Data Types in C

A data types in C refers to the type of data used to store information. For example, a person's name would be an array of characters, whereas their age would be in integers. On the other hand, a student's grades would necessitate using a data type that can store decimal values.

There are four types of data types in C to differentiate and store various data types. They are listed below:

- Basic data types in C
- Derived data types in C
- Enumeration data types in C
- Void data types in C

Classification of Data Types in C

Data types in C are classified in various ways. These classifications help to organize the different types of data that can be used in C programs and provide a framework for understanding how data types can be used and manipulated in the language.

Numeric Data Types

- int
- short
- long
- long long
- float
- double
- long double

Character Data Types

- char

Unsigned Data Types

- unsigned int
- unsigned short
- unsigned long
- unsigned long long

Boolean Data Type

- _Bool

Complex Data Type

- _Complex

Imaginary Data Type

- _Imaginary

Check out: Difference Between Fundamental Data Types and Derived Data Types

Data Types in C and Their Size

The table shown below carries the description of the data types in C along with their sizes as per the requirements specified by the C standard:

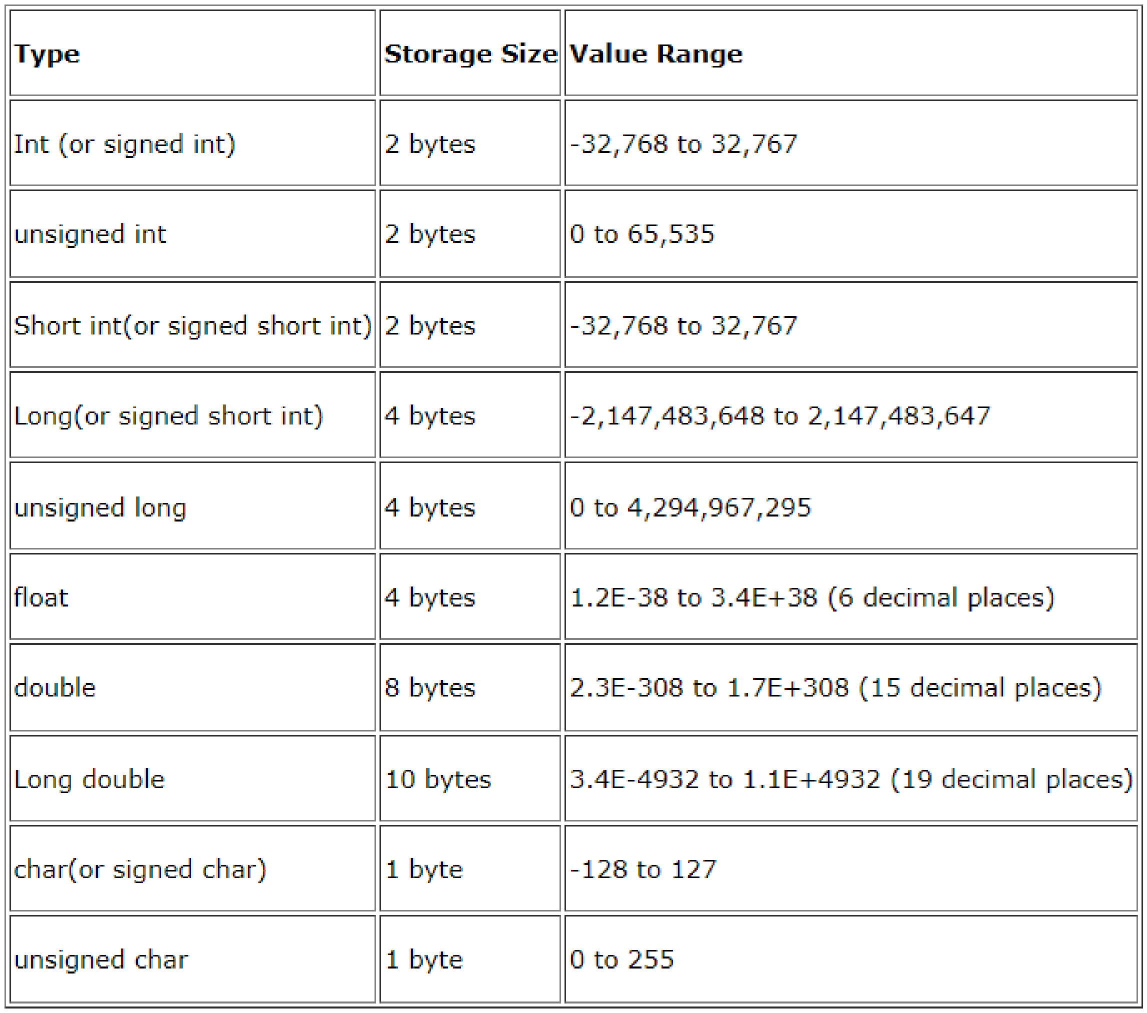
Data Type	Size (in bytes)
char	1
short	2
int	2 or 4
long	4 or 8
float	4
double	8
long double	10, 12, or 16
_Bool	1

Basic Data Types in C

Basic data types in C are used to store values in integer and decimal formats. It can handle both signed and unsigned literals. There are four basic or primitive data types in C that can be signed or unsigned that are as follows:

- Int
- Float
- Double
- Char

Depending on the operating system, the memory size of these data types in C can vary (32-bit or 64-bit). According to the 32-bit architecture, the table below shows the data types commonly used in C programming, along with their storage size and value range.



Type	Storage Size	Value Range
Int (or signed int)	2 bytes	-32,768 to 32,767
unsigned int	2 bytes	0 to 65,535
Short int(or signed short int)	2 bytes	-32,768 to 32,767
Long(or signed short int)	4 bytes	-2,147,483,648 to 2,147,483,647
unsigned long	4 bytes	0 to 4,294,967,295
float	4 bytes	1.2E-38 to 3.4E+38 (6 decimal places)
double	8 bytes	2.3E-308 to 1.7E+308 (15 decimal places)
Long double	10 bytes	3.4E-4932 to 1.1E+4932 (19 decimal places)
char(or signed char)	1 byte	-128 to 127
unsigned char	1 byte	0 to 255

Integer Data Types in C

An integer variable can hold zero, positive, and negative values without decimals. The integer data types in C are represented by the 'int' keyword, which can be signed or unsigned. If an integer variable's value is unsigned, it is assumed to be positive by default.

Short, int, and long data types are subsets of the integer data type in C. Short data types require 2 bytes of storage space, int data types require 2 or 4 bytes, and long data types require 8 bytes in 64-bit operating systems and 4 bytes in 32-bit operating systems.

Float Data Types in C

Float variables are used to store decimal values with up to six digits after the decimal place. The float variable has a storage size of 4 bytes, but this can vary depending on the processor, just like the 'int' data type.

The format specifier '%f' in C is used to represent float values. A variable with an integer value will also be printed with redundant zeroes in the floating type. The floating point data types in C refer to all real number values or decimal points, such as 40.1, 820.673, 5.9, and so on.

Double Data Types in C

The double data types in C are similar to the float data types in C, but it allows for up to ten digits after the decimal place. The double data types in C are represented by the 'double' keyword and are primarily used in scientific programs that require extreme precision.

The double data type occupies 8 bytes of memory and can store numbers with up to 15 decimal digits of precision. It has a range of approximately ± 5.0 × 10^−324 to ± 1.7 × 10^308.

n C, we can declare a double variable using the double keyword followed by the variable name. For example, we can declare a double variable to store the value of pi as follows:

```
double pi = 3.14159265358979323846;
```

We can also use the %lf format specifier to print the value of a double variable using printf(). For example:

```
double x = 1.234567890123456789;
printf("The value of x is %lf", x);
```

This will print:

```
The value of x is 1.234568
```

Note that the value is rounded off to 6 decimal places by default. If we want to print more decimal places, we can use the precision specifier %.nf, where n is the number of decimal places we want to print. For example:

```
double y = 1.234567890123456789;
printf("The value of y is %.15lf", y);
```

This will print:

The value of y is 1.234567890123457
Here, we have used %.15lf to print the value of y with 15 decimal places of precision.

Char Data Types in C

The character value char is used to store single-character values, including numerical values. When you create an array of the character data type, it transforms into a string that can store values like name, subject, etc.

Here's an example of creating an array of characters in C programming.

A char variable can be declared using the char keyword, followed by the variable name. For example, to declare a char variable c and assign it the value 'A', we can write:

```
char c = 'A';
```

In C, we can use single quotes to specify character literals. For example, the expression 'a' represents the character 'a', and the expression '5' represents the character '5'.

We can also use escape sequences to represent special characters such as newline (\n), tab (\t), and backslash (\\). For example, the expression '\\' represents the backslash character.

We can print the value of a char variable using the %c format specifier with printf(). For example, to print the value of the char variable c, we can write:

```
printf("The value of c is %c\n", c);
```

This will print:

```
The value of c is A
```

In C, we can also use arrays of char variables to represent strings. A string is a sequence of characters terminated by a null character (\0). For example, the following code declares a char array str and assigns it the value "Hello":

```
char str[] = "Hello";
```

Here, the array str has 6 elements, including the terminating null character.

We can print a string using the %s format specifier with printf(). For example, to print the value of the char array str, we can write:

```
printf("The string is %s\n", str);
```

This will print:

```
The string is Hello
```

Derived Data Types in C

The basic data types in C are used to store single values of various forms; however, what if you need to store multiple values of the same data type? In this case, derived data types enable you to combine basic data types and store multiple values in a single variable.

The user defines the derived data types in C, and you can aggregate as many elements of similar data types as you need. User-defined data type or derived data types are classified into four types:

- Array
- Pointer
- Structure
- Union
- Array

A collection of similar data-type elements stored in a contiguous memory location is referred to as an array. For example, float values such as a student's grades, integer values such as the roll number, and so on can be stored.

To understand the array, consider an example:

```
#include<stdio.h>

void main() {

int a[]={7,5,3,8,9};

printf("The array elements are: \n");

for(i=0;i<5;i++){

printf("%d \t",a[i]);

}

}
```

Output:

```
The array elements are:

7 5 3 8 9
```

Pointer

A pointer variable is used to store the address of another variable. The address of a variable should be stored in a pointer variable of the same data type. A null pointer is a pointer that has no address, whereas a void or general-purpose pointer has no data type.

In C, a pointer allows a user to perform dynamic memory allocation and pass variables by reference, which means that the user can pass the pointer containing the variable's address.

Structure

A structure in C is a user-defined data type in C that are a collection of items used to store the values of similar or different data types. For example, the structure can be used to store information about a student, such as their name, roll number, grades, and so on. A structured object will represent each student's record.

A structure can be built both inside and outside the main method. The keyword struct is used to define a structure. The structure's size would be the sum of the storage sizes required by each variable in the structure variable.

Union

Union is also a collection of elements with similar or dissimilar data types, but all of the elements share the same memory location. Union is a type of data type in C that allows you to declare multiple variables, but only one variable can store the value at any given time.

The keyword 'union' defines a union, where each object represents a single record. A union's size will be equal to the amount of memory required for the largest variable contained within the union variable.

Enumeration Data Types in C

Enumeration is a user-defined data type in C that are used to give names to integral constants and improve program readability. The keyword for enumeration is 'enum,' and its syntax is similar to the structure:

enum flag{const1, const2, const3……...};

Enumerations are preferable to '#define' for two reasons:

The compiler assigns default values to enum constants.

They are possible to declare in the local scope.

Void Data Types in C

Void is a data type in C that do not refer to any value of any type. It is commonly used as the return type in functions. You can declare void pointers to take the address of variables from any data type. These pointers are commonly referred to as 'generic pointers.'

A void function is a function that does not return a value. It is declared using the void keyword as the return type. For example:

```
void print_hello() {
printf("Hello!\n");
}
```

Here, the function print_hello() does not take any arguments and does not return a value. It simply prints the message "Hello!" to the console.

A void pointer is a pointer that can point to any type of object. It is declared using the void keyword as the data type. For example:

```
void* ptr;
```

Here, ptr is a void pointer that can point to an object of any type. To use a void pointer, we must first cast it to the appropriate type before dereferencing it. For example, if we have a void pointer ptr that points to an int value, we can cast it to an int* pointer and dereference it as follows:

```
void* ptr;
int x = 10;
ptr = &x;
int* int_ptr = (int*)ptr;
printf("The value of x is %d\n", *int_ptr);
```

This will print:

The value of x is 10

Here, we have assigned the address of the variable x to the void pointer ptr. We then cast ptr to an int* pointer and dereferenced it to obtain the value of x.

Uses of Data Types in C

Data types in C are primarily used for the purposes of understanding the data type used (for processing/returning). This allows both the programmer (optionally) and the compiler to use the values in the correct format. The programmer can also select the data type based on the requirements such as precision, integer values, range, and sign.

Another purpose of using the data types in C is that the memory size is associated with each data type. When appropriate data types are used, proper memory utilization is possible.