Extending C++ Libraries

Using a C++ Library

In the class handouts we used a class Data which basically inputs and outputs two arrays of doubles. We use the Data class as a base class for the Average class which did some math with the input data. As you might suspect the Data class could well be quite useful in a number of programs involving data manipulation (you first have to get the data!). Thus, it would be convenient have the data class as a library.

To use a C++ library you must do two things: **include the header file** for the library and **correctly link the library** at compilation. The header file - data.h- for the class Data library looks as follows and is basically the declaration of the class:

```
class Data {
  protected:
    double *x,*y;
    int N;
  public:
    Data(int i=1);
    ~Data();
    void getdata();
    void showdata();
};
```

The four member functions are separately compiled and archived into a library called **libdata.a** and placed in a personal directory /home/dje/lib. The command to compile a program using the Data class library could look like this:

```
CC -L "/home/dje/lib" prog.C -ldata
```

A Least-Squared-Fit Library

Another popular library of use to scientists & engineers would be one in which a set of data points are "best-fit" to a straight line. Here is the header file - lsf.h - which declares a class LSF which generates the slope and intercept of the "best" fit line:

```
class LSF : public Data {
 protected:
   double a,b,delta;
   double sigma(double* r);
   double sigma(double* r, double* s);
 public:
   LSF(int i);
   void make_a();
   void make_b();
   void display();
};
```

Your are given the lsf.h file and the liblsf.a library and told to call the member functions make_a() and make_b() to calculate intercept and slope. NOTE: class LSF is derived from class Data so compile and load both libraries:

```
CC -L "/home/dje/lib" -llsf -ldata -lm
```

On the following page is a program that uses the LSF Class to fit a least-squares line to data consisting of temperature and position of an iron bar heated at one end.

```
#include <iostream.h>
#include <math.h>
#include "data.h"
#include "lsf.h"

main () {
  LSF temp(9);
  temp.getdata();
  temp.make_a();
  temp.make_b();
  temp.display(); }
```

For input distances and temperatures:

```
1.0 15.6
2.0 17.5
3.0 36.6
4.0 43.8
5.0 58.2
6.0 61.6
7.0 64.2
8.0 70.4
9.0 98.8
```

the output from the above code is:

```
Slope is 9.40833
Y-intercept is 4.81389
```

Definitions of the LSF Class member functions

```
LSF::LSF(int i) : Data(i) {};
double LSF::sigma(double* r) {
    double sum=0.0;
    for(int i=0;i<N;++i)
      sum += r[i];
    return(sum);
}
double LSF::sigma(double* r,double* s) {
    double sum=0.0;
    for(int i=0;i<N;++i)
      sum += r[i]*s[i];
    return(sum);
}
void LSF::make_a() {
    delta=N*sigma(x,x)-pow(sigma(x),2.0);
    a=(1.0/delta)*(sigma(x,x)*sigma(y)-sigma(x)*sigma(x,y));
}
void LSF::make_b() {
    b=(1.0/delta)*(N*sigma(x,y)-sigma(x)*sigma(y));
}
void LSF::display() {
    cout << "Slope is " << b << endl;
    cout << "Y-intercept is " << a << endl;
}
```

An Improved Least-Squares Fit

The lsf library we have been provided with is quite simple. In fact, it is missing something we need: the uncertainties in the slope and intercept. We can't rewrite the LSF class, we don't even have the source code shown on the previous page: all we have is the pre-compiled library liblsf.a.

The power of C++ inheritance will allow to still use the LSF library but extend its capabilities to provide the uncertainty values. On the next page we show a program where we define a class LSF_err derived class LSF which adds the new capabilities.

```
#include <iostream.h>
#include <math.h>
#include "data.h"
#include "lsf.h"
class LSF_err : public LSF {
    double siga,sigb,s2;
  public:
    LSF_err(int i):LSF(i) {};
    void make_s2();
    void make_siga();
    void make_sigb();
    void display();
};

void LSF_err::make_s2() {
    double t;
    t=sigma(y,y)+N*pow(a,2.0)+pow(b,2.0)*sigma(x,x)-
      2.0*(a*sigma(y)-a*b*sigma(x)+b*sigma(x,y));
    s2=t/(N-2);
}

void LSF_err::make_siga() {
    make_s2();
    siga=sqrt(s2*sigma(x,x)/delta);
}
```

```
void LSF_err::make_sigb() {
    sigb=sqrt(s2*N/delta);
}
void LSF_err::display() {
    cout << "Slope is " << b << "+-" << sigb << endl;
    cout << "Y-intercept is " << a << "+-" << siga << endl;
}
main () {
  LSF_err temp(9);
  temp.getdata();
  temp.make_a();
  temp.make_b();
  temp.make_siga();
  temp.make_sigb();
  temp.display();
}
```

Slope is 9.40833**+-0.868302**

Y-intercept is 4.81389**+-4.88621**