13.20 Strukturen

Struktur: Menge von verschiedenartigen Variablen, die einen inhaltlich zusammengehörigen Datensatz beschreiben.

```
/* Deklaration */
struct Student {
    char      Name[25];
    long int  MatrikelNr;
    char      Studiengang[3];
};
```

```
/* Definition */
struct Student Stud1;

/* Wertzuweisung */
Stud1.MatrikelNr  = 125716;
```

Anm.: Die Struktur-Deklaration ist lediglich eine Beschreibung !
Dabei wird kein Speicherplatz reserviert.

Erst durch die Definition wird für die Struktur Speicherplatz angelegt.

13.20 Strukturen (Fortsetzung)

```
/* Deklaration und Definition
   in einem Schritt                */

struct Student {
    char       Name[25];
    long int   MatrikelNr;
    char       Studiengang[3];
} Std1, Std2, Std3;
```

```
/* Definitionn und Initialisierung in einem Schritt */

struct Student Stud1 = {"Meier", 125716, "TI"};

struct Student Stud2 = {"Gates", 125609, "TI"};
```

ÜBUNG: Struktur deklarieren, definieren und initialisieren

Deklarieren Sie eine Struktur "Datum".

Diese soll 3 Integerwerte für den Tag, Monat und das Jahr enthalten sowie ein Textfeld von 50 Zeichen für Stichworte (Memo).

Erzeugen Sie 2 Termine (Typ "Datum") und initialisieren Sie diese:

Silvesterparty : 31.12.2002, "Party ab 20:00."

Geburtstag : 07.02.2002, "Geburtstag Werner"

13.21 Arrays von Strukturen

```
/* Definition von 60 Datenstrukturen vom Typ Student */
struct Student Std[60];
```

```
/* Initialisierung einer Datenstruktur des Arrays */
strcpy( Std[0].Name , "Meier");
Std[0].MatrikelNr  = 125716;
strcpy( Std[0].Studiengang, „AI");
```

```
/* Definition und Initialisierung von 3 Datenstrukturen
   vom Typ Student                                      */
struct Student Std[3] = {{"Meier",  125716, „AI"},
                         {"Meiser", 125367, „AI"},
                         {"Gabski", 123362, "TI"}};
```

ÜBUNG: Felder von Strukturen definieren und initialisieren

Definieren Sie ein Feld (Name: *Termin*) von 20 Strukturen (Typ "Datum", s.o.) und initialisieren Sie die ersten Strukturen mit

Silvesterparty : 31.12.2002, "Party ab 20:00."
Geburtstag : 07.02.2002, "Geburtstag Werner"

a) Definition und Initialisierung in einem Schritt.
b) Definition und Initialisierung in getrennten Schritten.

13.22 Zeiger und Strukturen

```
/* Definition der Datenstruktur Student */
struct Student Std_1;

/* pStd ist Zeiger auf Datenstruktur Student   */
struct Student *pStd  = &Std_1;

/* Zugriff auf Strukturelemente über den Zeiger */
strcpy( (*pStd).Name, "Gallagher");
(*pStd).MatrikelNr = 125616;

/* ... oder kürzer (und ueblich)...             */
strcpy( pStd->Name, "Townsend");
pStd-> MatrikelNr = 125616;
```

ÜBUNG: Übergabe von Strukturen an Unterprogramme

Es ist ein Programm zu schreiben, welches jeweils nach Ablauf einer Sekunde die Uhrzeit in der Form HH:MM:SS ausgibt.

Die Uhrzeit soll in einer Struktur gespeichert sein:

```
struct tm {
   int hours;
   int minutes;
   int seconds;
};
```

Neben dem Hauptprogramm soll das Programm aus den beiden Funktionen

`update`: Aktualisieren der time-Struktur

`display`: Anzeigen der time-Struktur

bestehen.

13.23 Selbstdefinierte Datentypen

In C lassen sich neue Datentypen definieren. Hierzu dient das Schlüsselwort `typedef`.

```
/* Definition */

typedef unsigned char  BYTE;
typedef unsigned short WORD;
typedef unsigned long  LONGWORD;


/* Anwendung */

BYTE b1, b2=0xF1;
WORD w[10];
```

13.23 Selbstdefinierte Datentypen (Fortsetzung)

```
/* Definition eines neuen Struktur-Typen */

typedef struct {
        char      Name[25];
        long int  MatrikelNr;
        char      Studiengang[3];
} Student;


/* Anlegen von 20 Strukturen des Typs "Student" */
Student Std[20];

strcpy( Std[0].Name, "Winter");
```