Übungspaket 10

Fallunterscheidungen

Übungsziele:

1. Umgang mit der einfachen Fallunterscheidung,
2. sowie mehrfachen Fallunterscheidung und
3. problemangepasster Auswahl

Skript:

Kapitel: 24 und 25

Semester:

Wintersemester 2022/23

Betreuer:

Benjamin, Thomas und Ralf

Synopsis:

Die meisten C-Programme können nicht einfach von oben nach unten abgearbeitet werden. Zwischendurch muss der Programmablauf immer mal wieder in die eine oder andere Richtung geändert werden, wie wir es schon bei den ersten Programmieraufgaben gesehen haben. In diesem Übungspaket gehen wir nun genauer auf die einfache sowie mehrfache Fallunterscheidung ein.

Teil I: Stoffwiederholung

Aufgabe 1: Einfache Fallunterscheidung

Erkläre die einfache Fallunterscheidung anhand folgender Punkte mit eigenen Worten:

Struktogramm:

Schrittweise Verfeinerung:

C Syntax:

Zwei konkrete Beispiele:

Aufgabe 2: Einfache Fallunterscheidung geschachtelt

Erkläre an *einem Beispiel* eine geschachtelte Fallunterscheidung (eine innere Fallunterscheidung in einer äußeren)

Schrittweise Verfeinerung:

C-Beispiel:

Aufgabe 3: Mehrfache Fallunterscheidung

Erkläre mit eigenen Worten die mehrfache Fallunterscheidung anhand folgender Punkte:

Schrittweise Verfeinerung:

Struktogramm:

Ein konkretes Beispiel:

Teil II: Quiz

Aufgabe 1: Ablauf: einfache Fallunterscheidung

Gegeben sei folgendes kleines Programm:

```
 1 #include <stdio.h>
 2
 3 int main( int argc, char **argv )
 4     {
 5         int i;
 6         scanf( "%d", & i );
 7         if ( i > 1 )
 8            printf( "i= %d\n", i );
 9         if ( i > 1 && i < 6 )
10            printf( "i= %d\n", i );
11         if ( i > 1 || i == -4 )
12            printf( "i= %d\n", i );
13         if ( i < 0 )
14            if ( i > -10 )
15               printf( "i= %d\n", i );
16            else printf( "i= %d\n", i );
17     }
```

Notiere, welche `printf()`-Zeilen für welche Werte der Variablen `i` ausgeführt werden:

	Ausgeführte Zeilen				
i	8	10	12	15	16
-20	.....	.....	.....	.....	.....
-5	.....	.....	.....	.....	.....
-4	.....	.....	.....	.....	.....
0	.....	.....	.....	.....	.....
3	.....	.....	.....	.....	.....
8	.....	.....	.....	.....	.....

Mehrfache Fallunterscheidung:

Gegeben sei folgendes kleines Programm:

```
 1 #include <stdio.h>
 2
 3 int main( int argc, char **argv )
 4     {
 5        int i;
 6        scanf( "%d", & i );
 7        switch( i )
 8        {
 9           case  0: printf( "i= %d\n", i ); break;
10           case  2: printf( "i= %d\n", i );
11           case  4: printf( "i= %d\n", i );
12           case  6: printf( "i= %d\n", i ); break;
13           case -1: printf( "i= %d\n", i );
14           case -3: printf( "i= %d\n", i ); break;
15           default: printf( "i= %d\n", i ); break;
16        }
17     }
```

Notiere, welche `printf()`-Zeilen für welche Werte der Variablen `i` ausgeführt werden:

	Ausgeführte Zeilen						
i	9	10	11	12	13	14	15
-4	.....	.....	.....	.....	.....	.....	.....
-3	.....	.....	.....	.....	.....	.....	.....
-2	.....	.....	.....	.....	.....	.....	.....
-1	.....	.....	.....	.....	.....	.....	.....
0	.....	.....	.....	.....	.....	.....	.....
1	.....	.....	.....	.....	.....	.....	.....
2	.....	.....	.....	.....	.....	.....	.....
3	.....	.....	.....	.....	.....	.....	.....
4	.....	.....	.....	.....	.....	.....	.....
5	.....	.....	.....	.....	.....	.....	.....
6	.....	.....	.....	.....	.....	.....	.....
7	.....	.....	.....	.....	.....	.....	.....

Teil III: Fehlersuche

Aufgabe 1: Fehler in Fallunterscheidungen

In den folgenden Programmen sind Dr. Dream ein paar kleine Fehler unterlaufen. Finde diese und korrigiere sie direkt im Quelltext.

Einfache Fallunterscheidung:

```
 1 #include <stdio.h>
 2
 3 int main( int argc, char **argv )
 4     {
 5        int i, j, k;
 6        scanf( "%d",   i ); scanf( "%z", & j );
 7        scanf( "%d", & k )
 8        if ( i > j )
 9           if ( i == k )
10              printf( "i= %d\n", i );
11           else printf( "j= %d k= %d\n", j, k  ;
12        else {
13                if ( k == -1 || -3 )
14                   printf( "so doof hier\n" );
15                   printf( "kaum zu glauben\n" );
16                else printf( "na servus\n" );
17     }
```

Mehrfache Fallunterscheidung:

```
 1 #include <stdio.h>
 2
 3 int main( int argc, char **argv )
 4     {
 5        int i;
 6        scanf( "%d", & i );
 7        switch( i )
 8        {
 9           case 0: printf( "huhu " );
10           case 1: printf( "doofi\n" ); break;
11           case 2  printf( "wer ist hier ein doedel?\n" );
12           csae 3: printf( "jetzt reicht's aber\n" );
13           DEFAULT: printf( "jetzt ist endgueltig schluss\n" );
14        }
15     }
```

Teil IV: Anwendungen

Aufgabe 1: Korrelation zweier Variablen

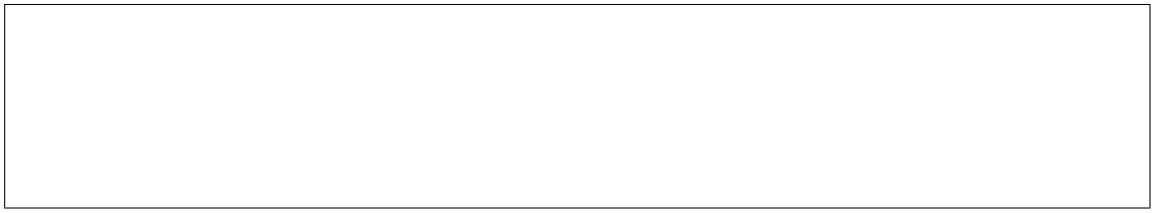
In dieser Aufgabe verwenden wir die *Fallunterscheidung*, um die Korrelation zweier Eingangsgrößen zu ermitteln. Die Programmentwicklung erfolgt natürlich wieder gemäß der Struktur des Software Life Cycle.

1. Aufgabenstellung

Gegeben seien zwei Variablen `i` und `j` vom Typ `int`. Diese beiden Variablen haben beliebige Werte. Ferner haben wir eine Ergebnisvariable `ergebnis`, die ebenfalls vom Typ `int` ist. Schreibe ein Programm, das die beiden Variablen wie folgt korreliert:

1. Ergebnis `1`, wenn beide Variablen das gleiche Vorzeichen haben
2. Ergebnis `-1`, wenn beide Variablen ein unterschiedliches Vorzeichen haben
3. Ergebnis `0`, wenn mindestens eine der beiden Variablen den Wert null hat.

2. Pflichtenheft: Aufgabe, Eingabe, Ausgabe, Sonderfälle



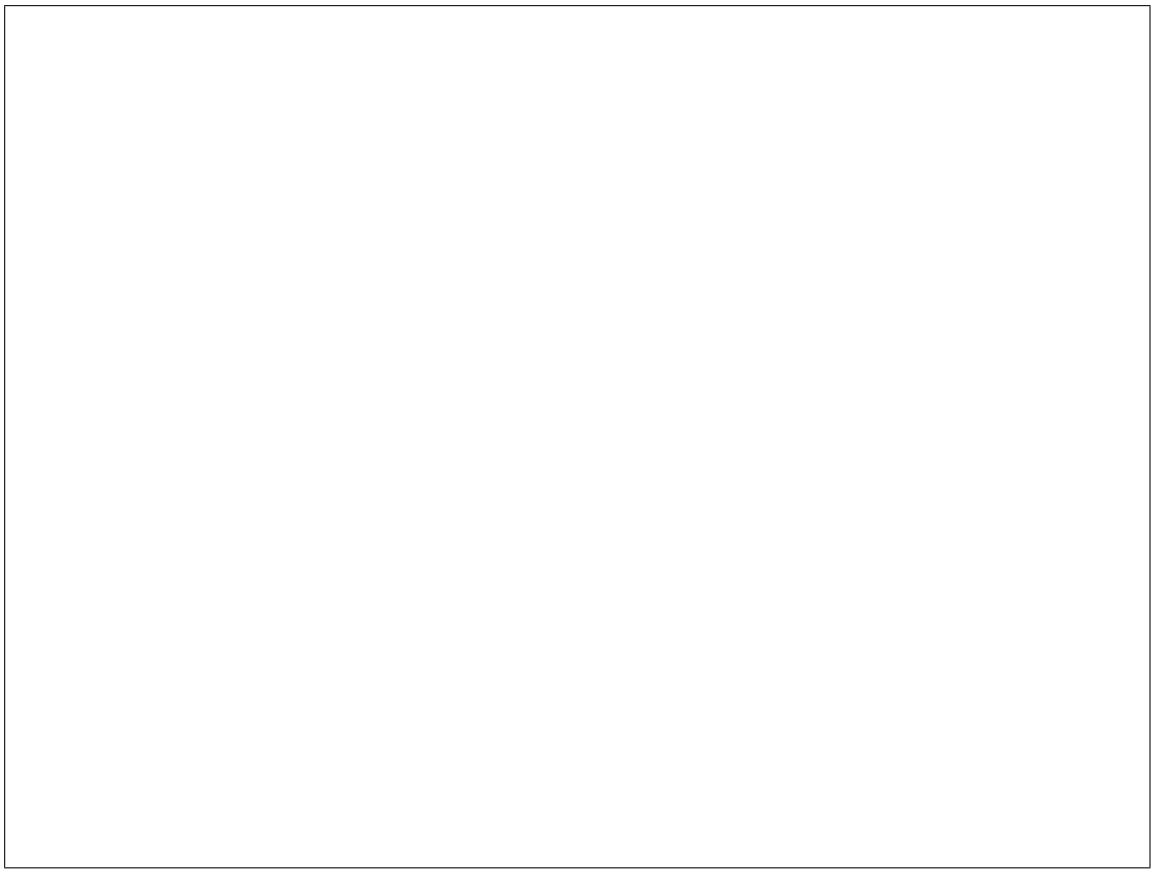
3. Testdaten

i	j	ergebnis
.....	.....	.....
.....	.....	.....
.....	.....	.....
.....	.....	.....

i	j	ergebnis
.....	.....	.....
.....	.....	.....
.....	.....	.....
.....	.....	.....

4. Implementierung (nur für die Bestimmung der Korrelation)

5. Kodierung



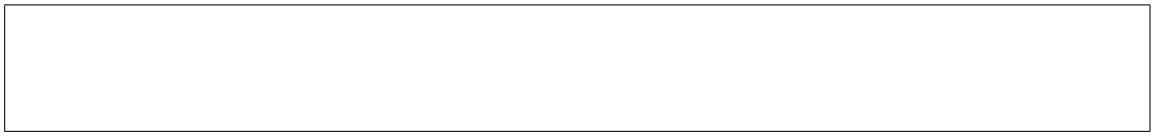
Aufgabe 2: Code-Umwandlung

1. Aufgabenstellung

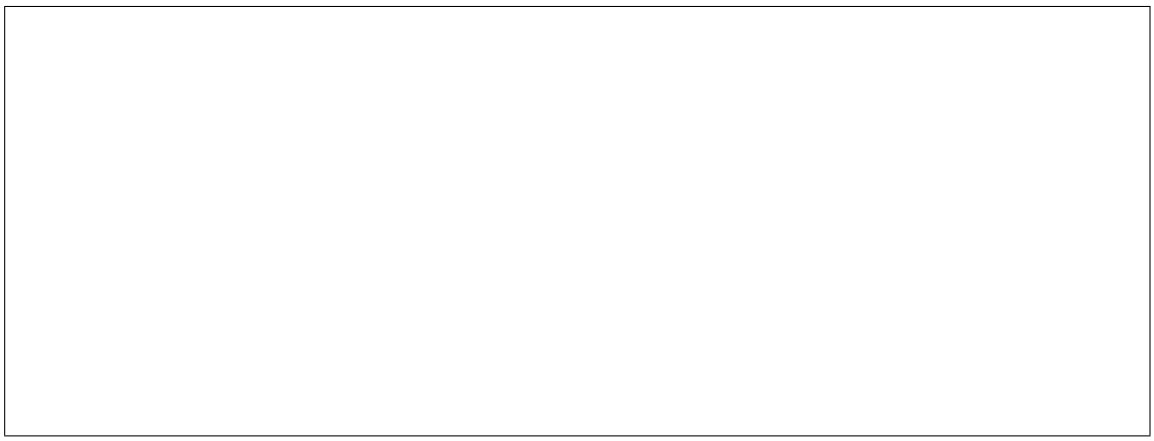
Schreibe ein Programm, das für die Anpassung einer Steuerelektronik (beispielsweise eines Elektroautos) ein paar Konvertierungen vornimmt. Hierzu müssen einige ausgewählte Eingabewerte in neue Werte umgewandelt werden; alle anderen Werte sollen unverändert bleiben. Die Umwandlungstabelle sieht wie folgt aus:

alter Wert	-1	0	4	5	12	13	14	17	34	35
neuer Wert	12	-3	8	4	11	-2	-3	-4	24	1

2. Pflichtenheft: Aufgabe, Eingabe, Ausgabe



3. Implementierung



4. Kodierung