<überschrift>

```
//
// Die zu vervollstõndigen Teile finden Sie am Programmende
//
/*****************************************************************************
* AdressVw.c        InformatikII:  Dynamische Adressverwaltung
*
* 29.05.2001
* Holger Kuefner, Copyright (c) Fachhochschule Schweinfurt
*----------------------------------------------------------------------------
* Synopsis:
*  Module implements an address database with simple user modification commands.
*  The database is realized with dynamic structure allocation and sorting.
*
* Known errors:
*
*----------------------------------------------------------------------------
* 29-05-01  User basic routines and database support            Holger Kuefner
* 29-11-02  Anpassung die Vorlesung Informatik II by            H.-J. Meier
*
*****************************************************************************/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
```

```
/*  adresse  */
```

```
/*====== Modul function declarations ======================================*/
static void EingabeElement(TAdresse **sctppKopfListe);
static void LoescheElement(TAdresse **sctppKopfListe); // Zu erstellen!
static void AusgabeElement(TAdresse *sctpKopfListe);
static void AusgabeListe(TAdresse *sctpKopfListe);
static void FreigabeListe(TAdresse *sctppKopfListe); // Zu erstellen!
static void PrintElement(TAdresse *sctpAusgabe);
```

```
/*  main  */
```

```
/*  EingabeElement  */
```

```
/*  AusgabeElement  */
```

```
/*  AusgabeListe  */
```

```
/*  PrintElement  */
```

```
/*  LoescheElement  */
```

```
/*  FreigabeListe  */
```

adresse

```
/*====== Modul type definitions ============================================*/
typedef struct adresse
    {
    char  cFeld32Nchn[32];
    char  cFeld24Vorn[24];
    char  cFeld40Ort[40];
    char  cFeld8Plz[8];
    char  cFeld40Str[40];
    char  cFeld8HsNr[8];
    struct adresse  *sctpNext;
    }
TAdresse;
```

main

```
Funktion /*====== Function definitions ===============================================*/

        /*-------------------------------------------------------------------------
          main:  Hauptschleife und Benutzerschnittstelle der Adressverwaltung. Abhõngig
                 von der Benutzereingabe wird in verschiedene Unterprogramme verzweigt.
                 Bei Beendigung des Programmes wird der allokierte Speicher vollstõndig
                 frei gegeben.
          RETURN:  --
        -------------------------------------------------------------------------*/
        void main(void)
    char      sControl[2];
    TAdresse  *sctpAdressListe=NULL;

    printf("\nADRESSVERWALTUNG");
    do  printf("\nAdresse (e)ingeben, (l)oeschen, (a)usgeben, ");
        printf("(k)omplett ausgeben, kom(p)lett loeschen, (b)eenden: ");
        gets(sControl);
        switch ( sControl[0] )
            case 'e' :
            case 'E' :
                EingabeElement(&sctpAdressListe);
                break;
            case 'l' :
            case 'L' :
                LoescheElement(&sctpAdressListe);
                break;
            case 'a' :
            case 'A' :
                AusgabeElement(sctpAdressListe);
                break;
            case 'k' :
            case 'K' :
                AusgabeListe(sctpAdressListe);
                break;
            case 'p' :
            case 'P' :
                FreigabeListe(sctpAdressListe);
                sctpAdressListe = NULL;
                break;
    while ( (sControl[0]!='b') && (sControl[0]!='B') );
    FreigabeListe(sctpAdressListe);
```

EingabeElement

```
Funktion  /*-------------------------------------------------------------------------
            EingabeElement:  Neue Struktur wird allokiert und mit Werten des Benutzers ge-
                             fuellt. Element wird nach lexiographischer Sortierung des
                             Nachnamens in die Liste eingef³gt.
            RETURN:  --
          -------------------------------------------------------------------------*/
          static void EingabeElement(TAdresse **sctppKopfListe)
```

```
TAdresse  *sctpTmp=NULL, *sctpEinf=NULL;

sctpTmp = (TAdresse *) malloc(sizeof(TAdresse));
```

if (sctpTmp != NULL)

then

```
// Eingabe des Namens und der Adresse:
printf("Eingabe des Nachnamens:   "); gets(sctpTmp->cFeld32Nchn);
printf("Eingabe des Vornamens:    "); gets(sctpTmp->cFeld24Vorn);
printf("Eingabe der Postleitzahl: "); gets(sctpTmp->cFeld8Plz);
printf("Eingabe des Ortes:        "); gets(sctpTmp->cFeld40Ort);
printf("Eingabe der Strasse:      "); gets(sctpTmp->cFeld40Str);
printf("Eingabe der Hausnummer:   "); gets(sctpTmp->cFeld8HsNr);

// Einfuegestelle suchen und Element in die Liste einsortieren:
// Ueberpruefung auf leere Liste und ggf. einf³gen:
```

if (*sctppKopfListe == NULL)

then

```
*sctppKopfListe = sctpTmp;
sctpTmp->sctpNext = NULL;
return  <ausdruck>  ;
```

else

```
// Ein Element mindestens vorhanden. Ueberpruefung ob Element am Kopf eingefuegt werden muss:
```

if (strcmp((*sctppKopfListe)->cFeld32Nchn, sctpTmp->cFeld32Nchn)>0)

then

```
sctpTmp->sctpNext = *sctppKopfListe;  // Einfuegen am Listenkopf
*sctppKopfListe = sctpTmp;
return  <ausdruck>  ;
```

else

```
// Einfuegen des Elementes innerhalb der Liste (mind. ein El. vorhanden):
sctpEinf = *sctppKopfListe;

while ( sctpEinf->sctpNext!=NULL )
```

if (strcmp(sctpEinf->sctpNext->cFeld32Nchn,sctpTmp->cFeld32Nchn)<0)

then

```
sctpEinf = sctpEinf->sctpNext; // Falsches El., zum naechsten El.
```

else

```
// Einfuegestelle zwischen 2 Elementen
sctpTmp->sctpNext = sctpEinf->sctpNext;
sctpEinf->sctpNext = sctpTmp;
return  <ausdruck>  ;
```

```
// Ende der Liste erreicht.
// An das Listenende angehaengen:
sctpTmp->sctpNext = NULL;
sctpEinf->sctpNext = sctpTmp;
sctpTmp = NULL;
```

```
printf("\n FEHLER bei Speicherallokierung !!\n");
```

AusgabeElement

```
Funktion  /*-------------------------------------------------------------------------
            AusgabeElement:  Benutzerabfrage des Namens und Ausgabe der kompletten Adresse
                             mittels des Unterprogrammes "PrintElement".
            RETURN:  --
           -------------------------------------------------------------------------*/
           static void AusgabeElement(TAdresse *sctpKopfListe)
```

```
TAdresse *sctpAusgabe=NULL;
char cFeld32NachnameTemp[32];
printf("Bitte Nachname der auszugebenden Person eingeben:");
gets(cFeld32NachnameTemp);
sctpAusgabe=sctpKopfListe;
```

```
while ( sctpAusgabe != NULL )
    if (strcmp(sctpAusgabe->cFeld32Nchn,cFeld32NachnameTemp)!=0)
    then                                                    else
        sctpAusgabe=sctpAusgabe->sctpNext;                  PrintElement(sctpAusgabe);
                                                            break;
```

C:\!hinher\AdressVw.cpp §5

AusgabeListe

```
Funktion  /*-------------------------------------------------------------------------
            AusgabeListe:  Alle Adressen werden entspr. der Sortierung der Liste nach-
                           einander ausgegeben.
            RETURN:  --
           -------------------------------------------------------------------------*/
           static void AusgabeListe(TAdresse *sctpKopfListe)
```

```
TAdresse *sctpAusgabe=NULL;
sctpAusgabe=sctpKopfListe;
```

```
while ( sctpAusgabe != NULL )
    PrintElement(sctpAusgabe);
    sctpAusgabe=sctpAusgabe->sctpNext;
```

PrintElement

```
Funktion  /*-------------------------------------------------------------------------
            PrintElement: Die angewaehlten Daten werden ausgegeben.
            RETURN:  --
            -------------------------------------------------------------------------*/
            static void PrintElement(TAdresse *sctpAusgabe)
  printf("Eingabe des Nachnamens:  %s\n", sctpAusgabe->cFeld32Nchn);
  printf("Eingabe des Vornamens:   %s\n", sctpAusgabe->cFeld24Vorn);
  printf("Eingabe der Postleitzahl:%s\n", sctpAusgabe->cFeld40Ort);
  printf("Eingabe des Ortes:       %s\n", sctpAusgabe->cFeld8Plz);
  printf("Eingabe der Strasse:     %s\n", sctpAusgabe->cFeld40Str);
  printf("Eingabe der Hausnummer:  %s\n", sctpAusgabe->cFeld8HsNr);
  printf("----------------------------------------\n");
```

C:\!hinher\AdressVw.cpp §7

LoescheElement

```
Funktion  /*-------------------------------------------------------------------------
            LoescheElement:  Element wird aus der Liste entfernt. Speicher wird freige-
                             geben.
                             Beachte: der Zeiger "sctpDel" zeigt auf das vorauslaufende
                                      Element zu dem zu loeschenden Element!
            RETURN:  --
           -------------------------------------------------------------------------*/
           static void LoescheElement(TAdresse **sctppKopfListe)
  TAdresse  *sctpTmp=NULL, *sctpDel=NULL;
  char      cFeld32NnTmp[32];

  printf("Eingabe des Nachnamens der Adresse die gel÷scht werden soll: ");
  gets(cFeld32NnTmp);
  // In der _bung zu erstellen
  // Zunõchst das zu l÷schende Element suchen (Vorgehen wie in der Funktion
  // "AusgabeElement".
  // Dann Element l÷schen (Fallunterscheidung gemõ¯ Vorlesung!)
```

C:\!hinher\AdressVw.cpp §8

FreigabeListe

```
Funktion  /*-------------------------------------------------------------------------
            FreigabeListe:  Alle allokierten Elemente der Liste werden freigegeben.
            RETURN:  --
           -------------------------------------------------------------------------*/
           static void FreigabeListe(TAdresse *sctppKopfListe)
  // In der _bung zu erstellen
```