Learning Objectives: Variables

- **Understand the rules of naming a variable**
- **Assign/overwrite a value to a variable**
- **Understand four basic data types: integers (ints), floats, boolean (bool), and strings**

Variables

What Is a Variable?

In computer science, we often need to use data. **Variables** are used to store a value for a particular type of data.

Each variable in C++ has:

1. a data type
2. a name
3. a value

We will discuss each of these parts over the rest of this reading assignment.

Three Actions for Variables

There are a few different actions taken involving variables:
1. **Declaring** - when you set or declare the *data type* and *name* of the variable. These two properties of a variable do *not* change.
1. **Assigning** - when you set the *value* of the variable. The value of a variable *can* change.
1. **Accessing** - when you retrieve the *value* of the variable by calling its *name*.

You *must* declare and assign a variable before you can access it.

Take a look at the visualizer on the left to see an example of how this works. Click on the `Forward >` button at the bottom of the page to repeatedly move through each stage of the process. *The visualizer may take a few seconds to load. Click on the `Refresh code` button in the upper left corner if you encounter an error message.*

Data Types: Integers

Integers

Integers (often called `ints`) are whole numbers. They can be positive or negative. Do not use a comma when typing large numbers.

Copy the code below into the text editor. Then click the `TRY IT` button.

```
int number;
number = 50;
cout << number << endl;
```

Next, let's modify the code to look like what's below and then click the `TRY IT` button again.

```
int number = 50;
cout << number << endl;
```

> important
>
> You may have noticed that we can declare a variable name and assign it a value all in one step by using `int number = 50;` instead of `int number;` followed by `number = 50;`. Both ways will produce the same result.

▼ **5 vs. "5"**

`5` is not the same thing as `"5"`. The first one is an integer, the second is a string. You will see in a later lesson the different operations you can perform on strings and numbers. Treating a string as a number can cause errors.

challenge

What happens if you:

- Change the variable to `5000`?
- Change the variable to `5,000`?
- Change the variable to `050`?
- Change the variable to `"5000"` (with double quotes)?

Data Types: Floating Point Numbers

Floating Point Numbers

Floating point numbers (often called floats) are numbers with a decimal. They can be positive or negative. Copy the code below and `TRY IT`.

```
double decimal = 0.5;
cout << decimal << endl;
```

Why Use Double Instead of Float?

In C++, there is a data type called **float**, but as it only uses 4 bytes, it is insufficient for most math. Instead, we use **double** which uses 8 bytes or double the space of a float.

challenge

What happens if you:

- Change the variable to `50.`?
- Change the variable to `.001`?

Data Types: Boolean

Boolean

A boolean variable (declared as a `bool`) can only take on the value of `true` or `false`. You will see how boolean values are used when we talk about conditionals and while loops. Copy the code below and `TRY IT`.

```
bool thisIsFun = true;
cout << boolalpha << thisIsFun << endl;
```

challenge

What happens if you:

- Change the variable to `false`?
- Remove the `boolalpha <<` command?
- Change the variable to `True`?
- Change the variable to `False`?
- Change the variable to `TRUE`?

important

You may have noticed that printing a boolean of `true` resulted in a `1` and a boolean of `false` resulted in a `0` when you remove the `boolalpha <<` command. In C++, the boolean value of `true` is associated with the integer `1` while the boolean value of `false` is associated with the integer `0`. Assigning the value of uppercase `True` or `False` to a boolean variable will cause an error message to appear.

Data Types: Strings

Strings

A string is a collection of text, numbers, or symbols. Strings are always surrounded by quotation marks. Copy the code below and `TRY IT`.

```
string words = "This is a string.";
cout << words << endl;
```

challenge

What happens if you:

- Forget one of the `"` quotation marks?
- Forget both `" "` quotation marks?
- Use single (`'`) quotation marks?
- Use uppercase `String` instead of lowercase `string`?

Notice that when you print a string, the quotation marks are not printed.

Declaring Variables

Declaring a Variable

Declaring a variable has two parts - setting or declaring the **data type** and the **name** of the variable. These two properties of a variable do **not** change.

To declare a variable, type the data type and name of the variable you want to create, and a `;` (semi-colon). Copy the code below and `TRY IT`.

```
string my_var;
```

You will notice we are not printing anything - that is because no value has been assigned yet. Thus, the message `Command was successfully executed.` appears when you click on the `TRY IT` button. The declaration step only sets aside empty memory.

challenge

What happens if you:

- Create two variables with the same type and name?
- Create two variables with the same name but different capitalization (i.e. `my_var` and `My_var`)?
- Create two variables of different types with the same name?

Variable Naming Rules

Here are the rules for naming a variable.

Rule	Correct	Incorrect
Start with a letter or underscore	`variable`, `_variable`	`1variable`
Remainder of variable name is letters,	`var_i_able`, `var1able`	`var-i-able`, `var!able`

numbers, or underscores		
Cannot use a C++ keyword	`my_class`	`class`
Variables are case sensitive	`variable`, `Variable`, and `VARIABLE` are all different variables	

What Are C++ Key Words?

C++ keys words are words that are reserved for specific functions or tasks within C++ programs. These words **cannot** be used to name variables and will result in errors if they are not handled correctly. Click below to see a list of C++ key words.

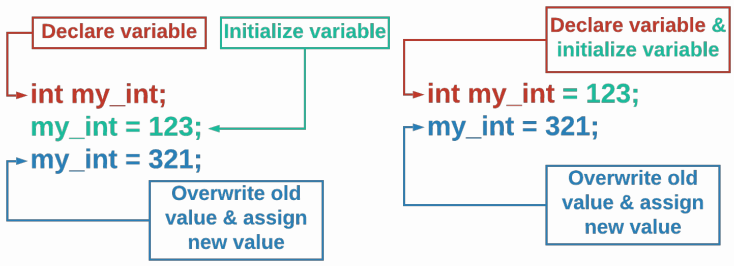
▼ **List of C++ key words**

and	and_eq	asm	auto	bitand
bitor	bool	break	case	catch
char	class	compl	const	const_cast
continue	default	delete	do	double
dynamic_cast	else	enum	explicit	extern
false	float	for	friend	goto
if	inline	int	long	mutable
namespace	new	not	not_eq	operator
or	or_eq	private	protected	pubic
register	reinterpret_cast	return	short	signed
sizeof	static	static_cast	struct	switch
template	this	throw	true	try
typedef	typeid	typename	union	unsigned
using	virtual	void	volatile	wchar_t
while	xor	xor_eq		

Initializing, Assigning, and Assessing

Initializing & Assigning Values

We call the process of setting the **initial** value of a variable **initialization**. Recall that you can do this separately after the declaration or combine it into the same statement as the declaration.



.guides/img/VariableAssignmentInt

Since the value stored in a variable can change, we call changing the value **assigning** or **re-assigning**. Use the assignment operator, `=`, to give a variable a new value.

Accessing Variables

Copy the code below and `TRY IT` to see the results of the `cout` commands. Click on the ++Code Visualizer++ link to see how the value of `my_int` changes.

```
int my_int = 123;
cout << my_int << endl;
my_int = 321;
cout << my_int << endl;
```

When we use a variable's name to get the value like in the `cout` statements above, we say we are **accessing** the variable.

Code Visualizer