6.4.F Extras/Dynamische Speicherverwaltung – Ergänzungen und Bilder

6.4.F.1 Record mit flexibler Array-Komponente

In einem Record ist (seit C99) folgendes erlaubt:

```
1  struct beispieltyp
2  {
3     double a; // <---- irgendwas
4     double b; //
5     int c[];  // <----- Hier hingucken
6  };
```

Die letzte Komponente darf ein Array ohne Größenangabe sein. Man nennt sie dann *flexible array member*, übersetzt flexible Array-Komponente. Nur die letzte Komponente darf ein *flexible array member* sein.

Das Array hat also keine Größenangabe. Wieviel Platz wird nun für dieses Array reserviert? Das Programm `flexkomp1.c` gibt die Antwort:

```
1  #include <stdio.h>
2
3  struct beispieltyp
4  {
5     double a;
6     double b;
7     int c[];  // <----- flexible array member
8  };
9  struct vergleichstyp
10 {
11    double a;
12    double b;
13 };
14 int main(void)
15 {
16    struct beispieltyp mit;
17    struct vergleichstyp ohne;
18    printf("Groesse mit Array: %zu Bytes\n", sizeof(mit));
19    printf("Groesse ohne Array: %zu Bytes\n", sizeof(ohne));
20    return 0;
21 }
```

Terminal

```
schueler@debian964:~$ gcc flexkomp1.c
schueler@debian964:~$ a.out
Groesse mit Array: 16 Bytes
Groesse ohne Array: 16 Bytes
```

Das Ergebnis ist ernüchternd: Für das Array wird überhaupt kein Speicherplatz reserviert, also kann es kein einziges Element aufnehmen (nicht einmal `c[0]`).

An dieser Stelle ist aber ein Trick möglich, und für diesen Trick braucht man unbedingt `malloc`. Wenn man mit `malloc` für solch ein Record mehr Speicher reserviert als nötig, dann kann man den zu viel reservierten Speicher für die Elemente des Arrays verwenden (siehe `flexkomp2.c`):

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
```

```
4  struct beispieltyp
5  {
6     double a;
7     double b;
8     int c[];  // <---- flexible array member
9  };
10 int main(void)
11 {
12    struct beispieltyp *precord;
13    precord=malloc(sizeof(struct beispieltyp)+3*sizeof(int));
14    precord->c[0]=97;
15    precord->c[1]=38;
16    precord->c[2]=65;
17    /* ... */
18    free(precord);
19    return 0;
20 }
```

In Zeile 13 wird mehr Speicherplatz angefordert als nötig, nämlich Platz für drei zusätzliche `int`-Variablen. Damit darf das Array mit maximal drei Elementen gefüllt werden.

Das folgende Programm zeigt, wie man damit einen String-Datentyp für unterschiedlich lange Strings bauen und verwalten kann (`flexkomp3.c`):

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4
5  struct stringtyp
6  {
7     unsigned int len;
8     char s[];  // <---- flexible array member
9  };
10
11 struct stringtyp *strmalloc(unsigned int len)
12 {
13    struct stringtyp *erg;
14    erg=malloc(sizeof(struct stringtyp)+len);
15    if(erg!=NULL)
16    {
17       erg->len=len;
18       memset(erg->s,'\0',len);
19    }
20    return erg;
21 }
22
23 int main(void)
24 {
25    struct stringtyp *ps, *pt;
26    ps=strmalloc(20);
27    strcpy(ps->s,"Hallo!␣");
28    pt=strmalloc(200);
29    strcpy(pt->s,"So␣ist␣es.␣");
30
31    struct stringtyp *sarr[2]={ps,pt};
```

```
32
33      for(int lauf=0; lauf<2; ++lauf)
34      {
35          printf("%s\n", sarr[lauf]->s);
36      }
37
38      for(int lauf=0; lauf<2; ++lauf)
39      {
40          free(sarr[lauf]);
41      }
42      return 0;
43  }
```