Программирование

А. Д. Орова

19 декабря 2015 г.

Оглавление

Глава 1

Основные конструкции языка

1.1 1. Банковская задача

1.1.1 Задание

Человек положил в банк сумму в s рублей под p% годовых (проценты начисляются в конце года). Сколько денег будет на счету через 5 лет?

1.1.2 Теоретические сведения

Для решения данной задачи используется формула вычисления сложного процента:

$$S = x + (1 + P)^n$$

, где S - конечная сумма, x - начальная сумма,P - процентная ставка и n - количество кварталов (лет).

Для реализации данного алгоритма был использован цикл for, счетчиком которого является количество лет, данное в задании. Также были применены функции библиотек stdio для ввода и вывода информации и math для выполнения необходимых вычислений.

1.1.3 Проектирование

В ходе проектирования было решено выделить две функции.

1. **bank** Функция вычисляет конечную сумму денег по вкладу в банк на 5 лет при определенном проценте, передаваемым в программу пользователем. Параметрами функции являются две переменные типа float: *summa* и *percent*. В первую переменную передается

первоначальная сумма, которую пользователь желает положить в банк, во вторую - процент, под который кладутся деньги.

2. **bank_console_ui** В этой функции реализованно взаимодействие с пользователем. В ней выполняется считывание 2 значений из консоли. Если данные введены правильно, то выполняется функция b, аргументами которой являются данные введенные пользователем, затем в зависимости от значения, которое вернула эта функция, в консоль выводится соответствующее сообщение.

1.1.4 Описание тестового стенда и методики тестирования

Среда разработки QtCreator 3.5.0, компилятор Qt 5.5.0 MinGW 32bit, операционная система Ubuntu 14.04. Были проведены ручные, а также модульные тесты.

1.1.5 Тестовый план и результаты тестирования

1. **Ручные тесты**

   I тест

   Входные данные: 1000 20
   Выходные данные: 2488,32
   Результат: Тест успешно пройден

   II тест

   Входные данные: 100 15
   Выходные данные: 201,13
   Результат: Тест успешно пройден

2. **Модульные тесты *Qt***

   I тест

   Входные данные: 200 25
   Выходные данные: 610,35
   Результат: Тест успешно пройден

   II тест

Входные данные: 10 90
Выходные данные: 247,60
Результат: Тест успешно пройден

1.1.6 Выводы

При выполнении задания я отработала свои навыки в работе с основными конструкциями языка и получила опыт в организации функций одной программы.

Листинги

bank.c

```c
#include"bank.h"

float bank(float summa, float percent)
{
    float result = summa;
    int i;
    for (i = 0; i < 5; i++)
        result *= (100 + percent) / 100;
    return result;
}
```

bank_console_ui.c

```c
#include<stdio.h>
#include"bank.h"
#include"bank_console_ui.h"

void bank_Console_UI()
{
    float summa, percent;
    printf(" Homework #1: Input, output and cycles\n\n");
    printf("\n");
    printf("  Exercise #1 \n\n");
    printf("    Please, input how much money You want to
       put to the bank: \n\t");
    scanf("%f", &summa);
    printf("    Please, input what is the percent at Your
        bank: \n\t");
    scanf("%f", &percent);

    printf("After 5 years You will have %f rubbles.\n\n",
        bank(summa, percent));
}
```

bank.h.c

```
#ifndef BANK_H
#define BANK_H
#include<stdio.h>
#include<math.h>

#ifdef __cplusplus
extern "C"{
#endif


float bank(float , float);


#ifdef __cplusplus
}
#endif

#endif // BANK_H
```

bank_console_ui.h

```
#ifndef BANK_CONSOLE_UI_H
#define BANK_CONSOLE_UI_H
void bank_Console_UI();
#endif // BANK_CONSOLE_UI_H
```

1.2 Задание

1.2.1 2. Возможность расположения домов на участке

Определить, можно ли на прямоугольном участке застройки размером a на b метров разместить 2 дома размером p на q и r на s метров? Дома можно располагать только параллельно сторонам участка.

1.2.2 Теоретические сведения

Для решения данной задачи необходимо знать, поместятся ли 2 дома на участке, и на каком участке. То есть если один дом не будет перекрывать другой, также находящийся на данной терртории, то программа выдаст положительный ответ. Иначе, если физически невозможно расположить 2 дома на однй территории, то программа выдаст отрицательный

ответ. Мной была использована конструкция if...else. А также функции библеотек stdio для ввода и вывода.

1.2.3 Проектирование

Для решения данной задачи я использую 6 переменных, в каждую из которых передаю линейную характеристику дома. В ходе проектирования были выделены следущие функции:

1. **home** В функции выполняется проверка того, могут ли быть два конкретных дома поместиться на конкретном участке. Проверка необходимо, так как в некоторых случаях два дома могут перекрывать участки друг друга. Условие состоит в том, чтобы такого перекрытия не было. Параметрами функции являются шесть переменных типа int. Если аргументы соответсвуют условию, то функциия вернет 1, в противном случае функция вернет 0.

2. **home_console_ui** Функция реализует взаимодействие с пользователем, который вводит длины домов. В случае, если предыдущая функция возвращает 1, то данная функция выведет на экран Yes, иначе No.

1.2.4 Описание тестового стенда и методики тестирования

Среда разработки QtCreator 3.5.0, компилятор Qt 5.5.0 MinGW 32bit, операционная система Ubuntu 14.04. Были проведены ручные, а также модульные тесты.

1.2.5 Тестовый план и результаты тестирования

1. **Ручные тесты**

   I тест

   Входные данные: 50 60 40 30 40 40
   Выходные данные: No
   Результат: Тест успешно пройден

   II тест

   Входные данные: 90 90 70 40 30 80

Выходные данные: Yes

Результат: Тест успешно пройден

2. **Модульные тесты *Qt***

I тест

Входные данные: 40 70 30 30 30 30

Выходные данные: Yes

Результат: Тест успешно пройден

II тест

Входные данные: 80 30 40 50 20 20

Выходные данные: No

Результат: Тест успешно пройден

1.2.6 Выводы

При выполнении задания я научилась использовать конструкцию if...else для решения не совсем тривиальных задач.

Листинги

home.c

```
int home(int length_horizontal_a, int length_vertical_a,
    int length_horizontal_h1, int length_vertical_h1, int
    length_horizontal_h2, int length_vertical_h2)
{
    if(((length_horizontal_h1 + length_horizontal_h2 <=
        length_horizontal_a) &&
        (length_vertical_h1 <= length_vertical_a) &&
        (length_vertical_h2 <= length_vertical_a)) ||
        ((length_vertical_h1 + length_horizontal_h2 <=
            length_horizontal_a) &&
        (length_horizontal_h1 <= length_vertical_a) &&
        (length_vertical_h2 <= length_vertical_a)) ||
        ((length_horizontal_h1 + length_vertical_h2 <=
            length_horizontal_a) &&
        (length_vertical_h1 <= length_vertical_a) &&
        (length_horizontal_h2 <= length_vertical_a)) ||
        ((length_vertical_h1 + length_vertical_h2 <=
            length_horizontal_a) &&
        (length_horizontal_h1 <= length_vertical_a) &&
        (length_horizontal_h2 <= length_vertical_a)))
```

```
15          return 1;
16      int temp = length_horizontal_a;
17      length_horizontal_a = length_vertical_a;
18      length_vertical_a = temp;
19      if(((length_horizontal_h1 + length_horizontal_h2 <=
            length_horizontal_a) &&
20          (length_vertical_h1 <= length_vertical_a) &&
21          (length_vertical_h2 <= length_vertical_a)) ||
22          ((length_vertical_h1 + length_horizontal_h2 <=
              length_horizontal_a) &&
23          (length_horizontal_h1 <= length_vertical_a) &&
24          (length_vertical_h2 <= length_vertical_a)) ||
25          ((length_horizontal_h1 + length_vertical_h2 <=
              length_horizontal_a ) &&
26          (length_vertical_h1 <= length_vertical_a) &&
27          (length_horizontal_h2 <= length_vertical_a)) ||
28          ((length_vertical_h1 + length_vertical_h2 <=
              length_horizontal_a) &&
29          (length_horizontal_h1 <= length_vertical_a) &&
30          (length_horizontal_h2 <= length_vertical_a)))
31          return 1;
32      return 0;
33  }
```

home_console_ui.c

```
1
2   #include<stdio.h>
3   #include"home.h"
4   #include"home_console_ui.h"
5
6   void home_Console_UI()
7   {
8       float length_horizontal_a, length_vertical_a,
            length_horizontal_h1, length_vertical_h1,
            length_horizontal_h2, length_vertical_h2;
9       printf(" Homework #2: Input, output and cycles\n\n");
10      printf("\n");
11      printf("Exercise #2 \n\n");
12      printf("Please, input length (horizontal)  of area a,
            b, p, q and r, s: \n");
13      scanf("%f", &length_horizontal_a);
14      scanf("%f", &length_vertical_a);
15      scanf("%f", &length_horizontal_h1);
16      scanf("%f", &length_vertical_h1);
17      scanf("%f", &length_horizontal_h2);
18      scanf("%f", &length_vertical_h2);
19      /*
```

```
20      printf("    Please, input linear dimensions of Your
            place: \n\n\t");
21      //scanf("%f %f", &a, &b);
22      printf("    Please, input linear dimensions of the
            first house: \n\n\t");
23      //scanf("%f %f", &p, &q);
24      printf("    Please, input linear dimensions of the
            second house: \n\n\t");
25      //scanf("%d %d", &r, &s);*/
26
27      if (home(length_horizontal_a, length_vertical_a,
            length_horizontal_h1, length_vertical_h1,
            length_horizontal_h2, length_vertical_h2) == 1)
28          printf("Yes\n");
29      else
30          printf("No\n");
31
32  }
```

home.h

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

home_console_ui.h

```
1  #ifndef HOME_CONSOLE_UI_H
2  #define HOME_CONSOLE_UI_H
3  void home_Console_UI();
4  #endif // HOME_CONSOLE_UI_H
```

Глава 2

Циклы

2.1 Таблица перевода из дюймов в сантиметры

2.1.1 Задание

Вывести на экран таблицу пересчета сантиметров в дюймы и обратно до заданного расстояния в сантиметрах, по возрастанию расстояний, как указано в примере (1 дюйм = 2.54 см). Пример для 6 см:

дюймы	см
0.39	1.00
0.79	2.00
1.00	2.54
1.18	3.00
1.57	4.00
1.97	5.00
2.00	5.08
2.36	6.00

2.1.2 Теоритические сведения

Для того, чтобы перевести из сантиметров в дюймы необходимо количество сантиметров поделить на эквивалент, равный 2,54. Для того чтобы перевести из дюймов в сантиметры - соответственно умножить на 2,54. Для выполнения задания использоваилсь функции библиотеки stdio для ввода и вывода.

2.1.3 Проектирование

В ходе проектирования было выделено три функции:

1. **cm_to_inch** Функция возвращает переданное ей количество сантиметров, поделенное на эквивалент. Параметром функции является переменная типа float.

2. **inch_to_cm** Функция возвращает переданное ей количество дюймов, поделенное на эквивалент. Параметром функции является переменная типа float.

3. **inch_to_cm_console_ui** В Функции реализованно взаимодействие с пользователем. В ней выполняется считывание из консоли числа, равного количеству сантиметров, котоое пользователь хочет перевести в сантиметры. Пользователю на экран выводится таблица от 1 сантиметра до введенного значения.

2.1.4 Описание тестового стенда и методики тестирования

Среда разработки QtCreator 3.5.0, компилятор Qt 5.5.0 MinGW 32bit, операционная система Ubuntu 14.04. Были проведены ручные, а также модульные тесты.

2.1.5 Тестовый план и результаты тестирования

1. **Ручные тесты**

   I тест

   Входные данные: 3
   Выходные данные:
   `"0,39\t 1,00\n 0,79\t 2,00\n 1,00\t 2,54\n 1,18\t 3,00\n"`
   Результат: Тест успешно пройден

   II тест

   Входные данные: 4
   Выходные данные:
   `"0,39\t 1,00\n 0,79\t 2,00\n 1,00\t 2,54\n 1,18\t 3,00\n 1,57\t 4,00`

Результат: Тест успешно пройден

2. **Модульные тесты *Qt***

I тест

Входные данные: 2
Выходные данные: `"0,39\t 1,00\n 0,79\t 2,00\n"`
Результат: Тест успешно пройден

II тест

Входные данные: 5
Выходные данные: `"0,39\t 1,00\n 0,79\t 2,00\n 1,00\t 2,54\n 1,18\t 3,00\n 1,57\t 4,00\n 1.97\t 5,00"`
Результат: Тест успешно пройден

2.1.6 Выводы

В ходе выполнения я отработала навыки работы с циклом с предусловием.

Листинги

cm_to_inch.c

```
1  #include <cm_to_inch.h>
2
3  double cm_to_inch(double cm)
4  {
5      return (cm/2.54f);
6  }
7
8  double inch_to_cm(double inch)
9  {
10     return (inch*2.54f);
11 }
```

cm_to_inch_console_ui.c

```
1  #include<stdio.h>
2  #include"cm_to_inch_console_ui.h"
3  #include"cm_to_inch.h"
4
5  void cm_to_inch_console()
```

```
6  {
7      int a;
8      printf("Input cm");
9      scanf("%d", &a);
10     double i, temp, tempInch = 1;
11     for (i = 1; i<=a; i++)
12     {
13         temp = cm_to_inch(i);
14         if (temp < tempInch)
15             printf("%.2f\t%.2f\n", temp, i);
16         else
17         {
18             printf("%.2f\t%.2f\n", tempInch, inch_to_cm(
                   tempInch));
19             i--;
20             tempInch++;
21         }
22
23     }
24 }
```

cm_to_inch.h

```
1  #ifndef CM_TO_INCH_H
2  #define CM_TO_INCH_H
3
4  #ifdef __cplusplus
5  extern "C"{
6  #endif
7
8  double cm_to_inch(double);
9  double inch_to_cm(double);
10
11 #ifdef __cplusplus
12 }
13 #endif
14
15 #endif // CM_TO_INCH_H
```

cm_to_inch_console_ui.h

```
1  #ifndef CM_TO_INCH_CONSOLE_UI_H
2  #define CM_TO_INCH_CONSOLE_UI_H
3  void cm_to_inch_console();
4  #endif // CM_TO_INCH_CONSOLE_UI_H
```

Глава 3

Массивы

3.1 Заполнение матрицы по спирали

3.1.1 Задание

Матрицу A(m,n) заполнить натуральными числами от 1 до m*n по спирали, начинающейся в левом верхнем углу и закрученной по часовой стрелке.

3.1.2 Теоритические сведения

Для выполнения задания использовался цикл for, конструкция if...else, а также функции бибилиотек stdlib для динамического выделения и освобождения памяти, stdio для ввода, вывода информации и работы с файлами.

3.1.3 Проектирование

В ходе проектирования были выделены четыре функции:

1. **initializeMatrix** Функция считывает из файла заданное количество целых чисел и сохраняет их в массив. Параметрами функции являются символьная строку содержащая имя файла, массив типа int, куда будут сохранятся считанные числа и переменная типа int, содержащая количество элемнтов. Функция возвращает количество успешно считанных из файла значений.

2. **fillSpiralMatrix** С помощью цикла с предусловием for() инкремента "++ а также конструкции if...else функция заполняет двумерный массив как и необходимо в задании, то есть по спирали.

3. **matrix_console_ui** Функция открывает файл и закрывает его после всех действий, считывает размеры матрицы(двумерного массива), выделяет память и позже её освобождает. Основная цель данной фунцкии - взаимодействие с пользователем.

4. **printMatrix** Функция выводит матрицу, заполненную по спирали на экран в консоль.

3.1.4 Описание тестового стенда и методики тестирования

Среда разработки QtCreator 3.5.0, компилятор Qt 5.5.0 MinGW 32bit, операционная система Ubuntu 14.04. Были проведены ручные, а также модульные тесты.

3.1.5 Тестовый план и результаты тестирования

1. **Модульные тесты *Qt***

   I тест

   Входные данные: 5 7
   Выходные данные:
   1 2 3 4 5
   20 21 22 23 6
   19 32 33 24 7
   18 31 34 25 8
   17 30 35 26 9
   16 29 28 27 10
   15 14 13 12 11
   Результат: Тест успешно пройден

2. **Статический анализ *cppcheck***

   Утилита *cppcheck* не выдала никаких предупреждений.

3.1.6 Выводы

При выполнении задания я поняла принцип организации программы при работе с выделением динамической памяти, научилася работать с файлами.

Листинги

matrix.c

```c
#include <stdlib.h>

int** initializeMatrix(int n, int m){
    int **array, i;
    array=(int **)malloc(n*sizeof(int*));
    for (i = 0; i<n; i++)
            array[i]=(int*)malloc(m*sizeof(int));
    return array;
}

void fillSpiralMatrix(int** array, int n, int m){
    int j, rows = 0, cols = 0, k = 1;
    int horbeg = 0, horend = m-1, vertbeg = 0, vertend =
       n-1;
    while(1){
        for(j = horbeg; j<horend+1; j++)
            array[horbeg][j] = k++;
        if (++rows == n) return;
        for(j = vertbeg+1; j<vertend+1; j++)
            array[j][horend] = k++;
        if (++cols == m) return;
        for(j = horend-1; j>=horbeg; j--)
            array[vertend][j] = k++;
        if (++rows == n) return;
        for(j = vertend-1; j>=vertbeg+1; j--)
            array[j][horbeg] = k++;
        if (++cols == m) return;
        horbeg++; horend--; vertbeg++; vertend--;
    }
}
```

matrix_console_ui.c

```c
#include<stdio.h>
#include"matrix_console_ui.h"
#include"matrix.h"

void matrix_console_UI(char* input_file_name, char*
   output_file_name){

    FILE* in;
    FILE* out;
    in = fopen(input_file_name, "r");
    out = fopen(output_file_name, "w");
    int m, n, i, j, k;
```

```
12
13      printf("input n");
14      scanf("%d", &n);
15      printf("input m");
16      scanf("%d", &m);
17      fscanf(in, "%i", &k);
18
19      int** array = initializeMatrix(n, m);
20      for (i = 0; i < n; ++i)
21          array[i] = (int*) malloc(n * sizeof(int));
22
23      for (i = 0; i < n; ++i)
24          for (j = 0; j < n; ++j)
25              fscanf(in, "%i\n", &array[i][j]);
26
27      fillSpiralMatrix(array, n, m);
28      printMatrix(array, n, m);
29
30      for (i = 0; i < n; ++i)
31      {
32          for (j = 0; j < n; ++j)
33              fprintf(out, "%i ", array[i][j]);
34          fprintf(out, "\n");
35      }
36
37      for (i = 0; i < n; ++i)
38          free(array[i]);
39      free(array);
40      fclose(in);
41      fclose(out);;
42  }
43
44  void printMatrix(int** array, int n, int m){
45      int i, j;
46      for (i = 0; i<n; i++){
47          for(j = 0; j<m; j++)
48              printf("%4d ", array[i][j]);
49          printf("\n");
50      }
51  }
```

matrix.h

```
1  #ifndef MATRIX_H
2  #define MATRIX_H
3
4  #ifdef __cplusplus
5  extern "C"{
6  #endif
```

```
7
8  int** initializeMatrix(int, int);
9  void fillSpiralMatrix(int**, int, int);
10
11
12 #ifdef __cplusplus
13 }
14 #endif
15 #endif // MATRIX_H
```

matrix_console_ui.h

```
1 #ifndef MATRIX_CONSOLE_UI_H
2 #define MATRIX_CONSOLE_UI_H
3 void matrix_console_UI();
4 void printMatrix(int**, int, int);
5 #endif // MATRIX_CONSOLE_UI_H
```

Глава 4

Строки

4.1 Выравнивание по ширине

4.1.1 Задание

Для реализации данной задачи было решено создать некоторое количество

4.1.2 Проектирование

В ходе проектирования было решено выделить 5 функций, 2 из которых отвечают за логику, а остальные – за взаимодействие с пользователем.

1. **cm_to_inch_console_ui** Функция для взаимодействия пользователем.
2. **initialize_text** Функция инициализирует введенный текст, а также выделяет на него память.
3. **initialize_string** Функция инициализирует переданную ей строку.
4. **input_text** Функция считывает строку.
5. **print_text** Функция выводит на экран текст.
6. **get_length** Функция считывает длину строки.
7. **count_spaces** Функция считает количество пробелов в строке.
8. **count_chars** Функция считает количество символов в строке.

9. **get_max_string_length** Функция ищет среди строк самую длинную.
10. **insert_char** Функция вставляет символ.
11. **insert_chars** Функция вставляет символы.
12. **get_string** Функция считывающая строку (массив).
13. **get_char_index** Функция, возвращающая индекс num-ового вхождения символа chr в строку str.
14. **spread_text** Функция, которая работает с пробелами.

4.1.3 Описание тестового стенда и методики тестирования

Среда разработки QtCreator 3.5.0, компилятор Qt 5.5.0 MinGW 32bit, операционная система Ubuntu 14.04. Были проведены ручные, а также модульные тесты.

4.1.4 Тестовый план и результаты тестирования

1. **Модульные тесты *Qt***

   I тест

   Входные данные: `"    dgrtf tfhrna f htya"`
   Выходные данные: `"dgrtf   tfhrna  f   htya"`
   Результат: Тест успешно пройден

2. **Статический анализ *cppcheck***

   Утилита *cppcheck* не выдала никаких предупреждений.

4.1.5 Выводы

В ходе работы я получила опыт в обработке строк, а также укрепила навык работы с файлами.

Листинги

strings.c

```c
#include<stdio.h>
#include<stdlib.h>
#include"strings.h"

char** initialize_text(int rows, int max){
    char** text;
    text = (char**) malloc(rows*sizeof(char*));
    int i;
    for (i = 0; i<rows; i++)
        text[i] = initialize_string(max);
    return text;
}

char* initialize_string(int max){
    return (char*) malloc(max*sizeof(char));
}

void input_text(char** text, int rows, int max){
    int i;
    //getchar();//считываем предыдущий enter
    for (i = 0; i<rows; i++)
        get_string(text[i], max);
}

void print_text(char** text, int rows){
    int i;
    for (i = 0; i<rows; i++)
        printf("%s\n", text[i]);
}

int get_length(char* string){
    int len = 0;
    while(*string++!=0) len++;
    return len;
}

int count_spaces(char* string){
    int count = 0;
    while(*string++!=0) count+=(*string==' '?1:0);
    return count;
}

int count_chars(char* string, char chr){
    int count = 0;
    while(*string++!=0) count+=(*string==chr?1:0);
    return count;
```

```
47 }
48
49 int get_max_string_length(char** text, int rows){
50     int max = get_length(text[0]), i;
51     for(i = 1; i<rows; i++)
52         max = (get_length(text[i])>max?get_length(text[i
               ]):max);
53     return max;
54 }
55
56 char* insert_char(char* str, int place, char chr){
57     int i;
58     char* result = initialize_string(get_length(str)+1);
59     for(i = 0; i<place; i++)
60         result[i] = str[i];
61     result[place] = chr;
62     for(i = place; i<get_length(str); i++)
63         result[i+1] = str[i];
64     result[get_length(str)+1] = 0;
65     return result;
66 }
67
68 char* insert_chars(char* str, int place, char chr, int
    count){
69     int i;
70     for(i = 0; i<count; i++)
71         str = insert_char(str,place,chr);
72     return str;
73 }
74
75 void get_string(char *str, int max){
76     int i = 0, ch;
77     while((ch = getchar()) != '\n')
78         if(str != NULL && i < max - 1)
79             str[i++] = ch;
80     if(str != NULL && i < max)
81         str[i] = 0;
82 }
83
84 //функция, возвращающее индекс num-ового вхождения символ
    а chr в строку str
85 int get_char_index(char* str, char chr, int num){
86     int i, temp = 0;
87     if(num>count_chars(str, chr))
88         return -1;
89     for(i = 0; i<get_length(str); i++)
90         if(str[i]==chr)
91             if(++temp == num)
92                 return i;
```

```
93      return i;
94
95  }
96
97  void spread_text(char** text, int rows){
98      int maxLength = get_max_string_length(text, rows);
99      int i;
100     for(i = 0; i<rows; i++)
101         if(get_length(text[i]) < maxLength){
102             int spaces = count_spaces(text[i]);
103             if (spaces==0)
104             {
105                 int count = maxLength-get_length(text[i])
                        ;
106                 text[i]=insert_chars(text[i],0,' ',count)
                        ;
107             }
108             else
109             {
110                 int count = maxLength - get_length(text[i
                        ]);
111                 int j;
112                 for(j = spaces; j>0; j--){
113                     //printf("%d\t\t%d\t%d - %d = %d\n",
                            i, spaces, maxLength, get_length(
                            text[i]), count);
114                     text[i] = insert_chars(text[i],
                            get_char_index(text[i],' ',j),' ',
                             count/spaces+(j>(spaces-count%
                            spaces)?1:0));
115                 }
116             }
117         }
118     printf("\n");
119 }
```

strings_console_ui.c

```
1
2   #include<stdio.h>
3   #include<stdlib.h>
4   #include"strings_console_ui.h"
5   #include"strings.h"
6   #define N 255
7
8   void strings_console_UI(){
9       int rows = 5;
10      char** text = initialize_text(rows, N);
11      input_text(text, rows, N);
```

```
12      printf("\n");
13      print_text(text, rows);
14      //printf("\nmaxLength = %d\t%d\n",
            get_max_string_length(text, rows), count_spaces(
            text[0]));
15      //text[0] = insert_char(text[0], 2, ' ');
16      //printf("%s\n", text[0]);
17      spread_text(text, rows);
18      print_text(text, rows);
19  }
```

strings.h

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

strings_console_ui.h

```
1  #ifndef MATRIX_CONSOLE_UI_H
2  #define MATRIX_CONSOLE_UI_H
3  void matrix_console_UI();
4  void printMatrix(int**, int, int);
5  #endif // MATRIX_CONSOLE_UI_H
```

Глава 5

Приложение к главам 1 - 4

5.1 Листинги

main.c

```
#include <stdio.h>
#include"bank_console_ui.h"
#include"home_console_ui.h"
#include"cm_to_inch_console_ui.h"
#include"matrix_console_ui.h"
#include"strings_console_ui.h"

void printHelp()
{
    printf("Запустите программу с одним из параметров:\n");
    printf("-bank - решение банковской задачи\n");
    printf("-home - решение задачи про дома\n");
    printf("-cm2inch - таблица перевода из сантиметров в дюймы\n");
    printf("-matrix - работа с матрицами\n");
    printf("-strings - выравнивание текста по ширине(5 строк)\n");
}

int strEquals(char* str1, char* str2)
{
    int i, res = 0;
    for(i = 0; str1[i] != '\0' && str2[i] != '\0'; i++)
        if (str1[i]!=str2[i])
            res++;
    //printf("\t\tS1%s\n", str1);
    //printf("\t\tS2%s\n", str2);
    return(res>0 ? 0 : 1);
```

```
}

int main(int argc, char *argv[])
{
    printf("\n\nПуть для терминала %s", argv[0]);
    printf("\n\nSTART OF WORK\n");
    if(argc>1){
        if(strEquals(argv[1], "-bank")) bank_Console_UI()
            ;
        if(strEquals(argv[1], "-home")) home_Console_UI()
            ;
        if(strEquals(argv[1], "-cm2inch"))
            cm_to_inch_console();
        if(strEquals(argv[1], "-matrix"))
            matrix_console_UI();
        if(strEquals(argv[1], "-strings"))
            strings_console_UI();
    }else{
        printHelp();
    }
    printf("\nEND OF WORK\n\n");

    return 0;
}
```

main.c

```
#include <QString>
#include <QtTest>
#include"bank.h"
#include"home.h"
#include"cm_to_inch.h"
#include"matrix.h"
#include"strings.h"

class TestTest : public QObject
{
    Q_OBJECT

public:
    TestTest();

private Q_SLOTS:
    void testCase1();
    void bank_test();
    void home_test();
    void cm2inch_test();
    void matrix_test();
    void strings_test();
```

```
23 };
24
25 TestTest::TestTest()
26 {
27 }
28
29 void TestTest::testCase1()
30 {
31     QVERIFY2(true, "Failure");
32 }
33
34 void TestTest::bank_test(){
35     QCOMPARE(bank(1000, 20), 2488.32f);
36 }
37
38 void TestTest::home_test(){
39     QVERIFY2(home(100, 100, 23, 32, 12, 35), "Failure");
40 }
41
42 void TestTest::cm2inch_test(){
43     QCOMPARE(cm_to_inch(3), 0.39);//     1.00
44                                   // 0.79     2.00
45                                   // 1.00     2.54
46                                   // 1.18     3.00));
47 }
48
49 void TestTest::matrix_test(){
50     int** res = initializeMatrix(2, 3);
51     res[0][0] = 1; res[0][1] = 2; res[0][2] = 3;
52     res[0][0] = 6; res[0][1] = 5; res[0][2] = 4;
53
54     int** tmp = initializeMatrix(2, 3);
55     fillSpiralMatrix(tmp, 2, 3);
56
57     for (int i = 0; i < 2; ++i)
58     {
59         for(int j = 0; j < 3; j++)
60         {
61             QCOMPARE(tmp[i][j], res[i][j]);
62         }
63     }
64
65 }
66
67 void TestTest::strings_test(){
68     char** resText = initialize_text(5, 255);
69     resText[0] = "banana banana";
70     resText[1] = "ban na ba  na";
71     resText[2] = "b  an  ba  na";
```

```
72      resText[3] = "b    nab     na";
73      resText[4] = "        banana";
74      char** tmpText = initialize_text(5, 255);
75      for (int i = 0; i < 5; ++i)
76      {
77          for(int j = 0; j < 255; j++)
78          {
79              QCOMPARE(tmpText[i][j], resText[i][j]);
80          }
81      }
82
83  }
84
85  QTEST_APPLESS_MAIN(TestTest)
86
87  #include "tst_testtest.moc"
```

Глава 6

Введение в классы C++

6.1 Задание 1. Инкапсуляция. Множество

6.1.1 Задание

Реализовать класс МНОЖЕСТВО (целых чисел). Требуемые методы: конструктор, деструктор, копирование, сложение множеств, пересечение множеств, добавление в множество, включение в множество.

6.1.2 Теоритические сведения

При разработке приложения была задействована объектная ориентированность языка C++.

6.1.3 Проектирование

В ходе проектирования было решено выделить 2 класса: set и Node.

Были выделены методы: *set()* - конструктор, s̃et()- деструктор, *copy(set source&)* - констрyтор копирования, *add(set added)* - сложение множеств, *contains(set s)* - пересечение множеств, *intersect(set s)* - добавление в множество, *intersect(set s)* - включение в множество. Также были выделены вспомогательные методы.

6.1.4 Описание тестового стенда и методики тестирования

Среда разработки QtCreator 3.5.0, компилятор Qt 5.5.0 MinGW 32bit, операционная система Ubuntu 14.04. Были проведены ручные, а также модульные тесты.

6.1.5 Тестовый план и результаты тестирования

ДОДЕЛАТЬ

6.1.6 Выводы

Я познакомилася с языком C++. Познакомилася с новой парадигмой программирования - ООП.

Листинги

main.cpp

```
#include <iostream>
#include "set.h"

using namespace std;

int main()
{
    set *s1 = new set();
    set *s2 = new set();
    s1->add(5);
    s1->add(2);
    s1->add(7);
    s1->add(9);
    s1->add(1);
    s1->add(9);
    s1->add(9);
    s1->add(9);
    s1->add(9);
    s2->add(-1);
    s2->add(-3);
    s2->add(-5);
    s2->add(2);

    return 0;
}
```

node.cpp

```
#include "node.h"

Node::Node(int d, Node* n){
    data = d;
    next = n;
}
Node::Node(int d){
```

```
    data = d;
    next = nullptr;
}
Node::~Node(){
}
```

set.cpp

```
#include "set.h"
#include "node.h"

set::set()
{
    root = nullptr;
}

set::~set()
{
    Node *temp;
    while(root != nullptr)
    {
        temp = root;
        root = root->next;
        temp->~Node();
    }
}

set set::copy(set source)
{
    Node *temp = source.root;
    set *result = new set();
    while(temp!=nullptr){
        result->addToBeg(temp->data);
        temp = temp->next;
    }
    return *result;
}

void set::addToBeg(int data)
{
    root = new Node(data, root);
}

Node* set::searchByKey(int data)
{
    Node *temp=root;
    while(temp!=nullptr && temp->data!=data)
        temp = temp->next;
    return temp;
```

```
42 }
43
44 void set::add(int data)
45 {
46     if (searchByKey(data)!=nullptr)
47         return;
48     addToBeg(data);
49 }
50
51 void set::add(set added)
52 {
53     Node *temp = added.root;
54     while(temp!=nullptr){
55         add(temp->data);
56         temp = temp->next;
57     }
58 }
59
60 bool set::contains(set s)
61 {
62     Node *temp = s.root;
63     bool result = true;
64     while(temp!=nullptr){
65         result = result && contains(temp->data);
66         temp = temp->next;
67     }
68     return result;
69 }
70
71 bool set::contains(int data)
72 {
73     return (searchByKey(data)!=nullptr);
74 }
75
76 set set::intersect(set s)
77 {
78     set *result = new set();
79     Node *temp = root;
80     while(temp!=nullptr)
81     {
82         if (s.contains(temp->data))
83             result->add(temp->data);
84         temp = temp->next;
85     }
86     return *result;
87 }
88
89 int set::count()
90 {
```

```
91      Node *temp = root;
92      int result = 0;
93      while(temp!=nullptr)
94      {
95          result++;
96          temp = temp->next;
97      }
98      return result;
99  }
100
101 bool set::isEmpty()
102 {
103     return (count()<=0);
104 }
```

node.h

```
1  #ifndef NODE_H
2  #define NODE_H
3
4
5  class Node
6  {
7  public:
8      int data;
9      Node* next;
10     Node(int);
11     Node(int, Node*);
12     ~Node();
13 };
14
15 #endif // NODE_H
```

set.h

```
1  #ifndef SET_H
2  #define SET_H
3  #include <iostream>
4  #include "node.h"
5
6
7  class set
8  {
9  public:
10
11     set();
12     ~set();
13     void add(int);
```

```
14      void add(set);
15      bool contains(int);
16      bool contains(set);
17      set copy(set);
18      set intersect(set s);
19      int count();
20      bool isEmpty();
21
22  private:
23      Node *root;
24      void addToBeg(int);
25      Node* searchByKey(int);
26  };
27
28  #endif // SET_H
```

Глава 7

Классы C++

7.1 Задание 1. Реализовать классы для всех приложений

7.1.1 Задание

Реализовать классы для всех приложений. Поработать с потоками.

7.1.2 Выводы

Получила опыт создания классов. Получила опыт в работе с потоками.

Листинги

bankconsoleuicpp.cpp

```
1  #include "bankconsoleuicpp.h"
2  #include "bankcpp.h"
3  #include <iostream>
4  using namespace std;
5
6  BankConsoleUICPP::BankConsoleUICPP()
7  {
8
9  }
10
11 void BankConsoleUICPP::doWork()
12 {
13     float summa, percent;
14     cout << "\tHomework #1: Input, output and cycles" <<
           endl;
```

```
15      cout << "Exercise #1" << endl << endl;
16      cout << "Please, input how much money You want to put
            to the bank:" << endl;
17      cin >> summa;
18      cout << "    Please, input what is the percent at
           Your bank:" << endl;
19      cin >> percent;
20      BankCPP bankWorker(summa, percent);
21      cout << "After 5 years You will have " << (bankWorker
           .doWork()) << "rubbles" << endl;
22  }
```

bankconsoleuicpp.h

```
1  #ifndef BANKCONSOLEUICPP_H
2  #define BANKCONSOLEUICPP_H
3
4
5  class BankConsoleUICPP
6  {
7  public:
8      BankConsoleUICPP();
9      void doWork();
10 };
11
12 #endif // BANKCONSOLEUICPP_H
```

cmtoinchconsoleuicpp.cpp

```
1  #include "cmtoinchconsoleuicpp.h"
2  #include "cmtoinchcpp.h"
3  #include <iostream>
4  using namespace std;
5
6  CmToInchConsoleUICPP::CmToInchConsoleUICPP()
7  {
8
9  }
10
11
12 void CmToInchConsoleUICPP::doWork()
13 {
14     int a;
15     cout << "Input cm";
16     cin >> a;
17     double i, temp, tempInch = 1;
18     CmToInchCPP cmToInchWorker;
19     for (i = 1; i<=a; i++)
```

```
20      {
21          temp = cmToInchWorker.cm_to_inch(i);
22          if (temp < tempInch)
23              cout << temp << "\t" << i << endl;
24          else
25          {
26              cout << tempInch << "\t" << cmToInchWorker.
                    inch_to_cm(tempInch) << endl;
27              i--;
28              tempInch++;
29          }
30
31      }
32  }
```

cmtoinchconsoleuicpp.h

```
1  #ifndef CMTOINCHCONSOLEUICPP_H
2  #define CMTOINCHCONSOLEUICPP_H
3
4
5  class CmToInchConsoleUICPP
6  {
7  public:
8      CmToInchConsoleUICPP();
9      void doWork();
10 };
11
12 #endif // CMTOINCHCONSOLEUICPP_H
```

homeconsoleuicpp.cpp

```
1  #include "homeconsoleuicpp.h"
2  #include "homecpp.h"
3  #include <iostream>
4  using namespace std;
5
6  HomeConsoleUICPP::HomeConsoleUICPP()
7  {
8
9  }
10
11 void HomeConsoleUICPP::doWork()
12 {
13     float length_horizontal_a, length_vertical_a,
           length_horizontal_h1, length_vertical_h1,
           length_horizontal_h2, length_vertical_h2;
14     cout << "Homework #2: Input, output and cycles\n\n\n"
           ;
```

```
15      cout << "Exercise #2 \n\n";
16      cout << "Please, input length (horizontal) of area a
            , b, p, q and r, s: \n";
17      cin >> length_horizontal_a;
18      cin >> length_vertical_a;
19      cin >> length_horizontal_h1;
20      cin >> length_vertical_h1;
21      cin >> length_horizontal_h2;
22      cin >> length_vertical_h2;
23      HomeCPP homeWorker(length_horizontal_a,
            length_vertical_a, length_horizontal_h1,
            length_vertical_h1, length_horizontal_h2,
            length_vertical_h2);
24
25      if (homeWorker.doWork() == 1)
26          cout << "Yes\n";
27      else
28          cout << "No\n";
29
30  }
```

homeconsoleuicpp.h

```
1  #ifndef HOMECONSOLEUICPP_H
2  #define HOMECONSOLEUICPP_H
3
4
5  class HomeConsoleUICPP
6  {
7  public:
8      HomeConsoleUICPP();
9      void doWork();
10 };
11
12 #endif // HOMECONSOLEUICPP_H
```

matrixconsoleuicpp.cpp

```
1  #include "matrixconsoleuicpp.h"
2  #include "matrixcpp.h"
3  #include <stdio.h>
4  #include <iostream>
5  using namespace std;
6  //MatrixConsoleUICPP matrixWorker; matrixWorker.doWork();
7  MatrixConsoleUICPP::MatrixConsoleUICPP()
8  {
9
10 }
```

```
11
12 void MatrixConsoleUICPP::doWork(char* input_file_name,
       char* output_file_name){
13
14     FILE* in;
15     FILE* out;
16     in = fopen(input_file_name, "r");
17     out = fopen(output_file_name, "w");
18     int m, n, i, j, k;
19
20     cout << "input n";
21     cin >> n;
22     cout << "input m";
23     cin >> m;
24     fscanf(in, "%i", &k);
25
26     MatrixCPP matrixWorker;
27
28     int** array = matrixWorker.initializeMatrix(n, m);
29
30     for (i = 0; i < n; ++i)
31         for (j = 0; j < n; ++j)
32             fscanf(in, "%i\n", &array[i][j]);
33
34     matrixWorker.fillSpiralMatrix(array, n, m);
35     printMatrix(array, n, m);
36
37     for (i = 0; i < n; ++i)
38     {
39         for (j = 0; j < n; ++j)
40             fprintf(out, "%i ", array[i][j]);
41         fprintf(out, "\n");
42     }
43
44     for (i = 0; i < n; ++i)
45         delete array[i];
46     delete array;
47     fclose(in);
48     fclose(out);;
49 }
50
51 void MatrixConsoleUICPP::printMatrix(int** array, int n,
       int m){
52     int i, j;
53     for (i = 0; i<n; i++){
54         for(j = 0; j<m; j++)
55             cout << array[i][j] << " ";
56         cout << endl;
57     }
```

```
58 }
```

matrixconsoleuicpp.h

```
1  #ifndef MATRIXCONSOLEUICPP_H
2  #define MATRIXCONSOLEUICPP_H
3
4
5  class MatrixConsoleUICPP
6  {
7  public:
8      MatrixConsoleUICPP();
9      void doWork(char*, char*);
10     void printMatrix(int**, int, int);
11 };
12
13 #endif // MATRIXCONSOLEUICPP_H
```

stringsconsoleuicpp.cpp

```
1  #include "stringsconsoleuicpp.h"
2  #include"stringscpp.h"
3  #define N 255
4
5  StringsConsoleUICPP::StringsConsoleUICPP()
6  {
7
8  }
9
10 void StringsConsoleUICPP::doWork(){
11     int rows = 5;
12     StringsCPP stringsWorker;
13     char** text = stringsWorker.initialize_text(rows, N);
14     stringsWorker.input_text(text, rows, N);
15     stringsWorker.print_text(text, rows);
16     //printf("\nmaxLength = %d\t%d\n",
           get_max_string_length(text, rows), count_spaces(
           text[0]));
17     //text[0] = insert_char(text[0], 2, ' ');
18     //printf("%s\n", text[0]);
19     stringsWorker.spread_text(text, rows);
20     stringsWorker.print_text(text, rows);
21 }
```

stringsconsoleuicpp.h

```
1  #ifndef STRINGSCONSOLEUICPP_H
2  #define STRINGSCONSOLEUICPP_H
```

```
3
4
5  class StringsConsoleUICPP
6  {
7  public:
8      StringsConsoleUICPP();
9      void doWork();
10 };
11
12 #endif // STRINGSCONSOLEUICPP_H
```

bankcpp.cpp

```
1  #include "bankcpp.h"
2
3  BankCPP::BankCPP(float summa, float percent)
4  {
5      this->summa = summa;
6      this->percent = percent;
7  }
8
9  float BankCPP::doWork()
10 {
11     float result = summa;
12     int i;
13     for (i = 0; i < 5; i++)
14         result *= (100 + percent) / 100;
15     return result;
16 }
```

bankcpp.h

```
1  #ifndef BANKCPP_H
2  #define BANKCPP_H
3
4
5  class BankCPP
6  {
7  public:
8      float summa;
9      float percent;
10     BankCPP(float, float);
11     float doWork();
12 };
13
14 #endif // BANKCPP_H
```

cmtoinchcpp.cpp

```cpp
#include "cmtoinchcpp.h"

CmToInchCPP::CmToInchCPP()
{

}

double CmToInchCPP::cm_to_inch(double cm)
{
    return (cm/2.54f);
}

double CmToInchCPP::inch_to_cm(double inch)
{
    return (inch*2.54f);
}
```

cmtoinchcpp.h

```cpp
#ifndef CMTOINCHCPP_H
#define CMTOINCHCPP_H


class CmToInchCPP
{
public:
    CmToInchCPP();
    double cm_to_inch(double);
    double inch_to_cm(double);
};

#endif // CMTOINCHCPP_H
```

homecpp.cpp

```cpp
#include "homecpp.h"

HomeCPP::HomeCPP(int length_horizontal_a, int
    length_vertical_a, int length_horizontal_h1, int
    length_vertical_h1, int length_horizontal_h2, int
    length_vertical_h2)
{
    this->length_horizontal_a  = length_horizontal_a;
    this->length_vertical_a    = length_vertical_a;
    this->length_horizontal_h1 = length_horizontal_h1;
    this->length_vertical_h1   = length_vertical_h1;
    this->length_horizontal_h2 = length_horizontal_h2;
    this->length_vertical_h2   = length_vertical_h2;
```

```
11 }
12
13 int HomeCPP::doWork()
14 {
15     if(((length_horizontal_h1 + length_horizontal_h2 <=
          length_horizontal_a) &&
16         (length_vertical_h1 <= length_vertical_a) &&
17         (length_vertical_h2 <= length_vertical_a)) ||
18         ((length_vertical_h1 + length_horizontal_h2 <=
             length_horizontal_a) &&
19         (length_horizontal_h1 <= length_vertical_a) &&
20         (length_vertical_h2 <= length_vertical_a)) ||
21         ((length_horizontal_h1 + length_vertical_h2 <=
             length_horizontal_a) &&
22         (length_vertical_h1 <= length_vertical_a) &&
23         (length_horizontal_h2 <= length_vertical_a)) ||
24         ((length_vertical_h1 + length_vertical_h2 <=
             length_horizontal_a) &&
25         (length_horizontal_h1 <= length_vertical_a) &&
26         (length_horizontal_h2 <= length_vertical_a)))
27         return 1;
28     int temp = length_horizontal_a;
29     length_horizontal_a = length_vertical_a;
30     length_vertical_a = temp;
31     if(((length_horizontal_h1 + length_horizontal_h2 <=
          length_horizontal_a) &&
32         (length_vertical_h1 <= length_vertical_a) &&
33         (length_vertical_h2 <= length_vertical_a)) ||
34         ((length_vertical_h1 + length_horizontal_h2 <=
             length_horizontal_a) &&
35         (length_horizontal_h1 <= length_vertical_a) &&
36         (length_vertical_h2 <= length_vertical_a)) ||
37         ((length_horizontal_h1 + length_vertical_h2 <=
             length_horizontal_a ) &&
38         (length_vertical_h1 <= length_vertical_a) &&
39         (length_horizontal_h2 <= length_vertical_a)) ||
40         ((length_vertical_h1 + length_vertical_h2 <=
             length_horizontal_a) &&
41         (length_horizontal_h1 <= length_vertical_a) &&
42         (length_horizontal_h2 <= length_vertical_a)))
43         return 1;
44     return 0;
45 }
```

homecpp.h

```
1 #ifndef HOMECPP_H
2 #define HOMECPP_H
3
```

```
4
5  class HomeCPP
6  {
7  public:
8      int length_horizontal_a;
9      int length_vertical_a;
10     int length_horizontal_h1;
11     int length_vertical_h1;
12     int length_horizontal_h2;
13     int length_vertical_h2;
14     HomeCPP(int, int, int, int, int, int);
15     int doWork();
16 };
17
18 #endif // HOMECPP_H
```

matrixcpp.cpp

```
1  #include "matrixcpp.h"
2  #include <stdlib.h>
3
4  MatrixCPP::MatrixCPP()
5  {
6
7  }
8
9
10 int** MatrixCPP::initializeMatrix(int n, int m){
11     int **array, i;
12     array=(int **)malloc(n*sizeof(int*));
13     for (i = 0; i<n; i++)
14             array[i]=(int*)malloc(m*sizeof(int));
15     return array;
16 }
17
18 void MatrixCPP::fillSpiralMatrix(int** array, int n, int
       m){
19     int j, rows = 0, cols = 0, k = 1;
20     int horbeg = 0, horend = m-1, vertbeg = 0, vertend =
           n-1;
21     while(1){
22         for(j = horbeg; j<horend+1; j++)
23             array[horbeg][j] = k++;
24         if (++rows == n) return;
25         for(j = vertbeg+1; j<vertend+1; j++)
26             array[j][horend] = k++;
27         if (++cols == m) return;
28         for(j = horend-1; j>=horbeg; j--)
29             array[vertend][j] = k++;
```

```
30          if (++rows == n) return;
31          for(j = vertend-1; j>=vertbeg+1; j--)
32              array[j][horbeg] = k++;
33          if (++cols == m) return;
34          horbeg++; horend--; vertbeg++; vertend--;
35      }
36  }
```

matrixcpp.h

```
1  #ifndef MATRIXCPP_H
2  #define MATRIXCPP_H
3
4
5  class MatrixCPP
6  {
7  public:
8      MatrixCPP();
9      int** initializeMatrix(int, int);
10     void fillSpiralMatrix(int**, int, int);
11 };
12
13 #endif // MATRIXCPP_H
```

stringscpp.cpp

```
1  #include<stdio.h>
2  #include<stdlib.h>
3  #include "stringscpp.h"
4
5  StringsCPP::StringsCPP()
6  {
7
8  }
9
10
11 char** StringsCPP::initialize_text(int rows, int max){
12     char** text;
13     text = (char**) malloc(rows*sizeof(char*));
14     int i;
15     for (i = 0; i<rows; i++)
16         text[i] = initialize_string(max);
17     return text;
18 }
19
20 char* StringsCPP::initialize_string(int max){
21     return (char*) malloc(max*sizeof(char));
22 }
```

```
23
24 void StringsCPP::input_text(char** text, int rows, int
       max){
25     int i;
26     //getchar();//считываем предыдущий enter
27     for (i = 0; i<rows; i++)
28         get_string(text[i], max);
29 }
30
31 void StringsCPP::print_text(char** text, int rows){
32     int i;
33     for (i = 0; i<rows; i++)
34         printf("%s\n", text[i]);
35 }
36
37 int StringsCPP::get_length(char* string){
38     int len = 0;
39     while(*string++!=0) len++;
40     return len;
41 }
42
43 int StringsCPP::count_spaces(char* string){
44     int count = 0;
45     while(*string++!=0) count+=(*string==' '?1:0);
46     return count;
47 }
48
49 int StringsCPP::count_chars(char* string, char chr){
50     int count = 0;
51     while(*string++!=0) count+=(*string==chr?1:0);
52     return count;
53 }
54
55 int StringsCPP::get_max_string_length(char** text, int
       rows){
56     int max = get_length(text[0]), i;
57     for(i = 1; i<rows; i++)
58         max = (get_length(text[i])>max?get_length(text[i
               ]):max);
59     return max;
60 }
61
62 char* StringsCPP::insert_char(char* str, int place, char
       chr){
63     int i;
64     char* result = initialize_string(get_length(str)+1);
65     for(i = 0; i<place; i++)
66         result[i] = str[i];
67     result[place] = chr;
```

```
68      for(i = place; i<get_length(str); i++)
69          result[i+1] = str[i];
70      result[get_length(str)+1] = 0;
71      return result;
72  }
73
74  char* StringsCPP::insert_chars(char* str, int place, char
        chr, int count){
75      int i;
76      for(i = 0; i<count; i++)
77          str = insert_char(str,place,chr);
78      return str;
79  }
80
81  void StringsCPP::get_string(char *str, int max){
82      int i = 0, ch;
83      while((ch = getchar()) != '\n')
84          if(str != NULL && i < max - 1)
85              str[i++] = ch;
86      if(str != NULL && i < max)
87          str[i] = 0;
88  }
89
90  //функция, возвращающее индекс num-ового вхождения символ
        а chr в строку str
91  int StringsCPP::get_char_index(char* str, char chr, int
        num){
92      int i, temp = 0;
93      if(num>count_chars(str, chr))
94          return -1;
95      for(i = 0; i<get_length(str); i++)
96          if(str[i]==chr)
97              if(++temp == num)
98                  return i;
99      return i;
100
101 }
102
103 void StringsCPP::spread_text(char** text, int rows){
104     int maxLength = get_max_string_length(text, rows);
105     int i;
106     for(i = 0; i<rows; i++)
107         if(get_length(text[i]) < maxLength){
108             int spaces = count_spaces(text[i]);
109             if (spaces==0)
110             {
111                 int count = maxLength-get_length(text[i])
                        ;
112                 text[i]=insert_chars(text[i],0,' ',count)
```

```
                    ;
113             }
114             else
115             {
116                 int count = maxLength - get_length(text[i
                        ]);
117                 int j;
118                 for(j = spaces; j>0; j--){
119                     //printf("%d\t\t%d\t%d - %d = %d\n",
                            i, spaces, maxLength, get_length(
                            text[i]), count);
120                     text[i] = insert_chars(text[i],
                            get_char_index(text[i],' ',j),' ',
                             count/spaces+(j>(spaces-count%
                            spaces)?1:0));
121                 }
122             }
123         }
124     printf("\n");
125 }
```

stringscpp.h

```
1  #ifndef STRINGSCPP_H
2  #define STRINGSCPP_H
3
4
5  class StringsCPP
6  {
7  public:
8      StringsCPP();
9      void get_string(char*, int);
10     char** initialize_text(int, int);
11     char* initialize_string(int);
12     void input_text(char**, int, int);
13     void print_text(char**, int);
14     int get_length(char*);
15     int get_max_string_length(char**, int);
16     int count_spaces(char*);
17     char* insert_char(char*, int, char);
18     char* insert_chars(char*, int, char, int);
19     void spread_text(char**, int);
20     int get_char_index(char*, char, int);
21     int count_chars(char*, char);
22 };
23
24 #endif // STRINGSCPP_H
```

tst_testcpptest.cpp

```
1  #include <QString>
2  #include <QtTest>
3
4  class TestCPPTest : public QObject
5  {
6      Q_OBJECT
7
8  public:
9      TestCPPTest();
10
11 private Q_SLOTS:
12     void testCase1();
13 };
14
15 TestCPPTest::TestCPPTest()
16 {
17 }
18
19 void TestCPPTest::testCase1()
20 {
21     QVERIFY2(true, "Failure");
22 }
23
24 QTEST_APPLESS_MAIN(TestCPPTest)
25
26 #include "tst_testcpptest.moc"
```