6.4 Extras/Dynamische Speicherverwaltung

6.4.1 Aufgabe

Eine in einem automatisierten Messsystem anfallende Größe soll systematisch kontrolliert werden. Die Anzahl der Messreihe soll dabei beliebig sein können. Sie (die Anzahl) soll zu Beginn eingegeben werden.

Danach soll aus der Messreihe folgende Kennwerte ermittelt werden:

- $\overline{x_i}$ (arithmetisches Mittel),
- $\sigma_n = \sqrt{\overline{(x_i - \overline{x_i})^2}}$ (Standardabweichung σ_n)

Abbildung 1 zeigt das Struktogramm dazu.

Eingabe anzahl
Für alle lauf von 0 bis anzahl-1
▏ Eingabe mw[lauf]
summe←0
Für alle lauf von 0 bis anzahl-1
▏ summe←mw[lauf]
mittel←summe/anzahl
abwsumme←0
Für alle lauf von 0 bis anzahl-1
▏ abwsumme←abwsumme+(mw[lauf]-mittel)^2
stdabw← $\sqrt{abwsumme/anzahl}$
Ausgabe mittel, stdabw

Abbildung 1: Struktogramm zur Gesamtaufgabe

6.4.2 Lösung mit statischem Arry

`stat0.c` zeigt die Lösung:

```
1  #include <math.h>
2  #include <stdio.h>
3  #define MAXWERTE 4
4
5  int main(void)
6  {
7      int       lauf;
8      int       anzahl;
9      double    mw[MAXWERTE];
10     double    summe, mittel, abwsumme, stdabw;
11
12     printf("Bitte die Anzahl der Messwerte eingeben:");
13     scanf("%i", &anzahl);
14     while(getchar()!='\n'){}
15                                        /* Messwerte einlesen */
16     for(lauf=0;lauf<anzahl;++lauf)
17     {
18         printf("Eingabe Messwert Nr. %i:", lauf+1);
19         scanf("%lf", &mw[lauf]);
```

```
20      }
21                                          /* Mittelwert berechnen */
22      summe = 0.0;
23      for(lauf=0;lauf<anzahl;++lauf)
24          summe = summe+mw[lauf];
25      mittel = summe/anzahl;
26                                          /* Standardabw. berechnen */
27      abwsumme = 0.0;
28      for(lauf=0;lauf<anzahl;++lauf)
29          abwsumme = abwsumme+pow(mw[lauf]-mittel,2);
30      stdabw = sqrt(abwsumme/anzahl);
31                                          /* Ergebnisse ausgeben */
32      printf("Mittelwert=%f\n", mittel);
33      printf("Standardabw.=%f\n", stdabw);
34      return 0;
35  }
```

Es gibt aber ein Problem: Was ist, falls mehr Messwerte vorhanden sind als das Array fasst? Unser Beispielprogramm läuft dann schief.

Terminal

```
schueler@debian964:~$ gcc -o stat0 stat0.c -lm
schueler@debian964:~$ ./stat0
Bitte die Anzahl der Messwerte eingeben:6
Eingabe Messwert Nr. 1:47.3
Eingabe Messwert Nr. 2:42.9
...
Eingabe Messwert Nr. 10:10
Speicherzugriffsfehler
schueler@debian964:~$
```

Aber der Fehler kann abgefangen werden. Falls mehr als `MAXANZAHL` Messwerte eingegeben werden sollen, wird eine Fehlermeldung ausgegeben und das Programm beendet. Dazu ein Auszug aus `stat1.c`:

```
1       printf("Bitte␣die␣Anzahl␣der␣Messwerte␣eingeben:");
2       scanf("%i", &anzahl);
3       if(anzahl>MAXWERTE)
4       {
5           printf("Fehler:␣Anzahl␣zu␣gross!\n");
6           return 1;
7       }
```

6.4.3 Lösung mit VLA

Eine andere Lösung sind Arrays variabler Länge. (*variable length arrays*, VLAs). Es gibt sie in C99. Bei ihnen kann man zur Laufzeit des Programms, und zwar bei Betreten eines Blockes, ein Array anlegen, dessen Größe durch eine *Variable* bestimmt wird. `stat2.c` zeigt das Programm mit VLA.

```
1   #include <math.h>
2   #include <stdio.h>
3   int main(void)
4   {
5       int     lauf;
```

```
6     int       anzahl;
7     double    summe, mittel, abwsumme, stdabw;
8
9     printf("Bitte die Anzahl der Messwerte eingeben:");
10    scanf("%i", &anzahl);
11    /***************** Block beginnt *********************/
12    {
13        double mw[anzahl];              /* Hier ist das VLA */
14                                        /* Messwerte einlesen */
15        for(lauf=0;lauf<anzahl;++lauf)
16        {
17            printf("Eingabe Messwert Nr. %i:", lauf+1);
18            scanf("%lf", &mw[lauf]);
19        }
20                                        /* Mittelwert berechnen */
21        summe = 0.0;
22        for(lauf=0;lauf<anzahl;++lauf)
23            summe = summe+mw[lauf];
24        mittel = summe/anzahl;
25                                        /* Standardabw. berechnen */
26        abwsumme = 0.0;
27        for(lauf=0;lauf<anzahl;++lauf)
28            abwsumme = abwsumme+pow(mw[lauf]-mittel,2);
29        stdabw = sqrt(abwsumme/anzahl);
30    }
31    /***************** Block endet ***********************/
32                                        /* Ergebnisse ausgeben */
33    printf("Mittelwert=%f\n", mittel);
34    printf("Standardabw.=%f\n", stdabw);
35    return 0;
36 }
```

VLAs haben aber auch Nachteile: So sind sie in C++ (und mehreren anderen C-ähnlichen Sprachen) nicht bekannt. Seit C11 sind sie nur noch als optional im Standard enthalten. Man kann sie nicht initialisieren. Falls die Anzahl der Elemente zu groß ist, wird das Programm durch eine Schutzverletzung beendet; es passiert ein Stack-Überlauf, falls ein Prozess zu irgendeinem Zeitpunkt zu viel Speicher im Stack-Bereich (=Variablen, Parameter und Rücksprungadressen) belegt[1]

6.4.4 Lösung mit dynamischem Speicher

Viele Programmierer benutzen deshalb gerne so genannten *dynamischen Speicher*. `dyn0.c` zeigt, wie das geht:

```
1  #include <math.h>
2  #include <stdio.h>
3  #include <stdlib.h>
4
5  int main(void)
6  {
7      int       lauf;
8      int       anzahl;
```

[1] Unter Linux kann man die für den Stack verfügbare Speichermenge mit dem Befehl `ulimit -s` ermitteln und anpassen.

```
9     double  *mw;
10    double  summe, mittel, abwsumme, stdabw;
11
12    printf("Bitte die Anzahl der Messwerte eingeben:");
13    scanf("%i", &anzahl);
14    while(getchar()!='\n'){}
15                                        /* Dynam. Speicher holen */
16    mw=malloc(anzahl*sizeof(double));
17    if(mw==NULL)
18    {
19        printf("malloc-Fehler!\n");
20        return 1;
21    }
22                                        /* Messwerte einlesen */
23    for(lauf=0;lauf<anzahl;++lauf)
24    {
25        printf("Eingabe Messwert Nr. %i:", lauf+1);
26        scanf("%lf", &mw[lauf]);
27    }
28                                        /* Mittelwert berechnen */
29    summe = 0.0;
30    for(lauf=0;lauf<anzahl;++lauf)
31        summe = summe+mw[lauf];
32    mittel = summe/anzahl;
33                                        /* Standardabw. berechnen */
34    abwsumme = 0.0;
35    for(lauf=0;lauf<anzahl;++lauf)
36        abwsumme = abwsumme+pow(mw[lauf]-mittel,2);
37    stdabw = sqrt(abwsumme/anzahl);
38                                        /* Dyn. Speicher freigeben */
39    free(mw);
40                                        /* Ergebnisse ausgeben */
41    printf("Mittelwert=%f\n", mittel);
42    printf("Standardabw.=%f\n", stdabw);
43    return 0;
44 }
```

Mit der Bibliotheksfunktion `p=malloc(n)` holt sich das Programm vom Betriebssystem einen zusammenhängenden Block mit einer Größe von `n` Bytes. Die Startadresse wird in `p` geschrieben. Falls nicht mehr genug Speicher da ist, erhält `p` den Wert `NULL`.

Danach kann man mit dem Speicherbereich ab Adresse `p` wie mit einem normalen (statisch genannten) Array arbeiten.

Sobald man diesen Speicherbereich nicht mehr braucht, gibt man ihn mit `free(p)` wieder zurück.

Für `malloc()` und `free()` muss man die Headerdatei `<stdlib.h>` einbinden.