SE3

Aufgabenblatt 8

Softwareentwicklung 3: Logikprogrammierung - WS 2014/2015 - W. Menzel

Anwendungen der Listenverarbeitung (1)

Gesamtpunktzahl: 30 Abgabe der Lösungen bis zum 15.12.2014

Aufgabe 1: Listenverarbeitung (3) 14 Punkte

maximale Bearbeitungszeit: 60 Minuten

Zur digitalen Verarbeitung werden Signaldaten als Folge von Amplitudenwerten repräsentiert. Die Dateien `stimmhaft.pl` und `stimmlos.pl` enthalten jeweils fünf Beispiele für stimmhafte bzw. stimmlose Laute des Deutschen, die mit einer Abtastfrequenz von 16 kHz digitalisiert wurden, wobei das Signal als Liste von aufeinanderfolgenden Abstastwerten kodiert ist:

```
% sound(Laufende_Nr,Lautklasse,Signal)
% Laufende_Nr ist eine natuerliche Zahl, Lautklasse ein Name und
% Signal eine Liste von Abtastwerten (Gleitkommazahlen)

sound(1,f,[
-0.0016174316,
-0.020721436,
 ... ,
0.054901123,
0.013549805]).
```

1. Implementieren Sie ein Prädikat zur automatischen Anpassung des Signalpegels (Lautstärke), das alle Abtastwerte so verstärkt bzw. abschwächt, dass ein gegebener, maximal verfügbarer Wertebereich für die Kodierung der Abtastwerte optimal ausgenutzt wird.

2. Ein sehr einfach zu ermittelnder Signalparameter ist die Nulldurchgangsdichte, die als mittlere Anzahl von Kreuzungspunkten zwischen dem Signal und der Nulllinie pro Zeiteinheit definiert ist.

   Implementieren Sie ein Prädikat zur Ermittlung der Nulldurchgangsdichte (in Hz) für einen gegebenen Signalabschnitt.

3. Ermitteln Sie die Nulldurchgangsdichten für die Signale 1 bis 10. Analysieren Sie die Ergebnisse und überlegen Sie sich eine einfache Methode, mit der sich stimmhafte von stimmlosen Lauten unterscheiden lassen. Implementieren Sie auf dieser Grundlage ein Prädikat zur Klassifikation von Signalen nach ihrer Stimmhaftigkeit.

4. Klassifizieren Sie mit Hilfe dieses Prädikats die Signale 11 bis 18 in Datei `test.pl`.

Aufgabe 2: Listenverarbeitung (4) Hash-Suche 16 Punkte

maximale Bearbeitungszeit: 90 Minuten

Eine Hash-Tabelle bildet einen gegebenen Zugriffsschlüssel auf einen oder mehrere, diesem Schlüssel zugeordnete Werte ab. Um den Suchaufwand beim Einfügen und Auslesen von Schlüssel-Wert-Paaren zu minimieren, werden die Schlüssel über eine Hash-Funktion in Äquivalenzklassen eingeteilt, auf die mit Hilfe einer Hash-Funktion direkt zugegriffen werden kann. Dazu sollte die Hash-Funktion den gegebenen Schlüssel auf das Intervall natürlicher Zahlen von 1 bis N abbilden und so einen Index für die jeweilige Äquivalenzklasse bereit stellen.

Hash-Tabellen können z.B. als Index verwendet werden, der auf alle Vorkommen der jeweils relevanten Suchbegriffe in einer großen Textsammlung verweist.

1. *Präsenzaufgabe*: Definieren Sie zwei alternative Datenstrukturen für eine Hash-Tabelle. Achten Sie dabei besonders auf eine angemessene Behandlung von Kollisionen. Eine Kollision liegt vor, wenn die Hash-Funktion zwei unterschiedlichen Zugriffsschlüsseln die gleiche Äquivalenzklasse zuweist.

   Diskutieren Sie die Vor- und Nachteile der verschiedenen Repräsentationen.

2. *Präsenzaufgabe*: Entscheiden Sie sich für eine geeignet erscheinende Repräsentation und implementieren Sie dafür ein Prädikat, das eine leere Hash-Tabelle mit einer vorgegebenen Anzahl von Äquivalenzklassen (Buckets) erzeugt.

3. Implementieren Sie für die ausgewählte Repräsentation ein Prädikat, das aus einem neuen Schlüssel-Wert-Paar und einer gegebenen Hash-Tabelle eine neue Hash-Tabelle berechnet. Definieren Sie sich dazu eine geeignete Hash-Funktion.

   Hinweise:
   Einen Namen können Sie mit Hilfe des Prädikats `atom_codes/2` in eine Liste von Ascii-Werten zerlegen.

   Durch Definition des Prädikats `portray/1` z.B. als

   ```
   portray(X) :- write_term(X,[max_depth(100)]).
   ```

   können Sie bei Bedarf die Obergrenze für die Einbettungstiefe bei der Ausgabe von Listen verändern.

4. Implementieren Sie ein Prädikat, das die in der Datei `textindex.pl` gegebene Liste von Schlüssel-Wert-Paaren `[Wort,Position_im_Text]` in eine Hash-Tabelle umwandelt.

5. Implementieren Sie ein Prädikat, das für einen gegebenen Schlüsselwert den Wert aus der Hash-Tabelle ausliest.

6. Wie geht Ihre Implementierung mit mehrdeutigen Schlüssel-Wert-Zuweisungen (ein Wort kommt mehrfach im Text vor) um?

7. Implementieren Sie ein Prädikat, mit dem Sie für die in `textindex.pl` gegebenen Daten die minimale, die maximale und die mittlere Belegung der Äquivalenzklassen in Ihrer Hash-Tabelle berechnen können.

8. Finden Sie heraus, ob Sie mit unterschiedlichen Hash-Funktionen bzw. Tabellengrößen eine gleichmäßigere Verteilung von Schlüsseln und Werten auf die Äquivalenzklassen erreichen können.