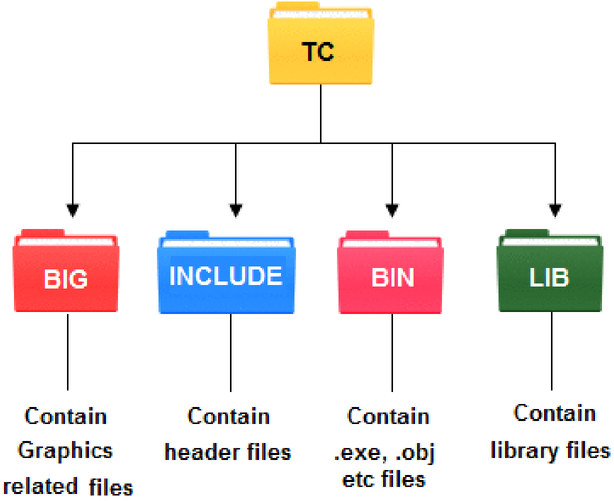
Unit–2
Basics of 'C'

❖ Basics of 'C'

- General structure of 'C' program and standard directories
- Advantages of C language.
- Character set,
- 'C' tokens
- Keywords and Identifiers , Constants and Variables
- Data Types in 'C'
- Rules for defining variables
- Declaration and Initialization
- Dynamic initialization
- Type modifiers and type conversion
- Constant and volatile variable
- Input and Output statements in 'C'
- Write, compile, execute a simple 'C' program

❖ Installation of TC

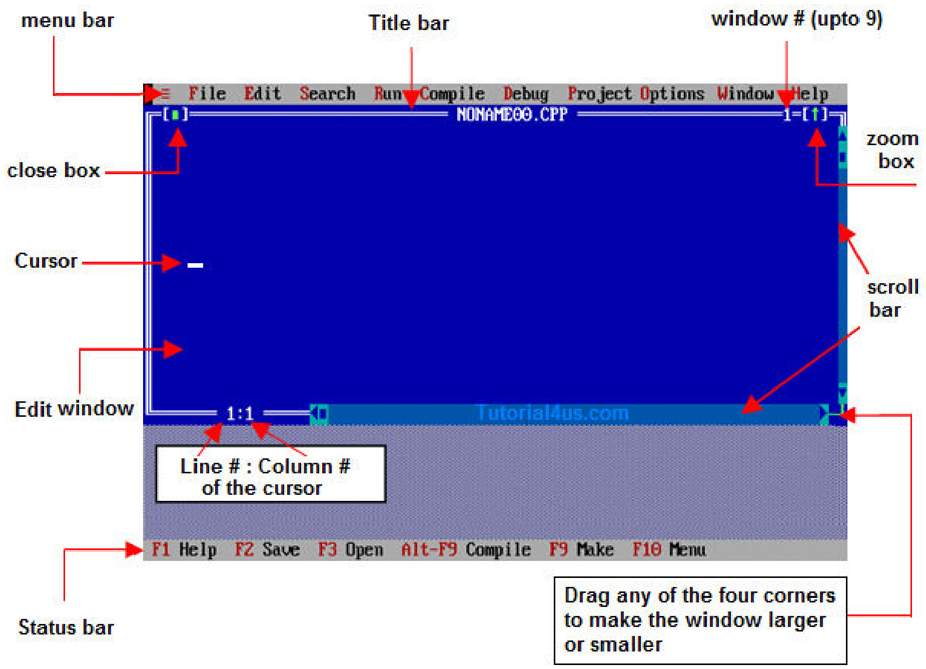
- Installation of TC is very simple just download turbo C or C++ and run .exe files
- When you install the Turbo C compiler on your system, then TC directory is created on the hard disk and various sub directories such as INCLUDE, and LIB etc. are created under TC.



- INCLUDE: Contain the header files of C.
- LIB: Contain the library files of C.
- BGI: Contain Graphics related files.
- BIN: Contain .exe, .obj etc files.

❖ TC Editor

- TC Editor is very simple and easy to use; here i will give you all tips related to TC Editor and some shortcut keys related to TC Editor which is very useful at the time of coding. Turbo C is a most common C language compiler. Below i will discuss all about its Interfaces.


menu bar
Title bar
window # (upto 9)
≡ File Edit Search Run Compile Debug Project Options Window Help
NONAME00.CPP
1=[↑]
close box
zoom box
Cursor
scroll bar
Edit window
1:1
Tutorial4us.com
Line # : Column # of the cursor
F1 Help F2 Save F3 Open Alt-F9 Compile F9 Make F10 Menu
Status bar
Drag any of the four corners to make the window larger or smaller

History of C

- C language is developed by **Mr. Dennis Ritchie** in the year 1972 at bell laboratory at USA
- C is a simple and structure Oriented Programming Language.
- In the year 1988 C programming language standardized by ANSI (American national standard institute), that version is called ANSI-C.
- In the year of 2000 C programming language standardized by ISO that version is called C-99.
- All other programming languages were derived directly or indirectly from C programming concepts.

Prerequisites

- Before learning C Programming language no need of knowledge of any Computer programming language
- C is the basic of all high level programming languages.
- C is also called mother Language.

What Is C?

- C is also called **mother Language** of all programming Language.
- It is the most widely use computer programming language
- This language is used for develop system software and Operating System.
- All other programming languages were derived directly or indirectly from

Where we use C Language/Application

- Design Operating system
- Design Language Compiler
- Design Database
- Language Interpreters
- Utilities
- Network Drivers
- Assemblers

❖ General structure of C program and standard directories

Documentation Section
Link Section
Definition Section
Global Declaration Section

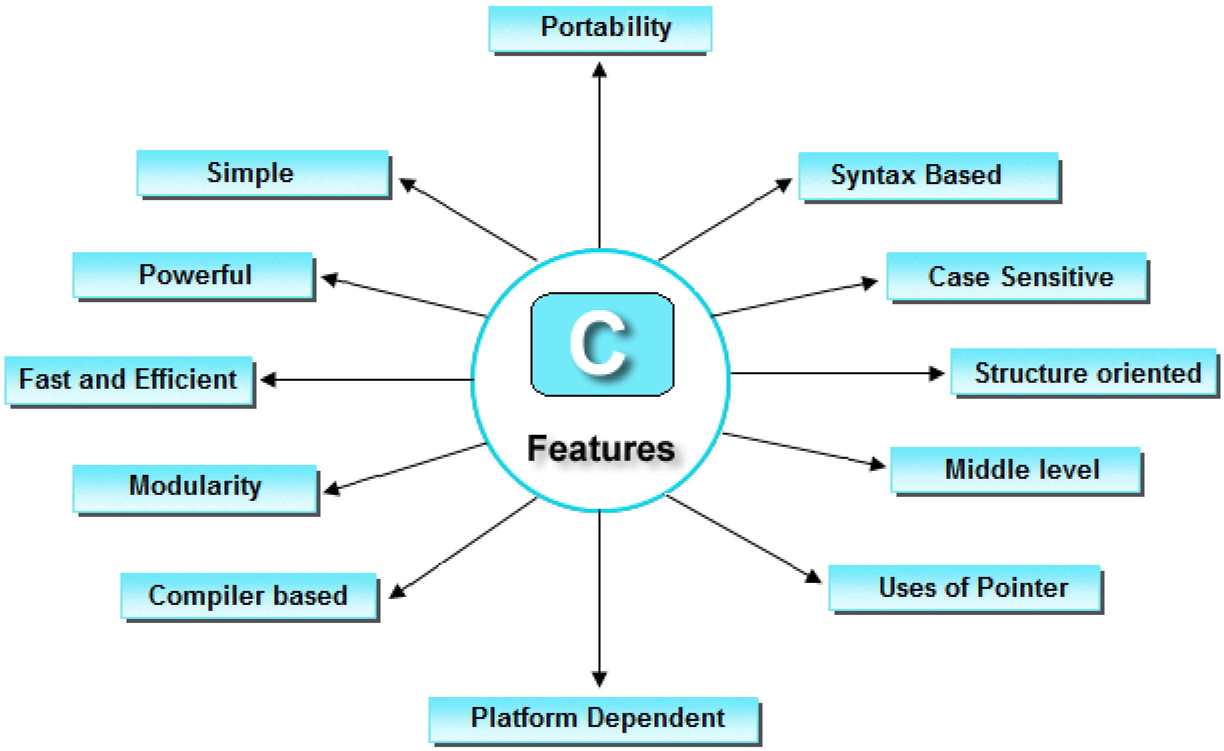
main()
{
Declaration Section
Executable part
}

Subprogram section Function 1 Function 2 …… function n

- **The Documentation Section** consists of a set of comment lines giving the name of the program and other details.
- **The Link Section** provides instructions to the compiler to link functions from the system library.
- **The Definition Section** defines all symbolic constants.
- **The Global Declaration Section:** There are some variables and those variables are declared in this section that is outside of all functions.
- **main() function:**
  - Every C program must have one main function section.
  - This section contains two parts, declaration and executable part.
  - **Declaration Part** declares all the variables used in the executable part.
  - There should be at least one statement in the **executable part** which contains instructions to perform certain task.
  - The declaration and executable part must appear between the opening and closing braces.
  - All statements in the declaration part should end with the semicolon.
- The **Subprogram Section** contains all the user defined functions that are called in the main function.
  - **Example**

```
#include<stdio.h>
#include<conio.h>
void main()
{
    clrscr();
    printf("Hello wrold");
    getch();
}
```

- ❖ **Advantages/ Features of C language.**



- ➢ **Simple**
- ➢ **Portability**
- ➢ **Powerful**
- ➢ **Platform dependent**
- ➢ **Structure oriented**
- ➢ **Case sensitive**
- ➢ **Compiler based**
- ➢ **Modularity**
- ➢ **Middle level language**
- ➢ **Syntax based language**
- ➢ **Use of Pointers**
- ➢ **Simple**
  - ▪ Every c program can be written in simple English language so that it is very easy to understand and developed by programmer.
- ➢ **Platform dependent**
  - ▪ A language is said to be platform dependent whenever the program is execute in the same operating system where that was developed and compiled but not run and execute on other operating system.
  - ▪ C is platform dependent programming language.
  - ▪ **Note:** .obj file of C program is platform dependent.
- ➢ **Portability**
  - ▪ It is the idea of moving the instruction from one system to another system.
  - ▪ In C Language **.C** file contain source code, we can edit also this code.
  - ▪ **.exe** file contain application, only we can execute this file.
  - ▪ When we write and compile any C program on window operating system that program easily run on other window based system.

- **Powerful**
  - C is a very powerful programming language, it have a wide verity of data types, functions, control statements, decision making statements, etc.
- **Structure oriented**
  - C is a Structure oriented programming language.
  - Structure oriented programming language designed on simplicity of program
  - reduce the difficulty of code, using this approach code is divided into sub-program/subroutines.
  - These programming have rich control structure.
- **Modularity**
  - It is concept of designing an application in subprogram that is procedure oriented approach.
  - In c programming we can break our code in subprogram.
  - For example we can write a calculator programs in C language with divide our code in subprograms.
  - Example

    ```
    void sum()
    {
     .....
     .....
    }
    void sub()
    {
     .....
     .....
    }
    ```
- **Case sensitive**
  - It is a case sensitive programming language.
  - In C programming 'break and BREAK' both are different.
  - If any language treats lower case latter separately and upper case latter separately than they can be called as **case sensitive** programming language
  - Example:- c, c++, java, .net
- **Middle level language**
  - C programming language can supports two level programming instructions with the combination of low level and high level language that's why it is called middle level programming language.
- **Compiler based**
  - C is a compiler based programming language that means without compilation no C program can be executed.
  - First we need compiler to compile our program and then execute.
- **Syntax based language**
  - C is a strongly tight syntax based programming language.
  - If any language follow rules and regulation very strictly known as strongly tight syntax based language.
  - Example C, C++, Java, .net etc.
- **Efficient use of pointers**
  - Pointers is a variable which hold the address of another variable, pointer directly direct access to memory address of any variable due to this performance of application is improve.

- In C language also concept of pointer are available.

Source Code Vs Object Code

Source Code

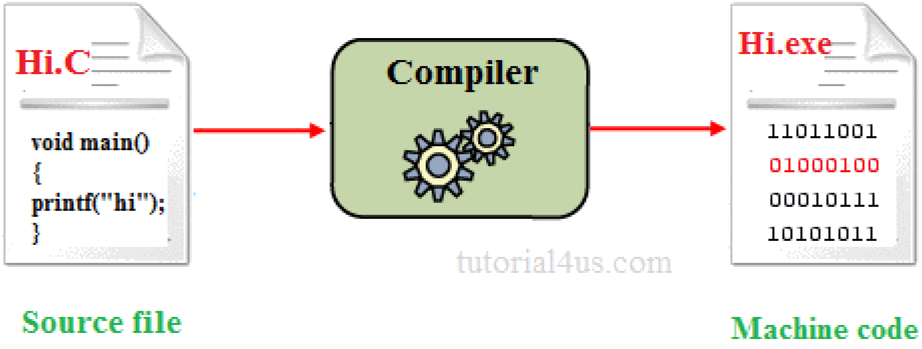
- Source code is in the form of Text form.
- Source code is Human Readable Code.
- Source code is Generated by Human or Programmer.
- Source code is receive Compiler as a Input.

Object Code

- Object Code is in the form of Binary Numbers.
- Object Code is in Machine Readable formats.
- Object Code is Generated by Compiler.
- Object Code is Generated by Compiler as a Output.

Compiler in C

- A compiler is system software which converts programming language code into binary format in single steps.
- In other words Compiler is a system software which can take input from any programming language and convert it into lower level machine dependent language.



```
void main()
{
printf("hi");
}
```

```
11011001
01000100
00010111
10101011
```

Interpreter

- It is system software which is used to convert programming language code into binary format in step by step process.

Assembler

- An assembler is system software which is used to convert the assembly language instruction into binary format in step by step process.

Compiler Vs Interpreter

No	Compiler	Interpreter
1	Compiler takes Entire program as input at a time.	Interpreter takes Single instruction as input at a time.
2	Intermediate Object code is generated	No Intermediate Object code is generated
3	It execute conditional control statements fast.	It execute conditional control statements slower than Compiler
4	More memory is required.	Less memory is required.
5	Program need not to be compiled every time	Every time higher level program is converted into lower level program

6	It display error after entire program is checked	It display error after each instruction interpreted (if any)
7	Example: C	Example: BASIC

Comments in C

- Generally **Comments** are used to provide the description about the Logic written in program. Comments are not display on output screen.
- When we are used the comments, then that specific part will be ignored by compiler.
- In 'C' language two types of comments are possible
  - Single line comments
  - Multiple line comments
- **Single line comments**
  - Single line comments can be provided by using / /...................
- **Multiple line comments**
  - Multiple line comments can be provided by using /*.....................*/
- **Note:** When we are working with the multiple line comments then nested comments are not possible.
- **Example:-**

```
// W.A.P Hello World
#include<stdio.h>
#include<conio.h>
void main()
{
        /* Clear
        Screen */
        clrscr();
        printf("hello world”);
        getch();
}
```

Character set

- Every C program basically consists of keywords, identifiers, constants, operators and some special symbol
  - Letters (all alphabets a to z & A to Z).
  - Digits (all digits 0 to 9).
  - Special characters, (such as colon :, semicolon ;, period ., underscore _, ampersand & etc).
  - White spaces (New line , Tab, Blank space.)

Why C calls middle level language?

- The C support bitwise operator like AND, OR, NOT etc. normally these operations do not supported by higher level languages.
- But commonly supported by machine/assembly level language. so C support some features of low level language and some features of high level language
- Because of this reason C is not considered as fully higher level language. But known as **middle level language**.

- **‘C’ tokens**
  - A smallest individual element of C program is known as **Token**. When we compile the program, the computer recognizes them for building expression and statements.
  - Character’s are categorized as one of six types of **tokens**
    - Identifiers
    - Keywords
    - Constants
    - special characters
    - operators
    - strings

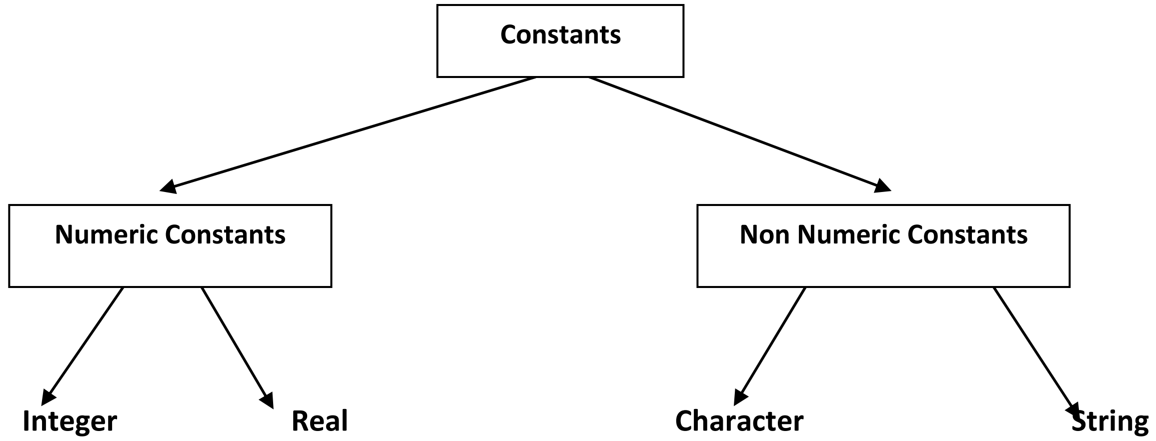
- **Keywords and Identifiers , Constants and Variables**
  - **Keywords**
    - Keywords are those words whose **meaning is fixed** and known by compiler.
    - Keywords are also known as “Reserved words”.
    - The programmer cannot change the meaning of keywords.
    - There are 32 keywords available in C.
    - All the keywords must be written in lowercase.
    - **32 Keywords in C Language**

auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
const	float	short	unsigned
continue	for	signed	void
default	goto	sizeof	volatile
do	if	static	while

  - **Identifiers**
    - In C language identifiers are the names given to variables, constants, functions and user-define data.
    - **Identifier**s are those words which are defined by programmer in program.
    - It consists of letters, digits and underscore.
    - Both uppercase and lowercase letters are allowed in the name of identifier.
    - Rules for an Identifier
      - An Identifier can only have alphanumeric characters (a-z, A-Z , 0-9) and underscore(_).
      - The first character of an identifier can only contain alphabet (a-z , A-Z) or underscore (_).
      - Identifiers are also case sensitive in C. For example *name* and *NAME* are two different identifier in C.
      - Keywords are not allowed to be used as Identifiers.
      - No special characters, such as semicolon, period, whitespaces, slash or comma are permitted to be used in or as Identifier.

  - **What is constant? Explain different types of constant.**
    - It is an identifier whose value can not be changed at the execution time of program.
    - In general **constant** can be used to represent as fixed values in a C program.

- Constants are classified into following types.



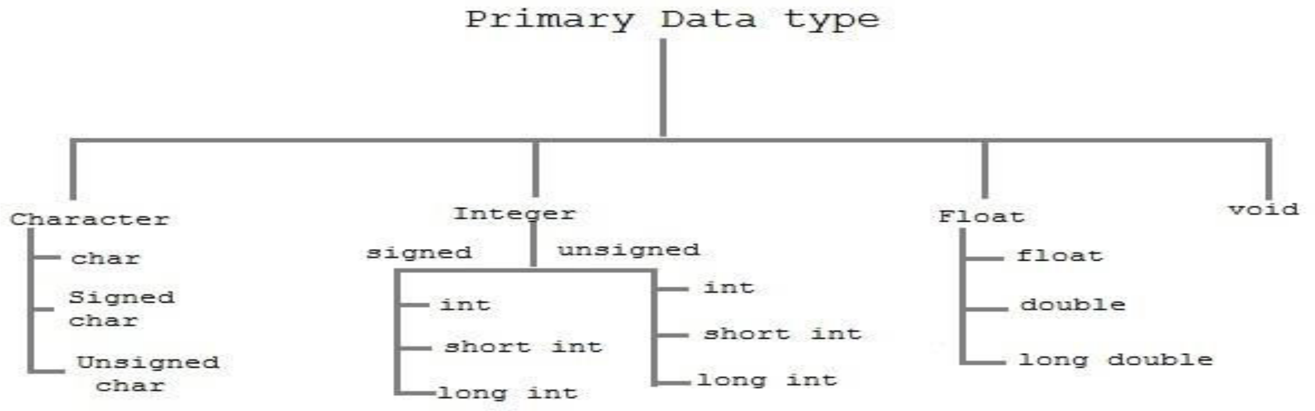
- **Integer constant:**
  - Integer constant is a sequence of digits. It can be positive or negative.
  - A decimal integer constant consists of any combination of digits from 0 to 9.
  - **Example:123**
- **Real constant:**
  - Real constant is a sequence of digits with decimal points.
  - Such a constant are used to represent weight, height etc.
  - **Example:12.34**
- **Character constant:**
  - Character constant is a single character enclosed in a single quotation mark.
  - **Example: ‘a’**
- **String Constant:**
  - String constant is a sequence of character enclosed in a double quotation mark.
  - **Example:”Hello’’**

- **Variable**
  - A variable is a data name that may be used to store a data value.
  - Unlike constants that remain unchanged during the execution of a program, a variable may take different values at Different times during execution.
  - A variable name can be the programmer in a meaningful way so as to reflect its function or nature in the program.
  - **The Six Rules for writing the variable names are given below.**
    1. Variable names consist of letters, digits and underscore.
    2. Variable names must begin with an alphabet or underscore.
    3. Variable names could have length up to 31 characters.
    4. Variable names are case sensitive i.e., "a" and "A" are not equivalent.
    5. Keywords should not be used as, a variable names.
    6. Blank spaces, commas and special symbols are not allowed within variable names.
  - **Variable declaration:-**
    - A variable stores data value of different types and to avoid the confusion to compiler, variables are declared with a data type.
    - It can store only the data values of the type associated with it.
    - **Syntax:** data type variable_name=data_value;

- **Example:** int x=10;
  - "x" is a variable that can store only integer value

Data Types in 'C'

- Data types specify how we enter data into our programs and what type of data we enter.
- C language has some predefined set of data types to handle various kinds of data that we use in our program.
- These datatypes have different storage capacities.
- C language supports 2 different type of data types,
  - **Primary data types**
    - These are fundamental data types in C namely integer(**int**), floating(**float**), character(**char**) and **void**.
  - **Derived data types**
    - Derived data types are like array, function, stucture, union and pointer. These are discussed in detail later.



- **Integer type**
  - Integers are used to store whole numbers.

Type	Size(bytes)	Range
int or signed int	2	-32,768 to 32767
unsigned int	2	0 to 65535
short int or signed short int	1	-128 to 127
unsigned short int	1	0 to 255
long int or signed long int	4	-2,147,483,648 to 2,147,483,647
unsigned long int	4	0 to 4,294,967,295

- **Floating type**
  - Floating types are used to store real numbers.

Type	Size(bytes)	Range
Float	4	3.4E-38 to 3.4E+38
double	8	1.7E-308 to 1.7E+308
long double	10	3.4E-4932 to 1.1E+4932

- **Character type**
    - Character types are used to store characters value.

Type	Size(bytes)	Range
char or signed char	1	-128 to 127
unsigned char	1	0 to 255

- **void type**
    - void type means no value. This is usually used to specify the type of functions.

How to declare a variable?(Declaration)

- Variable must be declared before using in it program.
- Declaration of variable begins with its datatype
- **Syntax:**
    - **d**atatype var1, var2…var n;
- here datatype could be any of int,float,char,double;
- All the variable names must be separated with comma (,).
- **Example:**

```
int sum, total;
float area;
char ch;
```

How to assigning values to variables?(Initialization)

- Once the variable is declared, it can be assigned a value in the program.
- Assignment operator (=) is used to assign value to a variable.
- **Syntax:**
    - datatype variable name=value;
    - Variable name=value;
- **Example:**

```
int n1=10;
int n2;
n2=20;
```

Dynamic initialization

- Assign value to a **variable at a time of declaration** is called dynamic initialization
- Syntax:- Datatype Variable_name=value;
- Example:-int a=10;

Type modifiers

- **C programming have several types of modifiers**
    - **signed**:-
        - The **signed** type modifier specifies that a **char** or **int** variable stores both positive and negative values
        - Example:- signed int a=+10
    - **unsigned:-**
        - The **unsigned** type modifier specifies that a **char** or **int** variable stores only positive value
        - **Example:-** unsigned int a=10;

- **short:-**
  - The **short** type modifier specifies that an **int** variable uses less-than or the same amount of storage as an **int** type
  - **Example:-** short int a;
- **long:-**
  - The **long** type modifier specifies that an **int** variable uses more-than or the same amount of storage as an **int** type
  - **Example:-** long int a;

❖ Type conversion

- A type cast is basically a conversion from one type to another. There are two types of type conversion:
  - **Implicit Type Conversion or Automatic conversion**
    - Done by the compiler on its own, without any external trigger from the user.
    - Generally takes place when in an expression more than one data type is present. In such condition type conversion (type promotion) takes place to avoid lose of data.
    - All the data types of the variables are upgraded to the data type of the variable with largest data type.
      - **bool -> char -> short int -> int ->**
      - **unsigned int -> long -> unsigned ->**
      - **long long -> float -> double -> long double**
    - It is possible for implicit conversions to lose information, signs can be lost (when signed is implicitly converted to unsigned), and overflow can occur (when long long is implicitly converted to float).
    - **Example of Type Implicit Conversion:**

```
// An example of implicit conversion
#include<stdio.h>
void  main()
{
   int a = 10;    // integer a
   // a is implicitly converted to float
   float b = a + 1.0;
    printf("a = %d, b = %f", a, b);
}
```

  - **Explicit Type Conversion or type casting**
    - This process is also called type casting and it is user defined.
    - Here the user can type cast the result to make it of a particular data type.
    - The syntax in C: (type) expression
    - Type indicated the data type to which the final result is converted.

```
// C program to demonstrate explicit type casting
#include<stdio.h>
 void main()
{
            double a= 1.2;
            // Explicit conversion from double to int
            int sum = (int)a + 1;
            printf("sum = %d", sum);
 }
```

Constant Variable

- constant variable means value of a variable does not change during execution of a program means, it remain constant
- In c constant variable is declared using const keyword
- Syntax:- const datatype variable_name=value;
- **Example:- const int a=10;**

```
#include<stdio.h>
 void main()
{
            const int a = 12;
            printf("%d",a);
 }
```

volatile variable

- volatile variable value me be change outside of the executing software
- syntax:- volatile datatype variable_name=value;
- **Example:- volatile int a=10;**

```
#include<stdio.h>
 void main()
{
            volatile int a = 12;
            printf("%d",a);
 }
```

Input and Output statements in ‘C’

- **printf()**
  - This function is used for displaying the output on the screen i.e the data is moved from the computer memory to the output device.
  - **Syntax:** printf(“format string”, arg1, arg2, …..);
    - In the above syntax, 'format string' will contain the information that is formatted.
    - They are the general characters which will be displayed as they are.arg1, arg2 are the output data items.
  - Example: printf(“Enter a value:”);
  - printf will generally look at from left to right of the string.
  - The characters are displayed on the screen in the manner they are encountered until it comes across % or \.

- **scanf()**
  - scanf is used when we enter data by using an input device.
  - Syntax: scanf (“format string”, &arg1, &arg2, …..);
    - The numbers of items which are successful are returned.
    - Format string consists of the conversion specifier.

    - Arguments can be variables or array name and represent the address of the variable.
    - Each variable must be preceded by an ampersand (&).
    - Array names should never begin with an ampersand.
  - Example:
    - int avg;
    - scanf(“%d ”,&avge):
  - scanf works totally opposite to printf.
  - The input is read, interpret using the conversion specifier and stores it in the given variable.
  - The conversion specifier for scanf is the same as printf.
  - scanf reads the characters from the input as long as the characters match or it will terminate.
  - The order of the characters that are entered are not important.

- **Example:-**

```
#include<stdio.h>
void main()
{
        int a;
        printf("Enter a value");
        scanf("%d",&a);
        printf( "display %d",a);
}
```

- **getchar() & putchar() functions**
  - The getchar() function reads a character from the terminal and returns it as an integer.
  - This function reads only single character at a time. You can use this method in the loop in case you want to read more than one characters.
  - The putchar() function prints the character passed to it on the screen and returns the same character. This function puts only single character at a time. In case you want to display more than one characters, use putchar() method in the loop.
  - Example:-

```
#include <stdio.h>
#include <conio.h>
void main( )
{
        int c;
        printf("Enter a character");
        c=getchar();
        putchar(c);
        getch();
}
```

- **gets() & puts() functions**
  - The gets() function reads a line from stdin into the buffer pointed to by s until either a terminating newline or EOF (end of file) occurs.
  - The puts() function writes the string s and a trailing newline to stdout.
  - Example:-

```
#include<stdio.h>
```

```
#include<conio.h>
void main()
{
     char a[100];
     printf("Enter a string");
     gets(a);
     puts(a);
     getch();
}
```

❖ Write, compile, execute a simple 'C' program

➢ First Program

- Programming in C language is very simple and it is easy to learn here i will show you how to write your first program.
- For writing C program you need Turbo C Editor.
- First you open TC and write code.
- **Example**

```
#include<stdio.h>
#include<conio.h>
void main()
{
     clrscr();
     printf("This is my first program");
     getch();
}
```

- **Output**
  - This is my first program

- After writing complete code save the program using **F2**
- Save your code with **.c** extension for example: hello.c
- After save the code you need to compile your code using **alt+f9**
- Finally Run the program using **clt+f9**

➢ Save C program

- Save any C program using **.c** Extension with file name. For example your program name is sum, then it save with **sum.c**.
- **Syntax:-** filename.c

➢ Compile and Run C Program

- for compile any C program you just press **alt+f9** , after compilation of your c program you press **clt+f9** for run your C program.
- **Syntax**
  - Compile -> alt+f9
  - Run -> clt+f9

- **Difference between Local variable and Global variable**
  - In C language, a variable can be either of global or local scope.
  - Global variable
    - Global variables are defined outside of all the functions, generally on top of the program.
    - The global variables will hold their value throughout the life-time of your program.
  - Local variable
    - A local variable is declared within the body of a function or a block.
    - Local variable only use within the function or block where it is declare.

- **What is backslash character constant? OR What is escape sequence character? Write and Explain with example.**
  - The combination of backslash and a character is known as escape sequence.
  - The backslash character constant is used to print non-printable characters. They are used in output.

Constant	Meaning
‘\a’	Alert(bell)
‘\b’	Back space
‘\f’	Form feed
‘\n’	New line
‘\r’	Carriage return
‘\t’	Horizontal tab
‘\v’	Vertical tab
‘\’’	Single quote
‘\’’’	Double quote
‘\?’	Question mark
‘\\’	Backslash
‘\0’	Null

- **Explain Enumerated data type with example?**
  - Enumerated data type is user defined data type.
  - Using enumerated data type we can create more than one symbolic constant at a time.
  - The general syntax for defining enumerated data type is follow:
  - **Syntax:**
    - enum identifier {value1, value2,value3};
  - Here, enum is a keyword to declare enumerated data type.
  - Identifier is a user defined enumerated data type.

- **Example:**
  - enum month {jan,feb,mar,apr,may,jun,jul,aug,sep,oct,nov,dec};
    - Here enum is keyword. month is datatype
    - The compiler automatically assigns integer digits to values starting with 0.
    - so, here jan=0,feb=1,mar=2,apr=3,…………….and dec=11.
  - enum month{jan=1,feb=2,mar=3,apr=4,may=5,jun=6,jul=7,aug=8, sep=9, oct=10, nov=11, dec=12};
    - here jan=1,feb=2,mar=3,apr=4,…………….and dec=12.
  - enum flag{false, true};
    - here false=0,true=1

- **Explain the storage class of data type?**
  - The storage class of a variable determines the scope and lifetime of a variable.
  - The variable can be stored in computer memory or in register of CPU
  - There are four storage classes:
    - **Auto OR local variable**
    - **External OR Global variable**
    - **Static**
    - **Register**
  - **Storage class** of a variable decides where the variable going to stored (in CPU or memory of computer).
    - **Syntax:-**
      - storage_specifier data_type variable_name;
      - storage_specifier data_type variable_name=value;
  - **Auto or local variable:**
    - The variable declared as auto is only visible to the function in which it is declared. It is also known as local variable.
    - Default storage class for any variable is auto.
    - **Example**: int a; or **auto** int a;
  - **Extern or Global variable:**
    - The variable is declared as extern is visible to all the function the file.
    - It is also known as global variable.
    - **Example**: extern int a;
  - **Static:**
    - The variable declared as static is only visible to the function in which it is declared.
    - But it retains its value even after the controls transferred to the calling function.
    - It is also knows as local variable but it contains life time of a global variable.
    - **Example**: static int a;
  - **Register:**
    - The variable declared as register is only visible to the function in which it is declared .
    - But the value of the variable is stored in the register to it can be accessed faster as compared to other variable.
    - **Example**: register int a;