

Data Types and Sizes

Get familiar with all the data types used in C and also the sizes they occupy in the memory.

We'll cover the following

- Binary representation
- How many bytes on your machine?

There are four basic data types in C. Their meaning, and their size (on my MacBook Pro 15-inch, Mid 2010) are as follows:

Type	Meaning	Size (bytes)	Size (bits)
<code>`char`</code>	a single byte, capable of holding one character	1 byte	8 bits
<code>`int`</code>	an integer	4 bytes	32 bits
<code>`float`</code>	single-precision floating point number	4 bytes	32 bits
<code>`double`</code>	double-precision floating point number	8 bytes	64 bits

Binary representation

There are also qualifiers `short`, `long`, `signed` and `unsigned`, that can be applied to these basic types.

Qualifier	Size (bytes)	Size (bits)
<code>`short int`</code>	2 bytes	16 bits
<code>`long int`</code>	8 bytes	64 bits
<code>`long double`</code>	16 bytes	128 bits

We have been talking about variable types and how many **bytes** they take up in memory. An important quantity to know about is that one **byte** is made up of 8 **bits**. One **bit** can take on two possible values: 0 or 1. An **unsigned** 8-bit variable can take on values between 0 and $(2^8)-1 = 255$. A **signed** 8-bit variable can take on values between -127 and +127.

So when a variable is **signed**, it can take on negative values, and half of its total range is spread below zero, and the other half above zero.

A signed `int` can take on values between -2,147,483,648 and +2,147,483,648. If we want to be able to represent integers larger than +2,147,483,648, then we can either use more bits (e.g., by using a `long int`), or by forcing all 32 bits of our `int` to be used on the positive side of zero. An `unsigned int` (4 bytes or 32 bits) can take on values between 0 and 4,294,967,295.

How many bytes on your machine?

Execute the following code that will print out the size of some basic C types.

```
#include <stdio.h>

int main(int argc, char *argv[]) {
    printf("a char is %ld bytes\n", sizeof(char));
    printf("an int is %ld bytes\n", sizeof(int));
    printf("an float is %ld bytes\n", sizeof(float));
    printf("a double is %ld bytes\n", sizeof(double));
    printf("a short int is %ld bytes\n", sizeof(short int));
    printf("a long int is %ld bytes\n", sizeof(long int));
    printf("a long double is %ld bytes\n", sizeof(long double));
    return 0;
}
```





We can assign values in these data types to any variable. As the name suggests, the data in a variable can change or “vary”. However, there is a certain type of variable whose value always remains the same. We’ll learn about this in the next lesson.