

# Курсовая Работа

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

Выполнили студенты группы ИП-911:

Казанцев А. С.

Николаев Д. О.

# Содержание

Введение и постановка задачи - 3

Техническое задание - 3

Функциональность проекта - 3

Формат входных данных - 3

Интерфейс приложения - 3

Аргументы командной строки - 3

План работ - 4

1. Определить способ передачи входного текста - 4

2. Получить текст через консольный ввод - 4

3. Получить текст из текстового файла - 4

4. Разбить текст на отдельные слова - 4

5. Отсортировать по алфавиту слова - 4

6. Вывести на экран слова - 4

Личный вклад в проект - 4

Приложение. Текст программы - 5

# 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
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...
```

# Tect TravisCI

```
1 Worker information worker_info 0.00s
6
7 Build system information system_info 0.01s
158
159
160 $ git clone --depth=50 --branch=master https://github.com/timerke/Alphabet.git timerke/Alphabet git_checkout 6.75s
170
171 $ export TRAVIS_COMPILER=g++ 0.00s
172 $ export CXX=${CXX:-g++}
173 $ export CXX_FOR_BUILD=${CXX_FOR_BUILD:-g++}
174 $ export CC=${CC:-gcc}
175 $ export CC_FOR_BUILD=${CC_FOR_BUILD:-gcc}
176 $ g++ --version
177 g++ (Ubuntu 5.4.0-6ubuntu1-16.04.11) 5.4.0 20160609
178 Copyright (C) 2015 Free Software Foundation, Inc.
179 This is free software; see the source for copying conditions. There is NO
180 warranty, not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
181
182 $ mkdir bin 0.00s
183 The command "mkdir bin" exited with 0.
184 $ mkdir -p build/src 0.00s
185 The command "mkdir -p build/src" exited with 0.
186 $ mkdir -p build/test 0.00s
187 The command "mkdir -p build/test" exited with 0.
188 $ make 1.26s
189 g++ -std=c++17 -Wall -Werror -c -o build/src/main.o src/main.cpp
190 g++ -std=c++17 -Wall -Werror -c -o build/src/functions.o src/functions.cpp
191 g++ -o bin/alphabet build/src/main.o build/src/functions.o
192 g++ -std=c++17 -Wall -Werror -c -o build/test/test.o test/test.cpp
193 g++ -std=c++17 -Wall -Werror -c -o build/test/test_functions.o test/test_functions.cpp
194 g++ -o bin/test build/test/test.o build/src/functions.o build/test/test_functions.o
195 The command "make" exited with 0. 0.01s
196 $ make test
197 ./bin/test
198 Test #1. Function under test is dismember_line():
199     function passed test.
200 Test #2. Function under test is dismember_line():
201     function passed test.
202 Test #3. Function under test is dismember_line():
203     function passed test.
204 Test #4. Function under test is dismember_text():
205     function passed test.
206 Test #5. Test under test is read_console():
207     function passed test.
208 Test #6. Test under test is read_file():
209     function passed test.
210 Test #7. Function under test is sort():
211     function passed test.
212 The command "make test" exited with 0.
213
214
215 Done. Your build exited with 0. To
```

# 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

```
1  /*
2  Приложение "Алфавит" читает текст и выводит из него по одному
3  слову в строке по алфавиту.
4  */
5  #include <locale>
6  #include <iostream>
7  #include "functions.h"
8
9  int main(int argc, char* argv[])
10 {
11     setlocale(LC_ALL, "Rus");
12
13     // Определяем способ передачи входного текста
14     char response = determine_way(argc, argv);
15
16     // Получаем исходный текст
17     std::vector<std::string> text;
18     if (response == '0')
19     {
20         // Пользователь хочет выйти из программы
21         finish();
22         return 0;
23     }
24     else if (response == '1')
25     {
26         // Входной текст будет передан через текстовый файл
27         if (!read_file(argc, argv, &text))
28         {
29             finish();
30             return 0;
31         }
32     }
33     else if (response == '2')
34     {
35         // Входной текст будет введен в консоли
36         read_console(argc, argv, &text);
37     }
38
39     // Получаем вектор со словами из текста
40     std::vector<std::string> words;
41     dismember_text(&text, &words);
42
43     // Сортируем вектор со словами по алфавиту
44     sort(&words);
45
46     // Выводим слова на экран
47     std::cout << "Слова из текста в алфавитном порядке:\n";
48     print_words(&words);
49
50     // Выход из приложения
51     finish();
52     return 0;
53 }
```

# functions.cpp

```
1  /*
2  Модуль functions.cpp содержит определения функций,
3  используемых программой "Алфавит".
4  */
5
6  #include "functions.h"
7  #include <fstream>
8  #include <iostream>
9
10 /**
11  * Функция определяет способ, которым будет передан входной текст.
12  * @param argc - количество параметров, передаваемых в командной строке;
13  * @param argv - массив из строк-параметров командной строки.
14  * @return символ, который показывает способ передачи входного текста
15  * в приложение:
16  * '0' - текст вообще не передается в приложение;
17  * '1' - текст передается через текстовый файл;
18  * '2' - текст передается через консольный ввод.
19  */
20 char determine_way(int argc, char* argv[])
21 {
22     std::string response; // параметр для чтения ответа пользователя
23     if (argc > 1)
24     {
25         // Через командную строку передаются аргументы
26         response = argv[1];
27         if (response == "0" || response == "1" || response == "2")
28         {
29             // Через командную строку передаются аргументы в правильной форме
30             return response[0];
31         }
32         else
33         {
34             // Через командную строку передаются аргументы в неправильной форме
35             std::cout << "Ошибка. Повторите ввод.\n";
36         }
37     }
38     std::cout << "Как будет передан текст? (Текстовый файл - 1,"
39     " консольный ввод - 2, выход - 0.)\n\t";
40     std::getline(std::cin, response);
41     while (response != "0" && response != "1" && response != "2")
42     {
43         std::cout << "Ошибка, повторите ввод:\n\t";
44         std::getline(std::cin, response);
45     }
46     return response[0];
47 }
48
49 /**
```

```
49 /**
50  * Функция разбивает строку на слова.
51  * @param line - строка со словами;
52  * @param words - вектор, в который будут записаны слова из строки.
53  */
54 void dismember_line(std::string* line, std::vector<std::string>* words)
55 {
56     // Знаки пунктуации, разбивающие строку на слова
57     const char punctuation[] = " \t.,;:-?!'\"()";
58     std::size_t begin_pos = line->find_first_not_of(punctuation);
59     std::size_t end_pos = line->find_first_of(punctuation, begin_pos + 1);
60     while (begin_pos != std::string::npos && end_pos != std::string::npos)
61     {
62         // Добавляем слово в вектор
63         words->push_back(line->substr(begin_pos, end_pos - begin_pos));
64         begin_pos = line->find_first_not_of(punctuation, end_pos + 1);
65         end_pos = line->find_first_of(punctuation, begin_pos + 1);
66     }
67     if (begin_pos != std::string::npos && end_pos == std::string::npos)
68     {
69         // Добавляем слово в конце строки в вектор
70         words->push_back(line->substr(begin_pos));
71     }
72 }
73
74 /**
75  * Функция разбивает входной текст на слова.
76  * @param text - текст, представляющий собой вектор из строк;
77  * @param words - вектор, в который будут записаны слова из текста.
78  */
79 void dismember_text(std::vector<std::string>* text,
80                     std::vector<std::string>* words)
81 {
82     for (std::vector<std::string>::iterator line = text->begin();
83          line < text->end(); line++)
84     {
85         dismember_line(&(*line), words);
86     }
87 }
88
89 /**
90  * Функция выводит сообщение при завершении программы.
91  */
92 void finish()
93 {
94     std::cout << "Завершение программы.\n";
95 }
96
97 /**
```

# functions.cpp

```
97  /**
98  * Функция выводит на экран по одному слову на строке из массива слов.
99  * @param words - вектор со словами.
100 */
101 void print_words(std::vector<std::string>* words)
102 {
103     for (std::vector<std::string>::iterator word = words->begin();
104          word < words->end(); word++)
105     {
106         std::cout << "\t" << *word << "\n";
107     }
108 }
109
110 /**
111 * Функция получает текст через консольный ввод.
112 * @param argc - количество параметров, передаваемых в командной строке;
113 * @param argv - массив из строк-параметров командной строки;
114 * @param text - вектор, в который будут записаны строки входного текста.
115 */
116 void read_console(int argc, char* argv[], std::vector<std::string>* text)
117 {
118     if (argc > 2)
119     {
120         // Через командную строку передается входной текст
121         for (int i = 2; i < argc; i++)
122         {
123             text->push_back(argv[i]);
124         }
125     }
126     else
127     {
128         // Через командную строку не передается входной текст
129         std::cout << "Введите текст. Чтобы завершить ввод, введите пустую строку."
130             << "\nВаш текст:\n";
131         std::string line = " "; // параметр для чтения строки
132         while (line.size())
133         {
134             std::getline(std::cin, line);
135             text->push_back(line);
136         }
137     }
138 }
139
140 /**
```

```
140 /**
141 * Функция получает текст через текстовый файл.
142 * @param argc - количество параметров, передаваемых в командной строке;
143 * @param argv - массив из строк-параметров командной строки;
144 * @param text - вектор, в который будут записаны строки входного текста.
145 * @return true, если текстовый файл был прочтен, иначе false.
146 */
147 bool read_file(int argc, char* argv[], std::vector<std::string>* text)
148 {
149     std::string file_path; // путь к файлу
150     if (argc > 2)
151     {
152         // Через командную строку передается имя файла
153         file_path = argv[2];
154     }
155     else
156     {
157         // Через командную строку не передается имя файла
158         std::cout << "Введите путь к текстовому файлу: ";
159         std::getline(std::cin, file_path);
160     }
161     // Открываем файл для чтения
162     std::ifstream in(file_path.data(), std::ifstream::in);
163     if (!in.is_open())
164     {
165         // Если файл не открыт
166         std::cout << "Ошибка. Файл " << file_path << " не открыт.\n";
167         return false;
168     }
169     // Если файл открыт для чтения
170     std::string line; // параметр для чтения строки
171     while (std::getline(in, line))
172     {
173         text->push_back(line);
174     }
175     in.close(); // закрываем файл
176     return true;
177 }
178
```

## functions.cpp

```
179  /**
180   * Функция сортирует вектор из слов по алфавиту. В этой функции
181   * используется пузырьковый метод сортировки (метод простым обменом).
182   * @param words - вектор со словами.
183   */
184  void sort(std::vector<std::string>* words)
185  {
186      std::vector<std::string>::iterator first = words->begin();
187      std::vector<std::string>::iterator last = words->end();
188      while (first < --last)
189      {
190          for (std::vector<std::string>::iterator i = first; i < last; ++i)
191          {
192              if (*(i + 1) < *i)
193              {
194                  std::iter_swap(i, i + 1);
195              }
196          }
197      }
198  }
```

## functions.h

```
1  /*
2  Заголовочный файл functions.h содержит объявления функций,
3  используемых в программе "Алфавит".
4  */
5
6  #pragma once
7  #ifndef FUNCTIONS_H
8  #define FUNCTIONS_H
9
10 #include <string>
11 #include <vector>
12
13 char determine_way(int, char* []);
14 void dismember_line(std::string*, std::vector<std::string>*);
15 void dismember_text(std::vector<std::string>*, std::vector<std::string>*);
16 void finish();
17 void print_words(std::vector<std::string>*);
18 void read_console(int, char* [], std::vector<std::string>*);
19 bool read_file(int, char* [], std::vector<std::string>*);
20 void sort(std::vector<std::string>*);
21
22 #endif
```