

Course Problem Solving and Programming

Course Teaching Scheme:

Course Code:ES1034

Credits:5

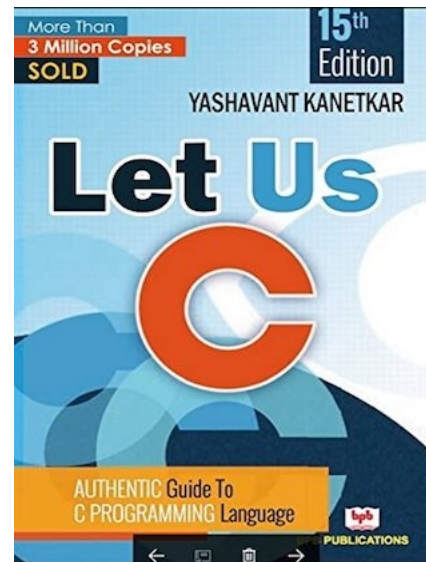
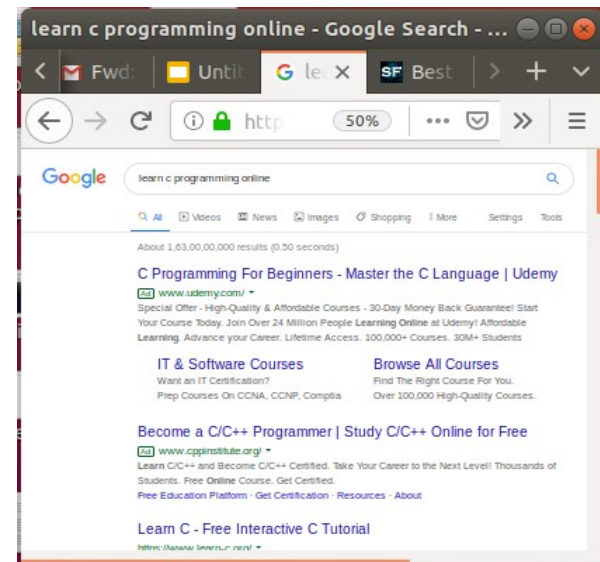
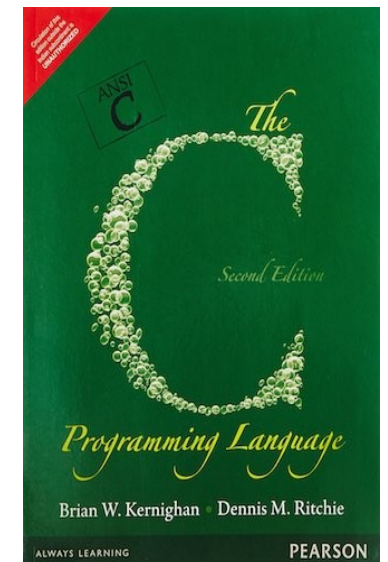
Teaching Scheme:Theory: 3 Hours/Week

Lab:2 Hours/Week

Tutorial:1 Hour / Week

For the first time learn C Programming from,

1. “Let Us C” by Yashavant Kanetkar
2. “The C Programming Language” by Brian W Kernighan / Dennis Ritchie
3. Over the web



Section 1:

1. **Computing Fundamentals**
2. **Overview of C**
3. **C variables, Constants and Data Types**
4. **Operators and Expressions in C**
5. **C Control Statements**

Section 2:

1. **Arrays**
2. **Strings**
3. **C Functions**
4. **Pointers in C**

How a data value is represented using binary symbols

- Representation of Information inside a Computer use Binary number system to represent all types of information inside the memory.
- Binary number system is suitable for this purpose due to following reasons:
 - • Electronic components in digital computers operate in binary mode. A switch is either on (1) or off (0); a transistor is either conducting (1) or non-conducting(0).
 - • Computers have to handle only two digits (bits) rather than 10. So binary system simplifies design, reduce the cost and improve the reliability of the computer.
 - • Everything that can be done with decimal system can also be done using a binary system

Introduction to numbering systems:

- The technique to represent and work with numbers is called **number system**. **Decimal number system** is the most common number system. Other popular number systems include
 - **binary number system,**
 - **octal number system,**
 - **hexadecimal number system, etc.**

Decimal Number System

- Decimal number system is a **base 10** number system having 10 symbols from 0 to 9. This means that any numerical quantity can be represented using these 10 digits.
- Decimal number system is also a **positional value system**. This means that the value of digits will depend on its position.
- Ex.
- In 734, value of 7 is 7 hundreds or 700 or 7×100 or 7×10^2
- In 971, value of 7 is 7 tens or 70 or 7×10 or 7×10^1
- In 207, value of 7 is 7 units or 7 or 7×1 or 7×10^0
- The weightage of each position can be represented as follows –

10^5	10^4	10^3	10^2	10^1	10^0
--------	--------	--------	--------	--------	--------

Binary Number System

- The easiest way to vary instructions through electric signals is two-state system – on and off. On is represented as 1 and off as 0, though 0 is not actually no signal but signal at a lower voltage. The number system having just these two symbols – 0 and 1 – is called **binary number system**.
- Each binary digit is also called a **bit**. Binary number system is also positional value system, where each digit has a value expressed in powers of 2, as displayed here.

2^5	2^4	2^3	2^2	2^1	2^0
-------	-------	-------	-------	-------	-------



In any binary number, the rightmost digit is called **least significant bit (LSB)** and leftmost digit is called **most significant bit (MSB)**.

And decimal equivalent of this number is sum of product of each digit with its positional value.

$$\begin{aligned} 11010_2 &= 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 \\ &= 16 + 8 + 0 + 2 + 0 \\ &= 26_{10} \end{aligned}$$

Octal Number System

- **Octal number system** has eight digits – 0, 1, 2, 3, 4, 5, 6 and 7. Octal number system is also a positional value system with where each digit has its value expressed in powers of 8, as shown here –

8^5	8^4	8^3	8^2	8^1	8^0
-------	-------	-------	-------	-------	-------

Decimal equivalent of any octal number is sum of product of each digit with its positional value.

$$726_8 = 7 \times 8^2 + 2 \times 8^1 + 6 \times 8^0$$

$$= 448 + 16 + 6$$

$$= 470_{10}$$

Hexadecimal Number System

- In this system, 16 digits used to represent a given number. Thus it is also known as the base 16 number system. Each digit position represents a power of 16. As the base is greater than 10, the number system is supplemented by letters. Following are the hexadecimal symbols: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

16^5	16^4	16^3	16^2	16^1	16^0
--------	--------	--------	--------	--------	--------

- Ex.
- $27FB_{16} = 2 \times 16^3 + 7 \times 16^2 + 15 \times 16^1 + 10 \times 16^0$
- $= 8192 + 1792 + 240 + 10$
- $= 10234_{10}$

Introduction to computing jargons: Nibble, Bit, Byte, Word, double word, KB, MB, GB etc

- | UNIT | ABBREVIATION | STORAGE |
|------------|--------------|-----------------------------|
| Bit | B | Binary Digit, Single 1 or 0 |
| Nibble | - | 4 bits |
| Byte/Octet | B | 8 bits |
| Kilobyte | KB | 1024 bytes |
| Megabyte | MB | 1024 KB |
| Gigabyte | GB | 1024 MB |
| Terabyte | TB | 1024 GB |
| Petabyte | PB | 1024 TB |
| Exabyte | EB | 1024 PB |
| Zettabyte | ZB | 1024 EB |
| Yottabyte | YB | 1024 ZB |

Introduction to C programming,

- C was originally developed in the 1970s, by Dennis Ritchie at Bell Telephone Laboratories, Inc.
- C is a High level , general –purpose structured programming language. Instructions of C consists of terms that are very closely same to algebraic expressions, consisting of certain English keywords such as if, else, for ,do and while
- C contains certain additional features that allows it to be used at a lower level , acting as bridge between machine language and the high level languages.
- This allows C to be used for system programming as well as for applications programming

Introduction to Compiler, Interpreter, Loader, linker, debugger

- **Compiler:-**

- A Compiler is a program that converts a number of statement of program into binary language, but it is more intelligent than interpreter because it goes through the entire code at once and can tell the possible errors and limits and ranges. But this makes it's operating time a little slower.it is platform-dependent.its help to detect error and get displayed after reading the entire code by compiler.
- Compilers turns the high level language to binary language or machine code at only time once.
- Ex. C,C++

- **Interpreter:-**An interpreter is also a program like a compiler that converts assembly language into binary but an interpreter goes through one line of code at a time and executes it and then goes on to the next line of the code and then the next and keeps going on until there is an error in the line or the code has completed. It is 5 to 25 times faster than a compiler but it stops at the line where error occurs and then again if the next line has an error too, where as a compiler gives all the errors in the code at once. ex. Java

Introduction to Compiler, Interpreter, Loader, linker, debugger

- **Loader :-**

- Loader is a part of operating system and is responsible for loading executable files into memory and execute them.
- It calculates the size of a program (instructions and data) and create memory space for it. It initializes various registers to initiate execution.

- **Linker :-**

- For a code to run we need to include a header file or a file saved from the library which are pre-defined if they are not included in the beginning of the program then after execution the compiler will generate errors, and the code will not work.
- Linker is a program that holds one or more object files which is created by compiler, combines them into one executable file. Linking is implemented at both time, load time and compile time. Compile time is when high level language is turns to machine code and load time is when the code is loaded into the memory by loader.

Introduction to Compiler, Interpreter, Loader, linker, debugger

- Debugger:-A **debugger** or **debugging tool** is a **computer program** used to **test** and **debug** other programs (the "target" program). The main use of a debugger is to run the target program under controlled conditions that permit the programmer to track its operations in progress and monitor changes in computer resources that may indicate malfunctioning code. Typical debugging facilities include the ability to run or halt the target program at specific points, display the contents of memory, CPU registers or storage devices and modify memory or register contents in order to enter selected test data that might be a cause of faulty program execution.

Indentations and comments

- **Indentation** improves the readability of the code. It is mainly used for code inside looping statements, control structures, functions etc. as good intended code is easy to maintain and is good looking. It makes the code more readable and easy to understand. Some programming languages like [Python](#) made indentation mandatory instead of using brackets, It makes code easy to read and understand.
- **Writing Comments in C**
 - `/* This is an example of a comment in C */`
 - Comments in the C language begin with `/*` and end with `*/` . Comments in C cannot be nested,

The Character set of 'C'

- C language consist of some characters set, numbers and some special symbols. The character set of C consist of all the alphabets of English language. C consist of
- Alphabets a to z, A to Z
- Numeric 0,1 to 9
- Special Symbols {,},[,],? ,+,-, *,/,%,!,;,and more
- The words formed from the character set are building blocks of C and are sometimes known as tokens. These tokens represent the individual entity of language. The
- following different types of token are used in C
- 1) Identifiers 2)Keywords 3)Constants
- 4) Operators 5)Punctuation Symbols

Identifiers

- A 'C' program consist of two types of elements , user defined and system defined. Identifiers is nothing but a name given to these elements.
- An identifier is a word used by a programmer to name a variable , function, or label.
- identifiers consist of letters and digits, in any order, except that the first character or label.
- Identifiers consist of letters and digits if any order,except that the first character must be letter.
- Both Upper and lowercase letters can be used

The rules for naming identifiers are as follows

- Identifier names are unique.
- Cannot use a keyword as identifiers.
- Identifier has to begin with a letter or underscore (_).
- It should not contain white space.
- Special characters are not allowed.
- Identifiers can consist of only letters, digits, or underscore.
- Only 31 characters are significant.
- They are case sensitive.

Keywords

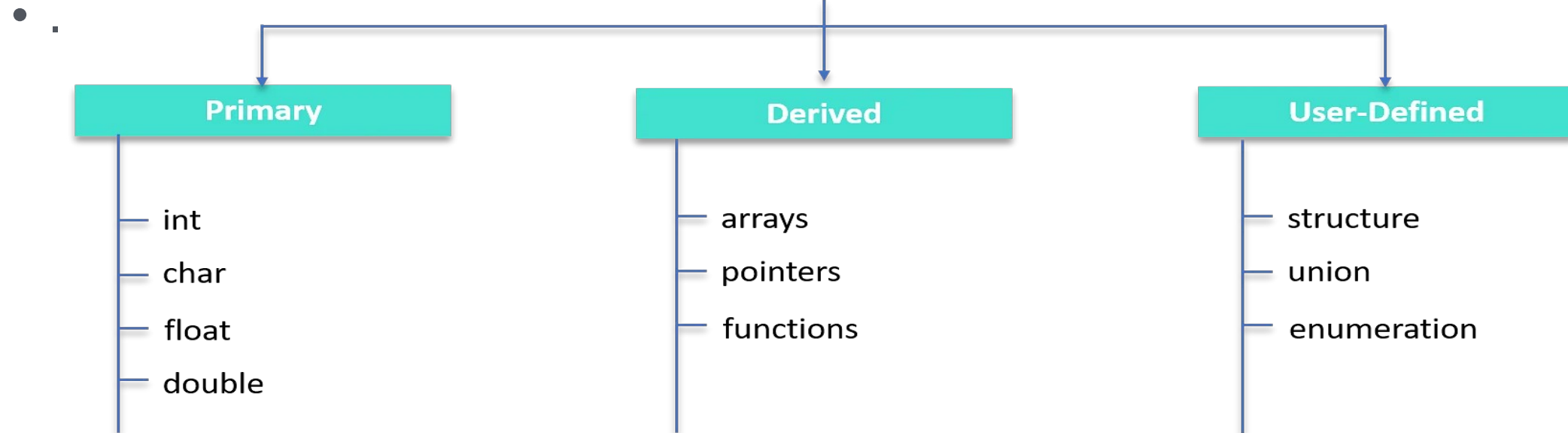
- Keywords are nothing but system defined identifiers.
- Keywords are reserved words of the language.
- They have specific meaning in the language and cannot be used by the programmer as variable or constant names
- C is case sensitive, it means these must be used as it is
- 32 Keywords in C Programming

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

Data types and modifiers in C

- Data is the primary need for any programming language. This data can be of various types, including integers, fractional numbers, characters, and symbols, or a group of entities. To handle these different types of data, the concept of data types has been introduced in C. **Data types** in C, signify the characteristics of the data stored in a variable. Every type of data has its own data type that cannot be used to store other types of data

DATA TYPES IN C



Primary Data Types in C

- Primary data types are also known as the fundamental data types because they are pre-defined or they already exist in the C language. All the other types of data types (derived and user-defined data types) are derived from these data types. Primary data types in C are of 4 types: int, char, float, and double. In this section, we are going to discuss all these data types in detail. The following table represents the memory consumed or size of each primary

DATA TYPE	MEMORY (bytes)	RANGE	FORMAT SPECIFIER
int	2 or 4 depends upon compiler	-32,768 to +32,767	%d
char	1	-128 to 127	%c
float	4	1.2E-38 to 3.4E+38	%f
double	8	2.3E-308 to 1.7E+308	%lf

int Data Type

- The int data type is used to store the integer values. Any number that falls in the range of $-\infty$ to $+\infty$ is said to be an integer. For example 0, 10, 105, -15, -103, etc are all integers. However, integers can be either negative or positive. And that's where we use data type modifiers, which we will be going to discuss later. The size of int is compiler dependent.
- In C language, the integers cannot be represented as floating or decimal point numbers. They can only be operated by/on integers themselves. For example, 4/3 is fine but 4/3.0 is not fi
- Syntax
- `int variable_name;`

Ex.`int age;`

`int marks1,marks2,marks3;`

`int roll_no;`

char Data Type

- The char data type is used to store the characters. The characters stored in a variable of char data type have a value equivalent to its integer code. These codes are often called American Standard Code For Information Interchange or ASCII codes. The ASCII code for upper case characters A to Z starts from 65 to 90 and for lower case characters a to z, it starts from 97 to 122.
- The general size of a character is 1 byte. However, just like the integers the size of a char can vary according to the data type modifiers.

- The float data type is used to store the floating-point numbers. The numbers that have a fractional part are called floating-point numbers. For example, 3.0, 5.57, -31.2, -3.12, etc are all floating-point numbers. Moreover, it should be noticed carefully that 8 is an integer but 8.0 is a floating-point number.
- The major advantage of floating point numbers over integers is that they can represent a much larger and wider range of digits as compared to int data type.
- Syntax
- `float variable_name;`
ex. `float radius, area;`
`float percentage;`

- The double data type is also used to store floating-point numbers. But still, it is considered as a distinct data type because of two reasons. The first one is that it occupies twice as much memory as type float. The second reason is that it is used to store a much larger range of floating-point numbers than a float data type.
- The double stores almost double the number of significant digits after the decimal point or precision of the float.
- Syntax
- `double variable_name;`

Ex.

```
double volume, area;
```

Derived Data Types in C

- Derived data types are derived from the primitive or fundamental data types. There are mainly 3 types of derived data types in C that we are going to discuss in this section.
- Arrays
- An array is a group of similar kinds of finite entities of the same type. These entities or elements can be referred to by their indices respectively. The indexing starts from 0 to (array_size-1) conventionally. An array can be one-dimensional, two-dimensional, or multidimensional.
- Syntax
- `data_type arr_name[size];`

Ex. `int student[60];`

`int branch[7];`

- Functions**

- A function is a piece of code that performs some specific task when invoked in the program. It can be called from anywhere and any number of times in the program. The return value i.e., what type of value it will return depends upon the return type of the function.

- Syntax**

- `return_type function_name(parameters);`

- `return_type`: This is the data type that specifies the type of value to be returned by the function. If the return type is void, then it is not mandatory for the function to return a value.

- `function_name`: This is the name of the function. To specify the name of a function, you must follow the same rules which are applicable while declaring a usual variable in C.

- `parameters`: The parameters are optional. They are passed according to the type of the function call.

ex.

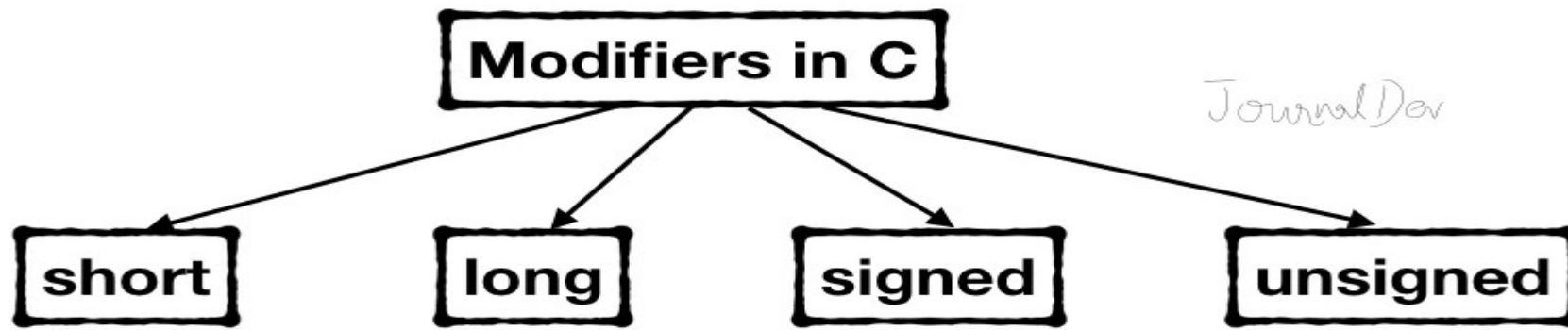
```
void addition( );  
int area( int r);
```

Derived Data Types in C

- **Pointers**
- A pointer can be defined as a variable that stores the address of other variables. This address signifies where that variable is located in the memory. If a is storing the address of b, then a is pointing to b. The data type of a pointer must be the same as the variable whose address it is storing.
- **Syntax**
- `type *pointer_name;`
- **type:** This is the data type that specifies the type of value to which the pointer is pointing.
- **pointer_name:** This is the name of the pointer. To specify the name of a pointer, you must follow the same rules which are applicable while declaring a usual variable in C. Apart from these rules, a pointer must always be preceded by an asterisk(*).
- Ex. `int *p;`
- `float * f;`

Data types and Modifiers in C

- These are keywords in C to modify the default properties of int and char data types. There are 4 modifiers in C as follows.
- Type Modifiers are special keywords defined in the programming language which are used to modify the default properties of the Built-in Data types.
- Type Modifier keywords are used as prefixes to the Built-in Data types in the declaration or initialization statements. They are used to modify the memory allocation of the variable. Type Modifiers also modify (increase or decrease) the range of the Data type.



Data types and Modifiers in C

- In C and C++, there are 4 Type Modifiers namely –

1.Signed

2.Unsigned

3.Short

4.Long

1.Short Type Modifier –

- This type modifier is used to save memory space. We use short for small integers. There does not exist any short floating type. Short Type Modifier is only used on the int data type.
- Example –
- *short int a = 21324;*

Data types and Modifiers in C

2 Long Type Modifier -

- It is used to increase the current size of the data type. This type modifier is used to store large numbers. The type long float is not a legal type. Type Modifier is used on int and double data type.
- Example -
- *long int x = 1234567 ;*
- *long double c = 10.2345 ;*

3. Signed Type Modifier -

- This special keyword is used for holding both positive as well as negative numbers. This type modifier is the default modifier for every data type.
- Example
- signed int x=200;
- signed int z=-100; //it occupies 16 bits in memory and MSB bit will be sign bit
- signed char='A';

Data types and Modifiers in C

4. Unsigned Type Modifier -

- This type modifier is used to hold only non-negative i.e. only positive values in the variable.
- Example -
- `unsigned int a=35 ; //correct declaration`
- `unsigned int b=-34; // wrong declaration`
- `unsigned char s='A' ; //all 8 bits for occupies for memoeey storage no sign bit`

Type Modifiers and their Ranges

Type	Bytes	Range
signed char	1	-128 to +127
unsigned char	1	0 to 255
int	2	-32,768 to +32,767
signed int	2	Same as int
unsigned int	2	0 to 65,535
signed short int	2	Same as int
unsigned short int	2	0 to 65,535
long	4	-2,147,438,648 to +2,147,438,647
signed long	4	Same as long
unsigned long	4	0 to 4,294,967,295
float	4	-3.4e38 to +3.4e38
double	8	-1.7e308 to +1.7e308
long double	10	-1.7e4932 to +1.7e4932

Variable declaration

- A variable is nothing but a name given to a storage area that our programs can manipulate. Each variable in C has a specific type, which determines the size and layout of the variable's memory; the range of values that can be stored within that memory; and the set of operations that can be applied to the variable. After operations applied value stored will be vary so the name variable.

Constants

- A constant is a value or an identifier whose value cannot be altered in a program. For example: 1, 2.5,
- As mentioned, an identifier also can be defined as a constant. eg. `const double PI = 3.14`
- Here, PI is a constant. Basically what it means is that, PI and 3.14 is same for this program.

Integer constants

- A integer constant is a numeric constant (associated with number) without any fractional or exponential part. There are three types of integer constants in C programming:
 - decimal constant(base 10)
 - octal constant(base 8)
 - hexadecimal constant(base 16)

Constant

Floating-point constants

- A floating point constant is a numeric constant that has either a fractional form or an exponent form. For example: 2.0, 0.0000234, -0.22E-5

Character constants

- A character constant is a constant which uses single quotation around characters. For example: 'a', 'l', 'm', 'F'

String constants

- String constants are the constants which are enclosed in a pair of double-quote marks. For example: "good" , "x", "Earth is round\n"