Программирование на C и C++

Лукашенко Р.Б.

27 декабря 2015 г.

Оглавление

Глава 1

Основные конструкции языка

1.1 Задание 1.1. Перевод дюймов в метрическую систему

1.1.1 Задание

Задано целое число от 0 до 999. Определить сумму его цифр.

1.1.2 Теоретические сведения

Для реализации данной задачи были использованы стандартные функции ввода-вывода `scanf`, `printf`, `puts` из стандартной библиотеки языка `C`, объявленные в заголовочном файле *stdio.h*.

При помощи операторов ветвления `if-else` и `switch` реализовано интерактивное подменю для более удобного взаимодействия пользователя с программой.

1.1.3 Проектирование

В ходе проектирования было решено выделить 5 функций:

1. Вычисление суммы

   ```
   int c_calc_sum_of_digits(int);
   ```

   Параметром функции является число типа `int`, которое вводит пользователь. Возвращается число типа `int`, которое и является суммой разрядов.

2. Меню с первоначальным пользовательским взаимодействием

`void c_sum_of_digits_ui();`

Пользователю предлагается выбрать консольный ввод, вызов справки, возварт к главное меню или завершение программы.

3. Основное пользовательское взаимодействие

   `void c_sum_of_digits_inp();`

   Пользователю предлагается ввести число, после чего вызывается функция для для решения задачи и вывода результата в консоль.

4. Решение задачи и вывода результата в консоль

   `void c_sum_of_digits_solution(int);`

   В качестве параметра передаётся число типа `int`. Вызывается функция вычисления суммы. Результат выводится в консоль.

5. Вспомогательная информация

   `void c_sum_of_digits_help();`

   Вывод справки в консоль.

1.1.4 Описание тестового стенда и методики тестирования

Среда разработки QtCreator 3.5.1, компилятор MinGW (x86 64 bit), операционная система Windows 10 64 bit. В процессе выполнения задания производилось ручное тестирование. Модульное тестирование реализовано при помощи фреймворка QtTest.

1.1.5 Тестирование

Таблица 1.1: Тестирование вычисления суммы

Введённое число	Сумма разрядов	Тип теста	Результат
1234	10	Модульный	Успешно
4321	10	Ручной	Успешно

Все тесты пройдены успешно.

1.1.6 Выводы

При выполнении задания были закреплены навыки в работе с основными конструкциями языка `C` и получен опыт в организации многофайлового проекта и создании модульных тестов.

Данная задача также была переписана на языке `C++` с использованием объектно-ориентироанного проектироания.

1.1.7 Листинги

c_sum_of_digits.h

```h
#ifndef C_SUM_OF_DIGITS
#define C_SUM_OF_DIGITS

#ifdef __cplusplus

extern "C" {

#endif

int c_calc_sum_of_digits(int);

#ifdef __cplusplus
}
#endif

#endif // C_SUM_OF_DIGITS
```

c_sum_of_digits.c

```c
#include"c_sum_of_digits.h"

int c_calc_sum_of_digits(int num)
{
    int counted=0, temp=0;
    while(num!=0)
    {
        temp=num%10;
        num=num/10;
        counted=counted+temp;
    }
    return counted;
}
```

c_sum_of_digits_ui.h

```h
#ifndef C_SUM_UI
#define C_SUM_UI

#include "c_sum_of_digits.h"

void c_sum_of_digits_ui();
void c_sum_of_digits_inp();
void c_sum_of_digits_solution(int);
void c_sum_of_digits_help();

#endif // C_SUM_UI
```

c_sum_of_digits_ui.c

```c
#include "c_main.h"
#include "c_sum_of_digits.h"
#include "c_sum_of_digits_ui.h"

void c_sum_of_digits_ui()
{
    int way;
    puts("Summ of digits task");
    puts("Choose option:");
    puts("1. Input number from console");
    puts("2. Help");
    puts("9. Back to main menu");
    puts("0. Exit");
    printf("Your choice: ");
    if (scanf("%d", &way) == 1)
    {
        switch (way)
        {
        case 0:
            system("cls");
            break;
        case 1:
            system("cls");
            c_sum_of_digits_inp();
            c_sum_of_digits_ui();
            break;
        case 2:
            system("cls");
            c_sum_of_digits_help();
            c_sum_of_digits_ui();
            break;
        case 9:
            system("cls");
            ui();
            break;
        default:
            system("cls");
            puts("Error. There's no such option.\n");
            c_sum_of_digits_ui();
            break;
        }
    }
    else
    {
        system("cls");
        puts("Error. Wrong input data type.\n");
        c_sum_of_digits_ui();
```

```
48      }
49  }
50
51  void c_sum_of_digits_inp()
52  {
53      int num_t1;
54      puts("Type your number: ");
55      scanf("%d", &num_t1);
56      c_sum_of_digits_solution(num_t1);
57  }
58
59  void c_sum_of_digits_solution(int num_t1)
60  {
61      int result_t1 = c_calc_sum_of_digits(num_t1);
62      printf("Result: %d\n\n", result_t1);
63      getch();
64      system("cls");
65  }
66
67  void c_sum_of_digits_help()
68  {
69      puts("HELP:");
70  }
```

1.2 Задание 1.2. Поиск кратных

1.2.1 Задание

Заданы три целых числа a, b и c. Найти среди них пары чисел, в которых одно число делится на
другое.

1.2.2 Теоритические сведения

Были использованы стандартные функции ввода-вывода `scanf`, `printf`, `puts` из стандартной библиотеки языка `C`, объявленные в заголовочном файле *stdio.h*.

При помощи операторов ветвления `if-else` и `switch` реализовано интерактивное подменю для более удобного взаимодействия пользователя с программой.

1.2.3 Проектирование

В ходе проектирования было решено выделить 5 функций:

1. Поиск кратных

   ```
   int c_calc_multiples(int*);
   ```

   Параметром функции является массив из трёх чисел типа `int`, которые вводит пользователь. Возвращается число типа `int`, которое является количеством кратных чисел. Вывод производится прямо в этой функции.

2. Меню с начальным пользовательским взаимодействием

   ```
   void c_multiples_ui();
   ```

   Пользователю предлагается выбрать консольный ввод, вызов справки, возварт к главное меню или завершение программы.

3. Основное пользовательское взаимодействие

   ```
   void c_multiples_inp();
   ```

   Пользователю предлагается последовательно ввести 3 числа, после чего вызывается функция для решения задачи и вывода результата в консоль.

4. Решение задачи и вывода результата в консоль

   `void c_multiples_solution(int*);`

   В качестве параметра передаётся массив из трёх чисел типа `int`. Вызывается функция поиска. Результат выводится в консоль

5. Вспомогательная информация

   `void c_multiples_help();`

   Вывод справки в консоль.

1.2.4 Описание тестового стенда и методики тестирования

Среда разработки QtCreator 3.5.1, компилятор MinGW (x86 64 bit), операционная система Windows 10 64 bit. В процессе выполнения задания производилось ручное тестирование. Модульное тестирование реализовано при помощи фреймворка QtTest.

1.2.5 Тестирование

Таблица 1.2: Тестирование нахождения кратных

m1	m2	m3	Количество пар кратных	Тип теста	Результат
2	4	8	3	Модульный	Успешно
8	4	2	3	Ручной	Успешно

Все тесты пройдены успешно.

1.2.6 Выводы

При выполнении задания закреплены навыки в работе с основными конструкциями языка `C` и получен опыт в организации многофайлового проекта и создании модульных тестов.

Данная задача также была переписана на языке `C++` с использованием объектно-ориентироанного проектироания.

1.2.7 Листинги

c_multiples.h

```
#ifndef C_MULTIPLES
#define C_MULTIPLES

#ifdef __cplusplus

extern "C" {

#endif

int c_calc_multiples(int*);

#ifdef __cplusplus
}
#endif

#endif // C_MULTIPLES
```

c_multiples_ui.h

```
#ifndef C_MULTIPLES_UI
#define C_MULTIPLES_UI

#include "c_multiples.h"

void c_multiples_ui();
void c_multiples_inp();
void c_multiples_solution(int*);
void c_multiples_help();

#endif // C_MULTIPLES_UI
```

c_multiples_ui.c

```
#include "c_main.h"
#include "c_multiples.h"
#include "c_multiples_ui.h"

void c_multiples_ui()
{
    int way;
    puts("Search for multiples task");
    puts("Choose option:");
    puts("1. Input 3 numbers");
    puts("2. Help");
    puts("9. Back to main menu");
    puts("0. Exit");
    printf("Your choice: ");
```

```
15      if (scanf("%d", &way) == 1)
16      {
17          switch (way)
18          {
19          case 0:
20              system("cls");
21              break;
22          case 1:
23              system("cls");
24              c_multiples_inp();
25              c_multiples_ui();
26              break;
27          case 2:
28              system("cls");
29              c_multiples_help();
30              c_multiples_ui();
31              break;
32          case 9:
33              system("cls");
34              ui();
35              break;
36          default:
37              system("cls");
38              puts("Error! Invalid number.\n");
39              c_multiples_ui();
40              break;
41          }
42      }
43      else
44      {
45          system("cls");
46          puts("Error. Wrong input data type.\n");
47          c_multiples_ui();
48      }
49  }
50
51  void c_multiples_inp()
52  {
53      int i, numbers[3];
54      puts("Type your numbers:\n");
55      for (i=0; i<3; i++)
56      {
57          scanf("%d", &numbers[i]);
58      }
59      printf("\n");
60      c_multiples_solution(numbers);
61  }
62
63  void c_multiples_solution(int *numbers)
```

```
64 {
65     puts("Result:");
66     c_calc_multiples(numbers);
67     printf("\n\n");
68     getch();
69     system("cls");
70 }
71
72 void c_multiples_help()
73 {
74     puts("HELP:");
75 }
```

chapterЦиклы

1.3 Задание 2. Обратная запись числа

1.3.1 Задание

Записать число наоборот.

1.3.2 Теоретические сведения

Были использованы стандартные функции ввода-вывода `scanf`, `printf`, `puts` из стандартной библиотеки языка `C`, объявленные в заголовочном файле *stdio.h*.

При помощи операторов ветвления `if-else` и `switch` реализовано интерактивное подменю для более удобного взаимодействия пользователя с программой.

1.3.3 Проектирование

В ходе проектирования было решено выделить 5 функций:

1. Вычисление суммы

   ```
   int c_calc_reversed_num(int);
   ```

   Параметром функции является число типа `int`, которое вводит пользователь. Возвращается число типа `int`, которое и является обратной записью.

2. Меню с первоначальным пользовательским взаимодействием

   ```
   void c_reversed_num_ui();
   ```

   Пользователю предлагается выбрать консольный ввод, вызов справки, возварт к главное меню или завершение программы.

3. Основное пользовательское взаимодействие

   ```
   void c_reversed_num_inp();
   ```

   Пользователю предлагается ввести число, после чего вызывается функция для решения задачи и вывода результата в консоль.

4. Решение задачи и вывода результата в консоль

   ```
   void c_reversed_num_solution(int);
   ```

В качестве параметра передаётся число типа `int`. Вызывается функция вычисления обратного числа. Результат выводится в консоль.

5. Вспомогательная информация

   `void c_reversed_num_help();`

   Вывод справки в консоль.

1.3.4 Описание тестового стенда и методики тестирования

Среда разработки QtCreator 3.5.1, компилятор MinGW (x86 64 bit), операционная система Windows 10 64 bit. В процессе выполнения задания производилось ручное тестирование. Модульное тестирование реализовано при помощи фреймворка QtTest.

1.3.5 Тестирование

Таблица 1.3: Тестирование обратного числа

Введённое число	Обратное число	Тип теста	Результат
1234	4321	Модульный	Успешно
4321	1234	Ручной	Успешно

Все тесты пройдены успешно.

1.3.6 Выводы

При выполнении задания были закреплены навыки в работе с основными конструкциями языка `C` и получен опыт в организации многофайлового проекта и создании модульных тестов.

Данная задача также была переписана на языке `C++` с использованием объектно-ориентироанного проектироания.

1.3.7 Листинги

c_reversed_num.h

```
#ifndef C_REVERSED_NUM
#define C_REVERSED_NUM

#ifdef __cplusplus

extern "C" {

#endif

int c_calc_reversed_num(int);

#ifdef __cplusplus
}
#endif

#endif // C_REVERSED_NUM
```

c_reversed_num.c

```
#include"c_reversed_num.h"

int c_calc_reversed_num(int num)
{
    int counted=0, temp=0;
    while(num!=0)
    {
        temp=num%10;
        num=num/10;
        counted=counted*10+temp;
    }
    return counted;
}
```

c_reversed_num_ui.h

```
#ifndef C_REVERSION_UI
#define C_REVERSION_UI

#include "c_reversed_num.h"

void c_reversed_num_ui();
void c_reversed_num_inp();
void c_reversed_num_solution(int);
void c_reversed_num_help();

#endif // C_REVERSION_UI
```

c_reversed_num_ui.c

```c
#include "c_main.h"
#include "c_reversed_num.h"
#include "c_reversed_num_ui.h"

void c_reversed_num_ui()
{
    int way;
    puts("Reversion of given number task");
    puts("Choose option:");
    puts("1. Input number from console");
    puts("2. Help");
    puts("9. Back to main menu");
    puts("0. Exit");
    printf("Your choice: ");
    if (scanf("%d", &way) == 1)
    {
        switch (way)
        {
        case 0:
            system("cls");
            break;
        case 1:
            system("cls");
            c_reversed_num_inp();
            c_reversed_num_ui();
            break;
        case 2:
            system("cls");
            c_reversed_num_help();
            c_reversed_num_ui();
            break;
        case 9:
            system("cls");
            ui();
            break;
        default:
            system("cls");
            puts("Error. There's no such option.\n");
            c_reversed_num_ui();
            break;
        }
    }
    else
    {
        system("cls");
        puts("Error. Wrong input data type.\n");
        c_reversed_num_ui();
```

```
48      }
49  }
50
51  void c_reversed_num_inp()
52  {
53      int num_t3;
54      puts("Type your number: ");
55      scanf("%d", &num_t3);
56      c_reversed_num_solution(num_t3);
57  }
58
59  void c_reversed_num_solution(int num_t3)
60  {
61      int result_t3 = c_calc_reversed_num(num_t3);
62      printf("Result: %d\n\n", result_t3);
63      getch();
64      system("cls");
65  }
66
67  void c_reversed_num_help()
68  {
69      puts("HELP:");
70  }
```

Глава 2

Массивы

2.1 Задание 3. Поворот матрицы

2.1.1 Задание

Содержимое квадратной матрицы A(n,n) повернуть на 90 градусов по часовой
стрелке вокруг центра матрицы.

2.1.2 Теоретические сведения

Были использованы стандартные функции ввода-вывода `scanf`, `printf`, `puts`, `fopen`, `fclose`, `fscanf`, `fprintf` из стандартной библиотеки языка `C`, объявленные в заголовочном файле *stdio.h*.

При помощи операторов ветвления `if-else` и `switch` реализовано интерактивное подменю для более удобного взаимодействия пользователя с программой.

Так же использовались функции для работы с памятью `malloc` и `free`, объявленные в заголовочном файле *stdlib.h*.

2.1.3 Проектирование

В ходе проектирования было решено выделить 7 основных функций:

1. Поворот

   ```
   void c_calc_matrix_turn(int size_of_matrix, int **matrix)
   ```

   Параметрами функции являются: число типа `int`, которое вводит пользователь(размер квадратной матрицы) и двумерный массив типа `int**`(квадратная матрица).

2. Меню с первоначальным пользовательским взаимодействием

   `void c_matrix_turn_ui();`

   Пользователю предлагается выбрать консольный ввод, файловый ввод, вызов справки, возварт к главное меню или завершение программы.

3. Консольное взаимодействие с пользователем

   `void c_matrix_turn_cinp();`

   Пользователю предлагается ввести размер матрицы и саму матрицу, после чего вызывается функция для решения задачи и вывода результата в консоль.

4. Файловое взаимодействие с пользователем

   `void c_matrix_turn_finp();`

   Пользователю предлагается ввести имя файла, откуда программа должна получить исходные данные, после чего вызывается функция для решения задачи и вывода результата в файл.

5. Решение задачи и вывода результата в консоль

   `void c_matrix_turn_csolution(int, int**);`

   Параметрами функции являются: число типа `int`, которое вводит пользователь(размер квадратной матрицы) и двумерный массив типа `int**`(квадратная матрица). Вывод производится в консоль.

6. Решение задачи и вывода результата в файл

   `void c_matrix_turn_fsolution(int, int**);`

   Параметрами функции являются: число типа `int`, которое вводит пользователь(размер квадратной матрицы) и двумерный массив типа `int**`(квадратная матрица). Вывод производится в указанный файл результата.

7. Вспомогательная информация

   `void c_matrix_turn_help();`

   Вывод справки в консоль.

2.1.4 Описание тестового стенда и методики тестирования

Среда разработки QtCreator 3.5.1, компилятор MinGW (x86 64 bit), операционная система Windows 10 64 bit. В процессе выполнения задания производилось ручное тестирование. Модульное тестирование реализовано при помощи фреймворка QtTest.

Все тесты пройдены успешно.

2.1.5 Выводы

При выполнении задания были закреплены навыки в работе с массивами и получен опыт в организации многофайлового проекта и создании модульных тестов.

Данная задача также была переписана на языке `C++` с использованием объектно-ориентироанного проектироания.

2.1.6 Листинги

c_matrix_turn.h

```
#ifndef C_MATRIX_TURN
#define C_MATRIX_TURN

#ifdef __cplusplus

extern "C" {

#endif

void c_calc_matrix_turn(int , int**);

#ifdef __cplusplus
}
#endif

#endif // C_MATRIX_TURN
```

c_matrix_turn.c

```
#include "c_matrix_turn.h"
#include "malloc.h"

void c_calc_matrix_turn(int size_of_matrix, int **matrix)
{
    int i, j;
    int **matrix_temp;
    matrix_temp = (int**)malloc(size_of_matrix*sizeof(int*));
    for(i=0; i<size_of_matrix; i++)
    {
        matrix_temp[i] = (int*)malloc(size_of_matrix*sizeof(
           int));
    }
    for(i=0; i<size_of_matrix; i++)
    {
        for(j=0; j<size_of_matrix; j++)
        {
            matrix_temp[i][j]=matrix[size_of_matrix-1-j][i];
        }
    }
    for(i=0; i<size_of_matrix; i++)
    {
        for(j=0; j<size_of_matrix; j++)
        {
            matrix[i][j]=matrix_temp[i][j];
        }
    }
```

```
    for(i=0; i<size_of_matrix; i++)
    {
        free(matrix_temp[i]);
    }
    free(matrix_temp);
}
```

c_matrix_turn_ui.h

```
#ifndef C_MATRIX_UI
#define C_MATRIX_UI

#include "c_matrix_turn.h"

void c_matrix_turn_ui();
void c_matrix_turn_cinp();
void c_matrix_turn_csolution(int, int**);
void c_matrix_turn_finp();
void c_matrix_turn_inp(int, int**, FILE*);
void c_matrix_turn_fsolution(int, int**);
void c_matrix_turn_out(int, int**, FILE*);
void c_matrix_turn_help();

#endif // C_MATRIX_UI
```

c_matrix_turn_ui.c

```
#include "c_main.h"
#include "c_matrix_turn.h"
#include "c_matrix_turn_ui.h"

void c_matrix_turn_ui()
{
    int num;
    puts("Matrix turn task");
    puts("Choose option:");
    puts("1. Input matrix from console");
    puts("2. Input matrix from file");
    puts("3. Help");
    puts("9. Back to main menu");
    puts("0. Exit");
    printf("Your choice: ");
    if (scanf("%d", &num) == 1)
    {
        switch (num)
        {
        case 0:
            system("cls");
            break;
        case 1:
```

```
24              system("cls");
25              c_matrix_turn_cinp();
26              c_matrix_turn_ui();
27              break;
28          case 2:
29              system("cls");
30              c_matrix_turn_finp();
31              c_matrix_turn_ui();
32              break;
33          case 3:
34              system("cls");
35              c_matrix_turn_help();
36              c_matrix_turn_ui();
37              break;
38          case 9:
39              system("cls");
40              ui();
41              break;
42          default:
43              system("cls");
44              puts("Error! Invalid number.\n");
45              c_matrix_turn_ui();
46              break;
47          }
48      }
49      else
50      {
51          system("cls");
52          puts("Error! Input a number.\n");
53          c_matrix_turn_ui();
54      }
55  }
56
57  void c_matrix_turn_cinp()
58  {
59      int size_of_matrix;
60
61      printf("Type size of square matrix: ");
62      scanf("%d", &size_of_matrix);
63      printf("\n");
64
65      int **matrix;
66      int i, j;
67      matrix = (int**)malloc(size_of_matrix*sizeof(int*));
68      for(i=0; i<size_of_matrix; i++)
69      {
70          matrix[i] = (int*)malloc(size_of_matrix*sizeof(int));
71      }
72
```

```
73      puts("Type matrix numbers: ");
74      for(i=0; i<size_of_matrix; i++)
75          for(j=0; j<size_of_matrix; j++)
76          {
77              scanf("%d", &matrix[i][j]);
78          }
79
80      c_matrix_turn_csolution(size_of_matrix, matrix);
81
82      for(i=0; i<size_of_matrix; i++)
83      {
84          free(matrix[i]);
85      }
86      free(matrix);
87  }
88
89  void c_matrix_turn_csolution(int size_of_matrix, int **matrix
        )
90  {
91      int i, j;
92
93      c_calc_matrix_turn(size_of_matrix, matrix);
94
95      puts("Result: ");
96      for(i=0; i<size_of_matrix; i++)
97      {
98          for(j=0; j<size_of_matrix; j++)
99          {
100             printf("%5d", matrix[i][j]);
101         }
102         printf("\n");
103     }
104
105     printf("\n");
106     getch();
107     system("cls");
108 }
109
110 void c_matrix_turn_finp()
111 {
112     int size_of_matrix, **matrix;
113
114     printf("Type size of square matrix: ");
115     scanf("%d", &size_of_matrix);
116     printf("\n");
117     printf("Type the input file name\n"
118            "(or 0 for default one): ");
119     char file_i[20];
120     scanf("%s", file_i);
```

```
121     printf("\n");
122     if (file_i[0] == '0')
123         strcpy(file_i, "input_m.txt");
124     FILE * finp = fopen(file_i, "r");
125     if (!finp)
126     {
127         system("cls");
128         puts("Error. Input file can't be opened.\n");
129         c_matrix_turn_ui();
130     }
131
132     c_matrix_turn_inp(size_of_matrix, matrix, finp);
133
134     fclose(finp);
135 }
136
137 void c_matrix_turn_inp(int size_of_matrix, int **matrix, FILE
        *finp)
138 {
139     int i, j;
140
141     matrix = (int**)malloc(size_of_matrix*sizeof(int*));
142     for(i=0; i<size_of_matrix; i++)
143     {
144         matrix[i] = (int*)malloc(size_of_matrix*sizeof(int));
145     }
146     for(i=0; i<size_of_matrix; i++)
147         for(j=0; j<size_of_matrix; j++)
148         {
149             fscanf(finp, "%d", &matrix[i][j]);
150         }
151
152     c_matrix_turn_fsolution(size_of_matrix, matrix);
153
154     for(i=0; i<size_of_matrix; i++)
155     {
156         free(matrix[i]);
157     }
158     free(matrix);
159 }
160
161 void c_matrix_turn_fsolution(int size_of_matrix, int **matrix
        )
162 {
163     printf("Type the output file name\n"
164            "(or 0 for default one): ");
165     char file_o[20];
166     scanf("%s", file_o);
167     printf("\n");
```

```
168     if (file_o[0] == '0')
169         strcpy(file_o, "output_m.txt");
170     FILE * fout = fopen(file_o, "w");
171     if (!fout)
172     {
173         system("cls");
174         puts("Error. Output file can't be opened.\n");
175         c_matrix_turn_ui();
176     }
177
178     c_matrix_turn_out(size_of_matrix, matrix, fout);
179
180     printf("Check result in %s\n\n", file_o);
181     fclose(fout);
182     getch();
183     system("cls");
184 }
185
186 void c_matrix_turn_out(int size_of_matrix, int **matrix, FILE
        *fout)
187 {
188     int i, j;
189
190     c_calc_matrix_turn(size_of_matrix, matrix);
191
192     for(i=0; i<size_of_matrix; i++)
193     {
194         for(j=0; j<size_of_matrix; j++)
195         {
196             fprintf(fout, "%5d", matrix[i][j]);
197         }
198         fprintf(fout, "\n");
199     }
200 }
201
202 void c_matrix_turn_help()
203 {
204     puts("HELP:");
205 }
```

Глава 3

Строки

3.1 Задание 4. Поиск ключей

3.1.1 Задание

В заданном тексте подсчитать частоту использования каждого буквосочетания, слова и
словосочетания из заданного списка..

3.1.2 Теоретические сведения

Были использованы стандартные функции ввода-вывода `scanf`, `printf`, `puts`, `fopen`, `fclose`, `fscanf`, `fprintf` из стандартной библиотеки языка `C`, объявленные в заголовочном файле *stdio.h*.

При помощи операторов ветвления `if-else` и `switch` реализовано интерактивное подменю для более удобного взаимодействия пользователя с программой.

Для поиска ключей использовалась функция `strstr`.

Так же использовались функции для работы с памятью `malloc` и `free`, объявленные в заголовочном файле *stdlib.h*.

3.1.3 Проектирование

В ходе проектирования было решено выделить 7 основных функций:

1. Поиск ключей

   ```
   void c_calc_keys_in_text(char**, char**, int, int, int*);
   ```

   Параметрами функции являются: 2 массива строк типа `char**`(текст и ключевые слова), 2 числа типа `int`(количество строк и ключей

соответственно) и одномерный массив типа `int*`(количество вхождений).

2. Меню с первоначальным пользовательским взаимодействием

   `void c_keys_in_text_ui();`

   Пользователю предлагается выбрать консольный ввод, файловый ввод, вызов справки, возварт к главное меню или завершение программы.

3. Консольное взаимодействие с пользователем

   `void c_keys_in_text_cinp();`

   Пользователю предлагается ввести текст и ключевые слова, после чего вызывается функция для решения задачи и вывода результата в консоль.

4. Файловое взаимодействие с пользователем

   `void c_keys_in_text_finp();`

   Пользователю предлагается ввести имя файла, откуда программа должна получить исходные данные, после чего вызывается функция для решения задачи и вывода результата в файл.

5. Решение задачи и вывода результата в консоль

   `void c_keys_in_text_csolution(char**, char**, int, int);`

   Параметрами функции являются: 2 массива строк типа `char**`(текст и ключевые слова), 2 числа типа `int`(количество строк и ключей соответственно). Вывод производится в консоль.

6. Решение задачи и вывода результата в файл

   `void c_keys_in_text_fsolution(char**, char**, int, int);`

   Параметрами функции являются: 2 массива строк типа `char**`(текст и ключевые слова), 2 числа типа `int`(количество строк и ключей соответственно). Вывод производится в указанный файл результата.

7. Вспомогательная информация

   `void c_keys_in_text_help();`

   Вывод справки в консоль.

3.1.4 Описание тестового стенда и методики тестирования

Среда разработки QtCreator 3.5.1, компилятор MinGW (x86 64 bit), операционная система Windows 10 64 bit. В процессе выполнения задания производилось ручное тестирование. Модульное тестирование реализовано при помощи фреймворка QtTest.

Все тесты пройдены успешно.

3.1.5 Выводы

При выполнении задания были закреплены навыки в работе со строками и получен опыт в организации многофайлового проекта и создании модульных тестов.

Данная задача также была переписана на языке C++ с использованием объектно-ориентироанного проектироания.

3.1.6 Листинги

c_keys_in_text.h

```
#ifndef C_KEYS_IN_TEXT
#define C_KEYS_IN_TEXT

#ifdef __cplusplus

extern "C" {

#endif

void c_calc_keys_in_text(char**, char**, int, int, int*);

#ifdef __cplusplus
}
#endif

#endif // C_KEYS_IN_TEXT
```

c_keys_in_text.c

```
#include "c_keys_in_text.h"
#include "string.h"

void c_calc_keys_in_text(char **strings, char **keys, int
    num_of_strings, int num_of_keys, int *result)
{
     int i, j, n;
     char * match;
     for (i=0; i<num_of_keys; i++)
     {
          n=0;
          for (j=0; j<num_of_strings; j++)
          {
               match = strstr (strings[j], keys[i]);
               while (match != NULL)
               {
                    match = strstr (match+1,keys[i]);
                    n++;
               }
          }
          result[i]=n;
     }
}
```

c_keys_in_text_ui.h

```
#ifndef C_TEXT_UI
#define C_TEXT_UI
```

```
3
4  #include "c_keys_in_text.h"
5
6  void c_keys_in_text_ui();
7  void c_keys_in_text_cinp();
8  void c_keys_in_text_csolution(char**, char**, int, int);
9  void c_keys_in_text_finp();
10 void c_keys_in_text_inp(FILE *, FILE *);
11 void c_keys_in_text_fsolution(char**, char**, int, int);
12 void c_keys_in_text_out(char**, char**, int, int, FILE*);
13 void c_keys_in_text_help();
14
15 #endif // C_TEXT_UI
```

c_keys_in_text_ui.c

```
1  #include "c_main.h"
2  #include "c_keys_in_text.h"
3  #include "c_keys_in_text_ui.h"
4
5  void c_keys_in_text_ui()
6  {
7      int way;
8      puts("Search of text constructs task");
9      puts("Choose option:");
10     puts("1. Input text and contstructs from console");
11     puts("2. Input text and contstructs from file");
12     puts("3. Help");
13     puts("9. Back to main menu");
14     puts("0. Exit");
15     printf("Your choice: ");
16     if (scanf("%d", &way) == 1)
17     {
18         switch (way)
19         {
20         case 0:
21             system("cls");
22             break;
23         case 1:
24             system("cls");
25             c_keys_in_text_cinp();
26             c_keys_in_text_ui();
27             break;
28         case 2:
29             system("cls");
30             c_keys_in_text_finp();
31             c_keys_in_text_ui();
32             break;
33         case 3:
34             system("cls");
```

```
35              c_keys_in_text_help();
36              c_keys_in_text_ui();
37              break;
38          case 9:
39              system("cls");
40              ui();
41              break;
42          default:
43              system("cls");
44              puts("Error. There's no such option.\n");
45              c_keys_in_text_ui();
46              break;
47          }
48      }
49      else
50      {
51          system("cls");
52          puts("Error. Wrong input data type.\n");
53          c_keys_in_text_ui();
54      }
55  }
56
57  void c_keys_in_text_cinp()
58  {
59      system("cls");
60
61      int max = 100, len= 255, num_of_strings, num_of_keys,
            temp, i;
62
63      puts ("Type text(100 strings max)");
64      scanf("%*c");
65      char strings[max][len];
66      for (num_of_strings=0; num_of_strings<max; num_of_strings
            ++)
67      {
68          printf ("%d: ", num_of_strings);
69          gets(strings[num_of_strings]);
70          if(!*strings[num_of_strings]) break;
71      }
72
73      temp=num_of_strings;
74
75      char **a_strings;
76      a_strings = (char**)malloc(temp*sizeof(char*));
77      for(num_of_strings=0; num_of_strings<temp; num_of_strings
            ++)
78      {
79          a_strings[num_of_strings] = (char*)malloc(strlen(
                strings[num_of_strings]));
```

```
80          a_strings[num_of_strings] = strings[num_of_strings];
81      }
82
83      system("cls");
84
85      puts ("Type keywords(100 words max)");
86      char keys[max][len];
87      for (num_of_keys=0; num_of_keys<max; num_of_keys++)
88      {
89          printf ("%d: ", num_of_keys);
90          gets(keys[num_of_keys]);
91          if(!*keys[num_of_keys]) break;
92      }
93
94      temp=num_of_keys;
95
96      char **a_keys;
97      a_keys = (char**)malloc(temp*sizeof(char*));
98      for(num_of_keys=0; num_of_keys<temp; num_of_keys++)
99      {
100         a_keys[num_of_keys] = (char*)malloc(strlen(keys[
                num_of_keys]));
101         a_keys[num_of_keys] = keys[num_of_keys];
102     }
103
104     system("cls");
105
106     c_keys_in_text_csolution(a_strings, a_keys,
            num_of_strings, num_of_keys);
107
108     for(i=0; i<num_of_strings; i++)
109     {
110         free(a_strings[i]);
111     }
112     free(a_strings);
113
114     for(i=0; i<num_of_keys; i++)
115     {
116         free(a_keys[i]);
117     }
118     free(a_keys);
119 }
120
121 void c_keys_in_text_csolution(char **strings, char **keys,
        int num_of_strings, int num_of_keys)
122 {
123     int i;
124
125     for (i=0; i<num_of_strings; i++)
```

```
126     {
127         printf("%s\n", strings[i]);
128     }
129
130     printf ("\n");
131
132     int *result;
133     result = (int*)malloc(num_of_keys*sizeof(int));
134
135     c_calc_keys_in_text(strings, keys, num_of_strings,
            num_of_keys, result);
136
137     for (i=0; i<num_of_keys; i++)
138     {
139         printf("Number of matches found for '%s'",keys[i]);
140         printf(": %d\n",result[i]);
141     }
142
143     free(result);
144     getch();
145     system("cls");
146 }
147
148 void c_keys_in_text_finp()
149 {
150     system("cls");
151
152     printf("Type the input file name with text\n"
153            "(or 0 for default one): ");
154     char file_i1[20];
155     scanf("%s", file_i1);
156     printf("\n");
157     if (file_i1[0] == '0')
158         strcpy(file_i1, "input_t.txt");
159     FILE * finp1 = fopen(file_i1, "r");
160     if (!finp1)
161     {
162         system("cls");
163         puts("Error. Input file can't be opened.\n");
164         c_matrix_turn_ui();
165     }
166
167     system("cls");
168
169     printf("Type the input file name with constructs\n"
170            "(or 0 for default one): ");
171     char file_i2[20];
172     scanf("%s", file_i2);
173     printf("\n");
```

```
174     if (file_i2[0] == '0')
175         strcpy(file_i2, "input_k.txt");
176     FILE * finp2 = fopen(file_i2, "r");
177     if (!finp2)
178     {
179         system("cls");
180         puts("Error. Input file can't be opened.\n");
181         c_matrix_turn_ui();
182     }
183
184     system("cls");
185
186     c_keys_in_text_inp(finp1, finp2);
187     fclose(finp1);
188     fclose(finp2);
189 }
190
191 void c_keys_in_text_inp(FILE *finp1, FILE *finp2)
192 {
193     int max = 100, len= 255, num_of_strings, num_of_keys, i,
        temp;
194
195     char strings[max][len];
196     for (num_of_strings=0; num_of_strings<max; num_of_strings
        ++)
197     {
198         fgets(strings[num_of_strings], len, finp1);
199         if(!*strings[num_of_strings]) break;
200         if ( strings[num_of_strings][strlen(strings[
            num_of_strings])- 1] == '\n') strings[
            num_of_strings][strlen(strings[num_of_strings])-
            1]='\0';
201     }
202
203     temp=num_of_strings;
204
205     char **a_strings;
206     a_strings = (char**)malloc(temp*sizeof(char*));
207     for(num_of_strings=0; num_of_strings<temp; num_of_strings
        ++)
208     {
209         a_strings[num_of_strings] = (char*)malloc(strlen(
            strings[num_of_strings]));
210         a_strings[num_of_strings] = strings[num_of_strings];
211     }
212
213     char keys[max][len];
214     for (num_of_keys=0; num_of_keys<max; num_of_keys++)
215     {
```

```
216         fgets(keys[num_of_keys], len, finp2);
217         if(!*keys[num_of_keys]) break;
218         if ( keys[num_of_keys][strlen(keys[num_of_keys])- 1]
               == '\n') keys[num_of_keys][strlen(keys[num_of_keys
               ])- 1]='\0';
219     }
220
221     temp=num_of_keys;
222
223     char **a_keys;
224     a_keys = (char**)malloc(temp*sizeof(char*));
225     for(num_of_keys=0; num_of_keys<temp; num_of_keys++)
226     {
227         a_keys[num_of_keys] = (char*)malloc(strlen(keys[
               num_of_keys]));
228         a_keys[num_of_keys] = keys[num_of_keys];
229     }
230
231     c_keys_in_text_fsolution(a_strings, a_keys,
           num_of_strings, num_of_keys);
232
233     for(i=0; i<num_of_strings; i++)
234     {
235         free(a_strings[i]);
236     }
237     free(a_strings);
238     for(i=0; i<num_of_keys; i++)
239     {
240         free(a_keys[i]);
241     }
242     free(a_keys);
243 }
244
245 void c_keys_in_text_fsolution(char **strings, char **keys,
       int num_of_strings, int num_of_keys)
246 {
247     printf("Type the output file name\n"
248            "(or 0 for default one): ");
249     char file_o[20];
250     scanf("%s", file_o);
251     printf("\n");
252     if (file_o[0] == '0')
253         strcpy(file_o, "output_t.txt");
254     FILE * fout = fopen(file_o, "w");
255     if (!fout)
256     {
257         system("cls");
258         puts("Error. Output file can't be opened.\n");
259         c_matrix_turn_ui();
```

```
260     }
261
262     c_keys_in_text_out(strings, keys, num_of_strings,
            num_of_keys, fout);
263
264     printf("Check result in %s", file_o);
265
266     fclose(fout);
267     getch();
268     system("cls");
269 }
270
271 void c_keys_in_text_out(char **strings, char **keys, int
        num_of_strings, int num_of_keys, FILE *fout)
272 {
273     int i;
274
275     for (i=0; i<num_of_strings; i++)
276     {
277         fprintf(fout, "%s\n", strings[i]);
278     }
279
280     printf ("\n");
281
282     int *result;
283     result = (int*)malloc(num_of_keys*sizeof(int));
284     c_calc_keys_in_text(strings, keys, num_of_strings,
            num_of_keys, result);
285     fprintf(fout, "\n");
286     for (i=0; i<num_of_keys; i++)
287     {
288         fprintf(fout, "Number of matches found for '%s'",keys
                [i]);
289         fprintf(fout, ": %d\n",result[i]);
290     }
291
292     free(result);
293 }
294
295 void c_keys_in_text_help()
296 {
297     puts("HELP:");
298 }
```

Глава 4

Инкапсуляция

4.1 Задание 5. БОЛЬШОЕ ЦЕЛОЕ ЧИСЛО

4.1.1 Задание

Реализовать класс БОЛЬШОЕ ЦЕЛОЕ ЧИСЛО (максимальное значение неограничено). Требуемые

методы: конструктор, деструктор, копирование, сложение, вычитание, умножение, преобразование к

типу int.

4.1.2 Теоретические сведения

Для реализации данной задачи были использованы библиотечные потоки ввода-вывода `STL`, такие как `cin` и `cout`, объявленные в заголовочном файле *iostream*.

При помощи цикла `for` и происходит иттерирование по массиву. Работа с памятью происходит при помощи операторов `new` и `delete`.

Для создания данного класса использован `vector`, т.к. количество разрядов бцч не ограничено. Все действия выполнялись порахрядно, наподобии вычисления в столбик.

4.1.3 Проектирование

В ходе проектирования для представления бцч было решено выделить класс `UnlimitedInt` .

Класс `UnlimitedInt` было решено спроектировать следующим образом:

1. Поле, содержащие бцч и его размер

   `vector<int> num; int sizeOfNum;`

2. Конструкторы:

   `UnlimitedInt();`

   Конструктор по умолчанию.

   `UnlimitedInt();`

   Конструктор. Принимает на вход размер числа. И заполняет его разряды 0-ми.

   `UnlimitedInt(int);`

   Конструктор копирования инициализирует создаваемый объект с помощью другого объекта этого класса.

   `UnlimitedInt(const UnlimitedInt& obj);`

3. Функции: Вывод числа `void showNum() const;` Получение размера числа `int getSize() const;` Установка размера числа `void setSize(int);` Конвертация в целочисленный тип `int toInt();`

4. Перегруженные операторы:

   Члены класса:

   `UnlimitedInt & operator=(UnlimitedInt &); int &operator[](int);`
   `UnlimitedInt & operator+=(UnlimitedInt &); UnlimitedInt & operator-=(Unlimit`
   `UnlimitedInt & operator*=(UnlimitedInt &);`

   Не являются членами класса:

   `UnlimitedInt & operator+(UnlimitedInt &, UnlimitedInt &); UnlimitedInt & ope`
   `UnlimitedInt & operator*(UnlimitedInt &, UnlimitedInt &);`

5. Деструктор

   `~UnlimitedInt();`

   В деструкторе происходит удалание выделенной памяти под хранение числа;+.

4.1.4 Тестирование

Все тесты пройдены успешно.

Таблица 4.1: Тестирование БЦЧ(в данный момент в обновлённой версии наблюдаются проблемы с оператором =, тесты приведены для старой версии)

Действие	Введённое число1	Введённое число2	Результат	Тип теста	Результат
Сложение	654321	123456	777777	Модульный	Успешно
Вычитание	654321	123456	530865	Модульный	Успешно
Умножение	4321	1234	5332114	Модульный	Успешно

4.1.5 Выводы

При выполнении задания были усвоены навыки проектирования и создания классов и получен опыт в организации многофайлового проекта.

4.1.6 Листинги

UnlimitedInt.h

```
#ifndef INT_UNLIM_UI_H
#define INT_UNLIM_UI_H

#include <iostream>


using namespace std;

class UnlimitedInt
{
private:
    int *num;
    int sizeOfNum;

public:
    UnlimitedInt();
    UnlimitedInt(int);
    UnlimitedInt(const UnlimitedInt& obj);
    ~UnlimitedInt();
    void showNum() const;
    int getSize() const;
    void setSize(int);
    UnlimitedInt & operator=(UnlimitedInt &);
    int &operator[](int);
    UnlimitedInt & operator+=(UnlimitedInt &);
    UnlimitedInt & operator-=(UnlimitedInt &);
    UnlimitedInt & operator*=(UnlimitedInt &);
    int toInt();
};

UnlimitedInt & operator+(UnlimitedInt &, UnlimitedInt &);
UnlimitedInt & operator-(UnlimitedInt &, UnlimitedInt &);
UnlimitedInt & operator*(UnlimitedInt &, UnlimitedInt &);

#endif // INT_UNLIM_UI_H
```

UnlimitedInt.cpp

```
#include "UnlimitedInt.h"
#include "iostream"

UnlimitedInt::UnlimitedInt()
{
    sizeOfNum=0;
    num=0;
}
```

```
 9
10 UnlimitedInt::UnlimitedInt(int size)
11 {
12     sizeOfNum = size;
13     num = new int [sizeOfNum];
14     for(int i = 0; i < sizeOfNum; i++)
15     {
16         num[i] = 0;
17     }
18 }
19
20 UnlimitedInt::UnlimitedInt(const UnlimitedInt& obj)
21 {
22     sizeOfNum = obj.getSize();
23     num = new int [sizeOfNum];
24     for (int i = 0; i < sizeOfNum; i++)
25     {
26         num[i] = obj.num[i];
27     }
28 }
29
30
31 UnlimitedInt::~UnlimitedInt()
32 {
33     delete[] num;
34 }
35
36 void UnlimitedInt::showNum() const
37 {
38     for(int i = 0; i < sizeOfNum; i++)
39     {
40         cout << num[i];
41     }
42 }
43
44 int UnlimitedInt::getSize() const
45 {
46     return sizeOfNum;
47 }
48
49 void UnlimitedInt::setSize(int size)
50 {
51     sizeOfNum = size;
52     num = new int [sizeOfNum];
53     for(int i = 0; i < sizeOfNum; i++)
54     {
55         num[i] = 0;
56     }
57 }
```

```
58
59 int & UnlimitedInt::operator[](int j)
60     {
61         return num[j];
62     }
63
64 UnlimitedInt & UnlimitedInt::operator=(UnlimitedInt &num2)
65 {
66     if(this!= &num2)
67     {
68         delete [] num;
69
70         sizeOfNum = num2.sizeOfNum;
71
72         num = new int [sizeOfNum];
73         for (int i = 0; i < sizeOfNum; i++)
74         {
75             num[i] = num2.num[i];
76         }
77      }
78
79     return *this;
80 }
81
82 UnlimitedInt & UnlimitedInt::operator+=(UnlimitedInt &num2)
83 {
84     int sizeOfMin;
85     if(sizeOfNum<num2.sizeOfNum)
86     {
87         sizeOfMin=sizeOfNum;
88     }
89     else
90     {
91         sizeOfMin=num2.sizeOfNum;
92     }
93     for(int i = 0; i < sizeOfMin; i++)
94     {
95         num[sizeOfNum-1-i]+=num2[num2.sizeOfNum-1-i];
96     }
97
98     for(int i = sizeOfNum-1; i > 0; i--)
99     {
100        if(num[i]>=10)
101        {
102            int overflow=num[i]/10;
103            num[i]=num[i]%10;
104            num[i-1]=num[i-1]+overflow;
105        }
106    }
```

```
107     return *this;
108 }
109
110 UnlimitedInt & operator+(UnlimitedInt &num1, UnlimitedInt &
        num2)
111 {
112     UnlimitedInt temp = num1;
113     return temp+=num2;
114 }
115
116 UnlimitedInt & UnlimitedInt::operator-=(UnlimitedInt &num2)
117 {
118     int sizeOfMin;
119     int trigger;
120     if(sizeOfNum<num2.sizeOfNum)
121     {
122         trigger=1;
123     }
124     else if(sizeOfNum>num2.sizeOfNum)
125     {
126         trigger=2;
127     }
128     else
129     {
130         for(int i=0; i<sizeOfNum; i++)
131         {
132             if (num[i] < num2.num[i])
133             {
134                 trigger=3;
135                 break;
136             }
137             else
138             {
139                 trigger=1;
140             }
141         }
142
143     }
144     if(trigger==1)
145     {
146         sizeOfMin=sizeOfNum;
147     }
148     else
149     {
150         sizeOfMin=num2.sizeOfNum;
151     }
152
153     for(int i = 0; i < sizeOfMin; i++)
154     {
```

```
155         num[sizeOfNum-1-i]-=num2[num2.sizeOfNum-1-i];
156     }
157
158     for(int i = sizeOfNum-1; i > 0; i--)
159     {
160         if(num[i]<0)
161         {
162             num[i]=num[i]+10;
163             num[i-1]--;
164         }
165     }
166
167     if (trigger==3)
168     {
169         num[0]*=-1;
170     }
171     return *this;
172 }
173
174 UnlimitedInt & operator-(UnlimitedInt &num1, UnlimitedInt &
        num2)
175 {
176     UnlimitedInt temp = num1;
177     return temp-=num2;
178 }
179
180 UnlimitedInt & UnlimitedInt::operator*=(UnlimitedInt &num2)
181 {
182     int temp_size=sizeOfNum;
183     setSize(sizeOfNum+num2.sizeOfNum-1);
184     int** mult_calc = new int* [num2.sizeOfNum];
185         for (int i = 0; i < sizeOfNum; i++)
186             mult_calc[i] = new int [sizeOfNum];
187
188     int k=0;
189     int l=sizeOfNum-1;
190     for(int i = num2.sizeOfNum-1; i >= 0; i--)
191     {
192         l=sizeOfNum-1;
193         for(int j = temp_size; j >= 0; j--)
194         {
195             mult_calc[k][l-k] = num2.num[i]*num[j];
196             l--;
197         }
198         k++;
199     }
200
201     for(int i = 0; i < num2.sizeOfNum; i++)
202     {
```

```
203         for(int j = sizeOfNum-1; j > num2.sizeOfNum-1-i; j--)
204         {
205             if(mult_calc[i][j]>=10)
206             {
207                 int overflow=mult_calc[i][j]/10;
208                 mult_calc[i][j]=mult_calc[i][j]%10;
209                 mult_calc[i][j-1]=mult_calc[i][j-1]+overflow;
210             }
211         }
212     }
213
214     for(int j = sizeOfNum-1; j >= 0; j--)
215     {
216         int sum_temp=0;
217         for(int i = 0; i < num2.sizeOfNum; i++)
218         {
219             sum_temp=sum_temp+mult_calc[i][j];
220         }
221         num[j]=sum_temp;
222     }
223
224     for (int i=0; i<num2.sizeOfNum; i++)
225     {
226         delete [] mult_calc[i];
227     }
228     delete [] mult_calc;
229
230     for(int i = sizeOfNum-1; i > 0; i--)
231     {
232         if(num[i]>=10)
233         {
234             int overflow=num[i]/10;
235             num[i]=num[i]%10;
236             num[i-1]=num[i-1]+overflow;
237         }
238     }
239
240     return *this;
241 }
242
243 UnlimitedInt & operator*(UnlimitedInt &num1, UnlimitedInt &
        num2)
244 {
245     UnlimitedInt temp = num1;
246     return temp-=num2;
247 }
248
249 int UnlimitedInt::toInt()
250 {
```

```
    int num_to_i=0;
    for (int i=0; i<sizeOfNum; i++)
    {
        num_to_i=num_to_i*10+num[i];
        if(i>=5)
        {
            break;
        }
    }
    cout<<endl;
    return num_to_i;
}
```

UnlimitedIntDemo.h

```
#ifndef CPP_INT_UNLIM_INP
#define CPP_INT_UNLIM_INP

#include "UnlimitedInt.h"
#include <string>

using std::cin;
using std::cout;
using std::endl;

class UnlimitedIntDemo
{
private:
    void inputNums();
    void checkSum();
    void checkSubt();
    void checkMult();
    void checkToInt();
    //void checkCopy(ArrayApp &) const;
    UnlimitedInt num1;
    UnlimitedInt num2;
public:
    void demo();
};

#endif // CPP_INT_UNLIM_INP
```

UnlimitedIntDemo.cpp

```
#include "UnlimitedIntDemo.h"

void UnlimitedIntDemo::demo()
{
    inputNums();
    checkSum();
```

```
7       checkSubt();
8       checkMult();
9       checkToInt();
10  }
11
12  void UnlimitedIntDemo::inputNums()
13  {
14      string s1, s2;
15      cout<< "Input 1st number"<< endl;
16      cin>>s1;
17      int size1=s1.length();
18      num1.setSize(size1);
19      for(int i=0; i<s1.length(); i++)
20      {
21          char c = s1[i];
22          if ((c>='0') && (c<='9'))
23          {
24              int n = c - '0';
25              num1[i]=n;
26          }
27      }
28      cout<< "Input 2nd number"<< endl;
29      cin>>s2;
30      int size2=s1.length();
31      num2.setSize(size2);
32      for(int i=0; i<s1.length(); i++)
33      {
34          char c = s2[i];
35          if ((c>='0') && (c<='9'))
36          {
37              int n = c - '0';
38              num2[i]=n;
39          }
40      }
41  }
42
43  void UnlimitedIntDemo::checkSum()
44  {
45      UnlimitedInt sum(1);
46      num1.showNum();
47      cout<<"+";
48      num2.showNum();
49      cout<<"=";
50      sum=num1+num2;
51      sum.showNum();
52      cout << endl;
53  }
54
55  void UnlimitedIntDemo::checkSubt()
```

```
56 {
57     UnlimitedInt subt(1);
58     num1.showNum();
59     cout<<"-";
60     num2.showNum();
61     cout<<"=";
62     subt=num1-num2;
63     subt.showNum();
64     cout << endl;
65 }
66
67 void UnlimitedIntDemo::checkMult()
68 {
69     UnlimitedInt mult(1);
70     num1.showNum();
71     cout<<"*";
72     num2.showNum();
73     cout<<"=";
74     mult=num1*num2;
75     mult.showNum();
76     cout << endl;
77 }
78
79 void UnlimitedIntDemo::checkToInt()
80 {
81     cout << "Result of conversion of num1 to int is: " <<num1
          .toInt();
82     cout << endl;
83     cout << "Result of conversion of num2 to int is: " <<num2
          .toInt();
84     cout << endl;
85 }
```

Все программы были проверенны cppcheck. Выявлены и устранены ошибки и большинство предупреждений.

Приложения

Листинги модульных тестов к заданиям с 1 по 4 включительно

```
#include <QString>
#include <QtTest>
#include "c_sum_of_digits.h"
#include "c_multiples.h"
#include "c_reversed_num.h"
#include "c_matrix_turn.h"
#include "c_keys_in_text.h"

int compare_matrix(int **matrix, int **matrix_r, int square)
{
    for (int i=0; i<square; i++)
        for (int j=0; j<square; j++)
        {
            if (matrix[i][j]!=matrix_r[i][j])
            return 1;
        }
    return 0;
}

int compare_text(int *result, int *result_r)
{
    for (int i=0; i<2; i++)
    {
        if (result[i]!=result_r[i])
        return 1;
    }
    return 0;
}

class CTestsTest : public QObject
{
    Q_OBJECT

public:
```

```
35      CTestsTest();
36
37  private Q_SLOTS:
38      void c_test_sum_of_digits();
39      void c_test_search_multiples();
40      void c_test_reversed_num();
41      void c_test_matrix_turn();
42      void c_test_keys_in_text();
43  };
44
45  CTestsTest::CTestsTest()
46  {
47  }
48
49  void CTestsTest::c_test_sum_of_digits()
50  {
51      int num = 1234;
52      QCOMPARE(c_calc_sum_of_digits(num), 10);
53  }
54
55  void CTestsTest::c_test_search_multiples()
56  {
57      int numbers[3];
58      numbers[0]=2;
59      numbers[1]=4;
60      numbers[2]=8;
61      QCOMPARE(c_calc_multiples(numbers), 3);
62  }
63
64  void CTestsTest::c_test_reversed_num()
65  {
66      int num = 1234;
67      QCOMPARE(c_calc_reversed_num(num), 4321);
68  }
69
70  void CTestsTest::c_test_matrix_turn()
71  {
72      int size_of_matrix=3;
73      int **matrix;
74      int i, j, k=0;
75      matrix = (int**)malloc(size_of_matrix*sizeof(int*));
76      for(i=0; i<size_of_matrix; i++)
77      {
78          matrix[i] = (int*)malloc(size_of_matrix*sizeof(int));
79      }
80      for(i=0; i<size_of_matrix; i++)
81          for(j=0; j<size_of_matrix; j++)
82          {
83              k++;
```

```
84              matrix[i][j]=k;
85          }
86      int **matrix_r;
87      matrix_r = (int**)malloc(size_of_matrix*sizeof(int*));
88      for(i=0; i<size_of_matrix; i++)
89      {
90          matrix_r[i] = (int*)malloc(size_of_matrix*sizeof(int)
                );
91      }
92      k=0;
93      for(i=0; i<size_of_matrix; i++)
94          for(j=0; j<size_of_matrix; j++)
95          {
96              k++;
97              matrix_r[i][j]=k+size_of_matrix*2-4*j-2*i;
98          }
99      c_calc_matrix_turn(size_of_matrix, matrix);
100     QCOMPARE(compare_matrix(matrix, matrix_r, size_of_matrix)
            , 0);
101     for(i=0; i<size_of_matrix; i++)
102     {
103         free(matrix[i]);
104         free(matrix_r[i]);
105     }
106     free(matrix);
107     free(matrix_r);
108 }
109
110 void CTestsTest::c_test_keys_in_text()
111 {
112     int num_of_strings=3, num_of_keys=2, temp, i;
113
114     temp=num_of_strings;
115
116     char strings[3][6] = {"ololo", "lol", "o"};
117
118     char **a_strings;
119     a_strings = (char**)malloc(temp*sizeof(char*));
120     for(num_of_strings=0; num_of_strings<temp; num_of_strings
            ++)
121     {
122         a_strings[num_of_strings] = (char*)malloc(strlen(
                strings[num_of_strings]));
123         a_strings[num_of_strings] = strings[num_of_strings];
124     }
125
126     temp=num_of_keys;
127
128     char keys[2][2] = {"o", "l"};
```

```
129
130     char **a_keys;
131     a_keys = (char**)malloc(temp*sizeof(char*));
132     for(num_of_keys=0; num_of_keys<temp; num_of_keys++)
133     {
134         a_keys[num_of_keys] = (char*)malloc(strlen(keys[
                num_of_keys]));
135         a_keys[num_of_keys] = keys[num_of_keys];
136     }
137
138     int *result;
139     result = (int*)malloc(num_of_keys*sizeof(int));
140
141     c_calc_keys_in_text(a_strings, a_keys, num_of_strings,
            num_of_keys, result);
142
143     int result_r[2] = {5, 4};
144
145     for (i=0; i<num_of_keys; i++)
146     QCOMPARE(compare_text(result, result_r), 0);
147
148     free(result);
149
150     for(i=0; i<num_of_strings; i++)
151     {
152         free(a_strings[i]);
153     }
154     free(a_strings);
155
156     for(i=0; i<num_of_keys; i++)
157     {
158         free(a_keys[i]);
159     }
160     free(a_keys);
161 }
162
163 QTEST_APPLESS_MAIN(CTestsTest)
164
165 #include "tst_cteststest.moc"
```

Листинги класса $cpp_sum_of_digits$

```
#ifndef CPP_SUM_OF_DIGITS
#define CPP_SUM_OF_DIGITS

#include <iostream>

using namespace std;

class SumOfDigits
{
private:
    int num;

public:
    SumOfDigits();
    SumOfDigits(int);
    int Sum();
};

#endif // CPP_SUM_OF_DIGITS
```

```
#include "SumOfDigits.h"

SumOfDigits::SumOfDigits()
{
    num = 123;
}

SumOfDigits::SumOfDigits(int number)
{
    num=number;
}

int SumOfDigits::Sum()
{
    int sum=0;
    int temp=0;
    while(num!=0)
    {
        temp=num%10;
        num=num/10;
        sum=sum+temp;
    }
    return sum;
}
```

Листинги класса $cpp_multiples$

```
#ifndef CPP_MULTIPLES_SEARCH
#define CPP_MULTIPLES_SEARCH

#include <iostream>
#include <vector>

using namespace std;

class Multiples
{
private:
    vector<int> numbers;
    vector<int> multiples;

public:
    Multiples();
    Multiples(vector<int> &);
    ~Multiples();
    vector<int> & findMultiples();
};

#endif // CPP_MULTIPLES_SEARCH
```

```
#include "Multiples.h"
#include "Exceptions.h"

Multiples::Multiples()
{
    numbers.push_back(5);
    numbers.push_back(10);
    numbers.push_back(11);
}

Multiples::~Multiples()
{
    numbers.clear();
    multiples.clear();
}

Multiples::Multiples(vector<int> &nums)
{
    for (int i=0; i<nums.size(); i++)
    {
        if (nums[i]>999)
        {
            throw OutOfRange(nums[i]);
        }
```

```
25          else
26          {
27              numbers.push_back(nums[i]);
28          }
29      }
30  }
31
32  vector<int> & Multiples::findMultiples()
33  {
34      for (int i=0; i<numbers.size(); i++)
35          for (int j=0; j<numbers.size(); j++)
36          {
37              if(i!=j && numbers[i]%numbers[j]==0)
38              {
39                  multiples.push_back(numbers[i]*1000+numbers[j
                        ]);
40              }
41          }
42      return multiples;
43  }
```

Листинги класса $cpp_reversed_num$

```
#ifndef CPP_REVERSED_NUM
#define CPP_REVERSED_NUM

#include <iostream>

using namespace std;

class ReversedNum
{
private:
    int num;

public:
    ReversedNum();
    ReversedNum(int);
    int Reversion();
};

#endif // CPP_REVERSED_NUM
```

```
#include "ReversedNum.h"

ReversedNum::ReversedNum()
{
    num=123;
}

ReversedNum::ReversedNum(int number)
{
    num=number;
}

int ReversedNum::Reversion()
{
    int reversion=0;
    int temp=0;
    while(num!=0)
    {
        temp=num%10;
        num=num/10;
        reversion=reversion*10+temp;
    }
    return reversion;
}
```

Листинги класса cpp_matrix_turn

```cpp
#ifndef CPP_MATRIX_TURN
#define CPP_MATRIX_TURN

#include <iostream>

using namespace std;

class MatrixTurn
{
private:
    int **matrix;
    int sizeOfMatrix;

public:
    MatrixTurn(int);
    MatrixTurn(int, int**);
    MatrixTurn(MatrixTurn &);
    ~MatrixTurn();
    void putNum(int , int, int);
    int getNum(int , int);
    void turnMatrix();
};

#endif // CPP_MATRIX_TURN
```

```cpp
#include "MatrixTurn.h"

MatrixTurn::MatrixTurn(int size)
{
    sizeOfMatrix=size;
    matrix = new int* [sizeOfMatrix];
    for (int i = 0; i < sizeOfMatrix; i++)
    {
        matrix[i] = new int [sizeOfMatrix];
    }

    for (int i=0; i<sizeOfMatrix; i++)
    {
        for(int j=0; j<sizeOfMatrix; j++)
        {
            matrix[i][j] = 0;
        }
    }
}

MatrixTurn::MatrixTurn(int size, int** imatrix)
{
```

```
23      sizeOfMatrix=size;
24      for(int i=0; i<sizeOfMatrix; i++)
25      {
26          for(int j=0; j<sizeOfMatrix; j++)
27          {
28              matrix[i][j]=imatrix[i][j];
29          }
30      }
31  }
32
33  MatrixTurn::MatrixTurn(MatrixTurn &obj)
34  {
35      sizeOfMatrix=obj.sizeOfMatrix;
36      for(int i=0; i<sizeOfMatrix; i++)
37      {
38          for(int j=0; j<sizeOfMatrix; j++)
39          {
40              matrix[i][j]=obj.matrix[i][j];
41          }
42      }
43  }
44
45  MatrixTurn::~MatrixTurn()
46  {
47      for (int i=0; i<sizeOfMatrix; i++)
48      {
49          delete [] matrix[i];
50      }
51      delete [] matrix;
52  }
53
54  void MatrixTurn::putNum(int i, int j, int num)
55  {
56      matrix[i][j] = num;
57  }
58
59
60  int MatrixTurn::getNum(int i, int j)
61  {
62      return matrix[i][j];
63  }
64
65
66  void MatrixTurn::turnMatrix()
67  {
68      int** matrix_turned = new int* [sizeOfMatrix];
69      {
70          for (int i = 0; i < sizeOfMatrix; i++)
71          {
```

```
72              matrix_turned[i] = new int [sizeOfMatrix];
73          }
74      }
75
76      for(int i=0; i<sizeOfMatrix; i++)
77      {
78          for(int j=0; j<sizeOfMatrix; j++)
79          {
80              matrix_turned[i][j]=matrix[sizeOfMatrix-1-j][i];
81          }
82      }
83
84      for(int i=0; i<sizeOfMatrix; i++)
85      {
86          for(int j=0; j<sizeOfMatrix; j++)
87          {
88              matrix[i][j]=matrix_turned[i][j];
89          }
90      }
91
92      for (int i=0; i<sizeOfMatrix; i++)
93      {
94          delete [] matrix_turned[i];
95      }
96      delete [] matrix_turned;
97  }
```

Листинги класса $cpp_keys_in_text$

```
#ifndef CPP_KEYS_IN_TEXT
#define CPP_KEYS_IN_TEXT

#include <iostream>
#include <vector>
#include <cstring>
#include <string>

using namespace std;

class KeysInText
{
private:
    vector<string> strings;
    vector<int> result;

public:
    KeysInText(vector<string> &);
    ~KeysInText();
    vector<int> & findKeys(vector<string> &);
};


#endif // CPP_KEYS_IN_TEXT
```

```
#include "KeysInText.h"


KeysInText::KeysInText(vector<string> &istrings)
{
    for (int i=0; i<istrings.size(); i++)
    {
        strings.push_back(istrings[i]);
    }
}

KeysInText::~KeysInText()
{

}

vector<int> & KeysInText::findKeys(vector<string> &keys)
{
    int n;
    char * match;
    for (int i=0; i<keys.size(); i++)
    {
```

```
23          n=0;
24
25          char * ckey = new char [keys[i].length()+1];
26          strcpy (ckey, keys[i].c_str());
27
28          for (int j=0; j<strings.size(); j++)
29          {
30              char * cstring = new char [strings[j].length()
                    +1];
31              strcpy (cstring, strings[j].c_str());
32
33              match = strstr (cstring, ckey);
34              while (match != NULL)
35              {
36                  match = strstr (match+1,ckey);
37                  n++;
38              }
39              delete[] cstring;
40          }
41          delete[] ckey;
42          result.push_back(n);
43      }
44      return result;
45  }
```

Листинги модульных тестов к заданиям с 1 по 4 включительно на языке C++

```
#include <QString>
#include <QtTest>

#include "UnlimitedInt.h"
#include "KeysInText.h"
#include "MatrixTurn.h"
#include "Multiples.h"
#include "ReversedNum.h"
#include "SumOfDigits.h"

class CppTestsTest : public QObject
{
    Q_OBJECT

public:
    CppTestsTest();

private Q_SLOTS:
    void testSumOfDigits();
    void testReversedNum();
    void testMultiples();
    void testMatrixTurn();
    void testKeysInText();
    /*void test_iu_sum();
    void test_iu_subt();
    void test_iu_mult();*/
};

CppTestsTest::CppTestsTest()
{
}

void CppTestsTest::testSumOfDigits()
{
    SumOfDigits num(248);
    int sum=num.Sum();
    QCOMPARE(sum, 14);
}

void CppTestsTest::testReversedNum()
{
    ReversedNum num(248);
    int reversion=num.Reversion();
    QCOMPARE(reversion, 842);
}
```

```
46
47 void CppTestsTest::testMultiples()
48 {
49     int trigger=0;
50     vector<int> numbers;
51     numbers.push_back(5);
52     numbers.push_back(10);
53     numbers.push_back(11);
54     numbers.push_back(22);
55     vector<int> multiples_test;
56     multiples_test.push_back(10005);
57     multiples_test.push_back(22011);
58     Multiples nums(numbers);
59     vector<int> multiples=nums.findMultiples();
60     for (int i=0; i<multiples.size(); i++)
61     {
62         if (multiples[i]!=multiples_test[i])
63         {
64             trigger=1;
65             break;
66         }
67     }
68     QCOMPARE(trigger, 0);
69 }
70
71 void CppTestsTest::testMatrixTurn()
72 {
73     int sizeOfMatrix=3, k=0, trigger=0;
74
75     MatrixTurn matrix(sizeOfMatrix);
76     for(int i=0; i<sizeOfMatrix; i++)
77     {
78         for(int j=0; j<sizeOfMatrix; j++)
79         {
80             k++;
81             matrix.putNum(i, j, k);
82         }
83     }
84
85     int **matrix_r = new int* [sizeOfMatrix];
86     for (int i = 0; i < sizeOfMatrix; i++)
87     {
88         matrix_r[i] = new int [sizeOfMatrix];
89     }
90
91     k=0;
92     for(int i=0; i<sizeOfMatrix; i++)
93     {
94         for(int j=0; j<sizeOfMatrix; j++)
```

```
95          {
96              k++;
97              matrix_r[i][j]=k+sizeOfMatrix*2-4*j-2*i;
98          }
99      }
100
101     matrix.turnMatrix();
102
103     for (int i=0; i<sizeOfMatrix; i++)
104     {
105         for (int j=0; j<sizeOfMatrix; j++)
106         {
107             if (matrix_r[i][j]!=matrix.getNum(i, j))
108             {
109                 trigger=1;
110                 break;
111             }
112         }
113     }
114
115     QCOMPARE(trigger, 0);
116
117     for (int i=0; i<sizeOfMatrix; i++)
118     {
119         delete [] matrix_r[i];
120     }
121     delete [] matrix_r;
122 }
123
124 void CppTestsTest::testKeysInText()
125 {
126     int trigger=0;
127     vector<string> strings;
128     strings.push_back("ololo");
129     strings.push_back("lol");
130     strings.push_back("o");
131
132     vector<string> keys;
133     keys.push_back("o");
134     keys.push_back("l");
135
136     KeysInText kit(strings);
137     vector<int> result=kit.findKeys(keys);
138
139     vector<int> r_test;
140     r_test.push_back(5);
141     r_test.push_back(4);
142
143     for (int i=0; i<r_test.size(); i++)
```

```
144     {
145         if (r_test[i]!=result[i])
146         {
147             trigger=1;
148             break;
149         }
150     }
151     QCOMPARE(trigger, 0);
152 }
153
154 /*void CppTestsTest::test_iu_sum()
155 {
156     UnlimitedInt num1(5);
157     num1[0]=1;
158     num1[1]=2;
159     num1[2]=3;
160     num1[3]=4;
161     num1[4]=5;
162     UnlimitedInt num2(5);
163     num2[0]=6;
164     num2[1]=5;
165     num2[2]=4;
166     num2[3]=3;
167     num2[4]=2;
168     UnlimitedInt sum(1);
169     sum=num1+num2;
170     QCOMPARE(sum, 77777);
171 }
172
173 void CppTestsTest::test_iu_subt()
174 {
175     UnlimitedInt num1(5);
176     num1[0]=1;
177     num1[1]=2;
178     num1[2]=3;
179     num1[3]=4;
180     num1[4]=5;
181     UnlimitedInt num2(5);
182     num2[0]=6;
183     num2[1]=5;
184     num2[2]=4;
185     num2[3]=3;
186     num2[4]=2;
187     UnlimitedInt subt(1);
188     subt=num1-num2;
189     QCOMPARE(subt, 77777);
190 }
191
192 void CppTestsTest::test_iu_mult()
```

```
193 {
194     UnlimitedInt num1(5);
195     num1[0]=1;
196     num1[1]=2;
197     num1[2]=3;
198     num1[3]=4;
199     num1[4]=5;
200     UnlimitedInt num2(5);
201     num2[0]=6;
202     num2[1]=5;
203     num2[2]=4;
204     num2[3]=3;
205     num2[4]=2;
206     UnlimitedInt mult(1);
207     mult=num1*num2;
208     QCOMPARE(mult, 77777);
209 }*/
210
211 QTEST_APPLESS_MAIN(CppTestsTest)
212
213 #include "tst_cppteststest.moc"
```