Homework

COMPUTING SYSTEMS ARCHITECTURE

Жулин Артем Германович | БПИ204 |

25,10,2021

Вариант - 131 - (5, 10)

Условие задачи

Обобщенный артефакт, используемый в задании	Базовые альтернативы (уникальные параметры, задающие отличительные признаки альтернатив)	Общие для всех альтернатив функции
Квадратные матрицы с действительными числами	1. Обычный двумерный массив 2. Диагональная (на основе одномерного массива) 3. Нижняя треугольная матрица (одномерный массив с формулой пересчета)	Вычисление среднего арифметического (действительное число)

Входные и выходные данные

1. В консоль поступает команда следующего типа:
Ввод из файла: "./task -f "input_file" "output_file_1" "output_file_2""
Случайный ввод: "./task -n <number> "output_file_1" "output_file_2""

2. Образец входных данных:
<1
2
1 2
3 4>
Где 1 (1 строка) - это тип матрицы, 2 (2 строка) - это размер матрицы, далее идут элементы самой матрицы. Предполагается, что входные данные корректные и являются типом Int

3. Матрица является некорректной в случае некорректного типа матрицы (от 1 до 3). В случае, если матрица является некорректной программа прекращает чтение.

4. После обработки данных, программа выводит их в выходной файл, после чего сортирует и выводит данные во второй выходной файл. Пример данных:
<Container contains 1 elements:
1: It's diagonal matrix. Dimension = 3, Sum = 6, Average: 0.66
1 0 0
0 2 0
0 0 3

Time: 0.0012 s>
В выходном файле также указывается количество времени (в секундах), затраченное программой на обработку данных.

Краткий отчет

1. Количество заголовочных файлов: 0 шт

2. Количество модулей реализации: 6, Размер: 25,4 kB

3. Время, затраченное на каждый тест, указано в выходном файле соответствующего теста

Сравнение реализаций

C++ процедурный подход	C++ ООП подход	Python динамическая типизация, ООП подход
Время на обработку 10000 элементов после каждого этапа: Генерация: 0,884 сек Сортировка: 17,343 сек Удаление: 17,351 сек	Время на обработку 10000 элементов после каждого этапа: Генерация: 0,717 сек Сортировка: 16,283 сек Удаление: 16,284 сек	Время на обработку 10000 элементов после каждого этапа: Генерация: 4,9794 сек Сортировка: 156,2371 сек Удаление: 156,2541 сек
Отсутствие удобства в реализации, плохая читаемость кода, уступает времени исполнения ООП подходу	Простая реализация, высокая читабельность кода, возможность простой поддержки кода.	Самая простая реализация, однако сильно проигрывает во времени строго типизированным языкам.

Отличие NASM реализации

1. В зависимости от реализации сравнивать временные различия NASM реализации от других некорректно, однако можно отметить, что при том факте, что при сортировке сортируется более 90 млн элементов не более чем за минуту (64 сек), говорит о том, что сортировка работает в несколько раз быстрее C++ реализации и в сотни раз Python.

Отличие NASM реализации

2. Сложность написания кода возрастает в несколько раз, плохая читабельность кода, трудная поддержка проекта.