Архитектура вычислительных систем.

Вариант 300

Задание 6, Функция 22

Выполнил:  
Лобанович Михаил Михайлович,  
группа БПИ-208

**Задача**

**Сущности:**

1. Самолеты (дальность полета – целое, грузоподъемность – целое)

2. Поезда (количество вагонов – целое)

3. Корабли (водоизмещение – целое; вид судна – перечислимый тип = (лайнер, буксир, танкер)

**Общие для всех альтернатив переменные:**

1. Скорость – целое;

2. Расстояние между пунктами отправления и назначения – действительное.

**Общая для всех альтернатив функция:**

Идеальное время прохождения пути (действительное число).

**Общая для все альтернатив структура:**

Транспорт.

### **Функционал**

1. Провести отладку и тестирование разработанной программы на заранее подготовленных тестовых наборах данных. При необходимости, программа должна правильно обрабатывать переполнение по данным. Тестовые наборы до 10 элементов должны вводиться из заранее подготовленных тестовых файлов с использованием сведений из рассматриваемой в задаче предметной области. Количество тестовых наборов данных (тестов, сделанных вручную), в которых четко представлены параметры предметной области (корректные названия, имена, фразы) – не менее пяти. Число уникальных элементов в тестовых наборах должно варьироваться от нуля до 10000. Тестовые данные с большим числом элементов должны порождаться программно с использованием генераторов случайных наборов данных. Данные, формируемые генератом случайных наборов должны поддерживать допустимые для обработки значения. Управление вводом данных задается из командной строки.
2. Использование модульной структуры и классов для представления программных объектов дает дополнительные бонусы к базовой оценке.
3. Описать архитектуру виртуальной Python машины с наложением на нее обобщенной схемы разработанной программы.
4. Зафиксировать для отчета основные характеристики программы, такие как: число интерфейсных модулей (заголовочных файлов) и модулей реализации (фалов с определением программных объектов), общий размер исходных текстов, полученный размер исполняемого кода (если он формируется), время выполнения программы для различных тестовых наборов данных.
5. Привести результаты сравнительного анализа полученных характеристик с теми, которые были получены для предыдущей программы. Сделать выводы о достоинствах и недостатках этого и предшествующего решения относительно друг друга.

**Команды для запуска**

**Случайная генерация**

-n \*число элементов( 0 < x < 10001)\* \*первый файл для вывода\* \*второй файл для вывода\*

**Чтение из файла**

-f \*имя файла с входными данными\* \*первый файл для вывода\* \*второй файл для вывода\*

--1--.

1. Интерфейсные модули:

transport.h

ship.h

random.h

train.h

plane.h

container.h

Всего: 6

2) Модули реализации:

transport.cpp

plane.cpp

train.cpp

ship.cpp

container.cpp

main.cpp

Всего: 6

3) Общий размер исходных текстов = 4 + 2 + 3 + 2 + 2 + 1 + 3 + 2 + 1 + 1 + 4 + 2 = 27КБ

4)Время выполнения программы для различных тестовых наборов данных: (тесты рассчитаны на значения в диапазоне от 1 до 20 вводимых элементов, начиная с 500 элементов программа тестировалась при помощи генерации случайных входных данных)

1 тест (1 элемент):

0.001 sec

2 тест (3 элемента):

0.001 sec

3 тест (5 элементов):

0.001 sec

4 тест (10 элементов):

0.001 sec

5 тест (15 элементов):

0.002 sec

6 тест (20 элементов):

0.002 sec

7 тест (500 элементов):

0.006 sec

8 тест(2500):

0.02299 sec

9 тест(5000 элементов):

0.0429 sec

10 тест(10000 элементов):

0.064 sec

--2--.

Программа загружена в систему Github.

--3--.

Тесты находятся в папке cmake-build-debug/tests

--4--.

1. **Описание структуры ВС**

**ТАБЛИЦА КЛАССОВ**

**Базовые Классы**

|  |  |
| --- | --- |
| **1** | **Container [80004 байт]** |
| Поля | **Static:** Отсутствует  **Local:** int len [4 байта]  Number \*storage[80000] байт |
| Методы | **Static**: -  **Local:**  void Clear();  void In(FILE \*file);  void InRandom(int size);  void Out(FILE \*file);  void MoveToTheEnd() ; |

**Базовый абстрактный класс**

|  |  |
| --- | --- |
| **1** | **Number** |
| Поля | **Static: -**  **Local: -** |
| Методы | **Static: (ГП)**  Transport \*StaticIn(FILE \*file);  Transport \*StaticInRnd(FILE \*file)  MoveToTheEnd();  Local: ~Number ();  Random rnd(); |

**Производные классы**

|  |  |
| --- | --- |
| **1** | **Plane: Transport[8 байт]** |
| Поля | **Static:** -  **Local:**  int range 4[0]  int lifting\_capacity 4[4] |
| Методы | **Static:** -  Local(override): ~Plane();  void In(FILE \*fileIn);  void InRandom();  void Out(FILE \*fileOut); |
| **2** | **Train: Transport [4 байта]** |
| Поля | **Static: -**  **Local:**  int wagons\_number 4[0] |
| Методы | Static: -  Local: ~ Train();  void In(FILE \*file);  void InRandom();  void Out(FILE \*file); |
| **3** | **Ship: Transport [8 байт]** |
| Поля | **Static**: -  **Local:**  int displasement 4[0]  enum ship\_types[4] |
| Методы | **Static:** -  **Local**: ~ Ship();  void In(FILE \*file);  void InRandom();  void Out(FILE \*file);  void printType(FILE \*file); |

**ЛОКАЛЬНЫЕ МЕТОДЫ**

|  |  |
| --- | --- |
| Container |  |
| Void In(Container \*self, FILE \*file)  Void InRnd(Container \*self, int size)  void Out(Container \*self)  Double MoveToTheEnd(Container \*self) |  |

ТАБЛИЦА ВИРТУАЛЬНЫХ МЕТОДОВ БК

|  |  |
| --- | --- |
| Void In(Number \*self, FILE \*file)  Void InRnd(Number \*self, int size)  void Out(Number \*self) |  |

ТАБЛИЦА ВИРТУАЛЬНЫХ МЕТОДОВ ПК

|  |  |
| --- | --- |
| Void In(Number \*self, FILE \*file)  Void InRnd(Number \*self, int size)  void Out(Number \*self) |  |

**ПАМЯТЬ ПРОГРАММЫ**

|  |  |
| --- | --- |
| Main (80032 байт) | |
| Container c | 80016 |
| Int size | 4[80020] |
| FILE \*input | 4[80024] |
| FILE \*output1 | 4[80028] |
| FILE \*output2 | 4[80032] |

**Формат ввода:**

Для ввода одного транспорта следует ввести следующие данные:

Формат ввода меняется в зависимости от типа вводимого транспорта:

1. Самолет: X1 X2 X3 X4 X5,

Где X1 – 1, тип самолета

X2 – скорость самолета

X3 - расстояние между пунктами отправления и назначения

X4 – дальность полета

X5 - грузоподъемность

1. Поезд: X1 X2 X3 X4,

Где X1 – 2, тип поезда

X2 – скорость поезда

X3 - расстояние между пунктами отправления и назначения

X4 – количество вагонов

1. Корабль: X1 X2 X3 X4 X5,

Где X1 – 3, тип корабля

X2 – скорость корабля

X3 - расстояние между пунктами отправления и назначения

X4 – водоизмещение

X5 – тип корабля из перечисляемых

**Расположение входных и выходных данных**

Входные данные лежат в:

cmake-build-debug/tests/input

Выходные лежат в:

cmake-build-debug/tests/output

**Cхема**

1. HEAP (Куча) - int\*
2. Стек

Изображение выглядит как текст, доска

Автоматически созданное описание

**Сравнительная таблица**

**По памяти**

|  |  |
| --- | --- |
| Процедурный подход | ОО подход |
| 24кб | 27кб |

**По времени**

|  |  |
| --- | --- |
| Процедурный подход | ОО подход |
| 0.001 | 0.001 |
| 0.001 | 0.001 |
| 0.001 | 0.001 |
| 0.001 | 0.001 |
| 0.001 | 0.002 |
| 0.001 | 0.002 |
| 0.003 | 0.006 |
| 0.01099 | 0.02299 |
| 0.021 | 0.0429 |
| 0.04599 | 0.064 |

Из таблицы можно понять, что размер файлов, содержащих код, практически идентичен. Это связано со схожей структурой выполнения обоих заданий с точки зрения композиции в файлах. В то же время благодаря объектно-ориентируемому подходу повышается понятность кода, он становится более структурированным с точки зрения построения проекта. Такой код был бы удобнее при работе команды над одним кодом. Однако время работы программы увеличилось. На небольших тестах оно осталось практически прежним, однако при повышении количества обрабатываемых элементов виден значительный рост времени работы.