integer constant is a numeric value. Numeric value consists of digits. Integer constant are whole number which have no decimal point. It required minimum size of 2 bytes and max 4 bytes.

Rules for Constructing Integer Constant

- It must have at least one digit
- It must not have a decimal point
- It may be positive or negative
- If no sign precedes an integer constant then it may be assumed to positive
- No comma or space are allowed within integer constant
- Allowable range of integer constant is maximum +32767 and minimum -32768

Real Constant: Real constant is also a numeric value. It is also called a floating point constant but it must have a decimal point. It required 4 bytes.

Rules for Constructing Integer Constant

- It must have at least one digit
- It must have a decimal point
- It may be positive or negative
- If no sign precedes a real constant then it may be assumed to positive
- No comma or space are allowed within real constant
- Allowable range of integer constant is maximum 3.4E+38 and minimum 3.4E-38
- Ex. +325.52, -526.53

Character constant

Character constant represented as a single character enclosed within a single quote. These can be single digit, single special symbol or white spaces such as ‘9’,’c’,’$’, ‘ ’ etc. Every character constant has a unique integer like value in machine’s character code as if machine using ASCII (American standard code for information interchange). Some numeric value associated with each upper and lower case alphabets and decimal integers are as: A------------ Z ASCII value (65-90) a-------------z ASCII value (97-122) 0-------------9 ASCII value (48-59) ; ASCII value (59)

String constant

A Set of characters are called string and when sequence of characters are enclosed within a double quote (it may be combination of all kind of symbols) is a string constant. String constant has zero, one or more than one character and at the end of the string null character(\0) is automatically placed by compiler. Some examples are “,sarathina” , “908”, “3”,” ”, “A” etc. In C although same characters are enclosed within single and double quotes it represents different meaning such as “A” and ‘A’ are different because first one is string attached with null character at the end but second one is character constant with its corresponding ASCII value is 65.

Variables

Variable is a data name which is used to store some data value or symbolic names for storing program computations and results. The value of the variable can be change during the execution. The rule for naming the variables is same as the naming identifier. Before used in the program it must be declared. Declaration of variables specify its name, data types and range of the value that variables can store depends upon its data types.

Syntax:

```
int a;
char c;
float f;
```

Variable initialization

When we assign any initial value to variable during the declaration, is called initialization of variables. When variable is declared but contain undefined value then it is called garbage value. The variable is initialized with the assignment operator such as
Data type variable name=constant;
Example: int a=20;
Or int a;
a=20;

Identifiers

Identifiers are user defined word used to name of entities like variables, arrays, functions, structures etc.

Rules for naming identifiers are:

- name should only consists of alphabets (both upper and lower case), digits and underscore (_) sign.
- first characters should be alphabet or underscore
- name should not be a keyword
- since C is a case sensitive, the upper case and lower case considered differently, for example code, Code, CODE etc. are different identifiers.
- identifiers are generally given in some meaningful name such as value, net_salary, age, data etc. An identifier name may be long, some implementation recognizes only first eight characters, most recognize 31 characters. ANSI standard compiler recognize 31 characters. Some invalid identifiers are 5cb, int, res#, avg no etc.
-

Keywords:

There are certain words reserved for doing specific task, these words are known as reserved word or keywords. These words are predefined and always written in lower case or small letter. These keywords cann't be used as a variable name as it assigned with fixed meaning. Some examples are int, short, signed, unsigned, default, volatile, float, long, double, break, continue, typedef, static, do, for, union, return, while, do, extern, register, enum, case, goto, struct, char, auto, const etc. In C there are 32 keywords are available.