ulm university universität
uulm

Inst. für Angew. Informationsverarbeitung
Prof. Dr. Franz Schweiggert
Michaela Weiss
Wolfgang Kaifler

07.12.2010
Lösung 6

Systemnahe Software I (WS 2010/2011)

Ausgabetermin: 07.12.2010

Aufgabe 1: Theorie (5 Punkte)

- **Erklären Sie mit eigenen Worten, was ein Zeiger ist.**
  Ein Zeiger ist eine Variable, die die Speicheradresse einer anderen Variablen beinhaltet und so auf diese „zeigt“.
- **Was liefert *&array[0]?**
  Dies liefert den Inhalt der Speicherstelle des 1. Elements des Vektors und ist daher gleichwertig mit array[0].
- **Wie kann der Ausdruck *(array+i) bei dem int array[10] vereinfacht werden?**
  Array ist ein Zeiger auf das 1. Element des Vektors. Dieses wird mit Hilfe von i verschoben. Daher ist dieser Aufruf identisch mit array[i].

Aufgabe 2: Ausgabeverwirrung (5 + 2 + 2 Punkte)

```
1   #include <stdio.h>
2
3   void tu_was (int* p) {
4           *p = *p - 1;
5   }
6
7   int tu_nochmal_was (int a, int b) {
8           tu_was(&a);
9           return a * b;
10  }
11
12  int main () {
13          int array[] = {3,5,7,9,11};
14          int alt;
15
16          for (int i = ((sizeof array) / sizeof (int)); i >0; ) {
17                  tu_was(&i);
```

```
18                      alt = array[i];
19                      array[i] = tu_nochmal_was(array[i], i);
20                      printf("%d: alt = %d neu = %d\n", i, alt, array[i]);
21              }
22
23              return 0;
24      }
```

Quellcode 1: verwirrung.c

- **Überlegen Sie sich, was das oben angegebene Programm ausgibt. Analysieren Sie hierzu zunächst, was die beiden Funktionen tu_was und tu_noch_was machen.**
  tu_was: Dekrementiert den Wert des übergebenen Parameters.
  tu_noch_was: Dekrementiert den Parameter a mit Hilfe der Funktion tu_was um 1 und multipliziert das Ergebnis mit b. $-->$ (a-1) * b
  Ausgabe:

```
4: alt = 11 neu = 40
3: alt = 9 neu = 24
2: alt = 7 neu = 12
1: alt = 5 neu = 4
0: alt = 3 neu = 0
```

- **Erklären Sie, zu welchem Seiteneffekt der Aufruf der Funktion tu_was führt und warum.**
  Die Funktion tu_was erhält als Parameter einen int-Pointer und somit eine Referenz auf die Speicherstelle der übergebenen Variablen i (vgl. Call-by-Reference). Eine Änderung des Wertes innerhalb der Funktion führt deshalb zu einer Veränderung der Original-Variablen.

- **Erklären Sie, warum der Aufruf der Funktion tu_was innerhalb der Funktion tu_noch_was zu keinen Änderungen an den von der main übergebenen Parametern führt.**
  Der Funktion tu_noch_was werden per Call-by-Value zwei int-Parameter übergeben. A und b stellen daher lokale Variablen der Funktion tu_noch_was dar. Diese werden zwar innerhalb dieser Funktion durch Aufruf der tu_was-Funktion verändert, „sterben“ aber nach Abarbeitung der Funktion und führen zu keinen Änderungen in der main-Funktion.