Name:

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

Matrikelnummer:	*Punktzahl:*
Ergebnis:	

Freiversuch ☐ *F1* ☐ *F2* ☐ *F3* ☐

Klausur im SS 2007 :

☐
☐

Programmierkonzepte
Informatik III

Medieninformatik ☐ *Praktische Informatik* ☐ *Technische Informatik* ☐

Informatik B. Sc. ☐ *Technische Informatik B. Sc.* ☐

Hilfsmittel sind bis auf Computer, Handy etc. erlaubt !	Bitte Aufgabenblätter mit abgeben !
Austausch von Hilfsmitteln mit Kommilitonen ist **nicht** erlaubt !	

Die Lösungen können größtenteils 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 manchmal 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

Planen Sie pro Punkt etwas mehr als eine Minute Aufwand ein.

Punktziel	Sonderpunkte	erreicht
Übungsaufgaben: 10 P.		
A1: 15 (2+2+2+1+3+3+2) P.		
A2: 19 (2+3+2+6+6) P.		
A3: 21 (3+4+2+3+4+2+3) P.		
A4: 12 (3+5+4) P.		
A5: 21 (1+5+2+3+2+3+2+3) P.		
A6: 12 (2+1+1+3+3+2) P.		
Summe 100 P.		

Aufgabe 1 : Stack-Heapspeicher

ca. 15 (2+2+2+1+3+3+2) Punkte

a) Veranschaulichen Sie grafisch die Stack-Speicherbelegung des folgenden Programmfragments am Funktionsende.

```
void f1() {
   double d = 22.4;
   int i = 4;
   }
```

(*——- Lösung hier. ———*)

b) Veranschaulichen Sie grafisch die Stack-Speicherbelegung des folgenden Programmfragments am Funktionsende.

```
void f2() {
   int i = 4;
   int* pi = &i;
   }
```

(*——- Lösung hier. ———*)

c) Veranschaulichen Sie grafisch die Stack- **und** Heap-Speicherbelegung des folgenden Programmfragments am Funktionsende.

```
void f3() {
   int i = 4;
   int* pi = new int;
   // Ihr Code
   }
```

(*——- Lösung hier. ———*)

d) Erweitern Sie die Funktion `f3`, sodass in die Speicherstelle, auf die `pi` verweist, eine `24` hineingeschrieben wird.
(*——- Lösung hier. ———*)

e) Veranschaulichen Sie grafisch die Stack-Speicherbelegung des folgenden Programmfragments am Funktionsende.

```
void f4() {
   int arr[4];
   for (int i=0; i<4;++i) {
      arr[i] = 22;
      }
   }
```

(*——- Lösung hier. ———*)

f) Veranschaulichen Sie grafisch die Stack- **und** Heap-Speicherbelegung des folgenden Programmfragments am Funktionsende.

```
void f5() {
   int i(11);
   int* arr = new int[4];
   for (int i=0; i<4;++i) {
      arr[i] = 22;
      }
   }
```

(*——- Lösung hier. ———*)

g) Welchen *Fehler* enthält die Funktion `f5` an ihrem Ende?

Aufgabe 2 : Klassen

ca. 19 (2+3+2+6+6) Punkte
Gegeben ist die folgende Deklaration der Klasse `Vogel`:

```
class Vogel {
public:
   Vogel(int e) {eier = e;}
   Vogel(const Vogel& v2);
   Vogel& operator=(const Vogel& v2);
   // set− und get−Methoden, siehe Aufgabe
private:
   int eier; // Anzahl Eier
   double gewicht; // in Gramm
   };
```

a.) Warum würde das folgende Codefragment beim Übersetzen einen Syntaxfehler ergeben?

```
Vogel vogelFeld[22];
```

(*——- Lösung hier. ———*)

b.) Implementieren Sie die beiden Methoden `getEier` und `setGewicht` direkt in der Klassenschnittstelle selbst (`inline`). Beachten Sie die korrekte Verwendung des Schlüsselworts `const`.
(*——- Lösung hier. ———*)

c.) Geben Sie einen Include-Wächter für die Header-Datei an.
(*——- Lösung hier. ———*)

d.) Implementieren Sie nun den Kopierkonstruktor und Zuweisungsoperator der Klasse `Vogel` (cpp-Datei). Vermeiden Sie dabei überflüssigen Code.
(*——- Lösung hier. ———*)

e.) Implementieren Sie jeweils einen Test für den Kopierkonstruktor und einen für den Zuweisungsoperator.
(*——- Lösung hier. ———*)

Aufgabe 3 : Konstruktoren

ca. 21 (3+4+2+3+4+2+3) Punkte

Im Folgenden dürfen Sie Ihre Ausgaben direkt unter den Funktionen oder auf einem Extra-Blatt notieren. Beachten Sie, dass in manchen Fällen zunächst eine **grafische Veranschaulichung** der Speicherbelegung gewünscht ist.

Gegeben ist die folgende Definition einer Klasse:

```
class Vogel {
public:
   Vogel(int e=4) {datei << " +V" << e; eier = e;}
   ~Vogel() {datei << " -V" << eier;}
private:
   int  eier; // Anzahl Eier
   };
```

a.) Welche Ausgabe erzeugt der Aufruf der Funktion `funk1` in die Datei?

```
void funk1() {
   datei << "\nfunk1";
   Vogel v1(5);
   Vogel v2(v1);
   }
```

(*——- Lösung hier. ———*)

b.) Veranschaulichen Sie die Speicherbelegung im Stack- **und** im Heap-Programmspeicher nach Ausführung der Anweisung `// 1` grafisch.

```
void funk2() {
   datei << "\nfunk2";
   Vogel* v1 = new Vogel();
   Vogel* v2 = new Vogel(11);
   Vogel* v3 = v1;
   v2 = v1; // 1
   delete v2;
   datei << " Ende";
   }
```

(*——- Extrablatt verwenden ———*)

c.) Welche Ausgabe erzeugt der Aufruf der Funktion `funk2` in die Datei?
(*——- Lösung hier. ———*)

Gegeben sei ferner die folgende Unterklasse:

```
class Spatz: public Vogel{
public:
   Spatz(int e, double g): Vogel(e), gew(g)
      {datei << " +S" << gew;}
   ~Spatz() {datei << " -S" << gew;}
private:
   double gew;
   };
```

d.) Welche Ausgabe erzeugt der Aufruf der Funktion `funk3` in die Datei?

```
void funk3() {
   datei << "\nfunk3";
   Spatz s1(4, 22.4);
   datei << " Ende";
   }
```

(*——- Lösung hier. ———*)

e.) Veranschaulichen Sie die Speicherbelegung im Stack- **und** im Heap-Programmspeicher nach Ausführung der Anweisung `// 2` grafisch.

```
void funk4() {
   datei << "\nfunk4";
   Spatz s1(1, 2.4);
   Vogel v2(5);
   v2 = s1; // 2
   datei << " Ende";
   }
```

(*——- Extrablatt verwenden ———*)

f.) Welche Ausgabe erzeugt der Aufruf der Funktion `funk4` in die Datei?
(*——- Lösung hier. ———*)

g.) Welche Ausgabe erzeugt der Aufruf der Funktion `funk5` in die Datei?

```
void funk5() {
   datei << "\nfunk5";
   Vogel* v1 = new Spatz(3, 4.4);
   datei << " Ende";
   delete v1;
   }
```

Aufgabe 4 : Polymorphie

ca. 12 (3+5+4) Punkte

Im Folgenden dürfen Sie Ihre Ausgaben direkt unter den Funktionen oder auf einem Extra-Blatt notieren.

Gegeben sei folgende Klasse mit ihren Unterklassen:

```
class Vogel {
public:
   virtual void ton() {datei << " kraechz ";}
   int federn() {return 10;}
   };

class Spatz: public Vogel{
public:
   virtual void ton() {datei << " piep ";}
private:
   };

class Pinguin: public Vogel{
public:
   virtual void ton() {datei << " pst ";}
   int federn() {return 0;}
private:
   };
```

a.) Zeichnen Sie ein (einfaches) UML-Klassendiagramm der drei Klassen.
(*——- Lösung hier. ———*)

b.) Welche Ausgabe erzeugt der Aufruf der Funktion `funk6` in die Datei? Geben Sie zunächst jeweils den dynamischen Typ der Variablen an, auf die die Zeiger `arr[0]` bis `arr[3]` verweisen.

```
void funk6() {
   datei << "\nfunk6";
   Vogel* v1 = new Spatz();
   Spatz* s2 = new Spatz();
   Vogel* v3 = new Vogel();
   Vogel* p4 = new Pinguin();
   // Zeigerarray initialisieren
   Vogel* arr[] = {v1, s2, v3, p4};
   for (int i=0; i<4; ++i) {
      datei << "\n" << i << " "
         << arr[i]->federn();
      arr[i]->ton();
      delete arr[i];
      }
   }
```

(*——- Lösung hier. ———*)

c.) Welche Ausgabe erzeugt der Aufruf der Funktion `funk7` in die Datei?

```
void funk7() {
   datei << "\nfunk7";
   Vogel v1;
   Spatz s2;
   Pinguin p3;
   datei << "\n" << v1.federn(); v1.ton();
   datei << "\n" << s2.federn(); s2.ton();
   datei << "\n" << p3.federn(); p3.ton();
   }
```

(*——- Lösung hier. ———*)

Aufgabe 5 : Textfragen, C++

ca. 21 (1+5+2+3+2+3+2+3) Punkte

a.) Womit kann man dynamisches Binden auch in einer nicht objektorientierten Sprache wie C implementieren (ein *richtiges* Wort reicht)?
(*——- Lösung hier. ———*)

b.) Gegeben sei die folgende Deklaration einer Klasse `Liste`:

```
class Liste {
public:
   int anzahl;
   // Methoden zum Einfügen von Elementen
   void fuegeEinPosition_1(int elem);
   void fuegeEinPosition_2(int elem);
   void fuegeEinPosition_3(int elem);
   void fuegeEinPosition_4(int elem);
   void fuegeEinPosition_5(int elem);
   void fuegeEinPos_N(int elem, int pos);
   // Löschen von Elementen
   void loescheErstes();
   void loescheZweites();
   void loescheDrittes();
   void loescheViertes();
   void loescheLetztes();
   void loeschePos_N(int pos);

   void erhoeheAlleWerteUm_2_undLoescheZweites();
   int f() const;
   void g(int w, int w2);
private:
   int daten[1000];
   };
```

Beschreiben Sie bitte kurz, warum diese Klassendeklaration ein ganz schlechtes Beispiel für die Implementierung einer guten Klassenschnittstelle ist (mehrere Gründe).
(*——- Extrablatt verwenden ———*)

c.) Geben Sie grafisch ein Beispiel für einen Netzplan mit 3 Vorgängen und ihren verschiedenen Dauern an. Geben Sie anschließend für jeden Vorgang den spätesten Anfang und das früheste Ende an (Genau beachten, welche der vier Werte gewünscht sind).
(*——- Extrablatt verwenden ———*)

d.) Geben Sie ein Beispiel für die Deklaration (Definition nicht erforderlich) einer Klasse mit einer minimalen Standard-Schnittstelle an.
(*——- Extrablatt ———*)

e.) Geben Sie ein Beispiel für eine ganz kurze abstrakte C++-Klasse an.
(*——- Lösung hier. ———*)

f.) Implementieren Sie **eine** C++-Funktion `viel`,

- die das Maximum von `a` und b
- **und gleichzeitig** den Durchschnitt von `a` und b

liefert.
(*——- Lösung hier. ———*)

g.) Warum sollten man beim Vergleich von Werten statt

if (a == 17) besser
if (17 == a)

verwenden?
(*——- Lösung hier. ———*)

h.) Implementieren Sie einen Test für die Funktion `viel`.
(*——- Lösung hier. ———*)

Aufgabe 6 : Textfragen, Team

ca. 12 (2+1+1+3+3+2) Punkte

a.) Warum ist redundanter Code (mehrfach vorhandener identischer Code) von Nachteil für den weiteren Projektfortschritt?
(*——- Lösung hier. ———*)

b.) Ihr Programm enthält einige Programmfragmente mehrfach. Was schlagen Sie vor, um diesen Missstand zu beheben (ein *richtiges* Wort reicht)?
(*——- Lösung hier (wenige Worte). ———*)

c.) Wie kann man sich (ein wenig) davor schützen, dass eine eben noch vorhandene Funktionalität plötzlich nach einer Änderung nicht mehr funktioniert und dieser Missstand erst Monate später entdeckt wird?
(*——- Lösung hier, wenige Worte ———*)

d.) Das Team hatte sich in der letzten Iteration (Dauer 20 Arbeitstage) Aufgaben im Umfang von 12 idealen Tagen vorgenommen. Erledigt wurden allerdings nur Aufgaben im Umfang von 10 idealen Tagen. Zusätzlich wurden ungeplante Aufgaben im Umfang von 2 idealen Tagen erledigt. Berechnen Sie den Load Factor (Rechenweg angeben).
(*——- Lösung hier. ———*)

Die neue Iteration dauert 50 Arbeitstage. Wie viel Aufgaben (Angabe in idealen Tagen) sollte sich das Team vornehmen, wenn der Load Factor der Iteration als Entscheidungsgrundlage verwendet wird? (Rechenweg angeben)
(*——- Lösung hier. ———*)

e.) Welche Vorteile ergeben sich, wenn man die Tests noch vor Beginn der Implementierung der eigentlichen Funktionalität spezifiziert?
(*——- Lösung hier (ein paar Sätze, mehrere Vorteile) ———*)

f.) Erklären Sie kurz und knapp: Was ist *Refactoring*?
(*——- Lösung hier. ———*)