Teil I

C++ Klausur (WS 05/06) bei Prof. Steinhauser

Aufgabe 1: **(5 Punkte)**

Gegeben sei die folgende Potenzreihe

$$f_n(x) = \sum_{k=1}^{n} \frac{e^{-k \cdot x}}{k^2}, \quad x \geq 0 .$$

Schreiben Sie ein vollständiges C++-Programm zur Berechnung dieser Summe. Der Benutzer soll die reelle Stelle x und die obere Summengrenze n eingeben können und das Programm dann den Summenwert ausgeben.

Aufgabe 2: **(5 Punkte)**

Schreiben Sie eine C++-Funktion, die in einem Feld von `double` Variablen im Bereich der Feldkomponenten `a,...,b` (einschließlich) den Index des betragsgrößten Elements bestimmt und diesen Index als Funktionswert zurück gibt. Die Feldgrenzen sind hierbei vom Datentyp `int`. Falls der betragsgrößte Wert bei mehreren Indizes auftreten sollte, ist unter diesen Indizes der kleinste zu wählen.
Es soll kein Zugriff auf globale Variablen erfolgen, d.h. sämtliche Parameter müssen übergeben werden.

Aufgabe 3: **(12 Punkte)**

Gegeben sei folgendes Programm, das eine Klasse für dreidimensionale Vektoren benutzt:

```
#include <iostream>
#include <cmath>
using namespace std ;

// ... hier fehlt ein Programmteil ...

int main()
{
    vec3d v1 (1.0, 2.0, 2.0) ;
    vec3d v2 (0.0, 1.0, 0.0) ;
    vec3d v3 ;
    v3 = kp(v1,v2) ; // Kreuzprodukt
    cout << "Laenge v3 vorher: " << sqrt(v3*v3) << endl ;
    v3.norm() ; // v3 auf Laenge 1 normieren
    cout << "Laenge v3 nachher: " << sqrt(v3*v3) << endl ;
    cout << "Ist v3 senkrecht auf (v1,v2)? " << v3*v1 << " " << v3*v2 << endl ;
}
```

Hierbei bezeichnet: '`kp`' Kreuzprodukt zweier Vektoren; '`*`' Skalarprodukt zweier Vektoren; '`norm`' Normierung des Vektors selbst auf Länge 1 per Euklidnorm.
Schreiben Sie die benötigte Klasse `vec3d` und alle benötigten Konstruktoren, Mitgliedsfunktionen und freien Funktionen, die in dem gegebenen Hauptprogramm aufgerufen werden. Die Vektorkomponenten selbst sollen von außen unzugängliche `double` Variablen sein. Die Definition der Mitgliedsfunktionen soll jeweils außerhalb der eigentlichen Klassendefinition erfolgen.

Aufgabe 4: **(10 Punkte)**

Setzen Sie an der markierten Stelle in folgendem Programm die letzte Ziffer Ihrer Matrikelnummer ein. Schreiben Sie dann bitte vollständig auf, was von dem Programm ausgegeben wird.

```
#include <iostream>
using namespace std ;

int f(int & x) {
    x += 1 ;
    cout << "UP1: " << x << endl ;
    return x*2 ;
}

double f(double x) {
    x += 2.0 ;
    cout << "UP2: " << x << endl ;
    return x*2 ;
}

int f(int r, int s) {
    if (s==0) return 0 ;
    else return r+f(r,s-1) ;
}

int main() {
    int i= 6 ; // <- setzen Sie hier bitte die letzte Ziffer
               //    Ihrer Matrikelnummer ein.
    double a=2.2 ;

    double b=f(a) ;
    cout << "HP: " << a << " " << b << endl ;

    int j=f(i) ;
    cout << "HP: " << i << " " << j << endl ;

    int* p ;
    p = &i ;
    cout << "HP: " << *p << endl ;
    f(i) ;
    cout << "HP: " << *p << endl ;

    cout << "HP: " << f(4,3) << endl ;
}
```

Aufgabe 5: **(8 Punkte)**

Schreiben Sie ein vollständiges C++-Programm, das eine positive ganze Zahl von der Tastatur einliest und die Primfaktorzerlegung der Zahl ausgibt. Jeder Primfaktor ist so oft auszugeben, wie er in der Zahl vorkommt. Rechenzeichen zwischen den Primfaktoren brauchen nicht ausgegeben zu werden, die Ausgabe kann beispielsweise in folgender Form erfolgen:

```
60: 2 2 3 5
```

Teil II
Lösungen

(keine offizielle Musterlösung!)

Aufgabe 1

```
//-----------------------------------------
#include <iostream>
#include <cmath>

using namespace std;

int main()
{
    double x;
    double sum = 0;
    int n;

    cout << "Bitte x eingeben: ";
    cin >> x;
    cout << "Bitte n eingeben: ";
    cin >> n;

    for( int k = 1; k <= n; k++ )
    {
        sum += exp( -k * x ) / ( k * k );
    }

    cout << "f_" << n << "(" << x << ")= " << sum;
    return( 0 );
}
//-----------------------------------------
```

Aufgabe 2

```
//-----------------------------------------
int getIndex( double field[], int a, int b )
{
    int erg = a; //wird später mit return zurück gegeben
    for( i = a; i <= b; i++ )
    {
        if( abs( field[i] ) > abs( field[erg] ) )
            erg = 1;
    }
    return( erg );
}
//-----------------------------------------
```

Dadurch, dass in der `if`-Anweisung „`>`“ anstelle von „`>=`“ verwendet wurde, wird gewährleistet, dass bei mehreren Indizes, die den betragsgrößten Wert liefern, stets der Kleinste zurück gegeben wird.

Aufgabe 3

```
//-----------------------------------------
//Deklarationen

class vec3d
{
    private:
        double x, y, z;
    public:
        vec3d(); //Defaultkonstruktor
        vec3d( double x_, double y_, double z_ ); //Konstruktor mit Initialisierungswerten
        void norm();

        friend double operator * ( vec3d v1, vec3d v2 );
};

friend vec3d kp( vec3d v1, vec3d v2 );

//Definitionen (Implementierung)

vec3d::vec3d() //Erzeugt den Nullvektor
{
    x = 0.0; y = 0.0; z = 0.0;
}

vec3d::vec3d( double x_, double y_, double z_ )
{
    x = x_; y = y_; z = z_;
}

void vec3d::norm()
{
    double len = sqrt( x*x + y*y + z*z );
    x /= len;
    y /= len;
    z /= len;
}

double operator * ( vec3d v1, vec3 v2 )
{
    return( v1.x * v2.x + v1.y * v2.y + v1.z * v2.z );
}

vec3d kp( vec3d v1, vec3d v2 )
{
    vec3d ergvec ( v1.y * v2.z - v1.z * v2.y ,
                   v1.z * v2.x - v1.x * v2.z ,
                   v1.x * v2.y - v1.y * v2.x );
    return( ergvec );
}
//-----------------------------------------
```

Aufgabe 4

Ausgabe des Programm, mit eingesetzter Ziffer „6“:

```
UP2: 4.2
HP: 2.2 8.4
UP1: 7
HP: 7 14
HP: 7
UP1: 8
HP: 8
HP: 12
```

Aufgabe 5

```
//----------------------------------------
#include <iostream>

using namespace std;

void factor( int n )
{
    int i = 2;
    if( n > 1 ) //Falls n = 1 keine erneute Rekursion
    {
        while( n % i != 0 )
            i++;
        cout << i << " ";
        factor( n / i ); //Rekursion mit abgeteiltem Primfaktor
    }
}

int main()
{
    int n;
    cin >> n;
    cout << n << ": ";
    factor( n );
    cout << endl;
    return( 0 );
}
//----------------------------------------
```

Wichtig!!! Es handelt sich bei den hier angegebenen Lösungen **nicht** um eine Musterlösung, sondern um eine Lösung, die ich in der Klausur verwendet habe. Allerdings habe ich sie hier noch ein wenig verbessert.
Wieder mal gilt: **Keine Gewähr oder Haftung!**
Falls ihr einen Mitschrieb der Vorlesung sucht, findet ihr ihn auf `www.castlearts.de/cplusplus`.

Magnus Schlösser