

Variables

What are Variables?

In programming, variables are used as placeholders for information. A variable is a place in the memory of a computer (or for our purposes, the cRIO) where data can be stored and accessed.

Defining a Variable

To use a variable in your program, you first have to define (create) it. All variables are defined using the same format. To define a variable you follow the structure:

`type name;`
ex. `int myInt;`

* Note that every statement in C++ must end with a semicolon.

Types

In C++, you must tell the compiler what type of data your program will store. Some common data types are:

<i>Type</i>	<i>What it stores</i>	<i>Examples</i>
int	Integer Values (from -32768 to 32767)	-1234, -5, 0, 1, 567
double	Decimal Numbers	-99999, -945, 0.0, 1.0, 3.14
float	Decimal Numbers	-99999, -945, 0.0, 1.0, 3.14
bool	A boolean (True or False) value	true or false, 1 or 0
char	A single character in single quotations	'a', '\$', '4', 'h'
string	A group of characters in double quotations	"hello world!", "MMRambotics", "56"

* If you're wondering why double and float look the exact same, it's because they are very similar; however, a double will store more precise values after the decimal

When choosing variable types, you need to look at the potential values that variable will store and choose one that matches.

For example, lets say we need a variable to store a port number of a motor. Possible port numbers include 1, 2, 3, 4, etc. These numbers are all integers, therefore we would choose int as the data type.

This time, lets say we want to store a speed. We tell the motors of the robot how fast to go by giving them a value between -1.0 and 1.0. In this case, we want to be able to have decimals such as 0.5 or 0.66 and would choose either double or float as the data type.

Make sure you know what types your variables are because you cannot assign a value to a variable that does not match its type. For example, an integer value can only store integers, not

floats or strings. Assigning a value to a variable that does not match its type will cause an error.

Naming Variables

We name variables to describe the data they store. For example, a variable that stores a port number could be called `leftMotorPort`. This is an example of a good variable name because it tells us exactly what the variable stores. It would make no sense to call the variable `varNumberOne` because no one would know what that variable stored. Variable names must follow the following rules:

- 1) They can only contain letters, numbers and underscores.
- 2) They must not begin with a number.
- 3) There can be no spaces within a variable name
 - if you have a multiword variable, it is a convention to separate each word by using either camel case (ex. `leftMotorPort`) or underscores (ex. `left_motor_port`)

* Note that C++ is a case sensitive language. The variable called `Name` is different than one called `name`.

Assigning Values to a Variable

Variables are mostly useless unless you assign a value to them. You can give a variable a value when you declare it, or later in the program. However you cannot give a variable a value before you define it. To assign a value to a variable, you use the equals operator (`=`). It is a good idea to always assign a value to a variable when you define it (called initializing it). You can assign values to a variable as many times as you want, however the new value will overwrite the old one.

Examples:

Initializing a variable:

```
int leftMotorPort = 1;
```

```
bool isArmOut = true;
```

```
string name = "Mike";
```

Assigning values after defining the variable:

```
speed = 1.0;
```

```
gyroPort = 4;
```

```
isArmOut = false;
```

Why Use Variables

Why do we use variables if we could just hard code the values instead?

- 1) We use variables because they greatly reduce code maintenance and do not require us to remember any values. For example, say we had a program that used a variable called `leftMotorPort` in 10 different places within the program. The motor is in port 1, so we assign the

value 1 to this variable. Instead of using a variable we could have wrote the number 1 wherever we wrote leftMotorPort. This is inconvenient because if we ever need to change the motor port, we would have to change 10 different lines of code, whereas we would only change one line with a variable. Also a variable is way to describe a value and document your code. If you wrote this piece of code without variables, when you look back at it later, you may wonder why the number 1 is everywhere. However, if you used a variable with a good name, you would know exactly what that variable is for.

2) Sometimes, there will be no other way than to use a variable. You might have no idea what a value you need to use is before the code is executing. Therefore you will need to use a variable to store that value during execution. For example you may need to read a value from a sensor. But the sensor's value will be constantly changing based on what the robot is doing. You will need to use variables to store changing data because you cannot hard code data that changes.

Summary

- To define a variable, use the format type name; or type name = value;
- The data type of the variable must match the data type of its value.
- Common data types include: int, double, float, bool, char, and string.
- Name a variable to describe the information it stores
- Variables can be named ONLY with letters, underscores, and numbers (it cannot start with a number)
- To assign a value to a variable, use the equals operator: name = value;