Betriebssysteme und Softwaretechnik

Abgabe: 11.05.2017

Adrian C. Hinrichs Matr.Nr. 367129
Jeremias Merten Matr.Nr. 367626
Georg C. Dorndorf Matr.Nr. 366511

Aufgabe 2

Aufgabe 2.1

Die Operation ist möglich. Sie bewirkt, dass im `char` Array am Index 1 die 2 mit einer 0 überschrieben wird.

a)

Die Operation ist möglich. Sie bewirkt, dass zunächst 3 auf den Pointer, der auf den Anfang des Arrays `list` zeigt, addiert wird. Dann wir durch `*` die Adresse dereferenziert. `i` enhält danach also den Wert des Arrays an der Position 4 also `list[3]`.

b)

Die Operation ist möglich jedoch potentiell gefährlich. `list[i]` gibt den Inhalt des Speichers zurück, der 7 Adressen weiter von Beginn des Arrays an steht. `list` ist jedoch nur 4 Speicherzellen groß – daher kann es beim zugriff auf diesen Speicher zu Fehlern kommen. Durch das Ampersand wird der Pointer zum Inhalt der Zelle zurückgegeben. Letztendlich wird dieser Pointer dann in `pi` gespeichert.

c)

Die Operation ist möglich. Der Pointer `pi` zeigt danach auf die Speicheradresse 42.

d)

Die Operation ist nicht möglich, da `list` ein `int` Array ist und `pi` vom Typ Pointer.

e)

Diese Operation ist möglich greift jedoch auf vorher nicht allokierte Speicherbereiche zu. Zunächst werden auf den Pointer `pi` 2 addiert. Dann wird der Pointer dereferenziert. `i` hat danach den Wert der Speicheradresse `pi + 2`.

f)

Die Operation funktioniert. An die Speicheradresse `str[3]` wird ein Char `'4'` geschrieben.

Adrian C. Hinrichs Matr.Nr. 367129
Jeremias Merten Matr.Nr. 367626
Georg C. Dorndorf Matr.Nr. 366511

Aufgabe 2.2

a)

Der Test ergab folgendes Ergebnis:

Eingabe (Celsius)	Ausgabe (Fahrenheit)	Tatsächlich (Fahrenheit)
-47	-15.000000	-52,6
0	32	32
20.3	52.299999	68.54
42	47	107,6

Abbildung 1: Eingaben und Ausgaben vs. richtigem Ergebnis

Offensichtlich sind alle Werte bis auf $0°C$ falsch umgerechnet worden. Dies liegt an folgendem Code-Teil:

```
float celsius_zu_fahrenheit(float temperatur) {
  return 9/5 * temperatur + 32;
}
```

Dadurch, dass sowohl `9` als auch `5` `Integer`-Werte sind, wird die Operation `9/5` als Ganzzahldivision ausgewertet, und liefert das Ergebnis 1 zurück, wodurch die gesammte Berechnung fehlerhaft wird.

b)

Es wurden nur eine Änderung in der umrechnungsmethode vorgenommen:

```
float celsius_zu_fahrenheit(float temperatur) {
  return 9/5.0 * temperatur + 32;
}
```

c)

Lösung: Ja! Bei der vorgenommenen Änderung handelt es sich nicht um eine explizite Typkonvertierung, da `5.0` nun von beginn an eine Gleitkommazahl ist.

Aufgabe 2.3

```
1  int a[6] = {2,11,23,42,13,37};
2  int * b = a;                          // b zeigt nun auf a[0]
3  b++;
4  int t1 = * b;
5  int ** c = & b;
6  ** c += 2;
7  int t2 = ** c + 1;
8  int t3 = a[3] & (***c);
9  b = a+5;
10 * b = 10;
```

3 b wird um 1 inkrementiert, zeigt nun also auf `a[1]`

4 `t1` wird nun der Wert von `a[1]` (also 11) zugwiesen

5 `**c` wird auf `& b` gesetzt, das bedeutet `c` zeigt nun auf `a[1]`

6 `**c` wird um zwei inkrementiert. Da `c` auf `a[1]` wird `a[1]` auf 13 gesetzt. Nun gilt: `a={2, 13, 23, 42, 13, 37}`

7 `t2` wird auf `**c +1`, also 14 gesetzt.

8 In `t3` wird die bitweise Verundung von `a[3]` = 42 und `**c`= 13 gespeichert. Da $42_{10} = 101010_2$ und $13_{10} = 001101_2$ ist die bitweise Verundung der beiden Zahlen = $00100_2 = 8_{10}$

9 `b` wird auf den Wert von `a + 5` gestzt, dies ist eine Operation, gänzlich mit Adressen gewesen, daher zeigt B nun auf `a[0]+5 = a[5]`

10 `*b` (also `a[5]` wird auf 10 gesetzt.
Nun gilt `a={2, 13, 23, 42, 13, 10}`

Am ende des Programmes haben folgende Variablen diese Werte:

```
a={2, 13, 23, 42, 13, 10}
t1=11
t2=14
t3=8
```

Aufgabe 2.4

a)

```
1   [main] : a=5, b=6
2   [summe]: a=1, b=2
3   a+b=11
4   [diff] : a=4, b=3
5   a-b=-1
```

Listing: Ausgabe des Programms funktionen_a.

1 `[main] : a=5, b=6` wird ausgegeben, da in Zeile 27 im Programm `printf()` mit den Paramtern `a,b` aufgerufen wird und an dieser Stelle die Werte der lokal in der Main-Methode in Zeile 23, 24 definierten Variablen `a,b` ausgegeben wird.

2 `[summe]: a=1, b=2` wird ausgegeben, da in Zeile 28 die Methode `summe()` aufgerufen wird und in dieser `printf() a,b` als Parameter erhält, jedoch keine lokalen definiert sind, sodass auf die Globalen Instanzen zurückgegriffen wird.

3 `a+b=11` wird ausgegeben, da der Aufruf von `summe()` in Zeile 28 die Summe der beiden übergebenen Integer zurück gibt und `summe()` mit den lokal in der Main-Methode definierten Werten `a,b` aufgerufen wurde.

4 `[diff] : a=4, b=3` wird ausgegeben, da in der Main-Methode in Zeile 30 die Methode `diff()` aufgerufen wird und in dieser die beiden lokal definierten Variablen `a,b` ausgegeben werden.

5 `a-b=-1` wird ausgegeben, da `diff()` die letztere von den beiden übergebenen Variablen `p1,p2` von der ersteren abzieht und das Ergebnis an die Adressenauf die der Pointer `int * d` zeigt schreibt. Von dort wird es dann in der Main-Methode in Zeile 31 gelesen und ausgegeben.

b)

```
#include <stdio.h>

/* Definition von globalen Variablen.
 * Diese Variablen koennen ueberall im Programm veraendert werden.
 */
int anz = 10;
int max = 0;
int array[10] = {4, 6, 2, 0, 9, 1, 5, 7, 8, 3};

/* Diese Funktion vergleicht und sortiert die beiden Feldeintraege
   "i1" und "i2" im Array "array".
 */
void exchange(int i1, int i2) {

  if(array[i1] < array[i2])
    {
      if (array[i1] > max)
        max = array[i1];
    }
  else
    {
      if(array[i2] > max)
        max = array[i2];
      int tmp;
      tmp = array[i1];
      array[i1] = array[i2];
      array[i2] = tmp;
    }
}

int main()
{
    int i, j;

    for(i=0; i<anz; i++)
      {
        for(j=i; j<anz; j++)
          exchange(i, j);
      }

    printf("Die Zahlen in sortierter Reihenfolge:");
    for(i=0; i<anz; i++)
      printf(" %d", array[i]);

    printf("\nDas Maximum: %d\n", max);

    return 0;
}
```

c)

```
#include <stdio.h>

/* Definition von globalen Variablen.
 * Diese Variablen koennen ueberall im Programm veraendert werden.
 */


/* Diese Funktion vergleicht und sortiert die beiden Feldeintraege
   "i1" und "i2" im Array "array".
 */
int exchange(int i1, int i2, int array[], int max) {

  if(array[i1] < array[i2])
    {
      if (array[i1] > max)
        max = array[i1];
    }
  else
    {
```

```
      if(array[i2] > max)
        max = array[i2];
      int tmp;
      tmp = array[i1];
      array[i1] = array[i2];
      array[i2] = tmp;
    }
  return max;
}

int main()
{
  int anz = 10;
  int max = 0;
  int array[10] = {4, 6, 2, 0, 9, 1, 5, 7, 8, 3};

  int i, j;

  for(i=0; i<anz; i++)
    {
      for(j=i; j<anz; j++)
        max = exchange(i, j, array, max);
    }

  printf("Die Zahlen in sortierter Reihenfolge:");
  for(i=0; i<anz; i++)
    printf(" %d", array[i]);

  printf("\nDas Maximum: %d\n", max);

  return 0;
}
```

Aufgabe 2.5

```
#include <stdio.h>
#include <stdlib.h>

typedef struct {
    int x;
    int y;
} Point;

Point mkpoint(int x,int y) {
  Point p;
  p.x = x;
  p.y = y;
  return p;
}

int comparator(const void *a, const void *b)
{
  return ( *(int*)a - *(int*)b );
}

int main()
{
  int i;
  Point points[] = { mkpoint(1,3), mkpoint(12,1), mkpoint(7,16), mkpoint
      ↪ (6,10), mkpoint(13,1) };
  printf("Vor qsort: \n");
  for (i = 0; i < 5; i++) {
    printf("%d: %d %d\n", i, points[i].x, points[i].y );
  }

  // MARKIERUNG: Sortieren Sie hier die Liste an Punkten aufsteigend
      ↪ bezueglich der X Koordinaten

  qsort(points, 5, sizeof(points[0]), comparator);

  printf("Nach qsort: \n");
  for (i = 0; i < 5; i++) {
```

```
    printf("%d: %d %d\n", i, points[i].x, points[i].y );
  }

  return(0);
}
```