ОГЛАВЛЕНИЕ

Общая цель курса:

Целью этого курса было приобретение профессиональных навыков в областях работы с программным кодом. А именно автоматизация тестирования программы, отладка программы и профилирование программы и ее отдельных модулей.

Рассмотрим подробнее задания, представленные в рамках практикума.

Задание №0. Настройка окружения

Для начала обучения, необходимо было установить вспомогательные пакеты:

1. Интерпретатор Питона с обвязкой и менеджером пакетов:
python3 pip3
2. Пакеты, используемые при выполнении лабораторных работ по первой части курса «Программирование на Си»:
git gcc clang clang-tools gdb
3. Среды разработки qtcreator и VS Code
4. Пакеты для выполнения лабораторных работ по второй части курса «Программирование на Си»:
make valgrind
5. Пакеты для выполнения заданий в рамках практикума:
diffutils gnuplot shellcheck
6. Пакет для работы программы Ramus, необходимой для рисования IDEF0 диаграмм (в рамках курса «Основы программной инженерии»):
java
7. Пакет для автоматического документирования кода в курсе «Основы программной инженерии»:
doxygen
8. И некоторые дополнительные:
nano, htop, ffmpeg

Задание №1. Автоматизация тестирования

Цель: Реализовать тестирующую систему для автоматического тестирования задач, написанных в рамках лабораторных работ по Си.

Реализация тестирующей системы включала в себя написание следующих скриптов:

1. Скрипты отладочной и релизной сборок.
2. Скрипты отладочной сборки с санитайзерами.
3. Скрипт очистки побочных файлов (объектных, исполняемых и т.д).
4. Компаратор для сравнения содержимого двух текстовых файлов.
5. Скрипт для проверки позитивного тестового случая.
6. Скрипт для проверки негативного тестового случая.
7. Скрипт запускающий функциональное тестирование.

Итог: Написание решения одной задачи на Си начинается с подготовки тестов (позитивных и негативных), на которых потом автоматически тестируется написанная программа, что обеспечивает быструю проверку правильности работы программы.

Задание №2. Этапы получения исполняемого файла

Цель: Изучить процесс получения исполняемого файла и процесс организации объектных и исполняемых файлов.

Вывод: Процесс получения исполняемых файлов состоит из 4 этапов — сначала программа обрабатывается препроцессором (подключаются все необходимые программы), затем полученная программа переводится на язык ассемблера, затем на основе полученного ассемблерного файла формируется объектный файл (компиляция) и в конце концов на основе объектного файла мы получаем исполняемый (компоновка).

Задание №3. Отладка с помощью gdb

Цель: Научиться самостоятельно проводить трассировку (пошаговую обработку) приложения, изучить представление различных типов данных языка Си в памяти (переменные, одномерные массивы, матрицы, строки и структуры)

Вывод: Поскольку ни один программист не застрахован от допущения ошибок, необходимо уметь их искать. Искать ошибки вручную довольно сложно и даже не всегда возможно. Поэтому человечеством была разработана утилита gdb. С ее помощью легко осуществляется трассировка приложения, поскольку gdb позволяет ставить точки останова (точки, в которых программа приостанавливает свою работу), чтобы проверить правильность выполнения программы в конкретный момент.

Кроме того в gdb есть команда, которая позволяет смотреть дамп память в конкретный момент выполнения программы (команда *x* — икс). С ее помощью я изучал, как представлены в памяти обычные переменные, массивы, матрицы, строки и структуры.

Задание №4. Постановка замерного эксперимента

Цель: Научиться замерять время работы подпрограммы с помощью функций языка Си (gettimeofday, clock_gettime, clock, __rdtsc), сравнить способы реализации сортировки выбором (адресная арифметика, указатели или индексация) и выяснить плюсы и минусы каждой, сравнить способы умножения матриц (с предварительным транспонированием и без него)

Вывод: В ходе проведения исследований удалось выяснить, что наиболее точной из функций замера времени оказалась функция clock(). С ее помощью и проводились последующие замерные эксперименты.

При проведении замерного эксперимента №1 (сортировка массив), была получена следующая таблица:

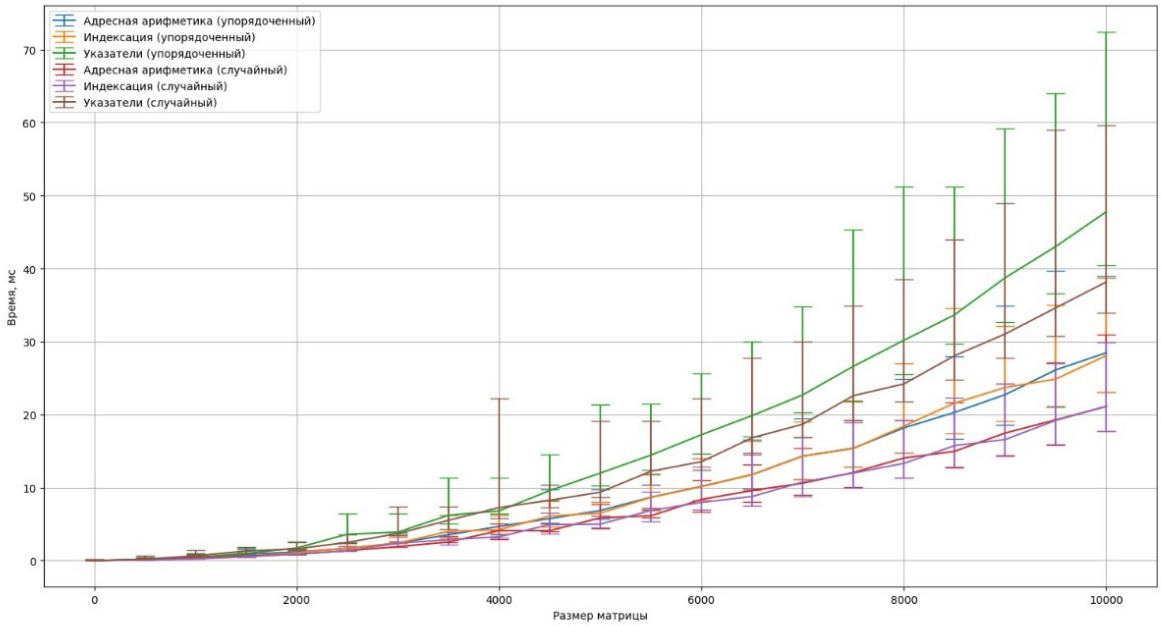

График с ошибкой для программ с уровнем оптимизации O2:

По ней видно, что методы реализации сортировки с помощью индексации и адресной арифметики заметно превосходят в скорости реализацию с помощью указателей.

При проведении замерного эксперимента №2 (умножение матриц) была получена следующая таблица:

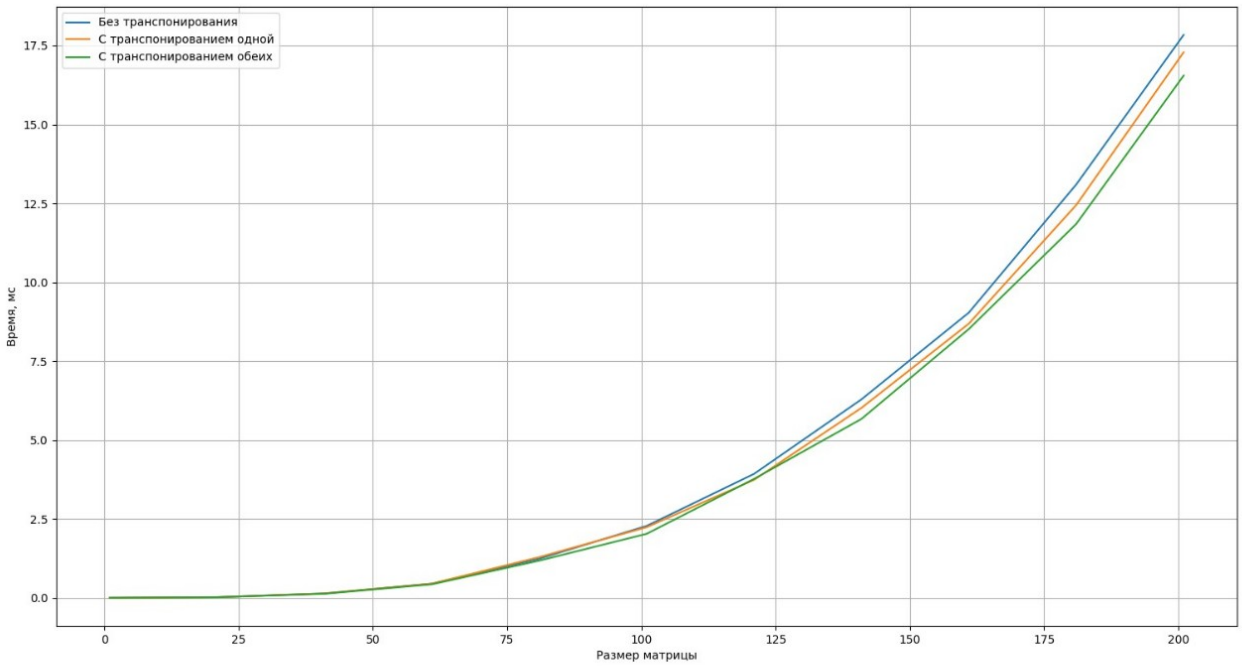

График с уровнем оптимизации O0

Удивительный результат. Реализация с предварительным транспонированием выигрывает у реализации без предварительного транспонирования, несмотря на то, что у первой больше операций. Весь фокус объясняется тем, как компилятор Си представляет память.

Общий вывод:

За время практики удалось приобрести навыки в организации тестирования программы (Задние 1), отладки программы (Задание 3) и профилировании программы (Задание 4)