More Examples

Michael Hanke

School of Engineering Sciences

Program construction in C++ for Scientific Computing

Outline

The Problem

Write a program with the following properties:

- The user provides the dimension n of two vectors.
- The program allocates two vectors of the required size and fills them with meaningful data.
- The program computes the scalar product of the two vectors and prints the result.

The Algorithm

- We choose the vectors to be

$$x_i = i$$
$$y_i = \sin x_i$$

  for $i = 0, \ldots, n-1$.
- The scalar product is given by

$$s = \sum_{i=0}^{n-1} x_i y_i$$

The Scalar Product Function

```
double scprod(int n, double x[], double y[]) {
   double sum = 0;
   for (int i=0 ; i < n ; i++) sum += x[i]*y[i];
   return sum;
}
```

The length n of the vectors must be given as a parameter. There are no means to find it out inside the routine.

The Main Program

```
#include <iostream>
#include <cmath>
using namespace std;
double scprod(int n, double x[], double y[]);
int main() {
    double *x, *y;
    cout << "Give vector size ";
    int n; cin >> n;
    x = new double[n];
    y = new double[n];
    for( int i = 0 ; i < n ; i++ ) {
      x[i] = i;
      y[i] = sin(x[i]);
    }
    cout << "Scalar product is " << scprod( n, x, y ) <
    delete[] x;
    delete[] y;
}
```

Good Programming Style: Data Abstraction And Encapsulation

- The main program must know how the scalar product function is called. However, it is not necessary to know how it is implemented.
- Consequence: Subdivide the function into a file containig the declaration (the so-called header file) and its definition.

Data Abstraction

Data abstraction is a programming (and design) technique that relies on the separation of *interface* and *implementation*.

Encapsulation

Encapsulation enforces the separation of interface and implementation.

While it is a good programming style in C, it is an essential feature of C++!

Interface: The Header File scprod.hpp

```
#ifndef SCPROD_HPP
#define SCPROD_HPP

double scprod(int n, double x[], double y[]);

#endif
```

The use of the macro SCPROD_HPP prevents a double inclusion (which may lead to a syntax error).

The Implementation scprod.cpp

```
#include "scprod.hpp"

double scprod(int n, double x[], double y[]) {
   double sum = 0;
   for( int i=0 ; i < n ; i++ ) sum += x[i]*y[i];
   return sum;
}
```

The header is included in order to ensure that the declaration and the definition of scprod coincide.

The Main Program

```
#include <iostream>
#include <cmath>
#include "scprod.hpp"
using namespace std;
int main() {
    double *x, *y;
    cout << "Give vector size ";
    int n; cin >> n;
    x = new double[n];
    y = new double[n];
    for( int i = 0 ; i < n ; i++ ) {
      x[i] = i;
      y[i] = sin(x[i]);
    }
    cout << "Scalar product is " << scprod( n, x, y ) <
    delete[] y;
    delete[] x;
}
```

The Problem

Write a library routine for the secant method and test it with some examples.

The Algorithm

The secant method works as follows for solving the equation $f(x) = 0$:

$$x_{i+1} = x_i - \frac{f(x_i) - f(x_{i-1})}{x_i - x_{i-1}} f(x_i)$$

with given x_0, x_1.

The Header secant.hpp

```
#ifndef SECANT_HPP
#define SECANT_HPP

typedef double (*FunctionPointer)(double);

double secant(double a, double b, FunctionPointer f);

#endif
```

The Implementation secant.cpp

```
#include <iostream>
#include <cstdlib>
#include <cmath>
#include "secant.hpp"
double secant(double a, double b, FunctionPointer f) {
  double tol = 1e-15;
  double x,y,fa,fb;
  int niter = 0;
  fa = f(a); fb = f(b);
  while(fabs(b-a)>tol && fa!=fb) {
  niter = niter+1;
    if(niter > 100) {
      std::cerr << "No convergence! Exiting..." << std::endl;
      exit(1);
    }
    x = a - (b-a)*fa/(fb-fa);
    y = f(x);
    b  = a ; a  = x;
    fb = fa; fa = y;
  }
  return a;
}
```

The Main Program

```
#include <iostream>
#include <iomanip>
#include <cmath>
#include "secant.hpp"
using namespace std;
double sinpluscos(double x) { return sin(x)+cos(x); }
int main(void) {
  double y1,y2,y3,y4;
  y1 = secant(-1.0,1.0,&sin);
  y2 = secant(3,4,&sin);
  y3 = secant(1,2,&cos);
  y4 = secant(-1,-2,&sinpluscos);
  cout << "f                x                f(x)" << endl;
  cout << scientific;
  cout << "sin(x)        " << y1 << "  " << sin(y1) << endl;
  cout << "sin(x)        " << y2 << "  " << sin(y2) << endl;
  cout << "cos(x)        " << y3 << "  " << cos(y3) << endl;
  cout << "sin(x)+cos(x) " << y4 << "  " << sinpluscos(y4) << endl;
  return 0;
}
```

Program Output

```
f                    x                 f(x)
sin(x)          0.000000e+00  0.000000e+00
sin(x)          3.141593e+00  1.224647e-16
cos(x)          1.570796e+00  6.123234e-17
sin(x)+cos(x) -7.853982e-01   -1.110223e-16
```

Summary

- We developed two numerical programs.
- Data abstraction and encapsulation

- What will come next:
  - The basics of object oriented programming: C++ classes