## Реферат на тему:

### Примітивні об’єкти даних

***Примітивними об’єктами*** даних є ***символи***, ***числа*** та ***конси***. muLisp має безліч функцій розпізнання, порівняння, комбінування та обробки цих об’єктів. Це дозволяє конструювати будь-які складні об’єкти даних. Як було сказано раніше, muLisp має два типи даних: ***атоми*** та ***списки***. Атоми поділяються на символи та числа. Списки є підмножиною об’єктів, які мають більш загальну структуру — бінарне дерево. Вони створені за допомогою консів.   
  
***Символ*** є об’єктом даних, з яким пов’язано 4 атрибути, кожен з яких є вказівником на:  
— ***PRINT - ім’я***. Це унікальний рядок ASCII символів, за допомогою якого система ідентифікує символ при операціях введення-виведення. PRINT - ім’я не може бути змінене. Імена обмежені за розміром: вони повинні мати не більше від 65536 символів.  
— ***поточне значення***. Значенням символа може бути будь-який об’єкт даних, який зберігається в комірці пам’яті. Якщо в середовищі Ліспу ввести PRINT-ім‘я символу, то на виході буде його значення. Поточне значення доступно як CAR - елемент символа.  
— ***список властивостей***. Він містить значення властивостей символа, проіндексованих за ключем, його форма має вигляд: (ім’я1 значення1 ім’я2 значення2 ... ім’яN значенняN). При ініціалізації системи список властивостей є порожнім (дорівнює NIL). Його можна змінити за допомогою функцій властивостей та прапорців. Доступний як CDR - елемент символа.  
— ***визначення функції***. При створенні символу в muLisp цей атрибут дорівнює "функція невизначена". Визначення функції складається або за шаблонами машинної мови, або на D-коді. Значення цього атрибута можна отримати в результаті виконання функції флагів (GETD символ).  
  
SYMBOLP є функція, яка розпізнає символ. Вона повертає Т, якщо аргумент є символом і NIL в протилежному випадку.

$ (SYMBOLP ‘XYZ) $ (SYMBOLP 41)

T NIL

$ (SYMBOLP ‘(q w)) $ (SYMBOLP ‘())

NIL T

В Коммон Ліспі (файл common.lsp) визначені функції SYMBOL-VALUE, яка повертає значення символа, та функція SYMBOL-PLIST, яка повертає весь список властивостей символа.

(DEFUN SYMBOL-VALUE (SYM) (DEFUN SYMBOL-PLIST (SYM)

((SYMBOLP SYM) (CAR SYM) ) ) ((SYMBOLP SYM) (CDR SYM) ) )

Узагальненою функцією присвоєння є SETF, яка визначена в common.lsp. Вона заносить данні в комірку пам’яті символа: (SETF <комірка пам’яті> <значення>). Через функцію SETF можна представити описані раніше функції SET та SETQ.

(SETQ x y) це (SETF x y)

(SET x y) це (SETF (SYMBOL-VALUE x) y)

Проміжки, дужки, коми, одинарні та подвійні лапки, крапка, крапка з комою відіграють спеціальну роль в Ліспі. Одинарним Escape-символом є \. Багатократним Еscape-символом є |. Спеціальні літери можуть використовуватися у PRINT-іменах символів, але для цього перед ними треба ставити \, або весь рядок брати в |. Вирази |q w e| та |sym(bol| є символами. Для використання літер \ та | в символах необхідно ставити перед ними \. Якщо виводиться на екран символ, який містить спеціальні літери, то він виводиться з багатократним escape-символом. Програмна змінна \*PRINT-ESCAPE\* булевського типу відповідає за виведення escape-символів. Якщо вона дорівнює NIL, то escape-символи на екран не виводяться. Подвійні лапки " грають роль літери |. Розглянемо приклади (спочатку \*PRINT-ESCAPE\*=T):

$ (SETQ |sym(bol| 3) $ (SETQ a |q w e|) $ s\a $ s\\a

$ |sym(bol| $ a sa |s\\a||

3 |q w e|

$ (SETQ \*PRINT-ESCAPE\* NIL) $ (SETQ a |q w e|) $ (SETQ "s\\a" 2)

$ s\\a $ a $ |s\\a|

s\a q w e 2

***Число*** є іншим примітивним об’єктом. Воно може бути цілим або дробовим. Ціле число вводиться як послідовність цифр, перед якою може стояти знак мінус. За внутрішнім поданням цілі числа діляться на малі цілі (до 65536) та великі цілі. Оскільки значенням числа завжди є саме число, то немає необхідності перед ним ставити апостроф. Чотири атрибути характеризують число як об’єкт даних:  
— ***елемент тотожності***. Це є вказівник на саме число. Він доступний як CAR-елемент числа.  
— ***знак***. Він містить один з наступних символів, які характеризують тип числа:

додатне від’ємне $ (CDR 5.6) $ (CAR 5.6)

мале NIL T MACRO 5.6

велике LAMBDA NLAMBDA $ (CDR 1212) $ (CDR -121212)

дробове MACRO SPECIAL NIL NLAMBDA

Значення атрибута знака доступне як CDR-елемент числа.  
— ***довжина***. Якщо число є малим цілим, то цей атрибут містить значення цілого. Якщо число — велике ціле, то елемент ‘довжина’ містить довжину слова вектора числа. Якщо число дробове — елемент містить вказівник на його чисельник, який обов’язково повинен бути цілим (додатним або від’ємним).  
— ***вектор***. Якщо число мале ціле, то значення атрибута є вказівником на інше мале ціле (хеш-з’єднувач). Якщо число велике ціле, то це поле містить вказівник на найменший значущий байт. Якщо число дробове — елемент містить вказівник на його знаменник, який повинен бути додатним цілим числом.  
  
Функція порівняння EQL може використовуватися для порівняння чисел. Але більш загальною функцією для порівняння множини чисел є рівність:

$ (EQL -3 4) $ (EQL 4 4) $ (= 2 2 2) $ (= 2 2 3 2)

NIL T T NIL

Дробові числа можуть подаватися у десятковому вигляді та з дробовою рискою. Внутрішня змінна \*PRINT-POINT\* відповідає за тип виведення дробових чисел. Якщо вона дорівнює NIL, то всі дробові числа подаються на виведення з дробовою рискою. Якщо \*PRINT-POINT\* = n, то дробові числа виводяться з n знаками після десяткової коми. При введенні дробового числа воно автоматично скорочується.

$ 3/4 $ 3/9 $ 5/1 $ 12/9

3/4 1/3 5 4/3

Внутрішня змінна \*PRINT-BASE\* відповідає за основу системи числення, в якій обробляються числа. Якщо значення цієї змінної є цілим та перебуває в інтервалі від 2 до 32, то такою і буде основа системи числення, інакше muLisp працює в десятковій системі числення.

$ (SETQ ten 10) $ (SETQ \*PRINT-BASE\* 2) $ 234

$ (SETQ \*PRINT-BASE\* 16) $ ten 11101010

$ ten 1010

0A

Функцією, яка розпізнає цілі числа, є INTEGERP. Вона повертає Т, якщо її аргумент є цілим числом та NIL інакше. Функція NUMBERP розпізнає число.

$ (INTEGERP 100) $ (INTEGERP 3.5)

T NIL

$ (NUMBERP 3.5) $ (NUMBERP 4/5)

T T

Число в подвійних лапках завжди є символом:

$ (SYMBOLP "23") $ (NUMBERP "23")

T NIL

Символи та числа є атомами. Наступні вирази повертають істину: (ATOM 3.5), (ATOM "23"), (ATOM ‘APPLE).  
  
***Конс*** є примітивним об’єктом, який вказує на будь-які два інші об’єкти даних.. Він не є атомом. Назва конс пішла від функції конструктора CONS. Кожен конс склада- ється з CAR- та CDR- елементів. Конс часто називають ***точковою парою***. Якщо X і Y об’єкти даних, то вираз (X . Y) є консом, CAR-елемент якого є X, а CDR-елемент – Y.

$ (SETQ A (cons X Y)) $ (CAR A) $ (CDR A) $ (CDR ‘(R . S))

$ A X Y S

(X . Y)

За допомогою точкового подання можна показати структуру будь-якого об’єкту. Список (x1 x2 x3) є ланцюгом консів, які зв’язані за допомогою CDR- елементів. Його CAR- елементи вказують на елементи списку. CDR- елемент останнього конса вказує на NIL. Вказаний список можна подати у вигляді (x1 . (x2 . (x3 . NIL))). Функція READ читання виразу розпізнає як точкове подання виразу, так і спискове. Функція виведення PRINT виводить об’єкти в списковому поданні.

$ (SETQ a ‘(q . (w . nil)) $ a $ (CONSP ‘(q . w)) $ (CONSP (q w))

(q w) (q w) T T

Функція (CONSP obj) розпізнає конси. Список не є примітивним об’єктом, а є ланцюгом консів. Отже, результатом застосування функції CONSP до списку буде Т.

### Функції властивостей

Розглянемо, як можна працювати зі ***списком властивостей*** символа. Його можна по необхідності створювати, обробляти та видаляти. Властивості символа є глобальними, тобто доступними з довільної точки програми, поки вони не будуть явно змінені чи видалені. Використання символа в якості змінної чи імені функції не впливає на список властивостей.  
  
***Функції властивостей*** керують властивостями символів. CDR - елемент символа вказує на список властивостей. Разом з функціями флагів вони полегшують процес побудови динамічних баз даних.  
  
1. (PUT *символ ключ об’єкт*). У список властивостей <символа> кладеться значення *об’єкта* відповідно до вказівника *ключ*.

$ (PUT ‘capital ‘usa ‘washington) $ (SETQ capital ‘world)

$ (PUT ‘capital ‘germany ‘bonn) $ (PUT ‘world ‘ocean ‘atlantic)

$ (PUT ‘capital ‘england ‘london)

$ (CDR ‘capital)

((ENGLAND . LONDON) (GERMANY . BONN) (USA . WASHINGTON))

$ (CDR capital)

((ocean . atlantic))

$ (CAR ‘capital) $ capital

world world

2. (GET *символ ключ*). Повертає значення властивості, яке відповідає *символу* відповідно до вказівника *ключ*. Якщо такого вказівника не існує, то повертається NIL. Якщо змінна *capital* має властивості, які їй були надані у попередньому прикладі, то:

$ (GET ‘capital ‘england) $ (GET 'capital 'germany)

london bonn

3. (REMPROP *символ ключ*). Видалення зі списка властивостей *символа* властивості, яка відповідає *ключу*. Повертається старе значення властивості, якщо воно знайдено, та NIL – інакше. Нехай символ capital має три попередні властивості.

(REMPROP ‘capital ‘germany)

bonn

(REMPROP ‘capital ‘usa)

washington

(CDR ‘capital)

((england . london))

### Функції розпізнання

***Функції розпізнання*** — це твердження, які використовуються для розпізнання або ідентифікації об’єктів даних muLisp. Ці функції мають тільки один аргумент, а повертають булеве значення. Вони розпізнають об’єкт, який може мати довільну структуру. Ми вже розглянули деякі функції розпізнання: SYMBOLP, INTEGERP, NUMBERP, ATOM, LISTP, NULL. Розглянемо інші.  
  
(ZEROP *obj*). Повертає Т, якщо *obj* — число 0.  
(PLUSP *obj*). Повертає Т, якщо *obj* — додатне ціле число.  
(MINUSP *obj*). Повертає Т, якщо *obj* — від’ємне ціле число.  
(ODDP *obj*). Повертає Т, якщо *obj* — непарне ціле число.  
(EVENP *obj*). Повертає Т, якщо *obj* — парне ціле число.  
  
Функція (ASCII *sym*) повертає ASCII-код символа *sym*. Функція (ASCII *num*) повертає символ, ASCII код якого дорівнює числу *num*. Для того, щоб визначити, чи є символ *sym* літерою, можна використати функцію: (< (ASCII ‘a) (ASCII sym) (ASCII ‘z)). Оскільки muLisp не розрізняє малі та великі літери, то (ASCII ‘s) = (ASCII ‘S) для будь-якого символа s. Функція ISCHAR розпізнає літери. Для знаходження ASCII кодів символів, які позначають цифри, необхідно використовувати одинарний Escape-символ.

$ (DEFUN ISCHAR (char) $ (ASCII ‘f) $ (ASCII 70)

(<= (ASCII ‘a) (ASCII char) (ASCII ‘z)) ) 70 F

$ (ASCII ‘\9) $ (ASCII 57)

57 \9

Наступні функції дають можливість розпізнавати символи та числа.  
(ALPHA-CHAR-P *obj*) – повертає T, якщо *obj* – літера.  
(NUMERIC-CHAR-P *obj*) – повертає T, якщо *obj* – цифра.  
(ALPHANUMERICP *obj*) – повертає T, якщо *obj* – літера або цифра.

$ (ALPHA-CHAR-P W) $ (ALPHA-CHAR-P \3) $ (ALPHA-CHAR-P ~)

T NIL NIL

$ (NUMERIC-CHAR-P W) $ (NUMERIC-CHAR-P \3) $ (NUMERIC-CHAR-P ~)

NIL T NIL

$ (ALPHANUMERICP W) $ (ALPHANUMERICP \3) $ (ALPHANUMERICP ~)

T T NIL

Зазначимо, що символ проміжку (‘ ‘ ) є літерою.

## Завдання

1. Перевірити, чи складається список лише з:

|  |  |
| --- | --- |
| а) рівних чисел | г) додатних та від’ємних малих цілих, які чергуються |
| б) символів | через одне, причому їхня кількість парна |
| в) від’ємних дробових чисел | д) від’ємних парних цілих чисел |

2. За допомогою примітивних об’єктів даних створити:  
а) збалансоване бінарне дерево висоти 3, листки якого є символами.  
б) список з підсписками глибини вкладеності 3, елементами якого є числа.  
  
3. Написати функцію, яка за вхідним списком будує необхідну структуру:

а) ‘(q w e r t y) —> (q (w (e (r (t (y))))))

б) ‘(q w e r t y —> ((((((y) t) r) e) w) q)

в) ‘(q w e r t y) —> ((q) (w) (e) (r) (t) (y))

г) ‘(q w e r t y) —> ((q w) (e r) (t y))

4. Написати функцію, яка за списком який дано в завданні 3 в другому стовпчику, будує лінійний список.