

Name:

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

Matrikelnummer:	Punktzahl:		
Ergebnis:			
Freiversuch <input type="checkbox"/>	F1 <input type="checkbox"/>	F2 <input type="checkbox"/>	F3 <input type="checkbox"/>

Klausur im SS 2004 :
Informatik II, C++-Teil

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

Notieren Sie Ihre Lösung auf separaten Blättern.

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.

Lesen Sie sich die Aufgabenstellung von jeder Aufgabe zunächst einmal komplett durch.

Geplante Punktevergabe

Punktziel	erreicht
A1: 8 P.	
A2: 8 P.	
A3: 28 P.	
A4: 6 P.	
Summe 50 P.	

Aufgabe 1 : Schleifen

ca. 8 Punkte

Was gibt das folgende Programm in die Datei datei aus?

Listing 1: (./Code1/schleife.cxx)

```
#include <fstream>
using namespace std;
ofstream datei("Schleife.txt");

int main() {
    for (int i=2; i<5; ++i) {
        datei << (i+1) << "\n";
    }

    datei << endl;
    for (int j=2; j<10; j=j+2) {
        datei << j << "\n";
    }

    datei << endl;
    int k=7;
    while (k) {
        datei << k-- << "\n";
    }

    datei << endl;
    int m=7;
    do {
        datei << --m << "\n";
    }
    while (m-- > 0);
    return 0;
}
```

Aufgabe 2 : Funktionen mit Parametern

ca. 8 Punkte

Implementieren Sie eine Funktion `Berechne` mit zwei positiven `int`-Eingangsparametern `a` und `b` (beide ungleich Null) und ggf. weiteren Parametern, die Folgendes zurückliefert:

- $a + b$,
- b Prozent von a ,
- $b^4 - \frac{a*a}{4}$.

Aufgabe 3 : Funktionen und Records

ca. 28 Punkte

Gegeben ist das folgende Programm

Listing 2: (./Code1/mainAufg.cxx)

```
#include <fstream>
using namespace std;
ofstream datei("Ausgabe.txt");

void func(double x[]) {
    int i;
    // Berechnung der Streckenlaenge von
    // x[0] bis x[5] aufaddiert
```

```
double len = 0.0;
for (i=0; i<5; ++i) {
    len += abs(x[i+1] - x[i]);
}

double speed = 10;
double time = len / speed;

datei << "Fuer_\n" << len << "\nMeter_\n"
    << "\nwerden_\n" << time
    << "\nSekunden_\ngebraucht." << endl;

/*****
// Berechnung der Streckenlaenge von
// x[2] bis x[3] aufaddiert
len = 0.0;
for (i=2; i<3; ++i) {
    len += abs(x[i+1] - x[i]);
}

speed = 5;
time = len / speed;

datei << "Fuer_\n" << len << "\nMeter_\n"
    << "\nwerden_\n" << time
    << "\nSekunden_\ngebraucht." << endl;
}
```

```
int main() {
    double xpos[6] = {11,8,6,9,2,1};
    func(xpos);
    return 0;
}
```

```
/* Die Funktion abs liefert den Betrag
(Absolutwert) ihres Arguments.
*/
```

- a.) Vereinfachen Sie das Programm durch Implementierung von den drei Anweisungs-Funktionen `BerechneLaenge`, `BerechneZeit` und `Ausgabe`. **Tipp:** `BerechneLaenge` enthält den Teil um die `for`-Schleife. `BerechneZeit` die Division und `Ausgabe` die Ausgabe in die Datei (schon fertig, siehe Listing 4).

Die Funktionen sollen alle als Rückgabotyp `void` besitzen, d.h. die Rückgabe von Werten erfolgt durch **Referenzen**.

Vergessen Sie nicht, die von Ihnen implementierten Funktionen in der Funktion `func` auch aufzurufen.

Setzen Sie das Schlüsselwort `const` bei der Parameterübergabe sinnvoll ein.

- b.) Fassen Sie nun die Variablen `speed`, `len` und `time` sowie Anfangs- und Endeindex in das Array `x` zu Strukturen `Flugphase` und `SpeedLenTime`, wie im Listing 5 angegeben, zusammen und vereinfachen Sie die Funktionen `func`, `BerechneLaenge` und `BerechneZeit` dadurch nochmals. Setzen Sie Referenzen und das Schlüsselwort `const` sinnvoll zur Dokumentation der Funktionsparameter ein.
- c.) Implementieren Sie die Funktion `BerechneZeit` nochmals. Dieses Mal erfolgt die Parameterrückgabe allerdings über Zeiger und nicht wie in der vorherigen Teilaufgabe über Referenzen. Geben Sie auch die geänderten Aufrufe der Funktion `BerechneZeit` an, siehe Listing 6.

Die Ausgabe von dem Programm ist übrigens:

Listing 3: (./Code1/Ausgabe.txt)

```
Fuer 16 Meter werden 1.6 Sekunden gebraucht.
Fuer 3 Meter werden 0.6 Sekunden gebraucht.
```

Um Ihnen die Arbeit zu erleichtern, sind für jeden Ausgabenteil schon Teillösungen angegeben, die Sie lediglich ergänzen sollen.

- a.) Ein Ausschnitt aus Ihrem Programm, das Sie noch weiter ergänzen sollen, sieht somit also wie folgt aus:

Listing 4: (./Code1/mainloesTeil.cxx)

```
#include <fstream>
using namespace std;
ofstream datei("Ausgabe.txt");

/* Ihre Funktionen
   BerechneLaenge
   BerechneZeit
   soll hier stehen.
   Bitte Extrablatt benutzen (Teil A1)
*/

/* Die Funktion Ausgabe sieht wie folgt
   aus */
void Ausgabe(double len, double speed,
             double time) {
    datei << "Fuer_" << len << "_Meter_"
          << "_werden_" << time
          << "_Sekunden_gebraucht." << endl;
}

void func(double x[]) {
    /*
    Hier steht Ihre Implementierung
    des neuen Code von func unter Nutzung
    von
        BerechneLaenge
        BerechneZeit und
        Ausgabe.
    Bitte Extrablatt benutzen (Teil A2)
    */
```

```
}

int main() {
    double xpos[6] = {11,8,6,9,2,1};
    func(xpos);
    return 0;
}
```

- b.) Ein Ausschnitt aus Ihrem Programm, das Sie noch weiter ergänzen sollen, sieht somit also wie folgt aus:

Listing 5: (./Code1/mainloesRecTeil.cxx)

```
#include <fstream>
using namespace std;
ofstream datei("Ausgabe.txt");

struct SpeedLenTime{
    double speed;
    double len;
    double time;
};

struct Flugphase{
    int anf;
    int end;
    SpeedLenTime slt;
};

/* Ihre Funktionen
   BerechneLaenge
   BerechneZeit
   soll hier stehen.
   Bitte Extrablatt benutzen (Teil B1).
*/

void Ausgabe(const Flugphase& f) {
    /* brauchen Sie nicht zu implementieren */
}

void func(double x[]) {
    /*
    Hier steht Ihre Implementierung
    des neuen Code von func unter Nutzung
    von
        BerechneLaenge
        BerechneZeit und
        Ausgabe.
    und der beiden Strukturen.
    Bitte Extrablatt benutzen (Teil B2).
    */
}

int main() {
    double xpos[6] = {11,8,6,9,2,1};
    func(xpos);
    return 0;
}
```

- c.) Ein Ausschnitt aus Ihrem Programm, das Sie noch weiter ergänzen sollen, sieht somit also wie folgt aus:

Listing 6: (./Code1/mainloesPtrTeil.cxx)

```

#include <fstream>
using namespace std;
ofstream datei("Ausgabe.txt");

struct SpeedLenTime{
    double speed;
    double len;
    double time;
};

struct Flugphase{
    int anf;
    int end;
    SpeedLenTime slt;
};

/* Ihre Funktionen
   BerechneZeit mit einem
   Pointer-Parameter soll hier stehen.
   Bitte Extrablatt benutzen (Teil C1).
*/

void func(double x[]) {
    /* ... wie in Aufgabe b */
    /* Ihr 1. Aufruf von BerechneZeit C2*/
    /* ... wie in Aufgabe b */
    /***/
    /* ... wie in Aufgabe b */
    /* Ihr 2. Aufruf von BerechneZeit C3*/
    /* ... wie in Aufgabe b */
}

int main() {
    double xpos[6] = {11,8,6,9,2,1};
    func(xpos);
    return 0;
}

```

```

const int FELD_GR = 1000000;
int feld[FELD_GR];

/* Die Funktion liefert das zweit groesste
   Element aus dem Array feld zurueck und
   entfernt es anschliessend
*/
int GibZweitGroesstesAusFeld();

/* Die Funktion fuegt das Element elem
   an geeigneter Stelle im Array feld
   ein, sodass GibZweitGroesstesAusFeld
   so schnell wie erforderlich, das
   zweitgroesste Element liefern kann.
*/
void FuegeEinInFeld(int elem);

```

Zur Lösung der Aufgabe ist lediglich eine Algorithmenkizze (z.B. Pseudo-Code) unter Verwendung von aus der Vorlesung bekannten Algorithmen erforderlich, z.B. die folgenden Funktionen:

```

// Prioritaetswarteschlange mit Heap
void Insert(int x);
void Remove(int& x);

```

Aufgabe 4 : Datenstruktur des Heap

ca. 6 Punkte

Aufwändige Aufgabe mit wenig Punkten

Gegeben sei das in Listing 7 angegebene Modul. Beweisen bzw. widerlegen Sie die folgende Aussage:

Es gibt keinen Algorithmus, sodass sowohl die Funktion `GibZweitGroesstesAusFeld` als auch die Funktion `FuegeEinInFeld` ihre Aufgabe mit maximal logarithmischer Zeitkomplexität erledigen, weil hierzu eine Sortierung des Arrays erforderlich ist und die schnellsten bekannten Sortierverfahren eine Zeitkomplexität von $O(N * \log N)$ aufweisen.

Listing 7: (./Code1/Heap.cxx)

```

#include <iostream>
using namespace std;

```