Friedrich-Alexander-Universität Erlangen-Nürnberg

Schriftliche Prüfung zur Vorlesung
“Einführung in die Programmierung für Ingenieure (C)”

Aufgabe 1	Aufgabe 2	Aufgabe 3	Aufgabe 4	Aufgabe 5	Gesamt
31	**8**	**10**	**6**	**5**	**60**

(Name) (Vorname) (Matrikel-Nr.)

(Fachrichtung) (Semester)

Erlangen, 16.02.2001

Organisatorische Hinweise

Die folgenden Informationen bitte aufmerksam lesen und die Erklärung am Blattende unterschreiben.

- Die Bearbeitungszeit beträgt 60 Minuten.
- Es sind **keine** Hilfsmittel zugelassen.
- Schmierpapier darf **nicht** mit abgegeben werden. Die Lösung einer Aufgabe soll auf das Aufgabenblatt in den dafür vorgesehenen freien Raum geschrieben werden, der jeweils der Aufgabe folgt. Sollte der Platz nicht ausreichen, so können Sie die Rückseiten der Aufgabenblätter mitverwenden.
- Bei Bedarf sind von der Aufsicht Schmierpapier (farbig) und zusätzliche Lösungsblätter (weiß) zu erhalten. Vermerken Sie vor dessen Verwendung unbedingt Ihren Namen und Ihre Matrikelnummer darauf! **Schmierpapier darf nicht mit abgegeben werden!**
- Fragen zu den Prüfungsaufgaben können grundsätzlich nicht beantwortet werden
- Tragen Sie Ihren Namen und Vornamen, Ihre Matrikelnummer, Studienfachrichtung und Fachsemesterzahl auf dem Deckblatt der Klausur ein.
- Bitte legen Sie Ihren Studenten- und einen Lichtbildausweis zur Kontrolle bereit.
- Sie dürfen den Raum nicht verlassen, bevor Ihre Personalien überprüft wurden und Sie die Klausurunterlagen der Aufsicht zurückgegeben haben.
- Bleiben Sie bitte am Ende der Bearbeitungszeit solange an Ihrem Platz sitzen, bis alle Klausurunterlagen eingesammelt sind.

Die Klausurergebnisse werden bis Ende der ersten Augustwoche am schwarzen Brett vor dem Sekretariat des Lehrstuhls für Betriebssysteme (vor Raum 0.047 RRZE Erdgeschoss) und im WWW unter folgender URL veröffentlicht: http://www4.informatik.uni-erlangen.de/result00c.html

Erklärung

Durch meine Unterschrift bestätige ich

- den Empfang der vollständigen Klausurunterlagen (12 Seiten inkl. Deckblatt).
- die Kenntnisnahme der obigen Informationen.

Ich bin damit einverstanden, daß mein Prüfungsergebnis unter Angabe der Matrikel-Nr. veröffentlicht wird.

☐ ja ☐ nein

Der Schein soll direkt an das Prüfungsamt geschickt werden:

☐ ja ☐ nein

Erlangen, 16.02.2001

(Unterschrift)

Aufgabe 1: (31 Punkte)

Schreiben Sie ein Programm **telefon** zum Erstellen eines Telefonbuches, bestehend aus einer beliebigen Anzahl von Einträgen.

Für jeden Eintrag sollen folgende Daten erfasst werden

- Nachname (max. 24 Buchstaben lang)
- Vorname (max. 24 Buchstaben lang)
- Telefonnummer (als Zahl).

Das Programm wird folgendermassen aufgerufen

telefon -o *Ausgabedatei*

und arbeitet dann wie folgt beschrieben:

- Aufrufparameter auswerten, bei falschem Aufruf Fehlermeldung (-o steht für "output" und muß immer mit angegeben werden)
- Jetzt werden vom Benutzer Einträge entgegengenommen. Dies wird von der Funktion **`erfasse_eintraege`** erledigt, die folgendes tut:
  - mit Hilfe der Funktion **`neuer_eintrag`** wird eine Datenstruktur zur Speicherung der Telefondaten erzeugt
  - diese Struktur wird mit Hilfe der Funktion **`erfasse_daten`** gefüllt
  - anschließend wird diese Struktur vorne in eine verkettete Liste solcher Strukturen eingehängt (das wird nicht durch `erfasse_daten` erledigt, sondern muss in der Funktion `erfasse_eintraege` erfolgen!)
  - dies wird wiederholt, bis **`erfasse_daten`** das Ende der Eingabe meldet.
- Nun werden alle Einträge nach Nachnamen sortiert (mit der Funktion **`sortiere_eintraege`**)
- Schliesslich werden alle in der Liste gespeicherten Telefondaten mit Hilfe der Funktion **`speichere_eintraege`** in die angegebene Ausgabedatei gespeichert.

Auf den folgenden Seiten finden Sie ein Gerüst für das beschriebene Programm. In den Kommentaren sind die Aufgaben der einzelnen, zu ergänzenden Programmteile beschrieben.


```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

/*
 * Deklaration der Struktur für Einträge ins Telefonverzeichnis
 */
struct eintrag {

      ______________________________________________________

      ______________________________________________________

      ______________________________________________________

      struct eintrag *naechster;
};

/*
 * Deklaration von Funktionen, die bei der Programmierung von
 * main benötigt werden und die im Rahmen dieser Aufgabe
 * (zusätzlich zu main) zu programmieren sind (ab Seite 6)
 */
struct eintrag *neuer_eintrag();
struct eintrag *erfasse_eintraege();
void speichere_eintraege(struct eintrag *, char *);


/*
 * Deklaration von Funktionen, die bei der Programmierung von
 * main benötigt werden, die aber nicht selbst programmiert
 * werden müssen
 */

/* erfasse_daten: liefert 0 am Ende der Eingabe, sonst 1 */
int erfasse_daten(struct eintrag *);

/* sortiere_eintraege: Liefert einen Zeiger auf das erste */
/* Element der sortierten Liste */
struct eintrag *sortiere_eintraege(struct eintrag *);
```

```
/*
 * Funktion main
 */
int main(int argc, char *argv[]) {
      /* Variablendefinitionen */

      /*
       * Überprüfen der Kommandozeile;
       * Hinweis: int strcmp(char *, char *) liefert 0, wenn die
       * übergebenen Zeichenketten gleich sind, ungleich 0 sonst
       */

      /*
       * Einträge vom Benutzer entgegennehmen
       */

      /*
       * Einträge nach Nachnamen sortieren
       */

      /*
       * Liste in Ausgabedatei speichern
       */

      return 0;
}
```



```
/*
 * Funktion erfasse_eintraege zum wiederholten Einlesen von
 * Telefonbucheinträgen. Gibt Zeiger auf den Anfang der hier
 * erzeugeten verketteten Liste der Einträge zurück.
 */
struct eintrag *erfasse_eintraege() {

      /*
       * Struktur für Einträge erzeugen, Daten erfassen,
       * vorne in Liste einketten, ...
       */

}
```

```
/*
 * Funktion neuer_eintrag zum Erzeugen eines neuen Telefonbuch-
 * eintrags (wenn beim Reservieren des Speichers ein Fehler
 * auftritt, gibt diese Funktion eine Fehlermeldung aus und
 * beendet das Programm).
 * Die neue Struktur soll nicht initialisiert werden!
 */
struct eintrag *neuer_eintrag() {

}
```



```
/*
 * Funktion speichere_eintraege
 * öffnet zuerst die übergebene Datei
 * - wenn hierbei ein Fehler auftritt, wird eine entsprechende
 *   Meldung ausgegeben und das Programm abgebrochen
 * anschliessend wird für jedes Listenelement eine Zeile in der Form
 * "Nachname Vorname Telefonnummer"
 * in die Datei geschrieben
 */
void speichere_eintraege(struct eintrag *telbuch, char *datei)
{
      FILE *fp;

      /* Ausgabedatei öffnen */

      /* Telefonbuch (Liste der Einträge) ausgeben */

}
```

Aufgabe 2: (8 Punkte)

Markieren und verbessern Sie die Fehler in dem untenstehenden Programm, das den Benutzer nach einer Laufzeit, einem Zinssatz sowie einer monatlichen Sparrate fragt und daraus Zinsen und Zinseszinsen berechnet. (Sie können davon ausgehen, daß max. 8 Fehler im Programm versteckt sind.)

Falsch erkannte Fehler werden negativ gewertet!

```
#include <stdio.h>            /* Default: Zinsgutschrift am
#define DEFAULT_WERTSTELLUNG 3; Quartalsende (alle 3 Monate) */

int parseargs(int argc, char *argv[]);

int main(int argc, char *argv[]) {
  int m, monate, wertstellung;
  double zins, zinssatz, betrag;
  wertstellung = parseargs(argc, *argv);

  printf("Laufzeit (Monate): ");
  scanf("%d", monate);
  printf("Zinssatz (in %):   ");
  canf("%lf", &zinssatz);
  printf("Betrag:            ");
  scanf("%lf", &betrag);

  zins = 0.0;
  for(m=0; m<monate; m++); {
    zins += betrag * (zinssatz / 100) / 12;
    if(m % wertstellung == wertstellung-1) {
      betrag += zins;
      zins = 0.0;
    }
  }
  printf("Betrag: %lf\n", &betrag);
  return 0;
}

int parseargs(int argc, char *argv[]) {
  int wertstellung = DEFAULT_WERTSTELLUNG;
  if(argc = 2) {
    switch(argv[1][1]) {
      case ’m’: wertstellung = 1; break;
      case ’j’: wertstellung = 12; break;
    }
  }

}
```


Aufgabe 3: (10 Punkte)

Was gibt das folgende Programm **`demo`** in Zeile 12 aus, wenn es wie folgt aufgerufen wird:

`demo out of memory`

Geben Sie die Inhalte der verwendeten Variablen nach jeder ausgeführten Anweisung an. Verwenden Sie dazu die untenstehende Tabelle!

```
 1  #include <stdio.h>
 2  #define LEN 5
 3
 4  int main(int argc, char *argv[])
 5  {
 6    int i;
 7    char *zk, erg[LEN];
 8    for (i=0; argv[i] != NULL ; i++) {
 9      zk = argv[i]; erg[i] = zk[0];
10    }
11    erg[i] = ’\0’;
12    printf("%s\n", erg);
13    return 0;
14  }
```

Zeile	i	zk	erg erg[0] erg[1] erg[2] erg[3] erg[4]

Aufgabe 4: (6 Punkte)

Sie benötigen für eine Berechnung die Fibonacci-Zahlen. Ein kurzer Blick in ein 24-bändiges Standardlexikon liefert folgende (hier leicht abgeänderte) Definition:

> "Fibonacci-Folge, nach L. FIBONACCI die Folge der Zahlen
> 1, 1, 2, 3, 5, 8, 13, ... , wobei jedes Glied gleich der Summe der beiden vorangehenden Glieder ist; allgemein: $a_n = a_{n-2} + a_{n-1}$ mit $a_1 = a_2 = 1$.

Schreiben Sie eine **rekursive** Funktion namens `fib`, die bei gegebenem n die entsprechende Fibonacci-Zahl berechnet (Fehlerabfragen sind nicht nötig).

Aufgabe 5: (5 Punkte)

a) Betrachten Sie das folgende Programmstück?

```
float f = 10.0;
f = f / 100.0;
f = f * 10.0;
```

Welchen Wert erwartet der "unerfahrene" Programmierer in der Variablen `f` nach Ausführung der letzten Zeile?

Womit dagegen rechnen Sie, und warum ist das so?

b) Was ist der Unterschied zwischen einer Funktionsdefinition und einer Funktionsdeklaration (Prototyp)? Warum werden Funktionsdeklarationen benötigt?