Lösungen zur Vorlesung
GRUNDLAGEN DER INFORMATIK I
Studiengang Elektrotechnik
WS 02/03

AG Betriebssysteme FB3
Kirsten Berkenkötter

Lösung Praktikum 4

1 Aufgabe 1: Sortieren mit Bubble-Sort

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

//die Größe des Arrays definieren
#define MAX 25

//eine Funktion zum Tauschen von zwei Variablen
void swap(int *a, int *b)
{
   int hilf;

   hilf = *a;
   *a = *b;
   *b = hilf;
}

//der eigentliche Bubble-Sort Algorithmus
void bubble_sort(int *array, int laenge)
{
   //zwei Index-Variablen
   int i, j;

   //die äußere Schritte gibt die Sortierschritte an
   //der letzte Schritt ist nicht nötig, da das letzte
   //Element automatisch am Ende steht, wenn die anderen
   //an ihren jeweiligen Platz geschrieben wurden!
   for (i = 0; i < laenge-2; i++)
   {
      //die innere Schleife startet beim letzten Element
      //sie vergleicht immer das j-te und das j-1-te Element
      //und tauscht, wenn sie nicht in der richtigen Reihen-
      //folge stehen
      //die Schleife läuft, solange j > i ist, da das i-1-te
      //Element bereits sortiert ist, und das i-te Element durch
      //die Schleife bereits abgedeckt wird (j-1-te Element!)
      for (j = laenge-1; j > i; j–)
         if (array[j] < array[j-1])
            swap(array+j, array+j-1);
```

```
    }
}

int main()
{
    //ein Array und einen Zähler deklarieren
    int array[MAX];
    int i;

    //Zufallszahlengenerator initialisieren
    srand(time(NULL));

    //Zufallszahlen erzeugen und das Array damit belegen
    for (i = 0; i < MAX; i++)
        array[i] = rand()%1000;

    //Array ausgeben, immer 10 Elemente in einer Zeile
    for (i = 0; i < MAX; i++)
    {
        printf("%3d ", array[i]);
        if (((i+1) % 10) == 0)
            printf("\n");
    }
    printf("\n");

    //Array sortieren
    bubble_sort(array, MAX);

    //sortieres Array ausgeben, immer 10 Elemente in einer Zeile
    for (i = 0; i < MAX; i++)
    {
        printf("%3d ", array[i]);
        if (((i+1) % 10) == 0)
            printf("\n");
    }
}
```

2 Aufgabe 2: Sortieren mit Merge-Sort

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

//Größe des Arrays definieren
#define MAX 25

//Teilarrays mischen
```

```
void mischen(int *array, int anfang, int mitte, int ende)
{
   //ein Hilfsarray deklarieren (es werden Indizes übergeben,
   //deshalb ende +1
   int hilf[ende+1];
   //Zähler, anfang und ende der beide Teilarrays
   int i, anfang1, anfang2, ende1, ende2;

   //Teilarrays bestimmen
   anfang1 = anfang;
   ende1 = mitte;

   anfang2 = mitte+1;
   ende2 = ende;

   //Zähler auf das erste Element des ersten Teilarrays setzen
   i = anfang1;

   //alle Elemente in der richtigen Reihenfolge von den Teilarrays
   //in das Hilfsarray eintragen
   //bis das Ende eines Teilarrays erreicht wird
   while((anfang1 <= ende1) && (anfang2 <= ende2))
   {
      if(array[anfang1] < array[anfang2])
         hilf[i++] = array[anfang1++];
      else
         hilf[i++] = array[anfang2++];
   }

   //ein Teilarray wurde bis zum Ende (alle Elemente) abgearbeitet
   //jetzt müssen die Reste des zweiten Teilarrays einsortiert werden
   //da ich nicht weiss, welches fertig ist, einfach beide abfragen
   //da mergeSort bis auf die kleinsten Teilarrays heruntergeht,
   //sind die Werte der Teilarrays schon sortiert
   //deshalb können die übrigen Werte einfach der Reihe nach eingetragen
   //werden
```

```
   while(anfang1 <= ende1)
      hilf[i++] = array[anfang1++];

   while(anfang2 <= ende2)
      hilf[i++] = array[anfang2++];

   //Werte aus dem Hilfsarray in das zu sortierende Array übertragen
   for(i=anfang; i<= ende; i++)
      array[i] = hilf[i];
}

//mergeSort-Funktion, teilt die Teilarrays ein und mischt sie
void merge_sort(int *array, int anfang, int ende)
{
   int mitte;

   //mergeSort ist eine rekursive Funktion, deshalb if-Abfrage
   //am Anfang, um festzustellen, wenn alle Arrays sortiert sind
   //wenn der Anfang eines Teilbereichs kleiner als das Ende ist,
   //kann weitergemacht werden
   if (anfang < ende)
   {
      //mitte für Teilarrays bestimmen
      mitte = (anfang + ende)/2;

      //mergeSort für Teilarrays aufrufen
      //dadurch werden die Teilarrays bis auf den kleinstmöglichen,
      //sinnvollen Bereich (zwei Elemente) gebildet
      merge_sort(array, anfang, mitte);
      merge_sort(array, mitte+1, ende);

      //Teilarrays mischen, d.h. in ein großes Array sortieren
      mischen(array, anfang, mitte, ende);
   }
}
```

```
//Hauptprogramm
int main()
{
   //Array und Zähler deklarieren
   int array[MAX];
   int i;

   //Zufallszahlengenerator initialisieren
   srand(time(NULL));

   //Array mit Zufallszahlen belegen
   for(i=0; i<MAX; i++)
      array[i] = rand()%1000;

   //Array vor dem Sortieren ausgeben
   for(i=0; i<MAX; i++)
   {
      printf("%4d ", array[i]);
      if(((i+1) % 10) == 0)
         printf("\n");
   }
   printf("\n");

   //Array sortieren
   merge_sort(array, 0, MAX-1);

   //Array nach dem Sortieren ausgeben
   for(i=0; i<MAX; i++)
   {
      printf("%4d ", array[i]);
      if(((i+1) % 10) == 0)
         printf("\n");
   }
   printf("\n");
}
```