Сушкова Дарья, БПИ204

Пояснительная записка

Описание полученного задания:

Задача номер 6 (пассажирский транспорт), функция номер 11 (сортировка выбором по убыванию).

- ✓ Обобщенный артефакт, используемый в задании – Пассажирский транспорт;
- ✓ Базовые альтернативы и их уникальные параметры:
  1. *Самолет*: дальность полета – целое, грузоподъемность – целое;
  2. *Поезд*: количество вагонов – целое;
  3. *Корабль*: водоизмещение – целое, вид судна – перечислимый тип (лайнер, буксир, танкер).
- ✓ Общие для всех альтернатив переменные: скорость – целое, расстояние между пунктами отправления и назначения – действительное.
- ✓ Общая для всех альтернатив функция: идеальное время прохождения пути (действительное).
- ✓ Функция: упорядочить элементы контейнера по убыванию используя сортировку Сортировка с помощью прямого выбора (Straight Selection). В качестве ключей для сортировки и других действий используются результаты функции, общей для всех альтернатив.

Структура проекта:

project/

tests/ - входные и выходные файлы для тестирования программы;

main.cpp – точка входа.

/…/ - заголовочные файлы и файлы реализации проекта.

Заголовочные файлы проекта: random.h, airplane.h, train.h, ship.h, transport.h, container.h

Файлы реализации проекта: main.cpp, airplane.cpp, train.cpp, ship.cpp, transport.cpp, container.cpp

Основные характеристики программы:

- Количество заголовочных файлов: 6
- Количество файлов реализации: 6
- Размер исходных файлов: ≈ 22.1 Кб
- Размер исполняемого файла: 221 Кб
- Время выполнения программы на различных тестовых файлах (взято усредненное значение по результатам нескольких запусков):

Количество элементов	*Тип входных данных*	*Время работы (mc)*
0	Ввод с файла	0.000711
2	Ввод с файла	0.000768
4	Ввод с файла	0.001140
8	Ввод с файла	0.001209
10	Ввод с файла	0.001483
10	Генерация данных	0.001698
100	Генерация данных	0.007228
1000	Генерация данных	0.016339
5000	Генерация данных	0.337841
10000	Генерация данных	1.396434

СРАВНЕНИЕ РЕЗУЛЬТАТОВ ПРОЦЕДУРНОГО ПОДХОДА И ОБЪЕКТНО-ОРИЕНТИРОВАННОГО:

Для обоих типов ввода данных время выполнения программы увеличилось в данной реализации. Это связано:

1) С наличием виртуальных методов, наличие которых в определенном классе приводит к необходимости хранения так называемых виртуальных таблиц. При позднем связывании вызову функции предшествует её поиск по указателю на объект, что, очевидно, приводит к временным затратам.
2) Реализация с помощью классов, а не с помощью структур, как в прошлой реализации, так как создание неизменяемой копии объекта эффективнее множества ссылок при переборе некоторого количества подобных объектов.

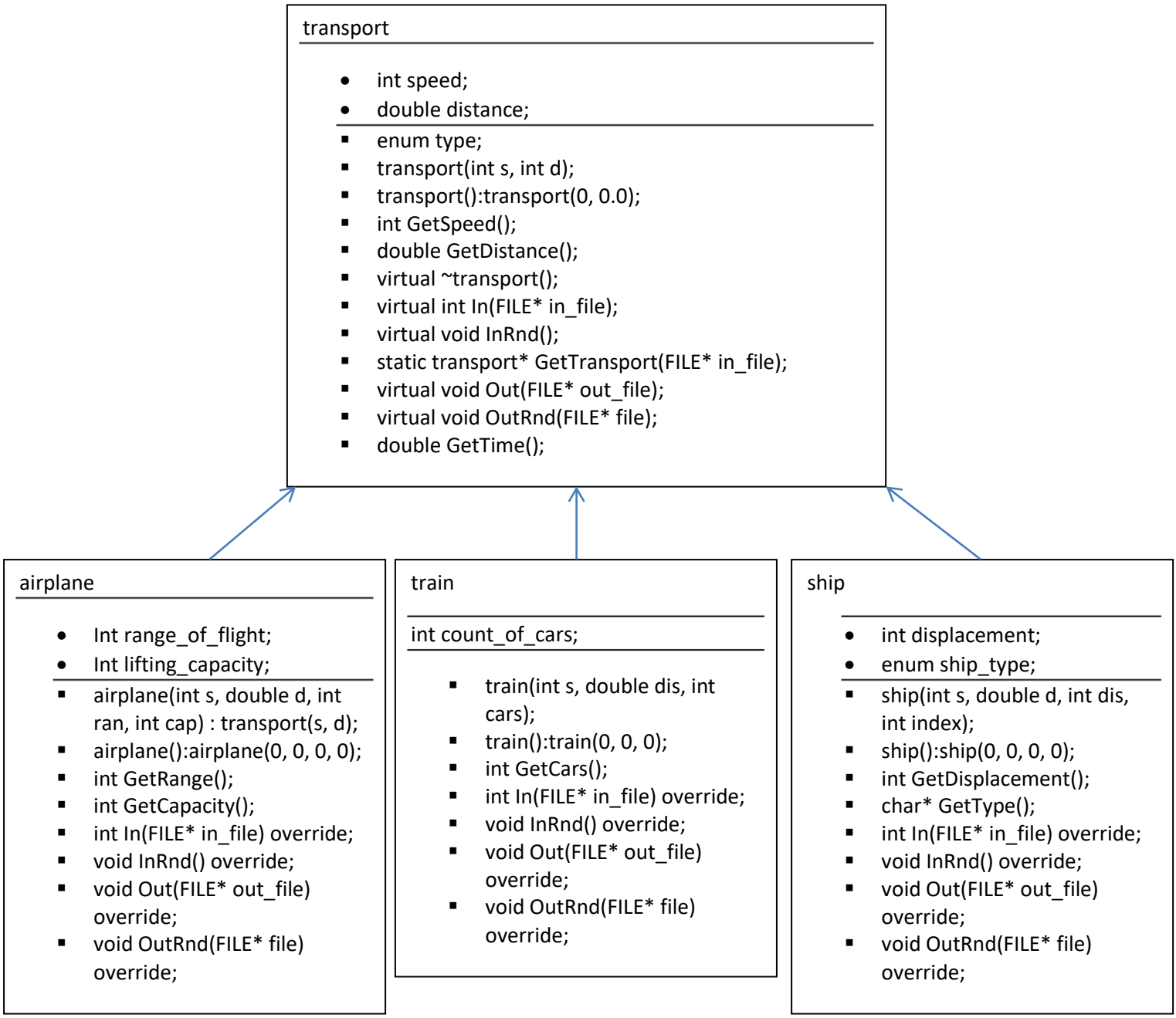
Примечание к тестовым файлам: для каждого типа входных данных отведено по 5 тестов, идущих по возрастанию количества обрабатываемых элементов. В выходном файле присутствуют и несортированный, и отсортированный

варианты. Пара входной-выходной файл имеет одинаковый индекс (1-5 для первого типа входных данных, 5-6 для второго типа входных данных).

Формат ввода в данной реализации для каждого типа входных данных свой. При вводе с файла описание элемента идет в порядке (тип – общие переменные – индивидуальные переменные), при генерации данных в порядке (общие переменные – тип – индивидуальные переменные).

Архитектура приложения:

Схема классов имеет следующий вид:



container

- static const int max_length = 10000;
- int length;
- transport** transports;

- container();
- ~container();
- int In(FILE* in_file);
- int InRnd(int count, FILE* file);
- void Out(FILE* out_file);
- void StraightSelection();

Инструментальные средства:

- Виртуальная машина Oracle VM VirtualBox (Linux, Ubuntu);
- Языки программирования: C/C++;
- IDE: Clion (совместно с CMake);
- Библиотеки: stdlib.h, stdio.h, time.h, math.h