Übungen zur Vorlesung

„Informatik für Ingenieure 2“ bzw. „Informatik 2“

für die Studiengänge AEB2 / IKB2 bzw. AED3

Allgemeine Hinweise:

Für alle Aufgaben muss am Beginn des Quelltextes der Programmkopf aus der Datei „Der für die Programme zu nutzende Programm-Kopf“ verwendet werden.

Hier sind Name, Vorname, Matr-Nr., Datum, Dateiname etc. einzutragen!

```
//------------------------------------------------------------------------
//
// Datei:        programmname.cpp
//
// Name:
// Matr.-Nr.:    123456
// Datum:        xx.xx.2009 V1.0
// geaendert:    xx.xx.2009
//
// Beschreibung: hier muss eine Kurzbeschreibung der Funktionalitaet stehen
//               z.B. Aufgabenstellung
//
// Eingabe:      Angabe der erforderlichen Eingaben (Benutzer / Datei)
//
// Ausgabe:      Angabe der vom Programm generierten Ausgaben
//
//------------------------------------------------------------------------
#include <stdio.h>           // Bibliothek fuer printf, scanf, ...
#include <conio.h>           // Bibliothek fuer getch, gets, ...
```

Bei Aufgaben mit mehreren Funktionen soll die main-Funktion vor den anderen Funktionen stehen.

Für die Abnahme gelten folgende Regeln:

- Das jeweilige Programm muss ohne Fehler und Warnungen übersetzt werden.
- Es müssen alle Aspekte der Aufgabenstellung erfüllt sein.
- Die Verwendung des Befehls „goto" ist verboten.
- Ausgaben am Bildschirm müssen eindeutig sein, sodass ein leichtes Bedienen des Programms möglich ist.
- Gängige Eingabefehler müssen abgefangen werden und dürfen nicht zu Fehlfunktion oder gar Absturz des Programms führen.
- Sie müssen das Programm im Detail erläutern können.
- Zu jedem Programm gehört ein Struktogramm bzw. Programm-Ablaufplan. (Pseudocode oder frei formuliert, kein Code)

UE01: Mein erstes C-Programm

Schreiben Sie ein Programm, das den Text "Hallo, Welt!" auf dem Bildschirm ausgibt! Erweitern Sie das Programm, so dass Sie eine ganze Zahl (Typ int) eingeben können, die in folgenden Formaten ausgegeben wird:

a) linksbündig

b) als Gleitkommazahl mit 2 Nachkommastellen

c) rechtsbündig mit einer Feldweite von 5 Stellen.

Das Programm soll beendet werden, wenn die Zahl 0 eingegeben wird.

Beispielausgabe auf dem Bildschirm:

```
Bitte geben Sie eine Zahl ein: 66
Die eingegebene Zahl ist: 66
Die eingegebene Zahl ist: 66.00
Die eingegebene Zahl ist:    66
Bitte geben Sie eine Zahl ein: 0
```

Hinweis:

Möglicherweise erhalten Sie vom Compiler die Warnung, dass eine lokale Variable nicht initialisiert sei. Sorgen Sie selbst für einen Startwert, z. B. `int z = 0.`

Die Formatangaben in `printf` und `scanf` müssen immer zu den Zahlenformaten passen. Wenn Sie eine Integerzahl `z` als Floatzahl ausgeben wollen, müssen Sie den Typ umwandeln, z. B. mit `(float) z.`

Achten Sie darauf, dass Sie in `scanf` die Adresse der einzulesenden Variable angeben müssen. Das geschieht mit dem Adressoperator &, z.B. `scanf("%d",&z).`

UE02: Temperatur-Umrechnung

Mit der Formel

```
tc = 5/9 * (tf - 32)
```

kann die Temperatur von Fahrenheit tf in Celsius tc umgerechnet werden. Schreiben Sie ein Programm, das eine Temperaturtabelle für beide Einheiten auf dem Bildschirm ausgibt!

a) Geben Sie die Tabellenwerte als ganze Zahlen aus.

b) Die Tabellenwerte sollen als Gleitkommazahlen mit einer Nachkommastelle ausgegeben werden.

Beispielausgabe auf dem Bildschirm für Teilaufgabe a):

```
Fahrenheit       Celsius
     0             -17
    20              -6
    40               4
   ...             ...
   200             ...
```

Hinweis:

Die Rechnung sollte mit `Double`-Zahlen erfolgen. Für die ganzzahlige Ausgabe werden die Zahlen dann entsprechend formatiert.

UE03: Ausgabe einer Zahlenpyramide

Folgende Zahlenpyramide soll auf dem Bildschirm ausgegeben werden:

```
         0
        101
       21012
      3210123
     432101234
    54321012345
   6543210123456
  765432101234567
 87654321012345678
9876543210123456789
```

Entwickeln Sie einen Algorithmus, der mit Hilfe mehrerer verschachtelter Schleifen die Zeichen auf dem Bildschirm ausgibt.

Hinweis:

1. Im ersten Schritt sollte nur jeweils die erste Ziffer jeder Zeile untereinander ausgegeben werden.
2. Dann sollte diese eine Ziffer in jeder Zeile mit Hilfe von Leerzeichen so eingerückt werden, dass die Kontur der linken Pyramidenseite abgebildet wird.
3. Jetzt werden in jeder Zeile die abwärts und wieder aufwärts zählenden Ziffern hinzugefügt.

UE04: Zahlenumwandlung Hex/Dezimal

Ein Programm soll in einer Schleife den Dezimalwert einer eingegebenen Hex-Ziffer berechnen und ausgeben. Falls das eingegebene Zeichen nicht zu einer Hex-Zahl gehört, erfolgt eine Fehlermeldung. Der Eingabewert @ (ASCII 64) soll das Programm beenden.

Beispielausgabe auf dem Bildschirm:

```
Bitte geben Sie eine Hex-Ziffer ein: F
Die Ziffer hat den Wert 15.
Bitte geben Sie eine Hex-Ziffer ein: 3
Die Ziffer hat den Wert 3.
Bitte geben Sie eine Hex-Ziffer ein: z
Die eingegebene Ziffer ist kein Hex-Wert!
Bitte geben Sie eine Hex-Ziffer ein: ...
```

Hinweis:

Die Eingabe erfolgt durch das Einlesen einzelner ASCII-Zeichen, zweckmäßig mit `getch()`.

ASCII-Werte: 'A' = 65; 'a' = 97; '0' = 48

Es sollen keine Funktionen aus `<string.h>` oder `<ctype.h>` wie z. B. `isdigit()` oder `isanum()` verwendet werden!

UE05: Eingabeprüfung von Zeichenfolgen

Eine Zeichenfolge soll eingelesen und in einem String der Länge 10 gespeichert werden. Ggf. muss der eingegebene String gekürzt werden. Der Rest soll wiederum in einem separaten String gespeichert und ausgegeben werden.

Die gesamte Eingabe soll überprüft und folgende Werte ausgegeben werden:

a) der gespeicherte String
b) die Länge der eingegebenen Zeichenfolge
c) ggf. der abgeschnittene Teil des Strings
d) die Anzahl der Ziffern und der Großbuchstaben im eingegebenen String.

Die Prüffunktion soll mehrfach hintereinander ausführbar sein.

Beispielausgabe auf dem Bildschirm:

```
Bitte geben Sie eine Zeichenfolge ein: AbcDefGhi099
Der gespeicherte String ist AbcDefGhi0.
Die eingegebene Folge enthält 12 Zeichen.
Folgende Zeichen wurden abgeschnitten: 99
Die Folge enthält 3 Ziffern und 3 Grossbuchstaben.

Bitte geben Sie eine Zeichenfolge ein: ...
```

Hinweis:

Die Eingabe soll nicht zeichenweise sondern als String eingelesen werden. Verwenden Sie `scanf("%s", ...)` oder `gets()`! Testen Sie das Programm auch mit einer Zeichenfolge, die kürzer als 10 Zeichen ist!

Funktionen aus `<string.h>` wie `strlen(), strcpy(), strstr()` etc. dürfen nicht verwendet werden.

UE06: Berechnung der Sinusfunktion mit einer Reihe

Die Sinusfunktion kann durch eine unendliche Reihe dargestellt werden.

$\sin x = x - x^3/3! + x^5/5! - x^7/7! + ...$

Beachten Sie das alternierende Vorzeichen!

Diese Formel gilt für den Winkel x im Bogenmaß.

Schreiben Sie ein Programm, das zur Eingabe eines Winkels in Grad auffordert und den Sinus dieses Winkels berechnet!

Die Berechnung der Reihe kann abgebrochen werden, wenn der Wert des nächsten Ausdrucks kleiner als 10^{-5} wird. Zur Kontrolle soll der Sinus daneben mit der Bibliotheksfunktion `sin(x)` aus `math.h` berechnet und ausgegeben werden.

Beispiel:

```
Bitte geben Sie einen Winkel in Grad ein: 60

sin(x) = 0.866  (errechneter Wert)
sin(x) = 0.866  (Vergleichswert mit Bibliotheks-Funktion)
```

Hinweis:

Die Berechnung soll so erfolgen, dass der jeweils nächste Term der Summe stets aus dem vorangehenden berechnet wird.

```
i = i+2;

sig = -sig;

summand_iplus1 = summand_i * x*x / i / (i-1);

sin_x = sin_x + sig * summand_iplus1;

...
```

Testen Sie das Programm auch mit negativen Winkelwerten!

UE07: Eingabe einer Doublezahl

Ein Programm soll prüfen, ob die eingegebene Zeichenfolge eine korrekte Gleitkomma- oder Integerzahl ist. Ausgegeben wird der Zahlenwert oder eine Fehlermeldung.

Die Prüfung erfolgt in einer Funktion `int isDouble(char* s)`.

Die Zeichenfolge s darf nur Ziffern, einen Punkt und einmal + oder – als erstes Zeichen enthalten. Dann wird der Wert 1 zurückgegeben, anderenfalls 0.

Falls die Prüfung der Eingabe mit `isDouble` keine Fehler ergibt, wird der Dezimalwert der Zeichenfolge s mit der Funktion `double atof(char* s)` aus `<stdlib.h>` berechnet.

Schreiben Sie die Funktion `isDouble` und ein Hauptprogramm, das mit einer Schleife immer wieder zur Eingabe auffordert. Nach Prüfung der Eingabe wird der Zahlenwert oder eine Fehlermeldung ausgegeben.

Beispielausgabe auf dem Bildschirm:

```
Bitte geben Sie eine Gleitkommazahl ein: -13
-13.0 wurde eingegeben.
Bitte geben Sie eine Gleitkommazahl ein: +45.3.
Eingabefehler!
Bitte geben Sie eine Gleitkommazahl ein: 4+
Eingabefehler!
Bitte geben Sie eine Gleitkommazahl ein: ...
```

UE08: Text eingeben und speichern

Schreiben Sie ein Programm, das einen Text zeichenweise von der Tastatur liest und diese ebenfalls zeichenweise in einer Datei speichert, deren Inhalt auf Wunsch angezeigt werden kann.

- Der Benutzer soll eine Datei zum Speichern angeben können.
- Falls die Datei bereits vorhanden ist, soll der Benutzer gefragt werden, ob er den Text anhängen oder die Datei überschreiben will.
- Der Text soll mehrere Zeilen enthalten können.
- Die Textausgabe soll seitenweise erfolgen, indem die ausgegebenen Zeilen gezählt und bei vollem Bildschirm die Aufforderung „Nächste Seite mit <Enter>" ausgegeben wird.

Hinweis:

Verwenden Sie für das Schreiben und Lesen der Zeichen in eine binäre Datei die Funktionen `fputc` und `fgetc`.

Beispielausgabe auf dem Bildschirm:

```
In welcher Datei soll der Text gespeichert werden?
c:\my_dir\my_file.txt
Die Datei existiert bereits. Soll sie überschrieben werden? (j/n): n
Der Text wird angehängt.
Geben Sie jetzt den Text ein (Ende mit @ und Return): ...
Datei anzeigen? (j/n): ...
```

Hinweis:

Die Texteingabe soll mit dem Zeichen @ (ASCII 64) beendet werden.

UE09: Telefonliste (statisch)

Schreiben Sie ein Programm, das eine Telefonliste verwaltet. Für jeden Teilnehmer soll der Name, der Vorname und die Telefonnummer eingetragen werden. Die Datensätze werden in einem Feld gespeichert.

Mit Hilfe eines Menüs sollen folgende **Funktionen** gewählt werden können:

a) **Eingabe**
falls noch Platz in der Liste ist, werden Name, Vorname, Telefonnummer eingegeben.

b) **Ausgabe**
auf dem Bildschirm wird eine Tabelle der Daten ausgegeben

```
Nr.   Name           Vorname       Tel-Nr.
```

c) **loeschen**
Die Nummer des zu löschenden Datensatzes wird abgefragt. Die übrigen Datensätze müssen so verschoben werden, dass im Feld keine Lücke entsteht.

Beispielausgabe auf dem Bildschirm:

```
Bitte wählen Sie:

1 - neuen Teilnehmer eingeben
2 - Telefonliste ausgeben
3 - Teilnehmer löschen
4 - Programm beenden

Auswahl:
```

Hinweis:

```
struct eintrag {
     char name[40];
     char vorname[20];
     char nummer[20];
}
struct eintrag telefonliste[20];
```

UE10: Verkettete Liste

Schreiben Sie ein Programm, das eine Wörterliste verwaltet. Die Wörter, die z. B. Namen, Orte oder Gegenstände bezeichnen, werden in einer verketteten Liste gespeichert. Mit Hilfe eines Menüs sollen folgende Funktionen gewählt werden können:

a) Liste erstellen
der Speicherplatz für einen neues Wort wird erstellt und das Wort eingegeben. Die Eingabe wird so lange wiederholt, bis "ENDE" eingegeben wird.

b) Element hinzufügen
der Speicherplatz für ein neues Wort wird erstellt und das Wort eingegeben. Es wird gefragt, vor welchem Element das neue Wort eingefügt werden soll.

c) Element löschen
Das zu löschende Wort wird abgefragt. Es wird gelöscht und sein Speicherplatz wieder freigegeben.

Nach jeder Bearbeitung wird die aktuelle Wortliste vor dem Hauptmenue wieder auf dem Bildschirm angezeigt.

Beispielausgabe auf dem Bildschirm:

```
HAUPTMENUE:
1 - Liste erstellen
2 - Element hinzufuegen
3 - Element loeschen
4 - Programm beenden
Bitte waehlen Sie:
```

Hinweis:

```
struct eintrag { // Deklaration eines Listenelements
     char wort[20];
     struct eintrag *next;
};
typedef struct eintrag element;
struct eintrag *liste;   //Zeiger auf den Anfang der Liste
```

Die Funktion `erstellen` wird zunächst mit dem Zeiger auf den Listenanfang aufgerufen, dann rekursiv jeweils mit dem Zeiger auf das nächste Element.

```
void erstellen (element* ptr){
     printf ("Wort eingeben ('ENDE' fuer Ende der Liste):
");
     scanf("%s", ptr->wort);
     if (strcmp(ptr->wort,"ENDE")==0) ptr->next=NULL;
     else {
  ptr->next = (element*) malloc(sizeof(element));
  erstellen (ptr->next);
     }
     return;
}
```

Das letzte Element der Liste enthält so das Wort `ende` und den Zeiger `next = NULL`. Die Funktion `anzeigen` wird auch mit dem Zeiger auf das erste Element aufgerufen, dann rekursiv mit dem `next`-Zeiger auf das jeweils nächste Element.

```
void anzeigen (element* ptr){
     if (ptr->next != NULL) {
  printf ("%s\n",ptr->wort);
  anzeigen (ptr->next);
     }
     return;
}
```

Beim Erstellen einer neuen Liste muss im Hauptprogramm zunächst der Speicherplatz für das Anfangselement mit der Adresse `liste` reserviert werden. Die Speicherreservierung für die folgenden Elemente erfolgt in der Funktion `erstellen`.

```
liste = (element*)malloc(sizeof(element));
erstellen(liste);
...
```