Klausur zur Vorlesung
Programmieren für Physiker — SS 2011

12. Juli 2011

Interfakultatives Institut für Anwendungen der Informatik
Institut für Theoretische Physik

Bitte deutlich schreiben.
Telefone bitte ausschalten.
Versehen Sie bitte alle Blätter mit Namen und Matrikelnummer.
Dauer: 90min.
Bei Klausurende bitte ALLE Blätter abgeben — auch das Aufgabenblatt.
Zum Bestehen sind 50% der erreichbaren Punkte notwendig.
Ergebnisse und Klausureinsicht: Freitag, 15.07.2011, 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 (4P)	2 (6P)	3 (4P)	4 (8P)	5 (8P)	Summe (30)
Punkte	2.5 BP	5 FR	4 M	7 M	8	26.5

MS

Aufgabe 1: Heronverfahren (4 Punkte)

Das Heronverfahren kann zur iterativen Berechnung der Quadratwurzel einer gegebenen Zahl a verwendet werden. Die Iterationsvorschrift ist:

$$x_0 = 1, \qquad x_{n+1} = \frac{x_n + \frac{a}{x_n}}{2} \quad (\text{für } n \geq 0).$$

Schreiben Sie ein vollständiges C++-Programm, in dem eine Gleitkommazahl a vom Benutzer abgefragt wird und dann mittels des Heronverfahrens so lange iteriert wird, bis zum ersten Mal $|x_n^2 - a| \leq 10^{-3}$ ist. Geben Sie dann diesen Wert x_n als Näherungswert für $\sqrt{a}$ aus.

Aufgabe 2: Betragssummennorm (6 Punkte)

Die Betragssummennorm eines n-komponentigen Vektors (x_i), $i = 1, \ldots, n$ ist definiert als Summe der Beträge der Komponenten:

$$\|x\|_1 = \sum_{i=1}^{n} |x_i|.$$

(a) Schreiben Sie eine C++-Funktion `bnorm`, die per Schleife die Betragssummennorm eine Feldes von n `double` Variablen berechnet und zurückgibt. Die Funktion soll die Länge n des Felds und das Feld selbst als Argument erhalten. Gültige Feldindizes sind im Bereich $0, \ldots, n-1$.

(b) Schreiben Sie eine rein rekursive Variante `bnormrec`, die die gleiche Aufgabe ohne explizite Schleife erledigt. Die Funktion soll die gleichen Argumente und den gleichen Rückgabewert wie in Teil (a) haben. Überlegen Sie sich hierzu, wie sie die Aufgabenstellung durch Abspalten des letzten Feldelements auf einen etwas kleineren Vektor übertragen können.

Hinweis: Den Betrag einer Zahl erhalten Sie durch `abs(.)`.

Aufgabe 3: Was wird ausgegeben? (4 Punkte)

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

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

char f(char& c) {
    c = 'R' ;
    return *((&c)-2) ;
}

char f(char* p) {
    p[2] = 'L' ;
    p++ ;
    return *p ;
}

int main() {
    char s1[3]={'␣','␣','␣'} ; // <- setzen Sie hier bitte die
                               //    *drei letzten* Ziffern
                               //    Ihrer Matrikelnummer ein.

    char s3 = f(s1) ;
    cout << "Main1: " << s1[2] << ", " << s3 << endl ;
    s3 = f(s1[2]) ;
    cout << "Main2: " << s1[2] << ", " << s3 << endl ;
}
```

Aufgabe 4: Intervall-Klasse (8 Punkte)

Eine Klasse soll zur Darstellung von abgeschlossenen Intervallen verwendet werden. Abgeschlossene Intervalle sind charakterisiert durch eine reelle untere und obere Grenze, die beide selbst jeweils noch zum Intervall dazugehören. Diese Variablen sollen von außen unzugänglich sein, d.h. nur durch Methoden und geeignete Funktionen gelesen oder geändert werden.
Schreiben Sie eine Klasse `intervall` und Funktionen, die folgendes leisten: (i) Initialisieren eines Intervalls; (ii) Verschiebung (beider Grenzen) eines Intervalls; (iii) Ermitteln der Intervallgröße; (iv) Überprüfen, ob ein Intervall komplett in ein zweites hinein passt. Das Ergebnis soll ein Logikwert sein.

Die Klasse kommt in folgendem Programm zum Einsatz:

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

// Klasse intervall und freie Funktion zu ergaenzen

int main()
{
    intervall i1 ( 1.5, 3) ;  // ein Intervall mit gegebenen Grenzen
    intervall i2 ( 2, 5) ;    // ... und noch eines

    i1 += 1 ;                 // Intervall wird verschoben (beide Grenzen),
                              // dabei wird i1 selbst veraendert

    cout << "Intervall i2 hat die Groesse " << size(i2) << endl ;

    if (i1.is_contained_in (i2))
       cout << "i1 ist mit dem gesamten Wertebereich innerhalb i2" << endl ;
}
```

Schreiben Sie die benötigte Klasse `intervall` und alle benötigten Konstruktoren, Mitgliedsfunktionen und freien Funktionen, die in dem gegebenen Hauptprogramm aufgerufen werden.

Aufgabe 5: Pseudo-Schnapszahl (8 Punkte)

Eine positive ganze Zahl sei genau dann als *Pseudo-Schnapszahl* definiert, wenn *nicht mehr als zwei verschiedene* Ziffern zu Ihrer Darstellung benötigt werden.

Bsp: Die Zahlen 1911911, 10, 555 und 7 sind Pseudo-Schnapszahlen.
Die Zahlen 99660 und 123 sind keine Pseudo-Schnapszahlen.

Schreiben Sie ein vollständiges C++-Programm, das für eine einzugebende Zahl ermittelt, ob diese Zahl eine Pseudo-Schnapszahl ist, und dies als Ergebnis ausgibt.

Kommentieren Sie die einzelnen Abschnitte in Ihrem Programm, so dass sofort verständlich ist, was gemacht wird. Sie brauchen nicht zu überprüfen, ob die eingegebene Zahl positiv ist.
Hinweis: Der Operator % ergibt den Rest einer Ganzzahl-Division.

Die ist keine offizielle Musterlösung. Alle Quellcodes können direkt kopiert und kompiliert werden.

Aufgabe 1:

```
#include <iostream>
#include <cmath>

using namespace std;

int main(){
   double a;
   cout << "Bitte geben Sie die Zahl ein, mit der gerechnet werden soll.\n";
   cin >> a;
   double x=1;
   while(fabs(x*x-a)>=0.001){
      x=fabs(0.5*(x-a/x));
   }
   cout << "Die Wurzel von " << a << " ist Naeherungsweise +/- " << x << endl;

   system("pause");
   return(0);
}
```

Aufgabe 2:

```
#include <iostream>
#include <cmath>

using namespace std;

// a)
double bnorm(double A[], int n){
   double erg=0;
   for(int i=0;i<n;i++){
      erg+=fabs(A[i]);
   }
   return erg;
}

// b)
double bnormrec(double A[], int n){
   double erg=0;
   if(n==1){    return fabs(A[0]);    }
   else{    erg+=fabs(bnormrec(A,n-1))+fabs(A[n-1]);    }
   return erg;
}

//Hauptprogramm war nicht gefragt, zur Kontrolle ist es hier jedoch angegeben.
int main(){
   int n;
   double A[100];
   cout << "Bitte geben sie dir Groesse ihres Feldes an (max. 100): ";
   cin >> n;
   cout << "Bitte geben die die einzelnen Feldkomponenten mit Enter oder Leerzeichen\ngetrennt
ein:\n";
   for(int i=0;i<n;i++){
      cin >> A[i];
   }
   cout << "Das Ergebnis mithilfe von a) lautet: " << bnorm(A,n) << endl;
   cout << "Das Ergebnis mifhilfe von b) lautet; " << bnormrec(A,n) << endl;

   system("Pause");
   return(0);
}
```

Aufgabe 3:

```
#include <iostream>

using namespace std;

char f(char& c){
   c='R';
   return *((&c)-2);
}

char f(char* p){
   p[2]= 'L';
   p++;
   return *p;
}

int main(){
   char s1[3]={'_','_','_'};   //<- setzen Sie hier bitte die *drei letzten* Ziffern Ihrer Matrikelnummer
ein.

//Diese Zeile wird zu wahren Eingabe benötigt
   cin >> s1[0] >> s1[1] >> s1[2];

   char s3=f(s1);
   cout << "Main1: " << s1[2] << ", " << s3 << endl;
   s3=f(s1[2]);
   cout << "Main2: " << s1[2] << ", " << s3 << endl;

   system("Pause");
   return(0);
}
```

Aufgabe 4:

```
#include <iostream>

using namespace std;

class intervall{
   private:
      double u,o;
   public:
      intervall(double, double);
      void operator+=(double);
      friend double size(intervall);
      bool is_contained_in(intervall);
};

intervall::intervall(double a, double b){
   u=a;
   o=b;
}

void intervall::operator+=(double v){
   u+=v;
   o+=v;
}

bool intervall::is_contained_in(intervall neu){
   if(u>=neu.u && o<=neu.o){    return true;    }
   else{    return false;    }
}

double size(intervall i){
   return i.o-i.u;
}

int main(){
   intervall i1 ( 1.5, 3);     // ein Intervall mit gegebenen Grenzen
   intervall i2 ( 2, 5);       // ... und noch eines

   i1+=1;                      // Intervall wird verschoben (beide Grenzen), dabei wird i1 selbst verändert

   cout << "Intervall i2 hat die Groesse " << size(i2) << endl;

   if(i1.is_contained_in (i2))
      cout << "i1 ist mit dem gesamten Wertebereich innerhalb i2" << endl;

   system("Pause");
   return(0);
}
```

Aufgabe 5:

```
#include <iostream>

using namespace std;

bool pseudo(int zahl){                                          //Untersuchen auf Richtigkeit
   int a=zahl%10;                                               //1. Restglied speichern
   zahl/=10;                                                    //1. Restglied abtrennen
   while(zahl%10==a){    zahl/=10;    }                                  //2. von 1. Restglied verschiedenes
bestimmen
   int b=zahl%10;                                               //2. Restglied speichern
   zahl/=10;                                                    //2. Restglied abtrennen
   while(zahl>0){                                               //schauen, ob restliche Glieder gleich a
oder b sind
      if(zahl%10==a || zahl%10==b){    zahl/=10;    }
      else{    return false;    }                              //wenn nein -> falsch
   }
   return true;                                                 //wenn ja -> richtig
}

int main(){
   cout << "Bitte geben Sie die Zahl ein.\n";                          //Eingabeabfrage
   int zahl;
   cin >> zahl;                                                 //Eingabe der Zahl
   if(pseudo(zahl)){
      cout << "Die Zahl " << zahl << " ist eine Pseudozahl.\n";
   }
   else{
      cout << "Die Zahl " << zahl << " ist eine keine Pseudozahl.\n";
   }

   system("pause");
   return(0);
}
```