technische universität dortmund

fakultät für informatik

Martin Apel Informatik VI, GB5/R 431; Tel.: 4775
Andreas Thom Informatik XI, OH14/R 213; Tel.: 7712
Christian Scheffer Informatik XI, OH14/R 213; Tel.: 7712
Wolfgang Paul Informatik XI, OH14/R 215; Tel.: 7714

Dortmund, den 11. November 2011

Übungen zur Vorlesung EINI (WS 2011)

Blatt 5

Block rot

Es können 4 Punkte erreicht werden.

Abgabedatum: 20. November 2011 23:59 Uhr

Hinweise

1) Bitte beachten Sie die aktuellen Hinweise unter

   http://ls11-www.cs.tu-dortmund.de/teaching/ep1112uebung/

   Für die Abgabe sind die Dateien: *Aufgabe_05_0.txt* und *Aufgabe_05_1.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) Verwenden Sie für die Textaufgaben reine Texteditoren. Das Abgabesystem erkennt **keine** Word- oder PDF-Dateien!!!

Spruch des Tages

(frei nach Murphy)

Würden Architekten Häuser bauen, wie Programmierer ihre Programme, dann könnte ein Specht die ganze Zivilisation zerstören.

Aufgaben

Aufgabe 0: Grundlagen (1 Punkt)

Legen Sie für Ihre Antworten eine Text-Datei *Aufgabe_05_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_05_0.txt*.

Aufgabe 1: Dynamischer Speicher (3 Punkte)

Legen Sie eine cpp-Datei *Aufgabe_05_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 dynamischem 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

$$A = \begin{pmatrix} 15 & 21 & 17 & 33 \\ 71 & 99 & 10 & 8 \\ 55 & 46 & 30 & 20 \\ 66 & 35 & 81 & 9 \end{pmatrix}$$

eingelesen, so soll

$$A^t = \begin{pmatrix} 15 & 71 & 55 & 66 \\ 21 & 99 & 46 & 35 \\ 17 & 10 & 30 & 81 \\ 33 & 8 & 20 & 9 \end{pmatrix}$$

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_05_1.cpp* (1 Punkt).

(Präsenz-)Aufgabe 2: Parameterübergabe (0 Punkte)

In der Vorlesung wurden die beiden Konzepte *call by value* und *call by reference* vorgestellt. Welcher Art sind die Aufrufe der folgenden vier Funktionen? Welche Ausgaben erwarten Sie und warum? Vermerken Sie neben den Funktionen den erwarteten Effekt auf die Variablen `x` und `y` und auf die Zeiger `z1` und `z2`.

```
1  #include <iostream>
2  using namespace std;
3
4  void function_1(int x, int y) {
5      y = x;
6  }
7
8  void function_2(int *x, int *y) {
9      y = x;
10 }
11
12 void function_3(int *x, int *y) {
13     *y = *x;
14 }
15
16 void function_4(int x, int& y) {
17     y = x;
18 }
19
20 int main() {
21     int y = 33;
22     int x = 19;
23     int *z1, *z2;
24
25     z1 = &x;
26     z2 = &y;
27
28     function_1(x, y);
29     cout << "x: " << x << " y: " << y << endl;
30 //x:    y:
31
32     y = 33;
33     function_2(z1, z2);
34     cout << "x: " << x << " y: " << y << endl;
35 //x:    y:
36
37     y = 33;
38     function_3(z1, z2);
39     cout << "x: " << x << " y: " << y << endl;
40 //x:    y:
41
42     y = 33;
43     function_4(x, y);
44     cout << "x: " << x << " y: " << y << endl;
45 //x:    y:
46
47     return 0;
48 }
```