Введение, многопоточное программирование

Юрий Литвинов
yurii.litvinov@gmail.com

06.09.2019

1. Введение

Начнём с того, о чём вообще этот курс. В этом семестре считается, что вы уже освоили основы объектно-ориентированного программирования, поэтому в этом курсе будет рассказано, как же его применять. Вообще, цель этого курса — познакомить вас с как можно большим количеством областей, с которыми должен быть знаком любой практикующий программист, чтобы после этого курса вы хотя бы слышали и имели небольшой опыт со всем, что обычно встречается программистам на практике. Это:

- многопоточное программирование — сейчас практически не бывает одноядерных процессоров, так что писать программы, имеющие всего один поток выычислений, становится всё более и более немодно; навыки многопоточного программирования считаются сейчас обязательными практически везде;

- сетевое программирование — существуют кучи облачных сервисов и приложений, которые их используют, пользователи ожидают, что любая программа умеет загружать что-нибудь в какое-нибудь облачное хранилище, общаться с третьесторонними сервисами и т.д., без навыков сетевого программирования писать такие приложения очень сложно;

- веб-программирование — можно сказать, что это подвид сетевого программирования, но вообще это отдельный мир, со своим стеком технологий, своими архитектурными подходами, своими даже языками программирования; к несчастью, большая часть создающихся нынче приложений — именно веб-приложения, потому что они работают в браузере, не требуют установки и доступны на любой платформе (хоть на мобильнике) из любой точки мира;

- продвинутое GUI-программирование — когда всё-таки нужен клиент, устанавливаемый на компьютер (например, если браузерное приложение адски тормозит или ему нужен доступ к локальным ресурсам); какие-то приложения с GUI мы до этого писали, но тут речь пойдёт про более современные библиотеки и про важные концепции, которые в прошлый раз не затрагивались;

- работа с базами данных — без них в веб-программировании никуда, да и вообще, чтобы что-нибудь хранить, они необходимы;

- рефлексия — вообще, она не очень нужна, но на ней построены многие из штук, про которые будет рассказываться, да и вообще, считается приличным уметь ей пользоваться;

- немного подробностей про внутреннее устройство платформы, на которой мы программируем (а в этом семестре это снова .NET) — профессиональные программисты должны хорошо представлять себе как устроены инструменты, которыми ои пользуются.

Практически про каждый пункт из этого списка дальше (с третьего курса) будут спецкурсы, а времени у нас очень мало, поэтому может показаться, что про всё очень быстро и скомканно, но такова уж особенность этого курса. Относитесь к этому именно как к введению и к знакомству с каждой темой, потом, если понравится, можно будет более глубоко их изучить на старших курсах. Программа матобеса, к несчастью, устроена так, что почти все спецкурсы по выбору, так что можно счастливо избежать всех этих знаний, но этот курс обязателен, так что необходимый минимум знаний всё-таки придётся получить.

С формальной точки зрения у нас, как в прошлом семестре, одна пара в неделю, в основном я буду что-то рассказывать, но будут и пары, где надо будет писать код прямо на паре (например, про рефлексию, потому что там рассказывать-то особо нечего, или веб-приложения, где рассказывать много чего, но пока руками не попробуете, толку мало). Будет две контрольные (одна где-то в середине семестра, одна в конце), доклады (как обычно, успешный доклад закрывает одну домашку), конечно же, много домашки (задач будет меньше, чем в прошлом семестре, но они будут несколько объёмнее), и главное — семестровая работа.

Семестровая работа — это большая, по возможности научная или практически полезная задача, которую предполагается делать весь семестр (по крайней мере, с получения темы, где-то с октября по декабрь) и которую потом надо будет защитить перед всей группой в конце семестра. Семестровая может быть групповой, но на защите каждый должен чётко сказать, чем конкретно он занимался и что у него получилось. Должна быть чёткая постановка задачи и чётко сформулированный список результатов. От темы требуется только чтобы там было что покодить (чтобы на защите не сложилось впечатление, что это можно сделать за выходные), язык реализации не имеет значения. Фактически, семестровая — это мини-курсовая, от которой не требуется особой научности и не надо писать текстовый отчёт.

Темы для семестровой работы каждый ищет сам и выбирает себе по вкусу. Стратегически мудро было бы сразу брать тему так, чтобы её можно было продолжить как весеннюю курсовую, как курсовую третьего курса и диплом, но также учтите, что второй курс — это время, когда можно попробовать позаниматься разными вещами и понять для себя, что нравится больше. Тему обычно даёт научный руководитель — человек, который потом следит за ходом выполнения работы, даёт советы и помогает с технической стороной дела. Научника и тему можно найти самим (на втором курсе научником может быть кто угодно), хорошим выбором было бы продолжить начатое в летней школе, записаться в студпроект или поспрашивать у знакомых преподов, нет ли у них тем для семестровых. Можно попросить меня познакомить с кем-нибудь из ближайших окрестностей — из лаборатории робототехники (они там не только робототехникой занимаются, но и низкоуровневым программированием вообще), лаборатории машинного обучения в разработке ПО, груп-

пы формальных методов анализа программ. Всегда можно придумать тему самому, так что если вы для себя пишете, например, игру на Unity, почему бы не сдать её как семестровую.

2. Многопоточность