Unit 1

Introduction to C Language

1.0 Introduction

C is a programming language developed at AT's &T's BELL laboratories of USA in 1972.

It was designed and written by a man named "DENNIS RITCHE".

C is a general purpose computer programming language.

It was used in operating systems, devices drives, but used in a wide range of application domains.

C is commonly used on computer architectures that range from the largest supercomputers to the smallest microcontrollers and embedded systems.

C programming is mainly used to develop UNIX Operating System.

C was originally developed at Bell, to construct uitilities running on Unix.

It was applied to re-implementing the kernel of Unix Operating System.

C programming is considered as the base for the programming languages. So it is called as Mother Language.It can be defined by the following ways:

- Mother Language
- System Programming Language
- Procedure-Oriented Programming Language
- Structured Programming Language
- Mid-Level Programming Language

Mother Language :

C Programming Language is considered as mother Language for all the modern Programming Languages. Because, most of the compilers like JVMS, kernels, etc.. are written in C Language and also most of the programming language follows C syntax. Eg: C++, Java, c#..etc.

System Programming Language:

C Language is used to create system software. Because, it can be used to do low-level programming and also used to create hardware devices, os drivers, kernals,etc...

Procedural-Oriented Programming Language:

C is a procedural programming language. Procedure is known as function, method, routine, subroutine. Procedural language specifies a series of steps for the program to solve the program.

Structured Programming Language :

It is a subset of procedural programming language. Structure means to break a program into parts or blocks .So that it may be easy to understand.
In c program, we break the program into parts by using functions. It makes program easier to understand and modify.

Mid-Level Programming Language:

C is a Mid-level programming language .because, it supports the features of both low-level and high-level languages.

Low level	High level
1. It is known as computer's native language	1. It is a programming language that means it is user friendly
2. These languages are considered high memory efficient.	2. These languages are considered as less memeory efficient.
3. It is difficult to Comprehend.	3. It is simple to Comprehend.
4.It is complex to debug.	4. It is straight forward to troubleshoot.
5.It is Difficult to maintain	5. It is easy to maintain
6.It is Non-Portable.	6.It is portable
7.Entirely dependent on the machine	7. Compatible with all operating systems.
8.Assembler is required for translation	8.Compiler or Interpreter is required

Features of C Programming :

- Simple
- Recursion
- Memory Management
- Fast & Efficient
- Modularity
- General Purpose Language
- Rich set of built in operations
- Portability
- Easy to extend
- Rich in libraries
- Middle level language
- Statically type

Characterstics:

- Small Size
- Extensive use of function calls
- Structured language
- Powerful programming language
- Portability
- Case Sensitive

Applications of C language:

- **C** language is used to create computer application software like Databases, Spreadsheets.
- Used to create system software as Operating systems.
- To create applications related to graphics like computer & mobile game.
- UNIX kernel has been fully developed in cLanguage.
- Used to create Network devices & Device drivers.
- Used to make a compiler.

Syntax:

```
Header files

Main method()

{
    Statement 1;
    Statement 2;
    return 0;
}
```

// Basic programm

```
#include<stdio.h>
#include<conio.h>
  void main()
{
   printf("hello world");
   return 0;
}
```

Output: hello world

1.1 History and Structure of C-language program

History

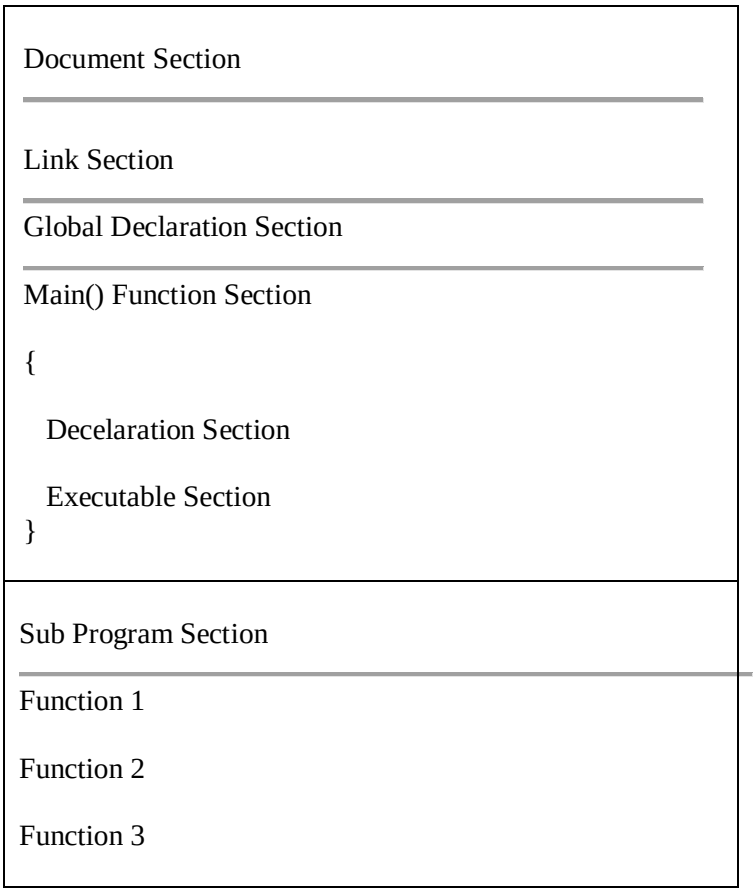
- The root of all modern language is ALGO(Algorithmic language), it is first computer programming language to use block structure.
- MARTIN RICHARDS developed a language called "BCPL"(Basic combined programming language).It is derived from "ALGO".
- In 1970 ken Thompson created "B" language by using BCPL.
- Both BCPL & B programming languages were typeless.
- After this "C" was deveploed using 'BCPL' &'B' by Dennis Ritchie at the bell lab in 1972.

Language	Year	Developed by
ALGO	1960	Internationl Group
BCPL	1967	Martin Richard
B	1970	Ken Thompson
Traditional C	1972	Dennis Ritiche
K & R c	1978	Kernighan & Dennis Ritchie
ANSI c	1989	ANSI committee
ANSI / ISO c	1990	ISO committee
C 99	1999	Standardization committe

Structure of C-language program

C program can be viewed as a group of building blocks called functions. A function is a subroutine that may include one or more statements designed to perform a specific task.

There is a specific structure to start the programming in c language. Without structure it is difficult to analyze the program and the solution. It gives the basic idea of the order of the sections in a program, when and where to use a particular statements, variables, functions, curly braces and paranthesis.



Document Section:

The Documentation Section Consist of a set of comment lines giving the name of the program, the author and other details, which the programmer would like to use latter.

Comment lines: It indicates the purpose of the program. It is represented as
/*……..*/
/……..//
Comment line is used for increasing the readability of the program. It is useful in explaining the program and generally used for documentation. It is enclosed within the decimeters. Comment line can be single or multiple line but should not be nested. It can be anywhere in the program except inside string constant & character constant.

Link section or Preprocessor Directive:

include tells the compiler to include information about the standard input/output library. It is also used in symbolic constant such as #define PI 3.14(value). The stdio.h (standard input output header file) contains definition &declaration of system defined function such as printf( ), scanf( ), pow( ) etc. Generally printf() function used to display and scanf() function used to read value.

Global Declaration:

This is the section where variable are declared globally so that it can be access by all the functions used in the program. And it is generally declared outside the function

Main() :

It is the user defined function and every function has one main() function from where actually program is started and it is encloses within the pair of curly braces. The main( ) function can be anywhere in the program but in general practice it is placed in the first position.

Syntax :

```
main()

{ ……..
  ……..
 ……..
}
```

The main( ) function return value when it declared by data type as

```
 int main( )

{
  return 0

 }
```

The main function does not return any value when void (means null/empty) as void main(void) or void main()

```
{

   printf (“C language”);
```

```
}
```

Output: C language

The program execution start with opening braces and end with closing brace.
And in between the two braces declaration part as well as executable part is mentioned. And at the end of each line, the semi-colon is given which indicates statement termination.

/*First c program with return statement*/

```
#include<stdio.h>

int main (void)

{

printf ("welcome to c Programming language.\n");

return 0;

}
```

Output: welcome to c programming language.

1.2 Programming style of C Language:

It is important for writing code that is readable,maintainable and less error-prone.

Some aspects of c programming style:

1.**Intendation:**

In C language we use consistent and clear indentation to make your code more readable.The standard is to use either tabs or spaces typically with 4- space indent

2.Brace Style:

There are two common brace style “K & R” and “ALLMAN”

K & R:

```
if (condition){

} else {

//code here

}
```

Allman

if (condition)

{
……
…….
}
else
{
……
……
}

3. Naming Convention:

1. Use meaningful variable and function names.
2. Use lowercase for variable & function name .Ex: (my_variable, calculating-sum)
3. Use uppercase for constants .Ex: MAX_VALUES
4. Use underscore to separate words in names.

4. Comments:

Comments explains logic &non-obvious code.

//….. single line comments
/*multiline comments*/

Comment line is used for increasing the readability of the program. It is useful in explaining the program and generally used for documentation. It is enclosed within the decimeters. Comment line can be single or multiple line but should not be nested. It can be anywhere in the program except inside string constant & character constant.

5. White Space :

In C program we use white space to improve readability. White spaces are used around operators

Ex: a = b + c ; instead of a=b+c;

After comma also we use white spaces
Ex: printf (“Hello, World”);

6. Function Structure :

Organize your functions logically with a clear structure.

Ex: 1. Start with function prototype.
2. follow with the main function.

3. keep functions short & focused on a single task.

7. Consistency :

Be consistent in your code style throughout the project. This makes it easier for others to read & understand your code.

8. Error Handling :

Check for errors and handle the appropriately. Don't ignore errors return values from functions.

9. Include Gaurds :

We use include gaurds in header file to prevent multiple inclusion of the same file.

10. Limit line length:

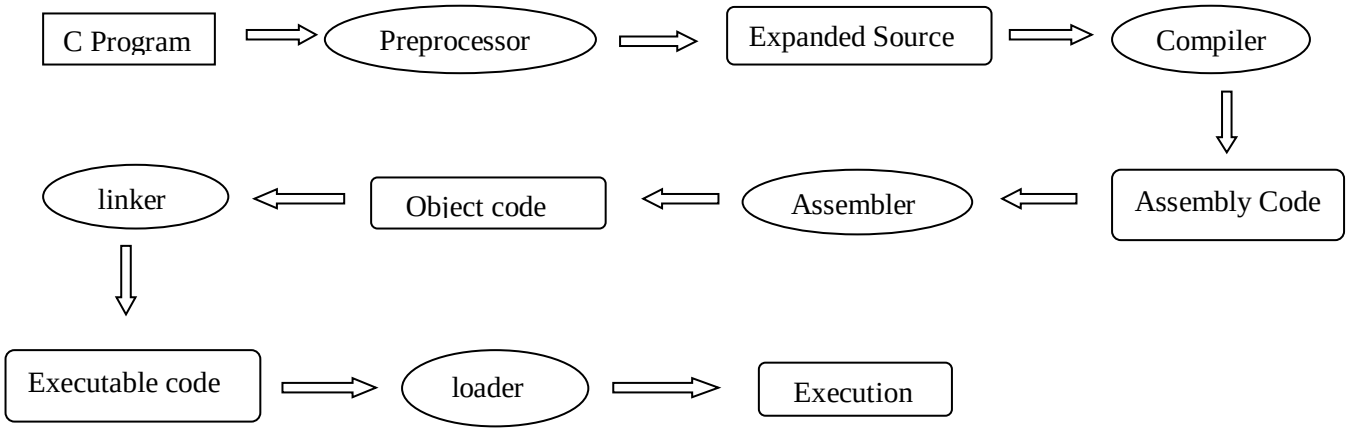
Avoid excessively long lines of code. Lines should be limited to 80-120 characters

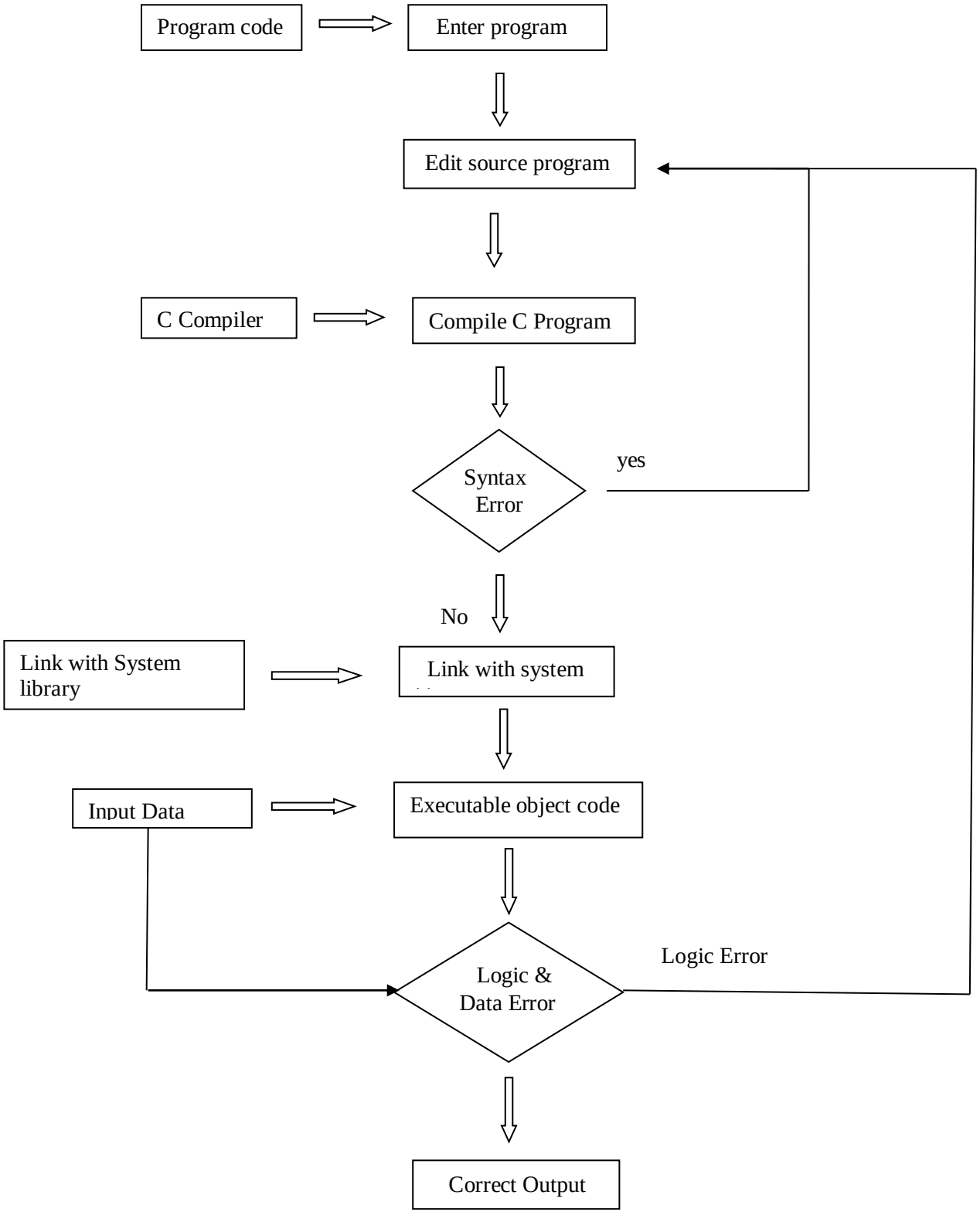
11. Use Meaningful whitespace:

Add spaces to enhance readability, especially around code blocks.

1.3 Steps Involved in Editing , Compiling ,Executing and Debugging of C Program

- Every file that contains a C program must be saved with 'C' extension.
- It is necessary for the compiler to understand that is a C program file.
- For example, file name is saved as first.c, is called source file, which keeps the code of the program.
- Source file is sent to preprocessor, Preprocessor is responsible to convert preprocessor directives.
- The preprocessor generates an expanded source code.
- Expanded source code is sent to compiler which compiles the code & check for errors and converts it into assembly code.
- Assembly code is sent to assembler then the code converts into object code, file.obj . it is not executable
- The object code is sent to linker which links it to the library such as headerfiles.
- Then it converts into executable code first.exe file is generated.
- Executable code is sent to loader which loads it into memory and then it executes, output is sent to console.




Program code
Enter program
Edit source program
C Compiler
Compile C Program
Syntax Error
yes
No
Link with System library
Link with system
Input Data
Executable object code
Logic & Data Error
Logic Error
Correct Output

When the program is executed, each of the statements of the program is sequentially executed in turn. If the program requests any data from the user, known as input, the program temporarily suspends its execution so that the input can be entered. Or, the program might simply wait for an event, such as a mouse being clicked, to occur. Results that are displayed by the program, known as output, appear in a window, sometimes called the console. If the program does not produce the desired results, it is necessary to go back and reanalyze the program's logic. This is known as the debugging phase, during which an attempt is made to remove all the known problems or bugs from the program. To do this, it will most likely be necessary to make changes to original source program.

1.4 Character set , C-Tokens, Keywords, Identifiers, Constants, Variables

Character set

A character denotes any alphabet, digit or special symbol used to represent information. Valid alphabets, numbers and special symbols allowed

Characters that are typically available on most keyboards and character encoding.

1.Alphabets : Upper case (A – Z)
Lower case (a – z)
2. Digits : 10 decimal digits (0 – 9)

3. Special characters
4. White spaces
5. Escape Sequence

C – Tokens

Generally in a passage of text, Individual words and Pnctuations marks are called "Tokens". Similarly in C program the smallest individual units are called as C tokens.

C has 6 types of tokens:

- Keywords
- Constants
- Identifiers
- Special symbols
- Strings
- Operators

We cannot create a sentence without words ,Similarly we cannot create a C program without using tokens. We can say that tokens in c is the building block or the basic component for creating a program in c language.

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.C language supports 32 key words, Some examples are **int, short, signed, unsigned, default, volatile, float, long, double, break, continue, typedef, static, do, for, union, return, while, extern, register, enum, case, goto, struct, char, auto, const, else, if, sizeof, void, switch.**

Identifiers

Identifiers are user defined word used to name of entities like variables, arrays, functions, structures etc. It can be composed of Uppercase letters, lowercase letters , Underscore and digits. Identifiers cannot be used as keywords.
Rules for naming identifiers are:
1) Name should only consists of alphabets (both upper and lower case), digits and underscore (_) sign.
2) First characters should be alphabet or underscore
3) Name should not be a keyword
4) Since C is a case sensitive, the upper case and lower case considered differently, for example code, Code, CODE etc. are different identifiers.
5) 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.

Strings

Strings in C are always represented as an array of characters having null character '/0' at the end of the strings. Strings are used for storing text/characters and they are enclosed with double quotes.

C does not have a string type to easily create string variables, so we use char type and create an array of characters to make a string in c. Characters are enclosed with single quotes.

The size of the string is number of characters that the string contains. The format Specifier " %s " to tell c that we are now working with strings, and format specifier "%c" is used to print single character. "/0" null terminating character must be included when creating the strings. It tells C that this is end of the string.

Describing the strings in different ways

char a [10] = "program"; //compiler allocates 10 bytes to the 'a' array.
char a [] = "program";//compiler allocates the memory at the runtime.
char a [10] = {'p','r','o','g','r','a','m','m','/0'};// string is represented in the form of characters.

```
#include<stdio.h>
int main ()
{
  char strings [] = "hello world";
   printf( "%s",strings);
}
```

Out put: hello world

```
#include<stdio.h>
int main ()
{
  char first_letter [] = " Hello world";
  printf("%c",first_letter [0]);
  return 0;
}
```

Output: H

```
#include<stdio.h>
int main ()
{
  char new_string [] = { ‘h’, ‘e’, ‘l’, ‘l’, ‘o’, ‘/0’};
  printf(“%s”,new_string);
  return 0;
}
```

Otput: hello

Special symbols

Some special characters are used in c, and they have a special meaning which cannot be used for another purpose.

1. Square brackets [ ] : The opening and closing brackets represent the single and multidimentional scripts

2. Simple brackets or parenthesis ( ) : Parenthesis used in function declaration and function calling.

3.Curly braces {} : Curly braces are used in opening and closing of the code containing more than one executable statements.

4. Comma (,) : Comma is used for separating for more than one statement. For example : Separating function parameters in a function call, Variables when printing the value of more than one variable.

5.Preprocessor (#) : Preprocessor is used for pre-processor directive , It is used automatically by the compiler to transform your program before actual compilation .

6. Asterisk(*) : Asterisk used to represent pointers and also used as an operator for multiplication.

7. Colon (:) : Colon is an operator that essentially invokes something called an initialization list.

8. Semi colon (;) : Semicolon known as statement terminator , it indicates the end of one logical entity.

9. Assignment Operator (=) : Assignment Operator is used to assign values and for logical operation validation.

10. Period (.) : Period is used to access members of a structure or union.

11. Tidle(~) : Tidle is used as a destructor to free some space from memory.

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.

float – stores floating point numbers

int - stores interger
char – stores single characters

Syntax:

```
Data type   Variable_Name =  Value ;
```

For eg:

```
 int  a = 15;
 int my_num = 20;
float f = 0.98;
char  c = ‘ a’;
```

Rules for creating a variable:

1. A variable can have alphabets, digits, and underscore .
2. Variable name can start with alphabet & underscore.
3. No whitespace is allowed within the variable name.
4. A variable name must not be any reserved word or a keyword.

// Programm for Declaring or creating variables.

```
#include<stdio.h>
int main ( )
{
   // creating variables

   int  my_num = 15;
   float my_num = 15.89;
  char my_letter = ‘k’;
// print variables
   printf (“ %d\n”,my_num);
   printf(“%f\n”,my_num);
   printf(“%c\n”,my_letter);
   return o;
}
```

Output :

```
15
15.89
k
```

// program for changing the variable values.

```
#include<stdio.h>
int main ( )
{
  int mynum = 13;
```

```
mynum = 34;
printf("%d",mynum);
return 0;
}
```

Output : 34

// Programm for assigning value to another number

```
#include<stdio.h>
int main ( )
{
  int a = 14;
  int b = 56;
  a = b;
  printf ("%d",a);
  return 0;
}
```

Output : 56

// program for multiple variables

```
#include<stdio.h>
int main ( )
{
   int x = 5, y = 8, z = 6;
   printf ("%d",x+y+z);
   return 0;
}
```

Output : 19

Types of Variable in C :

There are 5 types in variables they are:

1. Local variable
2. Global variable
3. Static variable
4. Automatic variable
5. External variable

Local Variable

Local variable in c is a variable that is declared inside a function or a block of code. They can be used only by statements that are inside that function or block of code. Local variables are not known to functions outside their own. Its scope is limited to the block or function in which it is declared.

```
#include<stdio.h>
void function ( )
```

```
{
   int x = 10;    // local variable
   printf (“%d”, x);
}
 int main ( )
{
  function ( );
}
```

(or)

```
#include<stdio.h>
 int main ( )
{
   int x = 10;
  printf (“%d”, x);
}
```

Output : 10

Output : 10

Global Variable :

A global variable in c is a variable that is declared outside the function or a block of code. usually on top of the program. Global variables hold their values throughout the lifetime of your program and they can be accessed inside any of the functions defined for the program. A global variable can be accessed by any function.
Its scope is the whole program.

```
#include<stdio.h>
int  x = 10;
void  function1 ( )
{
  printf (“%d\n”, x);
}
 void function2 ( )
{
   printf (“%d\n”,x);
}
int main ( )
{
  function1 ( );
  function2 ( );
}
```

Output: 10
10

Static Variable :

Static variable in C is a variable that is defined using the static keyword. It can be defined only once in a C program and its scope depends upon the region where it is declared.(can be global or local)

Syntax:

```
Static   data type  variable_name = value;
```

```
#include<stdio.>
 void function1 ( )
```

```
{
   int  x = 10;
   static int y = 20;
   x = x + 1;
   y = y + 1;
   printf (“%d %d”, x,y);
}
 int main ( )
{
  function1 ( );
}
```

Output : 11 21

```
#include<stdio.h>
void function ( )
{
  int x = 20;
  static int y = 30;
x = x+10;
y = y+10;
printf (“\t local : %d\n \t static : %d\n”,x,y);
}
 int main ( )
{
  printf (“ first call \n”);
   function ( );
  printf(“Second call \n”);
   function ( );
 printf (“Third call\n”);
  function ( );
}
```

Output :

```
First call
     local : 20
     static : 30
Second call
     local : 20
     static :40
Third call
     local : 20
     static : 50
```

Note : local variable print the same value for each function call but, static will print incremented value.

Automatic variable

All the local variables are automatic by default ,They are also known as auto variables. Their scope is local and their life time is till the end of the block. We can use ‘ auto’ keyword to define the auto variables . The default value of the auto variables is a garbage value.

Syntax :

```
auto  data type  variable name  = value;
```

```
#include<stdio.h>
void function ( )
{
   int x = 10;
   auto int y = 20;
   printf (“Auto variable : %d”,y);
 }
  int main ( )
{
 function ( );
 return 0;
}
```

Out put: 20

External variable:

External variable can be shared between multiple C files. We can declare an external variable using the extern keyword. Their scope is global.

Syntax

```
Extern  data type variable name = value;
```

```
#include<stdio.h>
extern int x = 10;
int main ( )
{
   printf (“%d”,x);
   return 0;
}
```

Output : 10

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.

Syntax :

```
Const  data type   var_name  = value ;
```

Eg: const int a = 15;

Defining the constants : there are two ways to define the constants. They are

1. Const keyword
2. #define preprocessor

Const keyword : by using the keyword const we can declare the constants.

```
#include<stdio.h>
int main ( )
{
 const float pi = 3.14;
printf("the value of pi is : %f",pi);
return 0;
}
```

Output
the value of pi is : 3.14

if we try to change the value of pi it will remained compile time error.

#define Preprocessor
#define preprocessor is also used to define constants.

```
#include<stdio.h>
#define pi 3.14
int main ( )
{
printf("%f",pi);
return 0;
}
```

Types of Constants :

Basically there are two types of constants they are: **Numeric constant**
Character constant

Numeric constant: Numeric constant consists of digits. It required minimum size of 2 bytes and max 4 bytes. It may be positive or negative but by default sign is always positive. No comma or space is allowed within the numeric constant and it must have at least 1 digit.

It is categorized a integer constant and real / floating constant.

Integer constant
An integer constants are whole number which have no decimal point. Types of integer constants are:
Decimal constant: 0-------9(base 10)
Octal constant: 0-------7(base 8)
Hexa decimal constant: 0----9, A------F(base 16)
Binary constant: 0 & 1(base 2)

Real / floating constant
Real constant is also called floating point constant. To construct real constant we must follow the rule of ,
-real constant must have at least one digit.
-It must have a decimal point.
-It could be either positive or negative.
-Default sign is positive.
-No commas or blanks are allowed within a real constant.
Ex.: +325.34 426.0 -32.76

Real/ floating points are two types:

1.Decimal form or real constants :
It contains a decimal part eg : 3.14,-1223.456
2.Exponential form :
Exponential form represents decimals or fractions in another way.
Eg:
$123e^{12} = 123 \times 10^{12}$
$123e^{-12} = 123 \times 10^{-12}$

Character constant

A character constant is written as one character with in single quotes such as 'a'. The value of a character constant is the numerical value of the character in the machines character set.
There are 4 types of constants:

1. Single Character Constant : A character which has been specified in a single quotes.
   Eg : 'A', '7', '$'

```
#include<stdio.h>
int main ()
{
char ch = 'a';
printf("character constant is : %c\n",ch);
return 0;
}
```

Out put: character constant is : a

2. String Character Constant : A group of characters which are enclosed with double quotes.
   Eg: "hello" , "5+0", " hello world".

3. Backslash Character Constant : Backslash characters which are executed directly by compiler.
   Eg : \n – new line.
   \t – horizontal tab
   \v-vertical tab
   \r-carriage return
4. Enumeration Character Constant : Enumeration constants are user defined constants created using 'enum' keyword. They are often used to create sets of named integer constants.
   Eg : enum days { Sunday,Monday}.

Data Types

Each variable in C has an associated data type. It specifies the type of data that the variable can store like integer, character, floating, double, etc. Each data type requires different amounts of memory and has some specific operations which can be performed over it. The data type is a collection of data with values having fixed values, meaning as well as its characteristics.

The data types in C can be classified as follows:

Primitive or Primary Data Types : Primitive data types are the most basic data types that are used for representing simple values such as integers, float, characters, etc.

User Defined Data Types : The user-defined data types are defined by the user himself.

Derived Types : The data types that are derived from the primitive or built-in datatypes are referred to as Derived Data Types.

Primitive data type	Integer, character, floating point, Double floating point,void.
User defined data type	Structure, Union, Enum, Typedef.
Derived data type	Function ,Pointer,Array

Integer Data Type

The integer datatype in C is used to store the integer numbers(any number including positive, negative and zero without decimal part). Octal values, hexadecimal values, and decimal values can be stored in int data type in C.

- **Range:** -2,147,483,648 to 2,147,483,647
- **Size:** 4 bytes
- **Format Specifier:** %d

Syntax of Integer

We use **int keyword** to declare the integer variable:
int *var_name;*

The integer data type can also be used as

1. **unsigned int:** Unsigned int data type in C is used to store the data values from zero to positive numbers but it can't store negative values like signed int.
2. **short int:** It is lesser in size than the int by 2 bytes so can only store values from -32,768 to 32,767.
3. **long int:** Larger version of the int datatype so can store values greater than int.
4. **unsigned short int:** Similar in relationship with short int as unsigned int with int.

```
// C program to print Integer data types.
#include <stdio.h>

int main()
{
  int a = 9;
  int b = -9;
 int c = 89U;
 long int d = 99998L;
printf("Integer value with positive data: %d\n", a);
printf("Integer value with negative data: %d\n", b);
printf("Integer value with an unsigned int data: %u\n",c);
printf("Integer value with an long int data: %ld", d);
return 0;
}
```

Output

```
Integer value with positive data: 9
Integer value with negative data: -9
Integer value with an unsigned int data: 89
Integer value with an long int data: 99998
```

Character Data Type

Character data type allows its variable to store only a single character. The size of the character is 1 byte. It is the most basic data type in C. It stores a single character and requires a single byte of memory in almost all compilers.

- **Range:** (-128 to 127) or (0 to 255)
- **Size:** 1 byte
- **Format Specifier:** %c

Syntax of char

The **char keyword** is used to declare the variable of character type:

```
char var_name;
// C program to print Integer data types.
#include <stdio.h>

int main()
{
        char a = 'a';
        char c;

        printf("Value of a: %c\n", a);

        a++;
        printf("Value of a after increment is: %c\n", a);

        // c is assigned ASCII values
        // which corresponds to the
        // character 'c'
        // a-->97 b-->98 c-->99
```

```
        // here c will be printed
        c = 99;

        printf("Value of c: %c", c);

        return 0;
}
```

Output

```
Value of a: a

Value of a after increment is: b

Value of c: c
```

Float Data Type

In C programming float data type is used to store floating-point values. Float in C is used to store decimal and exponential values. It is used to store decimal numbers (numbers with floating point values) with single precision.

- **Range:** 1.2E-38 to 3.4E+38
- **Size:** 4 bytes
- **Format Specifier:** %f

Syntax of float

The **float keyword** is used to declare the variable as a floating point:

float *var_name;*

```
// C Program to demonstrate use
// of Floating types
#include <stdio.h>

int main()
{
        float a = 9.0f;
        float b = 2.5f;

        // 2x10^-4
        float c = 2E-4f;
        printf("%f\n", a);
        printf("%f\n", b);
        printf("%f", c);

        return 0;
}
```

Output

```
9.000000

2.500000

0.000200
```

Double Data Type

A Double data type in C is used to store decimal numbers (numbers with floating point values) with double precision. It is used to define numeric values which hold numbers with decimal values in C.

It can easily accommodate about 16 to 17 digits after or before a decimal point.

- **Range:** 1.7E-308 to 1.7E+308
- **Size:** 8 bytes
- **Format Specifier:** %lf

Syntax of Double

The variable can be declared as double precision floating point using the **double keyword:**
double *var_name;*

```
// C Program to demonstrate
// use of double data type
#include <stdio.h>
int main()
{
        double a = 123123123.00;
        double b = 12.293123;
        double c = 2312312312.123123;
         printf("%lf\n", a);
         printf("%lf\n", b);
         printf("%lf", c);
        return 0;
}
```

Output

```
123123123.000000
12.293123
2312312312.123123
```

Data Type	Size (bytes)	Range	Format Specifier
short int	2	-32,768 to 32,767	%hd
unsigned short int	2	0 to 65,535	%hu
unsigned int	4	0 to 4,294,967,295	%u
int	4	-2,147,483,648 to 2,147,483,647	%d
long int	4	-2,147,483,648 to 2,147,483,647	%ld

unsigned long int	4	0 to 4,294,967,295	%lu
long long int	8	-(2^63) to (2^63)-1	%lld
unsigned long long int	8	0 to 18,446,744,073,709,551,615	%llu
signed char	1	-128 to 127	%c
unsigned char	1	0 to 255	%c
float	4	1.2E-38 to 3.4E+38	%f
double	8	1.7E-308 to 1.7E+308	%lf
long double	16	3.4E-4932 to 1.1E+4932	%Lf

Derived Data Types

Derived data types give the programmer the ability to handel heterogeneous data, directly modify memory and build complicated data structures.

Derived data types are classified as :

1. Arrays
2. Functions
3. Pointers
4. Structure

Arrays : Arrays are used to store multiple values in a single-variable, instead of declaring separate variables, It provides a mechanism for joining multiple targets of the same data type under the same name.

Index is used to access the elements of an array the size of array is fixed at declaration time and it cannot be changed.

Ex: int a [ ] = { 1,2,3,4,5}

Functions : **A** Function is a set of statements that when we called perform some specific task. It is the basic building block of c program that provides modularity and code reusability.

Pointers : A ***pointer*** is a derived data type that keeps track of another data type's memory address. When a ***pointer*** is declared, the ***data type*** it refers to is ***stated first***, and then the ***variable name*** is preceded by ***an asterisk (*)***.
Data type asterisk (*) variable name ;

Structure : Structure allows group of many data types under a single name .It gives you the ability to create your own unique data structures.

Structure members can be accessed by using dot(.) operator.

User defined data types :

The data types that are defined by the user are called the user defined data types.
These are classified into 4 types.

1. Structure
2. Union
3. Enumerations
4. Typedef

Structure : A Structure is a user defined data type in 'c'. A structure creates a dtatype that can be used to group of items of possibly different types into a single type. It store in different memory.

1. A structure's members or fields are used to refer to each variable within it.
2. Any data type, including different structures, can be a member of a structure.
3. A structure's members can be accessed by using the dot (.) operator.

```
#include <stdio.h>

#include <strings.h>

struct person

{

char name [50];

int citno;

float salary;

}

person1;

int main ( )

{
```

```
Strcpy(person1.name, " ram");

person1.citno = 1984;

person1.salary = 25000;

printf("Name : %s\n",person1.name);

printf("Citizenship : %d\n",person1.citno);

printf("Salary : %f\n",person1.salary);

return 0;

}
```

Out put

Name : Ram

Citizenship : 1984

Salary : 2500.00

Union

Union are similar to structures it is also collection of different data items, but with a common memory. A derived data type called a ***union*** enables you to store various data types in the same memory address. In contrast to structures, where each member has a separate memory space, members of a union all share a single memory space. A value can only be held by one member of a union at any given moment. When you need to represent many data types interchangeably, unions come in handy. Like structures, you can access the members of a union by using the ***dot (.)*** operator.

```
#include<stdio.h>

union job;

{

float salary;

int id;

}

J;

int main ( )

{
```

```
J.salary  = 20000;

J.id = 15;

printf("%f\n",J.salary);

printf("%d\n",J.id);

return 0;

}
```

Enumerations : A set of named constants or ***enumerators*** that represent a collection of connected values can be defined in C using the ***enumeration data type (enum). Enumerations*** give you the means to give names that make sense to a group of integral values, which makes your code easier to read and maintain.

Enumerations are mainly used to assign names to integrals constants, the names makes a program easy to read and maintain.

Here is an example of how to define and use an enumeration in C:

```
#include<stdio.h>

enum week {mon,tue,wed,thur,fri,sat,sun};

int main ( )

{

enum week day;

day = wed;

printf("%d",day);

return 0;

}
```

Output : 2

Typedef : typedef is a keyword used in c programming to provide some meaningful name to the already existing variable in c program it is like alias name.

We can say that this keyword is used to redefine the name of already existing variable.

Syntax : typedef <existing name> <alias name>

```
#include<stdio.h>

Typedef long long ll;

int main ( )

{

l l var = 20;

printf("%d",var);

return 0;

}
```

Out put : 20

Usage of Type Qualifiers :

In C programming language, type qualifiers are the keywords used to modify and change the properties of the variables.

TYPES OF C TYPE QUALIFIERS: There are two types of qualifiers available in C language. They are,

1. const

2. volatile

1. CONST KEYWORD:

- Constants are also like normal variables. But, only difference is, their values can't be modified by the program once they are defined.
- They refer to fixed values. They are also called as literals.
- They may be belonging to any of the data type.

Syntax: const data_type variable_name; (or) const data_type *variable_name;

When a variable is created with const keyword it becomes a constant variable. the value of the constant variable cannot be changed once it is defined.

2.VOLATILE KEYWORD:

When a variable is defined as volatile, the program may not change the value of the variable explicitly.

But, these variable values might keep on changing without any explicit assignment by the program. These types of qualifiers are called volatile.

For example, variable which is used to store system clock is defined as volatile variable .the value of this variable is not changed explicitly in the program but it is changed by the clock routine of the operating system.