4.3.V Datenstrukturen/Arrays – Versuch

4.3.V.1 Ziele

Ein Array ist mit folgenden Werten vereinbart und initialisiert worden:

```
1 double frequenz[5]={105.5, 93.2, 97.0, 100.5, 90.6};
```

Die Inhalte sollen nacheinander ausgeben werden.

4.3.V.2 Fragen

Wie behandelt man die Werte eines Arrays so, dass jeder Wert genau einmal an die Reihe kommt? Anhand der Aktion *Ausgabe des Wertes* soll dies betrachtet werden.

In jedem Abschnitt soll ein Programm erstellt und untersucht werden:

a) Wie lang ist der Quelltext (Programmieraufwand) bei 5 Elementen?

b) Wie lang wäre der Quelltext bei 500 Elementen?

c) Wie viele Bytes hat das Programm bei 5 Elementen? Compilieren Sie dazu das Programm mit der Option -c! Sie erhalten aus beispiel.c das Ergebnis beispiel.o.

d) Wie lang ist die Laufzeit des Programms bei 5 Elementen? Compilieren Sie dazu das Programm ohne die Option -c und testen Sie es, indem Sie es über den Befehl time aufrufen:

Terminal

```
schueler@debian964:~$ gcc beispiel.c
schueler@debian964:~$ time a.out
real    0m0,046s
user    0m0,046s
sys     0m0,000s
```

e) Gibt es Möglichkeiten für Flüchtigkeitsfehler bei der Quelltexteingabe, so dass ein Element vergessen oder doppelt angesprochen wird?

4.3.V.3 Schritt I

Von Hand kann man das Array so ausgeben:

```
1  double frequenz[5]={105.5, 93.2, 97.0, 100.5, 90.6};
2  printf("%f\n", frequenz[0]);
3  printf("%f\n", frequenz[1]);
4  printf("%f\n", frequenz[2]);
5  printf("%f\n", frequenz[3]);
6  printf("%f\n", frequenz[4]);
```

4.3.V.4 Schritt II

Das obige Programmstück funktioniert. Jetzt soll versucht werden, die Werte in einer Schleife auszugeben. Dazu muss (siehe C16!) es so verändert werden, dass immer wieder das Gleiche passiert. In einer Schleife passiert ja auch mehrfach das Gleiche. Dabei hilft es, für den Index statt einer Konstanten eine Variable zu nehmen. Die muss natürlich jedes Mal auf den richtigen Wert gesetzt werden:

```
1  double frequenz[5]={105.5, 93.2, 97.0, 100.5, 90.6};
2  int index=0;
3  printf("%f\n", frequenz[index]);
```

```
4
5     index=1;
6     printf("%f\n", frequenz[index]);
7
8     index=2;
9     printf("%f\n", frequenz[index]);
10
11    index=3;
12    printf("%f\n", frequenz[index]);
13
14    index=4;
15    printf("%f\n", frequenz[index]);
```

4.3.V.5 Schritt III

Das bisherige Programm hat für jedes Array-Element eine Anweisung, die jedes Mal anders ist. Das kann man verhindern:

```
1     int index=0;
2     printf("%f\n", frequenz[index]);
3
4     index=index+1;
5     printf("%f\n", frequenz[index]);
6
7     index=index+1;
8     printf("%f\n", frequenz[index]);
9
10    index=index+1;
11    printf("%f\n", frequenz[index]);
12
13    index=index+1;
14    printf("%f\n", frequenz[index]);
```

4.3.V.6 Schritt IV

Das bisherige Programm besteht nun aus einer Befehlsfolge, die fünfmal wiederholt wird:

```
1     printf("%f\n", frequenz[index]);
2
3     index=index+1;
```

In diesem Schritt ist es daher möglich, eine Schleife zu verwenden, die die Befehlsfolge in den Rumpf nimmt:

```
1  index=0;
2  while(index<5)
3  {
4      printf("%f\n", frequenz[index]);
5      index=index+1;
6  }
```

4.3.V.7 Schritt V

Die oben verwendete Schleife trägt die Merkmale einer Zählschleife. Dazu kann man in C die for-Anweisung verwenden[1]:

```
1 for(int index=0; index<5; index=index+1)
2 {
3     printf("%f\n", frequenz[index]);
4 }
```

4.3.V.8 Ergebnisse

4.3.V.9 Auswertung

[1] Seit C99 ist es erlaubt, die Laufvariable im ersten Ausdruck der for-Klammer vereinbaren.