Программирование

А. Д. Орова

22 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 была использована 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.h для ввода и вывода информации и math.h для выполнения необходимых вычислений.

1.1.3 Проектирование

В ходе проектирования было решено выделить две функции:

- double bank(double , double);
- void bank_console_UI();

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 bank, 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 gcc version 4.8.4 (Ubuntu 4.8.4-2ubuntu1 14.04), операционная система Ubuntu 14.04. Были проведены ручные, а также модульные тесты.

1.1.5 Тестовый план и результаты тестирования

Для наиболее точного понимания того, что происходит в ходе тестирования, далее будет описан процесс тестирования. Первое входное данное - это начальная сумма, которую пользователь хочет положить в банк. Второе входное данное - это количество процентов, под которое денежная сумма кладется в банк.

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 были получены навыки в работе с основными конструкциями языка, а также опыт организации функций одной программы.

Листинги

bank.c

```
#include"bank.h"

/// Да почему же float, а не double, float на несколько б
    айтом меньше double, но на порядки менее точный
///
/// И вы уже знаете, что эта функция просто делает рассче
    т сложного процента,
/// так и назовите ее
double bank(double summa, double percent)
{
    double result = summa;
    int i;
    for (i = 0; i < 5; i++)
        result *= (100 + percent) / 100;
    return result;
}
```

bank_console_ui.c

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

```
4
5  void bank_Console_UI()
6  {
7      double summa, percent;
8      printf(" Homework #1: Input, output and cycles\n\n");
9      printf("\n");
10     printf("  Exercise #1 \n\n");
11     printf("    Please, input how much money You want to
           put to the bank: \n\t");
12     scanf("%f", &summa);
13     printf("    Please, input what is the percent at Your
           bank: \n\t");
14     scanf("%f", &percent);
15
16     printf("After 5 years You will have %f rubbles.\n\n",
           bank(summa, percent));
17 }
```

bank.h.c

```
1  #ifndef BANK_H
2  #define BANK_H
3  #include<stdio.h>
4  #include<math.h>
5
6  #ifdef __cplusplus
7  extern "C"{
8  #endif
9
10
11 double bank(double , double);
12
13
14 #ifdef __cplusplus
15 }
16 #endif
17
18 #endif // BANK_H
```

bank_console_ui.h

```
1  #ifndef BANK_CONSOLE_UI_H
2  #define BANK_CONSOLE_UI_H
3  void bank_Console_UI();
4  #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. Была использована конструкция if...else, а также функции библеотек stdio.h для ввода и вывода.

1.2.3 Проектирование

Для решения данной задачи используются 6 переменных, в каждую из которых передаётся линейная характеристика дома или участка. В ходе проектирования были выделены следущие функции:

- int home(Size, Size, Size);
- void home_Console_UI();

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 были улучшены навыки использования конструкции if...else для решения не совсем тривиальных задач.

Листинги

home.c

```c
#include <home.h>

int home(Size home1, Size home2, Size area  )
{
    if(((home1.width + home2.width <= area.width) &&
        (home1.height <= area.height) &&
        (home2.height <= area.height)) ||
        ((home1.height + home2.width <= area.width) &&
        (home1.width <= area.height) &&
        (home2.height <= area.height)) ||
        ((home1.width + home2.height <= area.width) &&
        (home1.height <= area.height) &&
        (home2.width <= area.height)) ||
        ((home1.height + home2.height <= area.width) &&
        (home1.width <= area.height) &&
        (home2.width <= area.height)))
        return 1;
    int temp = area.width;
    area.width = area.height;
    area.height = temp;
    if(((home1.width + home2.width <= area.width) &&
        (home1.height <= area.height) &&
        (home2.height <= area.height)) ||
        ((home1.height + home2.width <= area.width) &&
        (home1.width <= area.height) &&
        (home2.height <= area.height)) ||
        ((home1.width + home2.height <= area.width ) &&
        (home1.height <= area.height) &&
        (home2.width <= area.height)) ||
        ((home1.height + home2.height <= area.width) &&
        (home1.width <= area.height) &&
        (home2.width <= area.height)))
        return 1;
    return 0;
}
```

home_console_ui.c

```c

#include<stdio.h>
#include"home.h"
#include"home_console_ui.h"

void home_Console_UI()
{
```

```
8       printf(" Homework #2: Input, output and cycles\n\n");
9       printf("\n");
10      printf("Exercise #2 \n\n");
11      printf("Please, input length (horizontal)  of area a,
            b, p, q and r, s: \n");
12      Size home1, home2, area;
13      scanf("%f", &area.width);
14      scanf("%f", &area.height);
15      scanf("%f", &home1.width);
16      scanf("%f", &home1.height);
17      scanf("%f", &home2.width);
18      scanf("%f", &home2.height);
19
20      if (home(area, home1, home2) == 1)
21          printf("Yes\n");
22      else
23          printf("No\n");
24
25  }
```

home.h

```
1   #ifndef HOME_H
2   #define HOME_H
3
4   typedef struct{
5       int width;
6       int height;
7   }Size;
8
9   #ifdef __cplusplus
10  extern "C"{
11  #endif
12
13      int home(Size, Size, Size);
14
15  #ifdef __cplusplus
16  }
17  #endif
18
19  #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 Теоретические 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.h 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  1,00"`
   `"0,79  2,00"`
   `"1,00  2,54"`
   `"1,18  3,00"`
   Результат: Тест успешно пройден

   II тест

Входные данные: 4

Выходные данные:

```
"0,39  1,00"
"0,79  2,00"
"1,00  2,54"
"1,18  3,00"
"1,57  4,00"
```

Результат: Тест успешно пройден

2. **Модульные тесты *Qt***

I тест

Входные данные: 2

Выходные данные:

```
"0,39  1,00"
"0,79  2,00"
```

Результат: Тест успешно пройден

II тест

Входные данные: 5

Выходные данные:

```
"0,39  1,00"
"0,79  2,00"
"1,00  2,54"
"1,18  3,00"
"1,57  4,00"
"1,97  5,00"
```

Результат: Тест успешно пройден

2.1.6 Выводы

В ходе выполнения были отработаны навыки работы с циклом с предусловием.

Листинги

cm_to_inch.c

```c
#include <cm_to_inch.h>

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

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

cm_to_inch_console_ui.c

```c
#include<stdio.h>
#include"cm_to_inch_console_ui.h"
#include"cm_to_inch.h"

void cm_to_inch_console()
{
    int a;
    printf("Input cm");
    scanf("%d", &a);
    double i, temp, tempInch = 1;
    for (i = 1; i<=a; i++)
    {
        temp = cm_to_inch(i);
        if (temp < tempInch)
            printf("%.2f\t%.2f\n", temp, i);
        else
        {
            printf("%.2f\t%.2f\n", tempInch, inch_to_cm(
                tempInch));
            i--;
            tempInch++;
        }

    }
}
```

cm_to_inch.h

```c
#ifndef CM_TO_INCH_H
#define CM_TO_INCH_H

```

```
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 $m \times n$ по спирали, начинающейся в левом верхнем углу и закрученной по часовой стрелке.

3.1.2 Теоретические сведения

Для выполнения задания был использован цикл for, конструкция if...else, а также функции бибилиотек stdlib.h для динамического выделения и освобождения памяти, stdio.h 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 с использование утилиты *cppcheck*, версии 1.71 для Ubuntu (64-bit)**

   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 = (int **)malloc(n*sizeof(int*));
20
21      for (i = 0; i < n; ++i)
22          array[i] = (int*) malloc(n * sizeof(int));
23
24      for (i = 0; i < n; ++i)
25          for (j = 0; j < n; ++j)
26              fscanf(in, "%i\n", &array[i][j]);
27
28      fclose(in);
29
30      fillSpiralMatrix(array, n, m);
31      printMatrix(array, n, m);
32
33      for (i = 0; i < n; ++i)
34      {
35          for (j = 0; j < n; ++j)
36              fprintf(out, "%i ", array[i][j]);
37          fprintf(out, "\n");
38      }
39
40      for (i = 0; i < n; ++i)
41          free(array[i]);
42      free(array);
43
44      fclose(out);;
45  }
46
47  void printMatrix(int** array, int n, int m){
48      int i, j;
49      for (i = 0; i<n; i++){
50          for(j = 0; j<m; j++)
51              printf("%4d ", array[i][j]);
52          printf("\n");
53      }
54  }
```

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:

   ```
   "banana banana"
   "ban na ba  na"
   "b  an  ba  na"
   "b   nab    na"
   "       banana"
   ```

   Выходные данные:

   ```
   "banana banana"
   "ban  na ba na"
   "b  an  ba  na"
   "b   nab    na"
   "   banana    "
   ```

Результат: Тест успешно пройден

2. **Статический анализ *cppcheck***

   Утилита *cppcheck* не выдала никаких предупреждений.

4.1.5 Выводы

В ходе работы был получен опыт в обработке строк, а также укреплен навык работы с файлами.

Листинги

strings.c

```
1  #include<stdio.h>
2  #include<stdlib.h>
3  #include"strings.h"
4
5  char** initialize_text(int rows, int max){
6      char** text;
7      text = (char**) malloc(rows*sizeof(char*));
8      int i;
9      for (i = 0; i<rows; i++)
10         text[i] = initialize_string(max);
11     return text;
12 }
13
14 char* initialize_string(int max){
15     return (char*) malloc(max*sizeof(char));
16 }
17
18 void input_text(char** text, int rows, int max){
19     int i;
20     //getchar();//считываем предыдущий enter
21     for (i = 0; i<rows; i++)
22         get_string(text[i], max);
23 }
24
25 void print_text(char** text, int rows){
26     int i;
27     for (i = 0; i<rows; i++)
28         printf("%s\n", text[i]);
29 }
30
31 int get_length(char* string){
32     int len = 0;
33     while(*string++!=0) len++;
```

```
34      return len;
35  }
36
37  int count_spaces(char* string){
38      int count = 0;
39      while(*string++!=0) count+=(*string==' '?1:0);
40      return count;
41  }
42
43  int count_chars(char* string, char chr){
44      int count = 0;
45      while(*string++!=0) count+=(*string==chr?1:0);
46      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  /// что делает в библиотеке консольный ввод?
76  /// А: Я понимаю, что легче бы было сделать toString, но
        что за код без костылей?..
77  void get_string(char *str, int max){
78      int i = 0, ch;
79      while((ch = getchar()) != '\n')
```

```
80          if(str != NULL && i < max - 1)
81              str[i++] = ch;
82      if(str != NULL && i < max)
83          str[i] = 0;
84  }
85
86  //функция, возвращающее индекс num-ового вхождения символ
        а chr в строку str
87  int get_char_index(char* str, char chr, int num){
88      int i, temp = 0;
89      if(num>count_chars(str, chr))
90          return -1;
91      for(i = 0; i<get_length(str); i++)
92          if(str[i]==chr)
93              if(++temp == num)
94                  return i;
95      return i;
96
97  }
98
99  void spread_text(char** text, int rows){
100     int maxLength = get_max_string_length(text, rows);
101     int i;
102     for(i = 0; i<rows; i++)
103         if(get_length(text[i]) < maxLength){
104             int spaces = count_spaces(text[i]);
105             if (spaces==0)
106             {
107                 int count = maxLength-get_length(text[i])
                        ;
108                 text[i]=insert_chars(text[i],0,' ',count)
                        ;
109             }
110             else
111             {
112                 int count = maxLength - get_length(text[i
                        ]);
113                 int j;
114                 for(j = spaces; j>0; j--){
115                     //printf("%d\t\t%d\t%d - %d = %d\n",
                            i, spaces, maxLength, get_length(
                            text[i]), count);
116                     text[i] = insert_chars(text[i],
                            get_char_index(text[i],' ',j),' ',
                             count/spaces+(j>(spaces-count%
                            spaces)?1:0));
117                 }
118             }
119         }
```

```
120     printf("\n");
121 }
```

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 = (char**) malloc(rows*sizeof(char*));
11    int i;
12    for (i = 0; i<rows; i++)
13        text[i] = (char*) malloc(N*sizeof(char));
14    input_text(text, rows, N);
15    printf("\n");
16    print_text(text, rows);
17    spread_text(text, rows);
18    print_text(text, rows);
19
20    for (i = 0; i<rows; i++)
21        free(text[i]);
22    free (text);
23 }
```

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

```h
#ifndef MATRIX_CONSOLE_UI_H
#define MATRIX_CONSOLE_UI_H
void matrix_console_UI();
void printMatrix(int**, int, int);
#endif // MATRIX_CONSOLE_UI_H
```

Глава 5

Приложение к главам 1 - 4

5.1 Листинги

main.c

```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++;
    return(res>0 ? 0 : 1);
}

```

```
27 int main(int argc, char *argv[])
28 {
29     printf("\n\nПуть для терминала %s", argv[0]);
30     printf("\n\nSTART OF WORK\n");
31     if(argc>1){
32         if(strEquals(argv[1], "-bank")) bank_Console_UI()
               ;
33         if(strEquals(argv[1], "-home")) home_Console_UI()
               ;
34         if(strEquals(argv[1], "-cm2inch"))
               cm_to_inch_console();
35         if(strEquals(argv[1], "-matrix"))
               matrix_console_UI();
36         if(strEquals(argv[1], "-strings"))
               strings_console_UI();
37     }else{
38         printHelp();
39     }
40     printf("\nEND OF WORK\n\n");
41
42     return 0;
43 }
```

main.c

```
1  #include <QString>
2  #include <QtTest>
3  #include"bank.h"
4  #include"home.h"
5  #include"cm_to_inch.h"
6  #include"matrix.h"
7  #include"strings.h"
8
9  class TestTest : public QObject
10 {
11     Q_OBJECT
12
13 public:
14     TestTest();
15
16 private Q_SLOTS:
17     void testCase1();
18     void bank_test();
19     void home_test_1();
20     void home_test_2();
21     void cm2inch_test();
22     void matrix_test();
23     void strings_test();
24 };
```

```
25
26 TestTest::TestTest(){
27
28 }
29
30 void TestTest::testCase1(){
31     QVERIFY2(true, "Failure");
32 }
33
34 void TestTest::bank_test(){
35     QCOMPARE(bank(200, 25), 610.35f);
36     QCOMPARE(bank(10, 90), 247.60f);
37 }
38
39 void TestTest::home_test_1(){
40     Size s1, s2, s3;
41     s1.height = 40;
42     s1.width = 70;
43     s2.height = 30;
44     s2.width = 30;
45     s3.height = 30;
46     s3.width = 30;
47    QVERIFY2(home(s1,s2,s3), "Failure");
48 }
49
50 void TestTest::home_test_2(){
51     Size s1, s2, s3;
52     s1.height = 80;
53     s1.width = 30;
54     s2.height = 40;
55     s2.width = 50;
56     s3.height = 20;
57     s3.width = 20;
58    QVERIFY2(home(s1,s2,s3), "Failure");
59 }
60
61 void TestTest::cm2inch_test(){
62     QCOMPARE(cm_to_inch(2), 0.39);        // 1.00
63                                           // 0.79     2.00
64
65     QCOMPARE(cm_to_inch(5), 0.39);//          1.00
66                                           // 0.79     2.00
67                                           // 1.00     2.54
68                                           // 1.18     3.00
69                                           // 1.57     4.00
70                                           // 1.97     5.00
71 }
72
73 void TestTest::matrix_test(){
```

```
74      int** res = initializeMatrix(5, 7);
75      res[0][0] = 1;    res[0][1] = 2;    res[0][2] = 3;
            res[0][3] = 4;    res[0][4] = 5;
76      res[0][0] = 20;   res[0][1] = 21;   res[0][2] = 22;
            res[0][3] = 23;   res[0][4] = 6;
77      res[0][0] = 19;   res[0][1] = 32;   res[0][2] = 33;
            res[0][3] = 24;   res[0][4] = 7;
78      res[0][0] = 18;   res[0][1] = 31;   res[0][2] = 34;
            res[0][3] = 25;   res[0][4] = 8;
79      res[0][0] = 17;   res[0][1] = 30;   res[0][2] = 35;
            res[0][3] = 26;   res[0][4] = 9;
80      res[0][0] = 16;   res[0][1] = 29;   res[0][2] = 28;
            res[0][3] = 27;   res[0][4] = 10;
81      res[0][0] = 15;   res[0][1] = 14;   res[0][2] = 13;
            res[0][3] = 12;   res[0][4] = 11;
82
83      int** tmp = initializeMatrix(2, 3);
84      fillSpiralMatrix(tmp, 5, 7);
85
86      for (int i = 0; i < 5; ++i)
87      {
88          for(int j = 0; j < 7; j++)
89          {
90              QCOMPARE(tmp[i][j], res[i][j]);
91          }
92      }
93
94  }
95
96  void TestTest::strings_test(){
97      char** resText = initialize_text(5, 255);
98      resText[0] = "banana banana";
99      resText[1] = "ban na ba  na";
100     resText[2] = "b  an  ba  na";
101     resText[3] = "b   nab    na";
102     resText[4] = "       banana";
103     char** tmpText = initialize_text(5, 255);
104     for (int i = 0; i < 5; ++i)
105     {
106         for(int j = 0; j < 255; j++)
107         {
108             QCOMPARE(tmpText[i][j], resText[i][j]);
109         }
110     }
111
112 }
113
114 QTEST_APPLESS_MAIN(TestTest)
115
```

```
116 #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 Теоретические сведения

При разработке приложения был задействован язык 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

В ходе выполнения заданий мной были получены навыки работы с одним из основныч отличий C++ от C - инкапсуляцией.

Листинги

main.cpp

```
1  #include <iostream>
2  #include "set.h"
3
4  using namespace std;
5
6  int main()
7  {
8      set *s1 = new set();
9      set *s2 = new set();
10     s1->add(5);
11     s1->add(2);
12     s1->add(7);
13     s1->add(9);
14     s1->add(1);
15     s1->add(9);
16     s1->add(9);
17     s1->add(9);
18     s1->add(9);
19     s2->add(-1);
20     s2->add(-3);
21     s2->add(-5);
22     s2->add(2);
23
24     return 0;
25 }
```

node.cpp

```
1  #include "node.h"
2
3  Node::Node(int d, Node* n){
4      data = d;
5      next = n;
6  }
7  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      /// И они у вас в public доступе, кто угодно может по
           менять в любой момент
9      int data;
10     Node* next;
11     Node(int);
12     Node(int, Node*);
13     ~Node();
14 };
15
16 #endif // NODE_H
```

set.h

```
1  #ifndef SET_H
2  #define SET_H
3  #include <iostream>
4  #include "node.h"
5
6  /// Остальные классы у вас называются с большой буквы
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

Был получен опыт создания классов, атакже 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 " << (BankCPP().
           doWork(summa, percent)) << "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 = CmToInchCPP().cm_to_inch(i);
22          if (temp < tempInch)
23              cout << temp << "\t" << i << endl;
24          else
25          {
26              cout << tempInch << "\t" << CmToInchCPP().
                    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 "rectangle.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      int tempW, tempH;
18      cin >> tempW;
19      cin >> tempH;
20      Rectangle Area(tempW, tempH);
21      cin >> tempW;
22      cin >> tempH;
23      Rectangle Rect1(tempW, tempH);
24      cin >> tempW;
25      cin >> tempH;
26      Rectangle Rect2(tempW, tempH);
27
28      if (Area.canInsert(Rect1, Rect2))
29          cout << "Yes\n";
30      else
31          cout << "No\n";
32
33  }
```

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  #include <fstream>
6  using namespace std;
7  //MatrixConsoleUICPP matrixWorker; matrixWorker.doWork();
8  MatrixConsoleUICPP::MatrixConsoleUICPP()
9  {
10
```

```
}

void MatrixConsoleUICPP::printMatrix(MatrixCPP matrix){
    int i, j;
    for (i = 0; i<matrix.getHeight(); i++){
        for(j = 0; j<matrix.getWidth(); j++)
            cout << matrix.getCell(i, j) << " ";
        cout << endl;
    }
}

void MatrixConsoleUICPP::doWork(char* input_file_name,
    char* output_file_name){
    ifstream in(input_file_name);
    ofstream out(output_file_name);
    int m, n;

    in >> n;
    in >> m;

    MatrixCPP matrix(n, m);

    cout << matrix;
    out << matrix;

    in.close();
    out.close();
}
```

matrixconsoleuicpp.h

```
#ifndef MATRIXCONSOLEUICPP_H
#define MATRIXCONSOLEUICPP_H
#include<matrixcpp.h>

class MatrixConsoleUICPP
{
public:
    MatrixConsoleUICPP();
    void doWork(char*, char*);
    void printMatrix(MatrixCPP);
};

#endif // MATRIXCONSOLEUICPP_H
```

stringsconsoleuicpp.cpp

```
#include "stringsconsoleuicpp.h"
```

```
#include"stringscpp.h"
#define N 255

StringsConsoleUICPP::StringsConsoleUICPP()
{

}

void StringsConsoleUICPP::doWork(){
    int rows = 5;
    StringsCPP stringsWorker;
    char** text = stringsWorker.initialize_text(rows, N);
    stringsWorker.input_text(text, rows, N);
    stringsWorker.print_text(text, rows);
    stringsWorker.spread_text(text, rows);
    stringsWorker.print_text(text, rows);
}
```

stringsconsoleuicpp.h

```
#ifndef STRINGSCONSOLEUICPP_H
#define STRINGSCONSOLEUICPP_H


class StringsConsoleUICPP
{
public:
    StringsConsoleUICPP();
    void doWork();
};

#endif // STRINGSCONSOLEUICPP_H
```

bankcpp.cpp

```
#include "bankcpp.h"

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

bankcpp.h

```
#ifndef BANKCPP_H
#define BANKCPP_H

class BankCPP
{
public:
    static float doWork(float, float);
};

#endif // BANKCPP_H
```

cmtoinchcpp.cpp

```
#include "cmtoinchcpp.h"

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

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

cmtoinchcpp.h

```
#ifndef CMTOINCHCPP_H
#define CMTOINCHCPP_H

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

#endif // CMTOINCHCPP_H
```

matrixcpp.cpp

```
#include "matrixcpp.h"
#include <stdlib.h>

MatrixCPP::MatrixCPP(int height, int width)
{
    this->width = width;
```

```
7       this->height = height;
8       data=(int **)malloc(height*sizeof(int*));
9       for (int i = 0; i<height; i++)
10              data[i]=(int*)malloc(width*sizeof(int));
11      fillSpiralMatrix();
12  }
13
14  MatrixCPP::~MatrixCPP(){
15      for(int i = 0; i < height; i++)
16          delete data[i];
17      delete data;
18  }
19
20  void MatrixCPP::fillSpiralMatrix(){
21      int j, rows = 0, cols = 0, k = 1;
22      int horbeg = 0, horend = width-1, vertbeg = 0,
            vertend = height-1;
23      while(1){
24          for(j = horbeg; j<horend+1; j++)
25              data[horbeg][j] = k++;
26          if (++rows == height) return;
27          for(j = vertbeg+1; j<vertend+1; j++)
28              data[j][horend] = k++;
29          if (++cols == width) return;
30          for(j = horend-1; j>=horbeg; j--)
31              data[vertend][j] = k++;
32          if (++rows == height) return;
33          for(j = vertend-1; j>=vertbeg+1; j--)
34              data[j][horbeg] = k++;
35          if (++cols == width) return;
36          horbeg++; horend--; vertbeg++; vertend--;
37      }
38  }
39
40  int MatrixCPP::getCell(int y, int x){
41      if (x<0||y<0||x>=width||y>=height)
42          return 0;
43      return data[y][x];
44  }
45
46  int MatrixCPP::getHeight(){
47      return height;
48  }
49
50  int MatrixCPP::getWidth(){
51      return width;
52  }
53
54  ostream& operator<<(ostream& os, MatrixCPP& matrix){
```

```
55      for (int i = 0; i< matrix.getHeight(); i++){
56          for (int j = 0; j<matrix.getWidth(); j++)
57              os << matrix.getCell(i, j) << " ";
58          os << "\n";
59      }
60      return os;
61  }
```

matrixcpp.h

```
1  #ifndef MATRIXCPP_H
2  #define MATRIXCPP_H
3  #include <iostream>
4  using namespace std;
5
6  class MatrixCPP
7  {
8  private:
9      int** data;
10     int width;
11     int height;
12     void fillSpiralMatrix();
13 public:
14     MatrixCPP(int, int);
15     ~MatrixCPP();
16     int getCell(int, int);
17     int getHeight();
18     int getWidth();
19     friend ostream& operator<<(ostream& os, MatrixCPP &
           matrix);
20 };
21
22 #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  /// разбираться не будем, но похоже, можно все методы сде
       лать static
5  /// Производит впечатление бредового класса
6  class StringsCPP
7  {
8  public:
9
10     StringsCPP();
11     static void get_string(char*, int);
12     static char** initialize_text(int, int);
13     static char* initialize_string(int);
14     static void input_text(char**, int, int);
15     static void print_text(char**, int);
```

```
16      static int get_length(char*);
17      static int get_max_string_length(char**, int);
18      static int count_spaces(char*);
19      static char* insert_char(char*, int, char);
20      static char* insert_chars(char*, int, char, int);
21      static void spread_text(char**, int);
22      static int get_char_index(char*, char, int);
23      static int count_chars(char*, char);
24  };
25
26  #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"
```