Der strukturierte Datentyp Array

Strukturierte Datentypen unterscheiden sich von den **einfachen** (*int*, *char*, *double*, *bool* usw.) dadurch, dass sie aus anderen Datentypen (einfachen oder strukturierten) zusammengesetzt sind. Die einzelnen Teile der strukturierten Daten heißen **Komponenten**.

Die **charakteristischen Unterscheidungsmerkmale** sind:

- Die **Anzahl** der Komponenten (fest oder variabel)
- Die **Typen** der Komponenten (gleich oder verschieden)
- Der **Zugriff** auf die Komponenten (direkt oder sequentiell)

Ein **Array** (auch Feld oder Vektor genannt) besitzt eine **feste Anzahl** von Komponenten, die alle **denselben Typ** haben müssen. Auf die Komponenten kann man **direkt über** einen in eckige Klammern gesetzten **Index zugreifen**.

Beispiel:

```
int x[5];
x[4] = 13;
```

Hiermit wird ein Feld *x* mit den 5 Komponenten *x*[0], *x*[1], *x*[2], *x*[3], *x*[4] vom Typ *int* vereinbart und der letzten Komponente der Wert **13** zugewiesen.

Achtung: Das erste Element eines Feldes hat immer den Index 0, das letzte Element für ein Feld der Größe *n* hat den Index *n*-1. In unserem Beispiel würde ein Zugriff auf *x*[5] also zu einem Fehler führen. Eine Bereichsüberprüfung durch den Compiler gibt es bei Visual C++ nicht.

Aufgabe 1:

Es sollen *n* Zahlen in ein Feld eingegeben und sortiert wieder ausgegeben werden.

Sortieralgorithmus BubbleSort

- In einer Liste von *n* Elementen werden fortlaufend zwei benachbarte Elemente verglichen und gegebenenfalls vertauscht.
- Falls in einem Durchlauf ein Tausch erfolgt, wird ein „Signal“ gesetzt (*signal* = true).
- Vor jedem neuen Durchlauf wird das „Signal“ wieder entfernt (*signal* = false).
- Es wird solange fortgefahren, bis kein Tausch mehr erfolgt.

Erarbeiten Sie ein Beispiel und ein Struktogramm für „Bubblesort“ und schreiben Sie ein Programm zur Lösung der Aufgabe. Realisieren Sie den Variablentausch dabei durch die Funktion *tausch* aus der Header-Datei (Arbeitsblatt S. 14) *mein.h*.

Aufgabe 2:

Schreiben Sie ein Programm, das nach Eingabe von Zahlen in eine quadratische Matrix (zweidimensionales Feld, Schema aus *n* Zeilen und *n* Spalten) folgende Summen berechnet:

- der Elemente der Hauptdiagonalen,
- der Elemente der Nebendiagonalen und
- aller Elemente.

Beispiel:

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} = \begin{pmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ a_{20} & a_{21} & a_{22} \end{pmatrix} \rightarrow \text{a) 15 b) 15 c) 45}$$

Hinweis:

Verwenden Sie zur Eingabe der Matrixelemente folgende globalen Konstanten- und Typvereinbarungen und folgende Funktion:

```
const int MAX = 11;
typedef int tMatrix[MAX][MAX];
void Eingabe(int n, tMatrix& a)
{ int i, j;
  cout << "Matrixelemente zeilenw. eingeben\n";
  cout << "Elemente einer Zeile durch space-Taste trennen\n";
  for (i = 0; i < n; i++)
    for (j = 0; j < n; j++) cin >> a[i][j];
}
```

Aufgabe 3:

Schreiben Sie ein Programm, das zwei Matrizen der gleichen Dimension nach Eingabe addiert und die Summenmatrix auf dem Bildschirm ausgibt. Nutzen Sie obige Funktion zur Eingabe, realisieren Sie die Matrixausgabe ebenfalls durch eine Funktion.

Zufallszahlen

Zum Erzeugen von Zufallszahlen existieren in `<stdlib.h>` folgende Funktionen:

`rand()` gibt eine ganzzahlige Pseudozufallszahl zwischen 0 und der Konstanten `RAND_MAX` zurück.

`srand(intZahl)` setzt den Zufallsgenerator auf einen beliebigen Startpunkt.

Aufgabe 4:

Interpretieren Sie folgendes Programm (aus der C++-Hilfe, leicht abgewandelt) und nutzen Sie es für die Lösung der nächsten Aufgabe.

```
/* zufhilfe.cpp
This program seeds the random-number generator with the time, then displays
10 random integers.*/
#include <iostream.h>
#include <stdlib.h>
#include <time.h>

int main()
{
  int i;
  /* Seed the random-number generator with current time
  so that the numbers will be different every time we run.*/
  srand((unsigned)time(NULL));
  /* Display 10 numbers. */
  for(i = 0; i < 10; i++) cout << rand() << endl;
  cout << "\nDie Konstante RAND_MAX hat den Wert "
        << RAND_MAX << "\n\n";
  return 0;
}
```

Aufgabe 5:

Machen Sie vom Programm zum Sortieren von Zahlen mit Hilfe von BubbleSort eine Kopie. Ändern Sie das kopierte Programm so ab, dass das Feld aus 100 Komponenten besteht und diese durch Zufallszahlen zwischen 0 und 1000 belegt werden. Lassen Sie sich zunächst die unsortierten, danach die sortierten Zahlen ausgeben.

Hinweis: $0 \leq \text{rand()} < \text{RAND_MAX}$ (z. B. $\text{RAND_MAX} = 2^{15} - 1 = 32\,767$)

$\Rightarrow 0 \leq \text{rand()} * n / \text{RAND_MAX} < n$