Abiturprüfung 2003

Aufgabe 1

1.1

In einer Reihung sind n ganze Zahlen gespeichert. Die Zahlen sollen sortiert werden. Beschreiben sie am Beispiel der folgenden Reihung, wie dies mit Hilfe des Algorithmus „Sortieren durch Auswählen“ abläuft.

a1	*a2*	***a3***	*a4*	*a5*	*a6*
10	*17*	***-36***	*48*	*0*	*20*

Beim Algorithmus „Sortieren durch Auswählen“ wird das Minimum im noch unsortierten Teil gesucht und mit dem ersten Element dieses noch unsortierten Teiles getauscht. Das Minimum befindet sich im Beispiel an Stelle a3 und hat einen Wert von –36. Es wird nun mit a1 (Wert: 10) getauscht.

a1	*a2*	***a3***	*a4*	*a5*	*a6*
-36	*17*	***10***	*48*	*0*	*20*

Die untere Grenze des noch nicht sortierten Teiles wird um eins erhöht. (Das Sortierverfahren wird im folgenden also auf die Elemente a2 bis a6 angewendet. Das Element a1 gehört nun zum bereits sortierten Teil und wird nicht mehr berücksichtigt)
Von a2 bis a6 wird das Minimum gesucht und mit a2 getauscht.
Minimum (0 an Position a5) wird mit 17 (an Position a2) getauscht.

a1	***a2***	*a3*	*a4*	***a5***	*a6*
-36	***0***	*10*	*48*	***17***	*20*

Diese Operationen (Erhöhen der unteren Grenze des unsortierten Teiles, Suche des Minimums, Tauschen mit dem ersten Element des noch unsortierten Teiles) werden so lange ausgeführt, bis der unsortierte Teil nur noch aus dem letzten Element besteht.

Die weiteren Tauschoperationen wären:

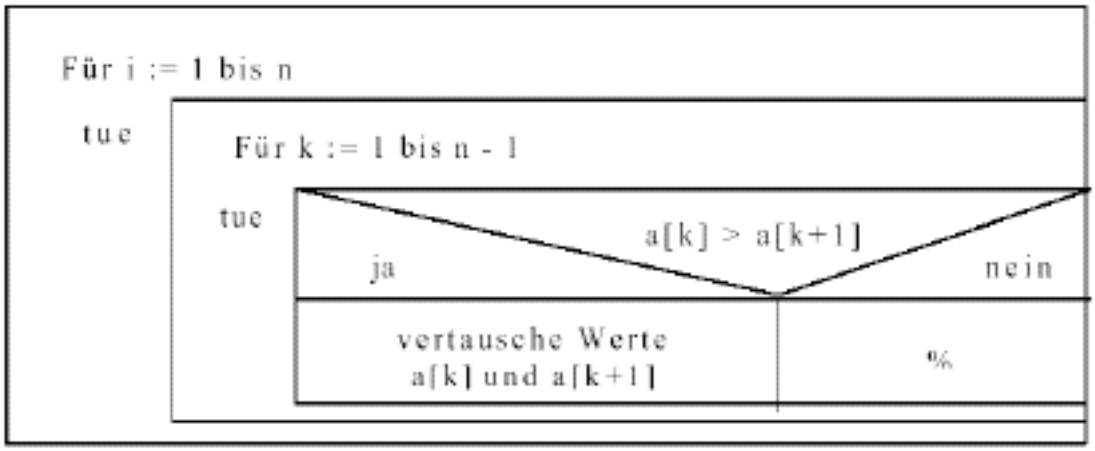
Die 10 wird nicht getauscht, da sie Minimum von a3 bis a6 ist.

Danach die 17 mit der 48, die 20 mit der 48. Danach ist im Rest nur noch ein Element enthalten (48), die Folge ist vollständig sortiert. Das Sortierverfahren bricht ab.

Die vollständig sortierte Folge lautet:

-36 0 10 17 20 48

1.2 Der folgende Algorithmus sortiert n ganze Zahlen in aufsteigender Reihenfolge:



a ist eine Reihung von n ganzen Zahlen.
i und k sind ganze Zahlen.

Geben Sie jeweils die Anzahl der Vergleiche und die Anzahl der Vertauschungen für die folgenden Fälle an:

1. Fall: Die n ganzen Zahlen der Reihung a sind bereits in aufsteigender Reihenfolge gegeben.

2. Fall: Die n ganzen Zahlen der Reihung a sind in absteigender Reihenfolge gegeben.

Geben Sie eine Möglichkeit zum Verbessern des Zeitverhaltens des gegebenen Algorithmus an!

Beim 1. und 2. Fall werden Vergleiche jeweils n(n-1) mal ausgeführt.*
Die erste Schleife wird n mal ausgeführt und die zweite (n-1) mal ausgeführt.

1. Fall: Bei aufsteigender Reihenfolge werden n(n-1) Vergleiche durchgeführt, aber es findet keine Vertauschung statt.*

*2. Fall: Bei absteigender Reihenfolge sind die Vertauschungen (n-1) * n/2*

Beim letzten Durchlauf werden keine Vertauschungen mehr durchgeführt, er kann also weggelassen werden. Nach dem ersten Gesamtdurchlauf ist die größte Zahl am Ende der Reihung, somit braucht man diese nicht mehr vergleichen.

1.3

Gegeben ist der Quelltext der Funktion f:

Oberon:

```
PROCEDURE f(x: INTEGER): INTEGER;
BEGIN
  IF x = 0 THEN RETURN 0
  ELSIF x > 0 THEN RETURN f(x - 3)
  ELSE RETURN f(x + 3)
  END
END f;
```

Turbo Pascal:

```
FUNCTION f(x: INTEGER): INTEGER;
BEGIN
  IF x = 0 THEN f := 0
  ELSE
     IF x > 0 THEN f := f(x - 3)
     ELSE f := f(x + 3)
END;
```

Begründen Sie, dass es sich bei der Funktion f um eine rekursive Funktion handelt!
Berechnen Sie die Funktionswerte f(3), f(9), f(8), f(-12) und f(-7) !
Nennen Sie Eigenschaften eines Algorithmus!
Überprüfen Sie, ob diese Eigenschaften von der Funktion f erfüllt werden!

Die Funktion ist teilweise durch sich selbst definiert, sie ruft sich an zwei Stellen – f(x-3) und f(x+3) - selbst auf, deswegen ist sie rekursiv.
f(3) Funktionswert 0
f(9) Funktionswert 0
f(8) endlose Rekursion
f(-12) Funktionswert 0
f(-7) endlose Rekursion

Eigenschafen eines Algorithmus:

1. *Jeder Algorithmus löst eine Klasse von Problemen.*
2. *endliche Beschreibung*
3. *endliche Abarbeitung*
4. *Determiniertheit (gleiche Eingabe → gleiche Ausgabe)*

Die endliche Abarbeitung ist nur für durch 3 teilbare Eingaben in die Funktion gegeben. Die restlichen Eigenschaften werden vom gegebenen Algorithmus erfüllt.

Aufgabe 2

2.1

Die Allgegenwart von Computern in der Gesellschaft vermittelt den Eindruck, als sei alles mit Computern berechenbar.

Beurteilen Sie diese These!

Es ist nicht alles mit Computern berechenbar. In manchen Fällen ist es nicht sinnvoll die Berechnungen vorzunehmen, da das Zeitverhalten ungünstig ist.
Unlösbare Probleme:
- *Halteproblem*
- *allgemeine Probleme der Menschheit, z.B. sichern des Weltfriedens, Erhaltung der Umwelt, ...*

NP- Probleme sind zwar lösbar haben jedoch ein ungünstiges Zeitverhalten.
Beispiele:
- *Turm von Hanoi*
- *Problem der Handlungsreisenden (Berechnung der kürzesten Rundreise-Route durch n Städte)*

P- Probleme sind lösbar und haben eine günstige (polynomiale) Zeitabhängigkeit.
Beispiel: - *Sortier- und Suchverfahren*

2.2 Würdigen Sie die Leistungen einer Persönlichkeit, die als Wegbereiter der Informatik bekannt ist!

Als ein Wegbereiter der Informatik gilt Konrad Zuse. Unter anderem folgende Leistungen sind als überragend zu bezeichnen:
- Er entwickelte mit dem Z1 den ersten funktionsfähigen Computers der Welt.
- Er setzte sich für die Verwendung von Dualzahlen im Computern ein.
- *Er entwickelte die erste Programmiersprache (Plankalkül) die jedoch nie auf einem Computer implementiert und eingesetzt wurde.*

Die von ihm gegründete Firma Zuse KG., die Computer herstellte, ging später in den Firmen Nixdorf und Siemens-Nixdorf auf.

2.3 Vor rund fünfzig Jahren erarbeitete John von Neumann ein später nach ihm benanntes Rechnermodell. Dieses Modell beschreibt auch heute noch die grundlegende Arbeitsweise eines PC.

Erläutern Sie das von-Neumann-Rechnermodell!

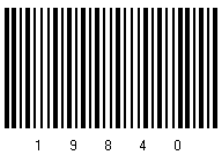
Der von- Neumann Rechner besteht aus dem Eingabewerk, dem Speicher, Ausgabewerk, Rechenwerk und Steuerwerk. Diese Einheiten kommunizieren über ein Bussystem miteinander.
Die Struktur des Rechners ist unabhängig von den zu lösenden Problemen.
Ohne Programm (Bearbeitungsvorschrift) ist der Rechner nicht arbeitsfähig.
Programme, Daten, Zwischen- und Endergebnisse werden im selben Speicher abgelegt.
Alle Daten, Befehle und Adressen werden binär codiert.
Alle heutigen Computer basieren auf diesem Rechnermodell.

2.4 Erläutern Sie Auswirkungen des Einsatzes von Computern in der Arbeitswelt der Gegenwart!

Computer werden häufig in der Arbeitswelt eingesetzt.
Es werden Arbeitsplätze geschaffen und vernichtet. Computer werden häufig in Büros, in der Industrie und im Servicebereich eingesetzt.
Die Vorteile von Computerarbeit liegen darin das sie billiger und schneller Produzieren, neue Industriezweige und Arbeitsräume geschaffen werden.
Außerdem kann man mit Hilfe des Internets weltweit Daten und Informationen austauschen.
Trotz der genannten Vorteile gibt es auch Nachteile. Arbeitsplätze werden vernichtet, Menschen isolieren sich und verlieren den sozialen Kontakt zu Mitmenschen. Außerdem können körperliche Schäden von Computerarbeit davon getragen werden, wie Haltungsprobleme.

Aufgabe 3

Der Barcode 2/5 Industrial wird zur Identifizierung von Objekten verwendet. Mit dem Barcode lassen sich beliebige nichtnegative ganze Zahlen codieren.
Nachfolgend wird erläutert, wie die Zahl 1984 codiert wird (siehe nebenstehende Abbildung). Die Ziffer 0 ist eine Prüfziffer.



1. Schritt:
Die Zahl wird ziffernweise in eine Folge von Nullen und Einsen überführt. Dabei wird die folgende Tabelle benutzt:

Ziffer	1. Stelle	2. Stelle	3. Stelle	4. Stelle	5. Stelle
0	0	0	1	1	0

1	1	0	0	0	1
2	0	1	0	0	1
3	1	1	0	0	0
4	0	0	1	0	1
5	1	0	1	0	0
6	0	1	1	0	0
7	0	0	0	1	1
8	1	0	0	1	0
9	0	1	0	1	0

Die Zahl 1984 wird in die Folge 10001010101001000101 überführt.

2. Schritt:
Zur Berechnung der Prüfziffer werden die Ziffern der gegebenen Zahl von links nach rechts abwechselnd mit dem Faktor 3 oder mit dem Faktor 1 multipliziert. Begonnen wird links mit dem Faktor 3. Die entstandenen Produkte werden addiert. Die Prüfziffer ist die kleinste nichtnegative ganze Zahl, die zu dieser Summe zu addieren ist, um ein Vielfaches von 10 zu erhalten.

Für 1984 ergibt sich die Summe 1*3 + 9*1 + 8*3 + 4*1 = 40 und daher die Prüfziffer 0. Die Prüfziffer wird mit Hilfe der Tabelle in eine Folge von Nullen und Einsen überführt und an die bisherige Folge angehängt. Für 1984 erhält man die Folge 10001010101001000101001 10.

3. Schritt:
Am Anfang und am Ende der Folge werden ein Start- und ein Stoppzeichen ergänzt. Das Startzeichen wird durch 110, das Stoppzeichen durch 101 dargestellt. Für 1984 ergibt sich nun die vollständige Folge 11010001010101001000101001 10101. Bei der Ausgabe des Barcodes wird für jede 1 ein breiter und für jede 0 ein schmaler schwarzer Strich gedruckt.

In den folgenden Aufgaben wird unter dem Barcode stets die vollständige Folge von Nullen und Einsen verstanden.

a) Erzeugen Sie den Barcode für die Zahl 286!

Barcode von 286 :
010011001001100

Prüfziffer:
*2*3 + 8*1 + 6*3 = 32*
32 + 8 = 40 → 8= Prüfziffer
Barcode von 8:
10010

Startzeichen + Barcode von 286 + Prüfziffer + Stoppzeichen =
11001001100100110010010101

b) Die Funktion ziffer(z) soll die Prüfziffer für die Zahl z ermitteln. Geben Sie den Quelltext der Funktion in Oberon oder Turbo Pascal an!

Im folgenden ist ein komplettes Modul angegeben. Zur Lösung reicht die fett hervorgehobene Funktionsprozedur.

```
MODULE Pruefziffer;

IMPORT In,Out;

VAR
  zahl: ARRAY 10 OF INTEGER;
  anzahl: INTEGER;

PROCEDURE ziffer (a: INTEGER): INTEGER;

VAR
    summe, faktor, i: INTEGER;
    pruef: INTEGER;
   (* Deklaration der Variablen *)

BEGIN

summe:=0;
faktor:=3;
(* sinnvolle Definition der Anfangswerte *)

FOR i:= 1 TO anzahl DO

   summe:= summe + zahl [i]*faktor;
   IF faktor=3 THEN faktor:=1 ELSE faktor:=3 END
   (* Damit wird der Wechsel des Faktors 3 – 1 – 3 – 1 - … realisiert. *)
END;

Out.Int(summe, 10);
Out.Ln;
Pruef:= summe MOD 10;
(* Ermittlung der Prüfsumme durch Restbildung *)

IF pruef # 0 THEN pruef:=10 – pruef END;
(* Nur wenn der Rest ungleich Null ist, muss die Ergänzung zur nächsten
durch 10 teilbaren Zahl gesucht werden *)
RETURN pruef

END ziffer;
```

```
PROCEDURE Start*;

BEGIN
   Anzahl:=0;
In.Open;
WHILE In.Done DO
Anzahl:= anzahl + 1;
In.Int(zahl[anzahl])
END;

Out.Int(ziffer(anzahl),5);
Out.Ln
END Start;

BEGIN
Out.Open;
END Pruefziffer.
```

c) Geben Sie eine Datenstruktur an, die sich zum Speichern eines Barcodes eignet! Begründen Sie Ihre Antwort!

Barcode: ARRAY 32 of INTEGER

In den vorgegebenen Beispielen besteht der Barcode aus maximal 31 verschiedenen Ziffern 0 und 1. Jede dieser Ziffern kann in das gegebene Feld eingelesen und an den Positionen Barcode[1] bis Barcode [31] gespeichert werden.
Möglich wären auch die Typen String und FILE.

d) Entwerfen Sie einen Algorithmus, der einen Barcode einliest und die durch ihn codierte Zahl ermittelt und ausgibt!

Um die codierte Zahl ermitteln zu können muss der Barcode eingelesen werden. Als zweites werden die Start- und Endzeichen, die ersten und letzten 3 Zeichen, entfernt.
Als nächstes erfolgt die Entfernung der Prüfziffer, das sind die letzten 5 Zeichen. Die noch übrige Zeichenkette wird von der Letzten bis zur Ersten Ziffer in jeweils Fünfer - Zeichenfolgen eingeteilt, und laut Tabelle übersetzt. Man multipliziert die letzte Ziffer mit dem Faktor 1 und speichert diesen Wert als Zahl. Die nächste Fünfer - Zeichenfolge wird übersetzt und mit 10 Multipliziert. Die darauffolgende Zeichenfolge mit 100, die Letzte mit 1000, multipliziert. Die Teilergebnisse müssen zusammen addiert werden
Der nun vollständig bestimmte Barcode wird ausgegeben.

e) Beim Einlesen eines Barcodes können Eingabefehler auftreten. Erläutern Sie, wie beim Barcode 2/5 Industrial Eingabefehler erkannt werden können! Gehen Sie in Ihrer Erläuterung auf die Codierung der einzelnen

Ziffern (siehe Tabelle) sowie auf die Verwendung einer Prüfziffer, eines Start- und eines Stoppzeichens ein!

Eingabefehler können wie folgt erkannt werden:
Jede Ziffer des Barcodes (0 ..9) unterscheidet sich von jeder anderen an mindestens zwei Positionen, wie man es der Tabelle entnehmen kann. Somit wird eine Verfälschung einer Position sicher erkannt. In diesem Fall würde das Lesegerät durch eine nicht gültige Ziffer auf einen Lesefehler hinweisen. Wenn es zu einer Verfälschung an zwei Positionen kommt, wird dies durch die Prüfziffer erkannt. Eine Verfälschung der Prüfziffer oder Start- und Stoppfolge wird ebenfalls sofort erkannt, da dadurch die Eingabe nicht ordnungsgemäß gestartet und gegebenenfalls auch beendet werden kann. In diesem Fall sind ebenso Missdeutung von Ziffern und Differenzen zwischen berechneter und angegebener Prüfziffer möglich.

Aufgabe 4

An einem Spiel nehmen vier Personen teil. Wir nennen diese Personen A, B, C und D. Für das Spiel werden ein Spielbrett, eine Münze und für jeden Spieler ein Spielstein benötigt. Das Spielbrett ist eine lineare Anordnung von 60 Feldern. Die Felder tragen die Nummern von 1 bis 60. Feld Nr. 1 ist das Startfeld, Feld Nr. 60 das Zielfeld. Die Münze hat eine Vorder- und eine Rückseite.

Start-feld	A C D	B					Ziel-feld
1	2	3	4	5	...	59	60

Zu Beginn des Spiels stehen alle Spielsteine auf dem Startfeld.
In jeder Spielrunde wird die Münze von den Spielern in der Reihenfolge A, B, C und D geworfen. Liegt die Vorderseite der Münze oben, so wird der Spielstein des jeweiligen Spielers um zwei Felder in Richtung Zielfeld gerückt. Liegt die Rückseite oben, wird der Spielstein um ein Feld in Richtung Startfeld gerückt. Der Spielstein wird nicht gerückt, wenn er sich auf dem Startfeld befindet und die Rückseite der Münze oben liegt oder wenn er sich auf dem Feld
Nr. 59 befindet und die Vorderseite oben liegt. Das Spiel wird beendet, wenn in einer Spielrunde mindestens ein Spieler das Zielfeld erreicht. Gewinner des Spiels sind alle Spieler, deren Spielsteine am Ende dieser Spielrunde auf dem Zielfeld stehen.

Entwerfen und implementieren Sie ein Oberon- oder Turbo Pascal-Programm, das dieses Spiel für vier Personen simuliert! Das Rücken der

Spielsteine soll während des gesamten Spiels auf dem Monitor dargestellt werden. Der Zufallszahlengenerator des Oberon- oder Turbo Pascal-Systems ist zur Simulation des Werfens der Münze zu verwenden.

Entwurf:

Die Orte der Spieler werden in einem ARRAY 5 OF INTEGER mit dem Namen Orte festgehalten.

Spieler A: Orte[1];
Spieler B: Orte[2];
Spieler C: Orte[3];
Spieler D: Orte[4];

Zu Beginn werden alle Orte auf den Wert 1 gesetzt.

Der Münzwurf wird mittels RandomNumbers.Uniform() realisiert:

zufall:=RandomNumbers.Uniform() liefert eine reelle Zufallszahl größer oder gleich 0 und kleiner als 1.

*zufall:=RandomNumbers.Uniform()*2 liefert eine reelle Zufallszahl größer oder gleich 0 und kleiner als 2.*

*zufall:=ENTIER(RandomNumbers.Uniform()*2) liefert ganze Zufallszahlen 0 oder 1 mit dem Datentyp LONGINT.*

*zufall:=SHORT(ENTIER(RandomNumbers.Uniform()*2))) liefert ganze Zufallszahlen 0 oder 1 mit dem Datentyp INTEGER.*

0 bedeutet Rückseite, 1 bedeutet Vorderseite der Münze.

Es wird solange gewürfelt bis einer der Orte den Wert 60 hat – ODER-Verknüpfung!

Neben RandomNumbers ist noch das Modul Out zu importieren.

Quelltext:

```
MODULE Spiel;

IMPORT In, Out, RandomNumbers;

VAR
  Orte:ARRAY 5 OF INTEGER;
```

```
PROCEDURE Anfang;

VAR
  i:INTEGER;

BEGIN
  FOR i:=0 TO 4 DO
    Orte[i]:=1;
  END;
  (* Zu Beginn werden alle Steine auf das erste Feld
gesetzt. *)
END Anfang;

PROCEDURE Start*;

VAR i,zufall:INTEGER;  v:LONGINT;


BEGIN
  Anfang;
  Out.String('Spieler1 Spieler2 Spieler3 Spieler4');
  Out.Ln;
  Out.String('----------------------------');
  Out.Ln;
  REPEAT
    FOR i:=1 TO 4 DO
      zufall:=SHORT(ENTIER(RandomNumbers.Uniform()*2));
      (* 0:=Rueckseite der Muenze; 1:=Vorderseite *)
      IF (zufall=0) & (Orte[i]#1) THEN
     (* Der Spielstein befindet sich nicht auf Feld 1 und
     kann demzufolge einen Schritt zurück gehen. *)
        Orte[i]:=Orte[i]-1;
      END;
      IF (zufall=1) & (Orte[i]#59) THEN
     (* Der Spielstein befindet sich nicht auf Feld 59 und
     kann demzufolge zwei Schritte vorrücken. *)
        Orte[i]:=Orte[i]+2;
      END;
    Out.Int(Orte[i],11);
    END;
    Out.Ln;
    FOR v:=1 TO 30000000 DO
        (* Verzoegerung *)
    END;
  UNTIL (Orte[1]=60) OR (Orte[2]=60) OR (Orte[3]=60) OR
(Orte[4]=60);
END Start;

BEGIN
  Out.Open
END Spiel.

Spiel.Start~
```