C++ Crash Course

Module 6: Classes

Human Computer Interaction
IOWA STATE UNIVERSITY
OF SCIENCE AND TECHNOLOGY

Classes

- Intro to classes
- Defining classes
- public, private, and protected
- Constructors
- Destructors
- Creating objects
- Member functions and variables
- Friendship
- Inheritance
- Adding files to your project

Intro to classes

- Classes are data structures which can contain both variables and functions.
- Signified by the class keyword.
  - Essentially the same as a struct in C.
- An object is an instance of a class.
  - Object-oriented programming

Defining classes

```
class className
{
    int variable1;
    double variable2;
    void someFunction(void);
};
```

public, private, and protected

- Public
  - Accessible anywhere
- Private (default)
  - Only accessible to members of the same class or declared friends
- Protected
  - Like private, but also accessible to derived classes

```
class someClass{
  public:
        int publicVariable;
  private:
        double privateFunction(void);
  protected:
        int otherVariable;
};
```

Constructors

- Automatically called when an object is created.
- Must be the same name as the class.
- Cannot have a return type.
- Can be overloaded like any function.

```
someClass( );                 //Default constructor
someClass(int x, int y);      //Specific constructor
```

Destructors

- Automatically called when an object goes out of scope or is explicitly deleted.
- Also must have the same name as the class but preceded with a tilde (~).
- Also cannot have a return type.

```
~someClass( )
{
   delete [ ] pArray;
}
```

Creating Objects

- Once a class is defined it can be treated like a variable type to create an object.

someClass myObject;

- …or with a specific constructor…

someClass myObject(1, 5);

Member functions and variables

- Can be accessed, depending on access, with a period.

```
myObject.x = 5;
myObject.someFunction( );
```

- Within the class itself, you don't need the "myObject."

Friendship

- When you want data to remain private but still allow access to certain non-member functions and classes, you can declare them as friend.

```
void addToFriend(int x);


class someClass{
    int value;
    friend void addToFriend(int x);
};
```

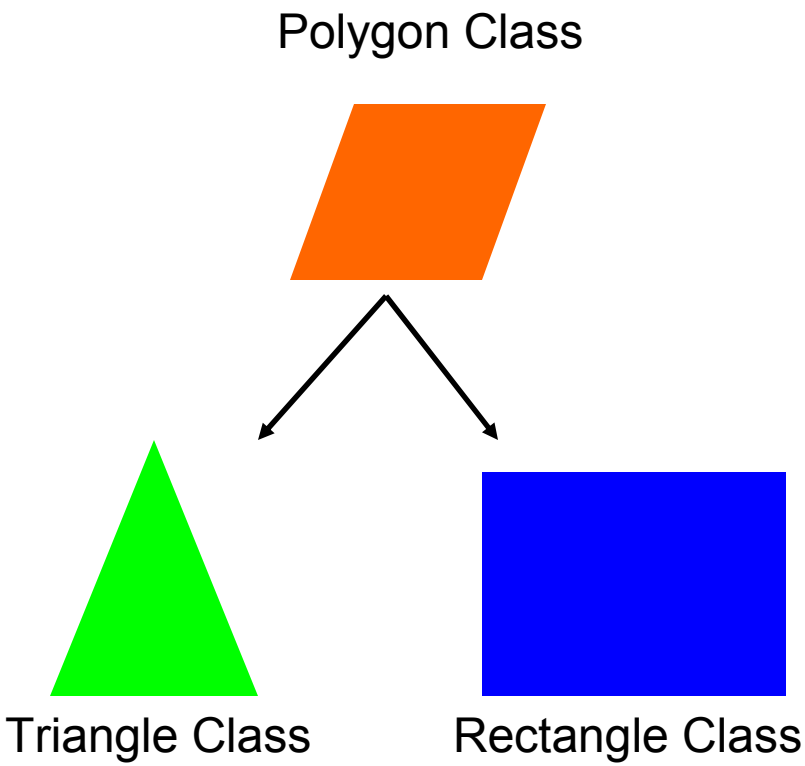
```
class friendClass{
    int value;
    int sumClassValues(int x);
};

class someClass{
    int value;
    friend class friendClass;
};
```

Inheritance

- It is often useful to create classes that are derived from other classes.
- Base class :: Derived class

  Parent :: Child
- Derived classes inherit everything but
  - Constructors/Destructors
  - Declared friends
  - Assignment operators



Inheritance

```
class Polygon{
    public:
        double area, perimeter;
    protected:
        string colorName;
};

class Triangle: public Polygon{
    double side1, side2, side3;
};

class Rectangle: public Polygon{
    double length1, length2;
};
```

Adding files to your project

- With all this expanded capability, it would be very confusing to have all this code in a single .cpp file.
- We can create our own header files and include them like any other.
- #include < > vs. #include " "

Adding files to your project

main.cpp

```
#include “someClass.h”

int main( )
{
    someClass myObject;
    myObject.someFunction( );
    return 0;
}
```

someClass.h

```
class someClass {
   someClass( );
   ~someClass( );

   void someFunction( );
};
```

someClass.cpp

```
#include “someClass.h”

someClass::someClass( )
{
   //Constructor stuff
}

someClass::~someClass( )
{
   //Destructor stuff
}

someClass::someFunction( )
{
   //Function stuff
}
```

The End

- See main.cpp, histogram.h, and histogram.cpp for sample code of how a basic class works.
- The surface has barely been scratched, but you should have the tools to get started.