LANDESBERUFSSCHULE 4 SALZBURG

Informatik

Operatoren, Schleifen und Entscheidungen

LBS 4

Dieses Skript dient als zusätzliche Lernunterlage für Informatik

Inhalt

Verzweigungen 1:

Dienen dazu Entscheidungen im Programmablauf zu erstellen. Es gibt zwei bekannte Strukturen. IF-ELSE und SWITCH-CASE

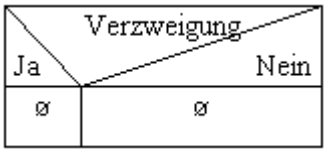
IF-ELSE

Diese Struktur dient zur einfachen Entscheidung. Nach der IF-Bedingung kann ein ELSE folgen. Muss aber nicht! If Anweisungen können verschachtelt werden. Also mehrere IF-Anweisungen hintereinander.

Aufbau einer IF-ELSE Struktur

```
IF <Bedingung = wahr>
        Anweisung 1;
ELSE IF <Bedingung = wahr>
        Anweisung 2;
ELSE <keine IF Bedingung = wahr>
        Anweisung 3;
```

Struktogramm für IF-ELSE



Beispiel C:

```
//if1.c
#include <stdio.h>
int main()
{
int zahl;
printf("Bitte geben sie eine Zahl ein : ");
scanf("%d", &zahl);
if(zahl<10)
{
printf("Die Zahl %d ist kleiner als 10", zahl);
}
else
{
printf("Die Zahl %d ist groesser/gleich 10", zahl);
}
printf("Bye\n");
return 0;
}
```

Operatoren:

Prinzipiell unterscheidet man zwischen fünf Arten von Operatoren.

Zuweisung	Arithmetische	Inkrement /Dekrement	Logische	Binäre
+=, -=, *=, /=	+, -, *, /, %,	++, --	&&, ||, <, >,	&, |, ^,~, <<, >>

Beispiel: Zuweisung

```
+=    a+= b      ist das selbe wie    a= a+b
-=    a-= b      ist das selbe wie    a= a-b
*=    a*= b      ist das selbe wie    a= a*b
/=    a/= b      ist das selbe wie    a= a/b
```

Beispiel: inkrementieren, dekrementieren

```
int i=1;
z= i++;
//bewirkt folgendes: 1. Schritt: z=i;
// 2. Schritt: i=i+1;
//z hat den Wert 1 und i hat den Wert 2
aber:
i=1;
z=++i;
//bewirkt folgendes: 1. Schritt: i=i+1;
// 2. Schritt: z=i;
//z hat den Wert 2 und i hat den Wert 2
```

Beispiel C-Code

```
//increment1.c
#include <stdio.h>
int main(){
int i=1;
printf("i=%d\n",i); /*Ausgabe: i=1*/
i++;
printf("i=%d\n",i); /* Ausgabe: i=2*/
printf("i=%d\n",i++); /* Ausgabe: i=2*/
printf("i=%d\n",i); /* Ausgabe: i=3*/
printf("i=%d\n",++i); /* Ausgabe: i=4*/
return 0;
```

Rangfolge von Operatoren

Operator	Rang	Hinweis
(), [ ], ->	1	Klammern, Elementsektor (Pointer)
!, ++, --	2	Verneinung, Inkrement, Dekrement
***, &, sizeof(type)**	2	Pointer, Adressoperator,
***, /, %**	3	Punktrechnung
+, -	4	Strichrechnung
<< >>	5	Schiebeoperator
<, <=, >, >=	6	kleiner, größer
==, !=	7	Gleichheit, Ungleichheit
&	8	Bitweise AND
^	9	Bitweise XOR
|	10	Bitweise OR
&&	11	logisch AND
||	12	logisch ODER
=, +=, -=, *=, usw	13	Zuweisungsoperator

Die Operatoren der Rangstufen 2 und 13 binden nach rechts, alle anderen fassen von links her zusammen.

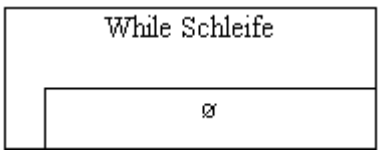
Schleifen

Schleifen dienen zum einfachen wiederholten Aufruf von Quellcode. Es gibt kopfgesteuerte Schleifen und fußgesteuerte Schleifen.

WHILE-Schleife

Bei der while-Schleife wird zuerst der Ausdruck überprüft. Wenn dieser WAHR (ungleich 0) ergibt werden die Schleifenanweisungen ausgeführt. Anschließend wird wieder der Ausdruck überprüft. Die while-Schleife endet, wenn der Ausdruck FALSCH (gleich 0) ergibt.

Als Struktogramm sieht die WHILE-Schleife folgendermaßen aus:



Der Aufbau einer WHILE-Schleife: (abweisende Schleife)

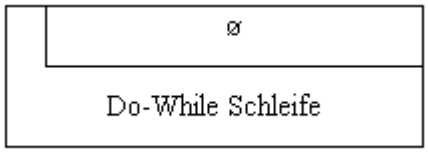
```
WHILE(Laufbedingung){
        [anweisungen]
        [break]
        [continue]
}
```

Beispiel in C:

```
//while1.c
//Zahlen von 1 bis 10 zusammenzählen
#include <stdio.h>
int main(){
int zahl=1, sum=0;
while(zahl<=10)
{
sum= sum + zahl;
zahl++; /*Zahl + 1*/
}
printf("Die Summe aus 1+2+3...+9+10= %d\n", sum);
return 0;
}
```

DO-WHILE Schleife (nicht abweisende Schleife)

Wird mindestens einmal durchlaufen. Eignet sich zum Erstellen für ein einfaches Menü.




```
DO{
        [Code]
        [break]
        [continue]
} WHILE(Laufbedingung);
```

Beispiel in C:

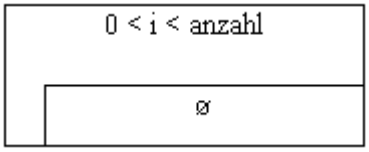
```
//do1.c
#include <stdio.h>
int main()
{
int auswahl;
do{
printf("-1- Auswahl1\
printf("-2- Auswahl2\
printf("-3- Auswahl3\
printf("-4- Programmende
printf("\n\nIhre Auswahl : ");
```

```
scanf("%d",&auswahl);
switch(auswahl)
{
case 1 : printf("Das war Auswahl 1");
case 2 : printf("Das war Auswahl 2");
case 3 : printf("Das war Auswahl 3");
case 4 : printf("Programmende");
default: printf("Unbekannte Auswahl");
}
}while(auswahl!=4);
return 0;
}
```

FOR-Schleife

Sie wird auch Zählschleife genannt. Geeignet wenn man weiß, wie oft der Code durchlaufen werden sollte.

Das Symbol für das Struktogramm:



Aufbau einer FOR-Schleife:

```
for (Initialisierung ; Abbruchbedingung ; Zähler)
{
        [break]
        [continue]
        [Anweisung(en)]
}
```

Beispiel C:

```
//for1.c
#include <stdio.h>
int main(){
int i;
for(i=1; i<=10; i++)
{
printf("Das ist der %d Durchlauf", i);
}
return 0;
}
```

Break und Continue

Die Anweisung **break;** bewirkt das sofortige Beenden der Schleife oder der Switch-Struktur

Die Anweisung **continue;** gilt nur für Schleifen (for, while, do) und bewirkt, dass die Anweisungen nach dem continue nicht mehr ausgeführt werden und direkt der nächste loop beginnt. Im Gegensatz zum break wird die Schleife nicht verlassen.

Beispiel C:

```
//continue.c
#include <stdio.h>
int main()
{
int i;
  for(i=1; i<=20; i++)
 {
   if(i%2)
    continue;
   printf("%d ",i);
 }
return 0;
}
```

Verzweigungen 2

SWITCH-CASE

Mit einer SWITCH-CASE Struktur wird eine Mehrfachauswahl bereitgestellt. IF-ELSE wird bei mehreren Entscheidungen sehr unübersichtlich. Darum ist es besser auf SWITCH-CASE zurück zu greifen

Aufbau einer SWITCH-CASE

Als erstes wird in CASE() ein Ausdruck ausgewertet. Das Ergebnis wird in den einzelnen CASE Anweisungen verglichen und bei Übereinstimmung der dazugehörige Befehl ausgeführt. Am Ende kann ein Default stehen. Dieser wird ausgeführt, wenn keine CASE-Anweisung übereinstimmt.

Jede CASE Klausel **muss mit einem break** beendet werden, ansonsten werden ohne Überprüfung alle weiteren Anweisungen der kommenden CASE‘s zusätzlich (ohne CASE-Klauseln) ausgeführt.

Beispiel C:

```
//Die default-Anweisung kann entfallen.
//switch1.c
#include <stdio.h>
int main(){
int a;
printf("Bitte eine Zahl von 1-5 eingeben : “);
scanf("%d",&a);
switch(a)
{
case 1 : printf("Das war eins\n");
break;
case 2 : printf("Das war zwei\);
break;
case 3 : printf("Das war drei\n");
break;
case 4 : printf("Das war vier\n");
break;
case 5 : printf("Das war fünf\n");
break;
default: printf(“Keine gültige Zahl”);
} /*Ende switch*/
return 0;
}
```