

# Introduction to Programming

MVRT 115  
Sept. 27, 2006

- This is an introduction to programming in the C language.
- We will be working on Vex robots, and later on the large robot.
- In this class, we assume that you know little or nothing about programming in general.
- If you have programmed in C, C++, or Java, or if you were programming on the team last year, there will be a different class for you.
  - In the advanced class, we will be working on the sensors and camera.
  - We will talk about style, efficiency, testing and debugging.

## Just the Basics

- A computer is an electronic machine that can do a limited number of things, like adding, subtracting, multiplying, dividing, or comparing two numbers.
- A computer can *input* numbers from keyboards, disk drives, and sensors.
- The computer can *output* numbers to a computer screen, a disk, a printer, or a motor.
- Input & output is collectively referred to as *I/O*.
- Computer instructions are done in sequence, one after the other until the last instruction is *executed*; then the computer stops.
- A set of computer instructions is known as a *program*.
- Numbers can be stored (& retrieved) from a computer's *memory*.
- We say a program runs or executes, we can perform or *execute* an instruction



## Programming Languages

- Programming in machine language is tedious & error-prone for humans.
  - Computer memory locations are numbers. E.g., the first 256 bytes of memory are locations 0 through 255. (Everything on computers starts counting at zero.)
  - For example, to add two numbers in memory locations 150 and 200:
    - Get the first number from memory location 150
    - Get the second number from memory location 200
    - Add the two numbers together
    - Store the addition result in memory (maybe location 204)
  - Programming languages, like C & Java, make things easier: adding two numbers can be done in one line of C, and you don't have to know memory locations!, e.g.,  
`sum = a + b;`
- \* A bit is a single binary digit: either 0 or 1.  
A byte is a set of 8 binary digits and can represent 256 different values.

## The Most Important Thing

- The most important thing to know about programming is that computers do what you tell them to do.
  - Not what you would like them to do.
  - Not what you think they should do.
  - You must know how to translate what you want into a form the computer can use.
  - The syntax must be correct.
  - The logic must be correct.
- When the computer does something "wrong," it's almost certain that you programmed it to do the "wrong" thing.

## Basic program Form

Every program must contain zero or more statements enclosed in curly brackets "{" and "}"

- Every statement must end with a semicolon ;
- A program is a series of statements
- Statements are executed in order until we run out of statements. But...
  - You can skip statements with "if" statements
  - You can repeat statements with loops
- You can block groups of statements with "{" and "}" This is useful for "if" and "while"

## Syntax

- Syntax just means the way a statement is put together.
- Computer and human languages must have correct syntax, and every language has its own syntax .
  - You understand "the dog my homework ate," although it's syntax is wrong for English. (However, the syntax would be right for German: "Der Hund meine Aufgaben frass.")
  - Computers never understand incorrect syntax!

## A C program

```
main()
{
    int x = 3;    // a "//" is a comment
    int y, z;    // x is a variable
                // y & z are variables
    x = x + 2;   // assignment statement
    if (x < 6)  // if statement
    {
        y = 2;   // more assignment statements
        z = 5;
    }
}
```

## Variables

- How does a computer get a program?
  - You load it into memory
- You store programs and information (data) in memory. You don't (usually) need to know where the program is in memory; the computer knows where to look.
- For example, your program might start at memory locations (the addresses of bytes of memory) 100 to 200 and your data at memory addresses 201 to 275.
- It's difficult for humans to deal with numbered memory addresses, so you can name them. Named addresses are called variables.
- You must declare (name) a variable before you can use it.  
int x; names a variable x. Where is it in memory? Who cares; you can just call it x.

## Assignment

- How do you put values into a variable? With an assignment statement.  
x = 3;            // put a 3 in variable x
- The "=" sign means put whatever is on the right side into the left side. The right side value is not changed.
- The right hand side of the "=" can be an expression.  
x = y \*3 + 48;

## if statements

- sometimes you want to do something only in certain conditions, so you can use an "if" statement. If the expression inside the parentheses is true, then you do the statements between the "{" and "}" otherwise you skip them.  
if (motor\_speed > 255)  
{  
    motor\_speed = 255;  
}

## Let's Go Try It!