КАЗАНСКИЙ ГОСУДАРСТВЕННЫЙ ЭНЕРГЕТИЧЕСКИЙ УНИВЕРСИТЕТ

Кафедра иностранных языков

**Beginning Perl for Bioinformatics**

James Tisdall

Publisher: O'Reilly

First Edition October 2001

ISBN: 0-596-00080-4, 384 pages

Перевод выполнила: Аспирантка кафедры ИК
 Габидинова А.Р.

Научный руководитель: Д.т.н., профессор
 Шарифуллин В.Н.

Проверил: Доцент кафедры ИЯ
 Молюков И.М.

КАЗАНЬ 2007

# Глава 4. Цепочки и строки

Изучая эту главу Вы сможете начать писать Perl-программы для манипуляций данными биологических последовательностей ДНК и белков.

Если файлы с последовательностями есть на Вашем компьютере, то создаваемые программы позволят проводить следующие операции с последовательностями:

- переводить генетический код ДНК в РНК
- сцеплять последовательности
- читать комплементарную последовательность в обратном направлении

- считывать последовательности из файлов
А так же напишем программу, которая выдает информацию о последовательностях, насколько ценный генетический код ДНК, какова гидрофобность протеинов. Вы познакомитесь с методологией программирования и сможете решить эти и аналогичные задачи.

После прочтения данной главы Вы обретете основные навыки языка Perl. Вот некоторые из них:

- скалярные переменные

- массивы

- операции со строками: подстановка и преобразование

- чтение данных из файла

## 4.1 Представление данных цепочки

Большинство действий, описанных в этой книге, это обработка символов, которые составляют записи цепочек ДНК и протеинов. В БиоИнформатике для аналитического представления биологических цепочек используются такие же обозначения как в литературе.

Как известно, ДНК состоит из 4 структурных видов - нуклеиновых кислот, которые так же называются нуклеотидами или основаниями. Белки состоят из 20 структурных видов - аминокислот, иначе - радикалов. Фрагменты белков называются пептидами. И ДНК, и протеины по существу полимеры, которые сформированы из структурных видов, скрепленных конец к концу. Таким образом, можно просуммировать структуру молекулы ДНК или белка просто сложив цепочки оснований или аминокислот.

Это краткие определения. Предполагаю, что Вы или уже знакомы с ними, или можете познакомиться поподробнее в вводном учебнике по молекулярной биологии. В табл. 4-1 представлены азотистые основания; добавьте сахарозу и Вы получите нуклеотиды - аденозин, гуанозин, цитидин, тимидин и уридин. Кроме того, добавив фосфат, Вы можете получить нуклеотиды - адениловую кислоту, гуанилиновую кислоту, цитидиновую кислоту, тимидиловую кислоту и уридиловую кислоту. Нуклеиновая кислота – это химически связанная последовательность нуклеотидов. Пептид – это небольшое число объединенных аминоксилот; длинная цепочка полипептидов. Протеин – это биологически функциональное соединение, составленное из одного или нескольких полипептидов. Радикал – это аминокислота в полипептидной цепи.

Для целесообразности, имена нуклеиновых кислот и аминокислот чаще представлены как одно- или трехсимвольные коды, как показано в Табл. 4-1 и Табл. 4-2. (В этой книге по большей части используются односимвольные коды аминокислот.)

*Table 4-1. Standard IUB/IUPAC nucleic acid codes*

*Table 4-2. Standard IUB/IUPAC amino acid codes*

Коды нуклеиновых кислот в Табл. 4-1 включают символы 4 основных нуклеиновых кислот. В большинстве случаев в этой книге я использую только A, C, G, T, U и N. Буквы A, C, G и T задают нуклеиновые кислоты для ДНК. U заменяется на T, когда ДНК переводится в РНК. N – общепринятое обозначение для “неизвестного”, как например, когда секвенсер не может определить азотистое основание с достоверностью. Позже, в Главе 9, нам понадобятся другие коды для групп нуклеиновых кислот, для программирования рестрикционных карт. Надо заметить, версии со строчными буквами этих однобуквенных кодов так же используются по случаю, чаще для ДНК, реже для белков.

В научных компьютерной и биологической терминологиях имеются небольшие различия в кодах Табл. 4-1 и Табл. 4-2. На научно-компьютерном языке эти таблицы определяют два алфавита с конечными наборами символов, которые могут составлять строки. Последовательность символов называется строкой. Например, это предложение является строкой. Язык (конечный или бесконечный) – это набор строк. В этой книге языки, в основном, последовательностей данных ДНК и белков. Вы чаще слышите, что биоинформатики говорят о реальной последовательности ДНК или белка как о “строке”, в противопоставление их представлению как о последовательности данных. Это пример терминологии двух дисциплин, пересекающихся в чем-то.

Как Вы видели в таблицах, данные изображены как простые буквы, подобно тем, которые написаны на странице. Но, в действительности, компьютеры используют дополнительные коды для изображения простых букв. Не беспокойтесь об этом, просто помните это при использовании текстового редактора, что их нужно сохранять как ASCII или простой текст.

Посредством ASCII компьютер загружает текстовую (и управляющую) информацию в память. Затем, когда программой, такой как текстовый редактор, читаются данные, о которых известно, что это ASCII, программа в действительности способна вывести на экран символы в понятном виде, поскольку запрограммировано распознавать этот конкретный код. То есть нижняя строка это: ASCII – это код для записи текста в компьютер.

Новая кодировка символов, называемая Юникодом, которая может обрабатывать все символы всех языков в мире, становится широко применимой, и поддерживается и в Perl.

4.2 Программа сохранения цепочки ДНК

Давайте напишем небольшую программу, которая сохраняет ДНК в переменной и распечатывает на экран. ДНК записана в обычном виде, как строка из букв A, C, G и T и мы назовем переменную $DNA. Другими словами, $DNA это название данных цепочки ДНК, использованных в программе. Заметьте, что в Perl переменная это в действительности название для используемых данных. Название дает полный доступ к данным. Пример 4-1 показывает всю программу.

*Example 4-1. Запись ДНК в компьютер*

Используя то, что вы узнали о текстовых редакторах и запуске Perl программ в Главе 2, введите код (или скопируйте его с сайта учебника) и сохраните в файл. Не забудьте сохранить программу в формате ASCII или ASCII-файл без элементов форматирования, иначе могут быть затруднения в Perl при чтении исходного файла.

Второй шаг – запуск программы. Как запустить программу зависит от типа вашего компьютера (см.Главу 2). Допустим, программа на вашем компьютере находится в файле с названием *example4-1*. Как вы помните из Главы 2 , если вы запускаете программу на Unix или Linux , то должны набирать в окне интерфейса следующее:

*perl example4-1*

А на Макинтош откройте файл в приложении MacPerl и сохраните его как droplet, и затем просто дважды щелкните на droplet. В Windows наберите следующее в командном окне MS-DOS:

*perl example4-1*

Если программа успешно запустилась, то вы увидите вывод результатов на экран компьютера.

### 4.2.1 Процесс управления

Пример 4-1 иллюстрирует много понятий, на которых основана наша Perl программа. Одно из этих понятий Control Flow – процесс управления или порядок, в котором задачи выполняются компьютером.

Каждая программа начинается с первой строки и выполняет команды одну за другой, пока не дойдет до конца, если не задано иначе. Пример 4-1 просто выполняется с верху до низу, без изменений.

В следующих главах вы узнаете как можно контролировать порядок выполнения в программе.

### 4.2.2 Переработка комментариев

А сейчас давайте обратим внимание на часть Примера 4-1. Вы заметите много пустых строк. Они использованы для того, чтобы сделать программу легкой для чтения человеком. Затем, обратите внимание на комментарии, которые начинаются со знака #. Вы помните из Главы 3, что при компиляции Perl отбрасывает их вместе с пустыми строками. По сути, для Perl следующий код, это точно такая же программа как Пример 4-1.

*#!/usr/bin/perl -w*

*$DNA = 'ACGGGAGGACGGGAAAATTACTACGGCATTAGC'; print $DNA;*

*exit;*

В примере 4-1 комментарии свободно использовались. Комментарии в начале кода могут объяснять для чего программа, кто ее написал и представить другую информацию, которая может быть полезна, когда кому-то нужно понять код. Комментарии также объясняют, для чего каждая часть кода и иногда объясняют как код выполняет свою задачу.

Чтобы понять значимость комментариев, достаточно сказать, что на университетском уровне программа без комментариев обычно оценивается низко или неудовлетворительно, а программист, который в работе не комментирует ответственно свой код, не может сделать хорошей карьеры.

### 4.2.3 Интерпретация команд

Первая строка выглядит как комментарий, поскольку начинается со знака #, но кажется не очень информативной.

#!/usr/bin/perl -w

Это специальная строка, которая называется интерпретацией команд и объясняет компьютеру под UNIX или LINUX, что это программа на Perl. Она может выглядеть немного по-разному на разных компьютерах. На некоторых компьютерах она, к тому же, и не нужна, так как компьютер узнает, что это Perl из другой информации. Конфигурация компьютеров под Windows обычно сконфигурированы ассоциировать любую программу, заканчивающуюся на .pl как Perl программу. В UNIX или LINUX , в командном окне Windows или в оболочке MacOS X, можно набрать perl my\_program и тогда Perl программе my\_program не нужно специальных строк. Однако, они часто используются, и будут встречаться в начале всех наших программ.

Обратите внимание, что в первой строке используется флаг –w. –w означает предупреждения и заставляет Perl выводить сообщения в случае ошибок. Очень часто сообщение об ошибке указывает номер строки, в которой, предположительно, начинается ошибка. Иногда номер строки бывает неверным, но обычно ошибка именно в той строке или в предыдущей строке. Далее в книге вы так же увидите формулировку use warnings, как альтернатива –w.

### 4.2.4 Операторы

Следующая строка примера 4-1 сохраняет ДНК в переменной:

$DNA = 'ACGGGAGGACGGGAAAATTACTACGGCATTAGC';

Это очень распространенная и важная вещь, выполняемая в языке программирования, давайте рассмотрим ее не спеша. Вы увидите некоторые базовые особенности Perl и языка программирования в целом. Как раз здесь следует перестать лишь бегло просматривать, а стоит начать действительно читать. Такая строка кода называется оператором. В Perl оператор заканчивается на ; (точка с запятой). Использование ; похоже на использование этого знака для пауз в предложении в английском языке.

Точнее говоря, эта строка кода – оператор присваивания. Его назначение в этой программе - сохранить некоторую ДНК в переменную под названием $DNA. Здесь есть несколько важных моментов, вы сможете узнать о них в следующем разделе.

#### 4.2.4.1 Переменные

Для начала давайте взглянем на переменную $DNA. Это название весьма произвольное. Вы можете подобрать другое название для неё, а программа будет работать также. Например, если вы заменяете две строки :

$DNA = 'ACGGGAGGACGGGAAAATTACTACGGCATTAGC';

print $DNA;

этими строками:

$A\_poem\_by\_Seamus\_Heaney =

'ACGGGAGGACGGGAAAATTACTACGGCATTAGC';

print $A\_poem\_by\_Seamus\_Heaney;

программа работает точно также – печатает ДНК на экран компьютера. Дело в том, что название для переменных в компьютерной программе дается на ваш собственный выбор. (В рамках обязательных ограничений: в Perl название переменной должно быть составлено из прописных или строчных букв, цифр, и символов подчеркивания. А также первый символ не должен быть цифрой.)

Это другой важный момент, подобный тем замечаниям, которые я делал, об использовании пустых строк и комментариев, чтобы сделать код более легким для чтения человека.

Компьютер не придает значения использованию названия переменой $DNA вместо $A\_poem\_by\_Seamus\_Heaney, но кто бы не читал программу, конечно же обратит внимание на это.

Одно название придает настоящий смысл, ясно указывает для чего эта переменная в программе и облегчает постоянные операции в понимании программы. Другое название делает непонятным, что делает программа или для чего эта переменная. Использование хорошо подобранных названий для переменных это часть так называемого самодокументирования кода. Комментарии все ещё будут нужны, но скорее всего уже не часто, если вы хорошо подобрали название переменной.

Вы заметили, что переменная $DNA начинается со знака доллара. В Perl такой вид переменой называется скалярной переменной, и эта переменная содержит один элемент данных. Скалярные переменные используются для таких типов данных как строковые или числовые типы (например, строка hello, или числа 25, 6,234, 3,5Е10, -0,8373). Скалярная переменная содержит только один элемент данных в каждый момент.

#### 4.2.4.2. Символьная строка

В примере 4-1 , скалярная переменная $DNA содержит некую ДНК, представленную обычным образом буквами A, C, G, и T. Как говорилось ранее, в компьютерной науке последовательность букв называется символьной строкой. В Perl, чтобы определить символьную строку, нужно заключить её в кавычки. Можно использовать одинарные кавычки, как в примере 4-1, или двойные кавычки. (Вы узнаете их различие немного позже). Таким образом ДНК выглядит так:

'ACGGGAGGACGGGAAAATTACTACGGCATTAGC'

#### 4.2.4.3.Присваивание

В Perl, чтобы присвоить определенное значение переменной используется знак =. Этот знак = называется оператором присваивания. В Примере 4-1, значение 'ACGGGAGGACGGGAAAATTACTACGGCATTAGC'

присваивается переменой $DNA. После присваивания Вы можете использовать название переменной, чтобы получить это значение, например как в операторе print в примере 4-1.

В операторе присваивания важен порядок частей. Значение, присвоенное чему-то, указывается справа от оператора присваивания.

Переменная, которая присваивает значение, всегда слева от оператора присваивания. В справочниках по программированию вы иногда встречаете термины lvalue и rvalue слева или справа от оператора присваивания.

Использование знака = имеет долгую историю в языке программирования. Однако, он может натолкнуть на определенный вывод: например, в математике использование знака = означает равенство двух значений по сторонам от знака. Итак, важно отметить, что в Perl знак = не означает равнозначность. Он присваивает значение переменной (Далее мы увидим, как показать равнозначность).

Итак, чтобы подвести пока итог того, что мы узнали об этом операторе:

$DNA = 'ACGGGAGGACGGGAAAATTACTACGGCATTAGC';

Это оператор присваивания, который присваивает скалярной переменной $DNA значение символьную строку, означающую некую ДНК.

#### 4.2.4.4. Печать

Оператор: print $DNA;

выводит ACGGGAGGACGGGAAAATTACTACGGCATTAGC на экран компьютера. Заметьте, что оператор print работает со скалярными переменными, распечатывая их значение, в этом случае символьную строку, которую содержит переменная $DNA. Позже вы узнаете о печати больше.

#### 4.2.4.5. Выход

В заключение, оператор выхода указывает компьютеру выйти из программы. Perl не требует оператора выхода при завершении программы, программа завершается автоматически, когда доходит до конца. Но не помешает ввести этот оператор и он ясно укажет, что программа завершилась. Вы увидите другие программы, которые завершаются из-за ошибки до того как программа завершится обычным образом, поэтому, оператор выхода определенно полезен.

## 4.3 Соединение фрагментов ДНК

Сейчас мы сделаем небольшое изменение Примера 4-1, чтобы показать, как связать 2 фрагмента ДНК. Связывание - это присоединение чего-то одного к концу чего-либо другого. Биолог хорошо осознает, что соединение последовательностей ДНК - это обычное задание в биологической лаборатории, например, когда клон вставляется в вектор клетки или при соединении экзона при экспрессии генов. Многие биоинформатические пакеты программ производят такие операции, поэтому выбран следующий пример. Пример 4-2 показывает больше операций, связанных со строками, переменными и операторами печати.

*Example 4-2. Соединение фрагментов ДНК*

Как вы видите, здесь три переменные $DNA1, $DNA2, $DNA3. Я добавил оператор печати для подстрочного комментария для того, чтобы результаты программы, которые появляются на экране компьютера, были более понятны и не были просто последовательностью фрагментов некоего ДНК.

Вот как выглядит результат примера 4-2:

*Here are the original two DNA fragments:*

Пример 4-2 очень схож с примером 4-1. Давайте посмотрим на различия. Для начала, оператор печати имеет несколько неясных дополнений:

*print $DNA1, "\n";*

*print $DNA2, "\n\n";*

В операторе печати имеются переменные, содержащие, как и ранее, ДНК, но помимо этого ещё запятую и затем "\n" или "\n\n". Это инструкции печатать разделитель строк. Разделитель строк невидим на странице или экране, но указывает компьютеру перейти на начало следующей строки перед последующей печатью.

Один разделитель строки "\n" просто ставит вас на начало следующей строки кода. Две новые строки "\n\n" переходят на следующую строку и потом ставят вас на начало строки, оставляя пустую строку внутри.

Посмотрите на код для Примера 4-2 и убедитесь, что видите действие этих указателей разделителей строк в выходных данных. Пустая строка кода - это строка, где ничего не напечатано. В зависимости от вашей операционной системы это может быть только символ новой строки или перевод страницы и возврат каретки ( в этих случаях, она может быть также названа незаполненной строкой кода), или она может содержать непечатаемый символ пустой строки такие как пробел или символ табуляции. Заметьте, что разделители строк заключены в двойные кавычки, это означает, что они части символьной строки. ( Вот здесь и разница между одинарными и двойными кавычками, о которой упоминалось ранее: "\n" печатает разделитель строки; '\n' печатает \n как есть.)

Обратите внимание на запятую в операторе печати. Запятая разделяет отдельные элементы в списке. Оператор печати печатает все перечисленные элементы. Так вот просто.

Сейчас давайте взглянем на оператор, который соединяет 2 фрагмента ДНК $DNA1, $DNA2 в переменную $DNA3:

$DNA3 = "$DNA1$DNA2";

Присваивание значения $DNA3 – это обычное присваивание значения, как вы видели а Примере 4-1, название переменной, за которой стоит знак = и после него следует значение, которое надо назначить.

Значение справа от оператора присваивания это символьная строка, заключенная в двойные кавычки. Двойные кавычки позволяют переменным в символьной строке заменяться своими значениями. Это называется интерполяцией символьной строки. [2] Таким образом, фактически, символьная строка здесь это всего лишь ДНК переменной $DNA1 за которой следует ДНК переменной "$DNA2. Это конкатенация двух фрагментов ДНК, которая затем присваивается переменной $DNA3.

2 Есть случаи, когда можно добавить фигурные скобки во время интерполяция символьной строки. Дополнительные фигурные скобки, чтобы убедиться, что название переменных не спутаны с какими-то символьными строками в двойных кавычках. Например, если у вас переменная $prefix и вы пытались бы интерполировать его в символьную строку I am $prefixinterested,Perl может не распознать переменную, путая ее с несуществующей переменной $prefixinterested. Но символьная строка I am ${prefix}interested не является двусмысленной для Perl.

После установления значения ДНК переменной $DNA3, вы распечатаете ее, и после следует пустая строка кода.

print "$DNA3\n\n";

Одно из хорошо известных выражений в Perl это «Существует не один способ сделать это!». Таким образом, следующая часть программы показывает другой способ соединения двух символьных строк, используя оператор точку. Оператор точка, когда стоит между двумя символьными строками, создает одну символьную строку, объединяющую 2 исходные символьные строки. Таким образом, следующая кодовая строка показывает использование этого оператора: $DNA3 = $DNA1 . $DNA2;

Оператор на языке программирования содержит какие-либо параметры, в данном случае - символьные строки $DNA1 и $DNA2, и производит действие, возвращая какое-то значение, в данном случае соединенную символьную строку в переменную $DNA3. Самые известные операторы - из арифметики – плюс, минус, умножение, деление – это операторы, которые берут исходные два числа как аргумент и возвращают число как значение.

В заключение, для тренировки в различных частях языка, давайте выполним ту же конкатенацию, используя только оператор печати.

print $DNA1, $DNA2, "\n";

Здесь оператор печати имеет 3 части, разделенные запятыми: 2 ДНК фрагмента в 2 переменных и разделитель строки. Вы можете достигнуть такого же результата со следующим оператором печати

print "$DNA1$DNA2\n";

Возможно нужно перефразировать слоган Perl : «Существует не только 2 способа сделать это!»

До завершения этой части, давайте, обратим внимание на другие применения переменных Perl. Вы уже увидели применение переменных для содержания символьной строки данных цепочки ДНК. Есть также другие типы данных, и для них тоже нужны переменные в языке программитрования. В Perl скалярная переменная, такая как $DNA, может содержать символьную строку, целое число и число с плавающей запятой (с десятичной запятой), логическое значение (true or false) и др. Когда необходимо, Perl вычисляет какие данные в переменной. А сейчас попытайтесь добавить следующие кодовые строки в Пример 4-1 или Пример 4-2 , сохраняя число в скалярную переменную и распечатывая ее.

## 4.4. Преобразование ДНК в РНК

Большая часть того времени, которое Вы, программист биоинформатики, тратите, уйдет на внесение изменений, аналогичных тем, что в Примере 4-1 и Примере 4-2. Вы получите какие-то данные, это будет ДНК, протеин, записи Генетического Банка или что-то еще, затем Вы будете оперировать этими данными и распечатывать какие-то результаты.

Пример 4-3 ещё одна программа, которая оперирует данными ДНК, она преобразует ДНК в РНК. В клетке преобразование ДНК в РНК результат работы хрупких, сложных, исправляющих ошибки молекулярных механизмов[3]. Здесь же это простая подстановка. Когда ДНК преобразуется в РНК, все T заменяются на U, и это все, что нужно знать нашей программе[4].

[3] Кратко говоря, кодирование ДНК цепочки – это обратное дополнение другой цепочки, которая используется как образец для синтезирования его обратного дополнения, такого как РНК, где Т заменено на U. C двумя обратными дополнениями, это то же, что кодирование замены T на U.

[4] Мы игнорируем механизм вырезания участков РНК вне интронов, это ясно. T обозначает Тимин; U обозначает Урацил.

*Example 4-3. Transcribing DNA into RNA*

*Вот вывод данных Примера 4-3:*

Эта короткая программа затрагивает важную часть Perl: способность возможность легко оперировать текстовыми данными, такими как символьная строка с ДНК. Операции могут быть различными: трансляция, реверсия, переупорядочивание и т.п. Эти возможности Perl – это одна из главных причин популярности в биоинформатике и среди программистов в целом.

Сначала программа делает копию ДНК и устанавливает ее в переменную с названием $RNA.

$RNA = $DNA;

Стоит заметить, что когда этот оператор выполнен, эта переменная $RNА содержит ДНК.[5] Помните, Вы вольны называть переменные как Вам угодно, но из-за неточных названий переменных возможны путаницы. В данном случае, после копирования следует информативный комментарий, а за ним оператор, суть которого заставить переменную $RNA содержать РНК, так что все в порядке. Это способ не допустить, чтобы в $RNA содержалось что-либо кроме РНК.

[5] Вернемся к обсуждению из Параграфа 4.2.4.3 о важности порядка частей в операторе присваивания. Здесь значение $DNA, которое является цепочкой данных ДНК, была сохранена в переменной $DNA и присваивается переменной $RNA. Если бы вы написали $DNA=$RNA, значение переменной $RNA (которое не содержит ничего) было бы присвоено переменной $DNA, фактически уничтожая цепочку данных ДНК в этой переменной и оставляя 2 пустые переменные.

($RNA = $DNA) =~ s/T/U/g;

В Примере 3-1 производится с помощью этого оператора:

$RNA =~ s/T/U/g;

В этом выражении два новых элемента: оператор связывания (=~) и команда подстановки s/T/U/g.

Очевидно, достаточно использовать оператор связывания (=~) для переменных, содеражщих символьные строки; здесь переменная $RNA содержит цепочку данных ДНК. Оператор связывания означает «применить операцию справа от символьной строки к переменной слева.

Оператор подстановки, показанный на Рисунке 4-1, требует немного больше объяснений. Разные части команды разделены (или разграничены) косой чертой. s в начале обозначает, что это подстановка. После первой / идет T, который указывает элемент в символьной строке, который должен быть заменен. После второй / идет U – это элемент для замены T. После третьей / стоит g. g означает «global» - это один из возможных модификаторов, которые могут ставиться в этой части оператора. Global означает «сделать это замещение во всей символьной строке», проще говоря, везде, где в этой символьной строке возможно.

*Figure 4-1. The substitution operator*

Таким образом, значение оператора в целом – «замещение всех T на U в символьной строке, содержащейся в переменной $RNA».

Оператор подстановки это пример использования регулярных выражений. Регулярные выражения это ключ к оперированию текстовыми данными, одно из мощнейших возможностей Perl, Вы сможете увидеть это в следующих главах.

## 4.5. Использование документации Perl.

Самый важный ресурс для Perl программиста это Perl-документация. Она скорее всего была установлена на Ваш компьютер, а так её можно найти в интернете на сайте Perl. Perl-документация может немного отличаться в различных компьютерных системах, но web-версия одинакова для всех. Это версия, на которую я буду ссылаться в данной книге. Посмотрите ссылки в Приложении A, чтобы узнать подробности о различных ресурсах с Perl- документацией.

Чтобы попробовать это, давайте найдем в справочнике оператор печати. Для начала, откройте свой веб-браузер и зайдите на http://www.perl.com. Затем кликните на сcылку Документация. Выберите “Perl’s Builin Functions”, затем “Alphabetical Listing of Perl’s Functions”. Вы увидите довольно длинный альфавитный список функций Perl. Возможно, как-нибудь Вы снова захотите найти эту страницу, поэтому можно сделать закладку на неё в веб-браузере, может оказаться, что вы часто обращаетесь к ней. Теперь кликните на Print, чтобы прочитать об операторе печати.

Попробуйте показанные примеры, чтобы почувствовать, как на самом деле используются языковые возможности. Это обычно самый быстрый способ извлечь необходимые знания.

Как-нибудь при просмотре документации Вы обнаружите, что чтение документации помогает ответить на некоторые вопросы, но так же вызывает другие. Документация направлена подать всю информацию в сжатом виде, что может испугать новичков. Например, документация по функции Print начинается так «Печатает символьную строку или список символьных строк, разделенных запятыми. Выдает TRUE в случае успеха». Но затем идет куча мусора (или так может показаться на данном этапе обучения): индекс файла, выходной поток, списковый контекст...

Вся эта информация необходима в документации, ведь в действительности Вам когда-то понадобится вся информация. Пока Вы можете не обращать внимание на то, что не требуется.

Perl документация также включает в себя несколько обучающих программ, которые могут сильно помочь в изучении Perl. Они иногда требуют больших знаний языка, чем те, которыми обладают новички, но скорее всего они окажутся Вам очень полезными. Изучение документации - это отличный способ ускорить изучение языка Perl.

## 4.6. Рассчет обратного дополнения с помощью Perl

Как вы помните из Главы 1, полимер ДНК составлен из нуклеотидов. Установив тесную взаимосвязь между 2 цепочками ДНК в двойной спирали, оказалось, что довольно просто написать программу, которая зная одну цепочку распечатывает другую. Данный расчет - важная часть многих биоинформатичеких программ. Например, при поиске ДНК в базе данных обычно автоматически идет и поиск обратного дополнения ДНК, в результате может быть найдена цепочка обратного порядка некоего известного гена.

Не затягивая более, рассмотрим Пример 4-4, в котором используется несколько известных свойств Perl. Как Вы можете видеть, сначала пробуется один метод, который оказывается неудачным, затем другой успешный.

*Example 4-4. Calculating the reverse complement of a strand of DNA*

Вот как должен выглядеть результат Примера 4-4 на вашем экране:

Вы можете проверить, действительно ли 2 цепочки ДНК это обратные наборы друг друга, прочитав один слева направо и другой справа налево, т.е. начав читать с разных концов. Затем сравните каждую пару основ, когда читаете 2 цепочки: они должны всегда быть в паре С с G и A с Т.

Прочитав только несколько символов начиная с начальной ДНК и обратного набора ДНК из первой попытки вы увидете, что первая попытка рассчета обратного дополнения не удалась. Это был неверный алгоритм.

С таким Вы иногда будете встречаться во время программирования. Вы напишите программу, завершите работу, а затем обнаружите, что она не работает так, как вы ожидали. В первом случае мы использовали возможности языка, с которыми уже были знакомы, и попытались их применить для новой задачи. Но это не сработало. Что было не так?

Вы обнаружите, что случай, когда вы пишете код, а он не работает, становится привычным! Поэтому, Вы или отлаживаете синтаксис (что обычно легко может быть сделано по сообщениям об ошибках), или тщательнее обдумываете программу, находите причину того, почему она не работает, а затем пытаетесь разработать новый и более успешный вариант. Часто для этого требуется просмотр документации языка программирования, поиск детальной информации о том, как устроен язык программирования, в надежде найти возможность устранить проблему. Если это может быть решено на компьютере, значит возможно решить и с помощью Perl. Вопрос в том, как именно?

В примере 4-4 первая попытка вычислить обратный набор не удалась. Каждая основа в символьной строке была транслирована как целая, используя 4 замены в глобальном виде. Нужен другой способ. Вы можете продвигаться по ДНК слева направо, смотреть на каждую основу друг за другом, делать изменения в дополнительном коде и затем смотреть на следующую основу в ДНК, передвигаясь к концу символьной строки. Затем просто развернуть символьную строку. Это на самом деле действенный метод и его совсем не сложно выполнить в Perl, хотя потребуются некоторые знания языка программирования, которые до Главы 5 не встречались.

Как бы ни было, в этом случае оператор *tr*, означающий трансляцию или перевод, точно подходит для решения задачи. Он выглядит так же, как оператор подстановки, с 3 косыми чертами, разделяющими различные части.

*Tr* делает точно то, что нужно. Он переводит один набор символов в другой набор сразу целиком. На Рисунке 4-2 показано, как он работает: набор переводимых символов находится между первыми двумя косыми чертами. Набор символов, которые замещают исходные, находится между второй и третьей косыми чертами. Каждый символ первого набора переводится в символы, что на тех же позициях во втором наборе. Например, в Примере 4-4 C – это второй символ в первом наборе, и он переводится во второй символ второго набора, а именно в G. Наконец, так как цепочка ДНК может кодироваться и заглавными, и строчными буквами (не смотря на то, что в этой программе ДНК представлена только заглавными буквами), оба случая в Примере 4-4 включены в *tr* оператор.

*Figure 4-2. The tr statement*

Функция reverse реализуется так же с минимумом усилий. Как можно видеть из Примера 4-4, она создана, чтобы изменить порядок элементов, включая символьные строки, на обратный.

## 4.7. Белки, файлы и массивы

Пока мы создавали программы с последовательностями ДНK. Сейчас мы добавим к ним не менее важные последовательности белка. Вот кратко то, что описано в следующих разделах:

Как использовать данные белковой последовательности в Perl программе

Как считать данные белковой последовательности в файл

Массивы в языке Perl

В оставшихся главах используются и ДНК последовательности, и белковые последовательности.

## 4.8. Считывание белков из файлов

Программа взаимодействует с файлами на диске компьютера. Эти файлы могут быть на жестком диске, CD, дискете, Zip накопителе, магнитной ленте – любом виде постоянного запоминающего устройства.

Давайте рассмотрим, как считывать данные белковой последовательности с файла. Для начала создайте файл на вашем компьютере (используйте текстовый редактор) и вставьте в него данные какой-нибудь белковой последовательности. Назовите файл NM\_021964fragment.pep (вы можете скачать его с сайта этой книги). Вы будете использовать следующие данные (часть белка указателя человеческого цинка NM\_021964):

Вы можете придумать любое название, кроме тех, которые уже использованы в той же папке.

Хорошо подобранные названия файлов и папок могут быть так же значимы, как и хорошо подобранные названия переменных важны для понимания программы. Если Ваш проект генерирует множество компьютерных файлов, Вам нужно тщательно продумывать, как называть и организовывать файлы и папки. Это необходимо как для индивидуальных разработок, так и для командных и многонациональных.

Имя файла NM\_021964fragment.pep взято из GenBank ID, где и был найден этот белок. Оно также обозначает фрагментарный тип данных и содержит расширение файла .pep, давая понять, что файл содержит пептид и белковую последовательность. Конечно, может быть вам могли бы подойти лучше другие схемы, но цель этого - понять что находится в файле перед просмотром содержимого.

Сейчас, когда вы создали или скачали файл с белковой последовательностью, давайте разработаем программу, которая считывает данные белковой последовательности из файла и сохраняет их в переменную. В Пример 4-5 показана первая попытка, которая будет дорабатываться в процессе.

*Example 4-5. Reading protein sequence data from a file*

Заметьте, что распечатывается только первая строка файла. Сейчас я разъясню почему.

# Vocabulary

**A**

**abbreviation** - сокращение (напр., текста); аббревиатура

**absolute value** - абсолютное значение, абсолютная величина, модуль (числа)

**abstraction** - абстракция, отвлечение

**access** - доступ; обращение (напр., к памяти ЭВМ, базе данных, каталогу) // обращаться

**accession** - 1. поступление; 2. инвентаризация; регистрация (новых поступлений)

**acknowledgement** - подтверждение приема (сообщения)

**acronym** - 1. акроним, слово, образованное по первым звукам или слогам словосочетания, которое оно заменяет 2. Преим. в СМС-сообщениях и электронной почте: слово, образованное из букв и цифр, заменяющих более длинные слова или целые словосочетания: например, L8R = later, CU = see you

**action** - 1. действие; воздействие; 2. деятельность; линия поведения; 3. операция

**adapt** - адаптировать, приспосабливать

**addition** - сложение, суммирование; прибавление

**advanced** - 1. улучшенный, усовершенствованный; 2. перспективный, прогрессивный; развитый

**agent** - 1. агент; представитель; посредник; доверенное лицо; 2. агент (субъект действия); 3. действующая сила; фактор

**allow** – разрешать, предоставлять

**alphanumeric** - буквенно-цифровой

**ambiguity** - неопределенность; неясность; неоднозначность

**amount** – количество, величина; равняться

**answer** - ответ; реакция

**appear** - казаться

**appendix** - приложение

**application** - 1. применение; использование; 2. прикладная задача

**approach** - 1. подход; 2. метод; принцип

**aptitude** - 1. пригодность; соответствие; 2. способность выполнять (определенную) работу

**arbitrary** - произвольный

**archive** - архив; хранилище

**area** - 1. (предметная) область; 2. поле (записи); 3. площадь

**argument** - 1. аргумент, доказательство, суждение; 2. аргумент, независимая переменная; параметр

**array** - 1. массив; 2. строй; 3. таблица, матрица, решетка;

**artificial intelligence (AI)** - искусственный интеллект

**assertion** - 1. утверждение; 2. суждение

**assign** - определять, присваивать, назначать; **assignment statement** – оператор присваивания

**associate** – ассоциировать, связывать (типы файлов с приложениями)

**assume** - предполагать

**atom** - элементарный объект, атом

**authority** - 1. полномочия; право; власть должностного лица; 2. уполномоченное лицо; уполномоченный орган; 3. нормативный [справочный] материал (напр., указатель)
**authority control** - нормативный контроль

**B**

**background** - 1. фон; фоновая работа // фоновый; низкоприоритетный; 2. пробельные места (на фотокопии, микроформе и т. п.); 3. подготовка; предварительные знания; предпосылка; 4. истоки; происхождение; биографические данные; 5. теоретические основы

**backspace** - возвращаться; перемещать обратно; реверсировать; возвращать на одну позицию

**backup** - 1) опора; опорный 2) архивирование, архивация 3) резервирование; резервный ресурс 4) (техническая) поддержка;

**bar** - (горизонтальная) полоса; барьер

**base** - 1. база; основа // базировать(ся); основывать(ся); 2. база (данных); 3. корень (слова); 4. основание; подложка

**behave** - 1) вести себя, поступать, держаться 2) работать без перебоев (о машине)

**behavior** - 1. поведение, линия поведения (пользователя); 2. режим (работы)

**benchmark** - 1. точка отсчета, начало отсчета; контрольная точка (для сравнения); 2. контрольная задача (для определения сравнительных характеристик производительности системы); 3. аттестация (программного изделия)

**benefit** - выгода, прибыль

**binary search** - двоичный [дихотомический] поиск

**binding operator** - связанный оператор

**bioinformatics** - биоинформатика (наука, использующая достижения биологии для разработки совершенно новых практических подходов для решения проблем здравоохранения и сельского хозяйства, для разработки принципиально новых технологий и материалов в различных отраслях народного хозяйства)**;** применение ЭВМ в биологии

**blank** - 1. пробельный материал; пробел; 2. пустая (незаполненная) страница; 3. бланк; 4. тире (вместо пропущенного или нецензурного слова); 5 заготовка

**block** - 1. блок; узел // блокировать // блочный; 2. группа, блок (напр., слов, чисел или знаков);

**boolean** - 1. булево выражение // булев; 2. логический

**borrow** - заимствовать, занимать

**bracket** - скобка // заключать в скобки

**branch** - 1. ветвление, (условный) переход (в алгоритме или программе); 2. ветвь (напр., алгоритма); 3. отделение, филиал (напр., фирмы)

**breakdown** - 1. распределение; расчленение; деление на категории; классификация; 2. поломка; авария

**brief** - краткое изложение; резюме; сводка; бюллетень //составлять краткое изложение; резюмировать

**browse** – просматривать

**bug** - 1. ошибка; дефект; 2. помеха

**built-in** – встроенный, свойственный, врожденый

**bunch** – связка, пучок, сгусток, пачка, букет

**C**

**call** - 1. вызов; обращение // вызывать; обращаться; 2. запрос// запрашивать; 3. переход к подпрограмме; 4. заявка

**capability** - 1. возможность; способность; 2. производительность; 3. характеристика

**capital letter** - заглавная [прописная] буква

**capture** - 1. сбор (данных)//собирать (данные); 2. фиксировать [останавливать] (изображение)

**carriage return** - обратный ход каретки; возврат каретки

**case** – наборная касса; **uppercase / lowercase** – верхний / нижний регистр
1. случай; обстоятельство; 2. падеж; 3. оболочка; кейс

**cell** - 1. ячейка; элемент; 2. (фото)элемент; 3. секция (единый участок памяти в файле)

**chain** - 1. цепь; цепочка; 2. последовательность; **chaining** - связывание; формирование цепочки; сцепление (программ)

**chapter** - 1. глава; 2. раздел; секция; сегмент (программы)

**character** - знак; символ; цифра; буква; иероглиф; **character set** - набор символов

**characterization** - определение [установление] характеристик, особенностей, свойств

**chart** - диаграмма; схема; таблица; график; карта

**check** – проверка; контролировать; **checking** - контроль; испытания

**chip** - 1. интегральная схема, ИС; чип; 2. микрокадр (на микрофише)

**chunk** - 1. блок (информации); 2. проф. кусок программы (на блок-схеме)

**class** - класс; группа; категория // класс ООП

**classification** - классификация, шифр [имя] класса

**clear** – четкий; очищать

**click** - нажимать клавишу (манипулятора "мышь");

**clone** - обеспечивать абсолютную совместимость

**clue** - ключ; подсказка

**cluster** - кластер//образовывать кластер (группу)

**coding** - программирование, кодирование (на машинном языке); кодировка;

**column** - колонка; столбец; графа

**combination** - соединение; комбинация; сочетание

**comma-separated** – список с разделителями-запятыми

**command** - 1. команда (см. также instruction); предписание; директива; 2. управлять

**command interpreter** - интерпретатор команд

**command prompt** - приглашение (пользователя) ко вводу команды

**command window** - командное окно (отображает параметры текущего состояния системы и может использоваться для ввода команд в качестве альтернативы их выбору в диалоговых окнах)

**comment** - комментарий; примечание;

**common name** - общее имя (нарицательное)

**compatibility** - совместимость

**compendium** - 1. компендиум, краткое руководство; 2. конспект; резюме

**compile** - 1. компилировать, составлять; 2. собирать (материал, факты и т.п.)

**compiler** - компилятор, транслятор программа, выполняющая трансляцию исходного текста разрабатываемой программы с языка программирования высокого уровня в эквивалентную программу на целевом языке ( target language ).

**complement** – дополнение // дополнять

**complete** – завершать // законченный

**computation** - вычисление, расчет; счет, подсчет

**computer language** - машинный язык

**computer power** - вычислительная мощность ЭВМ, вычислительный ресурс

**concatenation** - 1. связь, соединение; сочленение; 2. каскадное включение; 3. конкатенация, сцепление

**consist** - состоять из; заключаться в

**condition** - 1) условие 2) состояние 3) обусловливать, определять 4) кондиционировать (воздух) 5) режим

**configure** - 1) придавать форму 2) компоновать (станок) 3) изменять структуру (напр. системы управления)

**conjunction** - 1. конъюнкция, логическое умножение; 2. связь; соединение

**connection** - соединение; связь; связка; присоединение; включение; подключение

**consistency** - 1. согласованность; 2. непротиворечивость

**construct** - 1. логическая структура; 2. составлять; создавать

**contain** – содержать; **content** - 1. содержание; 2. pl оглавление

**context** - определённый смысл (напр. тип значения переменной); режим работы или состояние системы; окружение, среда исполнения программы;

**control** - управление; // регулировать; устройство управления; контроль

**control character** - управляющий символ

**control flow** - последовательность комад

**conversion** - преобразование; конвертирование; перенесение (данных с одного носителя на другой);

**core** - сердечник; ядро

**correlation** - корреляция, соотношение, взаимосвязь

**coverage** - охват; зона действия

**current** – текущий ~ **directory** - текущий каталог; ~ **value** - текущее значение

**curve** - кривая, график; траектория перемещения

**cycle** - цикл; период // циклически повторять(ся)

**D**

**DNA** – ДНК (дезоксирибонуклеиновая кислота)

**data** - данные; сведения; информация; **data analysis** - анализ данных; **data set** - набор данных

**database** - база данных

**decimal** - десятичный

**decision** - 1. решение; 2. выбор

**declare** – объявлять (переменную); утверждать

**default** – умолчание // подразумеваемый

**define** - задать (процедуру) ; определить, описать (переменную); **definition** - толкование; четкость, разрешение (изображения)

**delimiter** - разделитель символ (пробел, табуляция, запятая и т.п.); ограничитель

**demand** - запрос // запрашивать; требовать; потребность; опрос

**demo** - демонстрационный пример (работы системы)

**denied** – отказывать; отрицать

**depend** – зависеть; положиться

**description** - описание; характеристика; инструкция

**design** - 1. проектирование; конструирование; разработка // проектировать; конструировать; разрабатывать; 2. проект; замысел

**destination** - адресат информации; пункт назначения (передаваемого сообщения)

**development** - 1. разработка; развитие; совершенствование; 2. проявление (изображения)

**device** - 1. устройство; приспособление; аппарат; прибор; 2. способ; средствo; прием; 3. определитель (в классификации)

**difference** - 1. разность; 2. видовое отличие

**digit** - 1. цифра; разряд; 2. символ; знак

**directive** - указатель; директива // управляющий

**directory** - 1. руководство; 2. каталог; справочник; список адресов

**discard** - брак // отбраковывать

**display** - дисплей; (визуальное) представление, отображение, вывод (данных); изображение (на экране)

**distribution** – распределение (напр. мат.)

**division** - подразделение; подкласс; деление

**domain** - (предметная) область, домен, владение

**drive** - 1. движущий механизм // двигать 2. накопитель (на дисках или ленте)

**drop-down** – выпадающий (список, окно)

**duplex** – двухсторонний; спаренный

**E**

**echo** - эхо, повторное изображение (мешающее восприятию основного)

**edition** – выпуск; издание

**editor** - редактор, редактирующая программа

**effect** - 1. влияние; действие; воздействие; 2. эффект; следствие; результат

**efficiency** - 1. экономическая эффективность; производительность; продуктивность; прибыльность; 2. коэффициент полезного действия, к.п.д.; коэффициент использования

**eliminate** - устранять, аннулировать, исключать (неизвестное)

**embed** - вставлять

**empty** – пустой, полый; **empty list** - пустой список

**encapsulated** – инкапсулированный, содержащий внутри себя ещё что-нибудь

**encoding** - кодирование; шифрование

**enter** – ввод // вносить, вступать

**entity** - 1. сущность; существо; 2. вещь; объект ; **entire** - целостный

**entry** – 1. описание; запись 2. проникновение; 3. вхождение, элемент

**environment** - 1. окружающая среда; окружение, (окружающая) обстановка, условия; 2. контекст; 3. конфигурация (напр., сети или системы)

**equipment** - оборудование; приборы; аппаратура

**erasing** - стирание (записи); разрушение (информации)

**error** - ошибка; погрешность

**error message** - сообщение об ошибке

**escape character** - знак перехода (напр., с одного регистра на другой); знак смены регистра; знак переключения кода

**eval** – сокр. от evaluation 1) оценка 2) вычисление

**event** - 1. событие; 2. исход, результат

**evidence** - факты; свидетельства

**exception** - исключение; исключительная ситуация

**exchange** - 1. обмен; замена // обмениваться; заменять; 2. коммутационная станция; коммутатор; 3. автоматическая телефонная станция, АТС; 4. телефонная сеть

**exclamation mark** - восклицательный знак

**execution** - исполнение, выполнение (программы, команды)

**exists** – быть, выживать

**explanation** - 1. объяснение, разъяснение; 2. толкование

**explicit** - подробный, точный; определенный

**explore** - исследовать; изучать

**expression** - 1. выражение (какого-либо признака) ; экспрессия 2. отжимание, выжимание, выдавливание

**extension** - 1. расширение (напр., имени объекта); 2. продолжение, распространение(индексов); 3. объем (понятия); лог. экстенсия, протяженность

**extent** – 1. непрерывная область (напр., в памяти, резервируемая для данных); 2. степень; мера

**extra information** - дополнительная информация

**extract** - выдержка; извлечение // извлекать; выбирать;

**extraneous information** - посторонняя информация

**extreme programming** - XP экстремальное программирование, одна из методологий разработки ПО

**F**

**facility** - 1. устройство; 2. pl средства; оборудование; аппаратура; техническая база; 3. pl функциональные возможности; 4. удобство

**fail** - 1. повреждение; неисправность; сбой; отказ в работе // повреждаться; выходить из строя; отказывать в работе; 2. не достигать успеха (напр., в поиске информации)

**failure** - 1. повреждение; неисправность; сбой; отказ в работе; 2. неудача, неблагоприятный исход

**faster** - ускоритель (напр., логического поиска)

**feature** - 1. признак; свойство; особенность 2. функция; функциональная возможность

**feature based** - основанный на выделении признаков

**feedback** - обратная связь

**field** - 1. поле; пространство; область; зона; 2. поле, группа разрядов

**figure** - 1. цифра // обозначать цифрами; 2. число; 3. фигура; рисунок; чертеж

**file** - 1. (поисковый) массив; 2. файл, набор данных (в машиночитаемой форме) // формировать [организовывать] файл; заносить в файл; хранить в файле

**file attribute** - атрибут файла поддерживаемый файловой системой признак, позволяющий упорядочить работу с файлами.

**filehandle** - логический номер (индекс) файла, описатель файла

**filter** - 1. фильтр // фильтровать; 2. программа фильтрации

**fix** - 1. устанавливать; укреплять; закреплять; 2. фиксировать, закреплять (изображение); 3. выявлять [регистрировать] местоположение (напр., ошибки) // определение местоположения (напр., ошибки)

**flag** - 1. флаг, флажок; 2. признак; идентификатор; 3. помечать

**flow control** - управление потоками (данных)

**folder** - папка (директория)

**font** – шрифт

**forbidden** - запрещенный

**form** - 1. форма; 2. бланк; формуляр; 3. представление (информации в памяти машины); 4. страница (распечатки или печатного документа); **form of output** - форма выдачи (данных)

**fragmentation** - фрагментация

**frame** - 1. группа [блок] данных; 2. кадр на ленте; кадр (изображения); 3. фрейм (в искусственном интеллекте); 4. система координат; система отсчета; 5. рамка; (фото)копировальная рама

**free** - 1. свободный, несвязанный (в координатном индекси-ровании); 2. бесплатный

**frequency of occurrence** - частота встречаемости

**function** - 1. функция, назначение // функционировать, действовать; 2. функция (математическая); 3. функциональное подразделение

**functional unit** - функциональное устройство; функциональный блок, блок (реализации) функции; функциональный элемент

**G**

**GC** - генетический код

**gain** - 1. прирост, увеличение; 2. выигрыш, выгода; 3. получать, приобретать;

**gap** - интервал, промежуток; пропуск; пробел, лакуна

**gathering** - 1. комплектование [подборка] листов или тетрадей; 2. печатный лист; тетрадь (книжного блока); сфальцованный печатный лист; 3. сбор // собирать

**general rule** - общее правило

**generation** - 1. создание; образование; формирование; 2. генерирование; порождение; 3. поколение (напр., микрокопии)

**genetic algorithm (GA)** - генетический алгоритм (попытки намешать наиболее перспективные варианты решений из некоторого первоначального набора вариантов)

**genetic background** - генетическая среда

**genetic carrier** - носитель рецессивного гена

**genetic code overlapping** - перекрывание генетического кода

**genetic engineering** - генная инженерия область молекулярной биологии, требующая громадных вычислительных ресурсов

**genus** - род

**get in** - 1. уплотнять текст; 2. вгонять строку (напр., уменьшением интервалов)

**gibberish** - ненужные данные; ненужная информация (в памяти), проф. мусор, "мякина"

**goal** - цель; целевая установка

**grade** - класс; степень; ранг

**graph** - 1. график; диаграмма; кривая; номограмма; 2. граф

**guessing** - (нестрогие) рассуждения с использованием догадок (в системе искусственного интеллекта)

**guide** - 1. руководство, путеводитель; 2. разделительная карточка, разделитель // снабжать разделителями

**guidelines** - руководящие принципы

**H**

**hand** - 1. почерк; 2. подпись; 3. указательный знак в виде изображения руки; 4. подавать, вручать

**handler** - 1. блок управления; манипулятор; 2. программа обработки, обработчик

**hard disk** - жесткий магнитный диск

**hash** - тип данных Хэш 1. ненужные данные (мусор в памяти); помехи 2. перемешивать

**head** - заголовок; рубрика; "шапка" (см. также header, heading)

**health** - состояние; степень исправности

**hierarchical** - иерархический

**highlight** - 1. наиболее яркая область изображения; 2. блик; 3. высвечивать, выделять (часть текста или изображения на экране дисплея)

**hint** – подсказка, намек

**hit** - 1. ответ, ответная справка (при поиске); 2. совпадение (при поиске); 3. импульсная помеха; 4. вспышка // вспыхивать; 5. ударять (по клавише)

**holding** - хранение (информации)

**home directory** - начальный каталог

**human knowledge** - человеческие знания

**I**

**icon** - условное графическое представление, пиктограмма

**identification** - 1. идентификация; отождествление; 2. определение; распознавание; 3. обозначение

**identifier** - идентификатор; имя

**identity** - 1. идентичность, тождественность; 2. тождество

**include** – заключать (напр. файл в проект)

**index** - 1. индекс; указатель // снабжать указателем; индексировать; 2. указатель; каталог; картотека // составлять указатель; заносить в указатель; 3. показатель степени; коэффициент

**infer** - делать логический вывод

**informatics** - 1. информатика (научная дисциплина, изучающая структуру и свойства информации, историю ,методику и организацию ее сбора, обработки, хранения, поиска, распространения и использования во всех сферах деятельности) см.также information science; 2. автоматизированная обработка данных

**initial** - 1. инициал; 2. начальный, исходный

**initialize** - 1. инициализировать; 2. устанавливать в начальное [исходное] состояние; 3. задавать начальные условия

**input/output** 1) ввод-вывод 2) устройство ввода-вывода 3) данные ввода-вывода; 4) подавать на вход / выход
**~ file -** входной файл / выходной файл

**insert** - вставка // вкладывать; вставлять

**installation** - 1. установка; устройство; 2. установка, монтаж; 3. расположение, размещение

**instance** - 1. пример; 2. экземпляр (напр., объекта базы знаний)

**instruction** - 1. команда (см. также command, order); 2. инструкция; программа действий; 3. обучение

**integer** - целое число // целый (о типе данных)

**integral** - 1. целый; целостный; 2. объединенный; интегральный; 3. неотъемлемый

**intelligence** - 1. ум, интеллект; 2. сведения, сообщения; информация; 3. развитые; логические функции или возможности; 4. встроенные микропроцессорные средства; 5. разведка

**interact** - взаимодействовать

**interface** - 1. интерфейс; устройство сопряжения, сопряжение; средство сопряжения; 2. сопряжение; согласование // сопрягать; согласовывать; 3. взаимодействие // взаимодействовать

**interpolate** - 1. интерполировать; 2. делать вставки (в схему классификации или текст); вставлять слова или замечания

**interpreter** – интерпретатор - программа или среда программирования, в которой можно выполнить трансляцию исходного текста разрабатываемой программы и её выполнение без получения соответствующего машинного кода.

**introduction** - 1. введение; внесение; 2. представление; 3. предисловие

**issue** – 1. проблема, исход; 2. выпуск; издание // выпускать

**italic** - l. курсив; 2. курсивный (о шрифте)

**item** - 1. элемент; единица; элементарная группа; 2. пункт; параграф; статья; 3. печатная единица; единица хранения (в библиотеке); 4. элемент данных; 5. газетная заметка; сообщение

**iteration** - 1. повторение, итерация; 2. шаг (в итеративном процессе); 3. цикл

**J**

**join** - 1. объединение; операция включающая ИЛИ; 2. соединение (операция над отношениями в реляционных базах данных) // выполнять операцию соединения, выполнять соединение (над отношениями в реляционных базах данных); 3. объединять; соединять

**journal** - 1. журнал (лог, регистрация работы системы) 2. журнал, газета

**justification** - 1. оправдание; подтверждение; объяснение; 2. вертикальное выравнивание текста

**K**

**key** - 1. ключ; кнопка; клавиша; переключатель; 2. ключ, шифр, код; указание к решению; 3. переключать; работать ключом; коммутировать

**keyboard** - 1. коммутационная панель; 2. клавиатура, набирать на клавиатуре

**keyword** - ключевое слово

**kind** – качество, сорт, признак

**kit** - набор; комплект (напр., материалов по какому-либо предмету)

**L**

**label** - ярлык; этикетка; метка // помечать, обозначать, маркировать

**lag** - 1. запаздывание, задержка, отставание; 2. интервал (времени)

**layout** - 1. размещение; расположение; компоновка; 2. формат (напр., документа); 3. макет

**leaves** - листья (записи самого нижнего уровня иерархической базы данных)

**letter** - 1. буква; символ; знак // помечать буквами; 2. литера; шрифт; 3. письмо

**level** - 1. уровень // устанавливать [регулировать] уровень; 2. степень

**life** - 1. долговечность; срок службы; 2. ресурс

**limit** - предел, граница // ограничивать, служить границей

**line** - линия; ряд; строка

**line break** - концевая строка (абзаца, страницы)

**link** – 1. ссылка (на страницу в интернете); 2. связь, звено //связывать

**list** - 1. список // вести [составлять] список; 2. перечень; перечисление // составлять перечень; перечислять; 3. таблица

**listing** - листинг, распечатка; в программировании - распечатка исходного текста программы (или результатов её работы) на бумаге или отображение его на экране

**loading** - 1. загрузка (памяти или в память); 2. заправка (ленты); 3. зарядка (кассеты, магазина)

**location** - 1. расстановка; размещение; расположение; 2. местонахождение; местоположение; 3. ячейка (ЗУ); 4. адрес ячейки

**log** - 1. регистрация, запись (информации); протокол // регистрировать, записывать (информацию); протоколировать; 2. журнал регистрации

**look** - просмотр: поиск; шаг поиска;
**to look up** - искать; отыскивать

**loop** - 1. петля; 2. контур; (замкнутая) цепь; 3. цикл

**M**

**Mac** - макинтош

**machine** - 1. Машина; механизм; устройство, аппарат // машинный; 2. Вычислительная машина; счетная машина (см. также computer); 3. редк. процессор

**mailing list** - почтовый реестр

**main memory** - основная память, основное запоминающее устройство; оперативная память, оперативное запоминающее устройство; ОЗУ

**maintenance** - (техническое) обслуживание; поддержка; эксплуатация (системы); сопровождение (напр., системы программного обеспечения); ведение (напр., массива)

**management** - 1. руководство; управление; организация; 2. руководство, руководящий состав

**map** - 1. карта; план; схема // наносить на карту; составлять карту или схему; 2. карта (распределения); 3. отображение, соответствие // отображать, устанавливать соответствие; 4. преобразовывать данные (из одной формы в другую)

**mark** - 1. метка, маркер; знак // отмечать; маркировать; обозначать, ставить знак; 2. штамп, штемпель; фабричная марка, фабричное клеймо // штамповать, штемпелевать

**match** - 1. совпадение (признаков); равносильность; 2. сравнивать, сопоставлять; отождествлять, приводить в соответствие

**matter** - 1. материал; 2. сущность; содержание; 3. предмет (обсуждения и т. п.); 4. рукопись; оригинал; 5. напечатанный [набранный] материал (текст)

**measure** - 1. мера // мерить, измерять; 2. предел; степень; 3. масштаб; мерило; критерий; 4. ширина столбца

**memorize** - 1. запоминать, хранить; 2. передавать в память

**message** - сообщение; посылка, передаваемый блок информации

**metacharacter** – метасимвол( metasymbol ), символ метаязыка

**miss** – потеря, неудача, пропуск

**misspelling** - орфографическая ошибка, написание с орфографическими ошибками

**mode** - 1. способ; метод; принцип (работы); 2. режим (работы); 3. состояние; 4. вид; класс; 5. мода, наиболее вероятное значение

**modify** - модифицировать; изменять; видоизменять; **modifier** - модификатор

**multiple** - 1. кратное (число)//кратный; 2. многократный, множественный, многочисленный

**N**

**negation** - отрицание (логическая функция или операция)

**nesting** - 1. формирование гнезда; осуществление вложений, вложение; 2. вложенность; 3. рекурсивное построение сложных схем (на базе разработанных)

**notice** - 1. заметка; объявление; помета // замечать; 2. обозрение; рецензия // рецензировать; 3. извещение, уведомление; предупреждение // извещать, уведомлять; предупреждать

**O**

**obviously** - явно, понятно, тривиально

**octal** - восьмеричный

**open source code** - открытые исходные тексты, открытые исходники принцип, заключающийся в том, что исходный код разрабатываемой системы должен свободно и бесплатно предоставляться всем желающим её усовершенствовать

**operand** - операнд, объект (действия)

**operation** - 1. операция; действие; 2. работа; функционирование; 3. режим (работы); 4. срабатывание (прибора); 5. редк. управление

**operator** - 1. оператор; операция; 2. оператор (человек)

**option** - 1. выбор; 2. вариант, версия; элемент выбора, (факультативная) возможность; вариант дополнения

**order** - 1. порядок (расстановки), последовательность/упорядочивать, приводить в порядок; 2. порядок (величины); кратность, степень; 3. команда; 4. заказ // заказывать
**ascending** - возрастающий порядок; **descending** – убывающий

**ordered set** - упорядоченное множество

**original** - 1. оригинал, подлинник // подлинный; 2. первоисточник // первоначальный; исходный

**out of date** - устаревший, вышедший из употребления

**outline** - 1. контур; очертание; 2. схема; план; 3. элемент блок-схемы

**P**

**pair** - пара // соединять попарно; подбирать под пару // парный; двойной

**parent** – предок, родитель, источник

**parentheses** - круглые скобки

**parse** - синтаксический анализ, синтаксический разбор; грамматический разбор // анализировать, разбирать

**partition** - 1. раздел; часть; сегмент; сектор; 2. расчисление; разделение; разбиение // расчленять // разделять, разбивать

**path** - 1. путь; дорожка; тракт; 2. маршрут (в сети ); 3. цепь; ветвь (программа); 4. траектория

**pattern** - 1. образ, изображение; копия; 2. образец; шаблон; модель; стереотип (напр., поведения); 3. схема; структура; 4. (конкретный) набор; (конкретная) комбинация (напр., символов); 5 конфигурация

**permit** - разрешение

**plain text** - открытый (незакодированный) текст

**plant** - 1. помещение (напр., библиотеки); 2. объект управления; 3. прятать (команду в памяти)

**point** - 1. точка, пункт, место; 2. точка (знак); 3. запятая (в дробях); 4. ставить знаки препинания; 5. указывать, показывать

**pointer** - 1. указатель; 2. стрелка (напр., часов, измерительного прибора и т.п.)

**policy** - политика; стратегия

**population** - 1. совокупность; 2. заполнение (файла)

**position** - 1. позиция; (место) положение// позиционировать; 2. место (цифры); разряд (числа); 3. должность (в организации)

**precision** - точность

**predictable** - прогнозируемый, предсказуемый

**pretty printer** - программа структурной распечатки, которая анализирует исходный текст программы и печатает его таким образом, чтобы была ясно видна структура программы, например выделяет отступами уровни вложенности циклов

**printout** - распечатка, твердая копия, листинг (программы)

**privacy** - секретность, конфиденциальность (информации); личная тайна (характер информации); сохранение тайны (при хранении информации)

**processing** - обработка данных; **preprocessing** - предварительная обработка

**produce** – продукция // выводить; **reproduce** – воспроизводить

**programming methodology** - методология программирования совокупность взглядов на организацию разработки ПО и его логическую структуру, воплощенная в виде совокупности инструментальных и языковых средств, поддержанная принятой в организации нормативной базой, системой обучения и сложившейся корпоративной культурой

**programming style** - стиль программирования разновидность той или иной методологии программирования, используемая при кодировании программы

**programming techniques** - методы программирования; методология программирования

**project** - 1. проект; 2. проецировать

**prompt** - подсказка (пользователю со стороны диалоговой системы) // подсказывать

**property** – свойство, право, реквизит; **property sheet** - ведомость свойств (напр., подлежащего форматированию текста в текстовом редакторе)

**protein** - белок

**Q**

**qualifier** - определитель, классификатор

**quality** - качество

**quantity** - количество

**query** - запрос (информационный)

**query language** - язык запросов

**quote** – 1. цитировать // выдержка; 2. кавычки

**R**

**RNA** – (RiboNucleic Acid) рибонуклеиновая кислота

**random** - случайный, выбранный наугад; произвольный

**range** - 1. область; диапазон; интервал; ряд

**rank** - ранг // ранжировать, располагать в определенном порядке, устанавливать очередность // ранговый; котироваться

**rapid access** - быстрый доступ

**readability** - 1. читабельность; 2. удобочитаемость

**real time** - 1. реальное время; реальный масштаб времени // работающий в реальном (масштабе) времени; 2. истинное время; истинный масштаб времени // протекающий в истинном (масштабе) времени

**recall** - 1. отыскание в памяти, вызов из памяти // отыскивать в памяти, вызывать из памяти, вспоминать; 2. напоминание // напоминать

**recognize** - распознавать, опознавать; различать

**record** - 1. запись; регистрация (фактов) // записывать, регистрировать; 2. протокол (заседания и т.п.) // протоколировать, заносить в протокол; 3. официальный документ, запись, отчет; реестр; 4. граммофонная пластинка, запись на граммофонной пластинке; 5 запись (структурная единица информации); 6. зона (на магнитной ленте)

**redundancy** - 1. избыточность; 2. резервирование; 3. избыточное оборудование, резерв

**refer** - 1. посылать, отсылать, направлять; 2. обращаться, справляться; 3. относиться, иметь отношение; 4. ссылаться

**reference** - ссылка; сноска

**regular expression** - регулярное выражение

**relational** - 1. реляционный; 2. родственный (о данных); 3. относительный

**release** - опубликованный материал // выпускать (из печати и т.п.); release candidate (RC) предвыпускная версия, сдаточная версия версия ПО, собранная перед выпуском нового релиза. Над этой версией уже не ведутся никакие работы, кроме тестирования и подготовки к сдаче

**relevant** - релевантный, соответствующий информационному запросу

**remote** - дистанционный, удаленный

**remove** – смещать, устранять // удаленность

**replacing** - замена, замещение; подстановка; перестановка

**replication** – 1. возражение, ответ; 2. отражение; репродуцирование

**repository** - хранилище данных; информационный архив

**representation** - 1. представление (данных, знаний, изображений); изображение, отображение; 2. способ задания (функций)

**request** - 1. запрос; заявка // запрашивать; 2. требование; просьба // просить (о чем-л.)

**research and development** - (научные) исследования и разработки

**reserved word** - зарезервированное слово

**reset** - восстановление, сброс

**resist** - защитный слой // противостоять

**response** - 1. ответ, ответное действие, реакция; 2. характеристика; зависимость; 3. восприимчивость, чувствительность

**restriction** - 1. ограничение; 2. препятствие, помеха

**retrieve** - (снова) найти; отыск(ив)ать

**return** – 1. возврат // возвращать 2. оборот; доход

**reverse** – обратный // перевертывать

**review** - обозрение // рецензировать

**root** – корень (мат., системы); **root user** - привилегированный пользователь

**row** - ряд; строка (матрицы или многомерного массива)

**rule of thumb** - практическое правило, эмпирическое правило; практический прием (в отличие от научного)

**run** - 1. (однократный) проход, прогон (программы) 2. работа, ход (машины)

**S**

**sample** - образец; эталон

**scalar** - скалярная величина

**scale** - шкала; масштаб // масштабировать; 2. система счисления

**scheduling** - составление расписания, графика или (календарного) плана; планирование; распределение (напр. машинного времени)

**scheme** - 1. план; проект, программа; 2. схема, чертеж; 3. конспект; краткое изложение

**scope** – границы; сфера; масштаб; область видимости

**screen** - 1. экран; щит // прикрывать 2. изображение (на экране) // отображать на экране; 3. сито, фильтр // просеивать

**script** – подлинник; сценарий; почерк; script language - язык сценариев

**select** - выбирать, отбирать, подбирать; выделять

**semicolon** - точка с запятой

**sense** - 1. значение (слова); 2. считывание // считывать; 3. опознавание; восприятие // опознавать; воспринимать

**sentence** - предложение

**separator** - 1) тот, кто разделяет, отделяет, разъединяет 2) разделитель, разделительный знак

**sequence** - последовательность; порядок (следования) // устанавливать последовательность; упорядочивать

**set** - 1. набор, комплект; 2. множество; совокупность; 3. радиоприемник; телевизионный приемник, телевизор; 4. станция; установка; аппарат; устройство; 5. установка; регулирование; настройка // устанавливать; монтировать; регулировать; настраивать; 6. набор // набирать (текст); 7. комплект; гарнитура (шрифта); 8. психол. настрой

**setting** - 1. установка; регулирование, регулировка; настройка; 2. установка в определенное положение или состояние; 3. набор (текста)

**share** - 1. доля; часть; участие // разделять; совместно [коллективно] использовать; участвовать; 2. акция; пай // быть пайщиком

**shell** - оболочка

**shift** - 1. сдвиг; смещение // сдвигать; смещать; 2. смена регистра; 3. (рабочая) смена

**sign** - 1) знак; символ

**significance** - 1. значение, смысл (напр. слова); 2. значимость

**simulation** - моделирование; имитация; имитационное моделирование

**site** - 1. местоположение; 2. веб-сайт

**size** - 1. размер; 2. объем (выборки); 3. емкость (ЗУ)

**slash** - косая черта (символ или математический знак)

**source** – источник; **source code** - исходный текст [программы]

**spaghetti code** - запутанная, плохо спроектированная, слабо структурированная и трудная для понимания программа

**statement** - утверждение, оператор (синтаксическая конструкция)

**store** – запас; **restore** – восстанавливать; **storage** - хранение

**stream** – поток

**string** - строка а) в языках программирования - тип данных, позволяющий обрабатывать последовательности символов, поэтому иногда её называют символьной, или текстовой ( text string)

**string matching** - сопоставление строк

**structured programming** - структурное программирование методология проектирования программ

**subject** - предмет, тема

**subroutine** - подпрограмма; ~ library - библиотека подпрограмм

**subset** - подмножество

**substitute** - замена; **substitution** - подстановка

**success** – достижение; удача

**summary** - резюме

**support** - обеспечение, поддержка

**survey** - обзор

**syntax** - синтаксис, набор формальных правил записи предложений языка программирования или команд операционной системы. Синтаксис языка описывается его грамматикой ( grammar), а значение предложения определяется семантикой языка программирования ( semantics)

**system** - система; **operational ~** - операционная система, действующая система (находящаяся в рабочем состоянии)

**T**

**tab** - 1. этикетка, ярлык; 2. выступ

**table** - 1. таблица; табель; tabulation - сведение в таблицы; 2. стол

**tag** - 1. признак, метка // снабжать признаком, помечать, маркировать; 2. этикетка; бирка; ярлык

**tape** – лента

**target** - цель, мишень

**task** - 1. задача; задание // ставить задачу; 2. ветвь (программы)

**template** - шаблон; трафарет; маска; эталон (в системах распознавания)

**term** - 1. термин; 2. член (логического выражения); 3. срок; 4. гонорар

**text manipulation** - обработка текстов; преобразование текстов

**throughout** – по всему, во всех отношениях

**throughput** - производительность (машины); пропускная способность (канала)

**tool** - орудие, инструментальное средство

**trace** - след // следить; отслеживать

**track** - 1. дорожка (на диске); канал, тракт

**transcribe** - 1. переписывание (данных с преобразованием или без преобразования); 2. расшифровка

**transfer** - передача; перенос, переход // передавать; переносить

**translation** - 1. перевод (с одного языка на другой); 2. трансляция (программы); 3. преобразование

**treat** – обрабатывать; трактовать; лечить

**true** - <истина> (значение булева выражения) // истинный

**truncate** – усекать, отбрасывать; (досрочно) завершать процесс вычислений

**tutorial** – учебное пособие; наставнический

**type** - 1. тип; род; класс; вид; 2. литера; шрифты; 3. печатать, набирать на клавиатуре

**U**

**underscore** - 1. символ подчеркивания // подчеркивать; 2. давать заниженную оценку

**unit** - 1. единица; единое целое; 2. единица измерения; 3. устройство; узел; блок; прибор; звено; элемент

**up to date** - современный, новейший

**usage** - 1. частота использования, используемость; 2. словоупотребление

**V**

**validation** - проверка достоверности; подтверждение правильности; аттестация

**value** - 1. значение, величина; 2. оценка // оценивать, 3. стоимость; цена; ценность

**variable** - 1. переменная (величина); 2. изменяемый; переменный; регулируемый

**variable name** - имя [идентификатор] переменной

**vehicle** - носитель (данных, информации)

**verify** - контролировать; проверять; сличать

**void** – пустота, очищать

**volume** - 1. объем, вместительность, груда; 2. уровень громкости; 3. книга, том

**W**

**warning** - предупреждение

**workstation** - рабочая станция; автоматизированное рабочее место

**Z**

**zip** – проноситься (напр. курсор мыши на экране); zipped file = zip file архивный zip файл