Klausur Programmieren in C
Sommersemester 2007
Dipl. Biol. Franz Schenk
13. April 2007, 11.15-13.00 Uhr
Bearbeitungszeit: 105 Minuten

Vorname:

Nachname:

Matrikelnummer:

- Legen Sie zu Beginn Ihren Studentenausweis bereit.
- Schalten Sie ihr Mobiltelefon aus.
- Bei der Klausur ist **als einziges Hilfsmittel** eines unserer Skripte erlaubt.
- Schreiben Sie Ihre Antworten direkt auf die Aufgabenblätter. Falls benötigt, werden weitere Blätter gestellt. Benutzen Sie kein eigenes Papier für Ihre Lösungen!
- **Lassen Sie die Blätter zusammengeheftet!**
- Schreiben Sie mit blauem/schwarzem Kugelschreiber, Füller etc.; Bleistift ist nicht erlaubt.
- Bemühen Sie sich bitte um eine saubere, lesbare Schreibweise.

Zum **Bestehen** der Klausur sind **45** Punkte hinreichend.

☐ mein Ergebnis soll mit Matrikelnummer so bald wie möglich auf der Vorlesungs-Webseite veröffentlicht werden.

☐ ich erfahre das Ergebnis erst bei der Scheinausgabe (bei Frau Jachinke (Raum 01.112, IFI).

Wenn Sie keine Angaben machen, wird davon ausgegangen, dass Sie der Veröffentlichung des Ergenisses zustimmen.

Note:

Hinweise zu den Aufgaben:

- Gültiger Sprachstandard ist wie in der Vorlesung ANSI-C (89).
- Eine Aufgabe wird gewertet, wenn sie prinzipiell das Richtige tut. Auch richtige Teilabschnitte können Punkte geben. Systematische Syntaxfehler, nicht korrekte Umsetzung der Aufgabe oder Fehler im Algorithmus führen zu Punktabzügen.
- Falls nicht verlangt, so schreiben Sie keine vollständigen Programme, sondern nur C-Funktionen, wie angegeben! Wenn Sie Funktionen der Standardbibliothek verwenden, schreiben Sie die entsprechende `#include`-Zeile vor die Funktionsdefinition. Wenn nicht verlangt ist, dass eine Funktion Ausgaben produziert, dann dürfen Sie auch keine Ausgabefunktion benutzen!
- Mit den Aufgaben 1 und 2 sollen Sie Funktionen implementieren. Überlegen Sie erst, wie die Funktion aussehen soll. Achten Sie auf eine klare Strukturierung.
- Aufgaben 3, 4, und 5 sind keine Programmieraufgaben. Hier geht es um Ihr Verständnis von C.

Gutes Gelingen!

	Max. Punkte	Erwartet für “4”
Aufgabe 1 (Listensuche)	20	10
Aufgabe 2 (Stringersetzung)	30	15
Aufgabe 3 (Rekursion)	20	5
Aufgabe 4 (Programmfehler)	10	5
Aufgabe 5 (Ausdrücke)	25	10
Summe	105	45

Aufgabe 1 (Rechtschreibreform [20 Punkte])
Im Zuge der Rechtschreibreform ist noch immer einiges zu tun. Ein Redakteur möchte von Ihnen eine Funktion implementiert haben, welche in einem Text alle Vorkommnisse von ph (bzw. Ph) in f (bzw. in F) umwandelt.

```
void rechtschreibmod(char* satz);
```

Aufgabe: Schreiben Sie eine Funktion, welche obiger Deklaration genügt. Der Funktion wird ein String übergeben, in welchem alle vorkommenden *ph* durch *f* und alle *Ph* durch *F* ersetzt werden sollen.

- Sie sollen keine Funktionen aus anderen Bibliotheken benutzen.
- Sie sollen direkt mit dem übergebenen String (ohne Hilfsstring) arbeiten.

Lösung

```
#include <stdio.h>

void rechtschreibmod(char* satz)
{
  int lesezaehler = 0, schreibzaehler = 0;
  char zeichen;
  while (satz[lesezaehler])
  {
    zeichen = satz[lesezaehler];

    if (zeichen == 'h' && lesezaehler)
    {
      if (satz[lesezaehler-1]=='P')
         satz[schreibzaehler-1]='F';
      else if (satz[lesezaehler-1]=='p')
         satz[schreibzaehler-1]='f';
    }
    else
      satz[schreibzaehler++]=satz[lesezaehler];

    lesezaehler++;
  }
  satz[schreibzaehler]='\0';
}
```

Aufgabe 2 (Klausurauswertung [30 Punkte])
Gegeben ist in einem Programm folgende Struktur:

```
typedef struct
{
  char nachname[30];
  float note;
} student;
```

Schreiben Sie ein Programm unter Benutzung dieser Struktur, welches folgende Funktion erfüllen soll:

- Das Programm erfragt vom Benutzer die Anzahl N Studenten, zu welchen im nächsten Schritt Daten eingegeben werden sollen.
- Als nächstes werden N Datensätze bestehend aus Name und Note eingelesen.
- Das Programm gibt die Noten aller Studenten aus, welche die beste Note erhalten haben. (es können mehrere Studenten zugleich die besten sein!). Die beste Note ist nicht notwendigerweise 1.0 sondern die beste der erreichten Note.
- Das Programm berechnet schliesslich die Durchschnittsnote aller Studenten und gibt die Durchschnittsnote aus.
- Beachten Sie, dass Sie selbst Speicher auf dem Heap bereitstellen müssen.
- Überprüfen Sie, ob die Speicherallokation funktioniert hat.
- Überprüfen Sie ausserdem, ob die eingegebene Note gültig ist (Noten zwischen 1.0 und 5.0).

Lösung

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

typedef struct
{
  char name[30];
  float note;
} student;

void datensatz_eingabe(student * const stud)
{
  printf("Nachname eingeben: ");
  scanf("%s", stud->name);
  printf("Note eingeben: ");
  scanf("%f", &stud->note);
  if (stud->note <1.0 || stud->note >5.0){
    printf("Eingabe ungueltig, nochmal...\n");
```

```
    datensatz_eingabe(stud);
  }
}

int main(void)
{
  int n,i;
  float summe=0,best=5.0;
  student *studenten;
  printf("Anzahl Prueflinge eingeben: ");
  scanf("%d",&n);

  /* Speicher anfordern */
  studenten = (student*) malloc(n*sizeof(student));
  if(studenten == NULL)
    exit(1);

  for (i=0;i<n;i++)
    datensatz_eingabe(&studenten[i]);

  for (i=0;i<n;i++)
  {
     summe += studenten[i].note;
     if (studenten[i].note < best)
        best = studenten[i].note;
  }
  for (i=0;i<n;i++)
    if (studenten[i].note == best)
        printf("%s gehoert zu den besten!\n",studenten[i].name);


  printf("Durchschnitt: %f\n",summe/n);

  /* Speicher wieder freigeben */
  free(studenten);
  return 0;
}
```

Aufgabe 3 (Rekursion [20 Punkte])
Gegeben ist folgendes Programmfragment:

```
float mystery_function(int *a, int b)
{
  if (b <= 0)
     return a[0];
  else
     return (a[b] + b * mystery_function(a, b-1)) / (b+1);
}
```

Die Funktion wird wie folgt aufgerufen:

```
int a[] = {3,6,4,12,5};
int laenge =  5;
ergebnis = mystery_function(a, laenge - 1);
```

wobei die Variable *laenge* offensichtlich als Wert die Laenge des Feldes a enthaelt.

- Welche einfache mathematische Funktion wird mit der Funktion *mystery_function* implementiert?

  Lösung Mittelwert über alle Feldwerte.

- Welchen Wert nimmt die Variable *ergebnis* nach dem Funktionsaufruf an? Schreiben Sie auf, welche Werte $a[b]$ und b in jedem einzelnen Rekursionsschritt annehmen.

  Lösung 6

Aufgabe 4 (Programmfehler [10 Punkte])
Das folgende Programm soll die Summe aller natürlichen Zahlen berechnen, deren Obergrenze durch das Makro MAXIMUM bestimmt wird und gibt diese Summe aus.
Im Programm stecken zahlreiche Fehler.

- Einige dieser Fehler bewirken, dass das Programm nicht korrekt compiliert. Markieren und berichtigen Sie diese Fehler. [9 Punkte]
- Ein Fehler betrifft die Richtigkeit des Algorithmus (es wird also nicht, wie erwartet, die Summe gebildet). [1 Punkt]

  Zeigen Sie diese Fehler und schreiben Sie Korrekturen, so dass ein lauffähiges Programm ohne Warnungen nach dem ANSI-C 89 Standard compiliert wird (mit den üblichen Parametern -Wall -pedantic -ansi). Zeigen Sie auch, welcher Fehler verhindert, dass der Algorithmus korrekt funktioniert.

```
01   #include 'stdio.h'
02
03   #define MAXIMUM 23 ;
04
05   int main()
06   {
07     int zahl == 0;
08
09     for ( int i = 0 , i < MAXIMUM , i++) ;
10     {
11    zahl =+ i;}
12
13     printf('Summe von 1 .. %f: %f\n',MAXIMUM,zahl);
14   }
15
```

Lösung

```
01   #include <stdio.h>    /* Hochkomma statt spitzer Klammern*/
02
03   #define MAXIMUM 23   /*  semikolon fuehrt zu Fehler in for-Schleife!  */
04                        /*  und printf-Funktionsaufruf */
05   int main()
06   {
07     int zahl =  0;  /* Vergleich statt Zuweisung */
08     int i;  /* unerlaubte Definition innerhalb des Blocks */
09     for ( i = 0 ;  /* Komma statt Semikolon */ i < MAXIMUM ; i++)
10     {                                        /* Semikolon hinter for() */
11    zahl =+ /* Wertzuweisung, keine Addition*/ i; }
12
13     printf("Summe von 1 .. %d: %d\n",MAXIMUM,zahl);
13                            /*Anfuehrungszeichen , Formatbeschreiber*/
14   return 0;}  /*Fehlender return*/
15
```

Aufgabe 5 (Ausdrücke [25 Punkte])

- Gegeben ist ein Programm mit einigen Ausdrücke. Markieren Sie, welche der folgenden Ausdrücke compilieren oder die Compilation des Programms mit einer Fehlermeldung verhindern!

  [Jede richtige Antwort gibt 2 Punkte, jede Falsche Antwort -1 Punkt. Insgesamt negative Punkte für diesen Aufgabenteil können Sie aber nicht erreichen!]

Code	compiliert	compiliert nicht
int main(void) { int i; char c = 'c';		
int *****ip;	☐	☐
i = 'a'+ c;	☐	☐
c = (i==1);	☐	☐
i++ = c;	☐	☐
while (i--) i++;	☐	☐
return 0; }		

Fortsetzung nächste Seite!

Lösung

Code	compiliert	compiliert nicht
int *****ip;	☒	☐
i = 'a'+ c;	☒	☐
c = (i==1);	☒	☐
i++ = c;	☐	☒
while (i--)i++;	☒	☐

- Lesen Sie die folgenden Ausdrücke. Welchen Wert nehmen die Variablen nach den Zuweisungen an?

  [1 Punkt pro richtigem Variablenwert]

  (a) Lazy evaluation

```
int a=0,b=0,c=1, d=1,e=0;
e = (--c || --d) && (a++ && ++b);
```

  a..... b..... c..... d..... e.....

  Lösung a = 0 b = 0 c = 0 d = 0 e = 0

  (b) Bedingte Ausdrücke

```
int a = 0, b = 0, c = 1, d = 1;
( (a) ? (a=1) : (b=1) ) ? (c=0) : (d=0);
```

  a..... b..... c..... d.....

  Lösung a = 0 b = 1 c = 0 d = 1

  (c) Felder und Zeiger

```
int feld[2][4]={{1,2,3,4},{5,7,9,11}};
int* p;
int i,j,k;
k = 1;
p = &j;
for ( i = 0 ; i < 3 ; i++)
   j = feld[k][i];
k = *(feld[1])+2;
```

  feld[1][2] ..... j ..... k .....

  *p ..... **feld ..... *(feld[1] + 2) .....

  Lösung feld[1][2] = 9 j = 9 k = 7 *p = 9
  *feld = 1 *(feld[1]+2) = 9;