Программирование

А. Д. Орова

18 февраля 2016 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 для 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 стандартной библиотеки для ввода и вывода информации, а также для выполнения необходимых математических вычислений.

1.1.3 Проектирование

В ходе проектирования решено выделить две функции:

- double compoundInterest(double summa, double percent, int n);
- void bank_console_UI();

1. **compoundInterest** 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 double: *summa* и *percent*. В первую переменную передается первоначальная сумма, которую пользователь желает положить в банк, во вторую - процент, под который кладутся деньги. Фунцкция возвращает итоговую сумму.

2. **bank_console_ui** В этой функции реализованно взаимодействие с пользователем. В ней выполняется считывание двух значений из консоли. Если данные введены правильно, то выполняется функция **compoundInterest**, 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 4.8.4-2 ubuntu 1 14.04), операционная система Ubuntu 14.04.

Для тестирования программы был проведен динамический, а также статический анализ кода. Для динамического анализа использовались ручные и модульные тесты, а для статического - утилита Cppcheck 1.61.

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

3. **Статический анализ** Утилита *cppcheck* не выдала никаких предупреждений.

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

```
1  #include"bank.h"
2
3  double compoundInterest(double summa, double percent, int
       n)
4  {
5      double result = summa;
6      int i;
7      for (i = 0; i < n; i++)
8          result *= (100 + percent) / 100;
9      return result;
10 }
```

bank_console_ui.c

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

void bank_Console_UI()
{
    double 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",
        compoundInterest(summa, percent, 5));
}
```

bank.h.c

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

#ifdef __cplusplus
extern "C"{
#endif


double compoundInterest(double , double, int);


#ifdef __cplusplus
}
#endif

#endif // BANK_H
```

bank_console_ui.h

```c
#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 на b метров разместить 2 дома размером 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 два дома на участке, и на каком участке. То есть если один дом не будет перекрывать другой, также находящийся на данной территории, то программа выдаст положительный ответ. Иначе, если физически невозможно расположить два дома на однй территории, то программа выдаст отрицательный ответ. Была использована конструкция if...else, а также функции стандартной библиотеки для ввода и вывода.

1.2.3 Проектирование

Для решения данной задачи используются шесть 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 три переменных типа **Size** (структура, созданная для удобства в заголовочном файле, обладает полями *width* и *height*). 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 gcc version 4.8.4 (Ubuntu 4.8.4-2 ubuntu 1 14.04), операционная система Ubuntu 14.04.

Для тестирования программы был проведен динамический, а также статический анализ кода. Для динамического анализа использовались ручные и модульные тесты, а для статического - утилита Cppcheck 1.61.

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

3. **Статический анализ**

   Утилита *cppcheck* не выдала никаких предупреждений.

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>
```

```c
#include"home.h"
#include"home_console_ui.h"

void home_Console_UI()
{
    printf(" Homework #2: Input, output and cycles\n\n");
    printf("\n");
    printf("Exercise #2 \n\n");
    printf("Please, input length (horizontal)  of area a,
        b, p, q and r, s: \n");
    Size home1, home2, area;
    scanf("%lf", &area.width);
    scanf("%lf", &area.height);
    scanf("%lf", &home1.width);
    scanf("%lf", &home1.height);
    scanf("%lf", &home2.width);
    scanf("%lf", &home2.height);

    if (home(area, home1, home2) == 1)
        printf("Yes\n");
    else
        printf("No\n");

}
```

home.h

```c
#ifndef HOME_H
#define HOME_H

typedef struct{
    double width;
    double height;
}Size;

#ifdef __cplusplus
extern "C"{
#endif

    int home(Size, Size, Size);

#ifdef __cplusplus
}
#endif

#endif // HOME_H
```

home_console_ui.h

```
#ifndef HOME_CONSOLE_UI_H
#define HOME_CONSOLE_UI_H
void home_Console_UI();
#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 введеное с клавиатуры поделить на эквивалент, равный 2,54. Для того чтобы перевести из дюймов в сантиметры - соответственно умножить на 2,54. Для выполнения задания использоваилсь функции стандартной библиотеки для ввода и вывода.

2.1.3 Проектирование

В ходе проектирования было выделено три функции:

- double cm_to_inch(double);
- double inch_to_cm(double);
- void cm_to_inch_console();

1. **cm_to_inch** Функция возвращает переданное ей количество сантиметров, поделенное на эквивалент. Параметром функции является переменная типа double (количество сантиметров).
2. **inch_to_cm** Функция возвращает переданное ей количество дюймов, умноженное на эквивалент. Параметром функции является переменная типа double (количество дюймов).
3. **inch_to_cm_console_ui** В Функции реализованно взаимодействие с пользователем. В ней выполняется считывание из консоли числа, равного количеству сантиметров, которое 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 gcc version 4.8.4 (Ubuntu 4.8.4-2 ubuntu 1 14.04), операционная система Ubuntu 14.04.

Для тестирования программы был проведен динамический, а также статический анализ кода. Для динамического анализа использовались ручные и модульные тесты, а для статического - утилита Cppcheck 1.61.

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

3. **Статичесикий анализ**

Утилита *cppcheck* не выдала никаких предупреждений.

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

```
#ifndef CM_TO_INCH_H
#define CM_TO_INCH_H

#ifdef __cplusplus
extern "C"{
#endif

double cm_to_inch(double);
double inch_to_cm(double);

#ifdef __cplusplus
}
#endif

#endif // CM_TO_INCH_H
```

cm_to_inch_console_ui.h

```
#ifndef CM_TO_INCH_CONSOLE_UI_H
#define CM_TO_INCH_CONSOLE_UI_H
void cm_to_inch_console();
#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 стандартной бибилиотеки для динамического выделения и освобождения памяти, а также для ввода, вывода информации и работы с файлами.

3.1.3 Проектирование

В ходе проектирования были выделены три функции:

- void fillSpiralMatrix(int**, int, int);
- void matrix_console_UI(char*, char*);
- void printMatrix(int**, int, int);

1. **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.

2. **matrix_console_UI** Функция, параметрами которой, являются указатели на имена входного ивыходного файлов. Функция реализует взаимодействие с пользователем, а также осуществляет работу с файлами, выделением и освобождением памяти. Функция считывает с консоли m и n (размеры матрицы), в ней осущетствляется вызов функций **fillSpiralMatrix** и **printMatrix** из под проекта **lib**.

3. **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 gcc version 4.8.4 (Ubuntu 4.8.4-2 ubuntu 1 14.04), операционная система Ubuntu 14.04.

Для тестирования программы был проведен динамический, а также статический анализ кода. Для динамического анализа использовались ручные и модульные тесты, а для статического - утилита Cppcheck 1.61.

3.1.5 Тестовый план и результаты тестирования

1. **Ручные тесты**

   II тест

   Входные данные: 2 4

   Выходные данные:

   1 2
   8 3
   7 4
   6 5

   Результат: Тест успешно пройден

2. **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

3. **Статический анализ**

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

```
#include <stdlib.h>

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;

    printf("input n");
    scanf("%d", &n);
    printf("input m");
    scanf("%d", &m);
    fscanf(in, "%i", &k);

    int** array = (int **)malloc(n*sizeof(int*));

    for (i = 0; i < n; ++i)
        array[i] = (int*) malloc(n * sizeof(int));

    for (i = 0; i < n; ++i)
        for (j = 0; j < n; ++j)
            fscanf(in, "%i\n", &array[i][j]);

    fclose(in);

    fillSpiralMatrix(array, n, m);
    printMatrix(array, n, m);

    for (i = 0; i < n; ++i)
    {
        for (j = 0; j < n; ++j)
            fprintf(out, "%i ", array[i][j]);
        fprintf(out, "\n");
    }

    for (i = 0; i < n; ++i)
        free(array[i]);
    free(array);

    fclose(out);;
}

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  void fillSpiralMatrix(int**, int, int);
9
10
11 #ifdef __cplusplus
12 }
13 #endif
14 #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

Текст, состоящий из ряда строк, выровнять по правому краю так, чтобы каждая строка завершалась непробельным символом (буква, цифра, знак препинания). Выравнивание осуществить, вставляя дополнительные пробелы между словами – равномерно по всей строке.

4.1.2 Теоретические сведения

Для выполнения данного задания необходимо знать, какое количество пробелов нужно вставить в каком месте, в какой строке. Для реализации данной задачи использовались функции стандартной библиотеки.

4.1.3 Проектирование

В ходе проектирования было решено выделить 11 функций.

- int get_max_string_length(char**, int);
- int count_spaces(char*);
- char* insert_char(char*, int, char);
- char* insert_chars(char*, int, char, int);
- void spread_text(char**, int);
- int get_char_index(char*, char, int);

- int count_chars(char*, char);
- void strings_console_UI();
- void print_text(char**, int);
- void get_string(char*, int);
- void input_text(char**, int, int);

Более подробное описание опиание функций.

1. **get_max_string_length**
   Функция возвращает количество символов.
2. **count_spaces**
   Функция возвращаяет количество пробельных символов.
3. **insert_char**
   Функция вставляет символ.
4. **insert_chars**
   Функция вставляет символы.
5. **spread_text**
   Функция вставляет в нужные места пробельные символы.
6. **get_char_index**
   Функция, возвращающее индекс num-ового вхождения символа chr в строку str.
7. **count_chars**
   Функция считает количество символов.
8. **strings_console_UI**
   Функция взаимодействия с ползьвателем.
9. **print_text**
   Функция печати текста на экран в консоль.
10. **get_string**
    Функция считывания строки.
11. **input_text** Функция получени строки.

4.1.4 Описание тестового стенда и методики тестирования

Среда разработки QtCreator 3.5.0, компилятор gcc version 4.8.4 (Ubuntu 4.8.4-2 ubuntu 1 14.04), операционная система Ubuntu 14.04.

Для тестирования программы был проведен динамический, а также статический анализ кода. Для динамического анализа использовались ручные и модульные тесты, а для статического - утилита Cppcheck 1.61.

4.1.5 Тестовый план и результаты тестирования

1. **Модульный тест**

   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.

4.1.6 Выводы

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

Листинги

strings.c

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

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;
}

int get_max_string_length(char** text, int rows){
    unsigned int max = strlen(text[0]);
    int i;
    for(i = 1; i<rows; i++)
        max = (strlen(text[i])>max?strlen(text[i]):max);
    return max;
}

char* insert_char(char* str, int place, char chr){
    int i;
    char* result = (char*) malloc((strlen(str)+1)*sizeof(
       char));
    for(i = 0; i<place; i++)
        result[i] = str[i];
    result[place] = chr;
    for(i = place; i<strlen(str); i++)
        result[i+1] = str[i];
    result[strlen(str)+1] = 0;
    return result;
}

char* insert_chars(char* str, int place, char chr, int
   count){
    int i;
    for(i = 0; i<count; i++){
        char *tmp = str;
        str = insert_char(str,place,chr);
        free(tmp);
    }
```

```
45      return str;
46  }
47
48
49  //функция, возвращающее индекс num-ого вхождения символ
        а chr в строку str
50  int get_char_index(char* str, char chr, int num){
51      int i, temp = 0;
52      if(num>count_chars(str, chr))
53          return -1;
54      for(i = 0; i<strlen(str); i++)
55          if(str[i]==chr)
56              if(++temp == num)
57                  return i;
58      return i;
59
60  }
61
62  void spread_text(char** text, int rows){
63      int maxLength = get_max_string_length(text, rows);
64      int i;
65      char* tmp;
66      for(i = 0; i<rows; i++)
67          if(strlen(text[i]) < maxLength){
68              int spaces = count_spaces(text[i]);
69              if (spaces==0)
70              {
71                  int count = maxLength-strlen(text[i]);
72                  tmp = text[i];
73                  text[i]=insert_chars(text[i],0,' ',count)
                        ;
74                  free(tmp);
75              }
76              else
77              {
78                  int count = maxLength - strlen(text[i]);
79                  int j;
80                  for(j = spaces; j>0; j--){
81                      //printf("%d\t\t%d\t%d - %d = %d\n",
                            i, spaces, maxLength, get_length(
                            text[i]), count);
82                      text[i] = insert_chars(text[i],
                            get_char_index(text[i],' ',j),' ',
                             count/spaces+(j>(spaces-count%
                            spaces)?1:0));
83                  }
84              }
85          }
86      printf("\n");
```

```
87 }
```

strings_console_ui.c

```
1  #include<stdio.h>
2  #include<stdlib.h>
3  #include"strings_console_ui.h"
4  #include"strings.h"
5  #define N 255
6
7  void strings_console_UI(){
8      int rows = 5;
9      char** text = (char**) malloc(rows*sizeof(char*));
10     int i;
11     for (i = 0; i<rows; i++)
12         text[i] = (char*) malloc(N*sizeof(char));
13
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 }
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 void input_text(char** text, int rows, int max){
32     int i;
33     //getchar();//считываем предыдущий enter
34     for (i = 0; i<rows; i++)
35         get_string(text[i], max);
36 }
37
38 void get_string(char *str, int max){
39     int i = 0, ch;
40     while((ch = getchar()) != '\n')
41         if(str != NULL && i < max - 1)
42             str[i++] = ch;
43     if(str != NULL && i < max)
44         str[i] = 0;
45 }
```

strings.h

```
#ifndef MATRIX_H
#define MATRIX_H

#ifdef __cplusplus
extern "C"{
#endif

void fillSpiralMatrix(int**, int, int);


#ifdef __cplusplus
}
#endif
#endif // MATRIX_H
```

strings_console_ui.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

```
#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(compoundInterest(200, 25, 5), 610.35f);
36     QCOMPARE(compoundInterest(10, 90, 5), 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 i, j;
75      int** res = (int **)malloc(7*sizeof(int*));
76
77      for (i = 0; i < 7; ++i)
78          res[i] = (int*) malloc(5 * sizeof(int));
79      res[0][0] = 1;    res[0][1] = 2;    res[0][2] = 3;
            res[0][3] = 4;    res[0][4] = 5;
80      res[0][0] = 20;   res[0][1] = 21;   res[0][2] = 22;
            res[0][3] = 23;   res[0][4] = 6;
81      res[0][0] = 19;   res[0][1] = 32;   res[0][2] = 33;
            res[0][3] = 24;   res[0][4] = 7;
82      res[0][0] = 18;   res[0][1] = 31;   res[0][2] = 34;
            res[0][3] = 25;   res[0][4] = 8;
83      res[0][0] = 17;   res[0][1] = 30;   res[0][2] = 35;
            res[0][3] = 26;   res[0][4] = 9;
84      res[0][0] = 16;   res[0][1] = 29;   res[0][2] = 28;
            res[0][3] = 27;   res[0][4] = 10;
85      res[0][0] = 15;   res[0][1] = 14;   res[0][2] = 13;
            res[0][3] = 12;   res[0][4] = 11;
86
87      int** tmp = (int **)malloc(7*sizeof(int*));
88
89      for (i = 0; i < 7; ++i)
90          tmp[i] = (int*) malloc(5 * sizeof(int));
91      fillSpiralMatrix(tmp, 5, 7);
92
93      for (i = 0; i < 5; ++i)
94      {
95          for(j = 0; j < 7; j++)
96          {
97              QCOMPARE(tmp[i][j], res[i][j]);
98          }
99      }
100
101     for(i = 0; i < 7; i++){
102         free(tmp[i]);
103         free(res[i]);
104     }
105     free(tmp);
106     free(res);
107
108 }
109
110 void TestTest::strings_test(){
111
112     int i, j;
113     char** tmpText = (char**) malloc(5*sizeof(char*));
114     for (i = 0; i<5; i++)
115         tmpText[i] = (char*) malloc(255*sizeof(char));
```

```
116     tmpText[0] = "banana banana";
117     tmpText[1] = "ban na ba na";
118     tmpText[2] = "b an ba na";
119     tmpText[3] = "b nab na";
120     tmpText[4] = "banana";
121 
122     char** resText = (char**) malloc(5*sizeof(char*));
123     for (i = 0; i<5; i++)
124         resText[i] = (char*) malloc(255*sizeof(char));
125     resText[0] = "banana banana";
126     resText[1] = "ban na ba  na";
127     resText[2] = "b  an  ba  na";
128     resText[3] = "b   nab    na";
129     resText[4] = "       banana";
130     spread_text(tmpText, 5);
131     for (i = 0; i < 5; ++i)
132     {
133         for(j = 0; j < 255; j++)
134         {
135             QCOMPARE(tmpText[i][j], resText[i][j]);
136         }
137     }
138 
139 
140     for (i = 0; i<5; i++){
141         free(resText[i]);
142         free(tmpText[i]);
143     }
144     free (resText);
145     free (tmpText);
146 
147 }
148 
149 QTEST_APPLESS_MAIN(TestTest)
150 
151 #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&)* - конструтор 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 gcc version 4.8.4 (Ubuntu 4.8.4-2 ubuntu 1 14.04), операционная система Ubuntu 14.04.

Для тестирования программы был проведен динамический, а также статический анализ кода. Для динамического анализа использовались ручные и модульные тесты, а для статического - утилита Cppcheck 1.61.

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

```cpp
#include "node.h"

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

void Node::setData(int data){
    this->data = data;
}

void Node::setNext(Node* next){
    this->next = next;
}

int Node::getData(){
    return data;
}

Node* Node::getNext(){
    return next;
}
```

set.cpp

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

Set::Set()
{
    root = nullptr;
}
Set::Set(const Set& source)
{
    Node *temp = source.root;
    root = nullptr;
    while(temp!=nullptr){
        addToBeg(temp->getData());
        temp = temp->getNext();
    }
}

```

```
18 Set::~Set()
19 {
20     Node *temp;
21     while(root != nullptr)
22     {
23         temp = root;
24         root = root->getNext();
25         temp->~Node();
26     }
27 }
28
29 Set Set::copy(Set source)
30 {
31     Node *temp = source.root;
32     Set *result = new Set();
33     while(temp!=nullptr){
34         result->addToBeg(temp->getData());
35         temp = temp->getNext();
36     }
37     return *result;
38 }
39
40 void Set::addToBeg(int data)
41 {
42     root = new Node(data, root);
43 }
44
45 Node* Set::searchByKey(int data)
46 {
47     Node *temp=root;
48     while(temp!=nullptr && temp->getData()!=data)
49         temp = temp->getNext();
50     return temp;
51 }
52
53 void Set::add(int data)
54 {
55     if (searchByKey(data)!=nullptr)
56         return;
57     addToBeg(data);
58 }
59
60 void Set::add(Set added)
61 {
62     Node *temp = added.root;
63     while(temp!=nullptr){
64         add(temp->getData());
65         temp = temp->getNext();
66     }
```

```
67 }
68
69 bool Set::contains(Set s)
70 {
71     Node *temp = s.root;
72     bool result = true;
73     while(temp!=nullptr){
74         result = result && contains(temp->getData());
75         temp = temp->getNext();
76     }
77     return result;
78 }
79
80 bool Set::contains(int data)
81 {
82     return (searchByKey(data)!=nullptr);
83 }
84
85 Set Set::intersect(Set s)
86 {
87     Set *result = new Set();
88     Node *temp = root;
89     while(temp!=nullptr)
90     {
91         if (s.contains(temp->getData()))
92             result->add(temp->getData());
93         temp = temp->getNext();
94     }
95     return *result;
96 }
97
98 int Set::count()
99 {
100     Node *temp = root;
101     int result = 0;
102     while(temp!=nullptr)
103     {
104         result++;
105         temp = temp->getNext();
106     }
107     return result;
108 }
109
110 bool Set::isEmpty()
111 {
112     return (count()<=0);
113 }
```

node.h

```
#ifndef NODE_H
#define NODE_H


class Node
{
private:
    int data;
    Node* next;
public:
    Node(int);
    Node(int, Node*);
    ~Node();
    void setData(int data);
    void setNext(Node* next);
    int getData();
    Node* getNext();
};

#endif // NODE_H
```

set.h

```
#ifndef SET_H
#define SET_H
#include <iostream>
#include "node.h"

class Set
{
public:

    Set();
    Set(const Set& source);
    ~Set();
    void add(int);
    void add(Set);
    bool contains(int);
    bool contains(Set);
    Set copy(Set);
    Set intersect(Set s);
    int count();
    bool isEmpty();

private:
    Node *root;
    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 "exception.h"
4  #include <iostream>
5  using namespace std;
6
7  void BankConsoleUICPP::doWork()
8  {
9      double summa, percent;
10     cout << "\tHomework #1: Input, output and cycles" <<
           endl;
11     cout << "Exercise #1" << endl << endl;
12     cout << "Please, input how much money You want to put
           to the bank:" << endl;
13     cin >> summa;
```

```
14      cout << "    Please, input what is the percent at
            Your bank:" << endl;
15      cin >> percent;
16      //BankCPP bankWorker(summa, percent);
17      try{
18          cout << "After 5 years You will have " << (
                BankCPP().compoundInterest(summa, percent)) <<
                 "rubbles" << endl;
19      }
20      catch(UnderNullExceptionSumma& exp){
21          cout << "Summa is under Null. Current size: " <<
                exp.GetSumma()
22              << endl;
23      }
24
25      catch(MoreThanHundred& exp){
26          cout << "Percent is more than hundred. Current
                size: " << exp.GetPercent()
27              << endl;
28      }
29  }
```

bankconsoleuicpp.h

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

cmtoinchconsoleuicpp.cpp

```
1  #include "cmtoinchconsoleuicpp.h"
2  #include "cmtoinchcpp.h"
3  #include "exception.h"
4  #include <iostream>
5  using namespace std;
6
7  void CmToInchConsoleUICPP::doWork()
8  {
9      int a;
10     cout << "Input cm";
```

```
11      cin >> a;
12      double i, temp, tempInch = 1;
13      //CmToInchCPP cmToInchWorker;
14      try{
15          for (i = 1; i<=a; i++){
16              temp = CmToInchCPP().cm_to_inch(i);
17              if (temp < tempInch)
18                  cout << temp << "\t" << i << endl;
19              else{
20                  cout << tempInch << "\t" << CmToInchCPP()
                        .inch_to_cm(tempInch) << endl;
21                  i--;
22                  tempInch++;
23                  }
24              }
25          }
26      catch(UnderNullExceptionCm& exp){
27          cout << "Cm is under Null. Current size: " << exp
                .GetCm()
28                << endl;
29      }
30  }
```

cmtoinchconsoleuicpp.h

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

homeconsoleuicpp.cpp

```
1  #include "homeconsoleuicpp.h"
2  #include "rectangle.h"
3  #include <iostream>
4  #include "exception.h"
5  using namespace std;
6
7  void HomeConsoleUICPP::doWork()
8  {
9      cout << "Homework #2: Input, output and cycles\n\n\n"
           ;
```

```
10      cout << "Exercise #2 \n\n";
11      cout << "Please, input length (horizontal)  of area a
            , b, p, q and r, s: \n";
12      double tempW, tempH;
13      cin >> tempW;
14      cin >> tempH;
15      Rectangle Area(tempW, tempH);
16      cin >> tempW;
17      cin >> tempH;
18      Rectangle Rect1(tempW, tempH);
19      cin >> tempW;
20      cin >> tempH;
21      Rectangle Rect2(tempW, tempH);
22
23      try{
24          if (Area.canInsert(Rect1, Rect2))
25              cout << "Yes\n";
26          else
27              cout << "No\n";
28      }
29      catch(UnderNullExceptionH& exp){
30          cout << "Height is under Null. Current size: " <<
                exp.GetHeight()
31               << endl;
32      }
33      catch(UnderNullExceptionW& exp){
34          cout << "Width is under Null. Current size: " <<
               exp.GetWidth()
35               << endl;
36      }
37
38  }
```

homeconsoleuicpp.h

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

matrixconsoleuicpp.cpp

```cpp
#include "matrixconsoleuicpp.h"
#include "matrixcpp.h"
#include <stdio.h>
#include <iostream>
#include <fstream>
using namespace std;

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

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

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

#endif // MATRIXCONSOLEUICPP_H
```

stringsconsoleuicpp.cpp

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

void StringsConsoleUICPP::doWork(){
    int n;
    cout << "Input number of lines ";
    cin >> n;
    cout << "Input text\n";
    string* text = new string[n];
    for (int i = 0 ; i< n ; i++)
        cin >> text[i];
    StringsCPP().spread_text(text, n);
    for (int i = 0; i< n; i++)
        cout << text[i];
}
```

stringsconsoleuicpp.h

```
#include <iostream>
using namespace std;
#ifndef STRINGSCONSOLEUICPP_H
#define STRINGSCONSOLEUICPP_H


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

#endif // STRINGSCONSOLEUICPP_H
```

bankcpp.cpp

```
#include "exception.h"
#include "bankcpp.h"

double BankCPP::compoundInterest(const double summa,
    const double percent)
{
    if(summa < 0)
        throw UnderNullExceptionSumma(summa);
    if(percent > 100)
        throw MoreThanHundred(percent);
    double result = summa;
    int i;
```

```
12      for (i = 0; i < 5; i++)
13          result *= (100 + percent) / 100;
14      return result;
15  }
```

bankcpp.h

```
1  #ifndef BANKCPP_H
2  #define BANKCPP_H
3
4  class BankCPP
5  {
6  public:
7      static double compoundInterest(double, double);
8  };
9
10 #endif // BANKCPP_H
```

cmtoinchcpp.cpp

```
1  #include "cmtoinchcpp.h"
2  #include "exception.h"
3
4  double CmToInchCPP::cm_to_inch(const double cm)
5  {
6      if(cm < 0)
7          throw UnderNullExceptionCm(cm);
8      return (cm/KOEFF);
9  }
10
11 double CmToInchCPP::inch_to_cm(const double inch)
12 {
13     return (inch*KOEFF);
14 }
```

cmtoinchcpp.h

```
1  #ifndef CMTOINCHCPP_H
2  #define CMTOINCHCPP_H
3
4  class CmToInchCPP
5  {
6  private:
7      static const double KOEFF = 2.54f;
8  public:
9      static double cm_to_inch(double);
10     static double inch_to_cm(double);
11 };
```

```
12
13 #endif // CMTOINCHCPP_H
```

matrixcpp.cpp

```
1  #include "matrixcpp.h"
2  #include <stdlib.h>
3
4  MatrixCPP::MatrixCPP(const int height, const int width)
5  {
6      this->width = width;
7      this->height = height;
8      data = new int*[height];
9      for (int i = 0; i<height; i++)
10         data[i] = new int[width];
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(const int y, const int x){
41     if (x<0||y<0||x>=width||y>=height)
42         return 0;
43     return data[y][x];
```

```
}

int MatrixCPP::getHeight(){
    return height;
}

int MatrixCPP::getWidth(){
    return width;
}

ostream& operator<<(ostream& os, MatrixCPP& matrix){
    for (int i = 0; i< matrix.getHeight(); i++){
        for (int j = 0; j<matrix.getWidth(); j++)
            os << matrix.getCell(i, j) << " ";
        os << "\n";
    }
    return os;
}
```

matrixcpp.h

```
#ifndef MATRIXCPP_H
#define MATRIXCPP_H
#include <iostream>
using namespace std;

class MatrixCPP
{
private:
    int** data;
    int width;
    int height;
    void fillSpiralMatrix();
public:
    MatrixCPP(int height, int width);
    ~MatrixCPP();
    int getCell(int y, int x);
    int getHeight();
    int getWidth();
    friend ostream& operator<<(ostream& os, MatrixCPP &
        matrix);
};

#endif // MATRIXCPP_H
```

stringscpp.cpp

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

```
2  #include<stdlib.h>
3  #include "stringscpp.h"
4
5  string StringsCPP::insert_spaces(string str, int place,
       int count){
6      string result, tmp = "";
7      result.append(str);
8      for (int i = 0; i<count; i++)
9          tmp.append(" ");
10     result.insert(place, tmp);
11     return result;
12 }
13 int StringsCPP::count_spaces(string str){
14     int count = 0;
15     for (int i = 0; i<str.length(); i++)
16         if (str[i] == ' ')
17             count++;
18     return count;
19 }
20 int StringsCPP::get_max_string_length(string* text, int
       lines){
21     int max = text[0].length();
22     for(int i = 1; i<lines; i++)
23         max = (text[i].length()>max?text[i].length():max)
               ;
24     return max;
25 }
26 void StringsCPP::spread_text(string* text, int lines){
27     int maxLength = get_max_string_length(text, lines);
28     int i;
29     for(i = 0; i<lines; i++)
30         if(text[i].length()< maxLength){
31             int spaces = count_spaces(text[i]);
32             if (spaces==0)
33             {
34                 int count = maxLength - text[i].length();
35                 text[i] = insert_spaces(text[i], 0 ,
                       count);
36             }
37             else
38             {
39                 int lastSpace = text[i].find_last_of(' ')
                       ;
40                 int j = spaces;
41                 int count = maxLength - text[i].length();
42                 while(lastSpace != -1 && j != 0){
43                     text[i] = insert_spaces(text[i],
                           lastSpace, count/spaces+(j>(spaces
                           -count%spaces)?1:0));
```

```
                    j--;
                    lastSpace = text[i].find_last_of(' ',
                        lastSpace-1);
                }
            }
        }
}
```

stringscpp.h

```
#include <string.h>
#ifndef STRINGSCPP_H
#define STRINGSCPP_H
#include <string>
using namespace std;
class StringsCPP
{
private:
    static int get_max_string_length(string* text, int
        lines);
    static string insert_spaces(string str, int place,
        int count);
    static int count_spaces(string str);
public:
    static void spread_text(string* text, int lines);
};

#endif // STRINGSCPP_H
```

tst_testcpptest.cpp

```
#include <QString>
#include <QtTest>
#include "bankcpp.h"
#include "cmtoinchcpp.h"
#include "matrixcpp.h"
#include "stringscpp.h"
#include "exception.h"
class TestCPPTest : public QObject
{
    Q_OBJECT

public:
    TestCPPTest();

private Q_SLOTS:
    void bank_test();
    void home_test();
```

```
18      void cm2inch_test();
19      void matrix_test();
20      void strings_test();
21  };
22
23  TestCPPTest::TestCPPTest(){
24
25  }
26
27  void TestCPPTest::bank_test(){
28      QCOMPARE(BankCPP().compoundInterest(200, 25), 610.35f
            );
29      QCOMPARE(BankCPP().compoundInterest(10, 90), 247.60f)
            ;
30      QVERIFY_EXCEPTION_THROWN(BankCPP::compoundInterest
            (-100, 50), UnderNullExceptionSumma);
31      QVERIFY_EXCEPTION_THROWN(BankCPP::compoundInterest
            (100, 105), MoreThanHundred);
32  }
33
34  void TestCPPTest::home_test(){
35      Rectangle area(40,70);
36      Rectangle rect1(30,30);
37      Rectangle rect2(30,30);
38      QVERIFY2(area.canInsert(rect1, rect2), "Failure");
39  }
40
41  void TestCPPTest::cm2inch_test(){
42      QCOMPARE(CmToInchCPP().cm_to_inch(2), 0.39);        //
            1.00
43                                      // 0.79     2.00
44
45      QCOMPARE(CmToInchCPP().cm_to_inch(5), 0.39);//
                1.00
46                                      // 0.79     2.00
47                                      // 1.00     2.54
48                                      // 1.18     3.00
49                                      // 1.57     4.00
50                                      // 1.97     5.00
51      QVERIFY_EXCEPTION_THROWN(CmToInchCPP::cm_to_inch(-6),
            UnderNullExceptionCm);
52  }
53
54  void TestCPPTest::matrix_test(){
55      int res[7][5] {
56          { 1,  2,  3,  4,  5},
57          {20, 21, 22, 23,  6},
58          {19, 32, 33, 24,  7},
59          {18, 31, 34, 25,  8},
```

```
60          {17, 30, 35, 26,  9},
61          {16, 29, 28, 27, 10},
62          {15, 14, 13, 12, 11}
63      };
64
65      MatrixCPP matrix(7, 5);
66      for (int i = 0; i < 7; ++i)
67      {
68          for(int j = 0; j < 5; j++)
69          {
70              QCOMPARE(matrix.getCell(i, j), res[i][j]);
71          }
72      }
73
74  }
75
76  void TestCPPTest::strings_test(){
77      string tmpText[5];
78      tmpText[0] = "banana banana";
79      tmpText[1] = "ban na ba na";
80      tmpText[2] = "b an ba na";
81      tmpText[3] = "b nab na";
82      tmpText[4] = "banana";
83      (StringsCPP()).spread_text(tmpText, 5);
84
85      string resText[5];
86      resText[0] = "banana banana";
87      resText[1] = "ban na ba  na";
88      resText[2] = "b  an  ba  na";
89      resText[3] = "b   nab    na";
90      resText[4] = "       banana";
91
92      for (int i = 0; i < 5; ++i)
93      {
94          for (int j = 0; j < tmpText[i].length(); j++)
95          {
96              QCOMPARE(tmpText[i][j], resText[i][j]);
97          }
98      }
99  }
100
101 QTEST_APPLESS_MAIN(TestCPPTest)
102
103
104 #include "tst_testcpptest.moc"
```

exception.h

```
1  #ifndef EXCEPTION
```

```
2  #define EXCEPTION
3  #include <exception>
4  #include "bankcpp.h"
5  #include "cmtoinchcpp.h"
6  #include "rectangle.h"
7
8  class UnderNullExceptionSumma : public std::exception {
9  public:
10     UnderNullExceptionSumma(const int summa) : summa(
           summa) {}
11     int GetSumma() const {return summa;}
12 private:
13     int summa;
14 };
15
16 class MoreThanHundred : public std::exception {
17 public:
18     MoreThanHundred(const int percent) : percent(percent)
           {}
19     int GetPercent() const {return percent;}
20 private:
21     int percent;
22 };
23
24 class UnderNullExceptionCm : public std::exception {
25 public:
26     UnderNullExceptionCm(const int cm) : cm(cm) {}
27     int GetCm() const {return cm;}
28 private:
29     int cm;
30 };
31
32 class UnderNullExceptionW : public std::exception {
33 public:
34     UnderNullExceptionW(const int width) : width(width) {
           }
35     int GetWidth() const {return width;}
36 private:
37     int width;
38 };
39
40 class UnderNullExceptionH : public std::exception {
41 public:
42     UnderNullExceptionH(const int height) : height(height
           ) {}
43     int GetHeight() const {return height;}
44 private:
45     int height;
46 };
```

```
47
48 #endif // EXCEPTION
```