Data Types and Variables

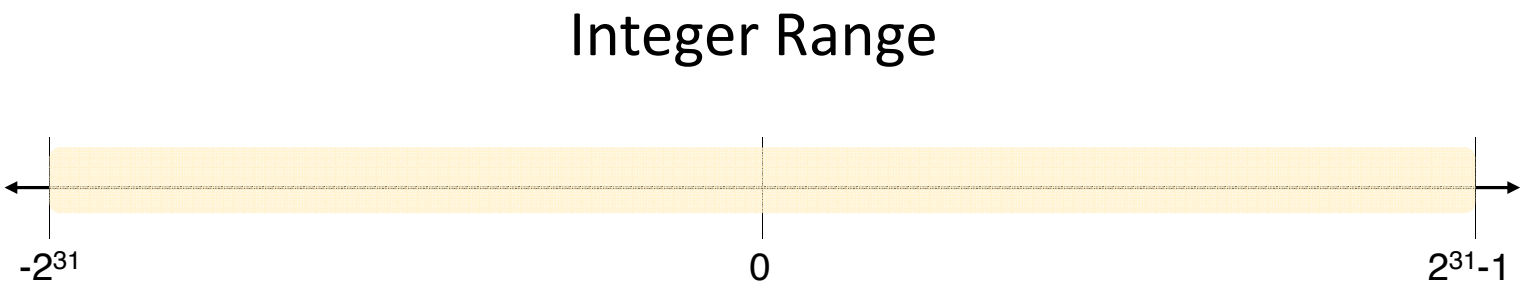
Data Types and Variables

- Unlike a number of modern programming languages, C requires that you specify the data type of every variable you create, the first time you use that variable.

- Let’s have a look at some of the data types that come with C and the data types we also provide for you in CS50.

Data Types and Variables

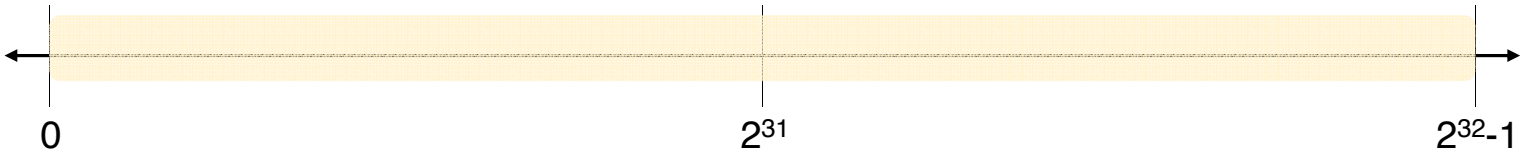
- `int`
  - The `int` data type is used for variables that will store integers.
  - Integers always take up 4 bytes of memory (32 bits). This means the range of values they can store is necessarily limited to 32 bits worth of information.



Data Types and Variables

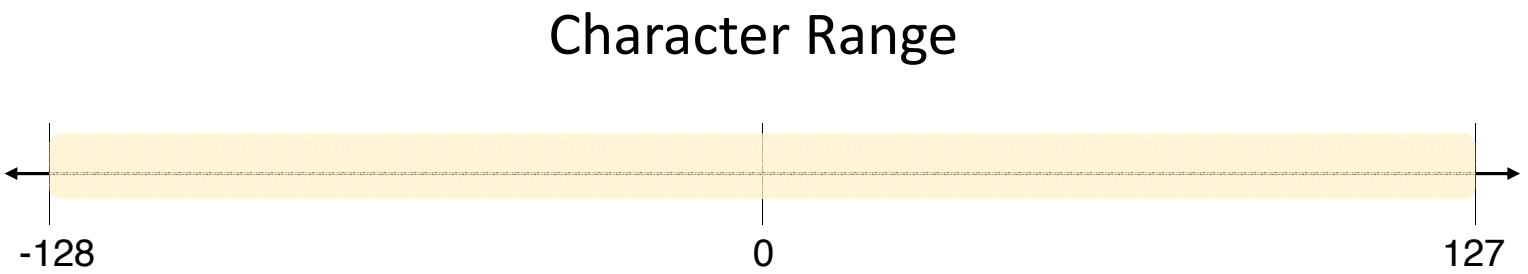
- `unsigned int`
  - `unsigned` is a *qualifier* that can be applied to certain types (including `int`), which effectively doubles the positive range of variables of that type, at the cost of disallowing any negative values.
  - You'll occasionally have use for `unsigned` variables in CS50.

Unsigned Integer Range



Data Types and Variables

- `char`
  - The `char` data type is used for variables that will store single characters.
  - Characters always take up 1 byte of memory (8 bits). This means the range of values they can store is necessarily limited to 8 bits worth of information.
  - Thanks to ASCII, we've developed a mapping of characters like A, B, C, etc… to numeric values in the positive side of this range.



Data Types and Variables

- `float`
  - The `float` data type is used for variables that will store floating-point values, also known as *real numbers*.
  - Floating points values always take up 4 bytes of memory (32 bits).
  - It's a little complicated to describe the range of a `float`, but suffice it to say with 32 bits of precision, some of which might be used for an integer part, we are limited in how *precise* we can be.

Data Types and Variables

- `double`
  - The `double` data type is used for variables that will store floating-point values, also known as *real numbers*.
  - The difference is that doubles are *double precision.* They always take up 8 bytes of memory (64 bits).
  - With an additional 32 bits of precision relative to a `float`, `double`s allow us to be specify much more precise real numbers.

Data Types and Variables

- `void`
  - Is a type, but not a *data type*.
  - Functions can have a `void` return type, which just means they don't return a value.
  - The parameter list of a function can also be `void`. It simply means the function takes no parameters.
  - For now, think of `void` more as a placeholder for "nothing". It's more complex than that, but this should suffice for the better part of the course.

Data Types and Variables

- Those are the five primary types you'll encounter in C.

- In CS50, we also provide you with two additional types that will probably come in handy.

Data Types and Variables

- `bool`
  - The `bool` data type is used for variables that will store a Boolean value. More precisely, they are capable only of storing one of two values: `true` and `false`.
  - Be sure to `#include <cs50.h>` atop your programs if you wish to use the `bool` type.

Data Types and Variables

- `string`
  - The `string` data type is used for variables that will store a series of characters, which programmers typically call a *string*.
  - Strings include things such as words, sentences, paragraphs, and the like.
  - Be sure to `#include <cs50.h>` atop your programs if you wish to use the `string` type.

Data Types and Variables

- Later in the course we'll also encounter structures (`structs`) and defined types (`typedefs`) that afford great flexibility in creating data types you need for your programs.

- Now, let's discuss how to create, manipulate, and otherwise work with variables using these data types.

Data Types and Variables

- Creating a variable
  - To bring a variable into existence, you need simply specify the data type of the variable and give it a name.

```
int number;
char letter;
```

  - If you wish to create multiple variables of the same type, you specify the type name *once*, and then list as many variables of that type as you want.

```
int height, width;
float sqrt2, sqrt3, pi;
```

  - In general, it's good practice to only *declare* variables when you need them.

Data Types and Variables

- Using a variable
  - After a variable has been *declared*, it's no longer necessary to specify that variable's type. (In fact, doing so has some unintended consequences!)

```
int number;          // declaration
number = 17;         // assignment
char letter;         // declaration
letter = 'H';        // assignment
```

  - If you are simultaneously declaring and setting the value of a variable (sometimes called *initializing*), you can consolidate this to one step.

```
int number = 17;     // initialization
char letter = 'H'; // initialization
```