Chapter-1 Introduction to C Language

C Language is a high level and general-purpose Programing Language. A **Programming language** is a special Language use to develop a program. A **Programm** is a set of instruction for computer to perform specific task.

Types Of Programming Languages.

Machine Level Language

Machine Language is the language written as String of binary 1's and 0's. It is the only lanuage that computer can understand directly. A programmmer needs to write numeric codes for the instruction and storage location of data.

Advatages:

- Fast execution

Disadvantages

- It is machine dependent
- It is difficult to program and write

Assembly Language

In Assembly Language alphanumeric numeric codes are used, instead of numeric codes. Assembly language can not understand by the computer. The set of symbols and letters forms the Assembly Language and a translator program is required to translate the Assembly Language to machine language. This translator program is called `Assembler'. It is considered to be a second-generation language.

Advatages:

- Easier to understand and saves a lot of time and effort of the programmer.
- It is easier to correct errors and modify program instructions.

Disadvantages:

- Execution is slow compare to Machine Level language.
- Diificult to remember Syntax, takes lots time to code program.

High-Level Programming Language

A high-level language is a designed to simplify computer programming. High-level source code contains easy-to-read syntax, it is a English-Like Language. Most common programming languages are : C , C++ , JAVA.

High level language cannot understand by computer. A translator program is required to translate the High-level Language to machine language. This translator program is called 'Compiler'.

It is considered as a third generation language.

Advantages:

- Easy to learn and Use
- Easier to maintain and Fewer Errors

Disadvantages:

- Execution is slow coompare to Machine Level language.
- Less Efficient.

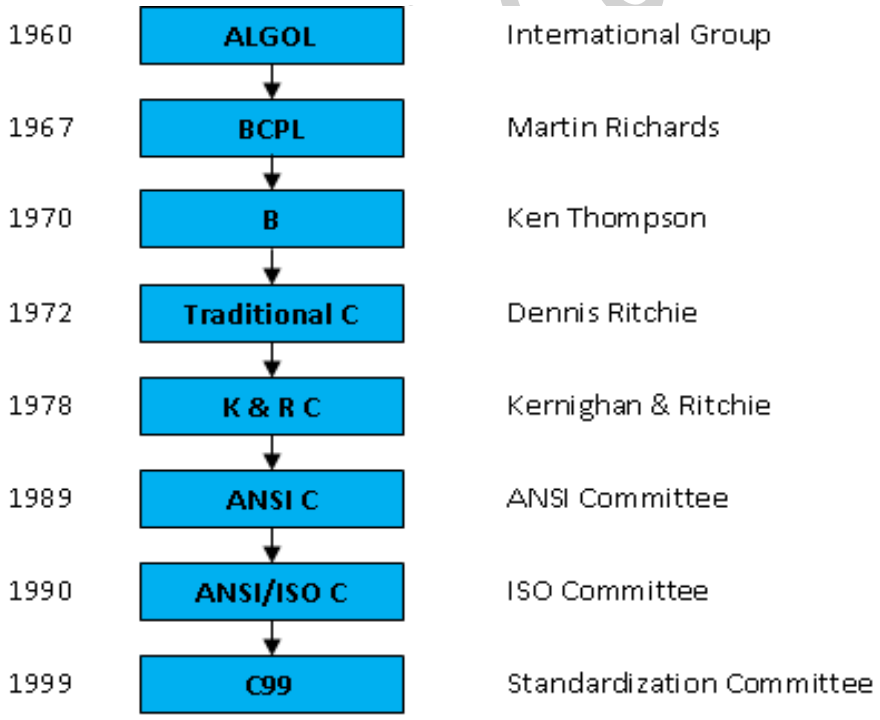
Procedural Vs. Non Procedural Programming Language.

Procedural Language	**Non-Procedural Language**
Procedural languages are also called third generation languages (3GL).	It is also called fourth generation languages (4GL).
This language tells the computer what to do and how to do.	Non-procedural languages tells the computer what to do, not how to do.
It is very difficult to learn as compared to non-procedural.	It is very easy to learn.
It is difficult to debug.	It is very easy to debug.
It requires a large number of procedural instructions.	It requires a few non-procedural instructions.

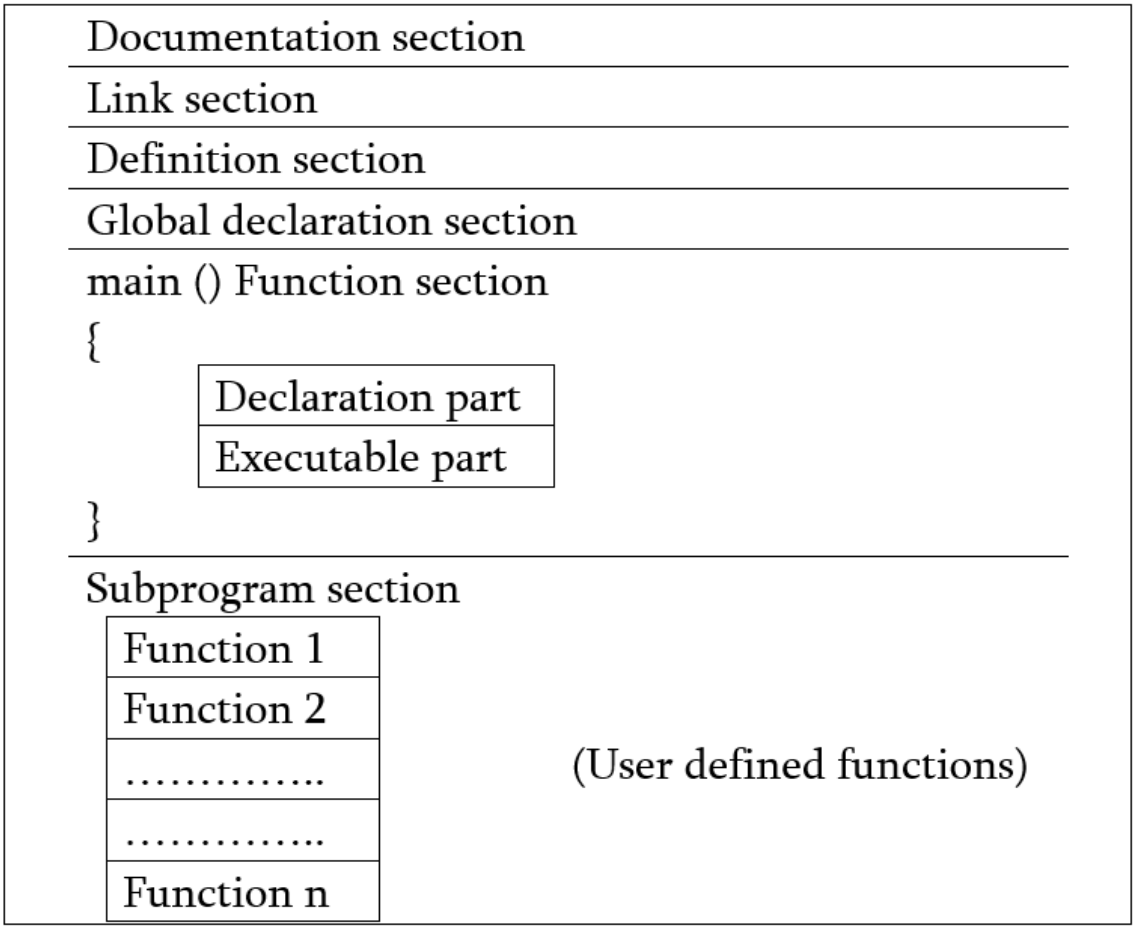
Difference between Compiler and Interpreter

Compiler	Interpreter
Compiler Takes Entire program as input.	Interpreter Takes Single instruction as input .
Errors are displayed after entire program is checked	Errors are displayed for every instruction interpreted (if any).
Intermediate Object Code is Generated	No Intermediate Object Code is Generated
Memory Requirement is More as Object Code is Generated.	Memory Requirement is Less.
Programming language like Python, Ruby use interpreters.	Programming language like C, C++ use compilers.

History of the C Language



Structure of C language



Documentation Section

This section consists of comment lines which include the name of programmer, the author and other details like time and date of writing the program. Documentation section helps anyone to get an overview of the program.

Link Section

The link section consists of the header files of the functions that are used in the program. It provides instructions to the compiler to link functions from the system library.

Definition Section

All the symbolic constants are written in definition section.

Global Declaration Section
The global variables that can be used anywhere in the program are declared in global declaration section.
main() Function Section
It is necessary have one main() function section in every C program. This section contains two parts, declaration and executable part. The declaration part declares all the variables that are used in executable part

Subprogram Section
The subprogram section contains all the user defined functions that are used to perform a specific task. These user defined functions are called in the main() function.

Comments

Comments are portions of the code ignored by the compiler. The statement that placed inside a comment is not executed as a part of program. Comment is is a note to yourself (or others) that you put into your source code.

Types of Comments

There are two types of comments available:

1) Single-Line Comment – It is used to comment single line in a program.
   The Syntax for single-line comment is:
   // Comment goes here

2) Multi-Line Comment - It is used to comment multiple line in a program and can be placed anywhere in your program.
   The syntax for a **comment** is:
   /*
   Comment goes here
   */

Chapter-2 Variable and Datatypes

C Tokens

The smallest individual unit in a c program is known as a token. C tokens can be classified as follows:

1) Keywords
2) Identifiers
3) Constants
4) Strings
5) Special Symbols
6) Operators

Keywords

Keywords have a special meanings to the compiler and these meaning cannot be changed. In C language there are 32 keywords.

The list of C keywords is given below:

Auto	Break	case	char	const
Continue	Default	do	double	else
Enum	Extern	float	for	goto
If	Int	long	register	return
Short	Signed	sizeof	static	struct
Switch	Typedef	union	unsigned	void
Volatile	While			

Identifier

The user-defined name given to the variable, array and function is called identifier.

There are certain rules that should be followed for c identifier.

- They must consist of only letters(a-z), digits(0-9), or underscore(_). No other special character is allowed.
- It should not start with letters(0-9).
- It should not be a keyword.
- It must not contain white space.
- Identifier name is case sensitive.

Constants

C constants refers to the data items that do not change their value during the program execution. Several types of C constants that are allowed in C are:

1. Integer Constants

Integer constants are whole numbers without any fractional part.

2. Real Constants

The numbers having fractional parts are called real or floating point constants. These may be represented in one of the two forms called *fractional form* or the *exponent form*

Example of valid real constants in fractional form or decimal notation
0.05, -0.905, 562.05, 0.015

3. Character Constants

A character constant contains one single character enclosed within single quotes.

Examples of valid character constants ‘a’ , ‘Z’, ‘5’

It should be noted that character constants have numerical values known as ASCII values, for example, the value of ‘A’ is 65 which is its ASCII value.

Escape Characters/ Escape Sequences

C allows us to have certain non graphic characters in character constants. These non graphic characters can be represented by using escape sequences represented by a backslash() followed by one or more characters.

Escape Sequence	Description
A	Audible alert(bell)
B	Backspace
F	Form feed
N	New line
R	Carriage return
T	Horizontal tab
V	Vertical tab
\	Backslash
“	Double quotation mark
‘	Single quotation mark
?	Question mark
\0	Null

String constant

String constants are sequence of characters enclosed within double quotes. For example,
“hello”
“abc”
“hello911”

Every sting constant is automatically terminated with a special character **‘’** called the **null character** which represents the end of the string.

Special Symbols

The following special symbols are used in C having some special meaning and thus, cannot be used for some other purpose.

[] () {} , ; : * … = #

Braces{}: These opening and ending curly braces marks the start and end of a block of code containing more than one executable statement.

Parentheses(): These special symbols are used to indicate function calls and function parameters.

Operators

C operators are symbols that triggers an action when applied to C variables and other objects. The data items on which operators act upon are called operands.

Depending on the number of operands that an operator can act upon, operators can be classified as follows:

1. **Unary Operators:** Those operators that require only single operand to act upon are known as unary operators.
2. **Binary Operators:** Those operators that require two operands to act upon are called binary operators.
3. **Ternary Operators:** These operators requires three operands to act upon.

Variables and Data types

Variable

Variables are used to store the value during the execution of a program. A variable is nothing but a name given to a storage area that our programs can manipulate. Variable name is known as a identifier and follows the same rules as identifier to given a name.

Datatypes

Each variable in C has a specific type, which determines the size and layout of the variable's memory. A variable definition tells the compiler where and how much storage to create for the variable.

There are three different types of datatype available in a c language.

1) Primitive Datatypes
2) Dervied Datatypes
3) User-defined Datatypes

Primitive Datatypes

There are basic five primitive datatypes available in a c language.

- int %d 2 bytes
- float %f 4 bytes
- double %lf 8 bytes
- char %c 1 byte

In C language we have two types of quialifier available

1) Sign quialifier – signed , unsigned
2) Size qualifier – short , long

Data Type	Size	Value Range
Char	1 byte	-128 to 127 **or** 0 to 255
unsigned char	1 byte	0 to 255
signed char	1 byte	-128 to 127
Int	2 or 4 bytes	-32,768 to 32,767 **or** -2,147,483,648 to 2,147,483,647
unsigned int	2 or 4 bytes	0 to 65,535 **or** 0 to 4,294,967,295
Short	2 byte	-32,768 to 32,767
unsigned short	2 byte	0 to 65,535
Long	4 byte	-2,147,483,648 to 2,147,483,647
unsigned long	4 byte	0 to 4,294,967,295

Derived Datatypes

- Array
- Pointer
- Function
-

User-defined Datatypes

- structure
- union
- enum

Chapter – 3 Operaors

An operator is a symbol which operates on a value or a variable. C programming has wide range of operators to perform various operations.

Types of Operators

1) Arithmetic Operators + - * / %
2) Relational Operator > >= < <= == !=
3) Logical Operator && || !
4) Assignment Operators =
5) Increment and Decrement ++ --
6) Conditional Operator ? :
7) Binary Operator >> << & | ^
8) Special Operator sizeof

Arithmetic Operator
Addition +
Substraction –
Multiplication *
Division /
Modulo %

Relational Operator

Relational Operator are used check the condition between two operand. If condition is satisfied then return true(1) value else return false(0).

Relational operator are used in a decision making and loops.

Operator	Meaning of Operator	Example
==	Equal to	5 == 3 returns 0
>	Greater than	5 > 3 returns 1
<	Less than	5 < 3 returns 0
!=	Not equal to	5 != 3 returns 1

Operator	Meaning of Operator	Example
>=	Greater than or equal to	5 >= 3 returns 1
<=	Less than or equal to	5 <= 3 return 0

Example

```c
#include <stdio.h>
void main()
{
    int a = 5, b = 5, c = 10;

    printf("%d == %d = %d \n", a, b, a == b); // true
    printf("%d > %d = %d \n", a, b, a > b); //false
    printf("%d < %d = %d \n", a, b, a < b); //false
    printf("%d >= %d = %d \n", a, b, a >= b); //true
    printf("%d != %d = %d \n", a, b, a != b); //false
    printf("%d != %d = %d \n", a, c, a != c); //true
    printf("%d <= %d = %d \n", a, b, a <= b); //true
}
```

Logical Operator

Logical Operator are used to combine multiple condition. It return the boolean values based on condition.

Operator	Meaning of Operator	Example
&&	Logial AND. True only if all operands are true	If c = 5 and d = 2 then, expression `((c == 5) && (d > 5))` equals to 0.
||	Logical OR. True only if either one operand is true	If c = 5 and d = 2 then, expression `((c == 5) || (d > 5))` equals to 1.

Operator	Meaning of Operator	Example
!	Logical NOT. True only if the operand is 0	If c = 5 then, expression `! (c == 5)` equals to 0.

Example

```
#include <stdio.h>
void main()
{
    int a = 5, b = 5, c = 10, result;

    result = (a == b) && (c > b);
    printf("(a == b) && (c > b) equals to %d \n", result);

    result = (a == b) || (c < b);
    printf("(a == b) || (c < b) equals to %d \n", result);

    result = !(a != b);
    printf("!(a == b) equals to %d \n", result);

}
```

Assignment Operator

Assignment Operator is used to assign value to the variable.
Eg – a=10;
In assignment operator always right side value is assigned to left side. If there is a same variable in left and right side of the variable then we can also write in a short hand representation.
a=a+10

a+=10

Increment/Decrement Operator

C programming has two operators increment ++ and decrement -- to change the value of an operand (constant or variable) by 1.

Increment ++ increases the value by 1 whereas decrement -- decreases the value by 1.

There are two types of increment and decrement operator
b=a++; // post increment
b=++a; // pre increment

b=a--; // post increment
b=--a; // pre increment

Conditional Operator

A conditional operator is a ternary operator, that is, it works on 3 operands.

Conditional Operator Syntax

```
conditionalExpression ? expression1 : expression2
```

The conditional operator works as follows:

- The first expression conditionalExpression is evaluated first. This expression evaluates to 1 if it's true and evaluates to 0 if it's false.
- If `conditionalExpression` is true, `expression1` is evaluated.
- If `conditionalExpression` is false, `expression2` is evaluated.

```
#include <stdio.h>
void main(){
   char February;
   int days;
   printf("If this year is leap year, enter 1. If not enter any
integer: ");
   scanf("%c",&February);
```

```
   days = (February == '1') ? 29 : 28;


   printf("Number of days in February = %d",days);
}
```

Special Operator

The **sizeof** is an unary operator which returns the size of data (constant, variables, array, structure etc).

It returns the no of bytes variable used in memory(depend on datatypes).

```
#include <stdio.h>
void main()
{
    int a;
    float b;
    double c;
    char d;
    printf("Size of int=%d bytes\n",sizeof(a));
    printf("Size of float=%d bytes\n",sizeof(b));
    printf("Size of double=%d bytes\n",sizeof(c));


}
```