Classes

“Absolute C++”
Chapter 6

Introduction to Classes

- Last lecture, we talked about streams and used functions called member functions.
- These functions operate on the variable they are “attached to”

```cpp
ifstream inFile;
inFile.open("input.dat");
if (inFile.fail()) {  // fail is a member function
    cout << "Couldn't open file" << endl;
}
```

- inFile is actually an instance of the ifstream class.
- But what is a class?

Classes

- What is a class?
  - A very simple definition: “A class is a user-defined type”
  - Are all user-defined types classes?
    - No.
    - C++ supports the C notion of a “struct”
    - A struct allows programmers to define their own data type structures
      - Similar to RECORDs in Pascal
  - Are all classes user-defined types?
    - Yes
    - There are no “built in classes” in C++
    - There are provided standard class libraries
      - list
      - string
  - OK, that’s great. But, what is a class?

Classes (cont)

- A class is a traditional data structure with a set of functions.
- Let’s start with a simple C style structure definition

```cpp
typedef struct {
    string name;
    string instructor;
    int numStudents;
} Course;
```

- Once defined I could use this “user defined” data type anywhere in my code:

```cpp
int main() {
    Course cs213;
    cs213.name = "COM S 213";
    cs213.instructor = "DiNapoli";
    cs213.numStudents = 45;
}
```

Classes (cont)

- Now, where do these functions fit in?
  - The functions (called member functions) are tied to the data structure
  - Any “field” of the data structure may be accessed by any member function as if it were in a global scope.
  - Let’s take a look at this before we go any further...

```cpp
class Course {
    public:
    // Define member functions
    int getStudentCount() { return numStudents; }
    // Define member variables
    string name;
    string instructor;
    int numStudents;
};
```

Demonstration #1

- Very Simple Class
Why bother with simple functions like `getStudentCount()`?

- It's a bad idea to directly access member variables
- Circumvent error checking, easy to screw up data.

Can't I just use the member variables directly anyway?

class Course
{
    public: // These can be seen outside the class
    // Define member functions
    int getStudentCount() { return numStudents; }

    private: // These can be seen inside the class only
    // Define member variables
    string name;
    string instructor;
    int numStudents;
};

OK, so how do I access private data outside of the class?

- You don't, that's the whole idea!
- You can use get/set functions (public) to return the values for you

class Course
{
    public: // These can be seen outside the class
    // Define member functions
    string getCourseName() { return name; }
    string getInstructor() { return instructor; }
    int getStudentCount() { return numStudents; }
    void setCourseName(string theName) { name = theName; }
    void setInstructor(string theInstructor) { instructor = theInstructor; }
    void setStudentCount(int count) { numStudents = count; }

    private: // These can be seen inside the class only
    ...

string Course::getCourseName()
{ return name; }

int Course::getStudentCount()
{ return numStudents; }

Note the use of Course:: to specify the class in question
Note how I'm using member variables as if they were some sort of global variable
Classes: More on Public vs. Private

- The public and private labels can appear as many times as you want them to in a class definition.

```cpp
class Course
{
    public: // These can be seen outside the class
        // Getter functions
        string getCourseName();
        string getInstructor();
        int getStudentCount();
    public:
        // Getter functions
        void setCourseName(string theName);
        void setInstructor(string theInstructor);
        void setStudentCount(int count);
    private: // These can be seen inside the class only
        // Member variables
        string name, instructor;
        int count;
};
```

Classes: More on Public vs. Private

- Member functions can be private as well.

```cpp
class Course
{
    public: // These can be seen outside the class
        // Getter and Setter functions
        string getCourseName();
        string getInstructor();
        int getStudentCount();
        void setCourseName(string theName);
        void setInstructor(string theInstructor);
        void setStudentCount(int count);
    private: // These can be seen inside the class only
        // Private member functions
        bool validateStudentCount(int count);
};
```

Classes: More on Public vs. Private

- You can still have public member variables if no public or private label is specified, private is assumed.

```cpp
class Course
{
    bool validateStudentCount(int count); // implicit private
    public:
        bool isFull; // publicly accessible member variable
        // Getter and Setter functions
        string getCourseName();
        string getInstructor();
        int getStudentCount();
        void setCourseName(string theName);
        void setInstructor(string theInstructor);
        void setStudentCount(int count);
};
```

Where should we Define Member Functions?

- How do you know when to define a member function in the class definition vs defining it outside of the class definition?
- There is a simple technical explanation.
- I'm not going to tell you yet :-(
- A good rule of thumb is:
  - If the definition is simple (one line of code) you should define it in the class definition.
  - Getter/Setter functions are prime examples.
  - Otherwise, define outside of the class definition, usually in a separate file.

What Files Should These Definitions Go In?

```cpp
// Course.h -- Header file for Course class
class Course
{
    public: // These can be seen outside the class
        // Define member functions
        string getCourseName();
        string getInstructor();
        int getStudentCount();
        void setCourseName(string theName);
        void setInstructor(string theInstructor);
        void setStudentCount(int count);
    private: // These can be seen inside the class only
        string name, instructor;
        int count;
};
```
What Files Should These Definitions Go In?

// Course.cpp -- Definition file for Course class
#include "Course.h"
string Course::getCourseName()
{
    return name;
}

string Course::getInstructor()
{
    return instructor;
}

string Course::getStudentCount()
{
    return count;
}
// etc., etc.