# Fundamentals of C programming :

C Introduction

* Keywords and Identifiers
* Variables & Constants
* C Data Types
* C Input/output
* C Operators
* C Introduction Examples

C Flow Control

* C if..else
* C for Loop
* C while Loop
* C break and continue
* C switch…case
* C Programming go to
* Control flow Examples

C Functions

* C programming Functions
* C User-defined Functions
* C Functions Types
* C Recursion
* C Functions Examples

C programming Arrays

* C Programming Arrays
* C Multi-dimensional Array
* C Arrays & Function

C Programming Pointers

* C programming Pointers
* C Pointers & Arrays
* C pointers and Functions
* C Memory Allcation
* Array & Pointer Examples

C Programming Strings

* C Programming String
* C String Functions
* C String Examples

Structure and Union

* C Structure
* C Struct & Pointers
* C Struct & Function
* C Unions
* C Struct Examples

C Programming Files

* C Files Input/output
* C Files Examples

1. What Is c programming?

C is a powerful general-purpose programming language. It can be used to develop software like operating systems, databases, compilers, and so on. C programming is an excellent language to learn to program for beginners. C Programming Language is a very popular computer programming language through which users and computers can communicate.

C is a computer programming language used to design computer software and applications.

# C Keywords and Identifiers

**Character set**

A character set is a set of alphabets, letters and some special characters that are valid in C language.

**Alphabets**

Uppercase: A B C ................................... X Y Z

Lowercase: a b c ...................................... x y z

C accepts both lowercase and uppercase alphabets as variables and functions.

**Digits**

0 1 2 3 4 5 6 7 8 9

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Special Characters in C Programming** | | | | |
| **,** | **<** | **>** | **.** | **\_** |
| **(** | **)** | **;** | **$** | **:** |
| **%** | **[** | **]** | **#** | **?** |
| **'** | **&** | **{** | **}** | **"** |
| **^** | **!** | **\*** | **/** | **|** |
| **-** | **\** | **~** | **+** |  |

**White space Characters**

Blank space, newline, horizontal tab, carriage return and form feed.

## C Keywords

Keywords are predefined, reserved words used in programming that have special meanings to the compiler. Keywords are part of the syntax and they cannot be used as an identifier. For example:

int money;

Here, int is a keyword that indicates money is a [variable](https://www.programiz.com/c-programming/c-variables-constants) of type int (integer).

As C is a case sensitive language, all keywords must be written in lowercase. Here is a list of all keywords allowed in ANSI C.

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

## C Identifiers

Identifier refers to name given to entities such as variables, functions, structures etc.

Identifiers must be unique. They are created to give a unique name to an entity to identify it during the execution of the program. For example:

int money;

double accountBalance;

Here, money and accountBalance are identifiers.

Also remember, identifier names must be different from keywords. You cannot use int as an identifier because int is a keyword.

### Rules for naming identifiers

1. A valid identifier can have letters (both uppercase and lowercase letters), digits and underscores.
2. The first letter of an identifier should be either a letter or an underscore.
3. You cannot use keywords like int, while etc. as identifiers.
4. There is no rule on how long an identifier can be. However, you may run into problems in some compilers if the identifier is longer than 31 characters.

You can choose any name as an identifier if you follow the above rule, however, give meaningful names to identifiers that make sense.

# C Variables, Constants and Literals

## Variables

In programming, a variable is a container (storage area) to hold data.

To indicate the storage area, each variable should be given a unique name ([identifier](https://www.programiz.com/c-programming/c-keywords-identifier)). Variable names are just the symbolic representation of a memory location. For example:

int playerScore = 95;

Here, playerScore is a variable of int type. Here, the variable is assigned an integer value 95.

The value of a variable can be changed, hence the name variable.

char ch = 'a';

// some code

ch = 'l';

### Rules for naming a variable

1. A variable name can only have letters (both uppercase and lowercase letters), digits and underscore.
2. The first letter of a variable should be either a letter or an underscore.
3. There is no rule on how long a variable name (identifier) can be. However, you may run into problems in some compilers if the variable name is longer than 31 characters.

**Note:** You should always try to give meaningful names to variables. For example: firstName is a better variable name than fn.

C is a strongly typed language. This means that the variable type cannot be changed once it is declared. For example:

int number = 5; // integer variable

number = 5.5; // error

double number; // error

Here, the type of number variable is int. You cannot assign a floating-point (decimal) value 5.5 to this variable. Also, you cannot redefine the data type of the variable to double. By the way, to store the decimal values in C, you need to declare its type to either double or float.

Visit this page to learn more about [different types of data a variable can store](https://www.programiz.com/c-programming/c-data-types).

## Literals

Literals are data used for representing fixed values. They can be used directly in the code. For example: 1, 2.5, 'c' etc.

Here, 1, 2.5 and 'c' are literals. Why? You cannot assign different values to these terms.

### 1. Integers

An integer is a numeric literal(associated with numbers) without any fractional or exponential part. There are three types of integer literals in C programming:

* decimal (base 10)
* octal (base 8)
* hexadecimal (base 16)

For example:

Decimal: 0, -9, 22 etc

Octal: 021, 077, 033 etc

Hexadecimal: 0x7f, 0x2a, 0x521 etc

In C programming, octal starts with a 0, and hexadecimal starts with a 0x.

### 2. Floating-point Literals

A floating-point literal is a numeric literal that has either a fractional form or an exponent form. For example:

-2.0

0.0000234

-0.22E-5

**Note:** E-5 = 10-5

### 3. Characters

A character literal is created by enclosing a single character inside single quotation marks. For example: 'a', 'm', 'F', '2', '}' etc.

### 4. Escape Sequences

Sometimes, it is necessary to use characters that cannot be typed or has special meaning in C programming. For example: newline(enter), tab, question mark etc.

In order to use these characters, escape sequences are used.

| Escape Sequences | |
| --- | --- |
| Escape Sequences | Character |
| \b | Backspace |
| \f | Form feed |
| \n | Newline |
| \r | Return |
| \t | Horizontal tab |
| \v | Vertical tab |
| \\ | Backslash |
| \' | Single quotation mark |
| \" | Double quotation mark |
| \? | Question mark |
| \0 | Null character |

For example: \n is used for a newline. The backslash \ causes escape from the normal way the characters are handled by the compiler.

### 5. String Literals

A string literal is a sequence of characters enclosed in double-quote marks. For example:

"good" //string constant

"" //null string constant

" " //string constant of six white space

"x" //string constant having a single character.

"Earth is round\n" //prints string with a newline

## Constants

If you want to define a variable whose value cannot be changed, you can use the const keyword. This will create a constant. For example,

const double PI = 3.14;

Notice, we have added keyword const.

Here, PI is a symbolic constant; its value cannot be changed.

const double PI = 3.14;

PI = 2.9; //Error

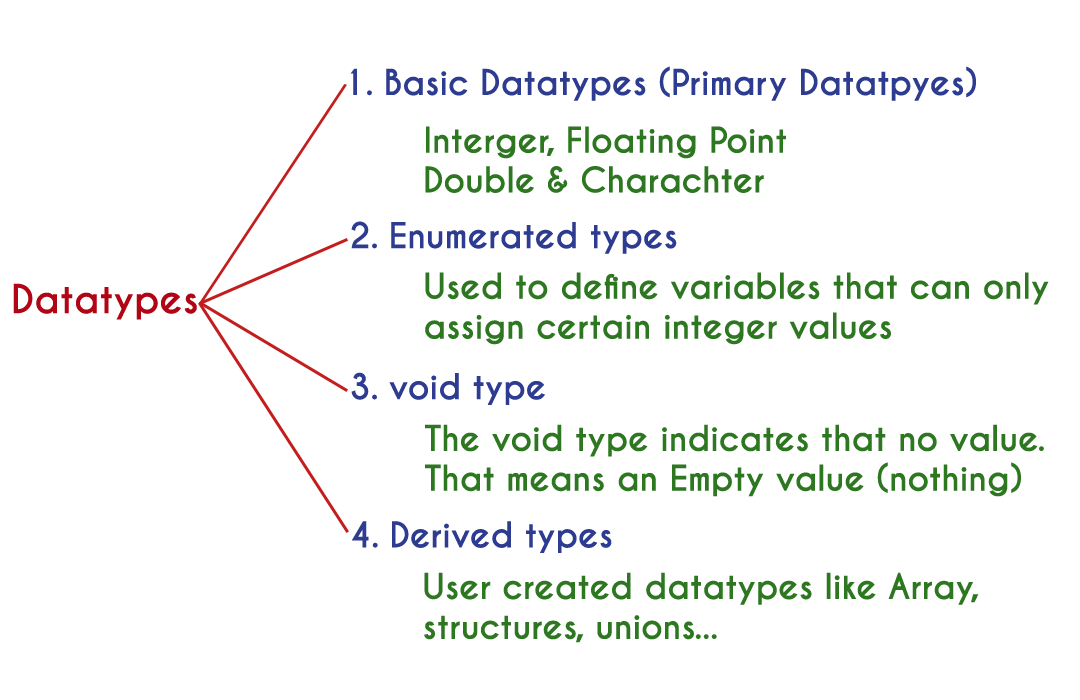
# C data types

Data used in c program is classified into different types based on its properties. In the C programming language, a data type can be defined as a set of values with similar characteristics. All the values in a data type have the same properties.  
  
Data types in the c programming language are used to specify what kind of value can be stored in a variable. The memory size and type of the value of a variable are determined by the variable data type. In a c program, each variable or constant or array must have a data type and this data type specifies how much memory is to be allocated and what type of values are to be stored in that variable or constant or array. The formal definition of a data type is as follows...

**The Data type is a set of value with predefined characteristics. data types are used to declare variable, constants, arrays, pointers, and functions.**

In the c programming language, data types are classified as follows...

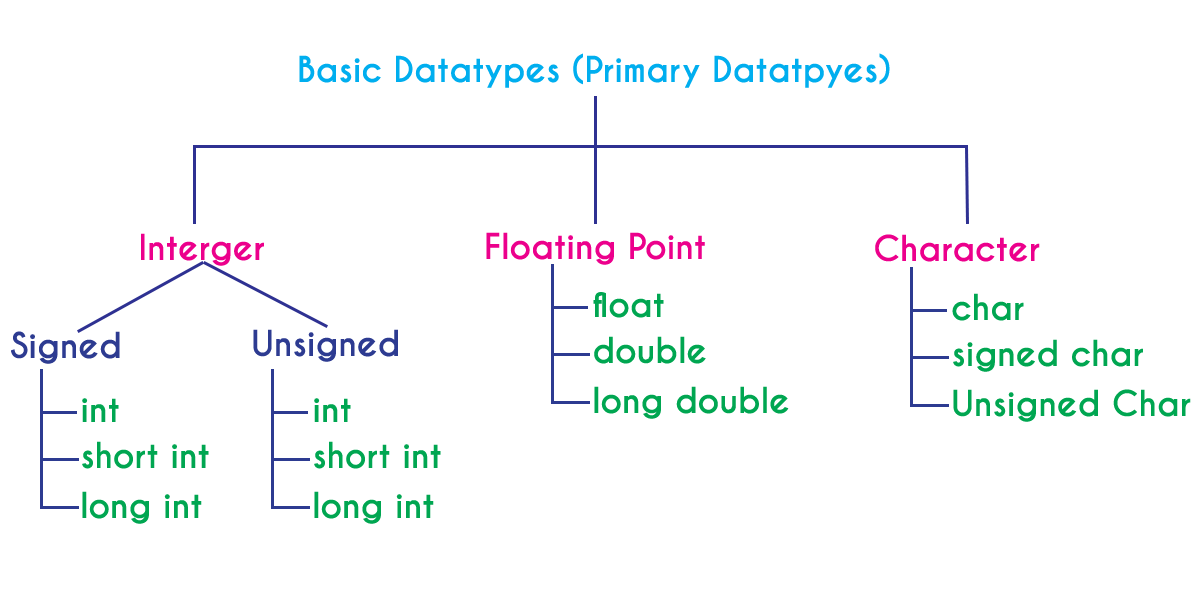
1. Primary data types (Basic data types OR Predefined data types)
2. Derived data types (Secondary data types OR User-defined data types)
3. Enumeration data types
4. Void data type



# Primary data types

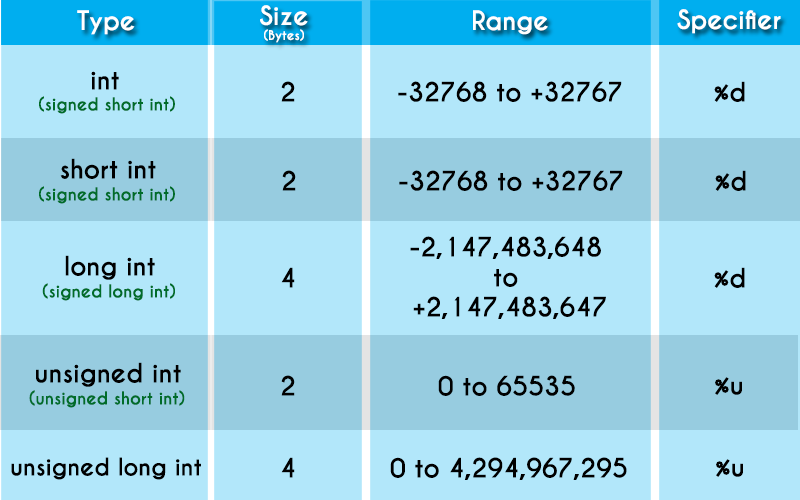
The primary data types in the C programming language are the basic data types. All the primary data types are already defined in the system. Primary data types are also called as Built-In data types. The following are the primary data types in c programming language...

1. Integer data type
2. Floating Point data type
3. Double data type
4. Character data type



# Integer Data type

The integer data type is a set of whole numbers. Every integer value does not have the decimal value. We use the keyword "**int**" to represent integer data type in c. We use the keyword int to declare the variables and to specify the return type of a function. The integer data type is used with different type modifiers like short, long, signed and unsigned. The following table provides complete details about the integer data type.

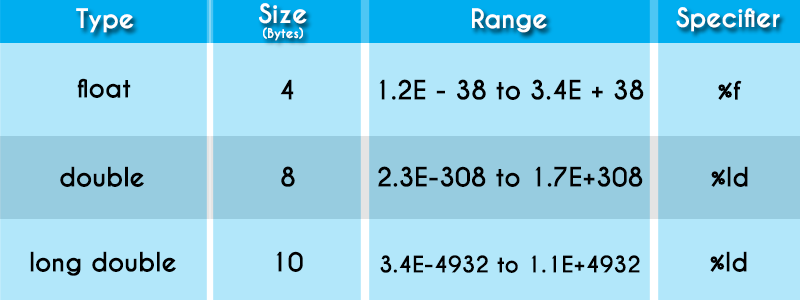


# Floating Point data types

Floating-point data types are a set of numbers with the decimal value. Every floating-point value must contain the decimal value. The floating-point data type has two variants...

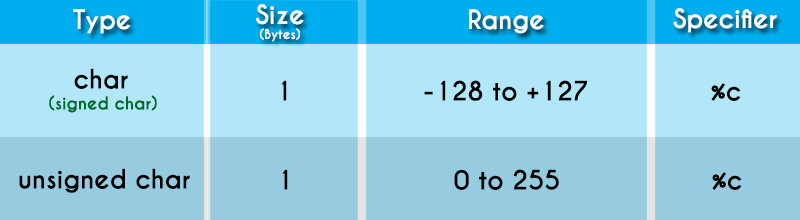
* float
* double

We use the keyword "**float**" to represent floating-point data type and "**double**" to represent double data type in c. Both float and double are similar but they differ in the number of decimal places. The float value contains 6 decimal places whereas double value contains 15 or 19 decimal places. The following table provides complete details about floating-point data types.

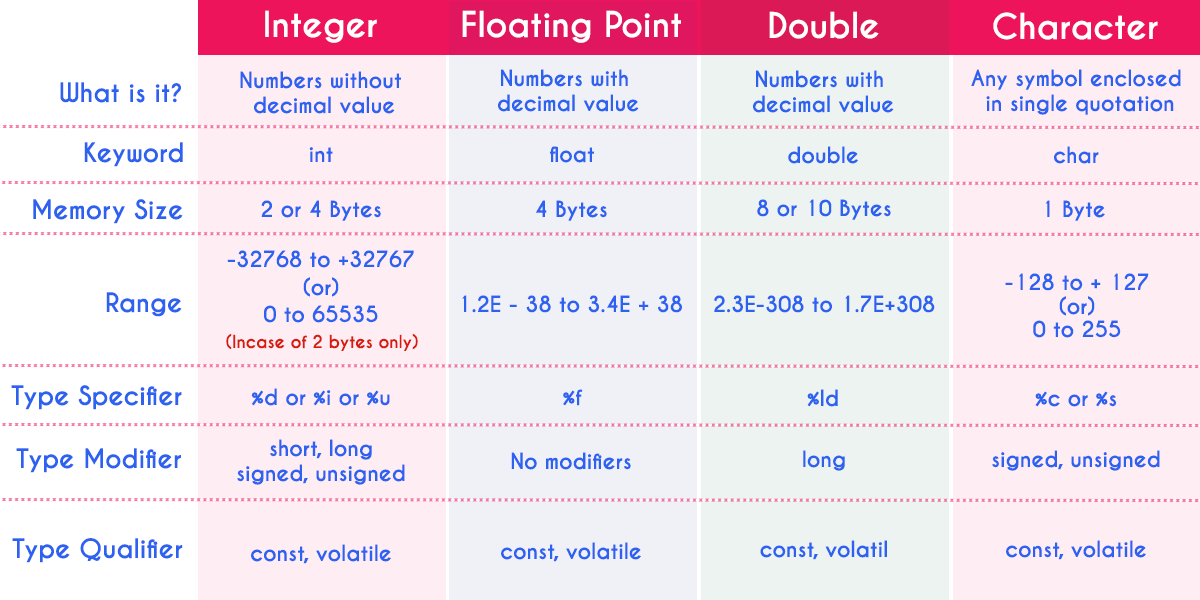


# Character data type

The character data type is a set of characters enclosed in single quotations. The following table provides complete details about the character data type.



The following table provides complete information about all the data types in c programming language...



# void data type

The void data type means nothing or no value. Generally, the void is used to specify a function which does not return any value. We also use the void data type to specify empty parameters of a function.

# Enumerated data type

An enumerated data type is a user-defined data type that consists of integer constants and each integer constant is given a name. The keyword "**enum**" is used to define the enumerated data type.

# Derived data types

Derived data types are user-defined data types. The derived data types are also called as user-defined data types or secondary data types. In the c programming language, the derived data types are created using the following concepts...

* Arrays
* Structures
* Unions
* Enumeration