5.2 Datenstrukturen II/Records und Funktionen

5.2.1 Einführung

Das Arbeiten mit Records sollte den Umgang mit den Personendaten vereinfachen. Nun ist die Frage: Verhalten sich Records bei der Benutzung mit Funktionen eher wie Arrays oder eher wie elementare Datentypen (`int`, `float` usw.)?

5.2.2 Records als Parameter

In `buchtyp2.c` werden Records zur Übergabe von Daten benutzt. Offenbar können Records ohne Probleme und Besonderheiten an Funktionen ausgegeben werden, ganz so wie elementare Datentypen. Die Typdefinition muss ab jetzt allerdings vor allen Funktionen erfolgen, damit in allen Funktionen bekannt ist, welche Komponenten der jeweilige Record hat.

```
1  #include <stdio.h>
2  struct buchtyp
3  {
4     char autor[31];
5     char titel[31];
6     unsigned short auflage;
7  };
8  /*********************************/
9  void printbuch(struct buchtyp buch)
10 {
11    printf("Autor:%s\n", buch.autor);
12    printf("Titel:%s\n", buch.titel);
13    printf("Aufl.:%hu\n",buch.auflage);
14 }
15 /*********************************/
16 int main(void)
17 {
18    struct buchtyp bucha, buchb;
19    printf("A:");scanf("%30[^\n]",buchb.autor);while(getchar()!='\n'){}
20    printf("T:");scanf("%30[^\n]",buchb.titel);while(getchar()!='\n'){}
21    printf("A:");scanf("%hu",&buchb.auflage);while(getchar()!='\n'){}
22    bucha = buchb;
23    bucha.auflage=123;
24    printbuch(buchb);
25    return 0;
26 }
```

5.2.3 Records als Rückgabewerte

Im Programm `buchtyp3.c` wird ein Record zur Rückgabe von Daten aus einer Funktion benutzt. Also ist auch eine Rückgabe von Funktionsergebnissen ohne Probleme möglich.

```
1  #include <stdio.h>
2  struct buchtyp
3  {
4     char autor[31];
5     char titel[31];
6     unsigned short auflage;
7  };
8  /*********************************/
```

```
9  void printbuch(struct buchtyp buch)
10 {
11     printf("Autor:%s\n", buch.autor);
12     printf("Titel:%s\n", buch.titel);
13     printf("Aufl.:%hu\n",buch.auflage);
14 }
15 /*********************************/
16 struct buchtyp scanbuch(void)
17 {
18     struct buchtyp buch;
19     printf("A:");scanf("%30[^\n]",buch.autor);while(getchar()!='\n'){}
20     printf("T:");scanf("%30[^\n]",buch.titel);while(getchar()!='\n'){}
21     printf("A:");scanf("%hu",&buch.auflage);while(getchar()!='\n'){}
22     return buch;
23 }
24 /*********************************/
25 int main(void)
26 {
27     struct buchtyp bucha, buchb;
28     buchb = scanbuch();
29     bucha = buchb;
30     bucha.auflage=123;
31     printbuch(buchb);
32     return 0;
33 }
```

Im Unterprogramm `scanbuch()` kommt wieder die Grundstruktur einer Funktion mit Rückgabe zu Vorschein:

```
1 struct buchtyp scanbuch(void) /* Rueckg.-Typ: struct buchtyp */
2 {
3     struct buchtyp buch; /* Rueckgabe-Variable anlegen */
4     /* ... */           /* Rueckgabe-Variable fuellen */
5     return buch;         /* Ende und Rueckgabe         */
6 }
```

5.2.4 Zeiger auf Record als Parameter

Das folgende Beispiel zeigt einen weiteren Fall. Ein Record soll durch eine Funktion verändert werden. Darum wird ein Zeiger auf das Record an die Funktion übergeben.

```
1 void erhoehe_auflage(struct buchtyp* pbuch)
2 {
3     (*pbuch).auflage = (*pbuch).auflage +1;
4 }
```

Dabei sind die Klammern um `*pbuch` nötig, damit zuerst der Inhalts- und dann erst der Punkt-Operator ausgeführt wird. Der Punkt-Operator hat nämlich eine höhere Priorität als der Inhalts-Operator und würde sonst zuerst ausgeführt, und zwar auf den Zeiger. Der hat aber gar keine Komponente namens `auflage`, was den Compiler zu einer Fehlermeldung veranlasst. Für die umständliche Schreibweise mit Klammern, Stern und Punkt gibt es jedoch eine Abkürzung, mit der man Zeile 3 so schreiben kann[1]:

[1] Das Minus- und das größer-als-Zeichen sollen hier zusammen einen Pfeil symbolisieren.

```
1    pbuch->auflage = pbuch->auflage +1;
```