Universität Ulm - Abteilung Angewandte Informationsverarbeitung

12. Übungsblatt (30.01.02 bis 13.02.02) zur Vorlesung
Allgemeine Informatik I für Wirtschaftswissenschaftler und Biologen

WS 2001/2002

1. Alles eine Frage der Logik! (6 Punkte)

Gegeben sei ein *Oberon*-Programm mit der Vereinbarung

```
VAR x,y,z : BOOLEAN;
```

Vereinfachen Sie die folgenden Ausdrücke unter Verwendung der allgemeinen Rechenregeln (vgl. *Abschnitt 6.7.12* im Skript):

```
(a) x OR (y OR x) OR ~y
(b) (x OR y) & (x OR ~y)
(c) x OR y OR ~x
(d) (x OR y) & (x OR ~y) & (~x OR y) & (~x OR ~y)
(e) (x & y) OR (x & ~y) OR (~x & y) OR (~x & ~y)
(f) (x & y) OR ( (x & ~x) & z )
```

Stellen Sie Ihrem Tutor Ihre Lösungen vor und erklären Sie ihm die wichtigsten Schritte!

2. Hit The Middle! (14 Punkte)

In den letzen Wochen sind in der Vorlesung die wichtigen *Oberon*-Sprachelemente `ARRAY`, `RECORD` und `PROCEDURE` besprochen worden. Um deren Verwendung nochmals zu trainieren, sollen Sie in diesem Übungsblatt ein kleines Spiel programmieren: **Hit The Middle**.

Das Spielfeld ist quadratisch (ähnlich zu einem Schachbrett) aufgebaut und hat eine *ungerade Seitenlänge*. Auf diesem Spielfeld springt man mit Hilfe eines *Zufallsgenerators* hin und her und versucht, möglichst viel Geld zu gewinnen.

Der Spieler startet das Spiel mit **$10,000**. In jeder Runde wird durch den *Zufallsgenerator* bestimmt, auf welches Feld er springen muß. Es gibt die folgenden Gewinn- bzw. Verlustfelder:

- Das Mittelfeld bringt einen *Gewinn* von **$50,000**.
- Alle Diagonalfelder (außer der Mitte) bewirken einen *Gewinn* von **$2,500**.
- Alle Felder am Rand (außer den vier Ecken, die zählen zu den Diagonalen) führen zu einem *Verlust* von **$4,000**.
- Alle übrigen Felder sind *neutral*, d.h. es gibt dort keine Gewinne oder Verluste.

Bei einer Spielfeldbreite von 7 ergibt sich aus diesem Schema eine Aufteilung in ein Mittel-, 12 Diagonal- und 20 Randfelder. Somit ist das Spiel auch mathematisch *fair*, denn der erwartete Gewinn für jede Runde ist 0.

Es gibt drei Möglichkeiten, wie das Spiel enden kann:

1. Der Spieler trifft die Mitte. In diesem Fall ist das Spiel vorbei und der Spieler erhält $50,000 als Gewinn - zusätzlich zu seinem aktuellen Kontostand.
2. Der Spieler ist pleite. Kommt er auf ein Feld, das seinen Kontostand ins Negative bringen würde, wird sein Kontostand auf 0 gesetzt und das Spiel ist beendet.
3. Das Spiel dauert länger als die (über eine Konstante festgelegte) maximale Spieldauer. In diesem Fall erhält der Spieler seinen aktuellen Kontostand als Gewinn ausgezahlt.

Was die Datenstruktur in *Oberon* angeht, so empfehle ich Ihnen die folgenden Konstanten- und Typvereinbarungen:

```
CONST MaxGameLength = 10;            (* How long do you want to play? *)
      FieldSize = 7;                 (* This number must be odd! *)

TYPE FieldPosition = RECORD
                       x,y : INTEGER;
                     END;
     SavedMoves = ARRAY MaxGameLength OF FieldPosition;
```

Die jeweils aktuelle Spielposition wird in einer Variable vom Typ `FieldPosition` abgelegt; außerdem sollen alle Positionen des Spiels in einem `ARRAY` vom Typ `SavedMoves` gespeichert und am Ende ausgegeben werden.

Lagern Sie möglichst viele Funktionen und Operationen in *Prozeduren* aus. Das Hauptprogramm sollte am Ende nur noch zur Steuerung des Spiels dienen und den Kontostand des Spielers verwalten.

Hier sind ein paar Anwendungs-Beispiele zu **Hit The Middle**.

Zum Schluß noch eine kleine ***Zusatzfrage***:
Spielen Sie Ihr Programm möglichst oft und notieren Sie sich, wie Ihre Spiele jeweils geendet haben. Was fällt Ihnen dabei auf? Widerspricht dies der (rechnerisch korrekten) Annahme, daß das Spiel *fair* ist?

Nützliche Hinweise:

- Sie sollten nach jeder Runde einen aktuellen Zwischenstand ausgeben. Ich empfehle Ihnen, das Spielfeld mit der jeweiligen Position grafisch auszugeben (siehe Anwendungs-Beispiele). Diese Ausgabe läßt sich mit zwei *geschachtelten Schleifen* erledigen, sie brauchen hierzu *kein* `ARRAY`.
- Um die Ausgabe etwas übersichtlicher zu gestalten, sollten Sie nach jeder Runde ein `Read.Ln`-Kommando einfügen. So kann man verhindern, daß das Programm einfach so "durchrasselt" und die Ausgaben können besser beobachtet werden.
- Achten Sie auf die Numerierung der Felder! Für den Benutzer (d.h. in der Ausgabe der Daten) sollen die Felder von `1` bis `FieldSize` numeriert sein, allerdings sollten Sie intern auf die in *Oberon* typische Numerierung von `0` bis `FieldSize-1` zurückgreifen. Das gleiche Problem tritt bei der Zählung der Runden auf, denn die Numerierung des `ARRAY`'s vom Typ `SavedMoves` beginnt natürlich bei 0.
- Bitte verwenden Sie keine globalen Variablen in Ihrem Programm. Außer den Konstanten sollen alle benötigten Werte als *Parameter* der entsprechenden Prozeduren übergeben werden. Entscheiden Sie sich hierbei jeweils für *CallByValue* bzw. *CallByReference*.
- Überlegen Sie sich für Ihr Programm eine *Struktur*, bevor Sie mit der konkreten

Implementierung beginnen. Dazu gehören insbesondere die Prozedurköpfe mit den entsprechenden Variablen-Deklarationen.

- Für die Bestimmung einer Zufallszahl brauchen Sie das *Oberon*-Modul `RandomGenerators`. Die folgdende *Funktionsprozedur* liefert Ihnen eine Zufallszahl zwischen 0 und max-1:

```
PROCEDURE GetRandomNumber(max : INTEGER) : INTEGER;
  (* Returns a random number between 0 and max-1 *)
BEGIN
RETURN (SHORT(RandomGenerators.Int32Val()) MOD max);
END GetRandomNumber;
```

Viel Erfolg!!!