C++ for FIRST Robots

Patrick Frampton

What we will cover (ideally)

- Where to find detailed information
  - (hint: the manual)
- C++ on and off the robot in a very small nutshell
- Very basic setup of robot program
  - Mobility first

So many possibilities, so little time

- Read the manual
- Location: http://first.wpi.edu/FRC/frcupdates.html

C++ IN A VERY SMALL NUTSHELL

What is C++?

- An object-oriented language based on C, once known as “C with Classes”
- Developed by Bjarne Stroustrup at Bell Labs, from 1979 to 1983, and ratified by ISO in 1998.

Why use C++?

- C++ is very powerful and flexible
- Classes allow programmer to reuse code
- All C++ source code is plain text
- Programming only requires a compiler and a text editor (we'll use WindRiver for both)

C++ Basics

- Every program starts and ends in the main function (like C).
- There are 3 basic types of data:
  - Integers (int)
  - Real numbers (float)
  - Characters (char) (not really needed for robots)

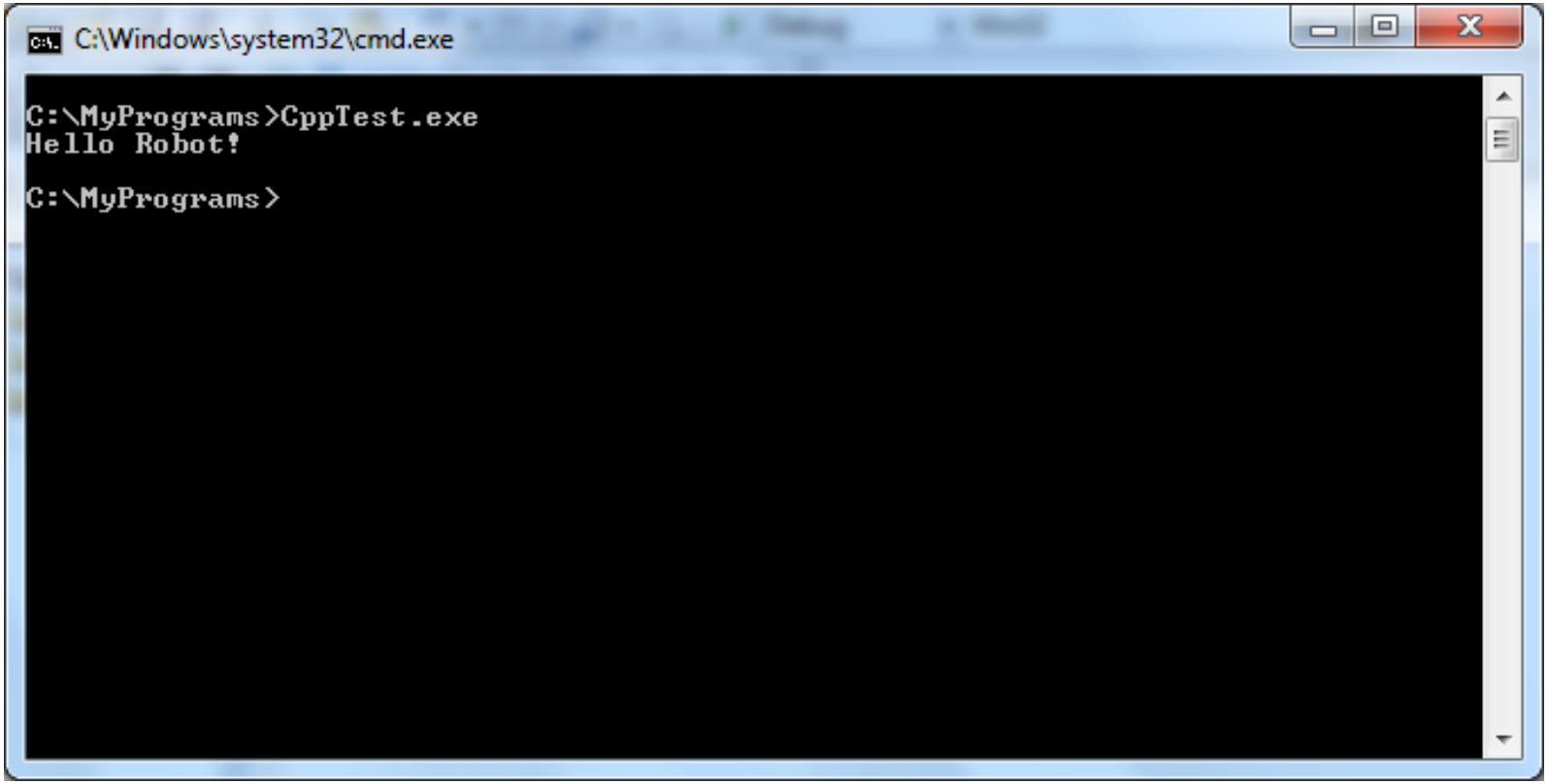
The Simplest C++ Program

```
int main() {
      return 0;
}
```

A C++ Program That Does Something

```
#include <iostream>
using namespace std;

int main() {
     cout << "Hello Robot!\n";
}
```

A C++ Program That Does Something (contd.)



C++ Datatypes

```
int main() {
	int   gear_teeth_count; // 32-bit integer
	float sensor_volts = 0.0; // 32-bit real number
	bool  target_in_range = false; // true/false

	// ... Program continues
}
```

C++ Assignment

```
gear_teeth_count = 42;

total_voltage = voltage_1 + voltage_2;

number_of_shots += 4;

power *= 3;

should_shoot = true;

count++;
```

C++ Conditionals

```
if (count == 13) {
        cout << "count is 13.\n";
}

if (count == 14) {
        cout << "count is 14.\n";
}
else {
        cout << "count is NOT 14.\n";
}

if (val > 30) {
        cout << "val is unusually high.\n";
} else if (val > 10) {
        cout << "val is higher than normal.\n";
} else {
        cout << "all is well.\n";
}
```

C++ Conditionals (cont.)

```
if (num_shots > 0 && target_locked == true) {
        cout << "FIRE\n";
}

if (limit_switch1 == true || limit_switch2 == true) {
        motor_1_out = 0.0;
}
```

C++ Functions

```
void shoot() {
	cout << "BANG!\n";
}

float get_joystick_y() {
	return 1.0 - get_joystick_raw()/2.0;
}

const float PI = 3.1415926; // constant

float degrees_to_radians(float deg) {
	return deg * PI / 180.0;
}
```

C++ Classes

- Interface in .h file
  - RobotSensor.h, RobotMotor.h, etc.
- Implementation in .cpp file
  - RobotSensor.cpp, RobotMotor.cpp, etc.

Example: RobotSensorPackage.h

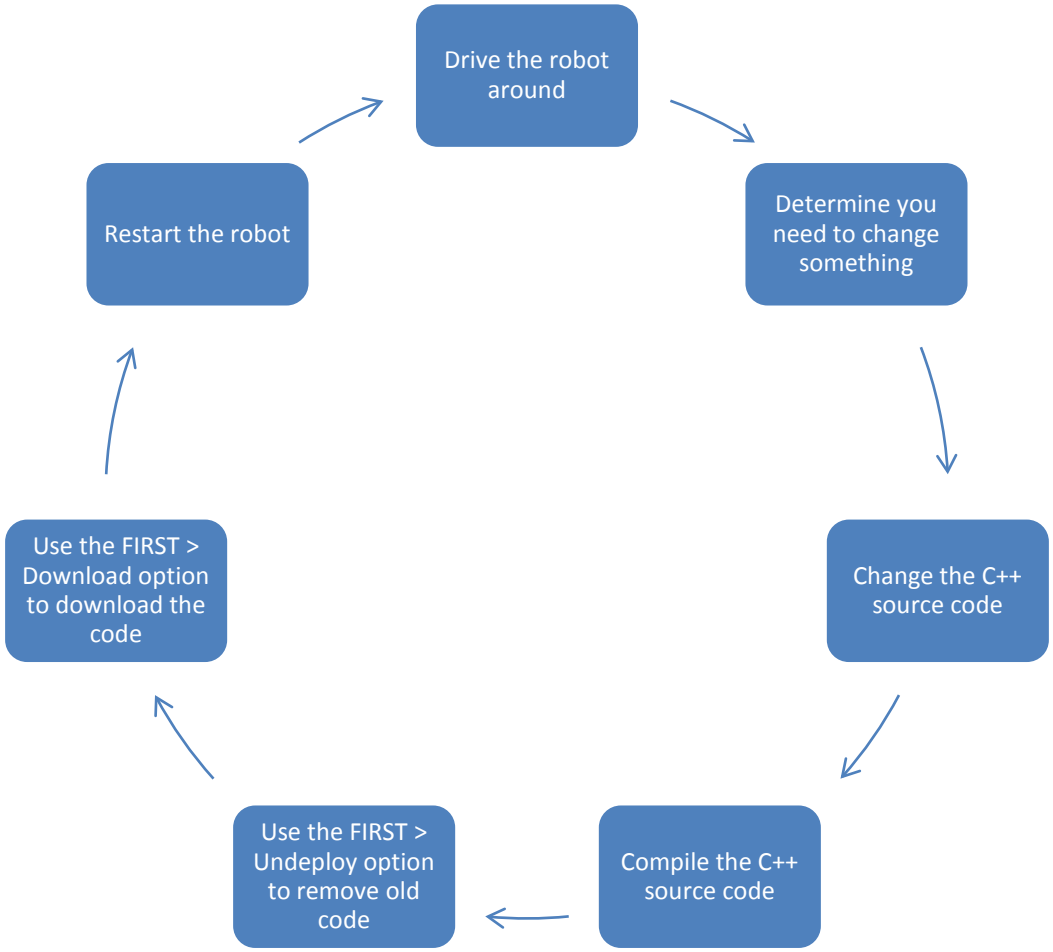
```
class RobotSensorPackage {
public:
	// This stuff is available for other
	// code to see and use
	RobotSensorPackage(); // Function to set up class
	~RobotSensorPackage(); // Function to take down class

	float SpeedInFtPerSecond() const;

	float TemperatureInFht() const;

	void ShutOff();
private:
	// This stuff can be used ONLY by
	// code for this class.
	float speed_in_meters_per_second;
	float temp_c;

	float CelsiusToFht(float c) const;
}; // <-- NOTE THE SEMICOLON
```

Example: RobotSensorPackage.cpp

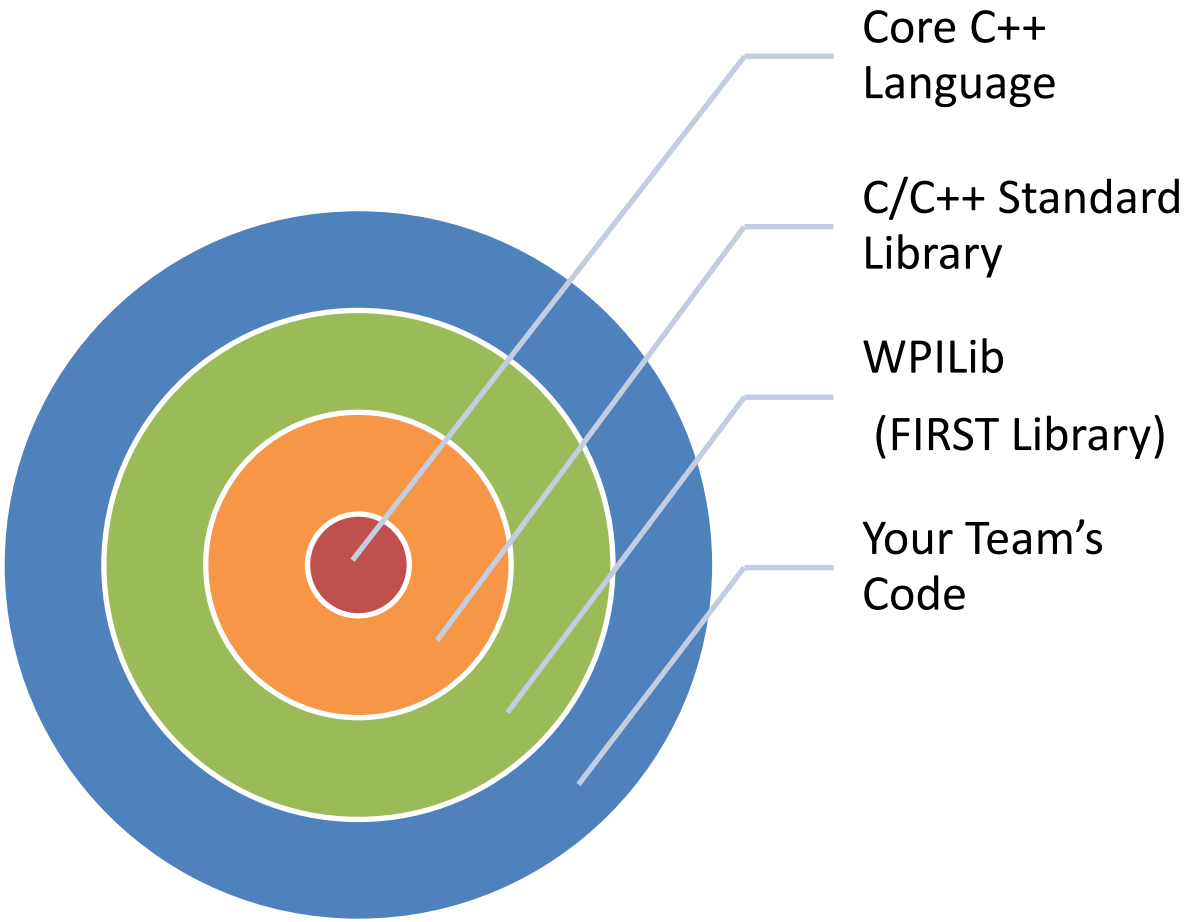
```
RobotSensorPackage::RobotSensorPackage()
{       // Set up variables
        speed_in_meters_per_second = 0;
        temp_c = 14.0;
}

float RobotSensorPackage::SpeedInFtPerSecond() const
{
        return speed_in_meters_per_second * 3.2808;
}

float RobotSensorPackage::TemperatureInFht() const
{
        return CelsiusToFht(temp_c);
}

float RobotSensorPackage::CelsiusToFht(float c) const
{
        return (9.0 / 5.0) * c + 32.0;
}
```

C++ ON THE ROBOT

How to Put Code on the Robot



The Onion of Robot Software



WPILib

- WPI = Worchester Polytechnic Institute
- WPILib = Helpful library to make FIRST programming easier
- WPILib
  - Uses a rich set of classes to represent control of the robot.
  - Is fully open and documented
  - Is **extensible**

Your Robot Class

```
#include "WPILib.h"

class QuantumBuzzards9931RobotClassOfAwesomeness : public IterativeRobot
{
private:
        RobotDrive m_robotDrive; // The drive train of the robot

        Joystick m_rightStick;    // The right joystick on our driver station
        Joystick m_leftStick;     // The left joystick on our driver station

        // ...
```

Your Robot Class (cont.)

```
        Joystick m_leftStick;     // The left joystick on our driver station

public:
        BuiltinDefaultCode()
                : m_robotDrive(1,2,3,4),  m_leftStick(1), m_rightStick(2)
        {
        }
```

Your Robot Class (cont.)

```
/******************** Init Routines ************************/

void RobotInit(void) {
	// Actions which would be performed once (and only once)
	// upon initialization of the robot would be put here.
}

void DisabledInit(void) {
	// This runs when the robot enters disabled mode
}

void AutonomousInit(void) {
	// This runs when the robot enters autonomous mode
}

void TeleopInit(void) {
	// This runs when the robot enters teleoperated mode
}
```

Your Robot Class (cont.)

```
/****************** Periodic Routines ********************/

void DisabledPeriodic(void)  {
        // This runs in a loop during the disabled period
}

void AutonomousPeriodic(void) {
        // This runs in a loop during the autonomous period
}


void TeleopPeriodic(void) {
        // This runs in a loop during the teleoperated period
        // Drive the robot using a simple "Arcade" drive mode.
        m_robotDrive.ArcadeDrive(
                m_rightStick->GetX() ,
                m_rightStick->GetY() );

        // We could also do a "Tank" drive mode like this:
m_robotDrive.TankDrive( m_leftStick->GetY() , m_rightStick->GetY() );

}
```

Your Robot Class (cont.)

```
}; // REMEMBER THIS SEMICOLON.

// Transform into a main() function (don't worry about this)
START_ROBOT_CLASS(QuantumBuzzards9931RobotClassOfAwesomeness);
```

That's It

- This has been a very shallow introduction to C++ and its use on the robot.
- You still need to read the manual.
- If you have questions, my email is framptonp@comcast.net
  - Check the manual first, though.