Архитектура вычислительных систем

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ПРАКТИЧЕСКОЙ РАБОТЕ НА ТЕМУ:

«Статически типизированная архитектура ВС,
ориентированная на процедурный подход»

Работу выполнил:

студент группы БПИ207
Пендищук Владислав

Москва
2021 г.

ОГЛАВЛЕНИЕ

ОПИСАНИЕ ЗАДАНИЯ

Задача состояла в разработке программного продукта с использованием процедурного подхода и статической типизацией на языке C или C-style C++.

Требования к функционалу

В соответствии с полученным вариантом задания (258) функционал, требуемый к реализации в программе, содержал в себе следующие пункты:

1. Реализация обобщённого артефакта – пассажирского транспорта, и его параметров:
    a. Скорость – целое число;
    b. Расстояние между пунктами отправления и назначения – действительное число;
2. Реализация базовых альтернатив и уникальных параметров, задающих их отличительные признаки:
    a. Самолёт, отличительные признаки:
        i. Дальность полёта – целое число;
        ii. Грузоподъёмность – целое число.
    b. Поезд, отличительные признаки:
        i. Количество вагонов – целое число.
    c. Корабль, отличительные признаки:
        i. Водоизмещение – целое число;
        ii. Вид судна – перечислимый тип (лайнер, буксир, танкер).
3. Реализация общей для всех альтернатив функции – вычисления идеального времени прохождения пути (действительное число).
4. Реализация контейнера для объектов типа обобщённого артефакта с массивом фиксированной длины в своей основе.
5. Реализация функции удаления из контейнера тех элементов, 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.

Требования к запуску и вводу\выводу

К процессу запуска программы и ввода\вывода при работе с ней были представлены следующие требования:

1. Запуск программы осуществляется из командной строки, в которой указываются: имя запускаемой программы; имя файла с исходными данными; имя файла с выходными данными.
2. Для каждого программного объекта, загружаемого в контейнер, исходный файл с тестовым набором должен содержать: признак

альтернативы, а также список параметров, необходимых этой альтернативе. Этот список должен быть представлен в формате, удобном для обработки компьютером.

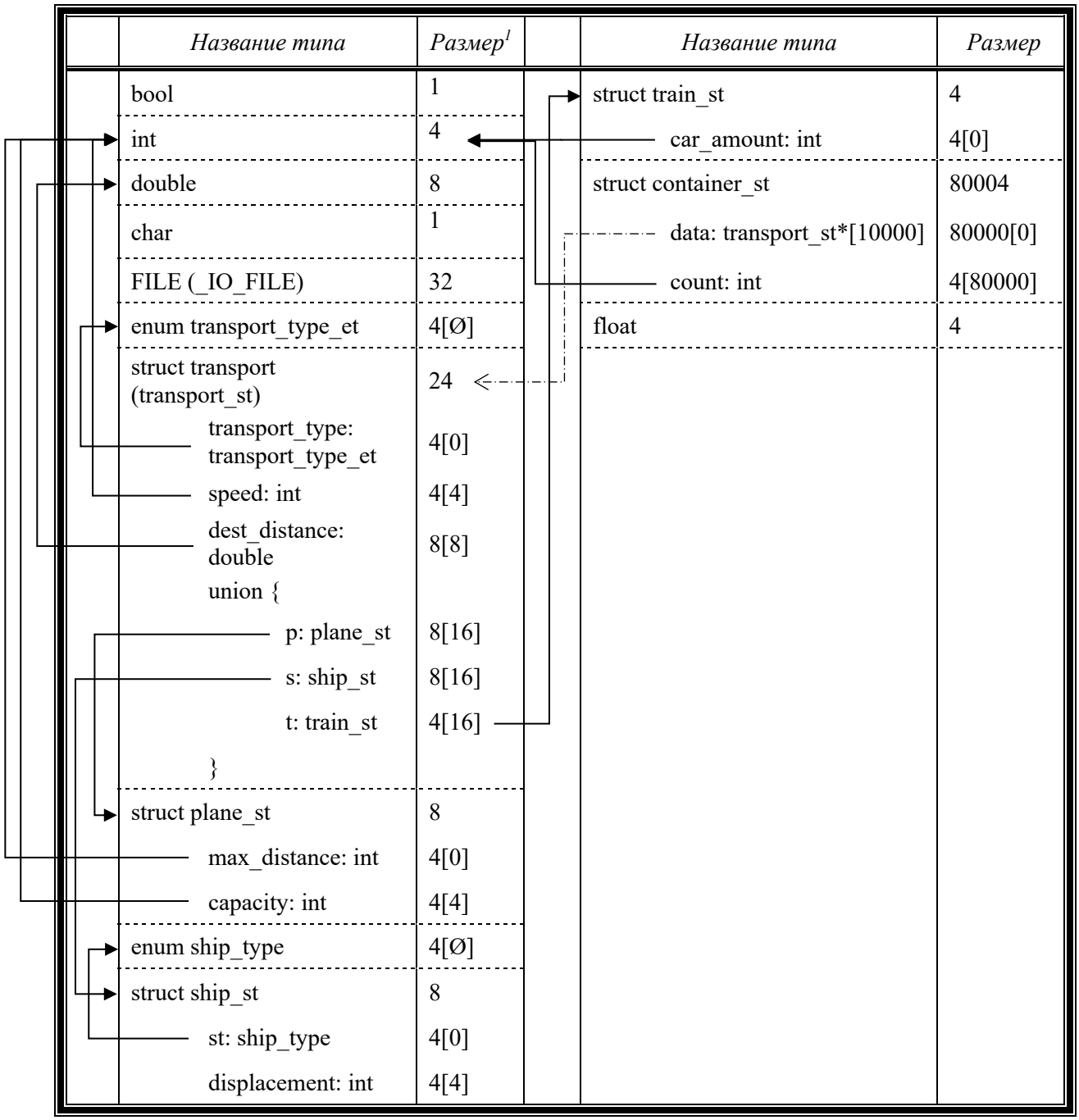
3. При больших данных во входном файле должны быть указаны только параметры для генератора случайных наборов данных, который и заполняет контейнер.
4. В выходной файл необходимо вывести введенные в контейнер данные. Помимо этого, необходимо вывести информацию об общем количестве объектов, содержащихся в контейнере. После этого в тот же файл необходимо вывести новые данные в соответствии с результатами, полученными в ходе работы программы. Информация для вывода должна быть представлена в форме, удобной для восприятия пользователем.

СТРУКТУРА ИЗУЧАЕМОЙ АРХИТЕКТУРЫ ВС

Объектом изучения в данной работе являлась статически типизированная архитектура ВС, ориентированная на процедурный подход. Разработка велась на языке С с соответствующими типами. Отобразим данную архитектуру на обобщённой схеме разработанной программы на примере 4 функций.

Таблица типов

Таблица типов, используемых в программе, необходима в силу статической типизации рассматриваемой архитектуры.



[1] Размеры даны в байтах для архитектуры x86-64.

Схема №1 – main

На данной схеме в обобщённом виде на статически типизированную архитектуру ВС отображена функция main:

Heap	*Память данных*[2]	
“proc-c-1”	argc: int	4[0]
“-f”	argv: char**	8[4]
“in.txt”	start: clock_t	4[12]
“out.txt”	c: container_st	80004[16]
…	ifstream: FILE*	8[80020]
	ofstream: FILE*	8[80028]
	end: clock_t	4[80036]
	seconds: float	4[80040]

Глобальная память	*Память команд (программы)*
0	`int main(int argc, char* argv[]) {…}`

Stack 1 (ввод из файла)	*Stack 2 (ошибка)*	*Stack 3 (ген. контейнера)*	*Stack 4 (ген. тестов)*
clock	clock	clock	clock
fprintf	fprintf	fprintf	fprintf
Initialize	…	Initialize	Initialize
strcmp	ArgNumError / InputModeError	strcmp	strcmp
fopen	fprintf	InRand	printf
In	exit	TransportInRand[n]	GenerateTestFiles
TransportIn[n]		Append[n]	snprintf[n]
Append[n]		Out	fopen[n]
Out		TransportOut[n]	GenerateTest[n]
TransportOut[n]		DeleteLessThanAverage	printf[n]
DeleteLessThanAverage		TimeToDest[n]	printf
TimeToDest[n]		RemoveAt[n]	
RemoveAt[n]		Out	
Out		TransportOut[n]	
TransportOut[n]			

[2] Память данных представлены для случая с успешным вводом из файла

Схема №2 – PlaneInRand

На данной схеме в обобщённом виде на статически типизированную архитектуру ВС отображена функция PlaneInRand:

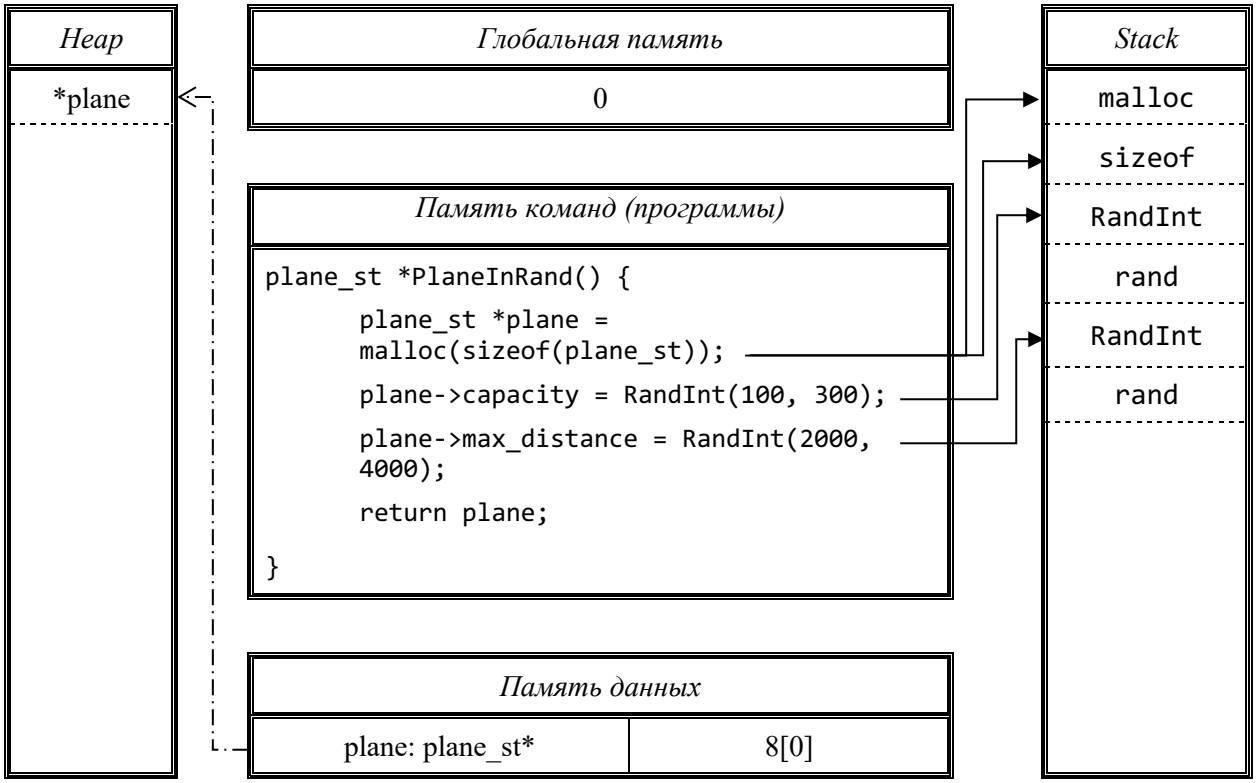


Схема №3 – DeleteLessThanAverage

На данной схеме в обобщённом виде на статически типизированную архитектуру ВС отображена функция DeleteLessThanAverage:

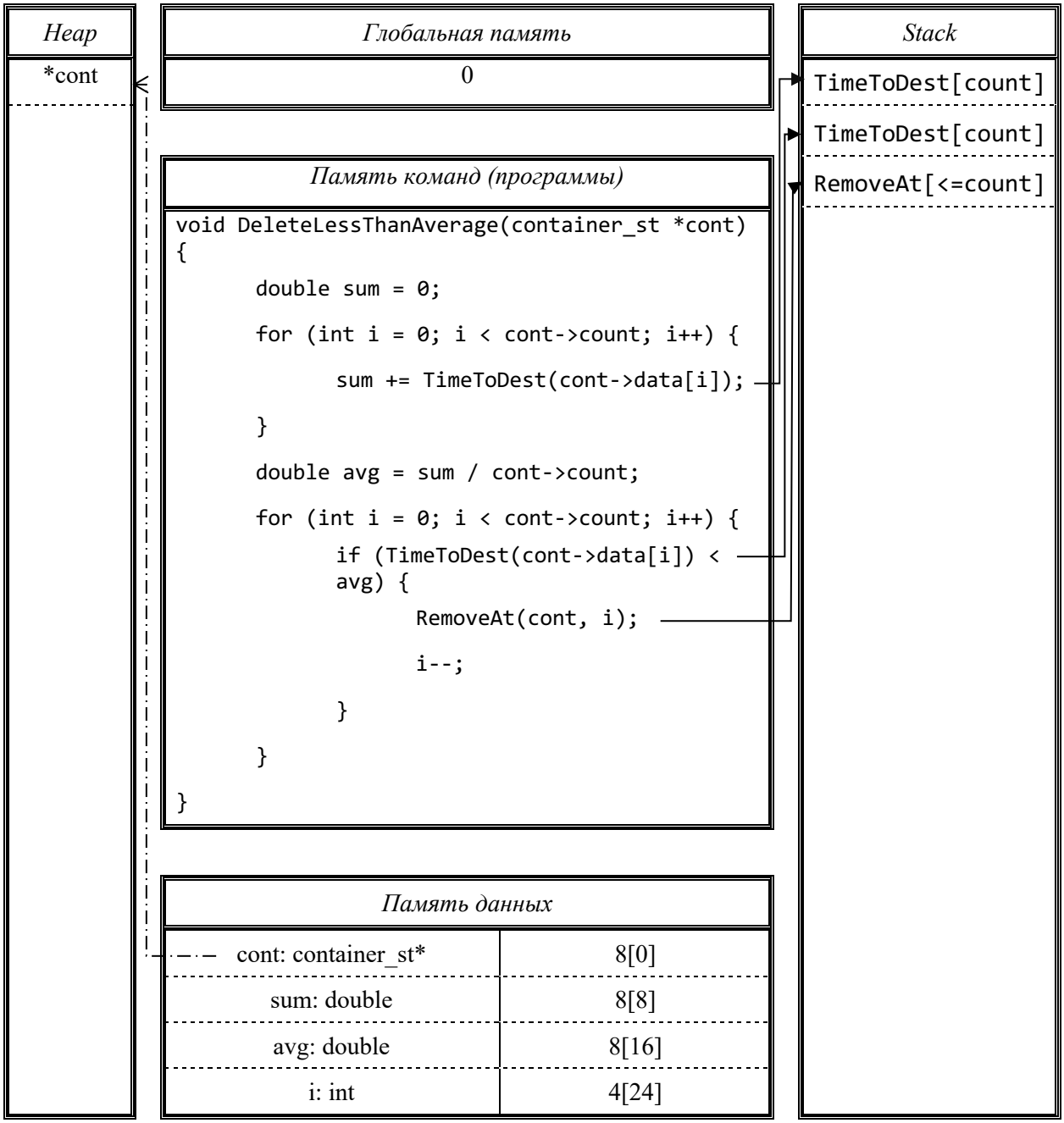
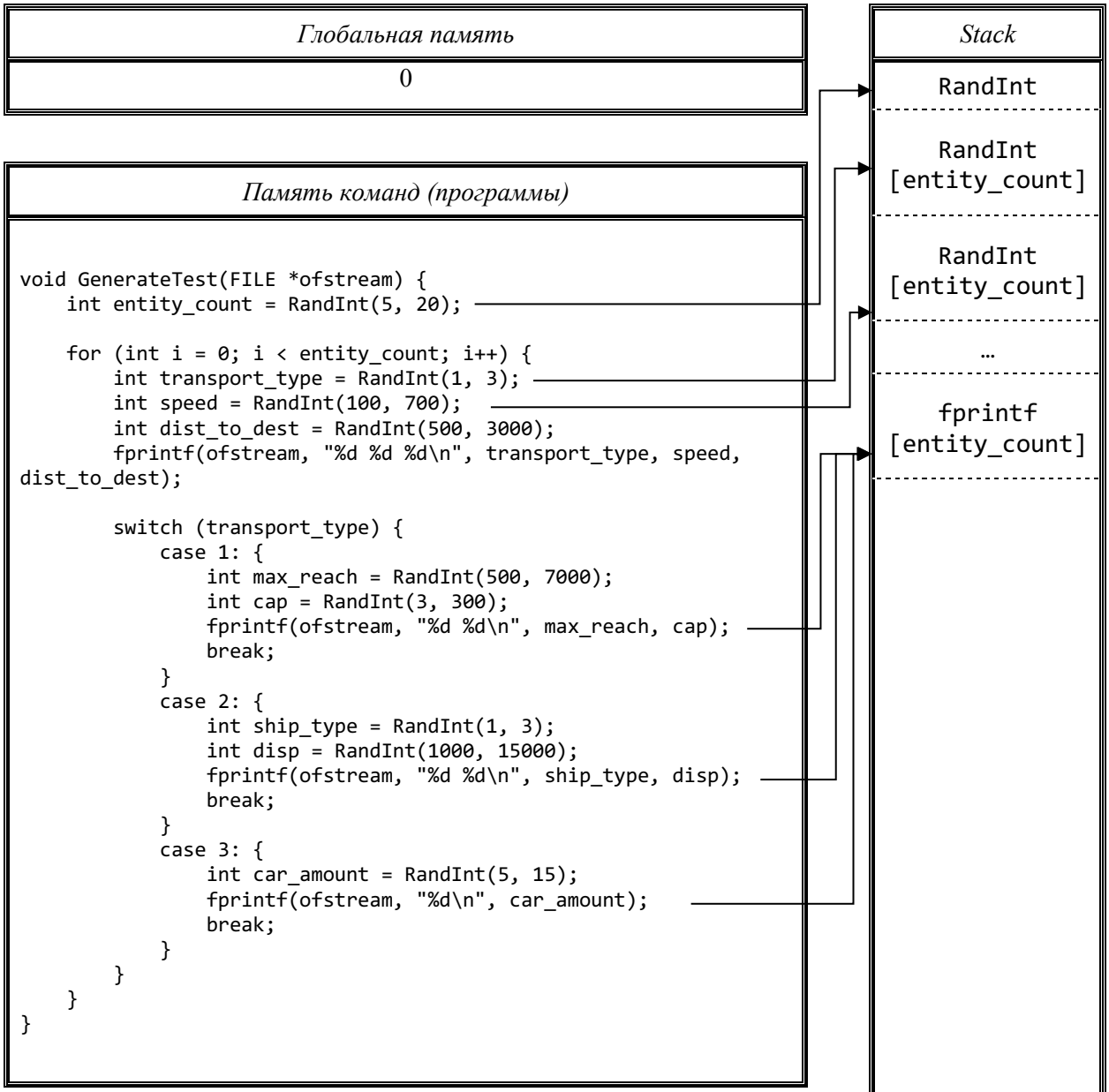


Схема №4 – GenerateTest

На данной схеме в обобщённом виде на статически типизированную архитектуру ВС отображена функция GenerateTest:



ОСНОВНЫЕ ХАРАКТЕРИСТИКИ ПО

Исходный код программы содержится в 17 файлах. 8 из них являются интерфейсными модулями (заголовочными файлами C), 9 – модулями реализации (файлами с определением программных объектов). Общий размер исходных текстов следующий:

	С форматированием и комментариями	*Без форматирования и комментариями*
С интерфейсными модулями	846	472
Без интерфейсных модулей	595	380

Размер исполняемого файла, полученного после компиляции кода на ОС Linux, равен 43896 байт (43,9 килобайт).

Результаты тестов с использованием тестовых файлов в соответствующих директориях следующие:

Тестовый файл	*Тестовый кейс*	*Время работы программы*
correct/test1.txt	5 корректных элементов	0.00021 секунды
correct/test2.txt	19 корректных элементов	0.00024 секунды
correct/test3.txt	13 корректных элементов	0.00022 секунды
error/test1.txt	Некорректное число аргументов в строке	0.00013 секунды
error/test2.txt	Некорректный идентификатор альтернативы	0.00019 секунды
error/test3.txt	Удалённая строка параметров	0.00016 секунды

Результаты тестов с использованием случайной генерации объектов контейнера следующие:

Количество элементов	*Время работы программы*
20 элементов	0.00023 секунды
1000 элементов	0.00298 секунды
10000 элементов	0.13457 секунды