

Name:

Dr.-Ing. Hartmut Helmke
Fachhochschule
Braunschweig/Wolfenbüttel
Fachbereich Informatik

Matrikelnummer:

Punktzahl:

Ergebnis:

Ersatzklausur im WS 2005/06 :

Informatik III

Teilnahmebedingungen für diese Klausur:

Sie dürfen an dieser Klausur teilnehmen, sofern Sie für die Informatik III Klausur im Januar 2006 ordnungsgemäß angemeldet waren. Falls Sie damals nicht angemeldet waren, müssen Sie dieses evtl. noch nachträglich mit dem Prüfungsausschuss klären.

Falls Sie im Januar teilgenommen **und** bestanden haben, dürfen Sie jetzt nur teilnehmen, sofern es sich im Januar um den Freiversuch gehandelt hat.

Um das Verfahren für mich zu vereinfachen, in welche Liste ich die Zensur eintragen muss, bitte ich um Beantwortung der folgenden Fragen:

a.) Haben Sie bereits an der Klausur im Januar teilgenommen, d.h. wurde die Klausur bereits mit einer Zensur bewertet?

ja: nein:

b.) Falls Bewertung im Januar: Haben Sie im Januar bestanden und wollen jetzt Ihre Zensur verbessern?

ja: ^{1, 2} nein: ³

c.) Falls keine Bewertung im Januar erfolgt ist und Sie im Januar nicht zur Klausur anwesend waren:

Lag ein Attest für die Januar-Klausur vor?

ja: *siehe Fußnote 5*

nein: *siehe Fußnote 3*

d.) Ist im Januar bei Ihnen keine Bewertung erfolgt, obwohl Sie im Januar zur Klausur anwesend waren⁴?

ja: ⁵

Hiermit bestätige ich, dass ich die Teilnahmebedingungen verstanden habe und Sie akzeptiere.

Wolfenbüttel, den

¹Dieses ist nur erlaubt, sofern es sich im Januar um den Freiversuch handelte.

²Das bessere Ergebnis wird zählen.

³Ihr Prüfungszähler wird um eins erhöht, d.h.

- im Januar Freiversuch, dann jetzt 1. Versuch,
- im Januar 1. Versuch, dann jetzt 2. Versuch,
- im Januar 2. Versuch, dann jetzt letzter Versuch.

Sie dürfen am Semesterende im Juni 2006 die Klausur von Prof. Isernhagen **nicht** mitschreiben, selbst wenn Sie heute nicht bestehen. Sie dürfen frühestens wieder im WS06/07 (Januar 2007) an der Informatik III Klausur teilnehmen!

⁴d.h., sie hatten nur die Zeit von 14 bis ca. 16 Uhr für die Klausur eingeplant.

⁵Sie dürfen so mitschreiben als ob dieses die Klausur im Januar wäre.

- im Januar Freiversuch, dann jetzt auch Freiversuch,
- im Januar 1. Versuch, dann jetzt auch 1. Versuch,
- im Januar 2. Versuch, dann jetzt auch 2. Versuch,
- im Januar 3. Versuch, dann jetzt auch letzter Versuch.

Sie dürften, sofern Sie nicht bestehen oder dieses ein Freiversuch ist, auch am Semesterende im Juni 2006 die Klausur von Prof. Isernhagen mitschreiben.

Name:

Dr.-Ing. Hartmut Helmke
Fachhochschule
Braunschweig/Wolfenbüttel
Fachbereich Informatik

Matrikelnummer:

Punktzahl:

Ergebnis:

Freiversuch

F1

F2

F3

Ersatzklausur im WS 2005/06 :

Informatik III

Hilfsmittel sind bis auf Computer, Handy etc. erlaubt !

Bitte Aufgabenblätter mit abgeben !

Austausch von Hilfsmitteln mit Kommilitonen ist **nicht** erlaubt !

Die Lösungen können in einigen Fällen hier auf dem Aufgabenblatt angegeben werden. Sie dürfen aber auch Ihre Lösungen, falls erforderlich, auf separaten Blättern notieren.

Bitte notieren Sie auf **allen** Aufgabenblättern und separaten Blättern Ihren Namen bzw. Ihre Matrikelnummer. Auf eine absolut korrekte Anzahl der Blanks und Zeilenumbrüche braucht bei der Ausgabe nicht geachtet zu werden. Dafür werden keine Punkte abgezogen.

Hinweis: In den folgenden Programmen wird sehr häufig die globale Variable *datei* verwendet. Hierfür kann der Einfachheit halber die Variable *cout* angenommen werden. Die Variable *datei* diene lediglich bei der Klausurerstellung dem Zweck der Ausgabeumlenkung.

Geplante Punktevergabe

Punktziel	Sonderpunkte	erreicht
Hausaufgaben: 10 P.		
A1: 16 P.		
A2: 30 P.		
A3: 25 P.		
A4: 9 P.		
A5: 20 P.		
Summe 100 P.		

Aufgabe 1 : Textfragen zu Klassen

ca. 16 Punkte

Die folgende sehr kurze Klassendeklaration deklariert eine Klasse A:

```
class A{
    public:
        double x;
};
```

1.) Es wurde hier keine Methode explizit deklariert. Welche Operatoren, Konstruktoren etc. werden aber implizit zusätzlich vom Compiler erzeugt (Stichwort *minimale Standardschnittstelle*)?

2.) Geben Sie einen Programmausschnitt an, in dem die obigen implizit erzeugten Operatoren, Konstruktoren etc. **alle** verwendet werden.

3.) Implementieren Sie die obigen implizit erzeugten Operatoren, Konstruktoren etc. explizit, sodass diese genau das gleiche machen wie die implizit erzeugten.

4.) Wie viele Destruktoren kann eine Klasse maximal

besitzen?

5.) Deklarieren Sie eine Klasse, die zwei verschiedene Konstruktoren besitzt (Schnittstelle/Deklaration reicht).

6.) Geben Sie ein Beispiel für ein Funktionsobjekt an.

7.) Warum ist die folgende Funktion *TestPut* nicht so sehr für einen **automatisch** ablaufenden Test geeignet?

```
void TestPut() {
    MeineMatrix mat(5, 4);
    mat.Put(1,2, 28.4);
    cout << mat.Get(1,2);
}
```

Aufgabe 2 : Softwareentwicklung / Testen

ca. 30 Punkte

Sie geben im Getränkemarkt Ihre leeren Flaschen und Kisten ab. Implementieren Sie eine C++-Funktion *BerechnePfund*, die Ihnen ermittelt, wie viel Pfand Sie für B abgegebene Bierflaschen und S abgegebene Saftflaschen erhalten. Für eine Bierflasche gibt es 8 Cent Pfand und für eine Saftflasche 15 Cent. Bedenken Sie, dass jeweils 6 Saftflaschen bzw. 24 Bierflaschen in einem Kasten sind und Sie für jeden abgegebenen Kasten dann auch zusätzlich zum Flaschenpfand 1 Euro 50 Cent erhalten.

Gehen Sie bei der Lösung der Aufgabe in den Schritten vor, wie sie in der Vorlesung vorgestellt wurden. Sie dürfen sich hierbei auf die folgenden Schritte beschränken:

(*_____*)

Notieren Sie die Lösung auf einem Extra-Blatt.

(*_____*)

- a.) Definieren Sie drei wirklich **verschiedene** Tests für die Aufgabenstellung, d.h. für den Test der Funktion (Es geht hier um die Beschreibung/Auflistung von konkreten Ein- und Ausgaben der Tests – noch nicht um deren Implementierung in C++).

Die Spezifikation der Funktion ist noch nicht eindeutig. Wo es noch Wahlmöglichkeiten gibt, wählen Sie eine aus und machen Sie Ihre Entscheidungen explizit (niederschreiben). Dies gehört zum Thema der Beseitigung von Missverständnissen und Unklarheiten.

- b.) Geben Sie die Headerdatei *Berechne.h* mit Include-Wächter an, d.h. u.a. den Prototyp der **echten** Funktion *BerechnePfund* (keine Anweisungsfunktion).
- c.) Implementieren Sie **einen** der zuvor definierten Tests *Test1*, *Test2* und *Test3* in C++. Anmerkung: Auf die Implementierung der Datei *Test.h* dürfen Sie verzichten.
- d.) Beschreiben Sie mit Worten, was die Funktion *BerechnePfund* macht, d.h., was die Eingangsvariablen sind und **wie** (im Aufgabenteil e.)) die Funktion aus den Eingangsparametern den Funktionswert ermitteln soll.
- e.) Implementieren Sie **nun erst** die C++-Funktion *BerechnePfund* selbst.

Beachten Sie nochmals, dass die Implementierung von *BerechnePfund* im Teil e.) nur ein kleiner Teil der Lösung ist.

Die zugehörige Hauptfunktion *main*, die die Tests ausführt und damit indirekt *BerechnePfund* aufruft, ist bereits vorgegeben.

Listing 1: (./Code1/Tests/main.cpp)

```
#include <iostream>
using namespace std;
#include <cstdlib> // wegen EXIT_SUCCESS
#include "Test.h"

int main()
{
    if (Test1() &&
        Test2() &&
        Test3() ){
        cout << "Alle Tests erfolgreich\n";
        return EXIT_SUCCESS;
    }
    else {
        cout << "Tests gescheitert\n";
        return EXIT_FAILURE;
    }
}
```

Aufgabe 3 : Konstruktoren und Destruktoren, Polymorphie

ca. 25 Punkte

Im Folgenden können Sie Ihre Ausgaben direkt unter den Funktionen oder auf einem Extra-Blatt notieren. Gegeben ist die Klasse A

```
#include <memory> // wg. auto_ptr
#include <fstream> // wg. ofstream
using namespace std;

// globale Variable, die verwendet wird, um die
// Ausgabe in eine Datei umzulenken. Kann
// durch cout ersetzt werden.
ofstream datei ("Ausgabe.txt", ios::out);

class A{
public:
    A(int par = 4): a(par)
        {datei << "+A_" << a << "_";}
    ~A() {datei << "-A_" << a << "_";}

    int a;
};
```

a.) Welche Ausgabe erzeugt der Aufruf von der Funktion `funk1`?

```
void funk1(){
    A a1(11);
    A a2;
}
```

b.) Welche Ausgabe erzeugt der Aufruf von der Funktion `funk2`?

```
void funk2(){
    A* p = new A(3);
    p = new A(5);
    datei << "Ende_";
    delete p;
}
```

c.) Welche Ausgabe erzeugt der Aufruf von der Funktion `funk3`?

```
void funk3(){
    auto_ptr<A> ap(new A(3));
}
```

d.) Welche Ausgabe erzeugt der Aufruf von der Funktion `funk4`?

```
void funk4(){
    A* p1 = new A(3);
    A* p2 = new A(2);
    auto_ptr<A> ap1(p1);
    auto_ptr<A> ap2(p2); /* 1 */
    datei << "Weiter_";
    ap2 = ap1; /* 2 */
    datei << "Weiter_";
}
```

Erklären Sie Ihre Lösung durch mehrere Zeichnungen, aus denen die Speicherbelegungen auf dem Heap und auf dem Stack nach den Anweisungen `/*1*/` und `/*2*/` hervorgehen.

e.) Welche Ausgabe erzeugt der Aufruf von der Funktion `funk5`?

```
void funkRef(A& par) {
    par.a = 17;
}

void funk5() {
    A a1(11);
    funkRef(a1);
    datei << a1.a;
}
```

f.) Welche Ausgabe erzeugt der Aufruf von der Funktion `funk6`?

```
void funkWert(A par) {
    par.a = 17;
}

void funk6() {
    A a1(11);
    funkWert(a1);
    datei << a1.a;
}
```

g.) Welche Ausgabe erzeugt der Aufruf von der Funktion `funk7`?

```
class B: public A{
public:
    B(int par = 11): A(par)
        {datei << "+B_";}
    ~B() {datei << "-B_";}
};

void funk7() {
    B b1;
    B b2(8);
}
```

h.) Welche Ausgabe erzeugt der Aufruf von der Funktion `funk8`?

```
void funk8() {
    B* b[120];
    datei << "Ende_";
}
```

i.) Welche Ausgabe erzeugt der Aufruf von der Funktion `funk9`?

```
void funk9() {
    B b[2];
    datei << "Ende_";
}
```

Aufgabe 4 : eXtreme Programming und Referenzen

ca. 9 Punkte

a.) Geben Sie drei (nicht mehr) Basistechniken des *extreme Programming* an.

b.) In der Softwareentwicklungsfirma *Meyer & Clever* wechseln häufig die Mitarbeiter. Durch welche Basistechnik(-en) (2 reichen) von eXtreme Programming (XP) kann verhindert/abgemildert werden, dass dies zum Scheitern eines Softwareprojektes in der Firma *Müller & Clever* führt?

c.) Was gibt das folgende Programmfragment in die Datei `datei` aus?

```
ofstream datei ("WerteRef.txt");

void funk1(double x, double y) {
    y = 13;
    x = 14;
}

void funk2(double& x, double& y) {
    y = 13;
    x = 14;
}

void funk3(double* x, double* y) {
    *y = 13;
    *x = 14;
}

void funk4(double* x, double* y) {
    x = y;
    *y = 2 * *x;
}
```

```
}

int main() {
    double x, y;
    x = 2; y = 3;
    datei << "funk1\n";
    funk1(x, y);
    datei << x << " " << y << endl;

    x = 2; y = 3;
    datei << "funk2\n";
    funk2(x, y);
    datei << x << " " << y << endl;

    x = 2; y = 3;
    datei << "funk3\n";
    funk3(&x, &y);
    datei << x << " " << y << endl;

    x = 2; y = 3;
    datei << "funk4\n";
    funk4(&x, &y);
    datei << x << " " << y << endl;
}
```


Aufgabe 5 : STL

ca. 20 Punkte

Geben Sie Ihre Lösungen auf einem Extrablatt an.

a.) Welche Ausgabe erzeugt das folgende Programm in `datei` (Sowohl die Funktion *ZensurenAusgeben* als auch die Funktion *ZensurEintragen* erzeugen Ausgaben)?

```

/* Zensurenverwaltung */
#include <fstream> // wg. ofstream
using namespace std;
#include <vector>
#include <utility> // wg. pair
#include <string>

using namespace std;

// globale Variable, die verwendet wird, um die
// Ausgabe in eine Datei umzulenken. Kann
// durch cout ersetzt werden.
ofstream datei ("STL.txt", ios::out);

/* 1 */
// Typdefinitionen zum Anpassen
typedef pair<string, double> Stud;
typedef vector<Stud> FH;
typedef vector<Stud>::iterator Iter;
typedef vector<Stud>::
    const_iterator ConstIter;

/** Ausgabe der Namen und der zugehoerigen
    Zensuren aus fh in den Strom datei.
*/
void ZensurenAusgeben(const FH& fh);

/* n wird im Vektor fh gesucht.
    Falls es gefunden wird, wird die
    zugehoerige Zensur z
    eingetragen/geaendert.
    Falls n nicht gefunden wird,
    werden n und z am Ende von dem
    Vektoren fh eingefuegt und
    eine Meldung nach datei
    ausgegeben.
*/
void ZensurEintragen(const string& n,
    double z, FH& fh )
{
    for (int i=0; i < fh.size(); ++i) {
        if (fh[i].first == n) {
            fh[i].second = z;
            return;
        }
    }
    datei << "Neu_list" << n <<endl;
    fh.push_back(make_pair(n, z));
}

```

```

/* Drei Studenten mit Dummy-Zensur 0
    werden in den Vektor eingetragen.
*/
void Init(FH& fh)
{
    fh.push_back(Stud("Fritz_Mey", 0.0));
    fh.push_back(Stud("Karl_Pey", 0.0));
    fh.push_back(Stud("Elke_Mopp", 0.0));
}

int main()
{
    FH fhwf;
    Init(fhwf);

    ZensurEintragen("Fritz_Mey", 2.3, fhwf);
    ZensurEintragen("Anke_Piep", 4.0, fhwf); // neu
    ZensurEintragen("Karl_Pey", 5.0, fhwf);
    ZensurEintragen("Elke_Mopp", 1.0, fhwf);

    ZensurenAusgeben(fhwf);
}

void ZensurenAusgeben(const FH& fh)
{
    ConstIter iter = fh.begin();
    while (iter != fh.end()){
        datei << iter->first << ":_"
            << iter->second << endl;
        ++iter;
    }
}

```

b.) Zur Implementierung der Funktion *ZensurEintragen* wurde der Index-Operator ("`[]`") verwendet. Ändern Sie diese Funktion, sodass hier wie in der Funktion *ZensurenAusgeben* Iteratoren verwendet werden.

c.) Als Container wurde zur Implementierung ein STL-Vektor verwendet. Es gibt jedoch eine bessere Container-Datenstruktur aus der STL, die zumindest bei größeren Studentenzahlen eine bessere Laufzeitperformance bringt. Welche ist das? Begründen Sie Ihre Entscheidung. Passen Sie anschließend die Zeilen mit den Typdefinitionen

```

typedef pair<string, double> Stud;
typedef vector<Stud> FH;
typedef vector<Stud>::iterator Iter;
typedef vector<Stud>::
    const_iterator ConstIter;

```

entsprechend an.

Hinweis: Bei Wahl des richtigen neuen Containers braucht die Funktion *ZensurenAusgeben* sogar gar nicht angepasst zu werden und funktioniert trotzdem genauso wie vorher. Bedenken Sie auch, dass es neben sequenziellen Containern auch assoziative Container gibt.

d.) Implementieren Sie nun die die Funktion *ZensurEintragen* nochmals mit dem neuen Container. Die anderen Funktionen sollen Sie **heute nicht** anpassen.