Практика программироавния. Заметки.

Семён Григорьев

2 октября 2020 г.

Содержание

1 Лекция 1: Введение

Программирование — не только написание кода. Документация, сборка, тестирование, версионирование, обработка отзывов пользователей.

Инфраструктура проекта, рабочее окружение, система контроля версий, непрерывная сборка.

Соответствующие решения на примере инфраструктуры вокруг GitHub. GithubActions, внешние сервисы для CI (`https://travis-ci.org/`, `https://www.appveyor.com/`, `https://circleci.com/`).

Практика развёртывания соответствующей инфраструктуры.

1. Для начала, завести аккаунт на GitHub (`https://github.com/`). Важно, чтобы имя аккаунта (логин) было `NameSurname` или `Name_Surname`.

2. Создаём репозиторий для проекта (для всех домашних работ). Название должно отражать сожержимое репозитория. Не забываем добавить описание. Лицензию, readme и gitignore лучше не добавлять.

3. Устанавливаем git (`https://git-scm.com/`) и графическую оболочку для работы с ним (если кому нужно).

4. Теперь пора приступать к созданию проекта. Так как дальше мы будем пользоваться F#, то в качестве шаблона предлагается использовать `https://github.com/TheAngryByrd/MiniScaffold`.

   (a) Установить .NET Core: `https://dotnet.microsoft.com/download`

   (b) Прочитать инструкции (https://github.com/TheAngryByrd/MiniScaffold#install-the-dotnet-template-from-nuget) и выполнить соответсвующие шаги. Нам нужно создать консольное приложение. Это может занять некоторое время.

5. Устанавливаем связь только что созданного локального репозитория с удалённым репозиторием: https://docs.github.com/en/github/importing-your-projects-to-github/adding-an-existing-project-to-github-using-the-command-line

С этого момента домашние работы только через GitHub с налаженной сборкой.

2 Домашняя работа 1

Задачи:

1. **(1 балл)** Инициализировать рабочее окружение: репозиторий на GitHub, CI, readme, лицензия. Добавить преподавателя в совладельцы. Оформить тестовый pull request: например, оформленное readme (поправить описание проекта, удалить лишнее, что досталось от шаблона, поправить ссылку на статус сборки). Запросить ревью у преподавателя.

3 Лекция 2

3.1 Первое знакомство с F#

Общие сведения о платформе .NET, общие представления об архитектуре платформы.

F# — это один из языков платформы. Некоторые ресурсы дл того, чтобы начать изучать F#.

- Главный сайт сообщества: `https://fsharp.org/`. Там много чего интересного, от библиотек и инструментов до ссылок на другие ресурсы.
- Неплохая книга по основам языка: `https://en.wikibooks.org/wiki/F_Sharp_Programming`
- Подборка обучающих материалов от сообщества: `https://fsharp.org/learn/`
- Официальная документация от Microsoft: `https://docs.microsoft.com/ru-ru/dotnet/fsharp/`

Структура проекта вообще и на языке F# в частности. Разбиение кода на модули, файлы, библиотеки. Переиспользование кода. Зависимости между модулями, библиотеками. Пространства имён.

Основные особенности F#, примеры кода, базовые языковые конструкции и типы.

Примитивные типы: логисекие, числовые, символы, строки. Массивы, основные функции работы с ними: инициализация, взятие элемента, запись элемента.

Базовые констукции управления: ветвления (if), циклы (различные варианты for, while). Двумерный синтаксис.

Структура программы. Точка входа, модули, функции. Модуль и пространство имён.

Основы работы с консолью, библиотека Argu.

3.2 Тестирование программ

Тестирование программ: ручное, автоматизированное, автоматическое.

Доказательство корректности vs тестирование. Тестирование не есть доказательство корректности. А можно ли всё таки доказать корректность? Да (иногда).

- Формальная верификация используя внешние инструменты.
- Использование специальных языков программирования, таких как Coq (на самом деле это целая система, так называемый proof assistant), Agda, Idris, F*,

Типы тестов и особенности их применения: модульные, интеграционные, unit и т.д. Автоматизация создания тестов. Intellitest — пример инстумента для автоматической генерации тестов. Примеры инструментов для тестирования: Expecto, FsUnit, NUnit, FsCheck, Canopy.

С этого момента все домашние работы должны быть снабжены автоматически запускаемыми при сборке тестами.

3.3 Отладка кода

Отладка кода. Некоторые методы отладки: отладочная печать, логгирование, использование пошаговых отладчиков. Некоторые шаги отладки: формулировка гипотезы и её проверка, локализация ошибки, работа с тестами. Практика по использованию отладчика.

4 Домашняя работа 2

В задачах, связанных с обработкой массивов на вход необходимо принимать длину массива и затем создавать случайный массив соответствующей длины. Для всех задач необходимо реализовать чтение входных данных из консоли и вывод результата в консоль.

Задачи:

1. **(1 балл)** Реализовать функцию, вычисляющую значение выражения $x^4+x^3+x^2+x+1$ "наивным" способом.
2. **(1 балл)** Реализовать функцию, вычисляющую значение выражения $x^4+x^3+x^2+x+1$, применив минимальное число умножений и сложений.
3. **(1 балл)** Вычислить индексы элементов массива, не больших, чем заданное число.
4. **(1 балл)** Вычислить индексы элементов массива, лежащих вне диапазона, заданного двумя числами.
5. **(1 балл)** Дан массив длины 2. Поменять местами нулевой и первый элементы, не используя дополнительной памяти/переменных.
6. **(1 балл)** Поменять местами i-й и j-й элементы массива не используя дополнительной памяти/переменных.

5 Лекция 3

Ещё раз произменения: в реквесте должно быть только то, что непосредственно относится к сдаваемой домашке.

Ещё раз про функции, про то, как выделять и разделять функциональность, не надо запихивать всё в одну функцию. Про то, где должны быть проверки.

Про консоль.

Про обработку крайних случаев. Про исключения.

Про тесты и ошибки: нашёл ошибку — создал тест.

Про стиль кодирования: про пробелы вокруг скобок и операций, про отступы и переводы строк. Про соглашения о наименовании. camlCase CamlCase

Про единицы измерения.

Базовые структуры данных, алгоритмы и их выражение в F#. Функция. Рекурсия и итерация. Базовые типы и основы работы с ними: матрицы, массивы, списки, структуры.

Числа Фибоначчи.

6 Домашняя работа 3

Для всех задач обеспечить чтение n из консоли и печать результата в консоль.

1. **(1 балл)** Реализовать вычисление n-ого числа Фибоначчи рекурсивным методом.
2. **(1 балл)** Реализовать вычисление n-ого числа Фибоначчи итеративным методом.
3. **(1 балл)** Реализовать вычисление n-ого числа Фибоначчи используя хвостовую рекурсию (не используя `mutable` и других изменяемых структур). Подсказка: нужно использовать рекурсию с аккумулятором.
4. **(2 балла)** Реализовать вычисление n-ого числа Фибоначчи через перемножение матриц "наивным" методом. Функции построения единичной матрицы, умножения и возведения в степень должны быть реализованы в общем виде.
5. **(2 балла)** Реализовать вычисление n-ого числа Фибоначчи через перемножение матриц за логарифм.
6. **(1 балл)** Реализовать вычисление всех чисел Фибоначчи до n-ого включительно.

7 Лекция 4

Рассказать про то, что выделенные значения — это плохо. Немного про исключительные ситуации. Про создание проектов. Про версии пакетов и вообще версии артефактов. Про то, что заимствование кода не поощрается. Тем более неправомерное заимствование. Про классный пример форматирования `x*x*x + x*x + x + 1`.

Работа с файлами.

Сортировки: пузырьком, вставкой, Хоара. Различные сценарии использования: поддержание отсортированного набора, сортировка всего набора целиком. Некоторые особенности реализации: наивная функциональная реализация Хоара, реализация на массиве.

Основы машинного представления данных. Представления чисел. Представление чисел с плавающей точкой. Проблемы переполнения. Битовые операции. Строки, кодировки.

8 Домашняя работа 4

Во всех задачах на сортировку необходимо реализовать чтение массива из файла и печать результата в файл. Функции чтения и записи необходимо переиспользовать.

В задачах на битовые операции продолжаем рабоать с консолью: чтение данных с консоли и печать результата туда же.

Для данной домашней работы необходимо создать отдельный проект.

При создании тестов необходимо, в задачах на сортировку, убедиться, что, во-первых, сортировки ведут себя одинаоково на одинаковых данных, во-вторых, что они ведут себя так же, как системные сортировки для соответствующих коллекций. Для задачи о запаковке и распаковке надо проверить, что реализованные функции являются взаимно обратными. FsCheck в помощь.

1. **(1 балл)** Реализовать сортировку пузырьком массива.
2. **(1 балл)** Реализовать сортировку пузырьком списка.
3. **(1 балл)** Реализовать быструю сортировку для списка.
4. **(1 балл)** Реализовать быструю сортировку для массива.
5. **(1 балл)** Реализовать запаковку двух 32-битных чисел в одно 64-битное и распаковку обратно.
6. **(1 балл)** Реализовать запаковку четырёх 16-битных чисел в одно 64-битное и распаковку обратно.