# Name:



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

| *Matrikelnummer:* | *Punktzahl:* |
|---|---|
| *Ergebnis:* | |

*Freiversuch* ☐ *F1* ☐ *F2* ☐ *F3* ☐

Ersatzklausur im WS 2005/06 :

## Informatik III — Lösungen —

| 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* diente lediglich bei der Klausurerstellung dem Zweck der Ausgabeumlenkung.

### Geplante Punktevergabe

| Punktziel | Im einzelnen | Pkte |
|---|---|---|
| Hausaufgaben: 10 P. | | |
| A1: 16 P. | (3+3+3+1+2+2+2) | |
| A2: 30 P. | (8+5+6+5+6 ) | |
| A3: 25 P. | (2+2+2+5+3+3+3+2+3) | |
| A4: 9 P. | 2+2+(1+1+1+2) | |
| A5: 20 P. | (6+5+4+5) | |
| Summe 100 P. | | |

### Aufgabe 1 : Textfragen zu Klassen

ca. 16 Punkte
3+3+3+1+2+2+2
Die folgende sehr kurze Klassedeklaration 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*)?
**Lösung:** Destruktor, Kopierkonstruktor, Zuweisungsoperator, Standardkonstruktor
**2.)** Geben Sie einen Programmausschnitt an, in dem die obigen implizit erzeugten Operatoren, Konstruktoren etc. **alle** verwendet werden.
**Lösung:**

```
void funk() {
   A a1;     // Standardkonstruktor
   A a2(a1); // Kopierkonstruktor
   a2 = a1;  // Zuweisungsoperator
}            // Destruktor
```

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

```
class A{
  public:
  A()  {;}
  ~A() {;}
  A(const A& a2): x(a2.x) {;}
  A& operator=(const A& a2) {
  x = a2.x; return *this;
  }
  public:
     double x;
};
```

**4.)** Wie viele Destruktoren kann eine Klasse maximal besitzen?
**Lösung:** Jede Klasse hat genau einen Destruktor.
**5.)** Deklarieren Sie eine Klasse, die zwei verschiedene Konstruktoren besitzt (Schnittstelle/Deklaration reicht).
**Lösung:**

```
class A{
  public
  A(int a);
  A();
};
```

**6.)** Geben Sie ein Beispiel für ein Funktionsobjekt an.
**Lösung:**

```
class A{
  public
  int operator() (int a);
};
```

**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);
}
```

**Lösung:** Das Ergebnis des Tests wird nicht nach außen weitergereicht, sodass eine (einfache) Auswertung des Testergebnisses durch den Aufrufer der Test-Funktion unmöglich ist. *TestPut* sollte somit ein `bool` zurückliefern.

### Aufgabe 2 : Softwareentwicklung / Testen

ca. 30 Punkte
Sie geben im Getränkemarkt Ihre leeren Flaschen und Kisten ab. Implementieren Sie eine C++-Funktion *BerechnePfand*, 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:

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 *BerechnePfand* (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 *BerechnePfand* 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 *BerechnePfand* selbst.

Beachten Sie nochmals, dass die Implementierung von *BerechnePfand* im Teil e.) nur ein kleiner Teil der Lösung ist.
Die zugehörige Hauptfunktion *main*, die die Tests ausführt und damit indirekt *BerechnePfand* 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;
   }
}
```

**Lösung:**
8+5+6+5+6

a.) Für 2 Bierflaschen und 3 Saftflaschen gibt es 61 Cent.
Für 48 Bierflaschen und 20 Saftflaschen gibt es 15,84 Euro.
Für 0 Bierflaschen und 6 Saftflaschen 2,40 Euro.

Durch die Tests wurde somit konkretisiert, dass es Kastenpfand nur für volle Kästen gibt.

b.) Listing 2: (./Code1/Tests/Berechne.h)

```
#ifndef BERECHNE_H
#define BERECHNE_H

/**
Die Funktion berechnet zunaechst wie
viel Pfand es fuer anzB Bierflaschen
und anzS Saftflaschen gibt.
Anschliessend wird ermittelt, wie
viel Bierkisten und Saftkisten
mit den anzB Bierflaschen
und anzS Saftflaschen gefuellt
werden koennen. Fuer
diese Anzahl wird dann noch
das Kistenpfand hinzuaddiert.
*/
double BerechnePfand(int anzB, int anzS);

#endif
```

c.) Listing 3: (./Code1/Tests/Test.h)

```
#ifndef TEST_H
#define TEST_H
  bool Test1();
  bool Test2();
  bool Test3();
#endif
```

Listing 4: (./Code1/Tests/Test.cpp)

```
#include "Test.h"
#include "Berechne.h"
#include <iostream>
#include <cmath>  // fabs
using namespace std;

const double EPS = 0.00001;

/**
Fuer 2 Bierflaschen und 3 Saftflaschen
gibt es 2*8 + 3*15 = 61 Cent
*/
bool Test1() {
   double pf = BerechnePfand(2, 3);
   if (fabs(pf - 0.61) < EPS) {
       return true;
       }
   else {
      return false;
   }
}

/**
Fuer 48 Bierflaschen und 20 Saftflaschen
gibt es 48*8 + 20*15  + (2+3) * 150 =
 384+300+750 = 15,84 Euro            */
bool Test2() {
   double pf = BerechnePfand(48, 20);
   if (fabs(pf - 14.34) < EPS) {
       return true;
       }
   else {
      return false;
   }
}

/**
Fuer 0 Bierflaschen und 6 Saftflaschen
gibt es 6*15  + (1) * 150 = 2,4 Euro */
bool Test3() {
   double pf = BerechnePfand(0, 6);
   if (fabs(pf - 2.40) < EPS) {
       return true;
       }
   else {
      return false;
   }
}
```

e. Listing 5: (./Code1/Tests/Berechne.cpp)

```
#include "Berechne.h"


const double FlBier = 0.08;
const double FlSaft = 0.15;
int BierKastenGr = 24;
int SaftKastenGr = 6;
const double KastenPfand = 1.5;

double BerechnePfand(int anzB, int anzS){
   double b = anzB * FlBier;
   double s = anzS * FlSaft;
   double ka = (anzB / BierKastenGr +
      anzS / SaftKastenGr) *
        KastenPfand;
   return b + s + ka;
  }
```

## 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␣";
}
```

–

**Lösung:**
(2+2+2+5+3+3+3+2+3)

```
funk1
+A 11 +A 4 -A 4 -A 11
funk2
+A 3 +A 5 Ende -A 5
funk3
+A 3 -A 3
funk4
+A 3 +A 2 Weiter -A 2 Weiter -A 3
funk5
+A 11 17-A 17
funk6
+A 11 -A 17 11-A 11
funk7
+A 11 +B +A 8 +B -B -A 8 -B -A 11
funk8
Ende
funk9
+A 11 +B +A 11 +B Ende -B -A 11 -B -A 11
```

### Aufgabe 4 : eXtreme Programming und Referenzen

ca. 9 Punkte

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

**Lösung:**
siehe unten

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?

**Lösung:**
Programmierstandard, 2 Leute ein Bildschirm, Gemeinsame Verantwortung, Testen

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;
}
```

**Lösung:**
2+2+(1+1+1+2)

```
funk1
2 3
funk2
14 13
funk3
14 13
funk4
2 6
```

### Aufgabe 5 : STL

ca. 20 Punkte
Geben Sie Ihre Lösungen auf einem Extrablatt an.
6+5+4+5
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 */
// Typedefinitionen 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 ist " << 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;
   }
}
```

**Lösung:**

```
Neu ist Anke Piep
Fritz Mey: 2.3
Karl Pey: 5
Elke Mopp: 1
Anke Piep: 4
```

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.

**Lösung:**

```
void ZensurEintragen(const string& n,
     double z, FH& fh ) {
   Iter iter = fh.begin();
   while (iter != fh.end()) {
      if (iter->first == n){
         iter->second = z;
         return;
      }
      ++iter;
   }
   datei << "Neu ist " << n <<endl;
   fh.push_back(make_pair(n, z));
}
```

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 Laufzeitperformanz 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.
**Lösung:** Eine Map ist der bessere Container, weil man hier beim Einfügen nicht linear suchen muss, ob der Student schon vorhanden ist oder nicht. Man kann mit binärer Suche den Aufwand auf eine logarithmische Zeitkomplexität reduzieren.

```
// Typedefinitionen zum Anpassen
typedef map<string, double> FH;
typedef FH::iterator Iter;
typedef FH::const_iterator ConstIter;
```

d.) Implementieren Sie nun die die Funktion *Zensur-Eintragen* nochmals mit dem neuen Container. Die anderen Funktionen sollen Sie **heute nicht** anpassen.
**Lösung:**

```
/* n wird in der Map namen gesucht.
   Falls es gefunden wird, wird die
   Zensur auf z geaendert.
   Falls n nicht gefunden wird,
   werden n und z neu in die Map
   eingetragen und eine Meldung
   nach datei ausgegeben.
*/
void ZensurEintragen(const string& n,
     double z, FH& fh)
 {
   Iter iter = fh.find(n);
   if (iter == fh.end()) {
      datei << "Neu␣ist␣" << n <<endl;
      fh[n] = z;
   }
   else {
      iter->second = z;
   }
}
```