Klausur zur Vorlesung
Programmieren für Physiker — WS 2008/09

10. Februar 2009

Interfakultatives Institut für Anwendungen der Informatik
Institut für Theoretische Teilchenphysik

Bitte deutlich schreiben.
Handys bitte ausschalten.
Versehen Sie bitte alle Blätter mit Namen und Matrikelnummer.
Bei Klausurende bitte ALLE Blätter abgeben — auch das Aufgabenblatt.
Zum Bestehen sind 50% der erreichbaren Punkte notwendig.
Klausurrückgabe und Scheinausgabe: Freitag, 13. Februar 2009 in den Übungen.
Beschwerden über die Korrektur können nur zu diesem Termin vorgetragen werden.

Name: Florian Rinke

Matrikelnummer: 7406388

Aufgabe	1 (4P)	2 (4P)	3 (6P)	4 (3P)	5 (5P)	6 (8P)	Summe (30)
Punkte	4 DM	4 Hg	5 fe	3 MS	5P MH	8 AH	29

Aufgabe 1: Teilersumme (4 Punkte)

Die Teilersumme einer Zahl ist definiert als Summe aller Ihrer Teiler. Beispiel: Die Teilersumme von 10 ist 18, wegen der Teiler 1, 2, 5 und 10.
Schreiben Sie ein vollständiges C++-Programm, das von der Tastatur eine `long`-Zahl einliest, die Teilersumme berechnet und ausgibt. Sie dürfen voraussetzen, dass die eingegebene Zahl größer null ist. Hinweis: Der Operator für den Divisionsrest ist `%`.

Aufgabe 2: Felder vergleichen (4 Punkte)

Schreiben Sie eine Funktion, die überprüft, ob zwei Felder mit ganzzahligen Einträgen im Indexbereich $0, \ldots, n-1$ identisch sind.
Die Funktion soll als Argument die beiden Felder mit dem Grunddatentyp `int` und die Größe der Felder n erhalten. Das Ergebnis ist als Logikwert von der Funktion zurückzugeben.

Aufgabe 3: Klasse für Polarkoordinaten (6 Punkte)

Schreiben Sie eine Klasse `polar2d`, die zur Beschreibung von Punkten in zwei Dimensionen dient. Ein Punkt soll dabei durch Polarkoordinaten (r, α) dargestellt werden. Der Winkel α wird dabei in üblicher Weise im Bogenmaß gegen den Uhrzeigersinn ab der positiven x-Achse gemessen.
Die Klasse soll in folgendem Programmteil zum Einsatz kommen:

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

// Klasse polar2d zu ergaenzen

int main()
{
    const double pi=3.141592654 ;

    polar2d p1 (1.0, pi/2) ; // vereinbare einen Punkt durch Radius und Winkel
    p1 *= 2.0 ;              // aendere Radius von p1 um angegebenen Faktor;
    p1.rotate(-pi/6) ;       // rotiere p1 um das Winkelargument ;
    cout << "Kartesische X-Koordinate: " << get_x (p1) << endl ;
}
```

Schreiben Sie die benötigte Klasse `polar2d` und alle benötigten Konstruktoren, Mitgliedsfunktionen und freien Funktionen, die in dem gegebenen Hauptprogramm aufgerufen werden. Der Radius und die Winkelvariable sollen dabei von außen unzugängliche `double`-Variablen sein, Sie brauchen bei beiden Variablen der Einfachheit halber keine Bereichsüberprüfungen vorzunehmen. Die Definition der Methoden soll außerhalb der eigentlichen Klassendefinition erfolgen. Die beide Methoden `*=` und `rotate` sollen jeweils den Punkt selbst ändern, beide Methoden haben jeweils ein `double` Argument. Die freie Funktion `get_x` soll die x-Koordinate des Punktes berechnen.

Florian Rinke, 1406358

Aufgabe 4: Trapezregel (3 Punkte)

Mit der Trapezregel kann ein Integral einer Funktion numerisch näherungsweise berechnet werden:

$$\int_a^b f(x)\,dx = \frac{b-a}{2}\Big(f(a)+f(b)\Big) + \text{Restglied}.$$

Schreiben Sie eine C++-Funktion, die als Argument einen Funktionszeiger und zwei Integrationsgrenzen a und b erhält und als Funktionswert die oben angegebene Trapezsumme berechnet. Die zu übergebende Funktion hat dabei folgende Signatur: `double f(double)`.

Aufgabe 5: Was wird ausgegeben? (5 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 ;

void f(int* q) {
    int i7 = *q ;
    (*q) += 3 ;
    i7   += 2 ;
    cout << "Fkt1: " << i7 << endl ;
}

void f(int& q) {
    q += 1 ;
    cout << "Fkt2: " << q << endl ;
}

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

int main() {
    int i1= 8 ; // <- setzen Sie hier bitte die letzte Ziffer
                //    Ihrer Matrikelnummer ein.
    int i2 = i1 ;
    f(i1) ;
    cout << "Main1: " << i1 << endl ;
    f(&i2) ;
    cout << "Main2: " << i2 << endl ;
    cout << "Main3: " << f(7,3) << endl ;
}
```

i7 = 8, kopie
i2 = 11
i7 = 10

i1 = 9;

kopie

f(7,3)
= 4 + f(6,4)
= 4 + 2 + f(5,5)
= 4 + 2 + 0
= 6

Aufgabe 6: Mirpzahlen (8 Punkte)

Mirpzahlen sind Zahlen, die vorwärts und rückwärts gelesen Primzahlen sind. Bsp: 13 ist Mirpzahl, da sowohl 13 als auch 31 prim sind.
Schreiben Sie ein vollständiges C++-Programm, das für eine eingegebene `long`-Zahl testet, ob sie Mirpzahl ist. Sie brauchen nicht zu überprüfen, ob die eingegebene Zahl positiv ist.

Hinweis: Durch Programmierung geeigneter Funktionen lässt sich die Aufgabe in Teilprobleme zerlegen.

④
```
double integrate ( double (*f)(double), double a, double b ) {
    double result = (b-a)/2 * (f(a) + f(b)); // vernachlässige hierbei Restglied
    return result;
}
```

; n = 8

⑤
```
Fkt2:  9
Main1:  9
Fkt1:  10
Main2:  11
Main3:  6
```

5P

⑥
```
#include <iostream>

using namespace std;

bool isPrime (long number) {
    for (long i=2; i<number; i++) { //ineffizient, aber funktioniert
        if ((number % i) ==0)
            return false;
    }
    return true;
}

long reverse (long number) {
    long flipped = (number % 10);
    while ((number /= 10)!=0 ) {
        flipped *= 10;
        flipped += (number % 10);
    }
    return flipped;
}
int main() {
    long input;
    input << cin
    bool normal = isprime (input);
    bool reversed = isprime (reverse (input));

    cout << "Die Zahl " << input << " ist " << ((normal && reversed)? "eine" : "keine");
    cout << " Mirpzahl" << endl;

    return 0;
}
```

(3)

(3)

(2)

388

①
```
#include <iostream>

using namespace std;

int main() {
    long input, divsum=0;          1
    input << cin;                  0,5
    cout << "Die Teiler sind: "
    for (int i = 1; i <= input ; i++) {   1
        if (input % i == 0) {             1
            cout << i << " ";
            divsum += i;
        }
    }
    cout << endl;                  0,5
    cout << "Die Teiler summe ist " << divsum << endl;
    return 0;
}
```

②
```
bool checkEquality (int data1[], int data2[], int size) {

    for (int i=0; i<size; i++) {
        if (data1[i] != data2[i])
            return false;
    }
    return true;
}
```
↳P.

③ ~~double get_x (polar2d &p);~~
```
class polar2d {                                   ✓ (0,5)
  private:
    double radius, angle;                         ✓ (1)
  public:
    polar2d operator*= (double factor);
    void rotate (double rotAngle);
    friend double get_x (polar2d &p);             (0,5)
    polar2d (double init_radius, double init_angle);
};
double get_x (polar2d &p);
polar2d::polar2d (double init_radius, double init_angle) {
    radius = init_radius;
    angle = init_angle;                           ✓ (1)
}
polar2d polar2d::operator*= (double factor) {
    polar2d returnValue (radius, angle * factor);
    return returnValue;                           (0,5)
}

void polar2d::rotate (double rotAngle) {
    angle += rotAngle;                            ✓ (1)
}

double get_x (polar2d &p) {
    return (p.radius * cos (p.angle));            (0,5)
}
```
radius *= factor;
somit wird der Input selbst nicht geändert