Национальный исследовательский университет «Высшая школа экономики»

Домашнее задание №2

Архитектура вычислительных систем

Пояснительная записка

Исполнитель

студент группы БПИ202

Галюта Ева Валерьевна

2021 г.

Задание

Вариант: 46

Номер задачи: 4

Номер функции: 6

Составление программы по условию(C++):

* Обобщенный артефакт, используемый в задании: объемная (трехмерная) геометрическая фигура.
* Базовые альтернативы (уникальные параметры, задающие отличительные признаки альтернатив):

1. Шар (целочисленный радиус)
2. Параллелепипед (три целочисленных ребра)
3. Правильный тетраэдр (длина ребра – целое)

* Общие для всех альтернатив переменные:

1. Плотность материала фигуры (действительное число)

* Общие для всех альтернатив функции

1. Вычисление площади поверхности (действительное число)

* Расположение всех альтернатив в едином контейнере.
* Упорядочивание элементов контейнера по возрастанию используя Shaker Sort. В качестве ключей для сортировки и других действий используются результаты функции, общей для всех альтернатив.
* Формат ввода:

1. Готовые тестовые файлы (./figures\_hw2 -f <input file with data> <output file> <output sorted file>) input file:
2. Первый аргумент тип фигуры число от 1 до 3 (1 – сфера, 2 – параллелепипед, 3 – правильный тетраэдр).
3. Второй аргумент уникальный (сфера – радиус (целочисленное значение), параллелепипед – 3 стороны (целочисленные значения), правильный тетраэдр – сторона (целочисленное значение)).
4. Третий аргумент общий для всех – плотность материала фигуры – действительное число.
5. Генерация случайных значений (./figures\_hw2 -n <number of figures> <output file> <output sorted file>)

* [Новое] Использование объектно-ориентированного подхода и статистической типизации

Структурная схема изучаемой архитектуры ВС с размещенной на ней разработанной программы

Таблица типов

|  |  |
| --- | --- |
| int  double | 4 байта [0]  8 байт [4] |
| struct Sphere | 12 байта |
| radius : int  density : double (struct Shape) | 4 байта [0]  8 байт[4] |
| struct Parallelepiped | 20 байт |
| a, b, c : int  density : double (struct Shape) | 3 \* 4 байт (12 байт) [0, 4, 8]  8 байт [12] |
| struct Tetrahedron | 12 байта |
| edge : int  density : double (struct Shape) | 4 байта [0]  8 байт[4] |
| Struct Container | 600008 байт |
| max\_size : int  len : int  list : Shape[30000] | 4 байта [0]  4 байта [4]  20 \* 30000 байт (600000 байт) [8] |
| Struct Shape | 8 байт |
| density : double | 8 байт[0] |

Память программы

|  |  |
| --- | --- |
| main(int argc, char \*argv[])  argc : int  argv : char\*  container : Container  size : int | 4 байта [0]  8 байт [4]  600008 байт [12]  4 байта [600020] |
| void ErrMessage1 |  |
| void ErrMessage2 |  |
| void Container::In(FILE \*file) |  |
| void Container::InRnd(int size) |  |
| void Container::ShakerSort(int size) |  |
| bool swapped  start, end, i : int  temp : Shape | 1 байт[0]  3 \* 4 байта (12 байт)[1, 5, 9]  20 байт [13] |
| void Container::Out(FILE \*fileOutout) |  |
| Int i | 4 байта[0] |
| void Container::Clear() |  |
| Int i | 4 байта[0] |

Стек вызовов (| - вызов одной из функций, ? – поведение программы при вызове функции)

|  |
| --- |
| main  ErrMessage1? – ~~main~~  Container::In (error – ~~main~~) | Container::InRnd | ErrMessage2? – ~~main~~  ~~Container::In~~ | ~~Container::InRnd~~  Container::Out  ~~Container::Out~~  Container::ShakeSort  ~~Container::ShakeSort~~  Container::Out  ~~Container::Out~~  Container::Clear  ~~Container::Clear~~  ~~main~~ |

Характеристики программы

Интерфейсные модули: 6

Модули реализации: 7

Общий размер исходных тестов: 496 Кб

Общий размер результатов тестов: 4,1Мб

Результаты тестов (в секундах)

|  |  |  |
| --- | --- | --- |
| Количество элементов | Ввод из файла | Генерация элементов |
| 20 (test\_1\_20.txt) | 0.000867 | 0.000766 |
| 100 (test\_2\_100.txt) | 0.002346 | 0.001917 |
| 1000 (test\_3\_1000.txt) | 0.069392 | 0.055530 |
| 10000 (test\_4\_10000.txt) | 3.914274 | 3.829114 |
| 20000 (test\_5\_20000.txt) | 12.225400 | 11.842741 |

Различия во времени не зависят от типа ввода, минимальные различия может выдавать сортировка, так как генерируемые данные могут быть частично упорядочены.

Заключение

Разработали программный продукт с использованием объектно-ориентированного подхода и статистической типизации. ООП значительно упростило читаемость, а также написание кода, что крайне важно в процессе разработки, а также уменьшила расход памяти, не пришлось хранить в обобщающем классе ссылки на каждого наследника. Можно заметить, что сама программа стала отрабатывать быстрее, чем в предыдущей версии. Подводя итог, можно сказать, что подход с использование ООП намного приятнее для работы.