Средства императивного программирования в Лиспе.

Лекция 9.

Специальности : 230105, 010501

Управление потоками.

В Common Лиспе, muLISPe ввод и вывод осуществляется независимо от конфигурации внешних устройств через потоки. Понятие потока в Лиспе сходно с аналогичным в C++ : потоки представляют собой логические каналы, из которых можно читать (поток ввода) или в которые можно писать (поток вывода) данные. Стандартным входным потоком (по умолчанию) считается поток из клавиатуры в программу, стандартным выходным потоком считается поток из программы на дисплей.

Для управления потоками в muLISPe имеются встроенные функции переключения потоков :

- Входного потока : (rds <устройство>)

- Выходного потока : (wrs <устройство>)

Устройством может быть клавиатура, дисплей, внешний файл, принтер. В случае использования внешнего файла нужно указать его имя (rds help.lsp) Переключение потоков на стандартные : (rds) – переключение входного потока на клавиатуру, (wrs) – переключение выходного потока на дисплей.

Работа с файлами на внешних носителях в muLISP’e.

Для чтения данных из входного потока в muLISP’e существует ряд функций, побочным эффектом которых является ввод s-выражений. Функции, применяемые для получения побочных эффектов, принято называть псевдофункциями. Основной функцией ввода является READ, которая читает s-выражение из входного потока и возвращает в качестве значения само это выражение. S-выражение может быть любым. Специализированные функции чтения (READ-CHAR, READ-LINE, READ-BYTE) позволяют читать из входного потока данные определенного типа. Если прочтенное значение необходимо сохранить, то вызов READ должен быть аргументом какой-нибудь формы, например, присваивания SETQ.

Функций типа EOF в Лиспе не существует. Для обозначения признака конца файла используется функция LISTEN. Вызов (listen) дает T, если во входном потоке есть хотя бы один символ.

Для идентификации входного и выходного потока в Лиспе существуют предикативные функции (inputfile *имя_файла*) и (outputfile *имя_файла*). Эти функции возвращают истинностное значение в том случае, когда имя входного/выходного потока совпадает с именем файла.

Порядок чтения из файла в muLISP’e.

- Подготовить структуру для записи компонент файла;
- Переключить входной поток на внешний файл (rds file_name);
- Чтение из файла в цикле с признаком конца файла в качестве условия завершения ((not (listen)));
- Обратное переключение входного потока на устройство ввода по умолчанию (rds);
- Передать полученное в цикле значение s-выражения в программу.

Чтение из файла в muLISP’е:пример.

```
(defun load_my_data (file_name)
 (setq my_data nil)
 (rds file_name)
 (loop
    ((not (listen)))
    (setq my_data (cons (read) my_data)))
 (rds) my_data)
```

Работа с потоками в Коммон Лиспе.

Для работы с потоком в Коммон Лиспе используются функции open и close, по назначению аналогичные одноименным функциям в newLISP-tk.

Пример : чтение из файла

```
(defun load_my_data (file_name)
 (setq my_data nil)
 (setq stream (open file_name :direction :input))
 (loop
    (if (not (listen stream))
       (return my_data)
    )
    (setq my_data (cons (read stream) my_data)))
 (close stream))
```

Пример : вызов (load_my_data "h:/test_file/test.dat") в качестве побочного эффекта дает формирование в памяти списка my_data с содержимым, считанным из файла "h:/test_file/test.dat".

Открытие и закрытие файла в Коммон Лиспе.

Вызов функции open имеет следующий синтаксис :

(open <имя файла>:direction :<значение>)

Допускаются следующие значения для параметра direction :

input – открыть поток для ввода;

output – открыть поток для вывода;

io – открыть двусторонний поток.

Потоковый объект, получаемый в результате вызова open, рекомендуется связать посредством setq с некоторым символом для последующего использования.

Закрытие файла осуществляется вызовом функции close :

(close <имя файла>)

Вывод информации в выходной поток.

В muLISP имеется ряд функций вывода информации в выходной поток.

Функция (print <s-выражение>) является псевдофункцией, значением которой является значение s-выражения, а побочным эффектом – вывод этого значения в выходной поток. Порядок выполнения :

1). Вычисляется s-выражение.

2). Полученное значение записывается в выходной поток.

3). Полученное значение возвращается в виде значения функции.

4). Осуществляется перевод курсора на новую строку.

Пример : (+ (print 2) 3) дает в качестве побочного эффекта вывод на экран числа 2, а в качестве значения – число 5.

Для последовательного вывода на одну строку более одного выражения следует использовать функции prin1 и princ. Действие аналогично print, отличие – в отсутствии перевода строки. Функция prin1 выводит в выходной поток значение с ограничивающими вертикальными скобками, princ – без. Рекомендуется prin1 использовать при записи в файл, а princ – для вывода на экран.

Нежелательного явления печати дважды можно избежать при использовании функций print, prin1 и princ на самом верхнем уровне.

Реализация функции print в newLISP-tk.

Специальных функций наподобие prin1 и princ в newLISP-tk нет. Их отсутствие компенсируется использованием управляющих символов в составе выводимой строки :

\n символ перевода строки (ASCII 10)

\r символ возврата каретки (ASCII 13)

\t символ табуляции (ASCII 9)

\nnn иной десятичный ASCII код в диапазоне от 000 до 255

В целом вычисление функции print выглядит аналогично тому, как это происходит в muLISP’e.

Другие функции вывода.

Вывод выражений и знаков часто желательно разбить на несколько строк. Перевод строки в muLISP'е можно осуществить специально предназначенной для этого функцией terpri, либо вызывая ее без аргументов (однократный перевод), либо с некоторым натуральным числом в качестве аргумента (вставка нескольких пустых строк).

Помимо print, princ и prin1, для вывода в выходной поток данных определенного типа в muLISP'е служат специализированные функции вывода write-line, write-string и write-byte. В отличие от print, princ и prin1, эти функции не вычисляют своего аргумента.

Функцию write-byte с числом 26 в качестве аргумента в muLISP'е рекомендуется использовать для обозначения конца файла.

Аналогом write-byte в newLISP-tk является функция write-char:

(write <№ устройства ввода/вывода> <выводимый байт>),

которая выводит значение байтового типа в файл. Значение <№ *устройства ввода/вывода*> может быть получено, например, вызовом функции *open*.

Работа с файлами в newLISP-tk.

Для работы с информацией, представленной в файлах на внешних носителях наряду с функциями вывода информации существуют функции *device*, *open* и *close*.

Вызов функции *open* аналогичен переключению потока на заданное устройство с помощью функции *rds* в muLISP'е (в суперпозицией с функцией *device*). Синтаксис :

(open <имя файла с указанием пути> <режим доступа>)

Имя файла с указанием пути заключается в кавычки. В отличии от принятого в DOS/Windows синтаксиса, названия директорий в указании пути отделяются друг от друга *обратной* косой чертой. Имя файла, директории (поддиректорий) в указании пути не должны содержать символов кириллицы.

Возможные режимы доступа к файлу :

"read" – только чтение;

"write" – только запись, старое содержимое файла удаляется;

"update" – чтение и запись новой информации поверх существующей;

"append" - чтение и запись новой информации в конец файла.

Переключение устройства ввода/вывода в newLISP-tk.

(device <№ устройства ввода/вывода>)

Переключает поток ввода/вывода на устройство, номер которого функция получает в качестве аргумента. Аргумент функции *device*, в частности, может быть значением, возвращаемым функцией *open*. Номер логического канала ввода/вывода, возвращаемый функцией *device*, служит для внутреннего использования функциями *print* и *read-line*.

Закрытие файла производится функцией close, которая имеет целочисленный аргумент - номер логического канала ввода/вывода, возвращаемый суперпозицией *device* и *open.*

Пример :

```
(device (open “D:/test_file/test.dat" “write"))

(print “Hello, World !")

(close (device))
```

В файл *test.dat*, находящийся в директории *test_file* на диске *D*, будет выведена строка “Hello, World !“. Старое содержимое файла будет уничтожено.

Чтение из файла в newLISP’е:пример.

```
(define (load_my_data file_name)
 (setq my_data '())
 (device (open file_name "read"))
 (while (setq el (read-line (device)))
    (setq my_data (cons (int el) my_data)))
 (close (device))
 my_data
)
```

Литература.

1. Хювенен Э., Сеппянен Й. Мир Лиспа. Т.1. – М.:Мир, 1990. С. 180-184.
2. Lutz Mueller newLISPTM For BSDs, Linux, Mac OS X, Solaris and Win32. Users Manual and Reference v.9.1 // www.nuevatec.com