

Программирование

М.В.Булгакова

24 декабря 2015 г.

Глава 1

Основные конструкции языка

1.1 Задание 1

1.1.1 Задание

Пользователь задает три корня кубического уравнения

$$x^3 + bx^2 + cx + d \quad (1.1)$$

(например, 1, 2, 3). Вывести значения b, c и d, например: b=-6, c=11, d=-6.

1.1.2 Теоритические сведения

С помощью `main.c`, находящейся в многомодульном проекте `subproject`, можно задать параметр запуска для автоматического выполнения `coefficients_of_equations.c`, находящейся в статической библиотеке `lib`, в параметрах нужно указать `--is-coefficients_of_equation` и через пробелы 3 значения, равные корням кубического уравнения. Так же при задании значения параметра запуска в виде `--interactive` включается интерактивный режим, где данная функция принимает значения, вводимые пользователем программы, выбор выполнения данной задачи описан в `main_menu.c`, где использовались операторы условного перехода `switch`. Ввод и вывод данных в пользовательском режиме происходит в подпроекте `app` в `coefficients_of_equation.c`, заголовочным файлом которого является `coefficients_of_equation.h`.

Были созданы модульные тесты в подпроекте `test`.

1.1.3 Проектирование

В `main.c` у пользователя запрашивают режим работы программы, состоящий из:

1. `--interactive` - Ручной ввод значений
2. `--is-coefficients_of_equation`- Автоматическая работа, через ввод параметров запуска

В `coefficients_of_equation.c`, находящейся в подпроекте `app`, реализовано взаимодействие с пользователем, считывая введенные значения с консоли, и передавая их в `coefficients_of_equations.c`, находящейся в статической библиотеке `lib`, где производится поиск коэффициентов уравнений.

Во время автоматической работы в `coefficients_of_equations.c` значения передаются из параметров запуска.

Модульные тесты находятся в `test tst_testtest.cpp`.

Листинги `main.c` и `main_menu.c` приведены в приложении

1.1.4 Описание тестового стенда и методики тестирования

Среда разработки QtCreator 3.5.0, компилятор GCC 4.8.4 (x86 64 bit), операционная система Linux Mint 17.2 Cinnamon 64 bit. В процессе выполнения задания производилось ручное тестирование. Модульное тестирование реализовано при помощи фреймворка QtTest.

1.1.5 Тестовый план и результаты тестирования

При вызове автоматического теста программа обращается к методу класса `TestTest`

```
test_i_coefficient_of_equation_function(),
test_j_coefficient_of_equation_function(),
test_k_coefficient_of_equation_function(),
```

 в которой по уже заданным значениям производится поиск коэффициентов и сравнение результатов с помощью процедуры `QCOMPARE`. Для модульного теста, в котором

$$x_1 = x_2 = x_3 = 2, \quad (1.2)$$

, все три коэффициента равны 10. Тест прошел успешно. Для ручного теста, в котором

$$x_1 = 3; x_2 = 2; x_3 = 1, i = -6; j = 11; k = -6. \quad (1.3)$$

Тест прошел успешно. Листинги модульных тестов приведены в приложении.

1.1.6 Выводы

При написании данной работы были приобретены навыки работы с отладкой(debug), навыки создания модульных тестов и умение разбивать задачи на подзадачи, отделяя общение с пользователем от бизнес-логики, и создание многомодульных проектов.

Листинги

```
1 #include <stdio.h>
2 #include "search_coefficients_of_equation_function.h"
3
4 void coefficients_of_equation(){
5     puts("Введите 3 значения x через Enter");
6     int x1, x2, x3;
7     scanf("%d", &x1);
8     scanf("%d", &x2);
9     scanf("%d", &x3);
10    int i;
11    i = i_coefficient_of_equation(x1,x2,x3);
12    int j;
13    j = j_coefficient_of_equation(x1,x2,x3);
14    int k;
15    k = k_coefficient_of_equation(x1,x2,x3);
16    printf("%d \n", i);
17    printf("%d \n", j);
18    printf("%d \n", k);
19 }
```

```
1 #ifndef POISK_ZNACHENIY_H
2 #define POISK_ZNACHENIY_H
3
4
5 void coefficients_of_equation();
6
7 #endif // POISK_ZNACHENIY_H
```

```
1
2 int i_coefficient_of_equation(int x1, int x2, int x3)
3 {
4     int result1, result2, result3;
5     int j;
6     int i, i_rez;
```

```

7   int k;
8   for(i=-100; i <100 ; i++) {
9       for(j=-100; j <100 ; j++) {
10          for(k=-100; k <100 ; k++) {
11              result1 = x1*x1*x1 + x1*x1*i+x1*j + k;
12              result2 = x2*x2*x2 + x2*x2*i+x2*j + k;
13              result3 = x3*x3*x3 + x3*x3*i+x3*j + k;
14              if (result1 == 0) {
15                  if(result2 == 0){
16                      if (result3 == 0 )
17                          {
18                              i_rez = i;
19                          }
20                      }
21                  }
22              }
23          }
24      }
25      return i_rez;
26  }
27
28  int j_coefficient_of_equation(int x1, int x2, int x3)
29  {
30      int result1, result2, result3;
31      int j, j_rez;
32      int i;
33      int k;
34      for(i=-100; i <100 ; i++) {
35          for(j=-100; j <100 ; j++) {
36              for(k=-100; k <100 ; k++) {
37                  result1 = x1*x1*x1 + x1*x1*i+x1*j + k;
38                  result2 = x2*x2*x2 + x2*x2*i+x2*j + k;
39                  result3 = x3*x3*x3 + x3*x3*i+x3*j + k;
40                  if (result1 == 0){
41                      if (result2 == 0){
42                          if (result3 == 0 )
43                              {
44                                  j_rez = j;
45                              }
46                          }
47                      }
48                  }
49              }
50          }
51      return j_rez;
52  }
53
54  int k_coefficient_of_equation(int x1, int x2, int x3)
55  {

```

```

56     int result1, result2, result3;
57     int j;
58     int i;
59     int k, k_rez;
60     for(i=-100; i <100 ; i++) {
61         for(j=-100; j <100 ; j++) {
62             for(k=-100; k <100 ; k++) {
63                 result1 = x1*x1*x1 + x1*x1*i+x1*j + k;
64                 result2 = x2*x2*x2 + x2*x2*i+x2*j + k;
65                 result3 = x3*x3*x3 + x3*x3*i+x3*j + k;
66                 if (result1 == 0){
67                     if (result2 == 0){
68                         if (result3 == 0 )
69                             {
70                                 k_rez = k;
71                             }
72                         }
73                     }
74                 }
75             }
76         }
77     return k_rez;
78 }

```

```

1  #ifndef POISK_ZNACHENIY2_H
2  #define POISK_ZNACHENIY2_H
3
4
5
6  #ifdef __cplusplus
7  extern "C"{
8  #endif
9
10 int i_coefficient_of_equation(int, int, int);
11 int j_coefficient_of_equation(int, int, int);
12 int k_coefficient_of_equation(int, int, int);
13
14 #ifdef __cplusplus
15 }
16 #endif
17
18 #endif // POISK_ZNACHENIY2_H

```

1.2 Задание 2

1.2.1 Задание

На шахматной доске стоят черный король и три белые ладьи (ладья бьет по горизонтали и вертикали). Определить, не находится ли король под боем, а если есть угроза, то от кого именно. Координаты короля и ладей вводить целыми числами.

1.2.2 Теоритические сведения

С помощью `main.c`, находящейся в многомодульном проекте `subproject`, можно задать параметр запуска для автоматического выполнения `treat_to_king_of_chess.c`, находящейся в статической библиотеке `lib`, в параметрах нужно указать `--is-poisk_ugrozi` и через пробелы 8 значения, равные координатам короля и ладей. Так же при задании значения параметра запуска в виде `--interactive` включается интерактивный режим, где данная функция принимает значения, вводимые пользователем программы, выбор выполнения данной задачи описан в `main_menu.c`, где использовались операторы условного перехода `switch`. Ввод и вывод данных в пользовательском режиме происходит в подпроекте `app` в `treat_to_king_of_chess.c`, заголовочным файлом которого является `treat_to_king_of_chess.h`.

Были созданы модульные тесты в подпроекте `test`.

1.2.3 Проектирование

В `main.c` у пользователя запрашивают режим работы программы, состоящий из:

1. `--interactive` - Ручной ввод значений
2. `--is-poisk_ugrozi`- Автоматическая работа, через ввод параметров запуска

В `treat_to_king_of_chess.c`, находящейся в подпроекте `app`, реализовано взаимодействие с пользователем, считывая введенные значения с консоли, и передавая их в `treat_to_king_of_chess.c`, находящейся в статической библиотеке `lib`, где производится поиск коэффициентов уравнений.

Во время автоматической работы в `treat_to_king_of_chess.c` значения передаются из параметров запуска.

Модульные тесты находятся в `test tst_testtest.cpp`.
Листинги `main.c` и `main_menu.c` приведены в приложении

1.2.4 Описание тестового стенда и методики тестирования

Среда разработки QtCreator 3.5.0, компилятор GCC 4.8.4 (x86 64 bit), операционная система Linux Mint 17.2 Cinnamon 64 bit. В процессе выполнения задания производилось ручное тестирование. Модульное тестирование реализовано при помощи фреймворка QTest.

1.2.5 Тестовый план и результаты тестирования

При вызове автоматического теста программа обращается к методу класса `TestTest`

`void test_treat_to_king_of_chess_function()`, в которой по уже заданным значениям производится поиск коэффициентов и сравнение результатов с помощью процедуры `QCOMPARE`. Для модульного теста, в котором координаты короля

$$x = 1, y = 3 \quad (1.4)$$

, координаты первой ладьи

$$x = 2, y = 2 \quad (1.5)$$

, координаты второй ладьи

$$x = 2, y = 1 \quad (1.6)$$

, координаты третьей ладьи

$$x = 4, y = 5 \quad (1.7)$$

угроза королю от второй ладьи. Тест прошел успешно. Для ручного теста, в котором

$$king_x = 1, king_y = 1 \quad (1.8)$$

$$rook1_x = 1, rook1_y = 5 \quad (1.9)$$

$$rook2_x = 10, rook2_y = 10 \quad (1.10)$$

$$rook3_x = 3, rook3_y = 15. \quad (1.11)$$

, угроза исходит от первой ладьи. Тест прошел успешно. Листинги модульных тестов приведены в приложении.

1.2.6 Выводы

При написании данной работы были улучшены навыки работы с отладкой(debug), навыки создания модульных тестов и умение разбивать задачи на подзадачи, отделяя общение с пользователем от бизнес-логики.

Листинги

```
1 #include <stdio.h>
2 #include "treat_to_king_of_chess_function.h"
3
4 void treat_to_king_of_chess(){
5     puts("Введите 8 цифр, обозначающих позиции короля и ладей
6         , через клавишу Enter");
7     /// ни о чем не говорящие имена для переменных
8     int king_x, king_y, y1, y2, z1, z2, a1, a2;
9     scanf("%d", &king_x);
10    scanf("%d", &king_y);
11    scanf("%d", &y1);
12    scanf("%d", &y2);
13    scanf("%d", &z1);
14    scanf("%d", &z2);
15    scanf("%d", &a1);
16    scanf("%d", &a2);
17    treats_to_king_of_chess(king_x, king_y, y1, y2, z1, z2, a1, a2);
18 }
```

```
1 #ifndef POISK_UGROZI_H
2 #define POISK_UGROZI_H
3
4 void treat_to_king_of_chess();
5
6 #endif // POISK_UGROZI_H
```

```
1 #include <stdio.h>
2 /// ни о чем не говорящие имена для переменных
3 /// и лучше сделать структуру для координат фигуры
4 int treats_to_king_of_chess(int king_x, int king_y, int y1,
5     int y2, int z1, int z2, int a1, int a2){
6     if (king_x==y1){
7         /// Вместо не понятных кодов, пусть функция возвращает
8         те единицы
9         return 1;
10    }
11    if (king_x==z1){
12        return 2;
13    }
14 }
```

```

12     if (king_x==a1){
13         return 3;
14     }
15     if (king_y==y2){
16         return 1;
17     }
18     if (king_y==z2){
19         return 2;
20     }
21     if (king_y==a2){
22         return 3;
23     }
24     else
25         return 0;
26 }

```

```

1  #ifndef POISK_UGROZI2_H
2  #define POISK_UGROZI2_H
3
4
5  #ifdef __cplusplus
6  extern "C"{
7  #endif
8
9
10 int treats_to_king_of_chess(int, int, int, int, int, int, int
    , int);
11
12 #ifdef __cplusplus
13 }
14 #endif
15 #endif // POISK_UGROZI2_H

```

Глава 2

Циклы

2.1 Задание 1

2.1.1 Задание

Составить из соответствующих цифр чисел M и N наибольшее возможное число. Примеры: 4157, 8024 > 8157; 323, 10714 > 10724.

2.1.2 Теоритические сведения

С помощью `main.c`, находящейся в многомодульном проекте `subproject`, можно задать параметр запуска для автоматического выполнения `max_composite_numbers.c`, находящейся в статической библиотеке `lib`, в параметрах нужно указать `--is-max_vozmojnoe` и через пробелы 2 значения, равные значениям двух чисел M и N . В `max_composite_numbers.c` используется цикл `while`, математические операции `pow`, `floor`, `fmod`. Также при задании значения параметра запуска в виде `--interactive` включается интерактивный режим, где данная функция принимает значения, вводимые пользователем программы, выбор выполнения данной задачи описан в `main_menu.c`, где использовались операторы условного перехода `switch`. Ввод и вывод данных в пользовательском режиме происходит в подпроекте `app` в `max_composite_number.c`, заголовочным файлом которого является `max_composite_number.h`. Были созданы модульные тесты в подпроекте `test`.

2.1.3 Проектирование

В `main.c` у пользователя запрашивают режим работы программы, состоящий из:

1. `--interactive` - Ручной ввод значений
2. `--is-max_vozmojnoe`- Автоматическая работа, через ввод параметров запуска

В `max_composite_number.c`, находящейся в подпроекте `app`, реализовано взаимодействие с пользователем, считывая введенные значения с консоли, и передавая их в `max_composite_numbers.c`, находящейся в статической библиотеке `lib`, где производится поиск коэффициентов уравнений.

Во время автоматической работы в `max_composite_numbers.c` значения передаются из параметров запуска.

Модульные тесты находятся в `test tst_testtest.cpp`.

Листинги `main.c` и `main_menu.c` приведены в приложении

2.1.4 Описание тестового стенда и методики тестирования

Среда разработки QtCreator 3.5.0, компилятор GCC 4.8.4 (x86 64 bit), операционная система Linux Mint 17.2 Cinnamon 64 bit. В процессе выполнения задания производилось ручное тестирование. Модульное тестирование реализовано при помощи фреймворка QtTest.

2.1.5 Тестовый план и результаты тестирования

При вызове автоматического теста программа обращается к методу класса `TestTest`

`void test_max_composite_number_function()`, в которой по уже заданным значениям производится поиск коэффициентов и сравнение результатов с помощью процедуры `QCOMPARE`. Для модульного теста, в котором

$$M = 1038, N = 5147 \quad (2.1)$$

, Максимальное возможное составное число - 5148. Тест прошел успешно. Для ручного теста, в котором

$$M = 38, N = 500 \quad (2.2)$$

, максимальное возможное составное число - 538. Тест прошел успешно. Листинги модульных тестов приведены в приложении.

2.1.6 Выводы

При написании данной работы были получены навыки работы со стандартной библиотекой `math.h`, навыки создания циклов `while`.

Листинги

```
1 #include <stdio.h>
2 #include <math.h>
3 #include "finding_max_composite_number_function.h"
4 void finding_max_composite_number(){
5     puts("Введите числа М и N через Enter");
6     int n2, m2;
7     scanf("%d", &n2);
8     scanf("%d", &m2);
9     printf("%d", findings_max_composite_number(m2, n2));
10
11 }
```

```
1 #ifndef MAX_VOZMOJNOE_H
2 #define MAX_VOZMOJNOE_H
3 void finding_max_composite_number();
4 #endif // MAX_VOZMOJNOE_H
```

```
1 #include <stdio.h>
2 #include <math.h>
3
4 int findings_max_composite_number(int m, int n){
5     int max=0, n_ostatok_ot_del, n_zhelaya_chast,
6         m_ostatok_ot_del, m_zhelaya_chast ;
7     int i=0;
8     while (n>0) {
9         n_ostatok_ot_del= floor(fmod(n, 10));
10        m_ostatok_ot_del= floor(fmod(m, 10));
11        n_zhelaya_chast=floor(n/10);
12        m_zhelaya_chast=floor(m/10);
13        if(n_ostatok_ot_del>m_ostatok_ot_del)
14            max=max+pow(10,i)*n_ostatok_ot_del;
15        else
16            max=max+pow(10,i)*m_ostatok_ot_del;
17        i=i+1;
18        n=n_zhelaya_chast;
19        m=m_zhelaya_chast;
20    }
21    if (m>n)
22        max=max+pow(10,i)*m;
23    else
```

```
23         max=max+pow(10,i)*n;
24
25
26     return max;
27 }
```

```
1  #ifndef MAX_VOZMOJNOE2_H
2  #define MAX_VOZMOJNOE2_H
3
4  #ifdef __cplusplus
5  extern "C"{
6  #endif
7
8
9  int findings_max_composite_number(int,int);
10
11 #ifdef __cplusplus
12 }
13 #endif
14
15 #endif // MAX_VOZMOJNOE2_H
```

Глава 3

Массивы

3.1 Задание 1

3.1.1 Задание

Каждый элемент вектора $A(n)$ (кроме двух крайних) заменить выражением:

$$a_i = (a_{i-1} + 2a_i + a_{i+1})/4 \quad (3.1)$$

, а крайние элементы – выражениями:

$$a_1 = (a_1 + a_2)/2, a_n = (a_{n-1} + a_n)/2 \quad (3.2)$$

.

3.1.2 Теоритические сведения

С помощью `main.c`, находящейся в многомодульном проекте `subproject`, можно задать параметр запуска для автоматического выполнения `replacement_of_elements_in_array.c`, находящейся в подпроекте `app`, в параметрах нужно указать `--is-zamena_elementov_mass`. Данная программа работает с файлами, таким образом с клавиатуры необходимо будет ввести путь к файлу, в котором хранятся элементы массива. В `replacement_of_elements_in_array.c` используется цикл `for`, функции работы с памятью `free`, `malloc` и функции работы с файлами: `fopen`, `fscanf`, `fclose`. Так же при задании значения параметра запуска в виде `--interactive` включается интерактивный режим, где данная функция принимает значения равное пути к файлу, вводимое пользователем программы, выбор выполнения данной задачи описан в `main_menu.c`, где использовались операторы условного перехода `switch`.

3.1.3 Проектирование

В `replacement_of_elements_in_array.c` реализовано взаимодействие с пользовательским файлом, считывая введенные значения с файла, указанного пользователем, `replacement_of_elements_in_array.c` производит замену элементов по заданному условию.

Листинги `main.c` и `main_menu.c` приведены в приложении

3.1.4 Описание тестового стенда и методики тестирования

Среда разработки QtCreator 3.5.0, компилятор GCC 4.8.4 (x86 64 bit), операционная система Linux Mint 17.2 Cinnamon 64 bit. В процессе выполнения задания производилось ручное тестирование.

3.1.5 Тестовый план и результаты тестирования

Во время выполнения ручных тестов сбоев не происходило, программа меняла значения элементов массива в соответствии с условием.

3.1.6 Выводы

При написании данной работы были получены навыки работы с файлами и массивами.

Листинги

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 void replacement_of_elements_in_array(){
5
6     FILE *mf;
7     char in_file[500];
8     printf("%s",&in_file);
9
10    mf=fopen(in_file,"r");
11    int n, i=0;
12    float *p;
13    fscanf(mf,"%d \n",&n);
14    p = (float *) malloc(n*sizeof(float));
15
```



```

16     for (i = 0; i <= (n-1); i++) {
17         fscanf(mf, "%f\n", &p[i]);
18     }
19
20     fclose(mf);
21
22     p[0] = (p[0] + p[1]) / 2;
23     p[n-1] = (p[n-2] + p[n-1]) / 2;
24
25     for (i = 1; i < (n-1); i++) {
26         p[i] = (p[i-1] + 2*p[i] + p[i+1]) / 4;
27     }
28
29     for (i = 0; i <= (n-1); i++) {
30         printf("%f\n", *(p+i));
31     }
32
33     free(p);
34 }

```

```

1 #ifndef ZAMENA_ELEMETOV_MASS_H
2 #define ZAMENA_ELEMETOV_MASS_H
3 void replacement_of_elements_in_array();
4 #endif // ZAMENA_ELEMETOV_MASS_H

```

Глава 4

Строки

4.1 Задание 1

4.1.1 Задание

Текст, не содержащий собственных имен и сокращений, набран полностью прописными русскими буквами. Заменить все прописные буквы, кроме букв, стоящих после точки, строчными буквами.

4.1.2 Теоритические сведения

С помощью `main.c`, находящейся в многомодульном проекте `subproject`, можно задать параметр запуска для автоматического выполнения `sentence_to_lower.c`, находящейся в подпроекте `app`, в параметрах нужно указать `--is-sentence_to_lower`. Данная программа работает с файлами, таким образом с клавиатуры необходимо будет ввести путь к файлу, в котором хранятся элементы массива, и путь к файлу, в который будет записываться отредактированный текст. В `sentence_to_lower.c` используется цикл `for`, функция работы с символами `tolower` и функции работы с файлами: `fopen`, `fgetc`, `fputc`, `fclose`. Так же при задании значения параметра запуска в виде `--interactive` включается интерактивный режим, где данная функция принимает значения равное пути к двум файлам, вводимые пользователем программы, выбор выполнения данной задачи описан в `main_menu.c`, где использовались операторы условного перехода `switch`.

4.1.3 Проектирование

В `sentence_to_lower.c` реализовано взаимодействие с пользовательским файлом, считывая введенные значения с файла, указанного пользователем, `sentence_to_lower.c` производит замену прописных букв на строчные и записывает во второй файл, указанный пользователем.

Листинги `main.c` и `main_menu.c` приведены в приложении

4.1.4 Описание тестового стенда и методики тестирования

Среда разработки QtCreator 3.5.0, компилятор GCC 4.8.4 (x86 64 bit), операционная система Linux Mint 17.2 Cinnamon 64 bit. В процессе выполнения задания производилось ручное тестирование.

4.1.5 Тестовый план и результаты тестирования

Во время выполнения ручных тестов сбоев не происходило, программа меняла предложения в соответствии с условием.

4.1.6 Выводы

При написании данной работы были получены навыки работы с файлами и символами.

Листинги

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4 #include <ctype.h>
5 void sentence_to_lower(){
6     int pointer_dot=1;
7     FILE* in;
8     FILE* out;
9
10    char in_file[500];
11    printf("%s",&in_file);
12    in=fopen(in_file,"r");
13
14    printf("%s",&in_file);
15    out=fopen(in_file,"r");
```

```

16
17     char str;
18     char ch = '.';
19
20
21     while (!feof(in)){
22         str=fgetc(in);
23         if (pointer_dot==1) fputc(str,out);
24             else
25                 fputc(tolower(str),out);
26         pointer_dot=0;
27         if (strcmp (str,ch)==0) pointer_dot=1;
28
29
30     }
31
32
33     fclose(in);
34     fclose(out);
35
36 }

```

```

1 #ifndef SENTENCE_TO_LOWER_H
2 #define SENTENCE_TO_LOWER_H
3 void sentence_to_lower();
4 #endif // SENTENCE_TO_LOWER_H

```

Глава 5

Стек

5.1 Задание 1

5.1.1 Задание

Для класса стек реализовать конструктор, конструктор копирования, деструктор, pop, push методы.

5.1.2 Теоритические сведения

Класс Stack находится в подпроекте `stack`. В заголовочном файле `stack.h` описан класс, а в исполняемом `stack.cpp` файле реализованы методы класса и конструкторы с деструктором. В `main.cpp` реализовано общение с пользователем и вызов методов и полей класса.

5.1.3 Проектирование

В `main.cpp` реализовано взаимодействие с пользователем. Метод класса Stack pop записывает элементы в поле класса `array`, метод pop выводит число в обратном порядке.

Пример.

$${}_1 = 0; {}_2 = 1; {}_3 = 4; \quad (5.1)$$

число -

$$410 \quad (5.2)$$

5.1.4 Описание тестового стенда и методики тестирования

Среда разработки QtCreator 3.5.0, компилятор GCC 4.8.4 (x86 64 bit), операционная система Linux Mint 17.2 Cinnamon 64 bit. В процессе выполнения задания производилось ручное тестирование.

5.1.5 Тестовый план и результаты тестирования

Во время выполнения ручных тестов сбоев не происходило

5.1.6 Выводы

При написании данной работы были получены навыки работы с классами.

Листинги

```
1 #include <iostream>
2 #include "stack.h"
3 using namespace std;
4
5 int main()
6 {
7     Stack st;
8     st.push();
9     cout<<st.pop();
10
11
12     return 0;
13 }
```

```
1 #ifndef STACK_H
2 #define STACK_H
3
4
5 class Stack
6 {
7 public:
8     Stack();
9     ~Stack();
10    Stack(const Stack &oStack);
11    void push();
12    int pop();
13 private:
```

```

14     int i;
15     int *array;
16 };
17
18 #endif // STACK_H

```

```

1  #include "stack.h"
2  #include <math.h>
3  #include <iostream>
4  using namespace std;
5
6  Stack::Stack()
7  {
8      array=new int[6];
9      i=0;
10 }
11
12 Stack::Stack(const Stack &oStack):
13     array(oStack.array),
14     i(oStack.i)
15 {
16
17 {
18
19     for (i=0; i<=5; i++)
20         array[i]=oStack.array[i];
21
22 }
23
24 Stack::~~Stack()
25 {
26     delete[] array;
27
28 }
29
30 void Stack::push()
31 {
32     i=0;
33     while(i<6){
34         cin>>array[i];
35         i++;
36     }
37
38 }
39
40 int Stack::pop()
41 {
42     i=5;
43     while(i>=0){

```

```
44         cout<<array[i];
45         i--;
46     }
47     return 0;
48
49 }
```


Глава 6

Приложение

Листинги

```
1 #include <stdio.h>
2 #include <string.h>
3 #include <stdlib.h>
4 #include "search_coefficients_of_equation_function.h"
5 #include "treat_to_king_of_chess_function.h"
6 #include "finding_max_composite_number_function.h"
7 #include "replacement_of_elements_in_array.h"
8 #include "main_menu.h"
9
10
11
12 int main(int argc, char* argv[])
13 {
14
15     if(argc == 2){
16
17         if(strcmp(argv[1], "--interactive") == 0){
18             main_menu();
19         }
20     }
21
22
23     if((strcmp(argv[1], "--is-max_vozmojnoe") == 0) && (
24         argc>=4)){
25         findings_max_composite_number(atoi(argv[3]),atoi(argv
26             [4]));
27         return(0);
28     }
29
30     if((strcmp(argv[1], "--is-poisk_ugrozi") == 0)&& (argc
31         >=5)){
```

```

29         treats_to_king_of_chess(atoi(argv[3]),atoi(argv[4]),
30             atoi(argv[5]),atoi(argv[6]),atoi(argv[7]),atoi(
31                 argv[8]),atoi(argv[9]),atoi(argv[10]));
32     return(0);
33 }
34 if(strcmp(argv[1], "--is-zamena_elementov_mass") == 0){
35     replacement_of_elements_in_array();
36     return(0);
37 }
38
39 return 0;
40 }

```

```

1  #include <stdio.h>
2  #include "coefficients_of_equation.h"
3  #include "treat_to_king_of_chess.h"
4  #include "max_composite_number.h"
5  #include "replacement_of_elements_in_array.h"
6  #include "sentence_to_lower.h"
7
8
9
10 void main_menu()
11 {
12
13     puts("1. Поиск коэффициентов");
14     puts("2. Поиск угрозы королю от ладьи");
15     puts("3. Составить из соответствующих чисел М и N наибольшее возможное число");
16     puts("4. Замена значений элементов массива(работа с файлом)");
17     puts("5. Перевод прописных букв текста в строчные(работа с файлом)");
18     int choice;
19     scanf("%d", &choice);
20     switch (choice) {
21     case 1:
22         coefficients_of_equation();
23         break;
24     case 2:
25         treat_to_king_of_chess();
26         break;
27     case 3:
28         finding_max_composite_number();
29         break;
30     case 4:
31         replacement_of_elements_in_array();

```

```

32         break;
33     case 5:
34         sentence_to_lower();
35         break;
36
37
38
39     }
40 }

```

```

1 #ifndef MAIN_MENU_H
2 #define MAIN_MENU_H
3 void main_menu();
4 #endif // MAIN_MENU_H

```

```

1 #include <QString>
2 #include <QtTest>
3 #include "finding_max_composite_number_function.h"
4 #include "search_coefficients_of_equation_function.h"
5 #include "treat_to_king_of_chess_function.h"
6
7 class TestTest : public QObject
8 {
9     Q_OBJECT
10
11 public:
12     TestTest();
13
14 private Q_SLOTS:
15     void testCase1();
16     void test_max_composite_number_function();
17     void test_treat_to_king_of_chess_function();
18     void test_i_coefficient_of_equation_function();
19     void test_j_coefficient_of_equation_function();
20     void test_k_coefficient_of_equation_function();
21 };
22
23 TestTest::TestTest()
24 {
25 }
26
27 void TestTest::testCase1()
28 {
29     QVERIFY2(true, "Failure");
30 }
31
32
33 /// Надо добавить еще тестовых вариантов
34 void TestTest::test_max_composite_number_function(){

```

```

35     QCOMPARE(findings_max_composite_number(1038,5147),5148);
36 }
37
38 /// Вы проверяете здесь только один вариант угрозы, было бы ж
орошо проверить все или почти все,
39 /// можно дописать в этот же тест
40 void TestTest::test_treat_to_king_of_chess_function(){
41     QCOMPARE(treats_to_king_of_chess(1,3,2,2,2,1,4,5),2);
42 }
43
44 void TestTest::test_i_coefficient_of_equation_function(){
45     QCOMPARE(i_coefficient_of_equation(2,2,2),10);
46 }
47
48 void TestTest::test_j_coefficient_of_equation_function(){
49     QCOMPARE(j_coefficient_of_equation(2,2,2),10);
50 }
51
52 void TestTest::test_k_coefficient_of_equation_function(){
53     QCOMPARE(k_coefficient_of_equation(2,2,2),10);
54 }
55
56
57 QTEST_APPLESS_MAIN(TestTest)
58 #include "tst_testtest.moc"

```