

Лабораторная работа №5

Создано системой Doxygen 1.9.6

1	Список файлов	1
1.1	Файлы	1
2	Файлы	3
2.1	Файл GoogleTest/test.cpp	3
2.1.1	Подробное описание	3
2.1.2	Функции	3
2.1.2.1	main()	4
2.1.2.2	TEST() [1/11]	4
2.1.2.3	TEST() [2/11]	4
2.1.2.4	TEST() [3/11]	5
2.1.2.5	TEST() [4/11]	5
2.1.2.6	TEST() [5/11]	5
2.1.2.7	TEST() [6/11]	6
2.1.2.8	TEST() [7/11]	6
2.1.2.9	TEST() [8/11]	6
2.1.2.10	TEST() [9/11]	7
2.1.2.11	TEST() [10/11]	7
2.1.2.12	TEST() [11/11]	8
2.2	Файл lib.h	8
2.2.1	Подробное описание	8
2.2.2	Функции	8
2.2.2.1	check_args()	8
2.2.2.2	sort()	9
2.3	lib.h	9
2.4	Файл main.cpp	10
2.4.1	Подробное описание	10
2.4.2	Функции	10
2.4.2.1	main()	10
	Предметный указатель	13

Глава 1

Список файлов

1.1 Файлы

Полный список документированных файлов.

lib.h	Заголовочный файл с функциями, написанными для решения лабораторной работы	8
main.cpp	непосредственно решение лабораторной работы	10
GoogleTest/ test.cpp	Тестирование работы функций	3

Глава 2

Файлы

2.1 Файл GoogleTest/test.cpp

Тестирование работы функций.

```
#include <gtest/gtest.h>
#include "/mnt/d/CLionProjects/laba-3/lib.h"
```

Функции

- `TEST` (CheckArgs, Test1)
- `TEST` (CheckArgs, Test2)
- `TEST` (CheckArgs, Test3)
- `TEST` (CheckArgs, Test4)
- `TEST` (CheckArgs, Test5)
- `TEST` (CheckArgs, Test6)
- `TEST` (CheckArgs, Test7)
- `TEST` (CheckArgs, Test8)
- `TEST` (CheckArgs, Test9)
- `TEST` (CheckSort, Test1)
- `TEST` (CheckSort, Test2)
- `int main` (int argc, char **argv)

2.1.1 Подробное описание

Тестирование работы функций.

2.1.2 Функции

2.1.2.1 main()

```

int main (
    int argc,
    char ** argv )
244 {
245     ::testing::InitGoogleTest(&argc, argv);
246     return RUN_ALL_TESTS();
247 }

```

2.1.2.2 TEST() [1/11]

```

TEST (
    CheckArgs ,
    Test1 )

```

Проверка check_args на правильность возвращаемого выражения при одном аргументе.

```

12     {
13         int len = 1;
14         char** a = new char*[1];
15         a[0] = new char[6];
16         a[0][0] = 'l'; a[0][1] = '/'; a[0][2] = 'm'; a[0][3] = 'a'; a[0][4] = 'i'; a[0][5] = 'n';
17         EXPECT_EQ(0, check_args(len, a));
18         delete []a;
19     }

```

2.1.2.3 TEST() [2/11]

```

TEST (
    CheckArgs ,
    Test2 )

```

Проверка check_args на правильность возвращаемого выражения при двух аргументах, один из которых –fromfile.

```

24     {
25         int len = 2;
26         char** a = new char*[2];
27         a[0] = new char[6];
28         a[0][0] = 'l'; a[0][1] = '/'; a[0][2] = 'm'; a[0][3] = 'a'; a[0][4] = 'i'; a[0][5] = 'n';
29         a[1] = new char[10];
30         a[1][0] = 'l'; a[1][1] = 'l'; a[1][2] = 'f'; a[1][3] = 'r'; a[1][4] = 'o'; a[1][5] = 'm'; a[1][6] = 'f';
31         a[1][7] = 'i'; a[1][8] = 'l'; a[1][9] = 'e';
32         EXPECT_EQ(1, check_args(len, a));
33         delete []a[0]; delete []a[1];
34         delete []a;
35     }

```


2.1.2.4 TEST() [3/11]

```
TEST (
    CheckArgs ,
    Test3 )
```

Проверка `check_args` на правильность возвращаемого выражения при двух аргументах, один из которых `--tofile`.

```
41     {
42         int len = 2;
43         char** a = new char*[2];
44         a[0] = new char[6];
45         a[0][0] = '.'; a[0][1] = '/'; a[0][2] = 'm'; a[0][3] = 'a'; a[0][4] = 'i'; a[0][5] = 'n';
46         a[1] = new char[8];
47         a[1][0] = '-'; a[1][1] = '-'; a[1][2] = 't'; a[1][3] = 'o'; a[1][4] = 'f';
48         a[1][5] = 'i'; a[1][6] = 'l'; a[1][7] = 'e';
49         EXPECT_EQ(2, check_args(len, a));
50         delete []a[0]; delete []a[1];
51         delete []a;
52     }
```

2.1.2.5 TEST() [4/11]

```
TEST (
    CheckArgs ,
    Test4 )
```

Проверка `check_args` на правильность возвращаемого выражения при двух аргументах, второй из которых неверный.

```
57     {
58         int len = 2;
59         char** a = new char*[2];
60         a[0] = new char[6];
61         a[0][0] = '.'; a[0][1] = '/'; a[0][2] = 'm'; a[0][3] = 'a'; a[0][4] = 'i'; a[0][5] = 'n';
62         a[1] = new char[8];
63         a[1][0] = '+'; a[1][1] = '+'; a[1][2] = 't'; a[1][3] = 'o'; a[1][4] = 'f';
64         a[1][5] = 'i'; a[1][6] = 'l'; a[1][7] = 'e';
65         EXPECT_EQ(-1, check_args(len, a));
66         delete []a[0]; delete []a[1];
67         delete []a;
68     }
```

2.1.2.6 TEST() [5/11]

```
TEST (
    CheckArgs ,
    Test5 )
```

Проверка `check_args` на правильность возвращаемого выражения при трех аргументах, второй: `--tofile`, третий: `--fromfile`.

```
74     {
75         int len = 3;
76         char** a = new char*[3];
77         a[0] = new char[6];
78         a[0][0] = '.'; a[0][1] = '/'; a[0][2] = 'm'; a[0][3] = 'a'; a[0][4] = 'i'; a[0][5] = 'n';
79         a[1] = new char[8];
80         a[1][0] = '-'; a[1][1] = '-'; a[1][2] = 't'; a[1][3] = 'o'; a[1][4] = 'f';
81         a[1][5] = 'i'; a[1][6] = 'l'; a[1][7] = 'e';
82         a[2] = new char[10];
83         a[2][0] = '-'; a[2][1] = '-'; a[2][2] = 'f'; a[2][3] = 'r'; a[2][4] = 'o'; a[2][5] = 'm'; a[2][6] = 'f';
84         a[2][7] = 'i'; a[2][8] = 'l'; a[2][9] = 'e';
85         EXPECT_EQ(3, check_args(len, a));
86         delete []a[0]; delete []a[1]; delete []a[2];
87         delete []a;
88     }
```

2.1.2.7 TEST() [6/11]

```
TEST (
    CheckArgs ,
    Test6 )
```

Проверка check_args на правильность возвращаемого выражения при трех аргументах, второй: "--fromfile", третий: "--tofile".

```
94     {
95         int len = 3;
96         char** a = new char*[3];
97         a[0] = new char[6];
98         a[0][0] = '.'; a[0][1] = '/'; a[0][2] = 'm'; a[0][3] = 'a'; a[0][4] = 'i'; a[0][5] = 'n';
99         a[1] = new char[10];
100        a[1][0] = '-'; a[1][1] = '-'; a[1][2] = 'f'; a[1][3] = 'r'; a[1][4] = 'o'; a[1][5] = 'm'; a[1][6] = 'f';
101        a[1][7] = 'i'; a[1][8] = 'l'; a[1][9] = 'e';
102        a[2] = new char[8];
103        a[2][0] = '-'; a[2][1] = '-'; a[2][2] = 't'; a[2][3] = 'o'; a[2][4] = 'f';
104        a[2][5] = 'i'; a[2][6] = 'l'; a[2][7] = 'e';
105        EXPECT_EQ(3, check_args(len, a));
106        delete []a[0]; delete []a[1]; delete []a[2];
107        delete []a;
108    }
```

2.1.2.8 TEST() [7/11]

```
TEST (
    CheckArgs ,
    Test7 )
```

Проверка check_args на правильность возвращаемого выражения при трех аргументах, третий неверный.

```
114     {
115         int len = 3;
116         char** a = new char*[3];
117         a[0] = new char[6];
118         a[0][0] = '.'; a[0][1] = '/'; a[0][2] = 'm'; a[0][3] = 'a'; a[0][4] = 'i'; a[0][5] = 'n';
119         a[1] = new char[10];
120        a[1][0] = '-'; a[1][1] = '-'; a[1][2] = 'f'; a[1][3] = 'r'; a[1][4] = 'o'; a[1][5] = 'm'; a[1][6] = 'f';
121        a[1][7] = 'i'; a[1][8] = 'l'; a[1][9] = 'e';
122        a[2] = new char[8];
123        a[2][0] = '+'; a[2][1] = '+'; a[2][2] = 't'; a[2][3] = 'o'; a[2][4] = 'f';
124        a[2][5] = 'i'; a[2][6] = 'l'; a[2][7] = 'e';
125        EXPECT_EQ(-1, check_args(len, a));
126        delete []a[0]; delete []a[1]; delete []a[2];
127        delete []a;
128    }
```

2.1.2.9 TEST() [8/11]

```
TEST (
    CheckArgs ,
    Test8 )
```

Проверка check_args на правильность возвращаемого выражения при трех аргументах, второй и третий неверный.

```
134     {
135         int len = 3;
136         char** a = new char*[3];
137         a[0] = new char[6];
138         a[0][0] = '.'; a[0][1] = '/'; a[0][2] = 'm'; a[0][3] = 'a'; a[0][4] = 'i'; a[0][5] = 'n';
139         a[1] = new char[10];
140        a[1][0] = '+'; a[1][1] = '+'; a[1][2] = 'f'; a[1][3] = 'r'; a[1][4] = 'o'; a[1][5] = 'm'; a[1][6] = 'f';
```

```

141 a[1][7] = 'i'; a[1][8] = 'l'; a[1][9] = 'e';
142 a[2] = new char[8];
143 a[2][0] = '+'; a[2][1] = '+'; a[2][2] = 't'; a[2][3] = 'o'; a[2][4] = 'f';
144 a[2][5] = 'i'; a[2][6] = 'l'; a[2][7] = 'e';
145 EXPECT_EQ(-1, check_args(len, a));
146 delete []a[0]; delete []a[1]; delete []a[2];
147 delete []a;
148 }

```

2.1.2.10 TEST() [9/11]

```

TEST (
    CheckArgs ,
    Test9 )

```

Проверка check_args на правильность возвращаемого выражения при четырех аргументах.

```

154 {
155     int len = 4;
156     char** a = new char*[4];
157     a[0] = new char[6];
158     a[0][0] = 'i'; a[0][1] = '/'; a[0][2] = 'm'; a[0][3] = 'a'; a[0][4] = 'i'; a[0][5] = 'n';
159     a[1] = new char[10];
160     a[1][0] = '-'; a[1][1] = '-'; a[1][2] = 'f'; a[1][3] = 'r'; a[1][4] = 'o'; a[1][5] = 'm'; a[1][6] = 'f';
161     a[1][7] = 'i'; a[1][8] = 'l'; a[1][9] = 'e';
162     a[2] = new char[8];
163     a[2][0] = '-'; a[2][1] = '-'; a[2][2] = 't'; a[2][3] = 'o'; a[2][4] = 'f';
164     a[2][5] = 'i'; a[2][6] = 'l'; a[2][7] = 'e';
165     a[3] = new char[4];
166     a[3][0] = 'b'; a[3][1] = 'o'; a[3][2] = 'o'; a[3][3] = 'm';
167     EXPECT_EQ(-1, check_args(len, a));
168     delete []a[0]; delete []a[1]; delete []a[2]; delete []a[3];
169     delete []a;
170 }

```

2.1.2.11 TEST() [10/11]

```

TEST (
    CheckSort ,
    Test1 )

```

Проверка сортировки двумерного массива, при длине вложенных массивов 1.

```

176 {
177     int len = 5;
178     int** a = new int*[5];
179     a[0] = new int[1]; a[0][0] = 5;
180     a[1] = new int[1]; a[1][0] = 4;
181     a[2] = new int[1]; a[2][0] = 3;
182     a[3] = new int[1]; a[3][0] = 2;
183     a[4] = new int[1]; a[4][0] = 1;
184
185     int** ans = new int*[5];
186     ans[0] = new int[1]; ans[0][0] = 1;
187     ans[1] = new int[1]; ans[1][0] = 2;
188     ans[2] = new int[1]; ans[2][0] = 3;
189     ans[3] = new int[1]; ans[3][0] = 4;
190     ans[4] = new int[1]; ans[4][0] = 5;
191
192     sort(a, len);
193     for(int i = 0; i < 5; ++i){
194         EXPECT_EQ(ans[i][0], a[i][0]);
195     }
196     for(int i = 0; i < 5; ++i){
197         delete []ans[i];
198         delete []a[i];
199     }
200     delete []ans;
201     delete []a;
202 }

```

2.1.2.12 TEST() [11/11]

```
TEST (
    CheckSort ,
    Test2 )
```

Проверка сортировки двумерного массива, при длине вложенных массивов 3.

```
208     {
209     int len = 10;
210     int** a = new int*[10];
211     int** answ = new int*[10];
212     for(int i = 0; i < 10; ++i){
213         a[i] = new int[3];
214         answ[i] = new int[3];
215         for(int j = 0; j < 3; ++j){
216             a[i][j] = 1;
217             answ[i][j] = 1;
218         }
219     }
220     a[0][0] = -10; a[1][0] = -20; a[2][0] = 0; a[3][0] = 100; a[4][0] = 1;
221     a[5][0] = 5; a[6][0] = 2; a[7][0] = 3; a[8][0] = 4; a[9][0] = 6;
222     answ[0][0] = -20; answ[1][0] = -10; answ[2][0] = 0; answ[3][0] = 1; answ[4][0] = 2;
223     answ[5][0] = 3; answ[6][0] = 4; answ[7][0] = 5; answ[8][0] = 6; answ[9][0] = 100;
224
225     sort(a, len);
226
227     for(int i = 0; i < 10; ++i){
228         for(int j = 0; j < 3; ++j){
229             EXPECT_EQ(answ[i][j], a[i][j]);
230         }
231     }
232
233     for(int i = 0; i < 10; ++i){
234         delete []answ[i];
235         delete []a[i];
236     }
237     delete []answ;
238     delete []a;
239
240
241 }
```

2.2 Файл lib.h

Заголовочный файл с функциями, написанными для решения лабораторной работы

Функции

- int `check_args` (int, char **)
- void `sort` (int **, int)

2.2.1 Подробное описание

Заголовочный файл с функциями, написанными для решения лабораторной работы

2.2.2 Функции

2.2.2.1 check_args()

```
int check_args (
    int argc,
    char ** argv )
```

Проверка вводимых флагов, вводимых в консоле при запуске программы, на правильность

Аргументы

argc	Количество полученных аргументов
argv	Сами полученные аргументы

Возвращает

При отсутствии дополнительных флагов вернет 0, если есть "--fromfile" - 1, "--togile" - 2, если оба - 2, при введении большего количества флагов либо наличие неверных вернет -1.

```

11         {
12     if(argc == 1){
13         return 0;
14     }
15     if(argc == 2){
16         if(!strcmp(argv[1], "--fromfile")){
17             return 1;
18         }
19         if(!strcmp(argv[1], "--tofile")){
20             return 2;
21         }
22     }
23     if(argc == 3){
24         if(!strcmp(argv[1], "--tofile") && !strcmp(argv[2], "--fromfile") ||
25             !strcmp(argv[2], "--tofile") && !strcmp(argv[1], "--fromfile")){
26             return 3;
27         }
28     }
29     return -1;
30 }
```

2.2.2.2 sort()

```

void sort (
    int ** array,
    int len )
```

Функция сортирует двумерный массив по первому элементу вложенного массива

Аргументы

array	ссылка на массив, который нужно отсортировать, поскольку сортируется изначальный массив, то функция void
len	длина сортируемого массива (имеется в виду его первого уровня, длины вложенных массивов не важны)

```

38     {
39     for(int i = 0; i < len; ++i){
40         for(int j = 0; j < len - 1; ++j){
41             if(array[j][0] > array[j + 1][0]){
42                 std::swap(array[j], array[j + 1]);
43             }
44         }
45     }
46 }
47 }
```

2.3 lib.h

[См. документацию.](#)

```

1
5 #ifndef LABA_3_LIB_H
6 #define LABA_3_LIB_H
7
15 int check_args(int, char**);
16
22 void sort(int**, int);
23
24 #endif //LABA_3_LIB_H

```

2.4 Файл main.cpp

непосредственно решение лабораторной работы

```

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

```

Функции

- int main (int argc, char **argv)

2.4.1 Подробное описание

непосредственно решение лабораторной работы

2.4.2 Функции

2.4.2.1 main()

```

int main (
    int argc,
    char ** argv )

```

Основной алгоритм заключается в том, что при считывании текста создается двумерный массив dots типа int, в каждом элементе которого хранится длина одного предложения, номера его первого и последнего элемента. Затем этот массив сортируется по длине предложений с помощью функции sort, после чего на его основе формируется вывод.

Решение задачи на 9. Задача решается на основе уже отсортированного массива dots. Цикл ищет пробел и знаки препинания, по которым можно выделить слова в предложении, с конца (то есть справа налево) самого длинного предложения, границы которого указаны в последнем элементе dots. При нахождении одного из разделительных символов слово выводится слева направо.

Аргументы

argc	количество флагов при вызове функции из консоли
argv	массив этих флагов

Возвращает

при выполнении без ошибок вернет 0, при ошибке ввода -1.

```

27     {
28     int args = check_args(argc, argv);
29     if(args == -1){
30         std::cerr << "Wrong arguments.";
31         return -1;
32     }
33
34     noskipws(std::cin);
35
36     //считывание названий файлов при наличие флагов
37     char* input_file = new char[50];
38     if(args == 1 || args == 3){
39         std::cout << "Please, type a filename to read from.\n";
40         int i = 0;
41         while(std::cin > input_file[i]){
42             ++i;
43             if(std::cin.peek() == '\n'){
44                 break;
45             }
46         }
47     }
48
49     char* output_file = new char[50];
50     if(args == 2 || args == 3){
51         std::cout << "Please, type a filename to write in.\n";
52         int i = 0;
53         while(std::cin > output_file[i]){
54             if(output_file[i] == '\n'){
55                 continue;
56             }
57             ++i;
58             if(std::cin.peek() == '\n'){
59                 break;
60             }
61         }
62     }
63
64
65     //считывание предложений
66     char* str = new char[1000];
67     int** dots = new int*[200];
68     int i = 0, dots_count = 0;
69     int pre_start = 0;
70     if(args == 1 || args == 3){
71         std::ifstream input(input_file);
72         if(!input.is_open()){
73             std::cerr << "Input file doesn't exist.\n";
74             return -1;
75         }
76         noskipws(input);
77         while(input > str[i]){
78             if(str[i] == '\n'){
79                 str[i] = ' ';
80             }
81             if(str[i] == '.'){
82                 dots[dots_count] = new int[3];
83                 while(str[pre_start] == ' '){++pre_start;}
84                 dots[dots_count][0] = i - pre_start + 1;
85                 dots[dots_count][1] = pre_start; dots[dots_count][2] = i;
86                 pre_start = i + 1;
87                 ++dots_count;
88             }
89             ++i;
90         }
91     }
92     input.close();
93 }
94 else{
95     std::cout << "Please, type an expression. Use ctrl + D at the end of it.\n";
96     while(std::cin > str[i]){
97         if(str[i] == '\n'){
98             str[i] = ' ';
99         }
100         if(str[i] == '.'){
101             dots[dots_count] = new int[3];
102             while(str[pre_start] == ' '){++pre_start;}
103             dots[dots_count][0] = i - pre_start + 1;
104             dots[dots_count][1] = pre_start; dots[dots_count][2] = i;
105             pre_start = i + 1;
106             ++dots_count;
107         }
108         ++i;
109     }

```

```

110 }
111
112 //учет неправильного ввода, то есть если предложений нет
113 if(dots_count == 0){
114     std::cerr << "Nothing found.";
115     return -1;
116 }
117
118 //сортировка и вывод
119 sort(dots, dots_count);
120
121 if(args == 2 || args == 3){
122     std::ofstream output(output_file);
123     for (int j = 0; j < dots_count; ++j) {
124         for (int jj = dots[j][1]; jj <= dots[j][2]; ++jj) {
125             output << str[jj];
126         }
127         output << "\n";
128     }
129     output.close();
130
131 } else {
132     std::cout << "\nAnswer:\n";
133     for (int j = 0; j < dots_count; ++j) {
134         for (int jj = dots[j][1]; jj <= dots[j][2]; ++jj) {
135             std::cout << str[jj];
136         }
137         std::cout << "\n";
138     }
139 }
140
141 //доп задание на 9
142 int end = dots[dots_count - 1][2];
143 std::ofstream nine_out("NINE.txt");
144 for(int j = dots[dots_count - 1][2]; j >= dots[dots_count - 1][1]; --j){
145     if(str[j] == ' ' || str[j] == ',' || str[j] == ':' || str[j] == ';'){
146         for(int jj = j + 1; jj < end; ++jj){
147             nine_out << str[jj];
148         }
149         if(end - (j + 1) >= 1) {nine_out << ' ';}
150         end = j;
151     }
152     if(j == dots[dots_count - 1][1]){
153         for(int jj = j; jj < end; ++jj){
154             nine_out << str[jj];
155         }
156     }
157 }
158 delete []str;
159 delete []input_file; delete []output_file;
160 for(int j = 0; j < dots_count; ++j){
161     delete []dots[j];
162 }
163 delete []dots;
164 return 0;
165 }

```


Предметный указатель

check_args
lib.h, [8](#)

GoogleTest/test.cpp, [3](#)

lib.h, [8](#)
check_args, [8](#)
sort, [9](#)

main
main.cpp, [10](#)
test.cpp, [3](#)
main.cpp, [10](#)
main, [10](#)

sort
lib.h, [9](#)

TEST
test.cpp, [4–7](#)

test.cpp
main, [3](#)
TEST, [4–7](#)