Программирование

М.В.Булгакова

25 декабря 2015 г.

Глава 1

Основные конструкции языка

1.1 Задание 1

1.1.1 Задание

Пользователь задает три корня кубического уравнения

$$x^3 + bx^2 + cx + d = 0 \tag{1.1}$$

(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, операционная система Debian 32 bit, Cppcheck 1.67

Модульное тестирование реализовано при помощи фреймворка QtTest. Так же код проверялся с помощью Cppcheck. Программа Cppcheck указала на ошибки в функциях scanf и printf стандартной библиотеки и на дублированный код. Данные ошибки были исправлены, когда подключили заголовочный файл `<stdio.h>` и исправили метод `findings_max_composite_number`, удаляя дублированные части кода.

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

```
#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;
    int j;
    int k;
    coefficient_of_equation(x1,x2,x3, &i, &j, &k);
    printf("%d \n", i);
    printf("%d \n", j);
    printf("%d \n", k);


}
```

```
#ifndef POISK_ZNACHENIY_H
#define POISK_ZNACHENIY_H


void coefficients_of_equation();

#endif // POISK_ZNACHENIY_H
```

```
void coefficient_of_equation(int x1, int x2, int x3, int*
    i_rez, int* j_rez, int* k_rez)
{
     int result1, result2, result3;
     int j;
     int i;
     int k;
     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 )
                         {
                             *i_rez = i;
                             *j_rez = j;
                             *k_rez = k;
                         }
                     }
                 }
             }
         }
     }
}
```

```
#ifndef POISK_ZNACHENIY2_H
#define POISK_ZNACHENIY2_H



#ifdef __cplusplus
extern "C"{
#endif

void coefficient_of_equation(int, int, int, 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-treat_to_king` 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.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-treat_to_king`- 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, операционная система Debian 32 bit, Cppcheck 1.67 В процессе выполнения задания производилось ручное тестирование. Модульное тестирование реализовано при помощи фреймворка QtTest. Так же код проверялся с помощью Cppcheck. Программа Cppcheck указала на ошибки в функциях scanf и printf стандартной библиотеки. Данные ошибки были исправлены, когда подключили заголовочный файл `<stdio.h>`.

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

$$king_x = 1, king_y = 1 \tag{1.8}$$

$$rook1_x = 1, rook1_y = 5 \tag{1.9}$$

$$rook2_x = 10, rook2_y = 10 \tag{1.10}$$

$$rook3_x = 3, rook3_y = 15. \tag{1.11}$$

, 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

```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);
    printf("%d",treats_to_king_of_chesss(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_chesss(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)||(king_y==rook1_y)){
        return 1;
}
    if ((king_x==rook2_x)||(king_y==rook2_y)){
        return 2;
    }

```

```
    if ((king_x==rook3_x)||(king_y==rook3_y)){
        return 3;
    }
    else
        return 0;

}
```

```
#ifndef POISK_UGROZI2_H
#define POISK_UGROZI2_H


#ifdef __cplusplus
extern "C"{
#endif


int treats_to_king_of_chesss(int, int, int, int, int, int,
    int, int);

#ifdef __cplusplus
}
#endif
#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_composite_number` 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_composite_number-` 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, операционная система Debian 32 bit, Cppcheck 1.67 В процессе выполнения задания производилось ручное тестирование. Модульное тестирование реализовано при помощи фреймворка QtTest. Так же код проверялся с помощью Cppcheck. Программа Cppcheck указала на ошибки в функциях scanf и printf стандартной библиотеки и на дублированный код. Данные ошибки были исправлены, когда подключили заголовочный файл `<stdio.h>` и исправили метод `findings_max_composite_number`, удаляя дублированные части кода.

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
        max=max+pow(10,i)*n;


    return max;
}
```

```
#ifndef MAX_VOZMOJNOE2_H
#define MAX_VOZMOJNOE2_H

#ifdef __cplusplus
extern "C"{
#endif


int findings_max_composite_number(int,int);

#ifdef __cplusplus
}
#endif

#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-replacement_in_array`. 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, операционная система Debian 32 bit, Cppcheck 1.67 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.

Так же код проверялся с помощью Cppcheck. Программа Cppcheck указала на ошибки в функциях fscanf и fprintf стандартной библиотеки и на инициализированную, но нигде не использующуюся переменную. Данные ошибки были исправлены, когда подключили заголовочный файл `<stdio.h>` и удалили неиспользующуюся переменную.

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
```

```
    mf=fopen(in_file,"r");
    int n, i=0;
    float *p;
    fscanf(mf,"%d \n",&n);
    p = (float *) malloc(n*sizeof(float));

    for (i = 0; i<=(n-1); i++){
        fscanf(mf,"%f\n",&p[i]);
    }

    fclose(mf);

    p[0]=(p[0]+p[1])/2;
    p[n-1]=(p[n-2]+p[n-1])/2;

    for (i = 1; i<(n-1); i++){
       p[i]=(p[i-1]+2*p[i]+p[i+1])/4;
    }

    for (i = 0; i<=(n-1); i++){
        printf("%f\n",*(p+i));
    }

  free(p);
}
```

```
#ifndef ZAMENA_ELEMETOV_MASS_H
#define ZAMENA_ELEMETOV_MASS_H
void replacement_of_elements_in_array();
#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, операционная система Debian 32 bit, Cppcheck 1.67 В процессе выполнения задания производилось ручное тестирование.

Так же код проверялся с помощью Cppcheck. Программа Cppcheck указала на ошибки в функциях fscanf и fprintf стандартной библиотеки, на дублированный код, на strcmp, которую лучше не использовать. Данные ошибки были исправлены, когда подключили заголовочный файл `<stdio.h>` и исправили метод `findings_max_composite_number`, удаляя дублированные части кода. Warning на strcmp исправлению не подлежал, так как именно его более эффективно использовать для данной программы.

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;
```

```
    FILE* in;
    FILE* out;

    char in_file[500];
    printf("%s",&in_file);
    in=fopen(in_file,"r");

    printf("%s",&in_file);
    out=fopen(in_file,"r");

    char str;
    char ch = '.';


    while (!feof(in)){
        str=fgetc(in);
        if (pointer_dot==1) fputc(str,out);
                else
              fputc(tolower(str),out);
        pointer_dot=0;
        if (strcmp (str,ch)==0) pointer_dot=1;


    }


    fclose(in);
    fclose(out);

}
```

```
#ifndef SENTENCE_TO_LOWER_H
#define SENTENCE_TO_LOWER_H
void sentence_to_lower();
#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 `stacklib.h` описан класс, а в исполняемом `stacklib.cpp` файле реализованы методы класса и конструкторы с деструктором. Класс Stackapp находится в подпроекте `stack`. В заголовочном файле `stackapp.h` описан класс, а в исполняемом `stackapp.cpp` файле реализованы методы класса и конструкторы с деструктором. Так же в `stackapp.cpp` "бросается"исключение, на случай если пользователь захочет ввести значение для стека, выходящее за рамки типа int. В `main.cpp` реализовано общение с пользователем.

5.1.3 Проектирование

Метод класса Stack `push` записывает элементы в поле класса `stackPtr`, метод `printStack` выводит число в обратном порядке.
Пример.

$$_1 = 0; _2 = 1; _3 = 4; \tag{5.1}$$

число -

$$410 \tag{5.2}$$

5.1.4 Описание тестового стенда и методики тестирования

Среда разработки QtCreator 3.5.0, компилятор GCC, операционная система Debian 32 bit, Cppcheck 1.67 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>
using namespace std;

#include "stackapp.h"
int main()
{
    cout<<"Введите количество элементов стека"<<endl;
    int n;
    cin>>n;
    Stackapp st(n);
    st.launch();


    return 0;
}
```

```
#include "stacklib.h"
#include "stackapp.h"
#include <iostream>
using namespace std;
#include <cassert>
#include <iostream>

#include <iomanip>

Stack::Stack(int maxSize) :
    size(maxSize)
```

```
12 {
13     stackPtr = new int[size];
14     top = 0;
15 }
16
17
18 Stack::Stack(const Stack & otherStack) :
19     size(otherStack.getStackSize()) //
20 {
21     stackPtr = new int[size];
22     top = otherStack.getTop();
23
24
25     for(int ix = 0; ix < top; ix++)
26         stackPtr[ix] = otherStack.stackPtr[ix];
27
28 }
29
30
31 Stack::~Stack()
32 {
33     delete [] stackPtr;
34 }
35
36
37 void Stack::push(const int number)
38 {
39
40
41     assert(top < size);
42
43     stackPtr[top++] = number;
44 }
45
46
47
48
49
50
51 void Stack::printStack()
52 {
53     for (int ix = top-1 ; ix >= 0; ix--){
54         cout << "|" << stackPtr[ix];}
55 }
56
57
58 int Stack::getStackSize() const
59 {
60     return size;
```

```
61 }
62
63
64
65
66 int Stack::getTop() const
67 {
68     return top;
69 }
```

```
1  #ifndef STACKLIB_H
2  #define STACKLIB_H
3
4
5
6  class Stack
7  {
8  private:
9      int *stackPtr;
10     const int size;
11     int top;
12 public:
13
14     Stack(int = 10);
15     Stack(const Stack &);
16     ~Stack();
17
18     void push(const int number);
19     void printStack();
20     int getStackSize() const;
21     int getTop() const;
22 };
23
24
25
26 #endif // STACKLIB_H
```

```
1  #ifndef STACKAPP_H
2  #define STACKAPP_H
3
4  #include<iostream>
5  #include "stacklib.h"
6
7  class Stackapp
8  {
9  public:
10     Stackapp(int=10);
11     ~Stackapp();
12     void launch();
```

```
    int maxSize;
private:
    const int size;
    Stack* st;
    void getStack();
    void printStack();

};

#endif // STACKAPP_H
```

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

Stackapp::Stackapp(const int maxSize):
size(maxSize)
{
  st = new Stack(size);
}

Stackapp::~Stackapp()
{
    delete st;
}
void Stackapp::launch()
{
   cout<<"Введите значения элементов стека"<<endl;
   try{
   getStack();
   }
   catch(int thr){
       cerr << " Слишком большое число!!!" << endl;
   }
    printStack();
}

void Stackapp::getStack()
{
    int number;
    for(int i=0; i<size; i++){
    cin >> number;
    if (number>100000)
        throw 1;
    st->push(number);}
}

void Stackapp::printStack()
```

```
39 {
40     st->printStack();
41 }
```

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 "sentence_to_lower.h"
#include "main_menu.h"



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

        if(argc == 2){

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


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

```

```
29      if((strcmp(argv[1], "--is-treat_to_king") == 0)&& (argc
            >=5)){
30          treats_to_king_of_chesss(atoi(argv[3]),atoi(argv[4]),
                atoi(argv[5]),atoi(argv[6]),atoi(argv[7]),atoi(
                argv[8]),atoi(argv[9]),atoi(argv[10]));
31          return(0);
32      }
33      if(strcmp(argv[1], "--is-replacement_in_array") == 0){
34          replacement_of_elements_in_array();
35          return(0);
36      }
37      if(strcmp(argv[1], "--is-sentence_to_lower") == 0){
38          sentence_to_lower();
39          return(0);
40      }
41
42
43
44  return 0;
45  }
```

```
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)");
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();
```

```
        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_coefficient_of_equation_function();
};

TestTest::TestTest()
{
}

void TestTest::testCase1()
{
    QVERIFY2(true, "Failure");
```

```
29 }
30
31
32 void TestTest::test_max_composite_number_function(){
33     QCOMPARE(findings_max_composite_number(1038,5147),5148);
34     QCOMPARE(findings_max_composite_number(38,500),538);
35 }
36
37 void TestTest::test_treat_to_king_of_chess_function(){
38     QCOMPARE(treats_to_king_of_chesss(1,3,2,2,2,1,4,5),2);
39     QCOMPARE(treats_to_king_of_chesss(3,5,2,2,2,1,4,5),3);
40 }
41
42 void TestTest::test_coefficient_of_equation_function(){
43     int i;
44     int j;
45     int k;
46     coefficient_of_equation(1,2,3, &i, &j, &k);
47     QCOMPARE(i,-6);
48     QCOMPARE(j,11);
49     QCOMPARE(k,-6);
50 }
51
52
53
54
55
56 QTEST_APPLESS_MAIN(TestTest)
57 #include "tst_testtest.moc"
```