Lecture 13

C++ Libraries and Classes

Instructor: Ashley Gannon

ISC3313 Fall 2021

Motivation

Motivation

- In Computational Science, you will often come into contact with problems that can be solved with the algorithms covered in this course. So far we have discussed and implemented root finding algorithms, we still have to cover algorithms for
  - Optimization
  - Integration and differentiation
  - Ordinary Differential Equations (ODEs)

Motivation

- In Computational Science, you will often come into contact with problems that can be solved with the algorithms covered in this course. So far we have discussed and implemented root finding algorithms, we still have to cover algorithms for
  - Optimization
  - Integration and differentiation
  - Ordinary Differential Equations (ODEs)
- Oftentimes you will use several of these algorithms together to solve one problem.

Motivation

- In Computational Science, you will often come into contact with problems that can be solved with the algorithms covered in this course. So far we have discussed and implemented root finding algorithms, we still have to cover algorithms for
  - Optimization
  - Integration and differentiation
  - Ordinary Differential Equations (ODEs)
- Oftentimes you will use several of these algorithms together to solve one problem.
- If we continue to write the algorithms the way we have been, we will have to copy and paste our functions over and over again into new .cpp files so that our `main()` can call them.

Motivation

- In Computational Science, you will often come into contact with problems that can be solved with the algorithms covered in this course. So far we have discussed and implemented root finding algorithms, we still have to cover algorithms for
  - Optimization
  - Integration and differentiation
  - Ordinary Differential Equations (ODEs)
- Oftentimes you will use several of these algorithms together to solve one problem.
- If we continue to write the algorithms the way we have been, we will have to copy and paste our functions over and over again into new .cpp files so that our `main()` can call them.
- Wouldn't it be nice to keep all our algorithms in one place, that we can access from the `main()`, without having to copy and paste them every time?

Libraries

Libraries

A **library** is a package of code that is meant to be reused by many programs. Typically, a C++ library comes in two pieces:

1. A header file that defines the functionality the library is offering to the programs using it.
2. A precompiled .cpp file that contains the implementation of that functionality.

We've used a few libraries already:

- `stdio`
- `iostream`
- `iomanip`
- `cmath`

There are two types of libraries: **static libraries** and **dynamic libraries**. We will likely only cover static libraries in this course.

Libraries

- A static library consists of routines that are compiled and linked directly into your program.
  - Namespaces
  - Classes
  - Functions
- When you compile a program that uses a static library, all the functionality of the static library that your program uses becomes part of your executable.
  - On Windows, static libraries typically have a `.lib` extension
  - On linux, static libraries typically have an `.a` extension.

  NOTE: We will be using visual studio to write our library, so everyone should end up with a `.lib` extension.

Example

The purpose of this course is to introduce you all to standard problems in computational science, as well as the basics of C++ – with the intention of facilitating the student's implementation of algorithms. So I won't go into the gritty details of this.

SciProgLib.h | Main.cpp

Miscellaneous Files - No Configurations | (Global Scope)

```
1   #include <iostream>
2   #include <cmath>
3
4    using namespace std;
5
6   namespace SciProgLib
7   {
8       class RootFinding
9       {
10      public:
11          static double Bisect(double a, double b, double (*f)(double m));
12      };
13  }
```

Example

The purpose of this course is to introduce you all to standard problems in computational science, as well as the basics of C++ – with the intention of facilitating your implementation of the algorithms we cover. So I won't go into the gritty details of this. If you would like more details, come see me or Liam during office hours.

SciProgLib.h | Main.cpp

Miscellaneous Files - No Configurations | (Global Scope)

```
1   #include <iostream>
2   #include <cmath>
3
4    using namespace std;
5
6   namespace SciProgLib
7   {
8       class RootFinding
9       {
10      public:
11          static double Bisect(double a, double b, double (*f)(double m));
12      };
13  }
```

Example

Say we want our friend “Bisect" to help with our homework. We’ll need to know how to get over to his place. We know that he lives in the “SciProgLib” neighborhood, on the “SciProgLib” street in the “RootFinding” house.

SciProgLib.h | Main.cpp

Miscellaneous Files - No Configurations | (Global Scope)

```
1   #include <iostream>
2   #include <cmath>
3
4    using namespace std;
5
6   namespace SciProgLib
7   {
8       class RootFinding
9       {
10      public:
11          static double Bisect(double a, double b, double (*f)(double m));
12      };
13  }
```

Example

The neighborhood would be the header file,
the street would be the namespace,
and the house would be the class,
and right now, think of Bisect as a person who lives in the house alone.

SciProgLib.h | Main.cpp

Miscellaneous Files - No Configurations | (Global Scope)

```
1   #include <iostream>
2   #include <cmath>
3
4    using namespace std;
5
6   namespace SciProgLib
7   {
8       class RootFinding
9       {
10      public:
11          static double Bisect(double a, double b, double (*f)(double m));
12      };
13  }
```

Example

A class is a data structure that can contain a data member or a function member. In this example, our class `RootFinding` contains the function member `Bisect`.

SciProgLib.h | Main.cpp

Miscellaneous Files - No Configurations | (Global Scope)

```
1   #include <iostream>
2   #include <cmath>
3
4    using namespace std;
5
6   namespace SciProgLib
7   {
8       class RootFinding
9       {
10      public:
11          static double Bisect(double a, double b, double (*f)(double m));
12      };
13  }
```

Example

Classes also contain something called an *access specifier*, which can be `public`, `private`, or `protected`. These specifiers modify the access rights for the members that follow them. In this case, thinking back to our simple example, because our class is `public` "Bisect" helps everyone with homework. But if he was `private`, he would only help people who live in the same house as him.

SciProgLib.h | Main.cpp

Miscellaneous Files - No Configurations | (Global Scope)

```
1   #include <iostream>
2   #include <cmath>
3
4    using namespace std;
5
6   namespace SciProgLib
7   {
8       class RootFinding
9       {
10      public:
11          static double Bisect(double a, double b, double (*f)(double m));
12      };
13  }
```

Example

SciProgLib.cpp | SciProgLib.h | Main.cpp

Miscellaneous Files - No Configurations | SciProgLib::Root

```cpp
#include "SciProgLib.h"

namespace SciProgLib
{
    double RootFinding::Bisect(double a, double b, double (*f)(double m))
    {
        //Check that there is a root in the chosen domain.
        if (f(a) * f(b) >= 0)
        {
            cout << "This interval has no roots." << endl;
            //exit if no root is found
            return 0;
        }

        double tol;
        double error = 100.0;
        double c_old = 0;
        double c_new;
        cout << "What is your tolerance?";
        cin >> tol;
        while (error > tol)
        {
            c_new = (a + b) / 2;
            if (f(a) * f(c_new) < 0)
            {
                b = c_new;
            }
            else
            {
                a = c_new;
            }
            error = abs((c_new - c_old) / c_new);
            c_old = c_new;
        }

        return c_new;
    }
}
```

Now we can use SciProgLib in another .cpp file. We `#include` the header file so that the compiler pulls in the declaration of the member function `Bisect`. All the compiler needs to know is that `RootFinding` is a class that has a public member function called `Bisect`.

Example

Now when we include our header file `"SciProgLib."` in the same .cpp file as our `main()`, we are able to call the `Bisect` function.

SciProgLib.cpp | SciProgLib.h | Main.cpp

Miscellaneous Files - No Configurations | (Global Scop

```cpp
#include "SciProgLib.h"
#include <iomanip>

double f(double m)
{
    //Define all your variables
    double v = 36.0;
    double g = 9.81;
    double cd = 0.25;
    double t = 4;
    //return the function value f(m)
    return sqrt(g * m / cd) * tanh(sqrt(g * cd / m) * t) - v;
}

int main()
{
    double root = SciProgLib::RootFinding::Bisect(140, 150, f);

    cout << "The root is " << fixed << setprecision(6) << root << endl;

    return 1;
}
```

Example

In this example, note that we are also passing in a pointer of the function f that we use in the `Bisect` function. This is useful in the sense that we don't have to define the function in the same space as `Bisect`, and recompile our library every time we change it.

SciProgLib.cpp SciProgLib.h Main.cpp

Miscellaneous Files - No Configurations (Global Scop

```cpp
#include "SciProgLib.h"
#include <iomanip>

double f(double m)
{
    //Define all your variables
    double v = 36.0;
    double g = 9.81;
    double cd = 0.25;
    double t = 4;
    //return the function value f(m)
    return sqrt(g * m / cd) * tanh(sqrt(g * cd / m) * t) - v;
}

int main()
{
    double root = SciProgLib::RootFinding::Bisect(140, 150, f);

    cout << "The root is " << fixed << setprecision(6) << root << endl;

    return 1;
}
```

Mini Assignment

We will take the remainder of the class period (and any extra time you need) to create a new project in visual studio that builds this library.

Submit your project as a **.zip** file to canvas. If you are having trouble generating a .zip file, please see these instructions:
https://www.hellotech.com/guide/for/how-to-zip-a-file-mac-windows-pc

Due by the start of next class.