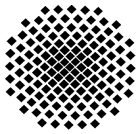
Universität Stuttgart
Fakultät 5, Institut FMI
EfidI 1, Professor Claus



WS 2002/2003
Bertele, Lewandowski
6. Februar 2003

Aufgabenblatt 15

ohne Besprechung

Aufgabe 1 Herbst 1999

Zeit: 1 Stunde

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

Abb. 1

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

Abb. 2

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

Abb. 3

Die Springer-Figur eines Schachspiels macht sogenannte Springer-Züge, siehe Abbildung 1. Von dem mit 0 bezeichneten Ausgangsfeld mit den Koordinaten (4,4) darf der Springer auf die mit 1 bezeichneten Felder springen. Alle anderen Felder sind nicht mit einem einzigen Zug erreichbar und daher mit einem Minuszeichen bezeichnet.

Zur Repräsentation der Erreichbarkeit der Felder des Schachbretts verwenden wir folgende Ada-95 Deklarationen:

```
type schachbrett is array(1..8,1..8)of Natural;
unbelegt: constant Natural:=999;
Feld: Schachbrett;
Startx, Starty: Integer;
maxZuege: Natural range 0..9;
```

Der Zahlenwert von `Feld(x,y)` soll dabei angeben, wieviele Züge ein Springer mindestens braucht, um das betreffende Feld mit den Koordinaten (x,y) zu erreichen. Das Startfeld des Springers hat hierbei den Wert 0, weil der Springer sich bereits auf diesem Feld befindet und daher keinen Zug machen muß, um es zu erreichen.
Falls ein Feld überhaupt nicht erreichbar ist, dann ist der Wert `Unbelegt` an der betreffenden Koordinate eingetragen.

a) Schreiben Sie eine Prozedur `Init(brett: out Schachbrett)`, die alle Felder des übergebenen Schachbretts auf `Unbelegt` setzt.
b) Schreiben Sie eine Prozedur `Ausgabe(brett: in Schachbrett)`, die die Erreichbarkeit der Felder wie in den obigen Abbildungen gezeigt auf den Bildschirm ausgibt.
c) Schreiben Sie eine (rekursive) Prozedur `Zug(x, y: in Integer; zugnr: Natural)`, die versucht, einen `zugnr`-ten Zug des Springers auszuführen, sofern der Zug möglich ist. Wenn der Zug möglich ist, soll er in die globale Variable `Feld` eingetragen werden. Falls noch nicht `MaxZuege` Züge in Folge ausgeführt worden

sind, sollen weitere Züge probiert werden, bis alle Möglichkeiten erschöpft sind, Felder mit höchstens `MaxZuege` Zügen zu erreichen. Achtung! Manche Felder sind auf mehreren Wegen erreichbar. Es interessiert nur die *geringste* Anzahl von notwendigen Zügen! Die Abbildung 2 zeigt zum Beispiel, welche Felder mit höchstens 2 Zügen von (1,1) aus erreichbar sind, Abbildung 3 zeigt die Erreichbarkeit mittels 4 Zügen.

d) Schreiben Sie ein Hauptprogramm, das die Koordinaten des Startfeldes sowie `MaxZuege` einliest und ungültige Eingaben zurückweist, anschließend die Erreichbarkeit der Felder berechnet und schließlich ausgibt. Verwenden Sie dazu die obigen Prozeduren. Sie können diese Teilaufgabe unabhängig von den anderen lösen.

Aufgabe 2 Frühjahr 2001

Zeit: 1 Stunde. Bei den aktuellen Prüfungen entfällt die Wahl der Programmiersprache. Ada95 ist nun Pflicht. Endliche Automaten gehören für die Wirtschaftsinformatiker in diesem Frühjahr nicht zum Prüfungsstoff.

a) Zeichnen Sie einen endlichen Automaten (in Form eines Diagramms), der folgende Sprache erkennt:
   - Jedes Wort beginnt mit dem Zeichen *0*
   - Danach kommt eine beliebig lange Folge von Zeichen *0* oder *1*
   - Jedes Wort endet mit dem Zeichen *b*
   - Anders aufgebaute Wörter kommen in der Sprache nicht vor.

b) Zeichnen Sie für die gleiche Sprache wie in Aufgabe a) ein Syntaxdiagramm.

c) Gegeben ist folgende Grammatik: $G_1=(N,T,P,S)$ mit $N=\{S,A,B\}$, $T=\{a,b\}$, $P=\{S\rightarrow AB|BA,\ A\rightarrow AA|a,\ B\rightarrow Bb|\varepsilon\}$. Geben Sie die von G_1 erzeugte Sprache in einer Mengenschreibweise an.

d) Ändern Sie G_1 so zu einer Grammatik G_2 ab, daß zusätzlich das Wort *abba* in der Sprache vorhanden ist, jedoch keine weiteren neuen Wörter.

e) Geben Sie eine Grammatik G_3 an, die genau alle diejenigen Wörter von G_1 produziert, bei denen die Anzahl der *a*'s echt größer als die Anzahl der *b*'s ist.

f) Schreiben Sie ein Ada- oder Modula-2-Programm (freie Wahl der Programmiersprache), das folgende Aufgabe löst:
In einer Schleife werden solange natürliche Zahlen eingelesen, bis eine *0* eingegeben wurde. Die Zahlen werden dabei in eine einfach verkettete Liste eingetragen (die Eintragung darf rückwärts erfolgen). Anschließend wird mittels einer **rekursiven** Prozedur aus dieser Liste eine zweite Liste erstellt, in der jeweils die **Summen** bzw. **Zwischensummen** aller bisher durchlaufenen Werte, und zwar in der gleichen Reihenfolge, abgespeichert werden. Diese neue Liste soll erst rückwärts, dann vorwärts ausgegeben werden. Lösungen, die nicht auf einfach verketteten Listen basieren, werden nicht gewertet.

Aufgabe 3 Herbst 2001

Zeit: 1 Stunde

Das Betreiben von Windkraftanlagen ist ein einträgliches Geschäft, da die Abnahmepreise für Strom in den nächsten 20 Jahren garantiert sind. Der einzige Unsicherheitsfaktor ist die Stärke, mit der der Wind bläst.

Schreiben Sie ein Programm in Modula-2 oder Ada, das die Rentabilität einer Windkraftanlage über 20 Jahre hinweg berechnet.

Beim Start des Programmes werden folgende Daten vom Benutzer abgefragt:

- Kaufpreis (volle DM)
- Eigenkapital (ebenfalls volle DM)
- Jährliche Unterhalts- und Wartungskosten (volle DM)
- Zu erwartende jährliche Einnahmen (volle DM) durch die Stromerzeugung bei "normalem" Wind

Zur Finanzierung der Anlage müssen Sie einen Kredit aufnehmen, der im nächsten Jahr 5% Zinsen auf den Stand des Kredites am Ende dieses Jahres kostet. Die Windkraftanlage wird im Jahr 0 gebaut (liefert also noch keinen Strom und kostet auch noch keinen Unterhalt) und muß in diesem Jahr finanziert werden (die Zinsen dieses Jahres werden erst im Jahr 1 fällig). Der Kredit bleibt in den Jahren 0 bis 5 tilgungsfrei (d.h. es ist keine Rückzahlung notwendig) und wird dann in den Jahren 6 bis 15 komplett getilgt (d.h. zurückgezahlt). In den Jahren 1 bis 15 müssen also die Zinsen auf den Kreditstand zum Ende des Vorjahres, in den Jahren 6 bis 15 die Tilgung (Kreditsumme geteilt durch 10) bezahlt werden. In den Jahren 1 bis 20 muß die jährliche Abschreibung (Kaufpreis geteilt durch 20, d.h. lineare Abschreibung) bezahlt werden, ebenso die jährlichen Unterhalts- und Wartungskosten.

Die Rechnung soll ein Worst-Case-Szenario und ein Best-Case-Szenario umfassen. Beim Worst Case wird davon ausgegangen, dass der Wind in jedem Jahr um 10% schlechter bläst als "normal", die Stromeinnahmen also um 10% geringer sind als bei Programmstart eingegeben wurde. Beim Best Case sind die Stromeinnahmen dagegen um 10% höher als eingegeben.

Ihr Programm sollte eine Tabelle ausgeben, die folgende Spalten besitzt:

- Nummer des Jahres (läuft von 0 bis 20)
- Kreditstand am Ende dieses Jahres (in den Jahren 0 bis 5 ist das der Kaufpreis minus Eigenkapital, ab dem Jahr 6 geht die jährliche Tilgung davon ab, und im Jahr 15 muss dieser Wert zu 0 geworden sein bzw. wegen des wegfallenden Divionsrestes **fast** zu 0 geworden sein)
- Kumulierter Ertrag Worst-Case (hat im Jahr 0 den Wert 0; ansonsten berechnet es sich folgendermassen: Worst-Case-Stromeinnahmen minus Zinsen auf den Stand des Vorjahres minus Tilgung minus Abschreibung minus Wartungskosten plus Ertrag des Vorjahres)
- Kumulierter Ertrag Best-Case (gleiche Berechnungsmethode, nur mit Annahme der Best-Case-Stromeinnahmen)

Falls das Eigenkapital höher als der Kaufpreis sein sollte oder falls in einem Jahr der kumulierte Ertrag im Worst-Case negativ werden sollte, bricht das Programm sofort mit einer entsprechenden Fehlermeldung ab. Beachten Sie, daß der Ertrag eines einzelnen Jahres negativ sein kann, ohne daß deshalb der kumulierte Ertrag negativ werden muß.

Aufgabe 4 Frühjahr 2001

Zeit: 1 Stunde

Schreiben Sie ein Modula-2 oder Ada-Programm, das Ihnen die Verwaltung eines Aktiendepots mit einer einzigen Sorte von Aktien abnehmen soll. Bei Programmstart wird zuerst die Anzahl der Aktien eingelesen, die sich bereits im Depot befinden. Anschliessend werden in einer Schleife die Tageskurse (als ganze Zahlen in Pfennigen) eingelesen, wobei die Eingabe einer *0* das Programm beendet.
Falls ein eingegebener Kurs höher als der Vortageskurs ausfällt, gibt das Programm die Meldung "kaufen" aus, ansonsten "verkaufen". Die Anzahl der Aktien wird dadurch entweder

um 1 erhöht oder um 1 vermindert. Falls keine Aktien mehr vorhanden sind, bricht das Programm von selber ab.
Immer nach Ablauf von 5 Börsentagen wird ein wöchentlicher Rapport ausgegeben, das ist eine kleine Tabelle, in der für jeden der 5 Tage folgendes vermerkt ist:

- Anzahl der Aktien
- Wert einer Aktie
- Gesamtwert aller Aktien.

Falls der Gesamtwert aller gekauften Aktien um mehr als 30\% unter den Höchststand fällt (der je während des Programmlaufes erreicht wurde), gibt das Programm die Meldung "PANIK!! Alles verkaufen!!" aus und bricht ab.
Hinweis: Durch schlaue Wahl der Datenstrukturen läßt sich der wöchentliche Rapport recht einfach implementieren.
Bemerkung: ob die von Ihnen zu implementierende Kauf-und Verkauf-Strategie auf dem realen Börsenparkett sinnvoll wäre, spielt für die Lösung der Aufgabe keine Rolle.

Allgemeine Hinweise:

- Dieses Blatt wird nicht mehr besprochen oder korrigiert. Dieses Blatt dient allein Ihrer Klausurvorbereitung.
- Scheine gibt es ab dem 21. Februar 2003 bei Herrn Lewandowski.
- Bei weiteren Fragen, wenden Sie sich bitte an das Forum (http://fachschaft.informatik.uni-stuttgart.de/forum/), Ihren Tutor, oder per Mail direkt an J. Bertele (inf@studbs.informatik.uni-stuttgart.de).
- Weitere Hinweise finden Sie auf unserer Veranstaltungswebseite unter: http://www.informatik.uni-stuttgart.de/ifi/fk/lehre/ws02-03/info_I_0203.html