Реферат на тему:

**Інтерпретатор muLisp**

**1. 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.

Кожний об’єкт даних конкретного типу складається із фіксованої кількості елементів - вказівників. Ці елементи можуть вказувати на інші об’єкти даних. Множина усіх об’єктів даних утворює зв’язану мережу вказівників, яка називається ***область даних*** MuLisp.

***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 ніж 65536 символів. Жодні два символи не можуть мати однакових PRINT -імен. Коли PRINT - ім’я генерується або читається, спрацьовує алгоритм хешування, який визначає існування атома з таким іменем. Якщо такий атом існує, то саме він і використовується, інакше – утворюється новий атом з новим PRINT - ім’ям.

— 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. 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 **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.

$ ѕ $ 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.

**2. Керування пам’яттю**

Динамічне автоматичне керування пам’яттю надає велику кількість переваг інтерпретатору muLisp. Немає необхідності власноручно програмісту розподіляти пам’ять під задачу, яка буде виконуватися. Пам’ять, яка не буде використовуватися програмою, доступна для створення нових структур даних.

При ініціалізації muLisp обчислюється розмір доступної пам’яті, яка потім розбивається на 4 області:

– область атомів (64К), яка забезпечує пам’ять для 4 елементів-вказівників, необхідних для кожного символа та числа.

– область векторів (128К), яка забезпечує пам’ять для кожного тіла PRINT-імені символа (64К) та числового двійкового вектора кожного числа (64К).

– область вказівників (256К), яка забезпечує пам’ять під 2 елементи-вказівники, необхідні для кожного cons-а та під D-код, необхідний для визначення функції. Оскільки cons є основною структурою даних Ліспу, область вказівників є найбільшою серед інших.

– область стеку (64К), яка забезпечує пам’ять для контрольного стеку та змінного стеку. Ці два стеки розташовані на протилежних кінцях області стеків.

Таким чином для роботи інтерпретатора muLisp необхідно 512К плюс пам’ять під DOS.

**3. Збір сміття**

MuLisp має алгоритм збору сміття з двома переглядами (помітка та чистка). Під час першого перегляду пам’яті помічаються усі активні об’єкти даних, доступ до яких забезпечується внаслідок зчеплення за допомогою елементів-вказівників, починаючи з елементів списку значень та властивостей усіх символів системи, або зі стеку змінних, або D-коду. Символи з автоматичним посиланням, які не мають властивостей та поточних визначень функцій, не помічаються. Такі символи автоматично видаляються зі списку під час другого перегляду.

У процесі другого перегляду збору сміття усі помічені об’єкти даних ущільнюються та збираються в одному з кінців відповідної області даних. Це дозволяє зберегти залишки областей даних для створення нових об’єктів.

**3.4. Перерозподіл областей даних**

Після збирання сміття одній або декільком з чотирьох областей даних може бракувати вільної пам’яті для того, щоб програми мали змогу продовжити виконання, незважаючи на те, що інші області даних мають достатню кількість вільної пам’яті. Якщо виникає така ситуація, то здійснюється перерозподіл областей даних шляхом ділення областей, додання пам’яті, якої не вистачає, одній або декільком областям. Але обмеження на розміри для кожної області даних, які описувались вище, мають бути дотриманими.

Отже, muLISP може реагувати на зміни вимог програм до розміру областей даних.

Хоча збір сміття та перерозподіл областей даних відбуваються автоматично, їхня поява не проходить непомітно для користувача, оскільки вони викликають коротку паузув роботі програм.

Точна сума часу для збирання сміття й перерозподілу залежить від кількості даних в системі. Збір сміття звичайно займає менше секунди. Точно так же, менше секунди звичайно відбувається й перерозподіл областей даних. В дійсності, це не повинно турбувати користувача, але при розробці систем реального часу, що використовують muLISP, це питання необхідно розглядати.

Явище, відоме як "thrashing" виникає в тому разі, коли система змушена витрачати непередбачену кількість часу на збір сміття для дуже маленького повернення області даних. Ознакою "thrashing" є значне зростання часу виконання даної задачі. Дана проблема може бути вирішена шляхом збільшення розміру пам’яті ЕОМ (до 512К) і (або) модификації програми з метою зменшення її вимог до пам’яті.

**5. Пакети переривань**

Пакети переривань muLISP викликаються регулювальником переривань та регулювальником затримки помилок. Коли переривання виникає, то після повідомлення про переривання чи про помилку на екран дисплея видається підказка у вигляді опцій:

Continue, Break, Abort, Top-level, Restart, System?

Потім система очікує, допоки користувач обере одну з опцій шляхом вказання її імені (С,В,А,Т,R чи S відповідно).

Відзначимо, що опції перераховані в порядку посилення їхньої дії.

- Continue (продовжити): повертає керування програмі, що викликала переривання. Якщо причиною переривання була команда переривання, послана з клавіатури, то виконання продовжується, ніби переривання не було.

Якщо переривання відбулося в результаті затримки помилки, величина, передана при перериванні регулювальником помилок, повертається як значення помилкової функції;

- Break (зупинка): тимчасово призупиняє виконання програми й виходить на наступний нижній рівень циклу "read-eval-print" ("читання-обчислення-друк"). Це дозволяє користувачеві перевірити або (і) змінити поточне середовище muLISP перед продовженням виконання програми. Для виходу з зупинки й відновлення роботи програми наберіть ( RETURN ) після знаку долара;

- Abort (переривання): перериває виконання програми, присвоює формальним параметрам, розміщеним в стеку змінних, початкові значення й повертає керування на поточний рівень циклу "read-eval-print". Визначення функцій, значення властивостей та глобальних змінних залишаються незмінними;

- Тop-level (верхній рівень): перериває виконання програми, присвоює початкові значення формальним параметрам, розташованим в стеку змінних, висвічує на консоль поточні вхідні й вихідні дані (CIS та COS) й повертає керування верхньому рівневі циклу "read-eval-print". Визначення функцій, значення властивостей та глобальних змінних залишаються незмінними;

- Restart (повторний старт): закриває всі відкриті файли, відмовляється від поточного середовища muLISP та ініціює нову систему muLISP. Всі зв’язки між змінними, функції та значення властивостей в поточному середовищі muLISP руйнуються;

- Sуstem (система): закриває всі відкриті файли, завершує виконання muLISP та повертає керування керівній ОС.

**6. Система переривань з консолі**

У будь-який час у ході виконання програми ініційована користувачем система переривань з консолі може зупинити виконання програми й поверне керування на консоль.

Переривання з консолі ініціюється шляхом натиснення клавіші <ESC> на клавіатурі консолі. Якщо на клавіатурі немає клавіші <ESC>, то символ переривання може бути зґенеровано шляхом натиснення клавіші лівої дужки ([) з одночасним натисненням клавіші <CTRL>.

Якщо ж і так не виходить, то символ для генерації переривань з консолі може бути змінений шляхом модифікації Default Readtable основної сторінки muLISP.

При виникненні переривання з консолі на екрані консолі висвічується повідомлення:

Console Interrupt Break: NIL

За ним на наступному рядку з’являється підказка у вигляді опцій переривання. Користувач може потім вирішити, як продовжити роботу.

Якщо з деяких причин немає відповіді на переривання з клавіатури, можливо здійснити переривання в системі шляхом переключення ЕОМ (хоча це завжди небажано).

**7. Повідомлення про помилки**

В даному розділі приводяться повідомлення про помилки в системі muLISP,а також опції, що є в розпорядженні користувача при появі помилок.

Коли muLISP виявляє помилковий стан, викликається функція BREAK. BREAK видає відповідне повідомлення про помилку, призупиняє виконання програми та забезпечує користувачеві опції продовження роботи на вибір.

Нижче в алфавітному порядку наведено повідомлення про помилки muLISP:

- DISK FULL (диск повний) : означає, що пам’яті для розміщення даних, записаних на дисковому файлі, бракує. Виконання програми припиняється, й виникає переривання за помилкою. Оскільки файл залишається відкритим, є можливість стерти й інші файли на всій дискеті (за допомогою функції EXETUTE ) та продовжити запис до файлу;

- END-OF-FILE (кінець файлу): означає, що здійснена спроба зчитати дані за межами кінця вхідного файлу ( CIF ) або з його порожніх місць.

Відразу за повідомленням "end-of-file" висвічується ім’я CIF у вигляді списку типу: "drive:name.type";

- FILE NOT FOUND (файл не знайдено) : означає, що вихідний та (або) SYS-файл, вказаний у командах ОС, що ініціюють muLISP, не знайдено, або SYS-файл невірної версії. SYS-файл може бути завантажений тільки під керуванням тієї версії muLISP, що використовується для зберігання файлу.

Вихідні та SYS-файли, крім того, можуть бути завантажені в muLISP з використанням команд RDS та LOAD відповідно. Коли одна з цих команд завершується, а файл не знайдено, замість повідомлення "file not found" команда повертає ознаку NIL;

- INSUFFICIENT ARGUMENTS (брак аргументів) : означає, що функція, яка потребує щонайменше один аргумент, викликається без аргументів. Функціями, які можуть викликати цей тип помилки, є: MAX, MIN, -, /, ADD1, SUB1, LCM, ABS, SIGNUM, NUMERATOR, DENOMINATOR, FLOOR, CEILING, TRUNCATE, ROUND, MJD, REM, DIVIDE, LOGNOT, BITLENGTH та SHIFT;

- INSUFFICIENT MEMORY, ABORTING (брак пам’яті, переривання): означає, що має місце нестача пам’яті для завантаження й функціонування середовища muLISP. Робота muLISP призупиняється, керування повертається до керівної ОС.

Відзначимо, що середовище muLISP, що зберігається в SYS-файлі, може бути завантажене до ЕОМ, що має менший об’єм пам’яті, ніж ЕОМ, на якій це середовище було створене. Помилка за браком пам’яті виникає тільки тоді, коли ЕОМ, на якій SYS-файл був завантажений, не володіє достатнім об’ємом пам’яті для розміщення середосища muLISP. Єдиний шлях завантаження SYS-файлів - це отримання більшого об’єму пам’яті для ЕОМ.

- MEMORY FULL (пам’ять заповнена) : означає, що пам’яті для продовження виконання програм muLISP не вистачає. Виконання програм призупиняється, виникає переривання за помилкою.

Дійсно, система керування пам’яттю забезпечує достатньою кількістю пам’яті кожну область даних для повного задоволення потреб програм muLISP. Якщо потреба в об’ємі пам’яті для розміщення об’єктів даних перевищує всі наявні ресурси, виникає ця помилка. Разом з повідомленням про помилку висвічується статистика в наступній формі:

GC: nnnn aaaa/aaaa vvvv/vvvv pppp/pppp ssss/ssss tttt/tttt

Шістнадцяткові цифри, що йдуть за "GC:", показують розмір пам’яті, що залищилася в кожній з основних 4-х областей даних. Отже, може бути визначена область даних, пов’язана з помилкою;

- NONINTEGER ARGUMENT (нецілий аргумент) : означає, що функція, яка потребує цілі аргументи, викликана з нецілим аргументом. Функції, для яких ця помилка може зустрітися, це: LOGAND, LOGIOR, LOGXOR, LOGNOT, SHIFT та BITLENGTH;

- NONINTEGER ARGUMENT (нечисловий аргумент) : означає, що функція, яка потребує числові аргументи, викликана з нечисловим аргументом. Така помилка може виникнути для наступних функцій: =, /=, <, >, <=, >=, MAX, MIN, +, -, \*, /, ADD1, SUB1, INCQ, DECQ, GCD, LCM, ABC, SIGNUM, NUMERATOR, DENOMINATOR, FLOOR, CEILING, TRUNCATE, ROUND, MOD, REM та DIVIDE;

- NONSYMBOLIC ARGUMЕNT (несимвольний аргумент) : означає, що функція, яка потребує символьні аргументи, викликана з несимвольним аргументом. До таких функцій відносяться: SET, SETQ, PSETQ, POP, PUSH, INCQ та DECQ;

- SYNTAX ERROR (синтаксична помилка) : означає, що функція READ зустріла або зайві праві дужки, або неточність у точковому зображенні, наприклад, (A.) чи (AB.CD). Оскільки переривання за даною помилкою енерується макросом правих дужок або ком, воно може бути модифіковане користувачем-проектувальником;

- UNDEFINED FUNCTION (невизначена функція) : означає, що здійснено спробу використання символа, що не має означення функції. Загальними діями при появі цієї помилки є: вибір опції BREAK, означення невизначеного символа та продовження вихідної програми за допомогою команди: ( RETURN ( EVAL BREAK ))

- ZERO DIVIDE (ділення на 0) : означає, що була викликана функція ділення з нульовим дільником. Такими функціями можуть бути: /, FLOOR, CEILING, TRUNCATE, ROUND, MOD, REM та DIVIDE.