Klausur
Kompaktkurs Einführung in die Programmierung
Dr. T. Weinzierl & M. Sedlacek
18. April 2012

Wichtig:

- Die Angabe besteht aus 10 Seiten — prüfen Sie bitte Ihr Exemplar auf Vollständigkeit.
- Kennzeichen Sie jedes Blatt lesbar mit Ihrem Namen und dieses Deckblatt zusätzlich mit Ihrer **Matrikelnummer und Unterschrift**; trennen Sie die Heftung nicht auf.
- Bearbeiten Sie die Aufgaben auf den Angabenblättern. Sollte der Platz für einen Aufgabenteil nicht reichen, benutzen Sie bitte die Rï¿½ckseite der Angabenblï¿½tter und machen Sie einen entsprechenden Vermerk bei der Aufgabe.
- Insgesamt können Sie 40+3 Punkte erreichen (3 Punkte Überhang), bei jeder Aufgabe ist die maximale Anzahl an erreichbaren Punkten angegeben. Zum Bestehen sind 17 Punkte erforderlich.
- Es ist alles außer elektronische Gerate als Hilfsmittel zugelassen.
- Eigene Funktionen dürfen in den folgenden Teilaufgaben weiterverwendet werden; von den Standardfunktionen sind, soweit nicht bei der Aufgabe anders angegeben, nur die zur Ausgabe zulï¿½ssig (`std::cout` bzw. `printf`, was immer Ihnen davon lieber ist).
- Die Bearbeitungszeit betragt 75 Minuten.

Aufg.	1)	2)	3)	4)
Pkt. (ca.)	/14	/8	/10	/8(+3)

Summe	/40(+3)
Note	

1 Ausdrücke und Funktionen (ca. 14 Punkte)

a) Was ist der Inhalt von `i` nach Ausführung dieser Anweisungen?

```
int i = 6;
i = i + i;
```

b) Was ist der Inhalt von `c1` und `c2` nach Ausführung dieser Anweisungen?

```
char c1 = 'c';
char c2 = c1 - 2;
c2++;c2++;
```

c) Was ist der Inhalt von `a` und `b` nach Ausführung dieser Anweisungen?

```
int a[4];
a[0] = -1;
a[1] = 0;
a[2] = -1;
a[3] = 1;
int b = 0;
for(int i = 0; i < 4; i++) {
    b = b + a[i];
    a[i] = b;
}
```

d) Es sei folgendes Programm gegeben:

```
int y[5];
for(int i = 0; i < 5; i++)
    y[i] = i*i;
```

Geben Sie nach Ablauf dieses Programms das Ergebnis zu folgenden Ausdrücken an. `y[0]` ist dabei an Adresse `0x35` gespeichert.

`y` =

`&y[0]` =

`*y` =

`*(y+3)` =

e) Welche einfache mathematische Funktion wird mit `foo(int *a, int b)` implementiert bzw. was gibt die Funktion zurück wenn man sie mit einem Feld `a` und dessen Länge `b` aufruft?

```
float foo(int *a, int b) {
    if (b <= 0)
        return a[0];
    return (a[b] + b*foo(a,b-1))/(b+1);
}
```

f) In folgender Funktion sind 5 Fehler implementiert. Finden Sie diese und geben Sie sie mit kurzer Begrï¿½ndung an. Sie können davon ausgehen, dass die Variable `laenge`, die die Anzahl der Elemente in `feld` angibt, korrekt übergeben wurde.

```
double bar(int feld, int laenge) {
    double sum == 0.0;
    double norm
    for (int i = 0, i < laenge, i++)
    sum += feld[i]*feld[i];
    norm = sqrt(sum);
}
```

2 Pascal'sches Dreieck (ca. 8 Punkte)

Im Folgenden soll eine Reihe von Funktionen realisiert werden, die ein Pascal'sches Dreieck wie im Bild rechts druckt. Dabei soll die Höhe des Dreiecks von einer Variablen `hoehe` abhängen. Im Bild rechts hat das Dreieck die `hoehe = 5`, da 5 Zeilen gedruckt werden.

				1				
			1		1			
		1		2		1		
	1		3		3		1	
1		4		6		4		1

a) Schreiben Sie zunächst eine Funktion `fac`, die zu einer übergebenen Zahl `a` deren Fakultät $a!$ berechnet und zurückgibt. Benutzen Sie hierzu eine **Iteration** und **keine Rekursion**! Beachten Sie den Fall $a = 1$ bzw. $a = 0$. Sie müssen hier die Signatur der Funktion selbständig angeben!

b) Im Pascal'schen Dreieck kann die k-te Zahl in der n-ten Reihe mittels

$$\binom{n}{k} = \frac{n!}{k!(n-k)!} \tag{1}$$

berechnet werden. Schreiben Sie unter zu Hilfenahme von Teilaufgabe a) eine Funktion `int pascalZahl(int n, int k)`, die die k-te Zahl in der n-ten Reihe gemäß (1) zurückgibt.

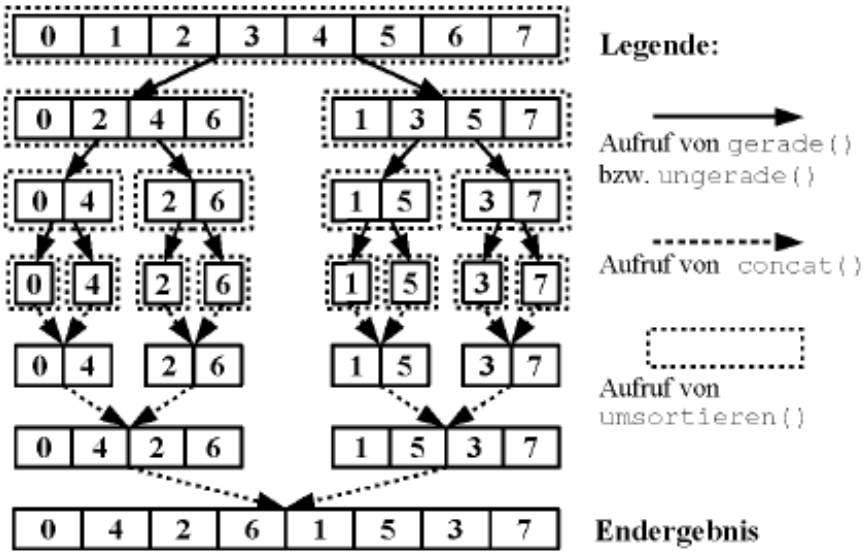
c) Schreiben Sie eine Funktion `void pascalDreieck(int hoehe)`, die das rechtsstehende Pascal'sche Dreieck **linksbï¿½ndig** druckt, abhängig vom übergebenen Parameter `hoehe`. Wahlweise können Sie auch zwei Funktionen implementieren, wovon die eine in `void pascalDreieck(int hoehe)` aufgerufen wird. Benutzen Sie hierzu die in Teilaufgabe b) erstellte Funktion. Beachten Sie die Leerpositionen (whitespaces) zwischen den Zahlen!
Hinweis: Es sollen Dreiecke beliebiger `hoehe` gedruckt werden können und nicht nur für `hoehe = 5`.

1								
1		1						
1		2		1				
1		3		3		1		
1		4		6		4		1

d) Schreiben Sie nun eine Funktion `void einruecken (int z)`, die z mal eine Leerposition (whitespace) druckt. Vervollständigen Sie anschließend Ihre Funktion `void pascalDreieck(int hoehe)` aus Teilaufgabe c) so, dass das Pascal'sche Dreieck in richtiger Form (zentriert) gedruckt wird. Hierzu müssen Sie Ihre Funktion aus Teilaufgabe c) nochmals abschreiben und mit der Funktion `void einruecken (int z)` korrekt vervollständigen.

3 Umsortieren (ca. 10 Punkte)

Folgende Abbildung zeigt einen Umsortierungsalgorithmus (auf Feldern), der beispielsweise bei der Fourier-Transformation zur Anwendung kommt. Hierbei werden die Daten in gerade und ungerade Teilmengen (Teilfelder) sortiert.



a) Implementieren Sie eine Funktion `double* ungerade(double* feld, int n)`, die ein neues Feld der Länge $\frac{n}{2}$ erzeugt und das alle Elemente mit ungeraden Indizes des übergebenen Feldes `feld` enthalten soll. Sie dürfen davon ausgehen, dass n stets eine gerade Zahl ist. Das übergebene Feld `feld` darf von der Funktion nicht verändert werden!

b) Schreiben Sie eine Funktion `double* concat(double* feld1, double* feld2, int n)`, die die Elemente der beiden übergebenen Felder (jeweils der Länge n) kombiniert zu einem neu anzulegenden Feld der Länge $2n$. Dabei sollen die Elemente von `feld1` in der ersten und jene von `feld2` in der zweiten Hälfte des Ergebniss-Feldes stehen.

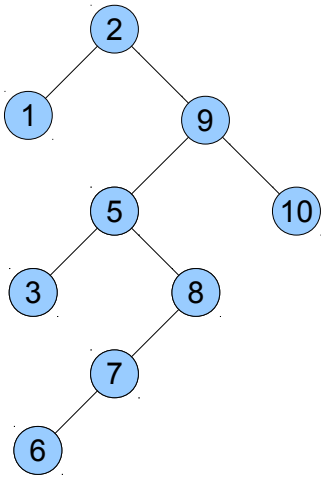
c) Gehen Sie davon aus, dass Sie analog zur Funktion `double* ungerade(double* feld, int n)` eine Funktion `double* gerade(double* feld, int n)` zur Verfï¿½gung haben, die ein Feld der Lï¿½nge $\frac{n}{2}$ erzeugt das alle Elemente mit geraden Indizes des ï¿½bergebenen Feldes `feld` enthï¿½lt. Schreiben Sie nun eine **rekursive** Funktion `void umsortieren(double*& feld, int n)` die ein Feld entsprechend dem Algorithmus im obigen Bild umsortiert. Sie kï¿½nnen davon ausgehen, dass n gerade und eine Zweierpotenz ist (z.B. $n = 8$). Dabei soll auf dem ï¿½bergebenen Feld operiert werden, d.h., `feld` wird hier verï¿½ndert ï¿½ber `double*& feld`. Beachten Sie auch die korrekte Freigabe des Freispeichers.

4 Funktionen auf Binärbäumen (ca. 8 (+3) Punkte)

Aus der Vorlesung und Übung sind Ihnen Binärbaume bekannt. Ein Binärbaum ist entweder leer (NULL) oder kann rekursiv definiert werden über zusammenhängende Strukturen der Form

```
struct knoten {
    int x;
    struct knoten *l, *r;
};
```

wobei `x` einen ganzzahligen Wert darstellt und `l`, `r` einen mï¿½glichen linken bzw. rechten Teilbinï¿½rbaum darstellen. Im Bild rechts sei ein beispielhafter Binï¿½rbaum dargestellt, der aus der Zahlenfolge




```
2,1,9,5,10,3,8,7,6
```

aufgebaut ist. Nehmen Sie fï¿½r alle folgenden Funktionen an, dass Sie einen Zeiger `struct knoten *binbaum` zur Verfï¿½gung haben, der auf einen vollstï¿½ndig aufgebauten Binï¿½rbaum zeigt.

a) Schreiben Sie eine Funktion `prefix_drucken(struct knoten* binbaum)`, die fï¿½r einen ï¿½bergebenen Binï¿½rbaum `binbaum` seine Werte in Prï¿½fix-Ordnung (zuerst Wurzel, dann linker Teilbaum, dann rechter Teilbaum) ausgibt. Fï¿½r den oben gezeigten Binï¿½rbaum wï¿½rde das bedeuten, dass seine Werte in folgender Reihenfolge ausgegeben werden sollten: `2 1 9 5 3 8 7 6 10`
Hinweis: Die Funktion soll fï¿½r Binï¿½rbï¿½ume **beliebiger** Gestalt funktionieren und nicht nur fï¿½r obig dargestellten.

b) Schreiben Sie eine Funktion `int min_wert(struct knoten* binbaum)`, die fï¿½r einen ï¿½bergebenen Binï¿½rbaum `binbaum` den minimalen Wert (minimales `x`) findet und zurï¿½ckgibt.

Fï¿½r obiges Beispiel wï¿½re das Minimum `1`.
Hinweis: Die Funktion soll fï¿½r Binï¿½rbï¿½ume **beliebigen** Inhalts funktionieren und nicht nur fï¿½r obig dargestellten.

c) Schreiben Sie eine Funktion, die fï¿½r einen ï¿½bergebenen Binï¿½rbaum die maximale Tiefe (Pfadlï¿½nge) ermittelt. Die Tiefe (ein Pfad) ist dabei ein Weg von der Wurzel zu einem Blatt auf unterster Ebene. Fï¿½r obiges Beispiel wï¿½re der maximale Pfad `2-9-5-8-7-6` und damit hï¿½tte der dargestellte Binï¿½rbaum eine maximale Tiefe von `5`.

d) Schreiben Sie eine Funktion `bool hat_pfad_summe(struct knoten* binbaum, int pfad_summe)`, die fï¿½r einen ï¿½bergebenen Binï¿½rbaum und eine ganze Zahl `pfad_summe` ï¿½berprï¿½ft, ob die Summe der Werte eines Pfades mit `pfad_summe` ï¿½bereinstimmt, der Binï¿½rbaum also einen Pfad der Summe `pfad_summe` hat. Beachten Sie, dass ein Binï¿½rbaum mehrere Pfade haben kann. Obiger Binï¿½rbaum enthï¿½lt z.B. eine Pfadsumme vom Wert `19` (Pfad: `2-9-5-3`) aber enthï¿½lt keine Pfadsumme vom Wert `20`.

e) Schreiben Sie die Funktion aus Teilaufgabe a) in objektorientierter Form als Methode einer Klasse.