1 Возникновение языка Perl

Язык программирования Perl создал американский программист Ларри Уолл (Larry Wall) в 1987 году, когда он работал системным программистом в компании Unisys. Цели, которые преследовал Ларри при разработке нового языка программирования, отражены в его названии - PERL, которое позднее стало расшифровываться как Practical Extraction and Report Language, то есть «практический язык извлечения “данных” и “создания” отчетов». (Нетрудно заметить, что первые буквы всех слов названия составляют слово PEARL – “жемчуг”. Одна из легенд о происхождении Perl гласит, что в то время уже существовал язык с таким названием, поэтому Ларри сократил название на одну букву, не изменив произношения. По иронии судьбы, сегодня тот язык не помнит никто, кроме историков, а Perl известен всему миру. Хотя, по другой версии, Ларри назвал созданный им язык по имени своей невесты.

Ларри Уолл в шутку (в которой, как водится, есть немалая доля истины) провозгласил три главных добродетели программиста: лень, нетерпение, самомнение (laziness, impatience, hubris). И Perl спроектирован так, чтобы соответствовать этим качествам разработчика. “Ленивый” программист, часто ограниченный во времени, может написать программу максимально компактно и быстро, поскольку в Perl есть множество способов кратко и просто записать довольно сложные алгоритмы.

Для языка Perl написана очень полная и подробная документация. Ее можно изучить через web-интерфейс на сайте www.perl.com/doc/. Для глубокого изучения Perl рекомендуем “Программирование на Perl” O’REILLY. Для тех, у кого курсовой проект на Perl – “Perl Сборник рецептов” O’REILLY.

2 Зачем нужны программы?

- Выполнять рутинную работу за человека. Например скачать тысячу последовательностей из UniProt.
- Выполнять сложные задачи. Например выровнять две длинные последовательности.

Компьютер железный – может сделать много работы. Надо лишь уметь его попросить.

3 Первая программа

Очень простые вещи в Perl делать очень просто. Если надо вывести на экран «hello world», то можно просто сказать:

Листинг 1: Первая программа

```
print "hello world\n";
```

4 Как запустить?

- На kodomo написать команду: `perl имяФайла`
- простые скрипты можно писать прямо в командной строке:
  perl −e 'print "hello world\n";'
- В windows в AbsolutePerl нужно нажать F9

5 Структура языка perl

Также, как и у любого "нормального" языка, у perl есть существительные и глаголы. Существительные в perl называются термами. С одним из них мы уже знакомы – это строка `"hello world\n"`. Строкой называют последовательность символов. Обычно строки выделяют двойными кавычками, чтобы не спутать их с другими словами в программе. Другой часто встречаемый терм – это число, например 10. Числа указывают без кавычек.

Листинг 2: Пример использования числа

```
print 10, "\n"; #Печатаем число 10 и символ новой строки
```

Символ # обозначает коментарий програмиста: все, что написано начиная с него и до конца строки не считается частью программы. Команда `print` умеет печатать несколько вещей за один раз, если перечислять их через запятую.

Существует еще парочка термов, но пока остановимся на двух.

Жизнь была бы скучна, если бы в языке были только термы (существительные). Для того, чтобы что-то делать с ними существуют операторы и функции. Оператор как и функция принимает один или несколько термов (бывает и ниодного) и возвращает другой терм. Комбинации термов и операторов называются выражением. Результатом вычисления выражения является терм. Многие из операторов вам знакомы со школы.

Листинг 3: Пример использования выражения

```
print 2+2, "\n"; #Печатаем результат сложения 2 и 2,
                 # а также символ новой строки
```

5.1 Некоторые простые операторы

Оператор	Описание	пример	результат
`+`	сложение	`3+2`	5
`-`	вычитание	`3-2`	1
`*`	умножение	`3*2`	5
`/`	деление	`3/2`	1.5
`**`	возведение в степень	`3**2`	9
`%`	взятие по модулю (остаток от деления)	`3%2`	1
`.`	объединение строк	`"Вася" . "Маша"`	ВасяМаша
`x`	повторение строки n раз	`"ATG" x 3`	ATGATGATG

6 переменные

Очень часто результат вычисления выражения удобно использовать не сразу, а где-то запомнить и потом использовать (вспомните калькулятор). Для этой цели служат переменные. Переменная – это нечто, куда можно что-то положить, а потом извлечь. Аналогия из жизни: коробка.

У переменной есть имя и значение. Пока мы познакомимся только с так называемыми скалярными переменными (в которых можно хранить текст или число). Их имена в перле должны начинаться с $

Листинг 4: Пример работы с переменными

```
my $variable; #Объявляем переменную
#Переменной можно присвоить значение:
$variable = "Hello";
#Или можно сразу объявить переменную и присвоить ей значение
my $variable2 = "world";
#Распечатаем значение прерменной:
print $variable, "\n";
#А теперь составим предложение
my $salutation = $variable1 . " " . $variable2 . "!\n";
#И распечатаем
print $salutation; #Распечатается: Hello world!
```

7 Полезные операции над строками

название	что делает	пример

chomp	удаляет в конце строки символ-разделитель записей	`my $string1="string\n";`
		`my $string2="string";`
		`chomp string1;`
		`chomp string2;`
		`print $string1 . $string2 . "\n";`
		`#напечатает string string`

chop	удаляет любой последний символ строки	`my $string1="string\n";`
		`my $string2="string";`
		`chop string1;`
		`chop string2;`
		`print $string1 . $string2 . "\n";`
		`#напечатает string strin`

index	выполняет поиск подстроки в строке, начиная с определенного смещения, и возвращает номер позиции найденной подстроки	`my $string="atgcgcgatg";`
		`print index $string, "atg";`
		#напечатает 0

rindex	ищет подстроку от конца строки и возвращает позицию последней подстроки в строке перед указанным смещением	`my $string="atgcgcgatg";`
		`print rindex $string, "atg";`
		#напечатает 7

substr	выделяет подстроку в строке	`my $string="string";`
		`print substr $string, 2, 4;`
		# отступаем от начала 2 символа,
		# берем подслово длины 4
		# напечатает ring
		`substr $string, 3, 1, "o";`
		`print $string . "\n";`
		# напечатает strong

lc	возвращает значение текстового выражения, преобразованное к строчным буквам	`my $string="stRinG";`
		`print lc $string;`
		#напечатает string

uc	возвращает значение текстового выражения, преобразованное к заглавным буквам	`my $string="stRinG";`
		`print lc $string;`
		#напечатает STRING

8 Общение с внешним миром

Вообще, программы, которые делают всякий раз одно и тоже и не берут никаких данных на вход, неинтересны. Можно, например, написать программу приветствия, используя переменную, которая хранит имя. Программа спрашивает имя пользователя и читает, то что ввел пользователь.

Листинг 12: Пример получения данных от пользователя

```
print "what is your name?\n";
my $name=<STDIN>; # <STDIN> – умеет читать сроку
                  #из стандартного потока ввода,
                  #т.е то, что введет пользователь
                  # Пока не важно понимать,
                  # что тут написано,
                  # важно что это работает.
print "hello, " . $name . "\n";
```

9 Домашнее задание

9.1 Задачи на 1 балл:

Написать программу которая:

1. Спрашивает имя, фамилию и отчество. Печатает вежливое приветствие и число букв в этом приветствии.

2. Просит ввести слово. Потом текст. После этого выдает слово и его длину, текст и его длину и кусок текста следующий за найденным словом.

3. Просит ввести имя, фамилию и возраст. Печатая имя и фамилию большими буквами поздравляет Вас с днем рождения столько раз сколько Вам лет.

9.2 Задачи на 2 балла:

Написать программу которая:

1. Просит ввести текст. Выводит его длину и просит ввести слово. Выводит координаты первого вхождения этого слова в текст и просит ввести другое слово. Заменяет в тексте первое слово на второе и печатает итоговый текст. Вся программа должна быть нечувствительна к регистру – т.е. не должна различать большие и маленькие буквы.