Пояснительная записка

Описание полученного задания (вариант 12, функция 11):
Создать консольное приложение, обрабатывающее данные в контейнере.

- Обобщенный артефакт, используемый в задании - животные.
- Базовые альтернативы и уникальные параметры - рыбы (место проживания – перечислимый тип: река, море, озеро…), птицы (отношение к перелету: перелетные, остающиеся на зимовку – булевская величина), звери (хищники, травоядные, насекомоядные… – перечислимый тип).
- Общие для всех альтернатив переменные - название (строка символов), вес в граммах (целое).
- Функция для обработки данных в контейнере - упорядочить элементы контейнера по убыванию используя сортировку с помощью прямого выбора (Straight Selection). В качестве ключей для сортировки и других действий используются результаты функции, общей для всех альтернатив.
- Функция, общая для всех альтернатив - частное от деления суммы кодов незашифрованной строки на вес (действительное число).

Структура проекта:

```
project_4/
    asm/
        main.nasm
    bin/
        program.elf
    c-code/
        main.cpp
        container.h
        animal.h
        beast.h
        bird.h
        fish.h
        rnd.h
    tests/
    commands.txt
```

Работа программы:
Пользователь может ввести данные в контейнер двумя способами:

- Задать входной файл с тестовыми данными с помощью команды `-f *input path* *output path*` , где *input path* и *output path* - пути входного и выходного файлов соответственно.
- Задать параметры для генерации тестовых данных: `-n *size* *files path*` , где *size* - количество элементов в контейнере, *files path* - путь, куда будут сохранены сгенерированный файл с входными данными

(***_generated.txt) и с обработанными выходными данными (***_generated.output.txt).

Результаты тестирования программы сохранены в папке project/tests. Тесты 1-7 с использованием входных данных из файла, тесты 8-14 с использованием сгенерированных входных данных.

Примеры использованных команд лежат в файле commands.txt

Основные характеристики программы:

- Число интерфейсных модулей = 6
- Число модулей реализации = 1
- Общий размер исходных текстов ≈ 15.6 КБ
- Общий размер исполняемых файлов ≈23.1 КБ
- Время работы программы для различных тестовых наборов данных:

Тип входных данных	Кол-во элементов контейнере	Время со структурами на C (мс)	Время с классами на C++ (мс)	Время с дин. Типизацией на Python (мс)	Время на Ассемблере (мс)
Входные данные из файла	10	0.385	0.442	1.386	0.331
	20	0.418	0.521	3.801	0.464
Входные данные генерируются	100	0.929	0.989	43.298	3.826
	1000	25.607	33.843	2630.548	27.535
	5000	723.708	763.432	63081.304	387.973
	10000	3162.394	3538.942	212349.851	1495.169

С помощью компиляторов код из первого проекта (выполненного на языке C) был перекомпилирован в NASM, после чего вручную очищен от комментариев, названий секций и прочего. По скорости программа на ассемблере выполняется сравнимо быстрее на больших данных.

Инструментальные средства:

- Виртуальная машина Oracle VM VirtualBox
  - Класс ОС: Linux
  - Версия ОС: Arch Linux (64-bit)
  - Кол-во процессоров виртуальной машины: 2
  - Оперативная память: 4096 МБ
- Языки программирования: C/C++, NASM
- Библиотеки: stdio.h, stdlib.h, time.h, string.h
- Интегрированная среда разработки: CLion, SASM
- Средства сборки проектов: CMake
- Компиляторы: gcc, g++