technische universität dortmund

fakultät für informatik

Martin Apel Informatik VI, GB5/R 431; Tel.: 4775
Nicola Beume Informatik XI, OH14/R 233; Tel.: 7704
Jürgen Mäter Informatik IV, GB5/R 402; Tel.: 2411

Dortmund, den 05. November 2010

Übungen zur Vorlesung EINI (WS 2010)

Blatt 4

Block gelb

Es können (4 + 3) Punkte erreicht werden.

Abgabedatum: 14. November 2010 23:59 Uhr

Hinweise

1) Bitte beachten Sie die aktuellen Hinweise unter

   http://www4.cs.uni-dortmund.de/home/maeter/UeEiniET10W/

   Für die Abgabe sind die Dateien: *Aufgabe_04_0.txt*, *Aufgabe_04_1.cpp*, *Aufgabe_04_2a.cpp* und *Aufgabe_04_2b.cpp* zu erstellen.
2) In der optionalen Datei *Anmerkungen.txt* können Sie allgemeine Anmerkungen bezüglich Ihrer Lösungen notieren.
3) Es ist ratsam, die Programme vor der Abgabe zu kompilieren und auszuführen.
4) Aufgabe 2 ist optional, d.h., daß Sie sich 3 Zusatzpunkte für den gelben Block erarbeiten können.
5) Verwenden Sie für die Textaufgaben reine Texteditoren. Das Abgabesystem erkennt **keine** Word- oder PDF-Dateien!!!

Spruch des Tages

(frei nach Murphy)

Die Ursache eines jeden Fehlers, der dem Rechner angelastet wird, besteht aus zwei menschlichen Fehlern. Mitgerechnet der, dem Computer den Fehler zuzuschieben.

Aufgaben

Aufgabe 0: Grundlagen (1 Punkt)

Legen Sie für Ihre Antworten eine Text-Datei *Aufgabe_04_0.txt* an.

a) Welcher Nachteil kann bei statischen Arrays auftreten (0,2 Punkte)?

b) Was versteht man unter dynamischem Speicher (0,2 Punkte)?

c) Wo wird der dynamische Speicher angelegt (0,2 Punkte)?

d) Welche Operatoren dienen der Erzeugung und dem Löschen dynamischer Datenstrukturen (0,2 Punkte)?

e) Warum müssen dynamische Objekte auch wieder gelöscht werden (0,2 Punkte)?

Speichern Sie Ihre Ergebnisse in der Ergebnisdatei *Aufgabe_04_0.txt*.

Aufgabe 1: Dynamischer Speicher (3 Punkte)

Legen Sie eine cpp-Datei *Aufgabe_04_1.cpp* an und erstellen Sie ein C++-Programm, welches eine $n \times n$ Matrix $A = (a_{ij})$ mit $n \in \mathbb{N}$ und $a_{ij} \in \mathbb{Z}$ einliest und speichert. Der Wert für n soll zur Laufzeit festgelegt werden können, d.h. vor Einlesen der Matrix ebenfalls abgefragt werden. Die Matrix muss also unter Verwendung von dynamischen Speicher zur Laufzeit erzeugt werden (1 Punkt). Anschließend soll die *Transponierte* $A^t = (a_{ji})$ der Matrix A berechnet und gespeichert werden (1 Punkt). Wird also z.B. die 4×4-Matrix

$$A = \begin{pmatrix} 10 & 21 & 42 & 55 \\ 33 & 71 & 43 & 66 \\ 61 & 82 & 17 & 77 \\ 3 & 2 & 1 & 0 \end{pmatrix}$$

eingelesen, so soll

$$A^t = \begin{pmatrix} 10 & 33 & 61 & 3 \\ 21 & 71 & 82 & 2 \\ 42 & 43 & 17 & 1 \\ 55 & 66 & 77 & 0 \end{pmatrix}$$

berechnet und gespeichert werden. Nach Berechnung der Transponierten A^t sollen die Elemente von A^t auf der Konsole ausgegeben werden (z.B. in der Form `Elemente der Transponierten: A_t(0,0)=10, A_t(0,1)=33, ...`).
Kompilieren Sie das Programm, führen Sie es anschließend aus und überprüfen Sie die Ergebnisse und kopieren Sie sie als Block-Kommentar an das Ende der cpp-Datei *Aufgabe_04_1.cpp* (1 Punkt).

(Optionale) Aufgabe 2: Funktionen (3 Punkte)

a) Für eine natürliche Zahl n wird $n!$ (`Fakulatät`) definiert durch

$$n! := \begin{cases} 1 \cdot 2 \cdot \ldots \cdot n & \text{für } n \geq 1, \\ 1 & \text{für } n = 0. \end{cases}$$

Legen Sie eine cpp-Datei *Aufgabe_04_2a.cpp* an. Definieren Sie in C++ eine Funktion `void fakultaet(int n)`, welche einen Parameter `n` vom Typ `int` besitzt und welche im Fall `n` ≥ 0 den Wert für `n`! berechnet und auf der Konsole ausgibt. Testen Sie die so definierte Funktion, indem Sie die `main`-Methode ihres Programms so anpassen, dass bei Ausführung ein Wert für eine Variable `m` vom Typ `int` von der Konsole eingelesen wird und im Fall `m` ≥ 0 anschließend die Werte für $0!, 1!, 2!, \ldots,$ `m`! mittels der Funktion `fakultaet` berechnet und ausgegeben werden. Für `m` < 0 soll die Fehlermeldung `Fakultaet nur fuer positive Zahlen definiert!` ausgegeben werden.
Kompilieren Sie das Programm, führen Sie es anschließend aus, überprüfen Sie die Ergebnisse und kopieren Sie die Ergebnisse als Block-Kommentar an das Ende der cpp-Datei *Aufgabe_04_2a.cpp* (1 Punkt).

b) Wir definieren eine *Fibonacci-Folge* durch $f_0 := 1$, $f_1 := 1$ und $f_{n+1} := f_n + f_{n-1}$ für $n \geq 1$. Die ersten beiden Fibonacci-Zahlen sind also gleich 1, alle weiteren Fibonacci-Zahlen sind jeweils gleich der Summe der beiden vorhergehenden Fibonacci-Zahlen.

Legen Sie eine cpp-Datei *Aufgabe_04_2b.cpp* an. Definieren Sie in C++ eine Funktion `void fibonacci(int n)`, welche einen Parameter `n` besitzt und für `n` ≥ 0 die `n`-te Fibonacci-Zahl $f_{\texttt{n}}$ berechnet und auf der Konsole ausgibt. Testen Sie Ihre Funktion, indem Sie in der `main`-Methode Ihres Programms zunächst einen Wert für eine Variable `N` vom Typ `int` einlesen und im Fall `N` ≥ 0 anschließend alle Fibonacci-Zahlen von 0 bis `N` mittels der Funktion `fibonacci` auf der Konsole ausgeben. Für `N` < 0 soll die Fehlermeldung `Fibonacci-Folge nur für positive Zahlen definiert!` ausgegeben werden.
Kompilieren Sie das Programm, führen Sie es anschließend aus, überprüfen Sie die Ergebnisse und kopieren Sie die Ergebnisse als Block-Kommentar an das Ende der cpp-Datei *Aufgabe_04_2b.cpp* (2 Punkte).