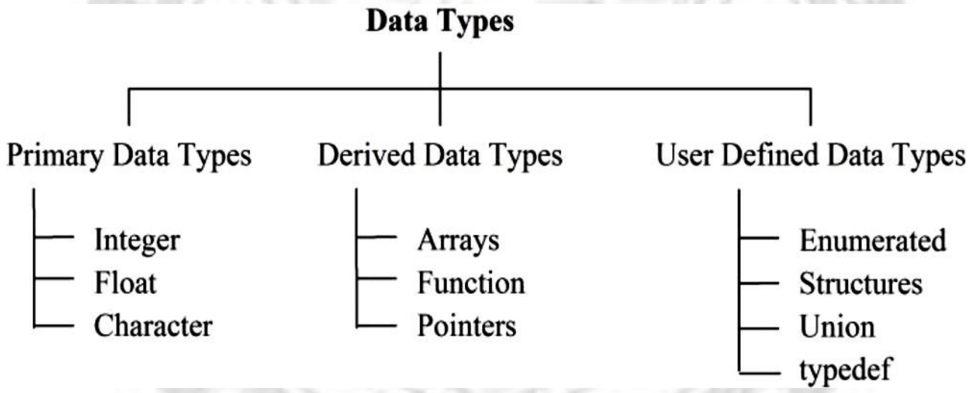
Data Types

- The C language supports different types of data. Each data may be represented differently within the computer memory. Typical memory requirements for each data will determine the possible range of values for that data type.
- The varieties of data types available allow the programmer to select the type appropriate to the needs of the application as well as the machine.

C supports three categories of data types:

1. Primary data type 2. Derived data type 3. User defined data type.



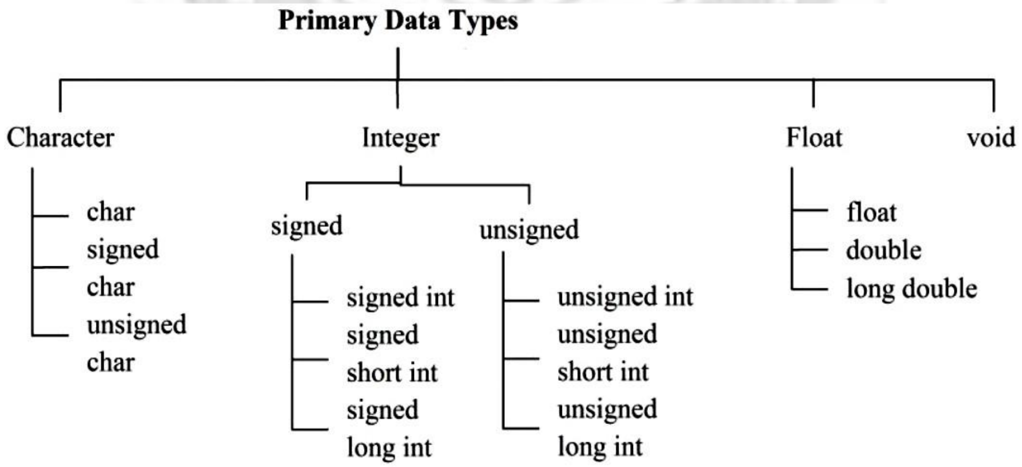


Fig. 1.5 Classification of 'C' Data types

Primary Data Types

- C compiler supports the following four Fundamental/ Primary/ Primitive/ Basic/ Built-in data types:

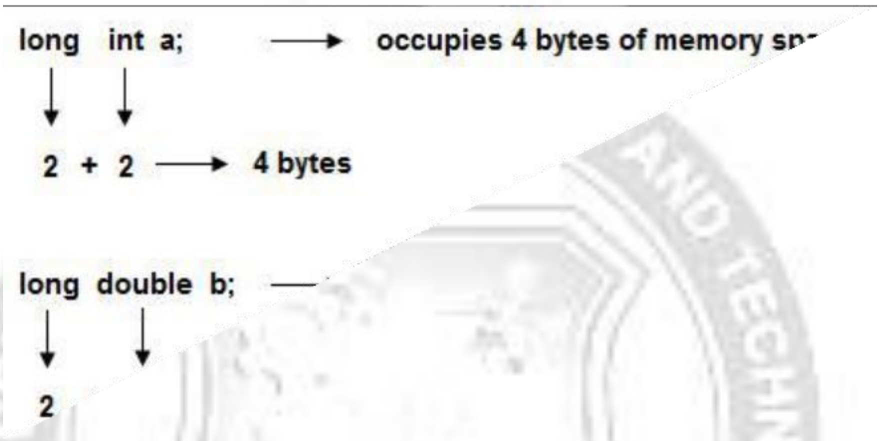
- **Character**: **Character data type** is used to store a character. A variable of character data type allocated only one byte of memory and can store only one character. Keyword char is used to declare variables of type character. The range of character (char) data type is -128 to 127. For Example: *char* ch = 'A';
- **Integer**: **Integer data type** is used to store a value of numeric type. Keyword *int* is used to declare variables of integer type. The memory size of a variable of integer data type is dependent on the operating system. For example the size of integer data type in a 32 bit computer is 4 bytes whereas size of integer data type in 16 bit computer is 2 bytes. For Example: *int* count = 10;
- **Float**: **Floating point data type** is used to store a value of decimal values. The memory size of a variable of floating point data type is dependent on the Operating System. Keyword *float* is used to declare variables of floating data type. For example the size of a floating point data type in a 16 bit computer is 4 bytes. For Example: *float* rate = 5.6;
- **Double**: **Double data type** is similar to floating data type except that it provides up to ten digits of precision and occupies eight bytes of memory. For Example: *double* d = 11676.2435676542;
- **Void Data Type:** *Void* is an empty data type that has no value. This can be used in functions and pointers.

Type modifiers in C

- In c language Data Type Modifiers are keywords used to change the current properties of data type. Data type modifiers are classified into the following types.
  - Long
  - Short
  - Unsigned
  - signed
- Modifiers are prefixed with basic data types to modify (either increase or decrease) the amount of storage space allocated to a variable. For example, storage space for int data type is 4 bytes for a 32 bit processor. We can increase the range by using long int which is 8 bytes. We can decrease the range by using short int which is 2 bytes.

long

- This can be used to increase the size of the current data type by 2 more bytes, which can be applied on int or double data types. For example int occupy 2 bytes of memory; if we use long with integer variable, then it occupies 4 bytes of memory.



Syntax

long a; —> **by default** which represent **long int**

short

- In general int data type occupies different memory spaces for a different operating systems; to allocated fixed memory space a short keyword can be used.

Syntax

short int a; —> occupies 2 bytes of memory space in every operating system

unsigned

- This keyword can be used to make the accepting values of a data type of positive data type.

Syntax

Signed

unsigned int a = 100; // right

unsigned int a = -100; // wrong

- This keyword accepts both negative and positive values and this is the default property or data type modifier for every data type.

  int a = 10; // right

  int a = -10; // right

  signed int a = 10; // right

  signed int a = -10; // right

Type	Size (bytes)	Range
int or signed int	2	-32,768 to 32,767
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

Table 1.5 Size and range of Integer type on 16-bit machine

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

Table 1.6 Size and range of Float type on 16-bit machine

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

Table 1.7 Size and range of Character type on 16-bit machine

Derived data types

- Derived data types are derived from the collection of primary data types. C supports the following derived data types.
- **Arrays**
    - Array is a collection of variables of same data type that are referenced by a common name.

      Syntax: <datatype> <variable name> [Index];

      Example: int a[10];
- **Functions**
    - A function is a self-contained program segment (block of statements) that carries out some specific, well defined task.

      Syntax for function prototype

      <return datatype> function name (forma) arg, formal arg2 … formal arg n);

      Syntax for function definition

      <return type> function name (parameter list)

      parameter declarations

      {

      body of the function;

      return (expression);

      }

Example

```
void swap (int x, int y)
{
int z;
z = x;
x = y;
y = z;
}
```

- **Pointers**
  - Pointer is a variable which stores the address of another variable.

  Example

  *int *p; // declaration of pointer*

  int x; // declaration of variable

  p=&x; // pointer variable stores the address of x variable.

  x=5 ; // x variable assigned with value 5.

User defined data types

- The user defined data types enable a program to invent his own data types and define what values it can taken on. Thus these data types can help a programmer to reducing programming errors.
- C supports the following user defined data types.
  - **Structures**

    A structure is a single entity representing a collection of data items of different data types.

    Example

    ```
    struct student
    {
    int  roll_no;
    char fname[25];
    char branch[15];
    int marks;
    }s1;
    ```
  - **Unions**

    A union is a data type in 'c' which allows overlay of more than one variable in the same memory area.

    Example

    ```
    union emp
    {
    ```

char name[20];

char eno[10];

}

union emp e1;

- **Enumerated data type**

  A enumerated data type is a set of values represented by identifiers called *enumeration constants***.** It is a user-defined data type and the general format of this data type is

  enum name {number 1, number 2, ... number n};

  In above format enum is a keyword, name is given by the programmer by the identifier rules. number 1, number 2, ... number n are the member of enumerated datatype.

- **Type definition**

  "type definition" that allows user to define an identifier that would represent a data type using an existing data type.

  Syntax:

  typedef type identifier;

  typedef <existing_data_type> <new_user_define_data_type>;

  Example

  typedef int number;

  typedef long big_ number;

  typedef float decimal;

  number visitors = 25;

  big_number population = 12500000;

  decimal radius = 3.5;

Constants

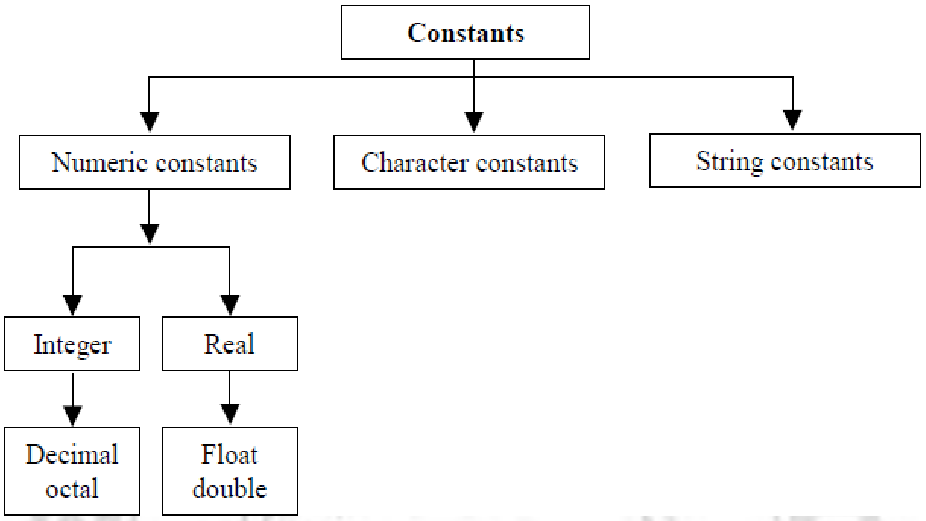


Fig: 1.6 Classification of 'C' Constants

- *Constants* are fixed values and they remain unchanged during the execution of the program. The constants are classified as follows:

Integer constants

- *It* consist of a sequence of digits without any decimal point. Integer constant can be written in three different number systems: decimal, octal and hexadecimal. A *decimal integer constant* can consist of any combination of digits taken from the set 0 through 9.
  1. Decimal number – 0 to 9
  2. Octal number – 0 to 7
  3. Hexadecimal number – 0 to 9, A, B, C, D, E, F
- **Examples**

  Decimal number – 10, 145,-89, 067 etc.

  Octal number – 037, 0, 057, 0456 etc.

  Hexadecimal number – 0x4, 0x9C, 0xAFE etc.
- **Rules for an integer constant**
  - It must have at least one digit.
  - Decimal point is not allowed.
  - It can be either positive or negative.

- If it is negative the sign must be preceded. For positive the sign is not necessary.
- No commas or blank spaces are allowed.
- The allowable range for integer constant is –32,768 to +32,767

Real Constant

- It is made up of a sequence of numeric digits with presence of a decimal point.
- It is to represent quantities that vary continuously such as distance, height, temperature etc.
- Example:

  Distance=134.9;

  Height=88.10;

- **Rules for a real constant**
  - It must have one digit.
  - It must have decimal point.
  - It can be either positive or negative.
  - If it is negative the sign must be preceded. For positive the sign is not necessary.
  - No commas or blank spaces are allowed.

Character constants

Single Character Constant

- It contains a single character enclosed within a pair of single quote marks.

  Example

  ‘d’, ‘r’, ‘6’, ‘_’

String Constant

- It is a sequence of characters enclosed in double quotes.
- The characters may be letters, numbers, special characters and blank spaces
- At the end of string ‘\0’ is automatically placed.
- **Example**

  “hai”

  “4565”