1 Хеши

Говоря сухим языком, хэш в перле - неупорядоченный набор скаляров (значений, values) и неупорядоченный набор строк (ключей, keys), между которыми установленно соответствие один к одному. А если отбросить детали это таблица из двух столбцов, например:

ключ	значение
Понедельник	0
Вторник	1
Среда	2
Четверг	3
Пятница	4
Суббота	5
Воскресение	6

Ключом здесь выступает день недели, а значением его «номер» в неделе. Приведем код такой таблицы.

Листинг 1: Создание хеша

```
my %day = (
   Monday =>     0,
   Tuesday =>    1,
   Whendsday =>  2,
   Thursday =>   3,
   Friday =>     4,
   Saturday =>   5,
   Sunday =>     6,
);
```

Префикс % указывает, что переменная хранит хэш. При создании хэшу ередается список пар ключ => значение. Символ => является синонимом запятой и введен для удобства программиста, вместо него можно указывать запятую – таким образом для хэша подойдет любой список, лишь бы он был четной длинны.

Теперь если вы забыли какой по счету день недели понедельник, то можете получить его вызовом **print** $day{Monday}; (Следует читать как print day of Monday). Опять обратите внимание – мы хотим получить скаляр (одно значение), поэтому перед day стоит префикс $. Также как и стоял бы при случае с массивом.

Кстати хэш очень схож с массивом. Приведем пример массива:

индекс	0	1	2	3	4	5	6
значение	Понед.	Вторник	Среда	Четверг	Пятница	Суббота	Воскр.

Этот массив может помочь вспомнить, какой день недели третий по счету, обратившись по индексу **print** $array[3], но вот обратную задачу с массивом решить не так просто, как с хэшом. Индексом массива не может быть строка. Более того, если в массиве существует тысячный элемент, то память

в компьютере выделяется на все ячейки от 0 до 1000, даже если мы используем только одну ячейку под номером 1000. В хэше же нет никаких ячеек, хотя и удобно мысленно представлять хэш как таблицу.

Хэш очень удобен, если вам нужно ассоциировать (установить соответствие) между двумя наборами данных, например между последовательностями и их именами. Приведем пример программы, которая читает выравнивание в формате fasta и выводит те последовательности, которые попросил пользователь:

Листинг 2: Программа извлечения последовательностей из выравнивания

```
#!/usr/bin/perl
$\ = "\n";
%sequnce = ();
@requested_seqs = @ARGV;
$SEQ_LINE_LENGTH = 80;

my $seq_name;
while(<STDIN>){
        chomp;
        # если строка начинается с >
        #установить имя новой последовательности
        if( substr($_, 0, 1) eq ">"){$seq_name = substr $_, 1;}
        #добавить строку к последовательности
        else{$sequence{$seq_name} .= $_;}
}
for my $seq_name (@requested_seqs){
        warn "no such sequence: $seq_name"
                and next if not defined $sequence{$seq_name};
        print ">$seq_name";
        #печатаем последовательность строками по 80
        print substr($sequence{$seq_name}, 0, $SEQ_LINE_LENGTH, "")
                while $sequence{$seq_name};
}
```

1.1 функции для работы с хэшами

функция	описание
keys %hash	возвращает ключи
values %hash	возвращает значения
exists $hash{key}	проверяет существования ключа
delete $hash{key}	удаляет ключ

2 Ссылки

Мы разобрали все структуры данных которые есть в перле. Однако в програмировании требуется много других более сложных типов данных, например таблицы с множеством колонок, а не только с двумя, или многомерные массивы (например для представления матриц).

При создании сложных типов данных используют ссылки. Ссылка это

терм, который ссылается на содержимое другой переменной. Являясь скаляром, ссылка, в тоже время, может ссылаться на любую структуру данных, будь то массив или хеш. Будучи скалярами, ссылки можно хранить в массивах или хэшах, создавая массивы массивов, хэши массивов итд.

2.1 Создание ссылок

Ссылка создается оператом backslash:

Листинг 3: Пример создания массива массивов

```
@row1 = (5, 10);
@row2 = (7, 8) ;
@matrix =();
$matrix[0] = \@row1;
$matrix[1] = \@row2;
```

Для того, чтобы создать ссылку на массив не обязательно придумывать имя переменной. Можно сразу создать ссылку на анонимный массив с помощью квадратных скобок: $matrix[0] = [5, 10]; Для создания ссылки на анонимный хеш применяют фигурные скобки: $hashref = {"key", "value"};'

2.2 Обращение с ссылками

Можно считать, что backslash представляет любой объект в виде скаляра. Попытка распечатать ссылку приведет к тому, что выведется тип данных объекта, на который указывает ссылка, и его адрес в памяти. Чтобы получить доступ к объекту за ссылкой, его надо разыменовать дополнительным префиксом.

Листинг 4: Пример разыменовывания ссылки

```
$arrayref = \@array;
print $arrayref; # выведет, например: 'ARRAY(0x335b04)
print $$arrayref[0]; # выведет, например: 5
```

Если структура данных очень сложная, то бывает утомительно писать все символы $. На помощь приходит оператор стрелка:

Листинг 5: Еще один пример

```
$longarray = [ [2, 3 ,5], [2, 2 ,2] ];
print $$longarray[0][1];
print $longarray->[0][1];
```

3 Задачи

1. 1 балл. Интерактивная телефонная книга. При вводе пользователем команды add программа предлагает ввести новую запись: фамилию, затем номер телефона. При вводе команды get программа предлагает ввести фамилию, по фамилии ищет номер телефона и выводит его на экран. Выход из программы происходит после ввода команды exit. При написании программы реализовать методы: add, который добавляет нового абонента в книгу, и get, который извлекает по имени телефонный номер.

2. 2 балла Дан файл с последовательностью ДНК. Программа ищет все возможные ORF и считает для всех ORF статистику аминокислот и кодонов:

   (a) частоты встречаемости аминокислот

   (b) для каждой аминокислоты частоту использования того или иного кодона. Для простоты будем считать, что стартовый кодон только atg, а стоп-кодон tga.