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
40	**10**	**10**	**6**	**14**	**80**

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

(Fachrichtung) (Semester)

Erlangen, 27.01.1999

Organisatorische Hinweise

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

- Die Bearbeitungszeit beträgt 120 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 am Ende (=Freitag!) der ersten Februarwoche am Abgabekasten ausgehängt!

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, 27.01.1999

(Unterschrift)

Aufgabe 1: (40 Punkte)

Schreiben Sie ein Programm **bsort** das mit Hilfe des "Bubble-Sort"-Algorithmus ein Array (= Feld) von ganzen Zahlen aufsteigend sortiert. Beim Aufruf soll man dem Programm auf der Kommandozeile mitteilen können, wieviele Zahlen einzulesen sind und ob die Summe aller Zahlen (Option **-s**) und/oder deren Mittelwert (Option **-m**) ausgegeben werden sollen. Programmaufrufe könnten z.B. wie folgt aussehen:

bsort -s 15 **bsort 12 -m -s** **bsort -m -s**

Der erste Aufruf würde bedeuten, daß der Benutzer 15 Zahlen eingeben soll, die anschließend sortiert werden und deren Summe bei der Ausgabe angezeigt wird.

Ihnen sind die untenstehende **main**-Funktion sowie die Deklarationen der benötigten Funktionen vorgegeben. Sie sollen nun die deklarierten Funktionen implementieren, sowie zu den Funktionen **init_array** und **sort_array** jeweils ein **Struktogramm** erstellen. (**Achtung**: Verwenden Sie keine globalen Variablen!)

Beschreibung der einzelnen Funktionen:

a) **parse_ops**:
Bearbeitet die Kommandozeilenoptionen. Für jede vorhandene Option (**-s** und/oder **-m**) soll die jeweils entsprechende Variable (**flag_s** , **flag_m**) den Wert 1 erhalten, andernfalls 0. Wenn eine Zahl vorhanden ist wird **num** auf den entsprechenden Wert gesetzt, ansonsten wird 10 eingestellt. Bei Fehlaufrufen (z.B. wenn die Zahl<=0 ist, etc.) soll das Programm eine Fehlermeldung ausgeben und terminieren. Zur Umwandlung einer Ziffernfolge in einen Integer-Wert kann die Funktion **int atoi(char *str)** verwendet werden (wobei **str** einen Zeiger auf den zu konvertierenden String darstellt). Hinweis: Falls **str** keine Zahl repräsentiert liefert **atoi** den Wert 0.

b) **init_array**:
Hier muß Speicher für das Array beschafft werden. (Falls hierbei etwas schiefgeht, muß das Programm natürlich entsprechend reagieren.) Anschließend sollen vom Benutzer über die Standardeingabe die Werte in das Array eingegeben werden. Die Funktion liefert als Ergebnis einen Zeiger auf das Array. (Zuerst Struktogramm erstellen!)

c) **sort_array**:
Die Werte im Array werden sortiert, indem das Array wiederholt von vorne durchlaufen wird und je zwei benachbarte Werte ihren Platz tauschen, falls der erste größer als der zweite ist. So steigt in einem Durchlauf das größte Element wie eine "Blase im Wasser" ans Ende des Arrays. Das wird wiederholt, bis in einem Durchlauf keine Elemente mehr getauscht werden mußten. (Struktogramm nicht vergessen!)

d) **print_array**:
Funktion zum Ausgeben aller Zahlen des Feldes und ggf. deren Summe (Ganzzahl) und Mittelwert (Fließkommazahl), falls die entsprechenden Optionen angegeben waren (falls eine Option nicht gesetzt ist, soll natürlich auch die entsprechende Ausgabe unterbleiben, die Berechnung des Wertes kann aber dennoch erfolgen). Hinweis: Es bietet sich an, die Summe während der Ausgabe der Arrayelemente zu berechnen.

e) **swap**:
Funktion zum Tauschen zweier Integerwerte.

f) **main**:
Ergänzen Sie die **main**-Funktion so, daß der dynamisch belegte Speicher wieder freigegeben wird.

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

void parse_ops(int argc, char *argv[], int *num, int *flag_s, int *flag_m);
int *init_array(int num);
void sort_array(int *arr, int num);
void print_array(int *arr, int num, int flag_s, int flag_m);
void swap(int *a, int *b);

int main(int argc, char *argv[])
{
   int num;
   int *arr;
   int flag_s, flag_m;

   parse_ops(argc, argv, &num, &flag_s, &flag_m);

   arr = init_array(num);
   sort_array(arr, num);
   print_array(arr, num, flag_s, flag_m);

   /* Speicher freigeben (Teilaufgabe f): */


   return 0;
}
```

Aufgabe 2: (10 Punkte)

Markieren und verbessern Sie die Fehler (nicht mehr als 20!) in dem untenstehenden Programm, welches den kürzesten Abstand von Erlangen zu einem beliebigen Ort auf dem Globus berechnet. Dazu müssen der Name des Ortes und vor allem seine Koordinaten (Breiten- und Längengrad; östliche Länge und südliche Breite sind negativ!) Angegeben werden. Falsch erkannte Fehler werden negativ gewertet!

```
#include <stdio.h>
#include <math.h>
#define ERD_RAD 6371.221;
#define DGR 0.017453;
#define NAMELEN 30

struct ort {
  oname  [NAMELEN],
  double breite,
  double laenge
}

int moin()
{
  double dist = 1.0;
  struct ort erl = ( "Erlangen", 49.36, -11.02 );
  struct ort ort = { ’’, 0.0, 0.0 };
  struct ort *o = &ort;

  while(dist != 0.0) do   /* Abbruch: Koordinaten von
  {                          Erlangen eingeben! */
    printf("Ortsname:    ");
    scanf(%s , o->oname);
    printf("Breitengrad: ");
    scanf("%lf", &o->breite);
    printf("Laengengrad: ");
    scanf("%lf", &o->laenge);
    if ((o->breite==erl.breite) & (o->laenge==erl.laenge))
      dist = 0.0;
    else {
      bgl = sin(DGR * erl->breite) * sin(DGR * o->breite) +
            cos(DGR * erl->breite) * cos(DGR * o->breite)
            * cos(DGR * (erl.laenge - o.laenge);
      dist = ERDRAD * acos(bgl);
    }
    printf("Abstand zwischen %s und %s: %lf km\n";
            erl.oname, o->oname, dist);
  }

}
```


Aufgabe 3: (10 Punkte)

Was gibt das folgende Programm **ey** in Zeile 12 aus, wenn es wie folgt aufgerufen wird: **ey ha** ? Geben Sie die Inhalte der verwendeten Variablen nach jeder ausgeführten Anweisung an! Verwenden Sie dazu die untenstehende Tabelle!

```
1   #include <stdio.h>
2   void f(char *a, char *b);
3   int main(int argc, char *argv[])
4   {
5     int i, j=-1;
6     char a[256];
7     for(i=0; i<argc; i++) {
8       j += strlen(argv[i]);
9       f(&a[j], argv[i]);
10    }
11    a[++j] = ’\0’;
12    printf("count: %d, string: %s\n", j, a);
13    return j;
14  }
15  void f(char *a, char *b)
16  {
17    while(*b)
18    {
19      *a = *b;
20      a--; b++;
21    }
22  }
```

Zeile	i_{main}	j_{main}	a_{main} a[0], a[1], a[2], a[3], a[4], a[5], a[6], ...	a_f	b_f

Aufgabe 4: (6 Punkte)

Schreiben Sie eine rekursive Funktion, die die n-te Fibonacci-Zahl berechnet. Die Fibbonacci-Zahlen sind durch folgende rekurrente Formel definiert:

$$fib(0) = 0$$

$$fib(1) = 1$$

$$fib(n) = fib(n-1) + fib(n-2), \qquad \text{für } n \geq 2$$

Falls der Funktion ein ungültiger Wert für n übergeben wird, soll dies durch Rückgabe des Wertes -1 ausgedrückt werden.

Aufgabe 5: (14 Punkte)

a) Welches unerwartete Verhalten kann bei folgendem Programmstück auftreten, und warum ist dies so (nur den Grundgedanken erwähnen)?

```
float f;
for ( f = 0.1; f <= 1.0; f = f + 0.1 )
      printf ( "%f\n", f);
```

b) Erstellen Sie folgende Strukturen:

- Personendaten mit Vor- und Nachname (jeweils maximal 24 Zeichen), sowie Geburtsdatum (10 Zeichen).
- Studentendaten, bestehend aus Personendaten, Matrikelnummer (Ganzzahl) und Studienfach (maximal 18 Zeichen). Die Studentendaten sollen in einer verketteten Liste verwaltet werden.

Kreuzen Sie bei der folgenden Teilaufgabe **alle wahren Aussagen** an. Es können pro Teilaufgabe mehrere Antworten richtig sein! Falsche Antworten zählen negativ.

c) Welche Aussagen gehören zur Definition des Begriffs "Algorithmus"?

- ❑ endliche Beschreibung
- ❑ eindeutige Operationsfolge; es steht fest, welches der jeweils nächste Schritt ist
- ❑ muß in einer Programmiersprache formuliert sein
- ❑ Zeichen werden durch Zahlen codiert
- ❑ dynamische Speicherverwaltung für verwendete Daten

Die Teilaufgaben d) bis f) beziehen sich auf die unten stehenden Programmodule, die zu einem lauffähigen Programm zusammengefügt werden (z.B. durch den Compileraufruf cc -Aa -o prog prog.c mod.c).

Kreuzen Sie bei den folgenden Teilaufgaben **alle wahren Aussagen** an. Es können pro Teilaufgabe mehrere Antworten richtig sein! Falsche Antworten zählen negativ.

d)

- ❑ Zeile 13 in **prog.c** verursacht einen Fehler, da im umschließenden Block kein **a** bekannt ist.
- ❑ Das **++** in Zeile 18 wirkt sich nicht mehr auf **a** aus, da durch return zuvor das Programm beendet wird.
- ❑ In **mod.c** kann in Zeile 9 nicht auf **a** zugegriffen werden; es resultiert ein Compilerfehler.
- ❑ Zeile 8 in **prog.c** teilt der **main**()-Funktion mit, daß sie das **a** aus Zeile 3 der selben Datei mit 0 initialisieren soll.

Datei prog.c:

```
1   #include "mod.h"
2
3   static int a;
4   static int b = 0;
5
6   int main()
7   {
8       int a = 0;
9       c = 0;
10      b = f(b);
11      f(b);
12      if(b > 5) {
13          int c = a++;
14          printf("a=%d c=%d\n", a, ++c);
15          c++;
16      }
17      printf("a=%d c=%d\n", a, c);
18      return ++a;
19  }
```

Datei mod.h

```
1   extern int c;
2
3   int f(int b);
```

Datei mod.c

```
1   #include "mod.h"
2
3   static int b;
4   int c;
5
6   int f(int b)
7   {
8       static int c = 0;
9       a += b;
10      b = 5;
11      c += b;
12      return c;
13  }
```


e)

- ❑ Zeile 3 in **mod.c** legt eine modulglobale Variable **b** an, auf die aber nirgends zugegriffen wird.
- ❑ Zeile 10 in **prog.c** ist fehlerhaft, da eine Variable (hier **b**) nicht gleichzeitig auf der linken und rechten Seite einer Zuweisung stehen darf.
- ❑ Durch die Zeilen 4 (**prog.c**) und 3 (**mod.c**) wird ein Compiler-Fehler verursacht, da nicht zwei static-Variablen mit gleichem Namen definiert werden können.
- ❑ Die Anweisung in **mod.c**, Zeile 10 ändert weder das in Zeile 3 der selben Datei definierte **b**, noch das **b** in **prog.c**, Zeile 4.
- ❑ In Zeile 11 von **prog.c** wird der Rückgabewert der Funktion **f()** nicht verwendet, was zu einem Compiler-Fehler führt.

f)

- ❑ In Zeile 13 in **prog.c** darf kein neues **c** definiert werden, da bereits ein globales **c** existiert (**mod.c**, Zeile 4) und durch die erste Zeile der Header-Datei überall sichtbar gemacht wird.
- ❑ In Zeile 9, **prog.c** kann der Wert von **c** des Moduls **mod.c** (Zeile 4) auf 0 gesetzt werden, da die erste Zeile der Header-Datei **c** überall sichtbar macht.
- ❑ Bei jedem Aufruf der Funktion **f()** wird zuerst **c** auf den Wert 0 gesetzt (**mod.c**, Zeile 8), anschließend um 5 erhöht (Zeilen 10 und 11), so daß immer 5 als Rückgabewert der Funktion geliefert wird.
- ❑ **printf()** in **main()**, Zeile 14 gibt für **a** und **c** den gleichen Wert aus.
- ❑ Zeile 15 in **prog.c** hat keinen Einfluß auf das globale **c** (**mod.c**, Zeile 4).