Символы и их свойства.

Лекция 8 (часть 2).

Базы данных и обработка символьной информации.

Специальности : 230105, 010501

Пример : машинный словарь основ.

В качестве примера использования списков свойств символов для построения динамических БД рассмотрим представление в памяти машинного словаря основ для задачи морфологического анализа. Согласно приведенному в [2] описанию, для каждой основы в словаре приводится порядковый номер (десятиричное число), буквенный код основы, номер флективного класса и номер основоизменительного класса. Рассмотрим принципиальную возможность построения подобного словаря с применением имеющихся в muLisp е средств работы с файлами на внешних носителях и списками свойств. Каждой представленной в словаре основе мы поставим в соответствие символьный объект с именем, соответствующем буквенному коду основы по словарю (azot, balk, ball, bank1, bank2). Порядковый номер основы по словарю, номера флективных и основоизменительных классов будем рассматривать как свойства соответствующего символа, описываемые списком свойств. Для создания динамической базы данных в памяти таблицу основ мы представим как объект, имеющий в качестве свойств данные конкретных основ.

В newLISP-tk подобная динамическая БД может быть организована на основе ассоциативных списков.

Морфологическая классификация слов.

В основу построения алгоритмов морфологического анализа и синтеза слов положено разбиение всех слов на морфологические классы.

<u>Определение.</u> Морфологический класс определяет характер изменения буквенного состава форм слов.

Изменение форм слов может быть связано с изменением буквенного состава либо основы слова, либо его окончания.

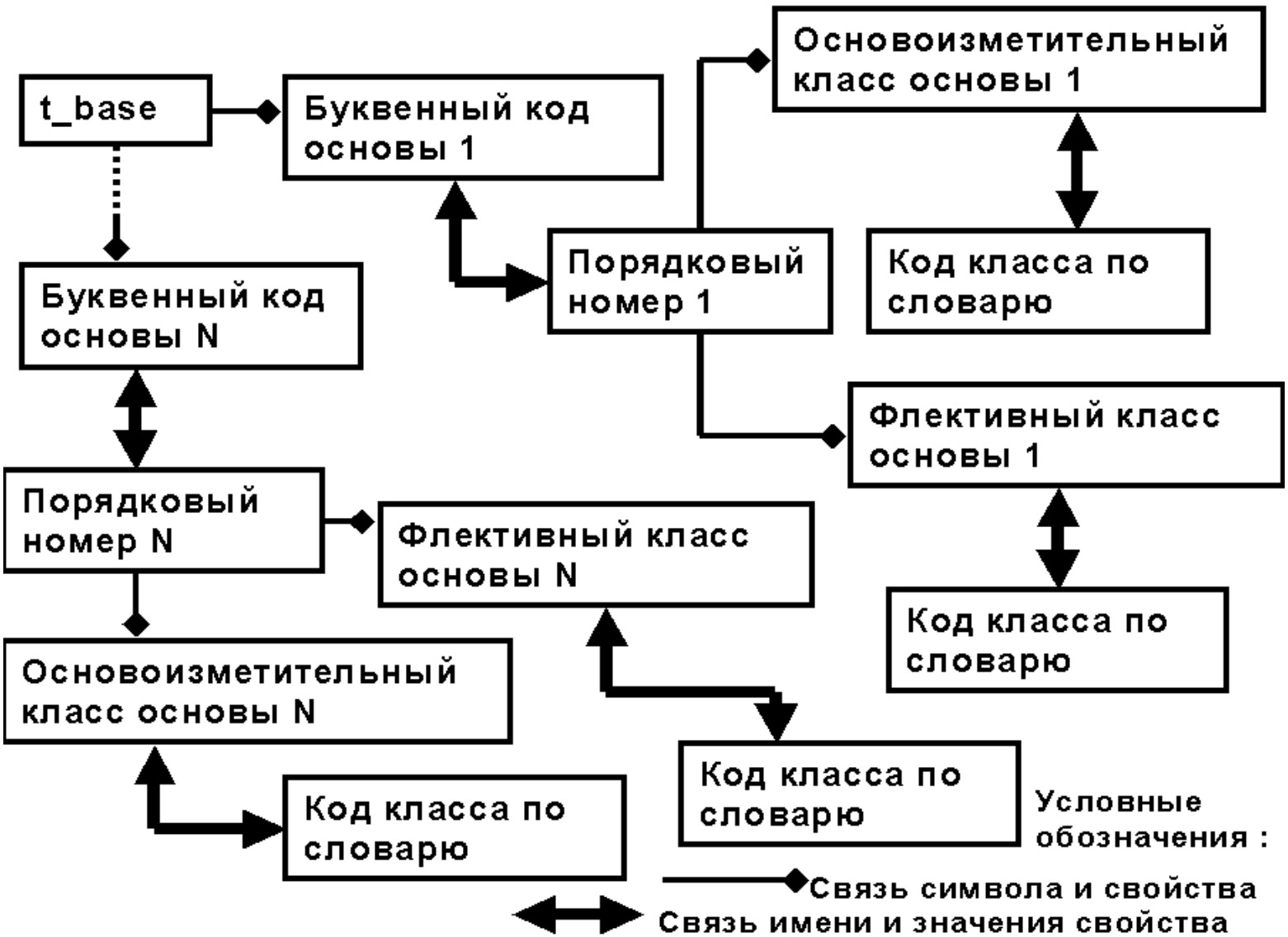
В силу вышесказанного морфологические классы слов принято подразделять на :

1). Основоизменительные классы, характеризующие систему изменения основ;

2). Флективные классы слов.

Флективные классы определены для изменяемых слов на основе анализа их синтаксической функции и систем падежных, личных и родовых окончаний. Флективный класс характеризуется либо системой признаков, либо словом-представителем.

Представление словаря основ с использованием списков свойств.



Этап 1 : считывание информации из внешних файлов.

В соответствие с особенностями ввода/вывода в Лиспе для облегчения последующего назначения свойств символам при формировании базы данных в памяти ЭВМ информацию словаря основ можно хранить в отдельных файлах : буквенных кодов основ (basesymb.txt), порядковых номеров основ (basenumb.txt), кодов основоизменительных классов (bchangcl.txt), кодов флективных классов (flect_cl.txt). Для записи компонент файлов мы создаем соответствующие списки. Рассмотрим пример для файла буквенных кодов основ.

```
(defun load_bases_symb_codes (file_name)
 (setq bases_symb_codes_list nil)
 (rds file_name)
 (loop
   ((not (listen)))
   (setq bases_symb_codes_list (cons (read-line) bases_symb_codes_list)))
 (rds) bases_symb_codes_list)
```

Вызов : (load_bases_symb_codes basesymb.txt)

Этап 2 : построение списков свойств.

В соответствии с выбранной структурой для представления информации словаря основ описание каждого из свойств с множественным значением будет представляться списком : (<имя свойства> (<значение 1> … <значение N>)). Так, для основоизменительных классов мы будем иметь следующее описание :

(basechange_class (<основоизменительный класс основы 1>. . . <основоизменительный класс основы N>)), для флективных классов :

(flect_class (<флективный класс основы 1 > . . . <флективный класс основы N>)).

На основе считанных из файлов списков кодов основоизменительных (basechange_class_list) и флективных классов (flect_class_list) формируем списочное описание свойств с множественным значением :

(list (list ‘basechange_class basechange_class_list) (list ‘flect_class flect_class_list))

Построение списков свойств (продолжение).

```
; Формирование БД с составной структурой свойств описываемых
;  объектов
(defun db_complex_props_make (db_name props vals vals_of_props_list)
; db_name - имя объекта
; props - список названий ключевых свойств объекта
; vals - список значений ключевых свойств объекта
; vals_of_props_list - списочное описание неключевых свойств объекта
;                    (<атрибут> <список значений атрибута>)
  ((null vals_of_props_list)(db_make db_name props vals))
  ((db_make db_name props vals)
        (complex_props_make vals vals_of_props_list))
)
```

Формирование БД в оперативной памяти.

```
; Функция формирования БД в оперативной памяти
(defun db_make (db_name props vals)
; Списки имен и занчений свойств имеют различную длину
 ((or
   (and (null props)(not (null vals)))
   (and (not (null props))(null vals))
  ) nil)
; Условие окончания рекурсии
 ((and (null (cdr props))(null (cdr vals)))
  (put db_name (car props)(car vals)))
 ((put db_name (car props)(car vals))
  (db_make db_name (cdr props)(cdr vals)))
)
```

Формирование структуры свойств.

```
(defun complex_props_make (vals vals_of_props_list)
 ((and (null (cdr vals))
       (not (null vals_of_props_list)))
  (complex_prop_make (car vals) vals_of_props_list))
 ((complex_prop_make (car vals) vals_of_props_list)
  (complex_props_make (cdr vals)(vals_of_props_make vals_of_props_list))
 ))
```

Функция complex_props_make с помощью вспомогательной функции complex_prop_make для каждого символа из списка vals ставит в соответствие свойства и их значения из списка vals_of_props_list. Каждый элемент списка vals_of_props_list соответствует описанию списочному описанию свойства с множественным значением (см. слайд 3). В содержательной интерпретации список vals содержит описание значений ключевых свойств объектов (в нашем случае – это порядковые номера основ по словарю). Вспомогательная функция vals_of_props_make для каждого очередного объекта с уже сформированным списком свойств удаляет описания значений его свойств из списка vals_of_props_list.

Формирование элементов списка свойств отдельного объекта.

```
(defun complex_prop_make (val vals_of_props_list)
 ((null (cdr vals_of_props_list))
  (put val (caar vals_of_props_list)
           (caadar vals_of_props_list)
  ))
 ((put val (caar vals_of_props_list)
           (caadar vals_of_props_list))
  (complex_prop_make val (cdr vals_of_props_list)))
)
```

Удаление описания значений свойств очередного объекта.

```
(defun vals_of_props_make (vals_of_props_list)
 ((null vals_of_props_list) nil)
 (cons (list (caar vals_of_props_list)(cdadar vals_of_props_list))
       (vals_of_props_make (cdr vals_of_props_list))
 ))
```

Полный текст программы формирования БД словаря основ в оперативной памяти приводится в файле basedict.lsp.

Тестовый пример.

После загрузки программы : mulisp.com basedict.lsp и последовательного вызова функций считывания информации из внешних файлов :

(load_bases_symb_codes basesymb.txt)

(load_bases_numbers basenumb.txt)

(load_basechange_classes bchangcl.txt)

(load_flect_classes flect_cl.txt)

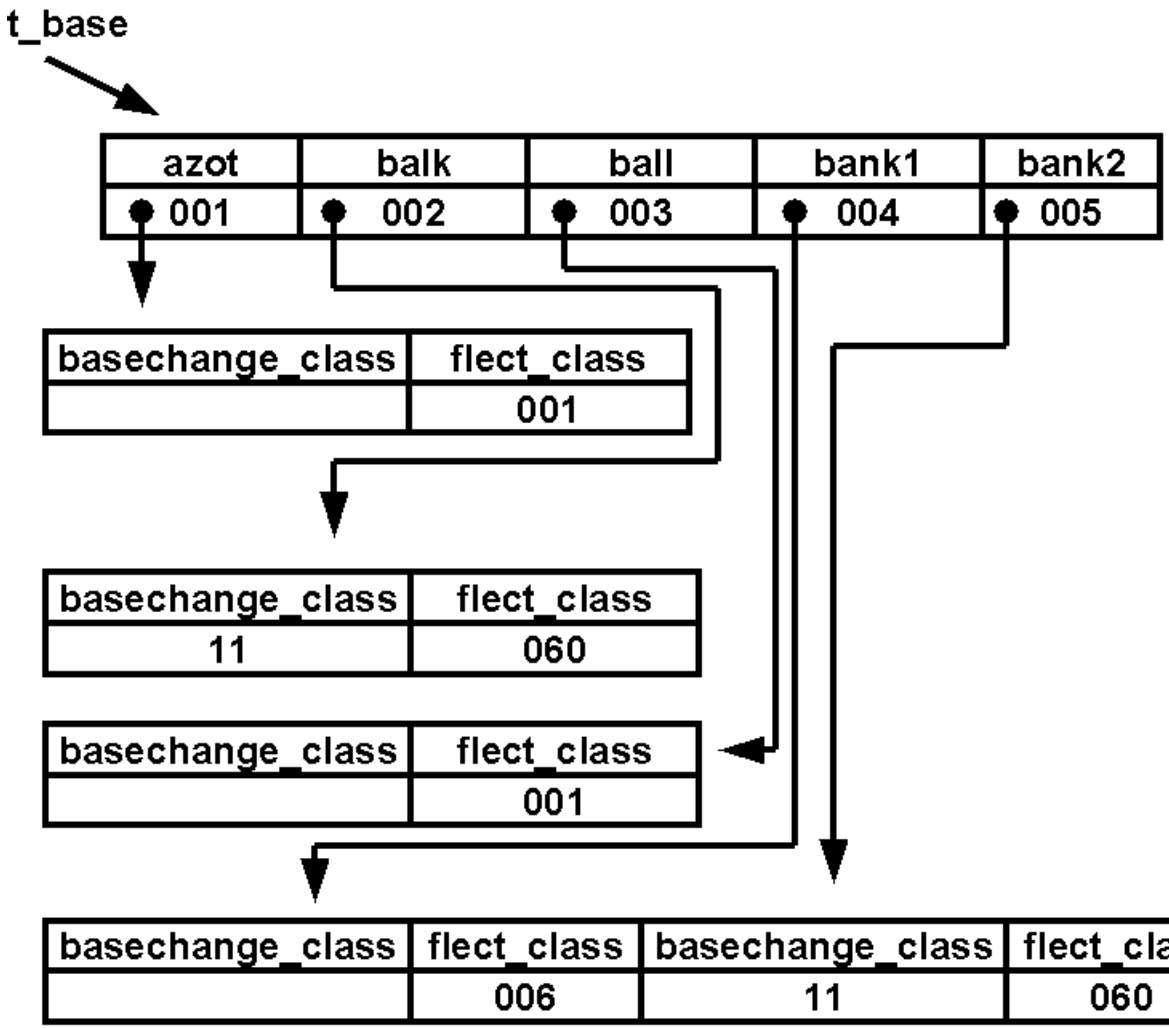
с помощью вызова функции test5 :

(test5 ‘t_base ‘basechange_class ‘flect_class) строится совокупность списков свойств в соответствии с приведенной на слайде 3 схемой.

Для просмотра свойств элементов созданной структуры достаточно вызвать функцию get из командной строки интерпретатора :

(get t_base “balk”) выдает номер основы по словарю основ, то есть “002”; (get “002” basechange_class) для основы с заданным номером выдает код основоизменительного класса, то есть “11”; (get “002” flect_class) для основы с заданным номером выдает трехразрядный восьмеричный код флективного класса, то есть “060”.

Сформированная БД для тестового примера.



Если слово имеет неизменяемую основу, то номер основоизменительного класса в словаре не указывается.

Применительно к табличной модели данных получаем, что имя символа соответствует либо названию таблицы (имени отношения), либо заглавию табличной строки, названия свойств – заглавиям столбцов, значения свойств – содержимому ячеек.

Строки как тип данных.

Строки относятся к простым типам данных. В Лиспе используется рассмотрение строк как одномерных массивов или векторов, элементами которых являются знаки. Многие более общие функции, определенные для массивов и последовательностей (чтение, сравнение элементов), наследуются строками.

Кроме этого, в Лиспе определен и ряд специальных функций для работы с этим типом данных :

- **Функции сравнения строк;**
- **Функции преобразования атомов и списков в строки и наоборот;**

Функции muLISP’a по работе со строками.

Функция (findstring <строка 1> <строка 2>) показывает, начиная с какой позиции в строке 2 содержится строка 1 в качестве подстроки. Пример : (findstring ‘tri ‘string metter) выдает в качестве результата 1.

Функция pack преобразует список символов в строку. Пример : (pack ‘(s t r l n g m e t t e r)) выдает в качестве результата строку STRINGMETTER.

Функция print-length выдает в качестве результата длину строки до первого пробела. Пример : (print-length ‘string metter) возвращает в качестве результата 6.

Функция (string-left-trim <строка 1> <строка 2>) в случае, когда строка 2 начинается строкой 1, возвращает в качестве результата подстроку строки 2 (до первого пробела) после вхождения строки 1. В противном случае возвращается пустая строка. Пример : (string-left-trim ‘at ‘atom and list) возвращает в качестве результата строку OM.

Функции muLISP’a по работе со строками.

Функция (char <строка> <n>) возвращает в качестве результата n-й символ строки, отсчет производится с 0. Пример : (char “cat Kuzya” 5) возвращает \u в качестве результата.

Функция (string= <строка 1> <строка 2>) возвращает результат сравнения строк, при этом строчные и заглавные буквы различаются.

Функция (substring <строка> <n> <m>) возвращает в качестве результата подстроку исходной строки, начиная с n-го и кончая m-м символом. Отсчет символов производится с 0. Аналогичная функция (slice <строка> <n> <m>) имеется в newLISP-tk, но аргумент <m> в ней – длина выделяемой подстроки.

Функция (unpack <строка>) преобразует строку в список символов, который возвращается в качестве результата. Пример : (unpack “Эксперимент на собаках”) возвращает в качестве результата список : (Э к с п е р и м е н т | | н а | | с о б а к а х). В newLISP-tk ей соответствует функция explode.

Литература.

1. Хювенен Э., Сеппянен Й. Мир Лиспа. Т.1. – М.:Мир, 1990. С. 61-63, 168-172 , 326-327, 337-340.

2. Белоногов Г.Г. и Богатырев В.И. Автоматизированные информационные системы. Под ред. К.В. Тараканова. – М.: Сов. радио, 1973.