Арунова Анастасия БПИ208

Пояснительная записка

Описание полученного задания

Вариант: 282

Начальное условие задач: 2.

Обобщенный артефакт, используемый в задании	Базовые альтернативы (уникальные параметры, задающие отличительные признаки альтернатив)	Общие для всех альтернатив переменные	Общие для всех альтернатив функции
2. Плоская геометрическая фигура, размещаемые в координатной сетке.	1. Круг (целочисленные координата центра окружности, радиус) 2. Прямоугольник (целочисленные координаты левого верхнего и правого нижнего углов) 3. Треугольник (целочисленные координаты трех углов)	Цвет фигуры (перечислимый тип) = {красный, оранжевый, желтый, зеленый, голубой, синий, фиолетовый}	Вычисление периметра фигуры (действительное число)

Функция обработки данных в контейнере: 21.

Удалить из контейнера те элементы, для которых значение, полученное с использованием функции, общей для всех альтернатив меньше, чем среднее арифметическое для всех элементов контейнера, полученное с использованием этой же функции. Остальные элементы передвинуть к началу контейнера с сохранением порядка.

Требуемые метрики, определяющие характеристики программы, для различных тестовых прогонов.

Программа содержит 0 интерфейсных модулей (заголовочных файлов) и 7 модулей реализации (файлов с определением программных объектов):

- `delete.asm` – 2,89 КБ
- `in_random.asm` – 8,13 КБ
- `input.asm` – 9,96 КБ
- `output.asm` – 9,55 КБ
- `main.asm` – 8,21 КБ
- `perimeter.asm` – 5,44 КБ
- `macros.mac` – 4,58 КБ

Общий размер исходных текстов: 49,2 КБ.

Время выполнения программы для различных тестовых наборов данных:

Количество фигур	Время работы программы при вводе данных из файла	Время работы программы при случайной генерации данных
1 фигура	0,004 sec	0,003 sec
3 фигуры	0,004 sec	0,003 sec
8 фигур	0,005 sec	0,004 sec
12 фигур	0,005 sec	0,004 sec
14 фигур	0,005 sec	0,004 sec
100 фигур	0,01 sec	0,006 sec
1000 фигур	0,05 sec	0,03 sec
10000 фигур	0,6 sec	0,5 sec

Результаты тестов

Программа была протестирована на 12 тестах. Все тесты расположены в папке проекта «…\CSA_HW4\tests» и имеют название «test<i>.txt», где i – числа от 0 до 11. Причём некорректный ввод проверяют файлы «test1.txt» (некорректно заданные фигуры), «test7.txt» (пустой контейнер), «test8.txt» (переполнение контейнера). При некорректном цвете фигуры, её координатах программа генерирует их случайным образом. Тесты «test9.txt», «test10.txt», «test11.txt» написаны для получения времени работы программы на 10000, 100 и 1000 фигурах соответственно.

Результаты прохождения i-го теста записываются в два файла: «output<i>.txt» и «output_deleted<i>.txt». В первом из них содержится информация о контейнере, во втором информация о контейнере после применении функции, данной в задании. Файлов «output7.txt» и «output_deleted7.txt», «output8.txt» и «output_deleted8.txt» нет, так как на соответствующих им тестах программа в виду некорректных данных заканчивает работу. Также проект содержит два файла: «outputR.txt» и «output_deletedR.txt», которые были получены в ходе тестирования программы на случайно сгенерированном контейнере. Все файлы с результатами работы находятся в папке проекта «…\CSA_HW4\outputs».

Формат аргументов командной строки и тестов

Для тестирования программы на тестах необходимо ввести следующую команду (i – число от 1 до 11):

```
-f tests/test<i>.txt outputs/output<i>.txt outputs/output_deleted<i>.txt
```

Для тестирования программы на случайно сгенерированных тестах необходимо ввести следующую команду:

```
-n <number> outputs/outputR.txt outputs/output_deletedR.txt
```

Также в обоих случаях допустимо указание других файлов вывода.

Фигуры в тестах написаны в следующем формате:

```
<figure_key> <coordinates> <color_number>
```

Номер фигуры – цифра от 1 до 3. Цифра 1 соответствует кругу, 2 – прямоугольнику, 3 – треугольнику. Номер цвета – цифра от 0 до 6, где каждая цифра соответствует цвету радуги по порядку. Координаты – числа, количество которых для каждой фигуры своё. У круга 3 числа в следующем порядке: x-координата цента, y-координата центра и радиус. У прямоугольника 4 числа в порядке x-координата левого верхнего угла, y-координата левого верхнего угла, x-координата правого нижнего угла, y-координата правого нижнего угла. У треугольника 6 чисел, аналогично по порядку три (x, y) координаты вершин.

Разница в работе данной и предыдущих программ

Время		Процедурный подход C++	ООП подход C++	Динамическая типизация, ООП Python	NASM
Файловый ввод	1 фигура	0,01 sec	0,007 sec	0,002 sec	0,004 sec
	3 фигуры	0,02 sec	0,008 sec	0,004 sec	0,004 sec
	8 фигур	0,04 sec	0,008 sec	0,004 sec	0,005 sec
	12 фигур	0,04 sec	0,01 sec	0,006 sec	0,005 sec
	14 фигур	0,05 sec	0,01 sec	0,006 sec	0,005 sec
	100 фигур	0,11 sec	0,015 sec	0,02 sec	0,01 sec
	1000 фигур	0,38 sec	0,02 sec	0,4 sec	0,05 sec
	10000 фигур	1,95 sec	0,23 sec	1,7 sec	0,6 sec
Рандомная генерация	1 фигура	0,01 sec	0,003 sec	0,002 sec	0,003 sec
	3 фигуры	0,03 sec	0,003 sec	0,002 sec	0,003 sec
	8 фигур	0,04 sec	0,004 sec	0,005 sec	0,004 sec
	12 фигур	0,04 sec	0,004 sec	0,007 sec	0,004 sec
	14 фигур	0,04 sec	0,004 sec	0,009 sec	0,004 sec
	100 фигур	0,08 sec	0,006 sec	0,02 sec	0,006 sec
	1000 фигур	0,32 sec	0,04 sec	0,2 sec	0,03 sec
	10000 фигур	2 sec	0,4 sec	2 sec	0,5 sec

Программа, написанная на NASM, работает значительно быстрее, чем прошлые три программы.

Память	Процедурный подход C++	ООП подход C++	Динамическая типизация, ООП Python	NASM
Размер текстов	18,5 КБ	18,9 КБ	16,1 КБ	49,2 КБ

Исходные тексты по сравнению с предыдущими домашними работами занимают намного больше памяти, что может быть связано с тем, что код на NASM получается очень объёмным.

Дополнительные задачи

Из дополнительных задач было реализовано: модульная структура, поясняющие комментарии, большое количество тестовых файлов, сделанных своим генератором, обработка некорректных чисел во входных данных.