

Einführung in die Informatik

Sonder-Blatt — Aufgaben zur Prüfungsvorbereitung

Wichtige Vorbemerkung:

Die Prüfung am Semesterende wird nicht am Computer durchgeführt. In einer Klausur wird vorrangig verlangt, gewisse Probleme algorithmisch aufzubereiten und in Form eines Programms (bzw. als Funktion) in C zu lösen.

Diese Testarbeit ist eine Hilfe dazu, sich rechtzeitig und systematisch auf die Klausur vorzubereiten. Es sind mehr Aufgaben als in der Klausur vorgegeben, damit wird das Spektrum der Aufgaben gezeigt. In der Klausur werden es etwa 5 Aufgaben sein.

Viel Spaß beim Grübeln!

Aufgabe 1: Lage eines Punktes

Im kartesischen Koordinatensystem sei ein Punkt $P(x,y)$ durch seine Koordinaten gegeben. Es ist ein Programm zu schreiben, das feststellt, ob sich der Punkt P im Ursprung, auf der x - bzw. y -Achse oder im I., II., III. bzw. IV. Quadranten befindet. Die Koordinaten x,y des betreffenden Punktes sind über Tastatur einzulesen, das Ergebnis ist verbal auf dem Bildschirm auszugeben.

Aufgabe 2: Matrizen

Führen Sie einen Typ `mat` für Matrizen ein, der globale Gültigkeit im gesamten Programm besitzt, z.B.

```
#define maxN 20
typedef float mat [maxN][maxN];
```

Es sind folgende Funktionen zu schreiben:

- Untersuchung einer quadratischen Matrix darauf, ob sie eine obere bzw. untere Dreiecksmatrix ist,
- Untersuchung einer Matrix darauf, ob sie symmetrisch ist, d.h. 1.) quadratisch ist und 2.) $A = A^T$ gilt.

Aufgabe 3: (Syntaktische und inhaltliche Analyse eines Programms)

Das folgende Programm ist ohne Nutzung des Computers zu analysieren.

- Im folgenden C-Programm sind alle (6 -9 kleinen) syntaktischen Fehler anzugeben und zu berichtigen. Fehlerhafte Stellen sind deutlich zu markieren (nicht mit Rotstift!!) und die Korrektur daneben anzubringen Jede fälschlich als syntaktisch falsch markierte Stelle wird mit einem Minuspunkt bestraft!
- Was erscheint auf dem Bildschirm, wenn nach Beseitigung der syntaktischen Fehler das Programm gestartet wird. Dabei sind Zeilen- und Spaltenstruktur genau zu beachten.

```
#include <stdio.h>
void fkt(int n, int p[], int m, int q[], int r[]);
{int i,k=0,j=0;
  printf(" i j k|      r\n");
  printf("=====\n");
  for (i=0; i<n+m; i++;)
  { if (k>=n)
    {r[i]=q[j]; j++;}
    else if j>=m
    {r[i]=p[k]; k++;}
    else if (p[k] > q[j])
    {r[i]=p[k]; k++;}
    else {r[i]=q[j]; j++;}

    printf(" %d %d %i|",i,j,k);
    for (N=0;N<=i;N++)
      printf("%3d",r[N]);
    printf("\n");
  } //for-Anw
} //fkt

int main()
{int m=4;
  int p[22]={12 , 9 , 8 , 4 , -4 , -6}, q[22]={13 , 9 , 6 , 5 , 5 , -3}, r[44];
  {Kommentar: die Felder p und q sind teilweise mit Anfangswerten belegt
    z.B. p[0]=12 , p[1]=9, ... , p[5]=-6}
  fkt(3,p,m,q,r,p);
  return 0;
} //main
```

Aufgabe 4:

Gegeben sei die unsortierte Folge von ganzen Zahlen, die in einem Feld abgespeichert sind: 8 3 5 7 0 1 6 2 . Zu simulieren ist die Arbeit eines Computers, wenn dieser die angegebene Folge mittels Insertion Sort sortiert. In einer Tabelle sind in den ersten 8 Spalten die Anfangs- und dann die jeweiligen Zwischenstnde des Feldes nach jedem Sortierschritt zu erfassen. Gleichzeitig ist die in diesem Sortierschritt erforderliche Anzahl von Vergleichen und von Verschiebungen anzugeben (Spalten 9 und 10), dabei ist der Test auf den Anfang des Feldes nicht zu bercksichtigen.

Aufgabe 5: (rekursive und nichtrekursive Programmierung)

Die Glieder einer von einem reellen Parameter p abhängigen Folge seien durch folgende rekursive Berechnungsvorschrift definiert:

$$b_{m+2} = \frac{-p^2}{(m+1)(m+2)} b_m \quad \text{mit } b_1 = p \text{ und } b_0 = 1$$

Es ist eine rekursive Funktion `term` zu schreiben, die bei vorgegebenem Index N das Glied b_N der oben angegebenen Folge berechnet.

Weiterhin ist eine Funktion `sum` mit dem Prototyp

```
float sum (int N, float p);
```

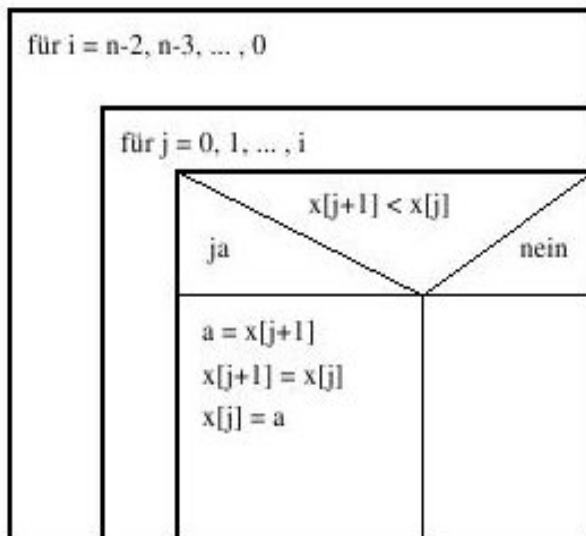
zu schreiben, die bei geradzahligem N alle Glieder der Folge mit geradem Index und bei ungeradzahligem N alle Glieder der Folge mit ungeradem Index aufsummiert, wobei die Indizes in beiden Fällen zwischen 0 und N variieren. Der Parameter p sei dem Betrag nach kleiner als 2 zu wählen.

Aufgabe 6: (Bubblesort)

- a) Für ein Feld von `float`-Werten ist das untenstehende Struktogramm für den Bubblesort-Algorithmus in einer Funktion mit dem Prototyp

```
void bubblesort ( float feld[], int anzahl);
```

exakt umzusetzen.



- b) Es ist ein Hauptprogramm zu schreiben, das die Funktion `bubblesort` testet: Dazu ist ein entsprechendes Feld zu deklarieren und über Tastatureingabe mit Werten zu belegen. Anschließend ist die Funktion aufzurufen und die Werte des sortierten Feldes auf dem Bildschirm auf 3 Stellen nach dem Komma genau auszugeben.

Aufgabe 7: Studentendatei

- a) Es ist eine Datenstruktur `schueler` mit den drei Komponenten `vorname`, `name` und `mathezensur` zu definieren, wobei `vorname` und `name` Zeichenketten mit maximaler Länge 20 und `mathezensur` vom Typ `int` sein sollen.
- b) Es ist eine Funktion `func_zensur` zu schreiben, die aus einem Feld der Länge N solcher Datenstrukturen `schueler` alle diejenigen heraussucht, die eine vorgegebene Zensur in Mathematik erreicht haben, und diese zeilenweise in eine Textdatei mit vorgegebenem Namen schreibt.
Das Feld, die Zensur und der Name der Textdatei sind als Parameter an die Funktion zu übergeben. Die Funktion soll eine Information dazu zurückgeben, ob es Schüler mit der vorgegebenen Zensur gibt oder nicht (z.B. durch einen `int`-Rückgabewert '0' bzw. '1').

Hinweis: Zum Schreiben auf bzw. Lesen von Textdateien stehen die Standardfunktionen `fprintf` und `fscanf` zur Verfügung.

Es ist **nicht** verlangt, eine Funktion `main` zu schreiben.