CalcLib Example of Calc::Cheb

```
Calc::Cheb Example
Objective: perform a Chebyshev curve-fit of a test function, its exact integral &
   derivative. Then, take derivatives & integrals of the Chebyshev curve-fits. Compare
   for errors.
References:
   - Press, William H, et al, Numerical Recipes, 2nd Ed, Cambridge Press, 1992.
   - Spiegel, MR, et al, Schaum's Mathematical Handbook, 5th Ed, McGraw-Hill, 2018.
Author: Everett George Copyright (C) 2023
   https://www.jefgeorge.com
   07/06/2023
```

Output:

```
calc::Cheb Class Example Application

exact evaluation of fn(x)          fn(x) = 1.5509+i0.32240  (exact)
interpolate fn(x)                 fn0(x) = 1.5630+i0.32558  (4 coefficients)
interpolate derivative of ifn(x)  fn1(x) = 1.5570+i0.32393  (3 coefficients)
interpolate integral of dfn(x)    fn2(x) = 1.5502+i0.32159  (5 coefficients)
```

C++ Source:

```
#include "Cpx.h"                                      // complex variable definitions
#include "Cheb.h"                                     // Chebyshev polynomial objects
#include "PrtC.h"                                     // auxiliary print stream object

using namespace std;                                  // standard C++ library namespace
using namespace calc;                                 // CalcLib namespace

typedef Cpx<float> CPXf;                              // alias for <yType>

// test function data
const CPXf C1(-1.0f,-1.0f);                           // test function constant
const CPXf C2(-1.0f, 1.0f);                           // test function constant
CPXf  fn(float x) {return C1*sin(C2*x);}              // test function
CPXf dfn(float x) {return C1*C2*cos(C2*x);}           // test function derivative
CPXf ifn(float x) {return C1/C2*(1.0f-cos(C2*x));}    // test function integral

// results print function
void results(ostream &ostr, const char *cDat, CPXf cpf, int nCnt)
{
   ostr<<"   "<<cDat<<" = ";                          // print out title
   cpf.stream(ostr)<<"  ("<<nCnt<<" coefficients)";   // print data values
   ostr<<endl;                                        // end the text line
}

int main(void)
{
   // set parameters
   const float xLo   =0.0f;                           // domain lower value
   const float xHi   =0.5f*Base::PI;                  // domain upper value
   const float yError=0.05f;                          // polynomial curve-fit error
   CPXf y,y0,y1,y2;                                   // evaluation variables

   // determine interpolation coefficients
   Interp<float,CPXf>  fn0( fn,xLo,xHi,yError);       // curve-fit test function
   Interp<float,CPXf> ifn1(ifn,xLo,xHi,yError);       // curve-fit integral test function
   Interp<float,CPXf> dfn2(dfn,xLo,xHi,yError);       // curve-fit derivative test function

   // Integrate or differentiate to get the same function
   Differ<float,CPXf> fn1=ifn1;                       // differentiate integral test function
   Integr<float,CPXf> fn2=dfn2;                       // integrate derivative test function

   // evaluate equivalent function at same point
   const float xEval =0.25f*Base::PI;                 // evaluation point
   try {
      y=fn(xEval);                                    // test function
      y0=fn0.eval(xEval);                             // interpolated function
      y1=fn1.eval(xEval);                             // differentiated function
      y2=fn2.eval(xEval);                             // integrated function
   }
   catch(ChebErr& chebErr) {cout<<chebErr<<endl; return 1;}
   catch(...) {cout<<"Unknown execution error..."<<endl; return 1;}

   // print out evaluations comparing the functions
   cout.precision(5);
   cout.setf(ios::showpoint,ios::showpoint);
   cout<<endl<<"   calc::Cheb Class Example Application"<<endl<<endl;
   cout<<"   exact evaluation of fn(x)          fn(x) = "; y.stream(cout)<<"  (exact)"<<endl;
   results(cout,"interpolate fn(x)                 fn0(x)",y0,fn0.getCount());
   results(cout,"interpolate derivative of ifn(x)  fn1(x)",y1,fn1.getCount());
   results(cout,"interpolate integral of dfn(x)    fn2(x)",y2,fn2.getCount());
   cout<<endl;
   cout<<flush;                                       // clear print buffer

   return 0;
}
```