Last modification: January 15, 2020

RBE 1001: Introduction to Robotics
C-Term 2019-20
HW 1.0: Programming Essentials – Getting started with C++

1 Introduction

Programming skills are essential to robot development. For a robot to do anything useful, it must have a program that gathers information from sensors, makes control decisions, and commands actuators to produce actions. You will apply your programming skills to design, test, deploy, and improve robots in this course and in the future. It is important to be able to translate problems from the real world into code, and to be able to read, understand, and update code that is provided to you, even if you gravitate more towards the mechanical or electrical subsystems of a robot.

`C/C++` are the programming languages of choice for *embedded systems*, systems that rely on relatively small computers, called *microcontrollers*, to accomplish command and control. `python` and `java` are other popular languages (the former is very popular for vision processing; the latter for higher-level decision-making on more powerful systems), but `C/C++` are still the "go to" languages.

In this course, it is assumed that you have some background in programming, whether through an introductory course or practical experience ("Arduino" is really just `C++` with some minor embellishments). Regardless of your previous experience, these early exercises are designed to help you produce well-written programs in `C/C++` and understand code that may be provided to you.

In particular, we have selected a number of practice problems (with solutions) for you to work through if you have little experience with `C++` or need a refresher, followed by homework problems that highlight the needed concepts. You can do the practice problems on your own schedule, but help sessions will be scheduled if you need more background. The homework assignments are designed to prepare you for lectures, where we will focus on the *application* of programming concepts and less on the details of syntax.

For learning about `C++`, we will use an online resource, *Programming in C++ for Engineering and Science* by Larry Nyhoff, available online through Gordon Library.

We recommend you download or bookmark the link for this and subsequent exercises.

1.1 Writing code for exercises

For coding exercises, you have several choices, including,

- A development environment on your computer (e.g., the Eclipse IDE), or
- Using an online editor. If you're new to `C++`, we highly recommend OnlineGDB, which also includes a step-through debugger, should you want to use one.

Note that you will have the option of installing Eclipse as part of the `sloeber` package, should you choose to go that route for Arduino programming. The programs you write for these exercises,

however, will **not** compile for Arduino (neither in the Arduino IDE nor `sloeber`), since they make use of certain input/output functions that aren't available in the Arduino environment (most notably, `cin` and `cout`, but possibly others). You can still use the Eclipse installation, just with the standard `C++` toolchain and without the Arduino compiling tools.

1.2 Submission guidelines

For all programming exercises, your code must be submitted as a `.cpp` file (or `.ino` for later assignments with Arduino programs). If you have multiple files, you may submit them as either one `.zip` file, or multiple `.cpp`/`.h` files. Do **not** submit `.docx`, `.pdf`, or `.txt` files.

Use comments and blank lines to make it easier for someone else to understand what each line or section of code is accomplishing. Use meaningful variable names (e.g., "`temperatureF`" instead of "`T`"). Doing so will reduce the number of comments you will need.

Your code must compile. If it does not compile, the TA will make a note and give you a zero for that portion of the assignment. Be sure to check the TAs' comments on your assignments, however, as compiler differences will sometimes prevent your working code from compiling on their computers. If that happens, don't fret – you'll have a chance to demonstrate that your code works on your machine.

2 Introduction to C++ syntax

Regardless of your prior experience, read Chapter 2 in Nyhoff, paying particular attention to Section 2.2. You might also want to read Chapter 1, which presents an interesting history of computing and covers terminology that you should be familiar with from previous programming courses.

2.1 Practice exercises

- In Nyhoff, do the "Test Yourself" problems for Chapter 2. The answers for these problems are at the end of the book.

3 Problems

From Nyhoff,

1. Chp. 2, Exercise 3.
2. Chp. 2, Programming problem 5. You will be graded for "a solid attempt" at the code, not on whether or not it is entirely correct (though it must compile!). We'd rather you struggle a little on your own and get some things wrong (and learn what you need to work on) than have someone just tell you how to do it.

3.1 To submit

For convenience, put the answer to each problem is a separate `.cpp` file and submit all the files separately or as one `.zip` file. Title each file with "`problem<prob#>.cpp`", replacing the placeholder as appropriate. For example, "`problem1.cpp`".