Сравнения Scheme с Perl5

Кевролетин В.В. 236гр.

27 мая 2011 г.

Короткая справка

Perl — высокоуровневый интерпретируемый динамический язык программирования общего назначения.

Ларри Уолл, автор язка программирования Perl начал его разработку в 1987 году. Первая версия программы, Perl 1.0, вышла в том же году. Первая версия современного Perl 5 увидела свет 17 октября 1994 года, с тех пор она притерпела некоторые изменения и в настоящий момент самая последняя версия 5.14.0.
Далее, говоря Perl я буду подразумевать современную версию языка программирования Perl 5.

В противовес языку Scheme - разработка интерпретатора языка Perl, ведётся централизованно, альтернативных версий интерпретатора не существует. Средством расширения возможностей языка являются модули, для которых существует централизованное хранилище www.cpan.org

Perl в первую очередь является императивным языком, в отличае от функционального Scheme. Но при аккуратном целенаправленном программировании, основные идеи функционального программирования можно воплотить в жизнь, используя Perl.

Дисциплина связывания значений и переменных

Отличительная особенность Perl по отношению к большенству динамеческих языков программирования - переменные связываются по значению. Простой пример кода эллюстрирует эту особеннойть:

```
my @a = (1, 2, 3);
my @b = @a;
$b[0] = 10;
print $a[0]; #prints 1
```

В этом примере после присваивания @b = @a происходит копирование содержимого массива @a в массив @b. После этого значение первого элемента @b меняется, но это никак не влияет на значение @a;
В отличие от Scheme Perl имеет тип данных "Ссылка"и позволяет создавать неименнованные объекты. Все данные передаются в процедуры по ссылке. Кроме того, рекурсивные процедуры данных в Perl невозможно создавать без использования ссылок. Т.о. средствами языка можно хранить в переменных вместо значения, ссылки на объект и работать с данными как при можели связывания переменных по ссылке.

Типизация

Типизация в Perl и Scheme – строгая, динамическая. Несмотрю на то, что в Perl есть 3 различных основных типа данных, принадлежность к которым определяется синтаксическими правилам именования переменных: скаляр(может содержать числовые значения, строку или ссылку), массив, хэш-таблица, тип переменной влияет только на её внутреннее представление, на допустимые операции с переменной и на то, как трактуется присваемое переменной значение.
Так в следующем примере переменным 3х разных типов присваиваются одинаковые значения:

```
my @a = (1, 2); #array
my %b = (1, 2); #hash
my $c = (1, 2); #scalar
```

при этом программа семантически и синтаксически корректна.
Фактически же скаляр может содержать в качестве значения строку, число или ссылку на объект. Ошибки типизации выявляются только в процессе выполнения программы. Поэтому можно утвердать, что Perl является языком с динамической типизацией
Преобразование типов производится неявно но может быть выполнено я явно. Ошибки типизации выявляются на этапе выполнения программы.

Пространство имен

В Scheme одному глобальному идентефекатору соответсвует один объект или функция. В Perl перед именем переменной ставится один из специальных символов: @, $, %. Перед именем функции ничего не пишется. В программе можно иметь массив, скаляр, хэш-таблицу и функцию с одинаковым именем, но при этом чтобы их использовать необходимо ставить перед именем переменной различающиеся специальные символы.
Основным способом расширения возможностей языка являются, как и в Scheme, модули. Внутри модулей можно объявлять глобальные и локальные переменные и функции.
Так же, как и в Scheme можно объявлять локальные переменные внутри функций, но нельзя объявлять вложенные функции.

Процедуры, как параметры и возвращаемые значения

В Perl можно создавать неименованные функции, которые запоминают окружение, в рамках которого они бы или созданы. Можно содать ссылку на неименованную функцию и присвоить её скаляру, отправить в качестве аргумента в функцию или вернуть из функции. Таким образом, можно достичь эффекта, использования lambda-функций в Scheme.
В следующем примере создаётся неименнованна функция, которая хранит ссылку на переменную $sum, принимает один аргумент и прибавляет во время вызова его значение к $sum. В качестве результата возвращает новое значение $sum:

```
sub make_accumulator {
    my ($sum) = @_;
    sub {
        $amount += $_[0];
    }
}

my $accum = make_accumulator(10);
print $accum->(20); # 30
print $accum->(10); # 40
print $accum->(-5); # 25
```

Создание структур данных с использованием пар

В Perl нет стандартного типа пара, но массив из 2х элементов вполне подходит для представления пар в программе. Ниже приведу пример эмулирующий пары при помощи массивов:

```
sub cons { [$_[0], $_[1]] }

sub car { $_[0]->[0] }

sub cdr { $_[0]->[1] }

sub is_null { @{$_[0]} == 0 }

sub simply_print_list {
```

```
    my ($list) = @_;
    my $res = '';
    while (!is_null($list)) {
        $res .= car($list) . ' ';
        $list = cdr($list);
    }
    print $res;
}

my $list = cons(3, cons(2, cons(1, [])));
simply_print_list($list); #print 3 2 1
```

Символьные данные

В Perl, в качестве символьных данных выступают только строки, в отличае от символов и строк в Scheme. Для работы со строками в Perl существует множество функций и механизм регулярных выражений, являющийся отличительной чертой Perl. Строки в Perl не являются составным типом данных, как в большинстве распространённых ЯП. Преобразование из числа в строку и из строки в число(если это возможно) происходит только автоматически. Ниже приведён пример, где скаляру присваивается число, затем к нему применятся функция для работы со строками:

```
my $num = 99;
substr($num, 1, 0, 'Hello');
print $num; # prints 9Hello9
```

Представление систем, как набора взаимодействующих объект

Perl имеет довольно скудые средства для ООП: возможность создавать объекты определённого типа и одиночное наследование единственное, что предоставляет Perl.
Существует множество модулей, расширяющих возможности ООП, самый популярный из них Moose. Moose позволяет: создавать явно типизированные атрибуты, методы, конструктор, деструктор; множественное наследование; инициализаторы атрибутов(отложенных вычисления);

Потоки и ленивые вычисления

В Perl нет стандартной реализации потоков и ленивых вычислений. Но существует несколько модулей для ленивых вычислений и ленивых массивов(понятия список в Perl нет, но массив в этом плане схожее понятие т.е. представляет, как и список линейную последовательность элементов). Кроме того, наличие в Perl неименованных функций позволяет создать собственную реализацию ленивых вычислений и потоков, аналогичную потокам в Scheme.