# Занятие последнее

## Алгоритмы

### Теория

1. Что такое алгоритм? Что такое временная сложность алгоритма?
2. Какие существуют алгоритмы сортировки? По каким характеристикам они отличаются?
3. Что понимается под рекурсией в программировании? В чём её плюсы и минусы? Приведи примеры задач, решаемых с помощью рекурсии. Чем поведение бесконечной рекурсии отличается от поведения бесконечного цикла?

### Практика

1. Напиши реализацию двоичного поиска. Какие ошибки были в процессе реализации, на каких данных их можно обнаружить?
2. Выбери два любых алгоритма сортировки. Приведи два набора данных, таких, что на первом наборе первый алгоритм сортировки будет эффективнее второго, а на втором — наоборот, второй алгоритм эффективнее первого. Посчитай сложность обоих алгоритмов на обоих наборах данных.  
   Примечание: Реализовывать самим эти алгоритмы не нужно, но желательно где-нибудь найти готовые реализации (можно даже не на Perl), чтобы иметь возможность их запустить на ваших данных и подтвердить гипотезы о сложности.
3. Напиши рекурсивную и нерекурсивную реализацию вычисления факториала.
4. Пафнутий написал скрипт для обработки данных лог-файлов. С точки зрения логики и функциональности код работает абсолютно корректно, однако он очень неоптимален с точки зрения производительности и количества выполняемых действий. Никодим должен переписать скрипт, оптимизировав его, но полностью сохранив логику работы.  
     
   open(my $file1, '<', 'data\_file');  
   my @file\_data1 = readline( $file1 );  
     
   my @matches;  
   for my $str1 (@file\_data1) {  
    open(my $file2, '<', 'data\_file');  
    my @file\_data2 = readline( $file2 );  
    process(@file\_data2);  
     
    for my $str2 (@file\_data2) {  
    $str2 =~ /^(\d\d)/;  
    my $category = get\_category\_from\_db($1);  
     
    next unless $str2 =~ /\d{4}-\d{2}-\d{2} \w+ \w+ [A-Z]{2}\s[\w!.,]+/;  
     
    my $template = get\_template\_from\_db($str1);  
     
    if ($str2 =~ /\d{4}-\d{2}-\d{2} $template $category [A-Z]{2}\s[\w!.,]+/) {  
    push @matches, $str2;  
    }  
    }  
   }  
     
   print 'match' if scalar @matches > 0;  
     
   sub process {  
    my @data = @\_;  
     
    print 'file size: ' . scalar(@data);  
   }  
     
   Читаемые файлы содержат n строк. Функции get\_category\_from\_db и get\_template\_from\_db делают по одному запросу в базу данных. Получаемые категории и шаблоны являются цифро-буквенными строками.  
     
   Какова вычислительная сложность скрипта относительно n? Какое количество операций чтения строки из файла, вызова функции, компиляции регулярного выражения, запросов в базу данных выполняет скрипт? Какое количество памяти потребляет скрипт? Сколько раз происходит копирование прочитанных из файла данных? Все цифры оценить до и после оптимизации.

# Занятие 7

## regex

### Теория

1. Для чего служат регулярные выражения?
2. Какие операторы сопоставления шаблону есть в Perl? Что они делают? Что возвращают эти операторы в разных контекстах и условиях?
3. Что такое метасимволы, чем их поведение отличается от обычных символов в re? Какие метасимволы есть в Perl re, для чего служит каждый из них?
4. Что такое квантификаторы? Какие квантификаторы есть в Perl re, как работает каждый из них?
5. Какие есть варианты поведения квантификаторов с точки зрения размера захватываемого совпадения? Как их регулировать?
6. Что такое модификаторы? Какие основные модификаторы есть в Perl re, на что они влияют?
7. Что такое классы символов? Для чего они нужны? Как определить класс символов? Как инвертировать класс символов?

### Вопросы

1. Пафнутий пытался сделать выборку из данных симулятора всех бойцов, чья фамилия начинается с «Иван», и вывести все эти фамилии. Он написал такой код:  
   $name =~ /(\w+) (\w+) (\w+)/;  
   if ($1 =~ /^(Иван).\*/) {  
    print $1;  
   }  
   Что не так с этим кодом?
2. Как сделать так, чтобы помещённый в скобки подшаблон не производил захвата в переменную?
3. Есть ли разница в интерпретации символов $ в следующих re:  
   /$foo$/  
   /$foo$bar$/  
   /$foo\n$bar/m  
   /($foo\n$)bar/m
4. Есть ли смысл в re \b{3}?
5. Какие из этих выражений является допустимыми? Почему?  
   "test" =~ /test/  
   "test" =~ s/te/to/  
   "test" =~ s/te/to/r

### Практика

1. Пафнутий накосячил с загрузкой данных в симулятор, в результате для части бойцов вместо отчества записалась фамилия. Например, «Хрустицкий Владислав Хрустицкий» вместо «Хрустицкий Владислав Владиславович». Никодим должен написать регулярное выражение, которое позволит найти такие битые данные.
2. Эммануил Гедеонович готовит к передаче заказчику техническую документацию по проекту X. Проект X включает несколько моделей с разными номерами (например, X-112). Для повышения маркетинговой привлекательности было решено присвоить каждой модели кодовое имя (например, модель X-100 — Exterminator, модель X-911 — Extractor и т. п.). У Никодима есть информация по кодовым именам всех моделей. Он должен написать регулярное выражение для замены всех номеров моделей в документации кодовыми именами.
3. Решить минимум три любые задачи из <http://callumacrae.github.io/regex-tuesday/>.

### Материалы

1. <http://perldoc.perl.org/perlre.html>

# Занятие 6

1. Допиливаем тесты.

# Занятие 5

## Юнит-тестирование

### Теория

1. Какие бывают виды тестов? Чем они отличаются?
2. Для чего нужны юнит-тесты, какую пользу они приносят?
3. Изучить документацию [Test::More](https://metacpan.org/pod/Test::More).

### Практика

1. Никодим закончил разработку симулятора и должен передать его на поддержку Пафнутию. Прежде чем это сделать, он намерен покрыть всю систему классов техники и вооружения юнит-тестами с помощью Test::More. Набор тестов должен покрывать все свойства и операции с объектами, а также их взаимодействие. Должна быть возможность запустить как весь набор тестов, так и тесты определённого класса.

# Занятие 4

1. Доделываем задачу по проектированию до состояния «чисто вылизано».
2. Начинаем работать над новостным порталом.
3. На досуге — получаем базовые знания по клиентским технологиям (HTML, CSS, JS).

# Занятие 3

## OOP

### Теория

1. Что такое класс, объект, инкапсуляция, наследование, полиморфизм?
2. Как в Perl определить класс, методы, поля? Как создать объект?
3. Чем отличаются методы класса от методов экземпляра?
4. Как определить конструктор объекта в Perl? Как определить деструктор объекта в Perl? В чём принципиальная разница между ними с точки зрения языка?
5. Чем отличается вызовы:  
   SomeClass::somemethod()  
   SomeClass->somemethod()  
   SomeClass->new()->somemethod()
6. Что такое аксессоры?
7. Как унаследовать класс в Perl?
8. Как работает множественное наследование в Perl? Что такое порядок разрешения методов (method resolution order)?
9. Как определить, наследует ли класс от указанного класса?
10. Как определить, есть ли в классе указанный метод?
11. Прочитать Moose::Manual.

### Вопросы

1. Можно ли bless-нуть массив?
2. Как сделать поле объекта приватным?

### Практика

1. Эммануил Гедеонович поручил Пафнутию разработать программный симулятор для тестирования ТТХ новых моделей своего завода. Одна из сущностей, которую нужно описать в объектной модели — член экипажа. Каждый боец имеет имя (ФИО через пробел), звание (\w+), специальность (командир|механик-водитель|наводчик|заряжающий|радист), срок службы (\d+ лет) и модель техники, на которой он прошёл обучение.  
     
   Пафнутий решил реализовать класс с помощью стандартных средств ООП в Perl, а кроме того аксессоры для каждого поля реализовать с помощью AUTOLOAD. Геттер и сеттер для поля должны иметь одно и то же имя, кроме того, должна проводиться валидация данных — нельзя, например, установить имя «Tim Towtdy» или специальность «мотострелок». Помогите Пафнутию выполнить его задачу.
2. Реализация проекта Пафнутием затянулась, поэтому он был передан Никодиму. Никодим получил задачу реализовать модель с учётом следующих требований:  
   — В системе есть танковая, корабельная, авиационная и артиллерийская техника.  
   — У каждой модели техники есть базовые характеристики: название модели, скорость передвижения, толщина брони, прочность. Скорость и прочность могут изменяться, например, при получении урона, но название модели и толщина брони задаются только при создании объекта. Кроме того, у всех полей должна быть базовая валидация.  
   — После создания объекта каждый тип техники должен выполнить определённые действия, например, артиллерия — занять позицию, а самолёт — взлететь.  
   — Каждый тип техники имеет метод «передвигаться», но внутри он выполняет специфичный для техники вид движения. При этом, например, при вызове метода «плыть» у танка он должен уничтожаться, и т. п. — Каждый тип техники имеет метод «получить попадание», который уменьшает значение прочности на переданное значение урона. При снижении прочности до нуля техника уничтожается. Кроме того, с вероятностью 10% после попадания у танка может сдетонировать боекомплект, корабль получить пробоину ниже ватерлинии, а самолёт потерять двигатель (всё это также приводит к уничтожению).  
   — Танки имеют пушки и пулемёты, корабли — пушки и торпеды, самолёты — ракеты и пулемёты, артиллерия — пушки. Орудие каждого типа на технике только одно. Если на технике есть пушка, то модель имеет метод «выстрелить из пушки» и т. д. для всех типов орудий.  
   — Каждый тип вооружения также описывается классом. Все они имеют поля тип, количество боеприпасов и методы «прицелиться» и «выстрелить». При каждом выстреле количество боеприпасов уменьшается, если боеприпасов не осталось, то попытка выстрелить должна приводить к ошибке. Пулемёты имеют размер магазина и перезаряжаются, когда магазин опустошён. Пушки перезаряжаются после каждого выстрела. Ракеты и торпеды не перезаряжаются.  
     
   Никодим использовал для решения своей задачи Moose. Что у него получилось?  
     
   (Результат должен быть опубликован в отдельном репозитории на Гитхабе.)

### Материалы

1. <https://metacpan.org/pod/Moose::Manual>

# Занятие 2

## DB & SQL

### Теория

1. Что такое реляционная база данных?
2. Что такое нормализация и денормализация БД? Для чего они нужны? Какие проблемы они могут вызвать?
3. Что такое транзакции? Для чего они нужны? Что такое ACID?
4. Что такое индексы? Для чего они нужны?
5. В чём разница между операторами INSERT, REPLACE и UPDATE?
6. В чём разница между операторами DELETE, TRUNCATE и DROP?
7. Из каких основных частей состоит оператор SELECT?
8. Какие виды JOIN есть в MySQL? Чем они отличаются?
9. Какие модули используются для работы с БД в Perl?
10. Что такое ORM? Какие ORM есть для Perl? В чём разница между ORM и генератором запросов?

### Вопросы

1. Есть ли разница в скорости работы между:  
   LIKE ‘test’  
   LIKE ‘test%’  
   LIKE ‘%test%’  
   Почему?
2. Чем отличается задание условий с помощью HAVING и WHERE?
3. Чем отличается char(n) от varchar(n)?
4. Никодим немного суеверен, поэтому, когда ему нужно удалить какие-то записи из таблицы, он сначала пишет в консоли условие WHERE для строк, которые нужно удалить, и только потом добавляет к нему DELETE FROM tbl\_name. Почему Никодим считает плохой приметой писать оператор DELETE в обычном порядке?

### Практика

1. <https://www.hackerrank.com/challenges/placements>
2. <https://www.hackerrank.com/challenges/weather-observation-station-18>
3. <https://www.hackerrank.com/challenges/the-report>
4. <https://www.hackerrank.com/challenges/the-company>
5. <https://www.hackerrank.com/challenges/symmetric-pairs>

### Материалы

1. <https://mariadb.com/kb/en/mariadb/documentation/> — документация по MariaDB.
2. <https://dev.mysql.com/doc/> — документация MySQL.
3. <https://www.hackerrank.com/domains/sql/> — задачи по MySQL на HackerRank.
4. <http://sql-ex.ru/> — задачи по SQL.

# Занятие 1

## Perl

### Теория

1. Что такое переменная? Что такое значение? Может ли существовать переменная без значения? Значение без переменной?
2. Чем отличается объявление переменной от её определения?
3. Какие основные типы переменных есть в Perl? Чем они отличаются?
4. Что такое сигилы? Для чего они нужны?
5. Чем отличаются массивы от хешей?
6. В чём разница между массивом и списком?
7. Что такое область видимости?
8. Для чего используются ключевые слова my, our, local, state? В чём их различие?
9. Какие виды операторов есть в Perl?
10. Что такое контекст? Какие виды контекста есть в Perl? На что они влияют?
11. Какие операторы работы со списками есть в Perl?
12. Что такое функция (процедура, подпрограмма)? Какими основными свойствами она обладает?
13. Чем отличаются понятия пакет и модуль в Perl?
14. Какие есть возможности вызвать функцию, объявленную в другом модуле?
15. Как прочитать файл в Perl?
16. Что такое perldoc?
17. Что такое CPAN? Какие клиенты для него существуют?
18. Что такое perlbrew? Для чего его можно использовать?

### Вопросы

1. Почему сигил для элемента массива отличается от сигила для самого массива?
2. Почему сигилы для скаляра, массива и хеша выглядят так, как они выглядят?
3. В чём разница двух вариантов записи:  
   my %hash = (“key1”, “value1”, “key2”, “value2”)  
   my %hash = (key1 => “value1”, key2 => “value2”)
4. Есть следующая структура данных:  
   my $s = { a => [[ 3, 2, 1 ], 0], b => 4 }  
   Как получить из неё массив, содержащий значения (2, 1)?
5. Есть ли разница между выражениями $x++ и ++$x?
6. В чём различие между числовыми и строковыми операторами сравнения? Приведи пример, когда они будут давать разные результаты с одними и теми же операндами.
7. Какие значения в Perl считаются истинными, а какие ложными? Приведи выдержку из perldoc.
8. Есть два выражения:  
   print($x = 1 and 0)  
   print($x = 1 && 0)  
   Чему будет равен $x после выполнения каждого из них? Что выведет print? Почему?
9. Что выведет выражение  
   $x = 1; $y = $y && $x++; print $x;  
   в случае, если до его выполнения $y == 1? $y == 0? Почему?
10. Чем отличается use от require?

### Практика

1. Директор работающего на ВПК завода Эммануил Гедеонович обратился к программисту Никодиму, чтобы тот разработал внутренний каталог продукции. Каждая модель, выпускаемая заводом, имеет очень секретный номер (порядковые целые числа начиная с нуля), не менее секретное кодовое имя (буквы английского алфавита от A до Z), а также несколько технических характеристик: цвет, вес и бронепробитие. Какое представление данных мог бы использовать Никодим для реализации каталога с поиском моделей как по номеру, так и по кодовому имени?
2. Завод Эммануила Гедеоновича имеет потребность выполнять определённые расчёты с ТТХ своих моделей. Сами функции расчётов уже реализованы, но они страшно секретные, поэтому Никодиму их показать не могут. Задача Никодима — написать универсальную функцию, которая может принимать на входе некую произвольную функцию, каталог и ключ, обозначающий нужную характеристику в каталоге, а на выходе выдавать массив, каждый элемент которого будет результатом применения переданной функции к указанной характеристике каждой модели в каталоге.

### Материалы

1. [perl.org](https://www.perl.org/) — официальный сайт Perl.
2. [PerlMonks](http://www.perlmonks.org/) — StackOverflow мира Perl.
3. Рассылка moscow.pm
4. [Modern Perl по-русски](https://web.telegram.org/" \l "/im?p=@modernperl) в Telegram.
5. [MetaCPAN](https://metacpan.org/)
6. [Learning Perl, 7th Edition](http://shop.oreilly.com/product/0636920049517.do) by Randal L. Schwartz, brian d foy, Tom Phoenix (*Llama Book*)
7. [Intermediate Perl, 2nd Edition](http://shop.oreilly.com/product/0636920012689.do) by Randal L. Schwartz, brian d foy, Tom Phoenix (*Alpaca Book*)
8. [Programming Perl, 4th Edition](http://shop.oreilly.com/product/9780596004927.do) by Tom Christiansen, brian d foy, Larry Wall, Jon Orwant (*Camel Book*)
9. [Modern Perl](http://modernperlbooks.com/books/modern_perl_2014/) by chromatic ([русский перевод](http://modernperlbook.ru/))

## GIT

### Теория

1. Что такое система контроля версий? Для чего она нужна, какие возможности даёт?
2. В чём различие централизованных и распределённых систем контроля версий? К какому типу относится Git?
3. Что такое рабочая директория (working directory), стейджинг (staging area), репозиторий (repository)? При каких действиях файлы могут перемещаться из одного в другое?
4. Что означают состояния untracked, unmodified, modified, staged? Какие операции переводят файл из одного состояния в другое?
5. Что такое ветка (branch)? Для чего они нужны, как их можно применять?
6. Что делают команды git init, clone, add, rm, commit, log, diff, pull, push, revert, branch, checkout, merge? Какие опции могут влиять на их работу (несколько основных примеров)?

### Вопросы

1. Пафнутий сделал свой первый коммит, но потом обнаружил, что в данные коммиттера попал ник и емейл с сайта знакомств. Пафнутий срочно поправил настройки, сделал revert коммита и закоммитил правки заново, после чего отправил их в центральный репозиторий. На следующий день на доске объявлений он обнаружил свою фотографию с подписью crazymacho28cm. Что пошло так? Что нужно было сделать Пафнутию?
2. Пафнутий решил закоммитить в репозиторий очень большой файл. У него были сомнения, не получит ли он за это по шапке, но он решил, что в случае чего потом просто удалит файл из репозитория. Верна ли логика Пафнутия?
3. Пафнутию поручили исправить баг. Он сделал коммит с именем fix, содержащий исправление. Оно оказалось недостаточным, поэтому Пафнутий сделал коммит fix2. Ещё через пару подобных итераций он наконец смог полностью исправить баг. Спустя несколько дней Пафнутий пришёл домой с работы с фингалом. Почему?

### Практика

1. Создай локальный репозиторий (дальше будем называть его *первым*).
2. Настрой имя пользователя (Фамилия Имя по-русски) и емейл (корпоративный емейл в домене reg.ru) для коммитера. Убедись, что они установлены верно, выведя соответствующие конфигурационные параметры.
3. Опубликуй репозиторий на Github (дальше будем называть его *центральным*).
4. Создай в каталоге с репозиторием файл task1.1, скопируй в него текст этого задания. Посмотри статус репозитория. Сделай так, чтобы это файл никогда не попал в репозиторий и не обрабатывался никакими командами git. Просмотри ещё раз статус репозитория.
5. Создай в репозитории файл task1.1\_answer. Этот файл будет ответом на упражнение и должен содержать все выполняемые команды и их вывод по каждому пункту задания. Впиши в этот файл уже выполненные команды и их результат. Зафиксируй выполненные правки и отправь их в центральный репозиторий.
6. Склонируй репозиторий в другую директорию/на другую машину (дальше будем называть его *вторым*).
7. Во втором репозитории создай файлы fileA и fileB с произвольной строкой текста в каждом. Добавь соответствующие команды в task1.1\_answer. Просмотри статус репозитория. Сделай так, чтобы правки файлов fileA и fileB (но не task1.1\_answer) попали в первый репозиторий. Просмотри статус репозитория.
8. В первом репозитории добавь в fileA ещё одну строчку текста. Просмотри текущие изменения в этом файле. fileB удали. Добавь соответствующие команды в task1.1\_answer. Отправь правки в центральный репозиторий.
9. Во втором репозитории зафиксируй оставшиеся правки файла task1.1\_answer. Просмотри лог в первом и втором репозитории. Сделай так, чтобы они попали в первый репозиторий вместе с правками этого файла, выполненными там.
10. В первом репозитории восстанови файл fileB, отправь его в центральный репозиторий.
11. Приведи оба репозитория к одинаковому финальному состоянию. Посмотри лог в первом и втором репозитории.
12. Создай в первом репозитории ветку feature1. Добавь произвольную строку в файл fileB, обнови task1.1\_answer и отправь в центральный репозиторий.
13. Во втором репозитории создай ветку feature2 из ветки master.
14. Получи ветку feature1 во втором репозитории, влей изменения из неё в ветку master, обнови task1.1\_answer, отправь изменения в центральный репозиторий.
15. Во втором репозитории переключись на ветку feature2. Просмотри список веток локально и в центральном репозитории. Обнови task1.1\_answer, отправь в центральный репозиторий.
16. В первом репозитории получи все изменения из центрального репозитория. Верни ветку master к состоянию до слияния с веткой feature1. Влей в master ветку feature2. task1.1\_answer после слияния должен содержать все изменения, выполненные в любых ветках.
17. Просмотри графический лог. Удали ветку feature1 (в том числе из центрального репозитория).
18. Обнови task1.1\_answer. Пометь текущее состояние мастера тегом ‘v1.0’. Отправь изменения в центральный репозиторий.
19. Синхронизируй состояние всех репозиториев.
20. Дай права на просмотр репозитория на GitHub’е преподавателю и своему ментору и отправь им ссылку на него.

### Материалы

1. <https://git-scm.com/> — официальный сайт Git.
2. <https://git-scm.com/doc> — официальная документация и ссылки на обучающие ресурсы.
3. <https://git-scm.com/book/en/v2> — книга «Pro Git».  
   <https://git-scm.com/book/ru/v2/> — русский перевод.
4. <https://githowto.com/> — пошаговый туториал (есть русская версия).
5. <https://www.codecademy.com/learn/learn-git> — базовый курс от CodeAcademy.
6. <http://learngitbranching.js.org/> — чуть менее базовый туториал с визуализацией работы с ветками.

# Занятие 0

1. Знакомимся. :)
2. Как выглядит отдел разработки.
3. Как будут проходить занятия?
4. Как работать с менторами?
5. Документы с ответами и решениями.