Intro to C++ for FRC: Lesson 3

Joel Kuehne

https://github.com/JoelKueh/frc-cpp

Contents

Review: C++ Syntax and Homework

- If you have not yet looked at the sections from the C++ tutorial from last lesson, you should look at that now.
- From this point on, we will be talking more about overarching C++ concepts.
- From this point on, if you ever need to know the syntax, you should GOOGLE IT.

Printing in C++ (I'm Sorry)

- We're three lessons into this C++ tutorial, and I haven't officially told you how to print.
- You can use this awful syntax to print.
  ```
  std::cout << /*LIST OF THINGS YOU WANT TO PRINT*/;
  ```
- You start with a `std::cout`, then can put a chain of things to print separated by <<.
- These things can be strings (`"Hello"`), numbers (10), or variables (x).
- Adding the special keyword `std::endl` (short for end-line) adds a new line to the output.
  - It is important that you write new lines periodically, as the output is not guaranteed to be written until `std::endl` is sent.
- Review https://www.geeksforgeeks.org/cpp/basic-input-output-c/ for more information.

Logic Compartmentalization: Functions

- We talked about functions before, and you should have seen them in the C++ tuturial.
- I wanted to talk about them more officially now.
- Functions are the first tool in a programmers arsenal for breaking up complex problems into smaller pieces.
- We aren't just talking about mathematical functions now.
  - Functions don't need to have inputs and they don't need to return an output.
  - They just need to accomplish a task.

Example 1: Reusable Printing Function

- This example is simple, but I think it gets the idea across.
- You can use functions to section off logic to be reused.

```
void print_age(int age) {
  std::cout << "Your age is: " << age << std::endl;
}

int main() {
  print_age(23); // Outputs "Your age is: 23"
  print_age(15); // Outputs "Your age is: 15"
  print_age(18); // Outputs "Your age is: 18"
  return 0;
}
```

Exercise 1: Stopping Distance

- The stopping distance of a car can be modeled by the equation $d = K * v^2$.
  - d is the distance, v is velocity, and K is a constant.
- For a particular car, with distance d in feet and velocity v in MPH, the constant K is $\frac{1}{4}$.
- Write a function that takes the distance to an object in feet and the speed of the car in MPH, and print whether or not the car will collide with the object.
- Boilerplate code is below.

```
#include <iostream>

float test_collision(float distance, float speed) {
}

int main() {
  test_collision(80.0, 70.0);
  test_collision(200.0, 40.0);
}
```

Exercise 1: Stopping Distance

- This exercise is intended to be a step up from the previous ones.
- Remember that $K = 0.25$ for our problem.
- Break the problem up into pieces.
  - How are you going to write the equation $d = K * v^2$ in C++?
  - There isn't an operator in C++ for raising a number to a power (no squaring operator) so you will have to implement it yourself. How will you do this?
  - How are you going to check the distance?
  - How will you print the result?

A Step Back

- This marks the end of the beginning of this course.
- At this point, I hope that you all have a pretty good idea of general programming topics.
- Variables, functions, conditions, etc…
- From this point on, I hope that you will keep up with the tutorials in https://www.geeksforgeeks.org/cpp/c-plus-plus/. I don't have anything specific for you to practice, but you all should go over the sections that you don't remember again, and condinue to use it as a reference when you need it.