Klausur zur Vorlesung
Programmieren für Physiker — SS 2013

17. Juli 2013

Interfakultatives Institut für Anwendungen der Informatik
Institut für Theoretische Teilchenphysik

Bitte deutlich schreiben.
Telefone bitte ausschalten.
Versehen Sie bitte alle Blätter mit Namen und Matrikelnummer.
Dauer: 90 Minuten.
Bei Klausurende bitte ALLE Blätter abgeben — auch das Aufgabenblatt.
Zum Bestehen sind 50% der erreichbaren Punkte notwendig.
Ergebnisse und Klausureinsicht: Freitag, 19.07.2013, ab 8:00 Uhr im Lehmann-HS. Weitere Details zu Klausurbesprechung, -einsicht und Scheinen stehen auf der Webseite zur Vorlesung.

Studiengang: (bitte ankreuzen oder angeben)

- [x] Bachelor of Science Physik
- [ ] Bachelor of Science Geophysik
- [ ] Bachelor of Science Meteorologie
- [ ] Diplom Physik
- [ ] Diplom Geophysik
- [ ] Diplom Meteorologie
- [ ] Anderer: _____

Nachname: [redacted]

Vorname: [redacted]

Matrikelnummer: [redacted]

Aufgabe	1 (6P)	2 (4P)	3 (9P)	4 (4P)	5 (7P)	Summe (30)
Punkte	6 J.G.	4 HB	9 DK	4 FH	7 MS	30 JG.

Aufgabe 1: Potenzen (6 Punkte)

Eine C++-Funktion mit den Argumenten $a \in \mathbb{R}$ und $n \in \mathbb{N}_0$ soll die Potenz a^n berechnen und als Funktionswert zurückgeben. Dies soll durch n-faches Aufmultiplizieren der Faktoren a geschehen (also insbesondere ohne Verwendung von `pow` oder der Exponentialfunktion). Schreiben Sie zwei voneinander unabhängige Varianten dieser Funktion:
(a) iterativ,
(b) rekursiv.
Sie brauchen nicht zu überprüfen, ob das Funktionsargument $n >= 0$ ist.

Aufgabe 2: Positives Feld? (4 Punkte)

Schreiben Sie eine C++-Funktion `ist_positiv`, die untersucht, ob in einem Feld von n `double` Variablen alle Werte echt größer als null sind. Die Funktion soll die Länge n des Felds und das Feld selbst als Argument erhalten. Gültige Feldindizes sind im Bereich $0, \ldots, n-1$.
Der Rückgabewert soll ein Logikwert sein.

Aufgabe 3: Kreis-Klasse (9 Punkte)

Eine Klasse `kreis` soll in der x-y-Ebene Kreise durch Mittelpunkt (x, y) und Radius r charakterisieren. Diese Variablen sollen von außen unzugänglich sein. Als Schnittstelle sind folgende Methoden und freie Funktionen zu programmieren: (i) Vereinbarung eines Kreises am Ursprung mit gegebenem Radius. (ii) Verschieben des Mittelpunktes um $(\Delta x, \Delta y)$. (iii) Radius mit einem Faktor multiplizieren. (iv) Überprüfen, ob der Ursprung innerhalb eines Kreises liegt.
Die Klasse kommt in folgendem Programm zum Einsatz:

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

// Klasse kreis und freie Funktion zu ergaenzen

int main()
{
    kreis k1(4.0) ;       // Kreis: mit Radius 4, Mittelpunkt ist (0,0)
    k1.move(2.0, 3.0) ;   // Mittelpunkt um (2,3) verschieben
    k1 *= 3.0 ;           // Radius mit Faktor multiplizieren

    if (enthaelt_ursprung(k1))
       cout << "Kreis k1 enthaelt den Ursprung" << endl ;
}
```

Schreiben Sie die benötigte Klasse `kreis` und alle benötigten Konstruktoren, Mitgliedsfunktionen und freien Funktionen, die in dem gegebenen Hauptprogramm aufgerufen werden. Die Definition der Mitgliedsfunktionen soll jeweils außerhalb der eigentlichen Klassendefinition erfolgen. Bei Bedarf kann die Wurzel einer Zahl mittels `sqrt(.)` berechnet werden und die Kreiszahl π mit 3.14 angesetzt werden.

Aufgabe 4: Was wird ausgegeben? (4 Punkte)

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

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

int f(double a) {
    a += 2;
    double* p = &a;
    (*p) -= 1.0 ;
    return 2*a ;
}

int f(int& z) {
    z += 5 ;
    return 3*z ;
}

int main() {
    int v1= 0 ; // <- setzen Sie hier bitte die letzte Ziffer
                //    Ihrer Matrikelnummer ein.
    double v2 = v1+0.1 ;

    int j=f(v1) ;
    cout << "Alpha: " << v1 << " " << j << endl ;

    j=f(v2) ;
    cout << "Beta: " << v2 << " " << j << endl ;
}
```

0.1 → Kopie → v2 bleibt gleich
2.1
Pointer auf a
1.1. *a wird um 1 erniedrigt
2.2
→ Referenz →
→ v1 += 5;

v1 = 5
j = 15

v2 = 0.1
j = int(2.2) = 2

Aufgabe 5: Einfach verkettete Liste (7 Punkte)

```
struct listelem {
  double zahl;
  listelem* next;
};
```

Mit obiger Struktur seien einfach-verkettete Listen aufgebaut. Schreiben Sie die folgenden beiden Funktion: eh:

(a) `getvalue`. Die Funktion soll als Argument einen Zeiger auf ein Listenelement erhalten und den `double`-Wert dieses Listenelements zurückgeben.

(b) `append`. Diese Funktion soll einen Zeiger auf ein gültiges Listenelement erhalten und ein `double`-Argument. Verfolgen Sie die Kette bis ans Ende und hängen Sie ein dynamisch erzeugtes, neues Listenelement an, welches den übergebenen `double`-Wert hat und als Listenende gekennzeichnet ist. Die Funktion hat keinen Rückgabewert.

Sie dürfen davon ausgehen, dass beide Funktionen stets gültige Zeiger auf Listenelemente erhalten und dass das Ende einer Listenkette mit dem Nullzeiger markiert ist.

Bitte geben Sie die Nummern der bearbeiteten Aufgaben deutlich lesbar an.

Aufgabe 1)

a)

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

double potenz1(double a, int n)
{ double erg = 1;
  for(int i = 1; i <= n; i++)
  {
     erg *= a;
  }
  return erg;
}
```

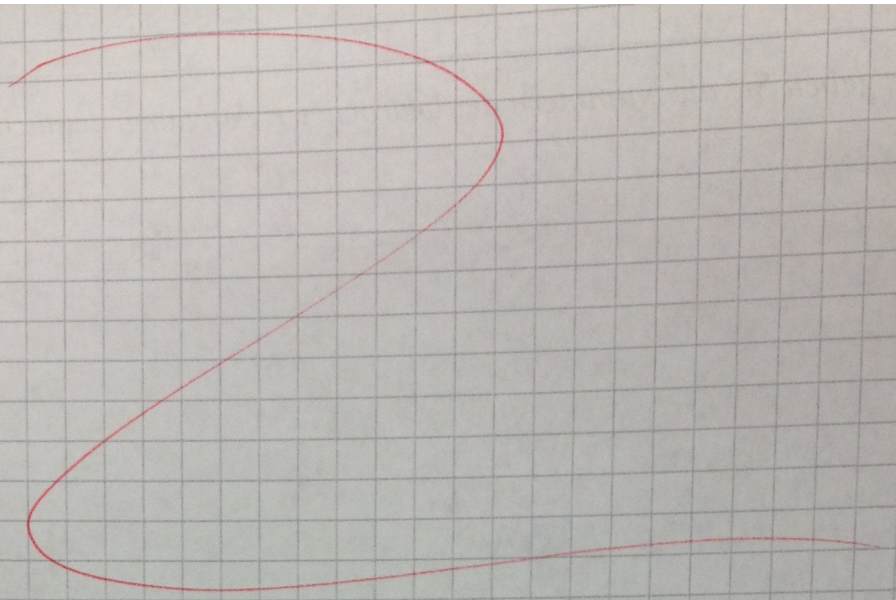
✓ 3P.

b)

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

double potenz2(double a, int n)
{
    if(n==0)
    {
       return 1.0;
    }
    return (potenz(a,n-1) * a);
}
```

✓ 3P.



Aufgabe 2)

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

bool ist_positiv(double a[], int n)
{
    bool antwort = true;
    for(int i = 0; i < n; i++)
    {
      if( a[i] <= 0)
      {
        antwort = false;
      }
    }
    return antwort;
}
```

4/4

Name: [redacted] Vorname: [redacted] Studienfach: Physik (Bachel

Bitte geben Sie die Nummern der bearbeiteten Aufgaben deutlich lesbar an.

Aufgabe 3)

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

class kreis{
private:
double x;
double y;
double r;

public:
kreis(double radius);
void move(double dx, double dy);
void operator*=(double faktor);
friend bool enthaelt_ursprung(kreis k);

};

kreis::kreis(double radius)
{
    x = 0.0;
    y = 0.0;
    r = radius;
}
```

```
void kreis::move(double dx, double dy)
{
    x += dx;
    y += dy;
}

void kreis::operator *=(double faktor)
{
   r *= faktor;
}

bool enthaelt_ursprung(kreis k)
{
    if(sqrt(k.x*k.x+k.y*k.y)<r)
    if(sqrt(k.x * k.x + k.y * k.y) <k.r)
    {
       return true;
    }
       return false;
}
int main()
{
    kreis k1(4.0);
    k1.move(2.0, 3.0);
    k1 *= 3.0;
    if(enthaelt_ursprung(k1))
    cout << "Kreis k1 enthaelt den Ursprung" << endl;
}
```