Warlocks 1507

FRC Programming in C++

Lesson 6

Lockport Robotics
November 2023

FIRST

Important Concepts

Variables	Storage	x = 10;
Conditionals	Decisions	if(x == 10)
Loops	Repeated instructions	for(i=0; i<10; i++)
Functions	Small Tasks	x = GetEncoderValue();
Classes	Objects	Motor leftMotor; Motor rightMotor;
Class Members	variables and functions in a class	leftMotor->Set(0.7);

Warlocks 1507

Functions

Example #1

```
#include <iostream>

//Function Prototypes
int Addition( int a, int b );

int main()
{
    int sum = Addition( 2, 3 );

    std::cout << "Sum is: " << sum << std::endl;
    return 0;
}


//Function Addition
int Addition( int a, int b )
{
    int c;
    c = a + b;
    return(c);
}
```

FIRST

Functions and Headers

Warlocks 1507

Example #3

File: main.cpp

```
#include <iostream>
#include "OurMath.h"        //Our new Header File
using namespace std;


//Global Variables
int a = 10;
int b = 5;
int c;

int main()
{
    int x = 7;
    int y = 2;
    int z = Addition( 12,4 );

    cout << "Result #1 is: " << z << endl;
    cout << "Result #2 is: " << Subtraction(a,x) << endl;
    cout << "Result #3 is: " << Addition(1,x) << endl;

    c = 3 + b + Subtraction(b,y);
    cout << "Result #4 is: " << c << endl;

    return 0;
}
```

File: OurMath.h

```
#ifndef OURMATH_H
#define OURMATH_H


//Function Prototypes
int Addition( int a, int b );
int Subtraction( int a, int b );



#endif // OURMATH_H
```

File: OurMath.cpp

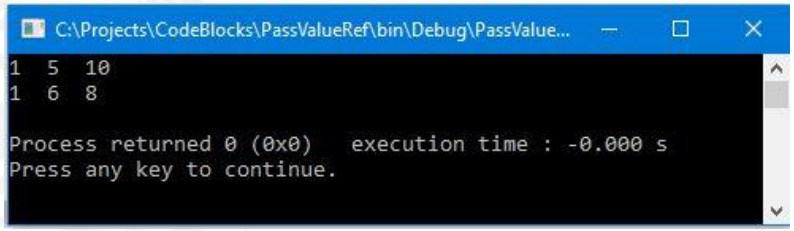
```
//Function Addition
int Addition( int a, int b )
{
    int c;
    c = a + b;
    return(c);
}

//Function Subtraction
int Subtraction( int a, int b )
{
    return(a-b);
}
```

FIRST

Pass-by-Value vs Pass-by-Reference

- Pass-by-Value : Make a copy of variable - is NOT modified outside of function
- Pass-by-Reference: Use actual variable - IS modified outside of function

```
#include <iostream>
using namespace std;
int PassTest( int p_value, int& p_ref );

int main()
{
    int a = 1;
    int b = 5;
    int c = 10;

    cout << a << "  " << b << "  " << c <<endl;
    c = PassTest(a,b);
    cout << a << "  " << b << "  " << c <<endl;
    return 0;
}

int PassTest( int p_value, int& p_ref )
{
    p_value += 1;
    p_ref   += 1;
    return( p_value + p_ref );
}
```



CLASSES

- Object Orientated Programming (OOP) – C++ solves this with “Classes”
- Classes can hold Data AND Functions
- Instantiation (creation) of a Class is an ***object***
- Objects are variables of type Class
- Variables and Functions defined in the class are ***members***
- Members have ***Access Specifiers*** to protect data from illegal access

Classes

- Declaration Format:

```
class ClassName{
private:
    //Members that only the Class can access

protected:
    //Members that only the Class and it's derived Class can access

public:
    //Members anyone can accss

}
```

- Instantiation:

```
ClassName objectName;        //Looks like a variable!
```

Warlocks 1507

Classes – Example #1

```
class Rectangle {
private:
    int m_width;
    int m_height;
public:
    void SetParam (int x, int y);
    int  GetArea  ( void );
};

void Rectangle::SetParam(int x, int y){
    m_width = x;
    m_height = y;
}

void Rectangle::GetArea(void){
    return( m_width * m_height );
}

void main () {
    Rectangle rect;
    rect.SetParam(5,6);
    cout << "Area: " << rect.GetArea();
    return;
}
```

FIRST

Constructors/Destructors

- Classes have Constructors and Destructors
- Constructor
  - Called immediately when object is instantiated (and ONLY when first instantiated!)
  - Initialize variables/members/stuff
  - Allocate dynamic memory
- Destructor
  - Called immediately when object is destroyed (out-of-scope or manually deleted)
  - De-allocate dynamic memory
  - Any clean up tasks

Warlocks 1507

Classes – Example #2

```
class Rectangle {
private:
    int m_width;
    int m_height;
public:
    Rectangle(void);        //Constructor
    ~Rectangle(void);       //Destructor
    void SetParam (int x, int y);
    int  GetArea  ( void );
};


//**** CONSTRUCTOR ****
void Rectangle::Rectangle(void){
    m_width  = 0;
    m_height = 0;
}

//**** DESTRUCTOR ****
void Rectangle::~Rectangle(void){
}


void main () {
    Rectangle rect;
    ...
    return;
}
```

FIRST

Warlocks 1507

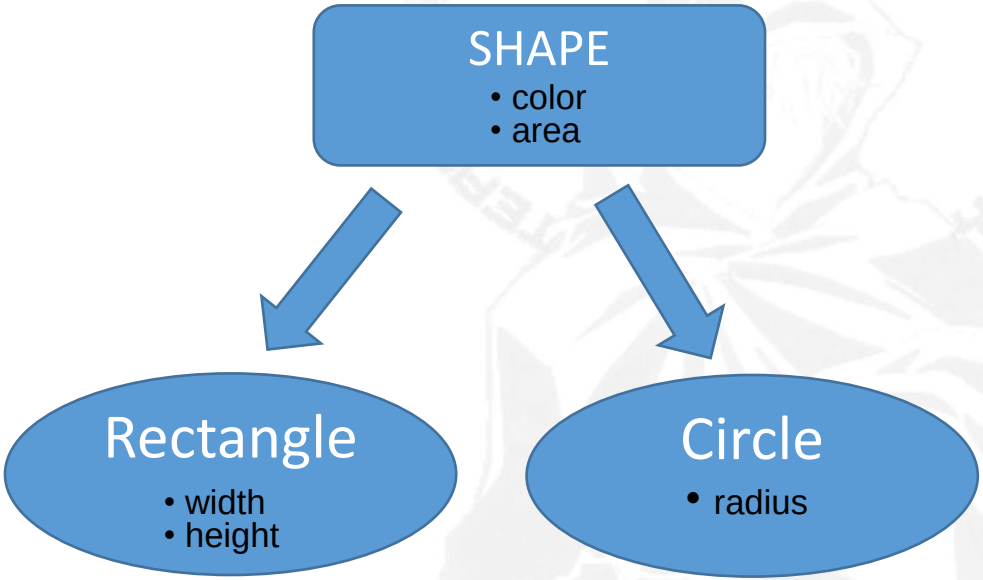
Classes - Example #3

```
class Rectangle {
private:
    int m_width;
    int m_height;
public:
    Rectangle(int x, int y);        //Constructor
    ~Rectangle(void);               //Destructor
//  void SetParam (int x, int y);
    int  GetArea  ( void );
};


//**** CONSTRUCTOR ****
void Rectangle::Rectangle(int x, int y){
    m_width  = x;
    m_height = y;
}

//**** DESTRUCTOR ****
void Rectangle::~Rectangle(void){
}


void main () {
    Rectangle rect(1,3);    //New init syntax for object
    ...
    return;
}
```

FIRST

Inheritance

- Inheritance: Sharing members from parent class with child class



Base Class

Derived Class