Universität Ulm - Abteilung Angewandte Informationsverarbeitung

5. Übungsblatt (07.06.01 bis 21.06.01)
zur Vorlesung Allgemeine Informatik II für Wirtschaftswissenschaftler

SS 2001

Wie investiere ich eine halbe Million? ... Teil II (20 Punkte)

Erinnern Sie sich? Sie haben kürzlich bei Günther Jauch eine Million Mark abgeräumt und sind nun dabei, das Geld möglichst gut und gewinnbringend zu investieren. Deshalb möchten Sie für die Hälfte (500.000 DM) europäische Aktien kaufen. Ein Freund hat Ihnen die Gewichtung eines europätische Aktienindex zukommen lassen, den Sie nun zusammen mit einer von Ihnen festgelegten Strategie als Grundlage für Ihre Anlage-Entscheidungen verwenden.

Nachdem Sie auf dem letzten Übungsblatt an der *Auswahl* der Aktien für das Investment der 500.000DM gearbeitet haben, geht es dieses Mal nun um die Bestimmung konkreter *Geldbeträge* für die anstehenden Käufe. Ihr Oberon-Programm soll also - nach Eingabe einer entsprechenden Strategie (vgl. Blatt 4) - für alle ausgewählten Firmen einen Anlagebetrag bestimmen, den sie in die jeweiligen Aktien investieren sollen. Die Struktur Ihres alten Programms können Sie für diese Aufagbe durchaus wiederverwenden. Wer aber lieber die offizielle Beispiellösung von Blatt 4 verwenden möchte, findet sie hier.

Dieses Mal sollen Sie auf jeden Fall (in Anlehnung an die Vorlesung) Listen verwenden! Um nach der Ausführung Ihres Programms die konkreten Investitionen geliefert zu bekommen, schlage ich den folgenden Weg vor:

- Zuerst einmal lesen Sie alle Datensätze in ein ARRAY ein. Diese Aufgabe haben Sie ja bereits letzte Woche programmiert.
- Nun durchsuchen Sie das ARRAY anhand der jeweiligen Strategie nach Treffern. Alle Treffer speichern Sie dann in einer Liste ab. So könnten entsprechende Deklarationen aussehen:

```
List = POINTER TO ListRec;
ListRec = RECORD
            company : CompanyInfo;
            next : List;
          END;

PROCEDURE InitList(VAR l : List);
   (* Creates new list for companys and sets anchor to NIL *)

PROCEDURE Insert(info : CompanyInfo; VAR l : List);
   (* Inserts a company file at the end of the list *)
```

  Bitte bauen Sie die Liste so auf, daß ein neues Element immer am *Ende* eingefügt wird. So behalten Sie die ursprüngliche Sortierung der Daten (in diesem Fall alphabetisch).
- Als nächstes wird für jede ausgewählte Aktie ein konkreter Geldbetrag ausgerechnet, der (analog zur Indexstruktur) zu investieren ist. Mathematisch gesehen müssen Sie hierfür zuerst die Gewichtungen der einzelnen Firmen in Ihrer Liste addieren (*GesamtGewichtung* ist der Anteil am Index, der durch Ihre Strategie abgedeckt wird) und dann die folgende Formel auf jeden Datensatz anwenden:

  *Investition=(Gewichtung/GesamtGewichtung)*PortfolioWert*

  Die Kunst liegt nun darin, die Prozedur

```
PROCEDURE AdjustWeights(l : List);
   (* Adjusts the weights for a given list in order to cover 100% *)
```

  ohne Verwendung von Schleifen zu programmieren - und zwar als Rekursion. Sie können das Feld für die Gewichtung im RECORD zum Abspeichern des jeweiligen Investitions-Betrages verwenden.
- Wenn Sie bis hierhin gekommen sind, haben Sie eine entsprechend den Eingabedaten sortierte Liste mit bereits kalkulierten Investments. Allerdings macht es doch wesentlich mehr Sinn, nach den eigentlichen Geldbeträgen zu sortieren, die am Ende auch investiert werden. Die Prozedur

```
PROCEDURE SortAsc(l : List);
   (* Sorts the given list in ascending order *)
```

  tut genau dies: Sie sortiert nach dem RECORD-Feld, in dem die Ergebnisse der Prozedur `AdjustWeights()` gespeichert wurden. Die Sortierung soll *aufsteigend* erfolgen, d.h. das kleinste Element rückt an den Anfang der Liste.
  Auch bei dieser Prozedur läßt sich durch "geschicktes" Programmieren relativ viel Aufwand sparen. Ich empfehle die Verwendung eines *BubbleSort* Algorithmus - den kennen Sie ja schon von Blatt 1.
- Die einzige Sache, die Sie nun noch brauchen, ist eine Prozedur zur Ausgabe der Daten. Es ist zweckmäßig, die Daten in *umgekehrter Reihenfolge* auszugeben, denn so steht (nach erfolgter Sortierung) die höchste Investition auch oben in der Liste.

```
PROCEDURE PrintListRev(outfile : Streams.Stream; l : List);
   (* Prints the list elements to 'outfile' in reverse order *)
```

  Auch hier sollten Sie versuchen, die Prozedur als Rekursion zu programmieren.

Am Ende fügen Sie alle diese Komponenten dann noch zu einem Programm zusammen - und fertig ist Ihr Investment-Tool für europäische Aktien! Hier können Sie sich ein paar Anwendungs-Beispiele anschauen.

Nützliche Hinweise:

- Da Sie die meisten Funktionen von Blatt 4 auch für dieses Programm brauchen werden, sollten Sie mit Ihrer alten Lösung beginnen. Eventuell müssen Sie noch ein paar Kleinigkeiten an der Struktur verändern, wenn Sie die neuen Prozeduren einbauen.
- Die Beispiele aus der Vorlesung sollten ein guter Anhaltspunkt sein. Schauen Sie also ruhig mal in Ihr Skript, bevor Sie mit dem Programmieren beginnen.
- Ein kleiner Tipp fuer die `Insert()` Prozedur: Sie sollen dort eine Fallunterscheidung machen, ja nachdem ob Sie in eine leere Liste einfügen oder dort schon etwas drinsteht.
- Alternativ zu meinem Lösungsvorschlag können Sie natürlich auch die Liste *absteigend* sortieren und die Elemente dann *vorwärts* ausgeben.
- Wenn Sie eine Rekursion programmieren, lohnt sich meistens eine kleine "Skizze" auf einem

Notizblatt. Hacken Sie nicht sofort alle Ihre Ideen in den Rechner, denn gerade hier können kleine Fehler eine große Wirkung haben!

- **Zusatzaufgabe 1:** Sie können die Liste auch anders definieren, und zwar analog zum Beispiel aus der letzten Vorlesung (*QueueDemo.om*). Ändern Sie Ihr Programm entsprechend ab und entscheiden Sie hinterher, welche Programmierung Ihrer Meinung nach "besser" ist.
- **Zusatzaufgabe 2:** Bestimmt kann man die Ausgabe der Treffer schöner (und übersichtlicher) gestalten. Vielleicht gibt Ihnen Ihr Tutor ja ein paar Bonuspunkte für Ihre Kreativität...

Viel Erfolg!!!