Todo list

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

$$x^3 + bx^2 + cx + d \tag{1.1}$$

(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 происходет 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 взаимодейтсвие 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 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 рузельтатов с помощью процедуры `QCOMPARE`. Для модульного теста, в котором

$$x_1 = x_2 = x_3 = 2, \tag{1.2}$$

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

$$x_1 = 3; x_2 = 2; x_3 = 1, i = -6; j = 11; k = -6. \tag{1.3}$$

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

```c
#include <stdio.h>
#include "search_coefficients_of_equation_function.h"

void coefficients_of_equation(){
    puts("Введите 3 значени x через Enter");
    int x1, x2, x3;
    scanf("%d", &x1);
    scanf("%d", &x2);
    scanf("%d", &x3);
    int i;
    i = i_coefficient_of_equation(x1,x2,x3);
    int j;
    j = j_coefficient_of_equation(x1,x2,x3);
    int k;
    k = k_coefficient_of_equation(x1,x2,x3);
    printf("%d \n", i);
    printf("%d \n", j);
    printf("%d \n", k);
}
```

```c
#ifndef POISK_ZNACHENIY_H
#define POISK_ZNACHENIY_H


void coefficients_of_equation();

#endif // POISK_ZNACHENIY_H
```

```c

int i_coefficient_of_equation(int x1, int x2, int x3)
{
    int result1, result2, result3;
    int j;
    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  {
```

```
    int result1, result2, result3;
    int j;
    int i;
    int k, k_rez;
    for(i=-100; i <100 ; i++) {
        for(j=-100; j <100 ; j++) {
            for(k=-100; k <100 ; k++) {
                result1 = x1*x1*x1 + x1*x1*i+x1*j + k;
                result2 = x2*x2*x2 + x2*x2*i+x2*j + k;
                result3 = x3*x3*x3 + x3*x3*i+x3*j + k;
                if (result1 == 0){
                    if (result2 == 0){
                        if (result3 == 0 )
                        {
                            k_rez = k;
                        }
                    }
                }
            }
        }
    }
    return k_rez;
}
```

```
#ifndef POISK_ZNACHENIY2_H
#define POISK_ZNACHENIY2_H



#ifdef __cplusplus
extern "C"{
#endif

int i_coefficient_of_equation(int, int, int);
int j_coefficient_of_equation(int, int, int);
int k_coefficient_of_equation(int, int, int);

#ifdef __cplusplus
}
#endif

#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_chesss.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 происходет в подпроекте `app` в `treat_to_king_of_chess` 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 взаимодейтсвие с пользователем, считывая введенные значения с консоли, и передавая их в `treat_to_king_of_chesss.c`, находящейся в статической библиотеке `lib`, где производится поиск коэффициентов уравнений.
Во время автоматической работы в `treat_to_king_of_chesss.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 QtTest.

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 рузельтатов с помощью процедуры `QCOMPARE`. Для модульного теста, в котором координаты короля

$$x = 1, y = 3 \tag{1.4}$$

, координаты первой ладьи

$$x = 2, y = 2 \tag{1.5}$$

, координаты второй ладьи

$$x = 2, y = 1 \tag{1.6}$$

, координаты третьей ладьи

$$x = 4, y = 5 \tag{1.7}$$

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

$$king_x = 1, king_y = 1, rook1_x = 1, rook1_y = 5, rook2_x = 10, rook2_y = 10, rook3_x = 3, rook3_y = 15. \tag{1.8}$$

, 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

```c
#include <stdio.h>
#include "treat_to_king_of_chess_function.h"

void treat_to_king_of_chess(){
    puts("Введите 8 цифр, обозначающих позиции короля и ладей, через клавишу Enter");
    int king_x, king_y, rook1_x, rook1_y, rook2_x, rook2_y, rook3_x, rook3_y;
    scanf("%d", &king_x);
    scanf("%d", &king_y);
    scanf("%d", &rook1_x);
    scanf("%d", &rook1_y);
    scanf("%d", &rook2_x);
    scanf("%d", &rook2_y);
    scanf("%d", &rook3_x);
    scanf("%d", &rook3_y);
    treats_to_king_of_chess(king_x,king_y,rook1_x,rook1_y,rook2_x,rook2_y,rook3_x,rook3_y);
}
```

```c
#ifndef POISK_UGROZI_H
#define POISK_UGROZI_H

void treat_to_king_of_chess();

#endif // POISK_UGROZI_H
```

```c
#include <stdio.h>
int treats_to_king_of_chess(int king_x, int king_y, int rook1_x, int rook1_y, int rook2_x, int rook2_y, int rook3_x, int rook3_y){
    if (king_x==rook1_x){
        return 1;
    }
    if (king_x==rook2_x){
        return 2;
    }
    if (king_x==rook3_x){
        return 3;
    }
    if (king_y==rook1_y){
        return 1;
    }
    if (king_y==rook2_y){
        return 2;
    }
```

```
18      if (king_y==rook3_y){
19          return 3;
20      }
21      else
22          return 0;
23  }
```

```
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. 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 происходет 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 взаимодейтсвие 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 рузельтатов с помощью процедуры `QCOMPARE`. Для модульного теста, в котором

$$M = 1038, N = 5147 \tag{2.1}$$

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

$$M = 38, N = 500 \tag{2.2}$$

, 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

```c
#include <stdio.h>
#include <math.h>
#include "finding_max_composite_number_function.h"
void finding_max_composite_number(){
    puts("Введите числа M и N через Enter");
    int n2, m2;
    scanf("%d", &n2);
    scanf("%d", &m2);
    printf("%d",findings_max_composite_number(m2,n2));

}
```

```c
#ifndef MAX_VOZMOJNOE_H
#define MAX_VOZMOJNOE_H
void finding_max_composite_number();
#endif // MAX_VOZMOJNOE_H
```

```c
#include <stdio.h>
#include <math.h>

int findings_max_composite_number(int m, int n){
    int max=0, n_ostatok_ot_del, n_zhelaya_chast,
       m_ostatok_ot_del, m_zhelaya_chast ;
    int i=0;
    while (n>0) {
            n_ostatok_ot_del= floor(fmod(n, 10));
            m_ostatok_ot_del= floor(fmod(m, 10));
            n_zhelaya_chast=floor(n/10);
            m_zhelaya_chast=floor(m/10);
            if(n_ostatok_ot_del>m_ostatok_ot_del)
                max=max+pow(10,i)*n_ostatok_ot_del;
            else
                max=max+pow(10,i)*m_ostatok_ot_del;
            i=i+1;
            n=n_zhelaya_chast;
            m=m_zhelaya_chast;
        }
    if (m>n)
    max=max+pow(10,i)*m;
        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:

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

, а крайние элементы – выражениями:

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

.

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 хранится 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 взаимодейтсвие 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     gets(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 хранится 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 взаимодейтсвие 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

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
void sentence_to_lower(){
    int pointer_dot=1;
    FILE* in;
    FILE* out;

    char in_file[500];
    gets(in_file);
    in=fopen(in_file,"r");

    gets(in_file);
    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 взаимодейтсвие с пользователем. Метод класса Stack `pop` записывает элементы в поле класса `array`, метод `pop` выводит число в обратном порядке.
Пример.

$$_1 = 2; _2 = 3; _3 = 4; \tag{5.1}$$

число -

$$431 \tag{5.2}$$

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

```
#include <iostream>
#include "stack.h"
using namespace std;

int main()
{
    Stack st;
    st.push(1);
    st.push(2);
    st.push(3);
    st.push(4);
    st.push(5);
    st.push(6);


    cout<<st.pop();


    return 0;
}
```

```
#ifndef STACK_H
#define STACK_H


class Stack
{
```

```
public:
    Stack();
    ~Stack();
    Stack(const Stack &oStack);
    void push(int);
    int pop();
    int b;
private:
    int i;
    int *array;
};

#endif // STACK_H
```

```
#include "stack.h"
#include <math.h>
#include <iostream>
using namespace std;

Stack::Stack()
{
    array=new int(6);
    i=0;
    b=0;
}

Stack::Stack(const Stack &oStack):
     array(oStack.array),
     i(oStack.i)


{

     for (i=0; i<6; i++)
         array[i]=oStack.array[i];

}

Stack::~Stack()
{
  delete[] array;

}

void Stack::push(int a)
{
    array[i]=a;
    i++;

```

```
36 }
37
38 int Stack::pop()
39 {
40     i=5;
41     while(i>=0){
42         cout<<array[i];
43         i--;
44     }
45 return 0;
46
47 }
```

Глава 6

Приложение

Листинги

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include "search_coefficients_of_equation_function.h"
#include "treat_to_king_of_chess_function.h"
#include "finding_max_composite_number_function.h"
#include "replacement_of_elements_in_array.h"
#include "main_menu.h"
#include "sentence_to_lower.h"


int main(int argc, char* argv[])
{

       if(argc == 2){

        if(strcmp(argv[1], "--interactive") == 0){
            main_menu();
        }
    }


       if((strcmp(argv[1], "--is-max_vozmojnoe") == 0) && (argc>=4)){
        findings_max_composite_number(atoi(argv[3]),atoi(argv[4]));
        return(0);
    }

    if((strcmp(argv[1], "--is-poisk_ugrozi") == 0)&& (argc>=5)){
```

```
29          treats_to_king_of_chess(atoi(argv[3]),atoi(argv[4]),
                atoi(argv[5]),atoi(argv[6]),atoi(argv[7]),atoi(
                argv[8]),atoi(argv[9]),atoi(argv[10]));
30          return(0);
31      }
32      if(strcmp(argv[1], "--is-zamena_elementov_mass") == 0){
33          replacement_of_elements_in_array();
34          return(0);
35      }
36      if((strcmp(argv[1], "---is-coefficients_of_equation") ==
            0)&& (argc>=5)){
37          i_coefficient_of_equation(atoi(argv[3]),atoi(argv[4])
                ,atoi(argv[5]));
38          j_coefficient_of_equation(atoi(argv[3]),atoi(argv[4])
                ,atoi(argv[5]));
39          k_coefficient_of_equation(atoi(argv[3]),atoi(argv[4])
                ,atoi(argv[5]));
40          return(0);
41      }
42      if(strcmp(argv[1], "--is-sentence_to_lower") == 0){
43          sentence_to_lower();
44          return(0);
45      }
46
47
48
49  return 0;
50  }
```

```
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 M и N наиболь
            шое возможное число");
16      puts("4. Замена значений элеметнов массива(работа с файло
            м)");
17      puts("5. Перевод прописных букв текста в строчные(работа
            с файлом)");
```

```
    int choice;
    scanf("%d", &choice);
    switch (choice) {
    case 1:
        coefficients_of_equation();
        break;
    case 2:
        treat_to_king_of_chess();
        break;
    case 3:
        finding_max_composite_number();
        break;
    case 4:
        replacement_of_elements_in_array();
        break;
    case 5:
        sentence_to_lower();
        break;



    }
}
```

```
#ifndef MAIN_MENU_H
#define MAIN_MENU_H
void main_menu();
#endif // MAIN_MENU_H
```

```
#include <QString>
#include <QtTest>
#include "finding_max_composite_number_function.h"
#include "search_coefficients_of_equation_function.h"
#include "treat_to_king_of_chess_function.h"

class TestTest : public QObject
{
    Q_OBJECT

public:
    TestTest();

private Q_SLOTS:
    void testCase1();
    void test_max_composite_number_function();
    void test_treat_to_king_of_chess_function();
    void test_i_coefficient_of_equation_function();
    void test_j_coefficient_of_equation_function();
    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
34 void TestTest::test_max_composite_number_function(){
35     QCOMPARE(findings_max_composite_number(1038,5147),5148);
36 }
37
38
39 void TestTest::test_treat_to_king_of_chess_function(){
40     QCOMPARE(treats_to_king_of_chess(1,3,2,2,2,1,4,5),2);
41 }
42
43 void TestTest::test_i_coefficient_of_equation_function(){
44     QCOMPARE(i_coefficient_of_equation(2,2,2),10);
45 }
46
47 void TestTest::test_j_coefficient_of_equation_function(){
48     QCOMPARE(j_coefficient_of_equation(2,2,2),10);
49 }
50
51 void TestTest::test_k_coefficient_of_equation_function(){
52     QCOMPARE(k_coefficient_of_equation(2,2,2),10);
53 }
54
55
56 QTEST_APPLESS_MAIN(TestTest)
57 #include "tst_testtest.moc"
```