C Data Types

In C programming, data types are declarations for variables. This determines the type and size of data associated with variables. For example,

```
int myVar;
```

Here, `myVar` is a variable of `int` (integer) type. The size of `int` is 4 bytes.

Basic types

Here's a table containing commonly used types in C programming for quick access.

Type	Size (bytes)	Format Specifier
`int`	at least 2, usually 4	`%d`, `%i`
`char`	1	`%c`
`float`	4	`%f`
`double`	8	`%lf`
`short int`	2 usually	`%hd`
`unsigned int`	at least 2, usually 4	`%u`
`long int`	at least 4, usually 8	`%ld`, `%li`
`long long int`	at least 8	`%lld`, `%lli`
`unsigned long int`	at least 4	`%lu`
`unsigned long long int`	at least 8	`%llu`
`signed char`	1	`%c`

Type	Size (bytes)	Format Specifier
`unsigned char`	1	`%c`
`long double`	at least 10, usually 12 or 16	`%Lf`

int

Integers are whole numbers that can have both zero, positive and negative values but no decimal values. For example, `0`, `-5`, `10`

We can use `int` for declaring an integer variable.

```
int id;
```

Here, `id` is a variable of type integer.

You can declare multiple variables at once in C programming. For example,

```
int id, age;
```

The size of `int` is usually 4 bytes (32 bits). And, it can take 2^{32} distinct states from `-2147483648` to `2147483647`.

float and double

`float` and `double` are used to hold real numbers.

```
float salary;
double price;
```

In C, floating-point numbers can also be represented in exponential. For example,

```
float normalizationFactor = 22.442e2;
```

What's the difference between `float` and `double`?

The size of `float` (single precision float data type) is 4 bytes. And the size of `double` (double precision float data type) is 8 bytes.

---

char

Keyword `char` is used for declaring character type variables. For example,

```
char test = 'h';
```

The size of the character variable is 1 byte.

---

void

`void` is an incomplete type. It means "nothing" or "no type". You can think of void as **absent**.

For example, if a function is not returning anything, its return type should be `void`.

Note that, you cannot create variables of `void` type.

---

short and long

If you need to use a large number, you can use a type specifier `long`. Here's how:

```
long a;
long long b;
long double c;
```

Here variables `a` and `b` can store integer values. And, `c` can store a floating-point number.

If you are sure, only a small integer (`[-32,767, +32,767]` range) will be used, you can use `short`.

```
short d;
```

You can always check the size of a variable using the `sizeof()` operator.

```
#include <stdio.h>
int main() {
  short a;
  long b;
  long long c;
  long double d;

  printf("size of short = %d bytes\n", sizeof(a));
  printf("size of long = %d bytes\n", sizeof(b));
  printf("size of long long = %d bytes\n", sizeof(c));
  printf("size of long double= %d bytes\n", sizeof(d));
  return 0;
}
```

Run Code »

---

signed and unsigned

In C, `signed` and `unsigned` are type modifiers. You can alter the data storage of a data type by using them:

- `signed` - allows for storage of both positive and negative numbers
- `unsigned` - allows for storage of only positive numbers

For example,

```
// valid codes
unsigned int x = 35;
int y = -35;  // signed int
int z = 36;  // signed int

// invalid code: unsigned int cannot hold negative integers
unsigned int num = -35;
```

Here, the variables `x` and `num` can hold only zero and positive values because we have used the `unsigned` modifier.

Considering the size of `int` is 4 bytes, variable `y` can hold values from -2^{31} to $2^{31}-1$, whereas variable `x` can hold values from `0` to $2^{32}-1$.

Derived Data Types

Data types that are derived from fundamental data types are derived types. For example: arrays, pointers, function types, structures, etc.

We will learn about these derived data types in later tutorials.

- bool type
- Enumerated type
- Complex types
