Mündliche Prüfungen für Informatik III sind zusammen mit Prof. Isernhagen am Dienstag und Mittwoch, den 28. bzw. 29. Januar 2003, geplant

a.) 2 mal sehr gut,

b.) 11 mal gut,

c.) 9 mal befriedigend,

d.) 15 mal ausreichend,

e.) 30 mal mangelhaft (von 67, also 45 %).

13 F2-Prüfungen, 0 F3-Prüfungen
bestes Ergebnis 78 Punkte von 83 Punkten

a.) ab 75,5 Punkte: 1,0

b.) ab 71,5 Punkte: 1,3

c.) ab 67,5 Punkte: 1,7

d.) ab 63,5 Punkte: 2,0

e.) ab 59,5 Punkte: 2,3

f.) ab 55,5 Punkte: 2,7

g.) ab 51,5 Punkte: 3,0

h.) ab 47,5 Punkte: 3,3

i.) ab 43,5 Punkte: 3,7

j.) ab 39,5 Punkte: 4,0

Name:

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

Matrikelnummer:	Punktzahl:
Ergebnis:	

Freiversuch ☐ F1 ☐ F2 ☐ F3 ☐

Klausur im WS 2002/03 :

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 den meisten Fällen hier auf dem Aufgabenblatt angegeben werden. Sie dürfen aber auch Ihre Lösungen, falls erforderlich, auf separaten Aufgabenblättern 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	
A1: 6 P	(1+1+2+2)	
A2: 11 P	(4+4+3(+2SP))	
A3: 8 P	4+4	
A4: 13 P	(7(+2SP)+6)	
A5: 45 P	6+4+5+3+4+4+5+7(+3SP)+4(+2SP)	
Summe 83 P		

Name:

Aufgabe 1 : Parameterübergabe an Funktionen

ca. 6 Punkte
Was wird bei Ausführung des folgenden `main`-Programms in `datei` ausgegeben?

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

ofstream datei("Parameter.txt", ios::out);

void funk1(int m, int x)
   {
   m=0;
   x=14;
   }

void funk2(int& m, int& x)
   {
   m=0;
   x=14;
   }

void funk3(int* m, int* x)
   {
   *m=0;
   *x=0;
   }

void funk4(int* m, int* x)
   {
   m=0;
   x=0;
   }

int main()
   {
   int i;
   int j;

   i=67; j=88;
   funk1(i,j);
   datei << i << " " << j << endl;

   i=67; j=88;
   funk2(i,j);
   datei << i << " " << j << endl;

   i=67; j=88;
   funk3(&i, &j);
   datei << i << " " << j << endl;

   i=67; j=88;
   funk4(&i, &j);
   datei << i << " " << j << endl;

   return 0;
   }
```

Lösung:
Bewertung: 1+1+2+2

```
67 88
0 14
0 0
67 88
```

Aufgabe 2 : Das Schlüsselwort `const`

ca. 11 Punkte

a.) Wo ist in der folgenden `main`-Funktion ein Syntaxfehler aufgrund von Problemen mit dem Schlüsselwort `const`?
Begründen Sie kurz ihre Entscheidung.

Lösung:
Bewertung: 4 + 4 + 3
Jeweils 1 Punkt Abzug, wenn Begründung bei tatsächlichem Syntaxfehler fehlt.

```
'funk2' : Konvertierung des Parameters 1 von
       'const int' in 'int &' nicht moeglich
        Durch die Konvertierung gehen Qualifizierer
       verloren
'f1' : this-Zeiger kann nicht von 'const class A' in
       'class A &' konvertiert werden
        Durch die Konvertierung gehen Qualifizierer
       verloren

b1:   L-Wert gibt ein konstantes Objekt an
```

```
class A{
   public:
      void f1();
      void f2() const;
   };

void funk1(const int m);
void funk2(int& m);
void funk3(const int f[]);
void funk4(const int f[]);

int main()
   {
   int m=87;
   const int k=88;
   int feld[444];

   funk1(m);
   funk2(k);
   funk3(&k);
   funk4(feld);

   A a1;
   const A a2;

   a1.f1();
   a1.f2();
   a2.f1();
   a2.f2();

   return 0;
```

```
   }

class B {
   public:
      void b1(const int& a1)const {k=a1;};
      int b2(int& a2)const {return k*a2;};
      void b3(int i, int w)const  {pi[i]=w;};

   private:
      int i;
      int k;
      int* pi;
   };
```

b.) Wo ist in der Deklaration der Klasse `B` ein Syntaxfehler, weil eine nicht konstante Methode als `const` vereinbart wurde?
Begründen Sie kurz ihre Entscheidung.

Die Methode `B::b3` ist zwar syntaktisch korrekt. Sie sollte aber nicht als konstante Methode vereinbart werden, da der Zeiger, hier wie ein Feld verwendet wird und dieses Feld in der Methode verändert wird. (2 SP)

Aufgabe 3 : Include-Wächter, Funktionen

ca. 8 Punkte

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

#include "IncDec.h"
ofstream datei("IncDec.txt", ios::out);

void IncDecPostPrae()
   {
   int wert=10;
   int wert2=wert++;
   datei << "wert="<<wert << " wert2="
      << wert2 <<endl;
   }

void IncDecFunk()
   {
   int wert=10;
   // Hier wird die zu implementierende
   //Funktion   C h a n g e  verwendet
   int wert2 = Change(wert); // entspricht wert++
   datei << "wert="<<wert << " wert2="
      << wert2 <<endl;
   }

main()
   {
   IncDecPostPrae();
   IncDecFunk();
   return 0;
   }
```

Implementieren Sie die in der Funktion `IncDecFunk` verwendete Funktion `Change`, sodass die beiden Funktionen `IncDecPostPrae` und `IncDecFunk` die gleiche Ausgabe in die Datei `datei` produzieren.

Die Funktion ist in der Header-Datei `IncDec.h` zu deklarieren und in der Quellcode-Datei `IncDec.cxx` zu definieren. Vergessen Sie nicht den Include-Wächter.
Header-Datei:

```
#ifndef INC_DEC_H
#define INC_DEC_H

   int Change(int& w); // w++

#endif /* INC_DEC_H */
```

Bewertung:
Include-Wächter: 2 Punkte
Deklaration: 2 Punkte
Include des Header in der Source-Datei: 1 Punkt
Wiederholung der Deklaration: 0,5 Punkte Algorithmus richtig: 2,5 Punkte
ohne Call-By-Reference mindestens 1,5 Punkte Abzug
Source-Datei:

```
#include "IncDec.h"
int Change(int& w) // w++
   {
   int tmp(w);
   w=w+1;
   return tmp;
   }
```

Aufgabe 4 : STL, Operatoren

ca. 7+6 Punkte
Im folgenden Programm wird in der Funktion `Print` ein Ausgabe-Operator für die Klasse `std::pair` verwendet. Dieser ist nicht Bestandteil der STL. Er ist daher von Ihnen in der Datei `PairOutput.h` zu vereinbaren. Der Operator soll dann für beliebige Paare funktionieren, sofern die beiden Elemente des Paares jeweils selbst den Ausgabeoperator vereinbart haben.

Für die folgenden Paare:

```
pair<int,int>      p1(7,3);
datei << p1 << " ";
pair<string,int> p2("Helmke",50);
datei << p2 << endl;
pair<float, double> p3(7.2f, 8.3);
datei << p3 << endl;
```

soll die Ausgabe wie folgt aussehen:

```
<7,3> <Helmke,50>
<7.2,8.3>
```

a.) Implementieren Sie den Ausgabeoperator für die STL Schablonenklasse `pair`, d.h. den Inhalt der Datei `PairOutput.h`, welche die Deklaration **und** die Definition des Ausgabeoperators enthält.

Lösung:

```
#ifndef PairOutput_h
#define PairOutput_h
#pragma warning( disable : 4663 )
#pragma warning( disable : 4786 )
#pragma warning( disable : 4097 )
#pragma warning( disable : 4514 )

#include <fstream>
#include <utility>
using namespace std;

template <class T1, class T2>
ostream& operator<<
   (ostream& str, const pair<T1,T2>& p)
   {
   str << "<" << p.first
        << "," << p.second << ">";
   return str;
   }

#endif /* PairOutput_h */
```

Bewertung:
1 SP, wenn Include-Wächter vorhanden
template mit 2 Parametern: 1,5 Punkte
Schnittstelle von operator: 2 Punkte
const von pair und Referenz 1 Punkt
Algorithmus: 2 Punkte
return str: 0,5 Punkt
include von utility und fstream und using namespace: + 2 SP
include von iostream, statt fstream: 0,5 SP

Hier das Programm:

```
// fuer Ausgabe-Operator der Klasse pair,
// der von Ihnen zu implementieren ist
#include "PairOutput.h"

#include <fstream>
#include <list>
#include <algorithm>
#include <string>
#include <utility>
using namespace std;

ofstream datei("STL.txt", ios::out);

typedef pair<string,int>  StrIntPair;
void Print(const StrIntPair& v)
   {
   datei << v << " ";
   }

int main()
{
list<StrIntPair> li1;
li1.push_back(make_pair(
   static_cast<string>("He"), 30));
li1.push_back(make_pair(
   static_cast<string>("Ha"), 20));
li1.push_front(make_pair(
   static_cast<string>("Ho"), 10));

for_each( li1.begin(),li1.end(),Print );
datei << endl;
for_each( li1.begin(), li1.end()++, Print );
datei << endl;
for_each( ++li1.begin(), --li1.end(), Print );
datei << endl;
return 0;
}
```

b.) Was gibt das obige `main`-Programm in die Datei `datei` aus.

Lösung:

```
<Ho,10> <He,30> <Ha,20>
<Ho,10> <He,30> <Ha,20>
<He,30>
```

Aufgabe 5 : Konstruktor/Destruktor und Polymorphie

ca. 45 Punkte
Beachten Sie, dass hier im Unterschied zur Vorlesung drei Klassen verwendet werden.

a.) Was gibt das folgende `main`-Programm durch Ausführung der Funktion `funk1` in die Datei `datei` aus?

Bewertung: (6)
Reihenfolge +F, +K, bzw. +F, +K, +R jeweils 1 Punkt
Reihenfolge -K, -F, bzw. -R, -K, -F: 1 Punkt
Zerstörung des Ringes vor Zerstörung des Kreises 1 Punkt
Erzeugung **und** Zerstörung von einer Figur beim Kreis bzw. einer Figur und eines Kreises beim Ring 2 Punkte

b.) Was gibt das folgende `main`-Programm durch Ausführung der Funktion `funk2` in die Datei `datei` aus?

Bewertung: (4)
Aufruf des Standardkonstruktors von Figur: 1,5 Punkt
Aufruf des Standardkonstruktors von Kreis: 1,5 Punkte
Rest 1 Punkt

c.) Was gibt das folgende `main`-Programm durch Ausführung der Funktion `funk3` in die Datei `datei` aus?

Bewertung: (5)
Keine Ausgabe für Feld: 1,5 Punkt
3 mal Aufruf von Konstruktorpaar: 1,5 Punkte
3 mal Aufruf von Destruktorpaar: 1,5 Punkte
Rest 0,5 Punkte

d.) Was gibt das folgende `main`-Programm durch Ausführung der Funktion `funk4` in die Datei `datei` aus?

Bewertung: (3)
Aufruf von Konstruktorpaar: 1 Punkt
Aufruf von Destruktorpaar: 2 Punkt

e.) Was gibt das folgende `main`-Programm durch Ausführung der Funktion `funk5` in die Datei `datei` aus?

Bewertung: (4)
Aufruf von Konstruktorpaar: 2 Punkte
Aufruf von Destruktor ohne Destruktor von Kreis: 2 Punkte

f.) Was gibt das folgende `main`-Programm durch Ausführung der Funktion `funk6` in die Datei `datei` aus?

Bewertung: (4)
Für jede richtige Zahl: 1 Punkt

g.) Was gibt das folgende `main`-Programm durch Ausführung der Funktion `funk7` in die Datei `datei` aus?

Bewertung: (5)
erkennen, dass bei nicht virtueller Methode nie Polymorphie passiert: 2 Punkte
erkennen dass bei polymorpher Methode dynamisches Binden passiert, einmal Methode von Ring einmal von Kreis: 2 Punkte
Rest 1 Punkt

h.) Was gibt das folgende `main`-Programm zwischen den Ausgaben `Aufgabe h` und `Ende` in die Datei `datei` aus (Ausgaben erfolgen in der Funktion `funk8` **und** in `main`)?

Bewertung: (7 + 3 SP)
keine Ausgabe der Copy-Konstruktoren, wenn Destruktor-Aufruf dafür vorhanden: 2 Punkte
Erkennen, dass in der Funktion durch Call-By-Value keine Polymorphie erfolgt: 2 Punkte
Zerstörung des Kreises nach der Funktion: 1 Punkte (+1 SP)
Zerstörung des Ringes nach Ausgabe von `Nach Aufruf` 2 Punkte (+1 SP)

i.) Warum kann von den Klassen `Figur` bzw `Ring` kein Array gebildet werden?

Bewertung: (4 +2 SP)
Figur: 2,5 Punkte Ring: 1,5 Punkte
SP für gute Begründungen

j.) In zwei Funktionen wird Heap-Speicher angefordert. Wie viel Prozent dieses Heapspeichers wird wieder freigegeben? Begründen Sie Ihre *Rechnung.*

Bewertung: 3
Es werden 2mal jeweils 2 Integer angefordert, also 16 Byte. Von diesen wird aber ein Kreis und damit ein Integer nicht wieder freigegeben, d.h. 12 Byte werden explizit wieder freigegeben, d.h. 12/16*100 = 75 Prozent werden explizit wieder freigegeben.

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

ofstream datei("ConstrDestr.txt", ios::out);

class Figur
   {
   int dummy;
   public:
      Figur()  {datei << "+F";}
      ~Figur() {datei << "-F";}
      void Umfang() const {datei << "F" << 0;}
      virtual void Flaeche()const=0;
   };

class Kreis: public Figur
   {
   protected:
      enum {PI=3};
      int rad;  // Aussenkreis-Radius
   public:
      Kreis(): rad(4)
         {
         datei << "+K" << rad << " ";
         }
      Kreis(int r)
         {
         rad=r;
         datei << "+K" << rad << " ";
         }
      ~Kreis()
         {
         datei << "-K" << rad << " ";
         }
      void Umfang()const
         {
         datei << "K" << 2*PI*rad;
         };
      virtual void Flaeche()const
         {
         datei << "K" << PI*rad*rad;
         };
   };

// Kreis mit Loch
class Ring: public Kreis
   {
   int rRad;  // Innenkreis-Radius
   public:
      Ring(int r)
         {
```

```
        rRad=r;
        datei << "+R" << rRad << " ";
        }
     Ring(int r, int rr):Kreis(r), rRad(rr)
        {
        datei << "+R" << rRad << " ";
        }
     ~Ring()
        {
        datei << "-R" << rRad << " ";
        }
     void Umfang()const
        {
        datei << "R" << 2*PI*(rad-rRad);
        }
     virtual void Flaeche()const
        {
        datei << "R" << PI*(rad*rad - rRad*rRad);
        }
  };
```

Hauptprogramm:

Beachten Sie, dass hier drei Klassen verwendet werden.

```
#include "Figur.h"

void funk1()
   {
   Kreis k(3);
   Ring ri(4,2);
   datei << endl;
   }

void funk2()
   {
   Kreis k;
   Ring ri(1);
   datei << endl;
   }

void funk3()
   {
   Kreis* k1[5];
   k1[0]=NULL;
   Kreis  k2[3];
   }

void funk4()
   {
   // Heapspeicherverwendung
   Kreis* p = new Kreis(2);
   delete p;
   }

void funk5()
   {
   // Heapspeicherverwendung
   Figur* p = new Kreis(2);
   delete p;
   }


// globale Variablen
Kreis k(1);
Ring ring(3,1);

void funk6()
   {
   k.Umfang();  datei << " "; k.Flaeche();
   datei<< " ";
   ring.Umfang();  datei << " "; ring.Flaeche();
   datei << endl;
   }

void funk7()
   {
   Figur* vec[2];
   vec[0]=&k;
   vec[1]=&ring;

   for (int i=0;i<2;i++)
      {
      vec[i]->Umfang(); datei << " ";
      vec[i]->Flaeche(); datei << " ";
      }
   }

void funk8(Kreis f)
   {
   f.Umfang(); datei << " ";
   f.Flaeche(); datei << " ";
   }


int main()
   {
   datei << "\nAufgabe a" << endl;
   funk1();
   datei << endl << endl;

   datei << "Aufgabe b" << endl;
   funk2();
   datei << endl << endl;

   datei << "Aufgabe c" << endl;
   funk3();
   datei << endl << endl;

   datei << "Aufgabe d" << endl;
   funk4();
   datei << endl << endl;

   datei << "Aufgabe e" << endl;
   funk5();
   datei << endl << endl;

   datei << "Aufgabe f" << endl;
   funk6();
   datei << endl << endl;

   datei << "Aufgabe g" << endl;
   funk7();
   datei << endl << endl;

   datei << "\nAufgabe h" << endl;
      {
      Ring r(ring);
      datei << "Vor Aufruf" << endl;
      funk8(r);
```

```
    datei << "\nNach Aufruf" << endl ;
    }
    datei << endl << "Ende" << endl;

  datei << "\nAufgabe i" << endl;
  /* geht nicht!!!
  Ring r_feld[10];
  Figur f_feld[10];
  */

  return 0;
  }
```

Lösung:

```
+F+K1 +F+K3 +R1
Aufgabe a
+F+K3 +F+K4 +R2
-R2 -K4 -F-K3 -F

Aufgabe b
+F+K4 +F+K4 +R1
-R1 -K4 -F-K4 -F

Aufgabe c
+F+K4 +F+K4 +F+K4 -K4 -F-K4 -F-K4 -F

Aufgabe d
+F+K2 -K2 -F

Aufgabe e
+F+K2 -F

Aufgabe f
K6 K3 R12 R24

Aufgabe g
F0 K3 F0 R24

Aufgabe h
Vor Aufruf
K18 K27 -K3 -F
Nach Aufruf
-R1 -K3 -F
Ende

Aufgabe i
-R1 -K3 -F-K1 -F
```

Aufgabe i:

Die Klasse `Figur` ist aufgrund der **rein virtuellen** Methode `Figur::Flaeche` abstrakt. Daher kann keine Instanz dieser Klasse gebildet werden und damit auch kein Array.
Die Klasse `Ring` besitzt keinen Standardkonstruktor. Der Default-Standardkonstruktor wird nicht erzeugt, da schon andere Konstruktoren der Klasse `Ring` existieren.