Национальный исследовательский ядерный университет «МИФИ»
(Московский Инженерно–Физический Институт)
Кафедра №42 «Криптология и кибербезопасность»

Отчёт
по результатам выполнения
Лабораторной работы №1
«Цена вызова»

Дисциплина:	Практические Аспекты Разработки Высокопроизводительного Программного Обеспечения (ПАРВПО)
Студент:	Гареев Рустам Рашитович
Группа:	Б22-505
Преподаватель:	Куприяшин Михаил Андреевич
Дата:	27.03.2025

Москва - 2025

Оглавление

Технологический стек

memory	8GiB Системная память
processor	11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz
siblings	8
cpu cores	4
bridge	11th Gen Core Processor Host Bridge/DRAM Registers
display	TigerLake-LP GT2 [Iris Xe Graphics]
gcc	version 13.3.0
OC	Ubuntu 24.04.2 LTS
IDE	Visual Studio Code 1.98.2

Ход работы

В ходе лабораторной работы были реализованы четыре версии программы:

1. **simple** – все функции размещены в одном файле;
2. **inline** – функции объявлены с ключевым словом inline;
3. **split** – функции вынесены в отдельный .cpp файл, компилируемый отдельно;
4. **onepass** – программа компилировалась сразу из нескольких файлов одним вызовом компилятора

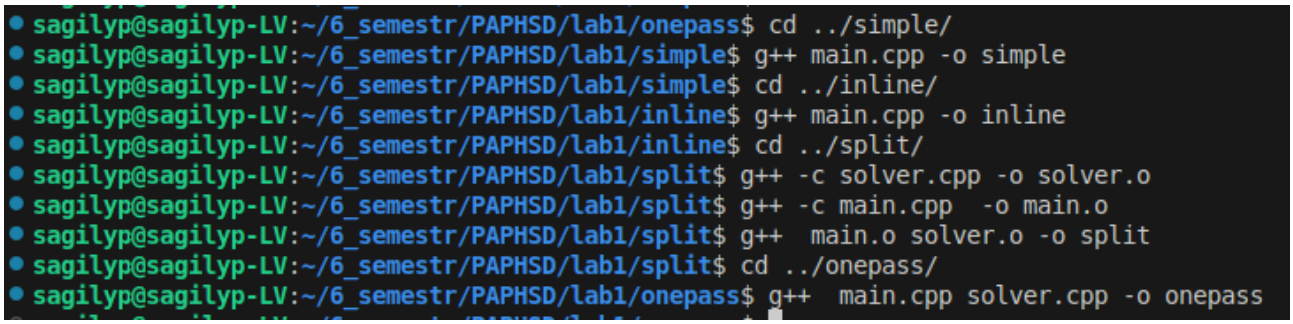

Рисунок 1 — Компиляция программ

Программа запускалась на одинаковом наборе входных данных(100 млн разных наборов коэффициентов, но этот набор не меняется от запуска к запуску). Измерено время выполнения каждой версии. Ниже приведён скриншот запуска соответствующих программ.

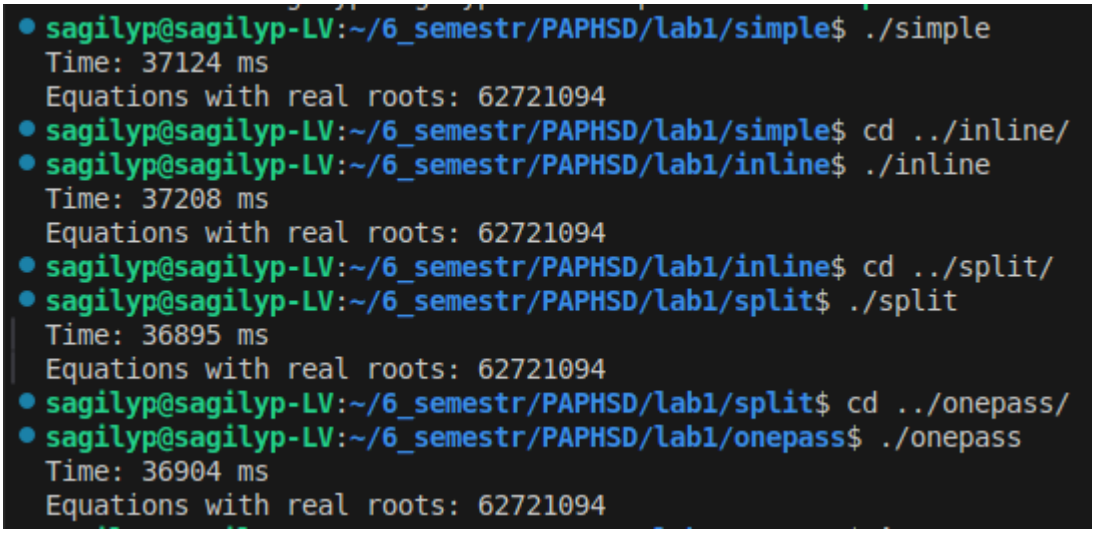

Рисунок 2 — Запуск программ

В результате были получены экспериментальные данные, приведенные в Таблице 1.

Таблица 1 — Результаты эксперимента

Режим работы	simple	inline	split	onepass
Время выполнения, мс	37124	37208	36895	36904

Для визуализации данных была построена столбчатая сравнительная диаграмма, представленная на рисунке ниже.

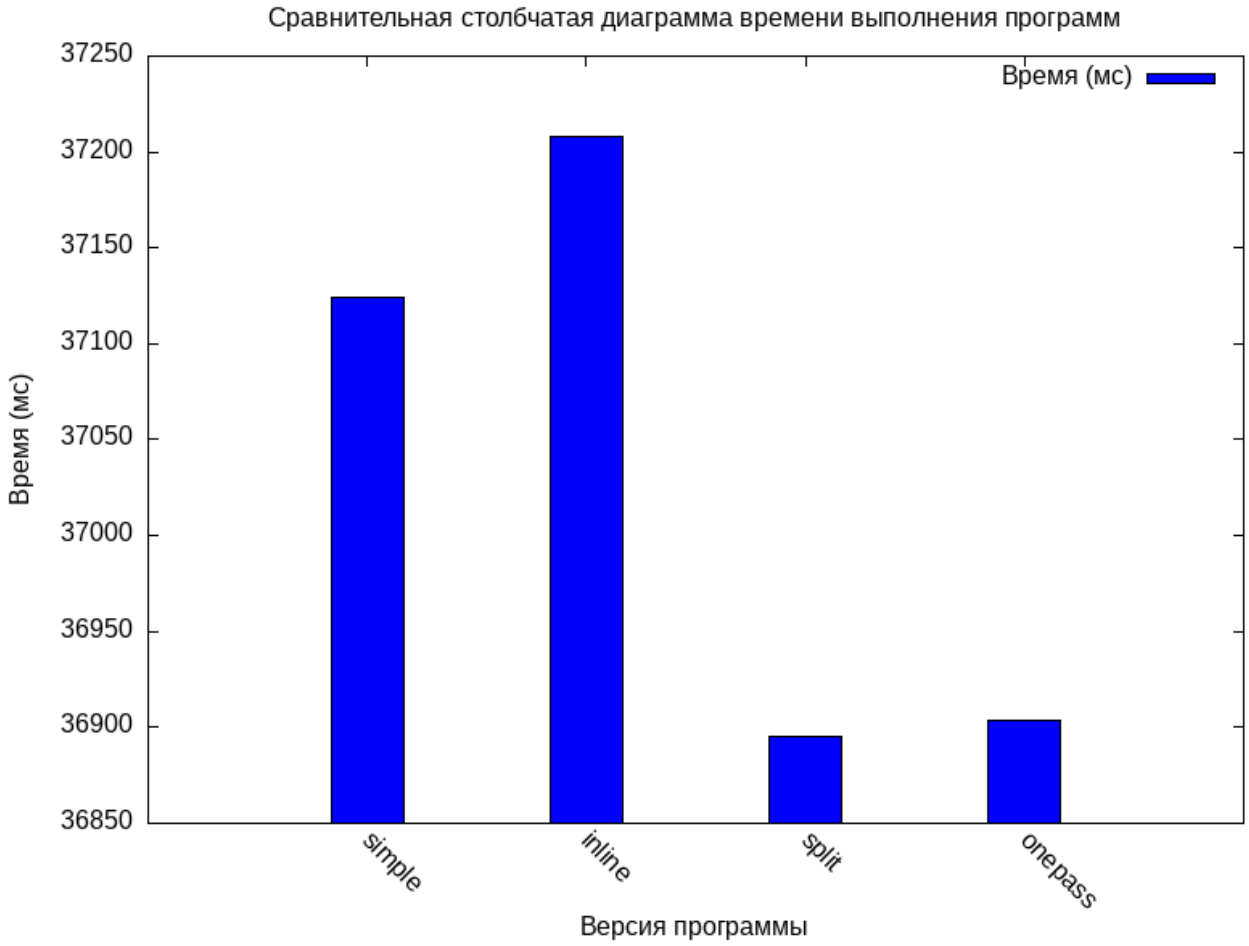

Рисунок 3 — Визуализация результатов эксперимента

Заключение

Результаты эксперимента показывают, что использование **inline** не дало заметного улучшения производительности. Это объясняется тем, что современные компиляторы автоматически принимают решения о встраивании функций, когда это действительно оправдано. Раздельная компиляция (версия **split**) показала наименьшее время работы, что может быть связано с лучшей организацией кэширования кода процессором. Однако разница между раздельной компиляцией и сборкой одним вызовом (**onepass**) минимальна, что подтверждает эффективность компилятора в оптимизации кода.

В целом, различия во времени выполнения всех четырёх версий программы не превышают 0.8%, что говорит о том, что накладные расходы на вызовы функций в данном эксперименте не играют решающей роли. Можно сделать вывод о том, что для вычислительно сложных задач организация кода имеет меньший эффект на производительность по сравнению с самим алгоритмом.

Таким образом, выбор способа компиляции следует делать, исходя не из скорости работы программы, а из удобства сопровождения кода и организации проекта.

Ссылка на гит-репоизторий : **https://github.com/sagilyp/PAPHSD-2/tree/main/lab1**