

INTRO TO C++ FOR FRC: LESSON 1

Joel Kuehne

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

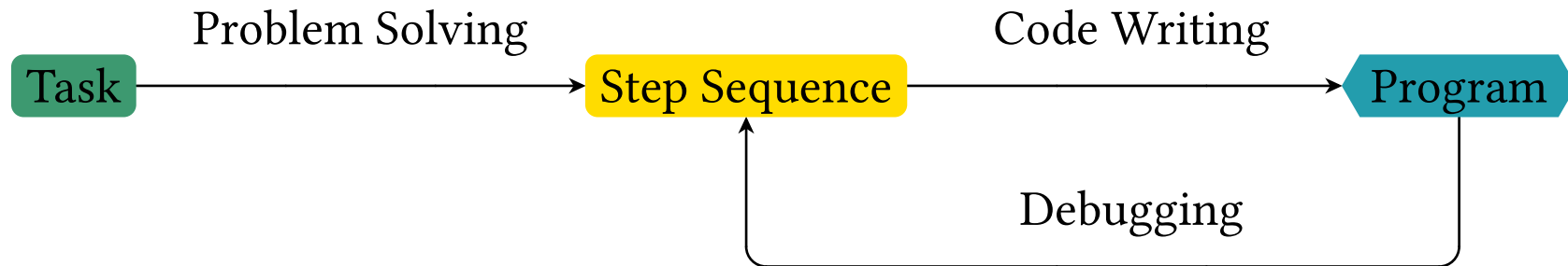
CONTENTS

1. About this Course
2. Programming as Logic
3. Programming as Math

- This course is an interactive introduction to C++ for FRC.
- Interspersed with “lecture” style content are real-world **EXAMPLES**.
- Additionally, a number of **EXCERSIZES** are prepared that you should do yourself.
 - You can write C++ online here: https://www.onlinegdb.com/online_c++_compiler¹
 - As we get further along, I will introduce more tools for you to use.

¹Online C++ Compiler: https://www.onlinegdb.com/online_c++_compiler

- Programming, at its root, is a means of turning complex logic into a series of statements that can be interpreted by a computer.
- The basic design flow for a program is as follows.
 1. Start with a task you want to automate.
 2. Break that task into a series of smaller steps (Problem Solving).
 3. Translate those smaller steps into a computer language (Code Writing).
 4. Test the resulting program and fix all of the errors (Debugging).



- Programming shares a lot of similarities with math.
- We compare computer programs programming to functions in math.
- Functions have inputs, outputs, and some “process” that maps inputs to outputs.
 - This is true for both math and programming.



$$f(x) = 10x + 13.3$$

Translates to...

```
float f(float x) {  
    return 10.0 * x + 13.3;  
}
```

1. Start with the logic that we want to implement.
 - Fahrenheit can be converted to Celsius using the following equation.

$$C(f) = (f - 32) \cdot \frac{5}{9}$$

2. Convert the logic function into a series of smaller steps
 - This doesn't really apply here, one equation is enough to get the job done.
3. Translate the steps into computer language.
 - `return (f - 32) * 5/9;`
4. Run the computer program and debug it.

Here is the completed code for the example. For now, you should ignore syntax details. These will be discussed further in the next lesson on syntax and data types.

```
float celcius(float fahrenheit) {  
    return (fahrenheit - 32) * 5 / 9;  
}
```

- How about you try one yourself now.
- We will be translating from Yards to Killometers.
- You can run the code that you write in https://www.onlinegdb.com/online_c++_compiler.
- Boilerplate code is provided for you on the next slide.
 - I recommend that you copy and paste the code on the following slide, then fill out the line that is marked “<**WRITE YOUR CODE HERE**>”
 - I will explain this code further in the next lesson.
- **HINTS**
 - I don't remember how to convert yards to killometers, and I bet you don't either. **LOOK IT UP**, Googling is a very important part of programming.
 - **DO NOT FEEL BAD IF YOU DO NOT GET IT.** I'm putting this example out here to get your brains working, but there's a lot I haven't taught you yet.
 - Once you've given it an honest attempt, move on to lesson 2.

```
#include <iostream>
```

```
// This is a function that should convert yards to killometers.
```

```
float killometers(float yards) {  
    return <WRITE YOUR CODE HERE>;  
}
```

```
// This is the entrypoint of our application, we start executing from here.
```

```
int main() {  
    // This line calls the function above and stores the output in "result".  
    float result = killometers(1000);  
    std::cout << "1000 yards = " << result << " killometers" << std::endl;  
    std::cout <<  
}
```