1.9 Programmstrukturen/Mehrfachauswahl

1.9.1 Wiederholte `if`-Abfragen

Das Menüprogramm aus dem vorigen Thema ist etwas unübersichtlich: Es gibt sehr viele `if`-Abfragen. Aber alle diese Abfragen beziehen sich auf die gleiche Variable (in diesem Fall das Ergebnis der Tastatureingabe mit `scanf()`). Bei vielen anderen Programmen ist es ähnlich; hier ein Beispiel (`olympiade1.c`):

```
1  #include <stdio.h>
2  int main(void)
3  {
4     int jahreszahl;
5     printf("Bitte␣Jahreszahl␣eingeben:␣");
6     scanf("%i", &jahreszahl);
7     printf("%i␣fanden␣Olympische␣Spiele␣statt␣in:␣", jahreszahl);
8     if(jahreszahl==1968)
9     {
10          printf("Mexiko");
11    }
12    else
13    {
14       if(jahreszahl==1972)
15       {
16          printf("BRD");
17       }
18       else
19       {
20          if(jahreszahl==1976)
21          {
22             printf("Kanada");
23          }
24          else
25          {
26             if(jahreszahl==1980)
27             {
28                printf("UdSSR");
29             }
30             else
31             {
32                if(jahreszahl==1984)
33                {
34                   printf("USA");
35                }
36                else
37                {
38                   printf("weiss␣nicht");
39                }
40             }
41          }
42       }
43    }
44    printf("\n");
45    return 0;
46 }
```

1.9.2 Abhilfe

Die meisten Programmiersprachen bieten daher eine Programmstruktur an, die für diesen Sonderfall Abhilfe bietet, die *Mehrfachauswahl*, auch *Fallunterscheidung* genannt: Je nachdem, welchen Wert ein bestimmter Ausdruck hat, wird genau *ein* Programmblock durchlaufen. Das zugehörige Struktogramm-Symbol sieht man in Abbildung 1. In der Sprache C wird das folgendermaßen

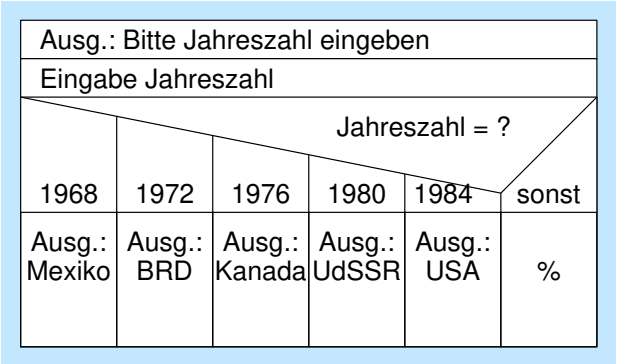

Abbildung 1: Struktogramm zur Mehrfachauswahl

ausgedrückt:

```c
#include <stdio.h>
int main(void)
{
    int jahreszahl;
    printf("Bitte Jahreszahl eingeben: ");
    scanf("%i", &jahreszahl);
    printf("%i fanden Olympische Spiele statt in: ", jahreszahl);
    switch(jahreszahl)
    {
        case 1968:
            printf("Mexiko");
            break;
        case 1972:
            printf("BRD");
            break;
        case 1976:
            printf("Kanada");
            break;
        case 1980:
            printf("UdSSR");
            break;
        case 1984:
            printf("USA");
            break;
        default:
            printf("weiss nicht");
    }
    printf("\n");
    return 0;
}
```

`(jahreszahl)` ist in Klammern der Bedingungsausdruck, von dessen Ergebnis der weitere Programmablauf abhängt. In C muss dieser Ausdruck ganzzahlig sein. Von den geschweiften Klammern wird die Menge der Anweisungen, die mit der Mehrfachauswahl zu tun haben, begrenzt[1]. `case` ist das zweite Schlüsselwort. Es markiert keine Anweisung, sondern ein *Sprungziel* (daher auch Marke genannt). Das Sprungziel wird ergänzt durch eine ganzzahlige Konstante (z.B. `1980`) und abgeschlossen durch einen Doppelpunkt.

Die Mehrfachauswahl funktioniert so: Falls der Bedingungsausdruck den gleichen Wert wie die Konstante hat, springt der Programmfluss an die erste Anweisung hinter der Sprungmarke – und geht von dort weiter. Falls das nicht der Fall ist, springt er an die erste Anweisung hinter der Marke mit dem besonderen Namen `default` (das dritte Schlüsselwort in diesem Zusammenhang).

Leider ist die `switch()`-Anweisung in C nur eine bedingte Sprunganweisung, ähnlich wie in BASIC oder FORTRAN Konstruktionen der Art `ON ERGEBNIS GOTO ...`). Der Nachteil dabei: Anders als bei `if(){}else{}` läuft der Programmfluss von einer Marke aus bis zum Ende der Mehrfachauswahl durch. Um das zu verhindern, muss vor der nächsten Sprungmarke ein viertes Schlüsselwort mit dem Namen `break` stehen. `break` ist eine Sprunganweisung an das Ende der Mehrfachauswahl.

1.9.3 Vor- und Nachteile der Mehrfachauswahl in C

+ Der Auswahlausdruck wird nur noch einmal abgefragt (anstatt mehrmals bei `olympiade1.c`). Somit kann ein Ausdruck wie `getchar()` zur Auswahl dienen.
+ Die Mehrfachauswahl kann schnelleren Code erzeugen, da derr Auswahlausdruck nur einmal berechnet werden muss.
+ Die Mehrfachauswahl ist oft optisch kürzer und übersichtlicher.
– Die case-Marken müssen jeweils eine ganzzahlige Konstante enthalten; die `if`-Anweisung dagegen erlaubt auch Gleitkomma-Ergebnisse.
– Auf diese Art können keine Zeichenketten miteinander verglichen werden.

1.9.4 `break`, `continue`, `return` und `goto`

Die `break`-Anweisung innerhalb der Mehrfachauswahl gehört zu einer Gruppe vonProgrammstrukturen, die eigentlich entweder zu älteren oder zu systemnäheren Sprachen gehören: Zu den *Sprunganweisungen*. Mit einer Sprunganweisung wird der bisherige Ablauf verlassen und an einer anderen Stelle wieder fortgesetzt. Die ursprünglichste Sprunganweisung ist `goto`. Mit `goto` kann man an eine beliebige Stelle[2] springen, die man mit einem Doppelpunkt markiert:

```
1  #include <stdio.h>
2  int main(void)
3  {
4     int zahl;
5  vorher:
6     printf("Zahl zwischen 1 und 6: ");
7     scanf("%i", &zahl);
8     if(zahl<1)
9        goto zuklein;
10    if(zahl>6)
11       goto zugross;
12    goto fertig;
13 zuklein:
```

[1] Man kann sie – wie bei `if` oder `while` – weglassen, wenn man nur eine solche Anweisung hat. Das tritt in der Praxis aber fast nie auf.

[2] Nur innerhalb desselben Funktionsbausteins

```
14      printf("Zahl zu klein!\n");
15      goto vorher;
16  zugross:
17      printf("Zahl zu gross!\n");
18      goto vorher;
19  fertig:
20      printf("Danke!\n");
21      return 0;
22  }
```

Das hat zwei Nachteile:

a) Die Blockstruktur kann außer kraft gesetzt werden

b) Man kann beliebig komplizierte Programmstrukturen aufbauen

Daher ist die Sprunganweisung mit `goto` in vielen aktuellen Programmiersprachen nicht mehr vorhanden; in C gibt es sie zwar; aber sie wird selten benutzt. Die anderen Sprunganweisungen sollen hier einmal kurz vorgestellt werden:

- `return` verlässt den aktuellen Funktionsbaustein (siehe später); gegebenenfalls kann man der Anweisung einen Wert mitgeben
- `break` verlässt sofort die aktuelle Mehrfachauswahl oder die aktuelle Schleife; bei einer Verschachtelung wird nur die innerste Mehrfachauswahl oder Schleife verlassen
- `continue` verlässt sofort den aktuellen Durchlauf der aktuellen Schleife; bei einer Verschachtelung wird nur der Durchlauf der innersten Schleife verlassen

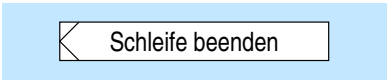

Abbildung 2: Darstellung eines Sprunges im Struktogramm

Im Struktogramm wird die Sprunganweisung mit einem Pfeil innerhalb des Rechtecks dargestellt; (Abbildung 2) die Beschriftung kann lauten:

a) `return`: „Funktion beenden“

b) `break` (in Schleifen): „Schleife beenden“

c) `continue`: „Zum Schleifenkopf“

Bei `continue` gibt es übrigens einen Unterschied zwischen der kopfgesteuerten Schleife (`while`) und der Zählschleife (`for`):

```
1  #include <stdio.h>
2  int main(void)
3  {
4      int i=0;
5      while(i<6)
6      {
7          printf("%i ", i);
8          if(i==2)
9          {
10             i=4;
11             continue;
```

```
12          }
13          ++i;
14      }
15      printf("\n");
16      return 0;
17  }
```

Terminal

```
schueler@debian964:~$ gcc continue_while.c
schueler@debian964:~$ a.out
0 1 2 4 5
```

```
1   #include <stdio.h>
2   int main(void)
3   {
4       int i;
5       for(i=0; i<6; ++i)
6       {
7           printf("%i ", i);
8           if(i==2)
9           {
10              i=4;
11              continue;
12          }
13      }
14      printf("\n");
15      return 0;
16  }
```

Terminal

```
schueler@debian964:~$ gcc continue_for.c
schueler@debian964:~$ a.out
0 1 2 5
```

Bei beiden Schleifen springt man mit `continue` zum Kopf der Schleife, das ist also gleich. Aber nur bei der Zählschleife wird *vor* dem nächsten Durchlauf noch die Weiterschaltanweisung `++i` ausgeführt. Somit haben die beiden Programme unterschiedliches Verhalten und unterschiedliche Struktogramme (Abbildung 3).

Es gilt heutzutage als guter Stil, die Sprunganweisungen `break` und `continue` in Schleifen weitestgehend zu vermeiden. Auf `goto` verzichtet man ganz.

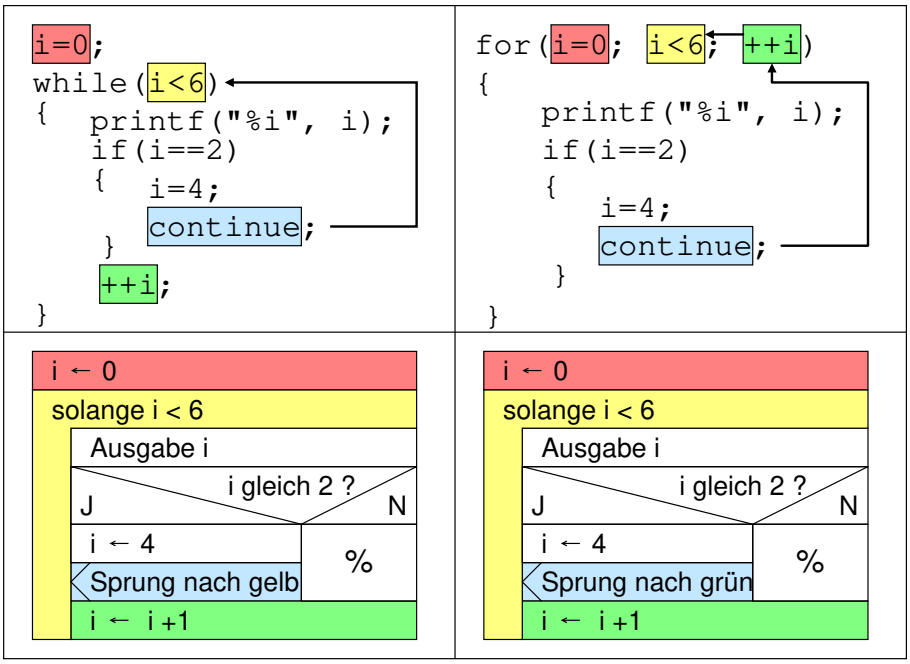

Abbildung 3: `continue` bei `while` und `for`