Universität Ulm - Abteilung Angewandte Informationsverarbeitung
Prof. Dr. Franz Schweiggert und Hans Braxmeier, Sommersemester 2003

8. Übungsblatt zur Vorlesung Allgemeine Informatik II

Abgabetermin: Donnerstag, 10.07.2003

Aufgabe 1: Sternenkatalog *(10 Punkte)*

Ihr Astronomieprofessor ist ein leidenschaftlicher „Sternengucker“ und hat sich deshalb einen Katalog mit den Namen aller sichtbaren Himmelsobjekte besorgt! Leider ist es sehr mühsam, die Daten eines Sternes bzw. einer Galaxie unter mehreren Millionen Einträgen zu finden. Er bittet Sie daher um Hilfe und fragt nach einem Programm mit dem die Himmelsobjekte effizient (schnelle Suche nach einem Objekt) verwaltet werden können.

Sie überlegen nicht lange und schlagen ihm vor die Daten in einem Array zu speichern! Aber halt: Sie erinnern sich an Allgemeine Info I und wissen, daß das Durchsuchen eines Arrays seriell erfolgt und somit sehr lange dauern kann. Von wegen effizient...

Auf der Suche nach einem schnelleren Algorithmus fragen sie einen bekannten Freund der Ihnen das Wort „Hashing“ ins Ohr flüstert (ein aus der Informatik bekanntes Verfahren, mit dem sich große Datenbestände schnell durchsuchen lassen).

Doch wie war das gleich mit dem Hashing-Algorithmus...

Motivation und Eigenschaften

- Schnelles Suchen in großen Datenbeständen
- Die Operationen Einfügen und Suchen werden gut unterstützt
- Die Operation Löschen weniger gut (aber wann explodiert schon ein Stern???)

Hashorganisation

- Eine *Hashtabelle* fixer Länge enthält Datenobjekte oder Verweise auf Objekte
- Die *Hashfunktion* weist jedem Datenobjekt einen Wert zwischen 0 und n-1 zu.
- Die Hashfunktion sollte möglichst *surjektiv* die Daten auf die Tabelle abbilden („gute Streuung“)
- Die Länge der Tabelle sollte deshalb eine *Primzahl* sein!

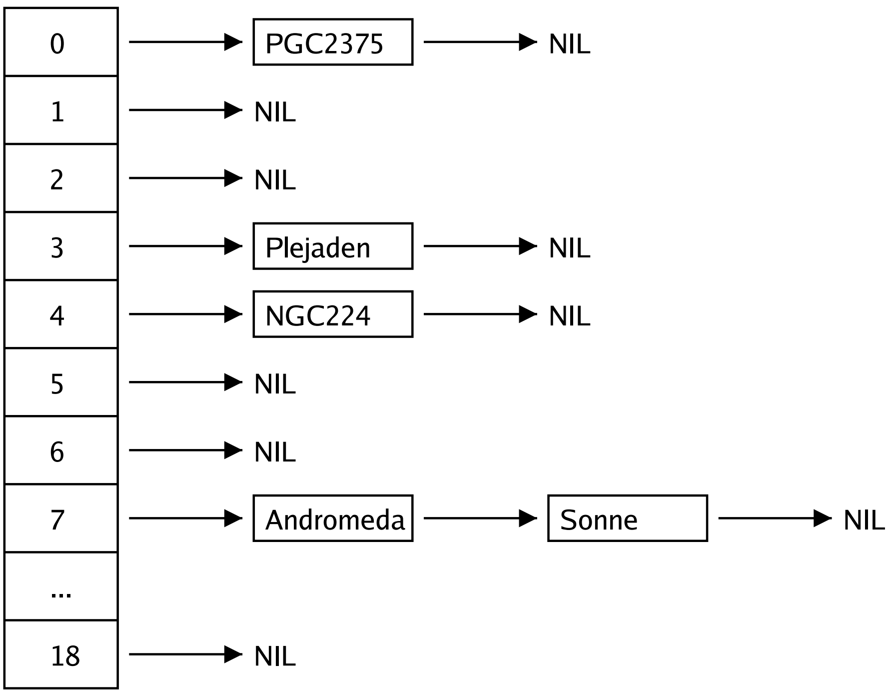
Datenstrukturen

- Datenobjekte: Records
- Hashtabelle: Array

Wie bekommen die Datenobjekte Ihren Tabellenplatz? Siehe Prozedure HashFkt:

```
PROCEDURE HashFkt(s: ARRAY OF CHAR): INTEGER;
VAR i, res: INTEGER;
BEGIN
  i := 0:
  res := 0;
  WHILE (s[i] # 0X) & (i < LEN(s)) DO
    res := res + ORD(s[i]) * (i+1) * (i+1);
    INC(i);
  END;
  RETURN res MOD prim; (* CONST prim = 19; *)
END HashFkt;
```

Wie wird das Array gefüllt? Beispiel:



Im Beispiel ist der Wert von $prim$ (siehe Prozedur $HashFkt$) $= 19$. Dadurch werden z.B. die „Plejaden“ auf Platz 3 abgebildet, „Sonne“ und „Andromeda“ landen auf Platz 7. Bei Doppelbelegung schafft eine verzeigerte Überlaufliste abhilfe, in der „gleichplazierte“ Himmelsobjekte (sortiert) eingetragen werden.

Was ist zu tun? Vervollständigen Sie das Programm unter Zuhilfenahme folgender Programmfragmente:

```
MODULE Sterne;

    IMPORT Read, Write;
    CONST prim = 19;
    TYPE Liste = POINTER TO Elements;
        Elements = RECORD
            content: ARRAY 30 OF CHAR;
            next: Liste;
        END;
        HashTable = ARRAY prim OF Liste;
    VAR counter: INTEGER;
        value: INTEGER;
        anchor: HashTable;

    PROCEDURE HashFkt (* siehe Code weiter oben *)

    (* Fuer jede moegliche Hashadresse eine Liste anlegen. *)
    PROCEDURE Init();
        VAR count: INTEGER;
    BEGIN
        count := 0;
        WHILE count < LEN(anchor) DO
            anchor[count] := NIL;
            INC(count);
        END;
    END Init;

    PROCEDURE Insert... (* zu implementieren *)

    PROCEDURE Ausgabe... (* zu implementieren *)

BEGIN

    Init();
    Insert("Sonne");
    Insert("PGC2375");
    ...
    Insert("Plejaden");
    Ausgabe(...);

END Sterne.
```

Viel Erfolg!