Курсовая Работа
По дисциплине «Технологии разработки программного обеспечения»
на тему «Программа Алфавит»

Выполнили студенты группы ИП-911:
Казанцев А. С.
Николаев Д. О.

Содержание

Техническое задание

- **Функциональность проекта**

  Программа «Алфавит» читает входной текст и выводит слова из него по одному в строке. Слова следуют друг за другом в алфавитном порядке.
- **Формат входных данных**

  Приложение получает в качестве входных данных текст. Текст может быть передан через консольный ввод либо через текстовый файл.
- **Интерфейс приложения**

  Программа работает в интерактивном режиме. Пользователю предоставляется возможность выбрать способ, с помощью которого будет передан входной текст.
- **Аргументы командной строки**

  Приложение принимает следующие аргументы из командной строки.
  int regime — способ передачи в программу входного текста. Если regime = 1, то программа читает текстовый файл для получения входного текста. Если regime = 2, то пользователь должен ввести текст в консольный ввод. При regime = 0 программа завершается без получения входного текста.
  char* file_text[] — массив строк. В зависимости от способа передачи входного текста в программу данный параметр означает следующее. Если regime = 1 и текст передается через текстовый файл, то file_text является именем файла. Если regime = 2 и текст вводится в консоли, то file_text и является входным текстом.

План работ

До начала реализации проекта был подготовлен план работ. Создание приложения «Алфавит» было разбито на несколько этапов, которые вкратце описаны ниже.

1. Определить способ передачи входного текста

Нужно написать функцию, которая будет узнавать у пользователя, каким способом пользователь собирается передать входной текст в программу.

2. Получить текст через консольный ввод

Нужно написать функцию для чтения текста, введенного пользователем через консольный ввод.

3. Получить текст из текстового файла

Нужно написать функцию для чтения текста из текстового файла.

4. Разбить текст на отдельные слова

Нужно написать функцию для выделения из входного текста отдельных слов и составления из них вектора. В слова не должны входить знаки пунктуации.

5. Отсортировать по алфавиту слова

Нужно написать функцию, которая будет сортировать в порядке возрастания (по алфавиту) слова из вектора.

6. Вывести на экран слова

Нужно написать функцию, которая будет получать в качестве аргумента массив слов и будет выводить в консоли по одному слову в строке.

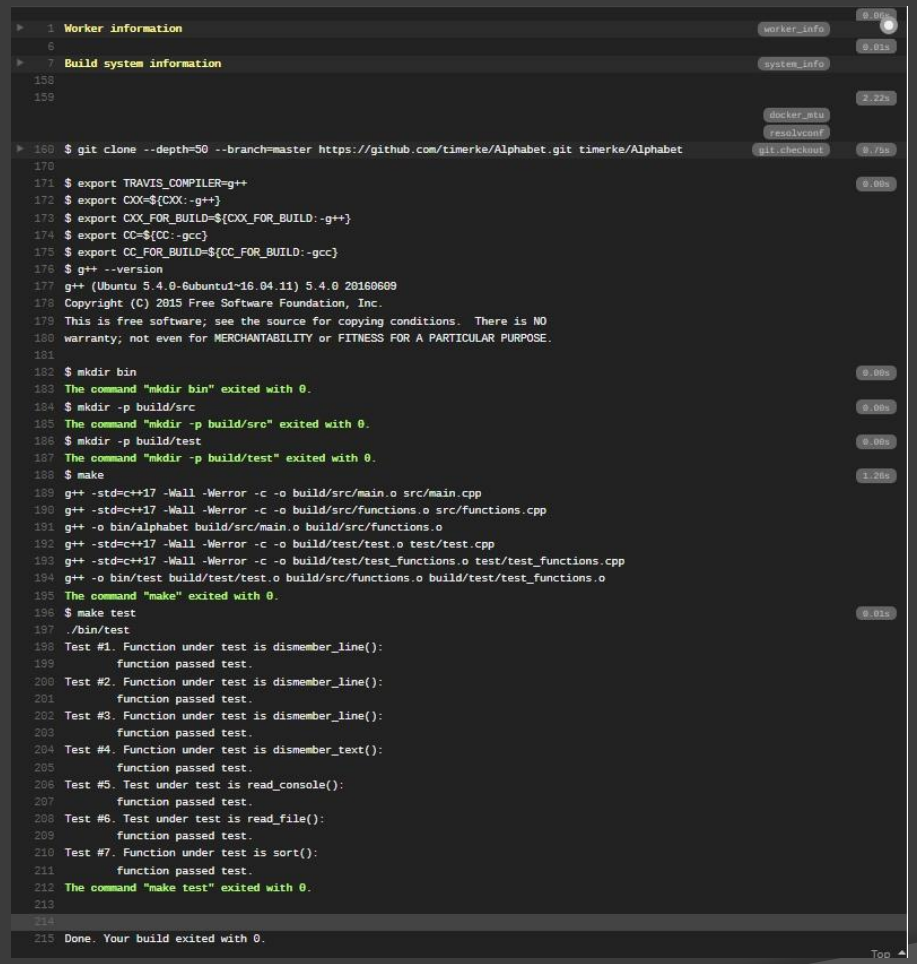
Работа программы

```
Как будет передан текст? (Текстовый файл - 1, консольный ввод - 2, выход - 0.)
        2
Введите текст. Чтобы завершить ввод, введите пустую строку.
Ваш текст:
Create a function that will find out from the user how the user is going to pass the input text to the program

Слова из текста в алфавитном порядке:
        Create
        a
        find
        from
        function
        going
        how
        input
        is
        out
        pass
        program
        text
        that
        the
        the
        the
        the
        to
        to
        user
        user
        will
Завершение программы.

C:\Users\Pizra\source\repos\Project6\Debug\Project6.exe (процесс 9320) завершил работу с кодом 0.
Нажмите любую клавишу, чтобы закрыть это окно…
```

Тест TravisCI



```
Worker information                                                                  worker_info   0.06s
...
Build system information                                                            system_info   0.01s
...
                                                                                    docker_mtu    2.22s
                                                                                    resolvconf
$ git clone --depth=50 --branch=master https://github.com/timerke/Alphabet.git timerke/Alphabet   git.checkout  0.75s

$ export TRAVIS_COMPILER=g++                                                                      0.00s
$ export CXX=${CXX:-g++}
$ export CXX_FOR_BUILD=${CXX_FOR_BUILD:-g++}
$ export CC=${CC:-gcc}
$ export CC_FOR_BUILD=${CC_FOR_BUILD:-gcc}
$ g++ --version
g++ (Ubuntu 5.4.0-6ubuntu1~16.04.11) 5.4.0 20160609
Copyright (C) 2015 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

$ mkdir bin                                                                       0.00s
The command "mkdir bin" exited with 0.
$ mkdir -p build/src                                                              0.00s
The command "mkdir -p build/src" exited with 0.
$ mkdir -p build/test                                                             0.00s
The command "mkdir -p build/test" exited with 0.
$ make                                                                            1.26s
g++ -std=c++17 -Wall -Werror -c -o build/src/main.o src/main.cpp
g++ -std=c++17 -Wall -Werror -c -o build/src/functions.o src/functions.cpp
g++ -o bin/alphabet build/src/main.o build/src/functions.o
g++ -std=c++17 -Wall -Werror -c -o build/test/test.o test/test.cpp
g++ -std=c++17 -Wall -Werror -c -o build/test/test_functions.o test/test_functions.cpp
g++ -o bin/test build/test/test.o build/src/functions.o build/test/test_functions.o
The command "make" exited with 0.
$ make test                                                                       0.01s
./bin/test
Test #1. Function under test is dismember_line():
        function passed test.
Test #2. Function under test is dismember_line():
        function passed test.
Test #3. Function under test is dismember_line():
        function passed test.
Test #4. Function under test is dismember_text():
        function passed test.
Test #5. Test under test is read_console():
        function passed test.
Test #6. Test under test is read_file():
        function passed test.
Test #7. Function under test is sort():
        function passed test.
The command "make test" exited with 0.


Done. Your build exited with 0.
```

Личный вклад в проект

Казанцев Артём

- Реализация функции, которая будет узнавать у пользователя, каким способом пользователь собирается передать входной текст в программу
- Реализация функции для чтения текста, введенного пользователем через консольный ввод.
- Реализация функции, которая будет получать в качестве аргумента массив слов и будет выводить в консоли по одному слову в строке.
- тестирование функций

Николаев Даниил

- Реализация функции для чтения текста из текстового файла
- Реализация функции для выделения из входного текста отдельных слов и составления из них вектора. В слова не должны входить знаки пунктуации.
- Реализация функции для сортировки в порядке возрастания (по алфавиту) слова из вектора.

Приложение. Текст программы

main.cpp

```cpp
/*
Приложение "Алфавит" читает текст и выводит из него по одному
слову в строке по алфавиту.
*/
#include <clocale>
#include <iostream>
#include "functions.h"

int main(int argc, char* argv[])
{
    setlocale(LC_ALL, "Rus");

    // Определяем способ передачи входного текста
    char response = determine_way(argc, argv);

    // Получаем исходный текст
    std::vector<std::string> text;
    if (response == '0')
    {
        // Пользователь хочет выйти из программы
        finish();
        return 0;
    }
    else if (response == '1')
    {
        // Входной текст будет передан через текстовый файл
        if (!read_file(argc, argv, &text))
        {
            finish();
            return 0;
        }
    }
    else if (response == '2')
    {
        // Входной текст будет введен в консоли
        read_console(argc, argv, &text);
    }

    // Получаем вектор со словами из текста
    std::vector<std::string> words;
    dismember_text(&text, &words);

    // Сортируем вектор со словами по алфавиту
    sort(&words);

    // Выводим слова на экран
    std::cout << "Слова из текста в алфавитном порядке:\n";
    print_words(&words);

    // Выход из приложения
    finish();
    return 0;
}
```

functions.cpp

```cpp
/*
Модуль functions.cpp содержит определения функций,
используемых программой "Алфавит".
*/

#include "functions.h"
#include <fstream>
#include <iostream>

/**
 * Функция определяет способ, которым будет передан входной текст.
 * @param argc - количество параметров, передаваемых в командной строке;
 * @param argv - массив из строк-параметров командной строки.
 * @return символ, который показывает способ передачи входного текста
 * в приложение:
 * '0' - текст вообще не передается в приложение;
 * '1' - текст передается через текстовый файл;
 * '2' - текст передается через консольный ввод.
 */
char determine_way(int argc, char* argv[])
{
    std::string response; // параметр для чтения ответа пользователя
    if (argc > 1)
    {
        // Через командную строку передаются аргументы
        response = argv[1];
        if (response == "0" || response == "1" || response == "2")
        {
            // Через командную строку передаются аргументы в правильной форме
            return response[0];
        }
        else
        {
            // Через командную строку передаются аргументы в неправильной форме
            std::cout << "Ошибка. Повторите ввод.\n";
        }
    }
    std::cout << "Как будет передан текст? (Текстовый файл - 1,"
        " консольный ввод - 2, выход - 0.)\n\t";
    std::getline(std::cin, response);
    while (response != "0" && response != "1" && response != "2")
    {
        std::cout << "Ошибка, повторите ввод:\n\t";
        std::getline(std::cin, response);
    }
    return response[0];
}

/**
 * Функция разбивает строку на слова.
 * @param line - строка со словами;
 * @param words - вектор, в который будут записаны слова из строки.
 */
void dismember_line(std::string* line, std::vector<std::string>* words)
{
    // Знаки пунктуации, разбивающие строку на слова
    const char punctuation[] = " \t.,:;-?!'\"()";
    std::size_t begin_pos = line->find_first_not_of(punctuation);
    std::size_t end_pos = line->find_first_of(punctuation, begin_pos + 1);
    while (begin_pos != std::string::npos && end_pos != std::string::npos)
    {
        // Добавляем слово в вектор
        words->push_back(line->substr(begin_pos, end_pos - begin_pos));
        begin_pos = line->find_first_not_of(punctuation, end_pos + 1);
        end_pos = line->find_first_of(punctuation, begin_pos + 1);
    }
    if (begin_pos != std::string::npos && end_pos == std::string::npos)
    {
        // Добавляем слово в конце строки в вектор
        words->push_back(line->substr(begin_pos));
    }
}

/**
 * Функция разбивает входной текст на слова.
 * @param text - текст, представляющий собой вектор из строк;
 * @param words - вектор, в который будут записаны слова из текста.
 */
void dismember_text(std::vector<std::string>* text,
    std::vector<std::string>* words)
{
    for (std::vector<std::string>::iterator line = text->begin();
        line < text->end(); line++)
    {
        dismember_line(&(*line), words);
    }
}

/**
 * Функция выводит сообщение при завершении программы.
 */
void finish()
{
    std::cout << "Завершение программы.\n";
}

/**
```

functions.cpp

```cpp
/**
 * Функция выводит на экран по одному слову на строке из массива слов.
 * @param words - вектор со словами.
 */
void print_words(std::vector<std::string>* words)
{
    for (std::vector<std::string>::iterator word = words->begin();
        word < words->end(); word++)
    {
        std::cout << "\t" << *word << "\n";
    }
}

/**
 * Функция получает текст через консольный ввод.
 * @param argc - количество параметров, передаваемых в командной строке;
 * @param argv - массив из строк-параметров командной строки;
 * @param text - вектор, в который будут записаны строки входного текста.
 */
void read_console(int argc, char* argv[], std::vector<std::string>* text)
{
    if (argc > 2)
    {
        // Через командную строку передается входной текст
        for (int i = 2; i < argc; i++)
        {
            text->push_back(argv[i]);
        }
    }
    else
    {
        // Через командную строку не передается входной текст
        std::cout << "Введите текст. Чтобы завершить ввод, введите пустую строку."
            "\nВаш текст:\n";
        std::string line = " "; // параметр для чтения строки
        while (line.size())
        {
            std::getline(std::cin, line);
            text->push_back(line);
        }
    }
}

/**
 * Функция получает текст через текстовый файл.
 * @param argc - количество параметров, передаваемых в командной строке;
 * @param argv - массив из строк-параметров командной строки;
 * @param text - вектор, в который будут записаны строки входного текста.
 * @return true, если текстовый файл был прочтен, иначе false.
 */
bool read_file(int argc, char* argv[], std::vector<std::string>* text)
{
    std::string file_path; // путь к файлу
    if (argc > 2)
    {
        // Через командную строку передается имя файла
        file_path = argv[2];
    }
    else
    {
        // Через командную строку не передается имя файла
        std::cout << "Введите путь к текстовому файлу: ";
        std::getline(std::cin, file_path);
    }
    // Открываем файл для чтения
    std::ifstream in(file_path.data(), std::ifstream::in);
    if (!in.is_open())
    {
        // Если файл не открыт
        std::cout << "Ошибка. Файл " << file_path << " не открыт.\n";
        return false;
    }
    // Если файл открыт для чтения
    std::string line; // параметр для чтения строки
    while (std::getline(in, line))
    {
        text->push_back(line);
    }
    in.close(); // закрываем файл
    return true;
}
```

functions.cpp

```cpp
/**
 * Функция сортирует вектор из слов по алфавиту. В этой функции
 * используется пузырьковый метод сортировки (метод простым обменом).
 * @param words - вектор со словами.
 */
void sort(std::vector<std::string>* words)
{
    std::vector<std::string>::iterator first = words->begin();
    std::vector<std::string>::iterator last = words->end();
    while (first < --last)
    {
        for (std::vector<std::string>::iterator i = first; i < last; ++i)
        {
            if (*(i + 1) < *i)
            {
                std::iter_swap(i, i + 1);
            }
        }
    }
}
```

functions.h

```cpp
/*
Заголовочный файл functions.h содержит объявления функций,
используемых в программе "Алфавит".
*/

#pragma once
#ifndef FUNCTIONS_H
#define FUNCTIONS_H

#include <string>
#include <vector>

char determine_way(int, char* []);
void dismember_line(std::string*, std::vector<std::string>*);
void dismember_text(std::vector<std::string>*, std::vector<std::string>*);
void finish();
void print_words(std::vector<std::string>*);
void read_console(int, char* [], std::vector<std::string>*);
bool read_file(int, char* [], std::vector<std::string>*);
void sort(std::vector<std::string>*);

#endif
```