Пояснительная записка к Домашнему заданию №2 по курсу Архитектура вычислительных систем

Шагаров Дмитрий Александрович
БПИ202

Октябрь
2021

Описание полученного задания

номер варианта	номер задания	номер функции
70	14	5

Разработка сущностей

1. **Грузовик** (грузоподъемность кг – целое, емкость топливного бака в литрах (целое), расход топлива на 100 км в литрах (действительное))
2. **Автобус** (пассажировместимость – короткое целое, емкость топливного бака в литрах (целое), расход топлива на 100 км в литрах (действительное))
3. **Легковой автомобиль** (максимальная скорость – короткое целое, емкость топливного бака в литрах (целое), расход топлива на 100 км в литрах (действительное))

Каждая сущность имеет функцию находения максимального расстояния, которое может пройти автомобиль в км (действительное число).

Все введеные сущности размещаются в разработанном контейнере, после чего к контейнеру применяется сортировка Шелла (ключ - значение функции максимального расстояния). Элементы контейнера до и после сортировки выводется в форматируемый поток.

Формат ввода

Программа расположена по пути `AoCS/Task2/bin/Task2`.
В программе предусмотрено два способа ввода данных при запуске из командной строки:

1. **Использование генераторов случайных наборов данных** - команда
   `./Task2 -n number_of_vehicles out_path sorted_out_path`
2. **Ввод из заранее подготовленных тестовых файлов** - команда
   `./Task2 -f in_path out_path sorted_out_path`

Второй способ предусматривает наличие файла in с описанием сущностей в формате: "одна строка = одно ТС.
Первый параметр - число от 1 до 3, где 1 - легковое авто, 2 - автобус, 3 - грузовик.
Второй параметр - уникальное для каждого ТС свойство 80 - 320 целое для легкового авто, 10 - 70 целое для автобуса, 1000 - 4000 целое для грузовика.
Третий параметр - объем топливного бака 50 - 150 целое для легкового авто, 80 - 250 целое для автобуса, 200 - 800 целое для грузовика.
Последний параметр - расход 6 - 25 вещественное для легкового авто, 12 - 30 вещественное для автобуса, 20 - 45 вещественное для грузовика.
Параметры указываются через один пробел, для вещественных разделитель - точка. Окончание файла - пустая строка.
Пример такого файла - `in_example.txt` находится в папке с проектом. Для корректной работы программы необходимо наличие файла, передаваемого как `in_path`.

Структурная схема программного продукта с использованием объектно-ориентированного подхода и статической типизации.

Таблица типов

short, int, double	2 байта, 4 байта, 8 байт
class Vehicle	20 байт
int tank_volume	4 байта[0]
double consumption	8 байт[4]
указатель на таблицу виртуальных методов	8 байт[12]
class Car	22 байта
Vehicle поля	20 байт[0]
short max_speed	2 байта[20]
class Bus	22 байта
Vehicle поля	20 байт[0]
short max_passengers	2 байта[20]
class Truck	24 байта
Vehicle поля	20 байт[0]
int max_weight	4 байта[20]
class Container	240032 байта
int len, size	8 байт[0, 4]
Vehicle **storage	max 240024 байта[8] (= 24 * 10001)

Память программы

main(int argc, char *argv[]) int argc char *argv Container c int size	 4 байта[0] 8 байт[4] 240032 байта[8] 4 байта[240040]
void StartMessage	
void ErrMessage1	
void ErrMessage2	
void Container::In(FILE *input)	
void Container::InRnd(int size)	
void Container::ShellSortByMaxDistance() int d, i, j Vehicle *temp	 12 байт[0, 4, 8] 24 байта[12]
double Vehicle::MaxDistance()	
void Container::Out(FILE *output) int i	 4 байта[0]
void Container::Clear() int i	 4 байта[0]

Class::method значит, что это метод класса

Стек вызовов

Возможны следующие варинты (| означает что вызыается одна из соответствующих функций, ? ? → - что будет, если программа вызовет данную функцию)

main
?ErrMessage1? → ~~main~~
StartMessage
~~StartMessage~~
Container::In (error → ~~main~~) | Container::InRnd | ?ErrMessage2? → ~~main~~
~~Container::In~~ | ~~Container::InRnd~~
Container::Out
~~Container::Out~~
Container::ShellSortByMaxDistance
~~Container::ShellSortByMaxDistance~~
Container::Out
~~Container::Out~~
Container::Clear
~~Container::Clear~~
~~main~~

Основные характеристики программы

интерфейсных модулей	модулей реализации
6	6 (+ main.cpp для тестирования)

общий размер исходных текстов	общий размер результатов тестов
12,82 Кб	12,6 Мб

Результаты тестов

Тесты расположены в папке tests и были сгенерированы с использованием написанного случайного генератора тестов. Результаты расположены в папке test_results.

Время работы программы в тестах (в секунднах)		
	тесты из файлов	случайная генерация в программе
1 элемент	0,000462	0,000487
50 элементов	0,000558	0,000893
500 элементов	0,003095	0,002449
2500 элементов	0,021011	0,018616
6000 элементов	0,083185	0,080179
10000 элементов	0,233643	0,227567

Как можно видеть, особо существенной разницы в зависимости от типа ввода нет. Разброс скорее обусловлен степенью начальной сортировки контейнера. Но файловый ввод все же слегка медленнее, так как требуется время, чтобы прочитать данные из файла.

Заключение

Объектно-ориентированный подход с использованием статической типизации показал заметно лучшие результаты как по времени (что для меня было немного неожиданным), так и по памяти (вполне ожидаемо). Память уменшилась в связи с тем, что в обобщенном ТС теперь не хранятся ссылки на каждое из возможных ТС. Кроме того, ООП позволяет писать более читабельный и лаконичный код с возможностью переиспользования кода (что так же отразилось на размерах исходников) (за счет методов класса и наследования). Однако опять же, ОПП в C++ заметно слабее (по удобности и возможностям) чем в том же `C#` или Java. Но, безусловно, даже так ООП подход кажется намного лучше процедурного.