

## Variable Declarations

These are two valid declarations of variables. The first one declares a variable of type `int` with the identifier `a`. The second one declares a variable of type `float` with the identifier `mynumber`. Once declared, the variables `a` and `mynumber` can be used within the rest of their scope in the program.

If you are going to declare more than one variable of the same type, you can declare all of them in a single statement by separating their identifiers with commas. For example:

```
int a, b, c;
```

This declares three variables (`a`, `b` and `c`), all of them of type `int`, and has exactly the same meaning as:

```
int a;
int b;
int c;
```

The integer data types `char`, `short`, `long` and `int` can be either signed or unsigned depending on the range of numbers needed to be represented. Signed types can represent both positive and negative values, whereas unsigned types can only represent positive values (and zero). This can be specified by using either the specifier `signed` or the specifier `unsigned` before the type name. For example:

```
unsigned short int NumberOfSisters;
signed int MyAccountBalance;
```

By default, if we do not specify either `signed` or `unsigned` most compiler settings will assume the type to be signed, therefore instead of the second declaration above we could have written:

```
int MyAccountBalance;
```

with exactly the same meaning (with or without the keyword `signed`)

An exception to this general rule is the `char` type, which exists by itself and is considered a different fundamental data type from `signed char` and `unsigned char`, thought to store characters. You should use either `signed` or `unsigned` if you intend to store numerical values in a char-sized variable.

`short` and `long` can be used alone as type specifiers. In this case, they refer to their respective integer fundamental types: `short` is equivalent to `short int` and `long` is equivalent to `long int`. The following two variable declarations are equivalent:

```
short Year;
short int Year;
```

Finally, `signed` and `unsigned` may also be used as standalone type specifiers, meaning the same as `signed int` and `unsigned int` respectively. The following two declarations are equivalent:

```
unsigned NextYear;
unsigned int NextYear;
```

To see what variable declarations look like in action within a program, we are going to see the C++ code of the example about your mental memory proposed at the beginning of this section: