Программирование

Д. А. Курякин

24 декабря 2015 г.

Оглавление

Глава 1

Основные конструкции языка

1.1 Задание 1. Вклад в банке

1.1.1 Задание

Задана сумма и процентная ставка. Определить какяя сумма будет через 5 лет вклада в банке.

1.1.2 Теоретические сведения

При разработке приложения были задействована стандартная библиотека <stdio.h>.

Было решено, написать формулу которая вситает вклад в банке.

1.1.3 Проектирование

В ходе проектирования было решено выделить две функций, одна из которых отвечает за логику, а друкая за взаимодействие с пользователем.

1. **Логика**

   - `double investition_sum(double sum, double percent)`

     Эта функция вычисляет сумму. Она содержит два параметра вещественного типа - первоначальную сумму и процентную ставку. Возвращаемое значение имеет вещественный тип.

2. **Взаимодействие с пользователем**

- `void input()`

  Эта функция выводит в консоль результат функции. Она читает две вещественные переменные и выводит их в функцию

- `double investition_sum(double sum, double percent)`.

1.1.4 Описание тестового стенда и методики тестирования

Интегрированная среда разработки: Qt Creator 3.5.0 (opensource)
Компилятор: GCC 4.9.1 20140922 (Red Hat 4.9.1-10)
Операционная система: Debian Windows 8.1 64-бита

На всех стадиях разработки приложения проходило автоматическо етестирование. Осуществлялось посредством модульных тестов *Qt*, основанных на библиотеке *QTestLib*.

1.1.5 Тестовый план и результаты тестирования

- **Модульные тесты *Qt***

  Входные данные: sum = 1000, persent = 100
  Выходные данные: 32000.0
  Результат: Тест успешно пройден

1.1.6 Выводы

В ходе выполнения работы автор получил опыт создания многомодульного приложения с отделением логики от взаимодействия с пользователем. Укрепил навыки создания функций. А также научился тестировать программу с помощью автоматических тестов.

Листинги

investition.c

```
1 #include "main.h"
2 #include "investition_logic.h"
3
4 void input()
5 {
6     float sum, percent;
```

```
7
8       printf("Enter sum\n");
9       scanf("%f", &sum);
10      printf("Enter interest\n");
11      scanf("%f", &percent);
12
13      printf("Your amount in 5 years:  %f\n",
            investition_sum(sum, percent));
14  }
```

investition_logic.c)

```
1  double investition_sum(double sum, double percent)
2  {
3       int year;
4       for(year = 0; year <= 4; year++)
5       {
6            sum = sum + (sum * (percent / 100));
7       }
8       return(sum);
9  }
```

1.2 Задание 2. Рамзмещение двух домов на участке

1.2.1 Задание

Дано длина и ширина участка, длина и ширина двух домов. Проверить можнол ли разместить дома на участке.

1.2.2 Теоретические сведения

При разработке приложения были задействованы следующие конструкции языка: операторы ветвления **if** и **if-else-if** и ***struct***– и были использованы функции стандартной библиотеки *printf*, *scanf*, определенные в заголовочном файле *stdio.h*; функции, определенные в *stdlib.h*.

В дано было указаны длина и ширина участка, длина и ширина двух домов. В теории длина и ширина двух домов должна быть не больше, длины и ширины участка.

1.2.3 Проектирование

В ходе проектирования было решено выделить две функций, одна из которых отвечает за логику, а друкая за взаимодействие с пользователем.

1. **Логика**

   - `int calculation(struct poligon plot, struct poligon house1, struct poli`
     Эта функция вычисляет, помещаются ли два дома в участок. Тип возвращаемого значения – ***int*** – 1, если два дома помещаются, и 0 – в противном случае.

2. **Взаимодействие с пользователем**

   - `void issituated())`
     Эта функция осуществляет ввод из консоли размера участка и домов. Имеет параметры типа ***int*** - это размер участка и домов.

1.2.4 Описание тестового стенда и методики тестирования

Интегрированная среда разработки: Qt Creator 3.5.0 (opensource)
Компилятор: GCC 4.9.1 20140922 (Red Hat 4.9.1-10)
Операционная система: Debian Windows 8.1 64-бита

На всех стадиях разработки приложения проходило автоматическо етестирование. Осуществлялось посредством модульных тестов *Qt*, основанных на библиотеке *QTestLib*.

1.2.5 Тестовый план и результаты тестирования

Модульные тесты *Qt*

I тест

Входные данные: 20, 10, 10, 10, 10, 10
Выходные данные: 1
Результат: Тест успешно пройден

1.2.6 Выводы

В ходе выполнения работы автор получил опыт создания многомодульного приложения с отделением логики от взаимодействия с пользователем. Были укреплены навыки в создании структурных типов, тестировании программы с помощью модульных тестов.

Листинги

issituated.c)

```c
#include "main.h"
#include "issituated_logic.h"

void issituated()
{

    struct poligon plot;
    struct poligon house1;
    struct poligon house2;
    printf("Print land x, y:");
    scanf("%d%d", &plot.length, &plot.width);
    printf("Print the coordinates of house 1:");
    scanf("%d%d", &house1.length, &house1.width);
    printf("Print the coordinates of house 2:");
    scanf("%d%d", &house2.length, &house2.width);

    if (calculation(plot, house1, house2) == 1)
    {
        printf("There is enough space for who houses\n");
    }
    else
    {
        printf("There is not enough space for who houses\
           n");
    }
}
```

issituated_logic.c

```c
#include "issituated_logic.h"

int calculation(struct poligon plot, struct poligon
   house1, struct poligon house2)
{

    if (((plot.length >= (house1.length + house2.length))
         && (plot.width >= house2.width)
```

```
8           && (plot.width >= house1.width))
9               || ((plot.width >= (house2.width + house1.
                    width))
10                   && (plot.length >= house1.length)
11                   && (plot.length >= house2.length)))
12      {
13          return 1;
14      }
15      else
16      {
17          return 0;
18      }
19  }
```

Глава 2

Циклы

2.1 Задание 1. Умножение в столбик

2.1.1 Задание

Даны натуральные числа M и N. Вывести на экран процесс их умножения в столбик.

2.1.2 Теоритические сведения

При разработке приложения были задействованы следующие конструкции языка: оператор выбора, операторы ветвления **if** и **if-else-if**, оператор цикла с предусловием **while** и оператор цикла со счётчиком **for** – и были использованы функции стандартной библиотеки *scanf*, определённые в заголовочном файле *stdio.h*, *malloc*, *free*, определённые в *stdlib.h*.

При реализации алгоритма решения задачи, автор воспользовался методом умножения в столбик целых чисел. Конкретно в таком виде алгоритм используется в России, Франции, Бельгии и других странах.

2.1.3 Проектирование

В ходе проектирования было решено выделить 8 функций, 4 из которых отвечают за логику, а оставшиеся – за взаимодействие с пользователем.

1. **Логика**

   - `void consider_element_multiplier(int multiplier1, int multiplier2)`
     Эта функция отвечает за подсчёт количества цифр произведения.

- `int spase_mult_1(int copy_multiplier1)`
  Эта функция отвечает за подсчёт пробелов первого множителя.

- `int spase_mult_2(int copy2_multiplier2)`
  Эта функция отвечает за подсчёт пробелов вторго множителя.

- `void count_data(int multiplier1, int multiplier2, int spase_consider, i`
  Эта функция отвечает за печать в консоль.

2. Взаимодействие с пользователем

- `void print_spase1(int spase_consider, int spase_multiplier1)`
  Эта функция отвечает за вывод пробелов в первого множителя.

- `void print_spase1(int spase_consider, int spase_multiplier2)`
  Эта функция отвечает за вывод пробелов в второго множителя.
  - `void help_quotient(void);`
    Эта функция выводит в консоль информацию о том, как запускать приложение **Деление уголком** из параметров командной строки. Она не имеет аргументов. Возвращаемое значение - ***void***.
  - `void print_equal_symbol(int spase_consider)`
    эта функция отвечает за печать ровно которое разделяет промежуточные действия при умножении.
  - `void multiply()`
    Эта функция отвечает за взаимодействие с пользователем при запуске приложения в интерактивном режиме. Тоесть считывает с клавиатуры первое и второе слагаемое.

2.1.4 Описание тестового стенда и методики тестирования

Интегрированная среда разработки: Qt Creator 3.5.0 (opensource)

Компилятор: GCC 4.9.1 20140922 (Red Hat 4.9.1-10)

Операционная система: Debian Windows 8.1 64-бита

На всех стадиях разработки приложения проходило ручное тестирование .

2.1.5 Выводы

В ходе выполнения работы автор получил опыт в использовании циклов, обработке массивов и динамическом выделении памяти.

Листинги

matrix_ui.c

```
1  #include "main.h"
2  #include "matrix_logic.h"
3
4  void matrix()
5  {
6      int a;
7      puts("Enter matrix dimension");
8      scanf("%d", &a);
9      int **matrix1, **matrix2;
10     int i, j;
11     matrix1 = (int**)malloc(a*sizeof(int*));
12     for(i = 0; i < a; i++)
13     {
14         matrix1[i] = (int*)malloc(a*sizeof(int));
15     }
16
17     matrix2 = (int**)malloc(a*sizeof(int*));
18     for(i = 0; i < a; i++)
19     {
20         matrix2[i] = (int*)malloc(a*sizeof(int));
21     }
22
23     FILE *file_matrix1 = fopen ("matrix1.txt", "r");
24     for (i = 0; i < a; i++)
25     {
26         for (j = 0; j < a; j++)
27         {
28             fscanf(file_matrix1, "%d ", &matrix1[i][j]);
29         }
30         fscanf(file_matrix1, "\n");
31     }
```

```
    fclose(file_matrix1);

    FILE *file_matrix2 = fopen ("matrix2.txt", "r");
    for (i = 0; i < a; i++)
    {
        for (j = 0; j < a; j++)
        {
            fscanf(file_matrix2, "%d ", &matrix2[i][j]);
        }
        fscanf(file_matrix2, "\n");
    }
    fclose(file_matrix2);

    if (are_matrixes_transposable(matrix1, matrix2, a))
    {
        puts("YES!!!");
    }
    else
    {
        puts("NO!!!");
    }
    for(i = 0; i < a; i++)
    {
        free(matrix1[i]);
        free(matrix2[i]);
    }
    free(matrix1);
    free(matrix2);
}
```

matrix_logic.c

```
int comparing_transport_areey_main(int **matrix1, int **
    matrix2, int a)
{
    int i , j, result = 1;
    int b = 0;
    for (i = 0; i < a; i++)
    {
        b++;
        for(j = 0; j < b; j++)
        {
            if (matrix1[i][j] != matrix2[j][i])
            {
                result = 0;
            }
        }
    }
    return result;
```

```
17 }
18
19 int comparing_transport_areey_secondary_diagonal(int**
       matrix1, int**matrix2, int  a)
20 {
21     int i, j, result = 1;
22     int b = a;
23     for (i = 0; i < a - 1; i++)
24     {
25         b--;
26         for(j = 0; j < b; j++)
27         {
28             if (matrix1[i][j] != matrix2[a-j-1][a-i-1])
29             {
30                 result = 0;
31             }
32         }
33     }
34     return result;
35 }
36 int are_matrixes_transposable(int** matrix1, int**
       matrix2, int a)
37 {
38     if (comparing_transport_areey_main(matrix1, matrix2,
           a) ||
39             comparing_transport_areey_secondary_diagonal(
                   matrix1, matrix2, a))
40     {
41         return 1;
42     }
43     else
44     {
45         return 0;
46     }
47 }
```

Глава 3

Матрицы

3.1 Задание 1. Транспортирование матрицы

3.1.1 Задание

Для двух заданных матриц $A(n, n)$ и $B(n, n)$ проверить, можно ли получить вторую из первой применением конечного числа (не более четырех) операций транспонирования относительно главной и побочной диагоналей

3.1.2 Теоритические сведения

При разработке приложения были задействованы следующие конструкции языка: оператор ветвления **if**, оператор цикла со счётчиком **for** – и были использованы функции стандартной библиотеки *fopen*, *fclose*, *fscanf*, *fprintf*, определённые в заголовочном файле *stdio.h*, *malloc*, *free*, определённые в *stdlib.h*.

Для реализации алгоритма решения задачи, автор оттранспортировал первую матрицу поглавной и побочной диагонали, и сравнил со второй матрицей.

3.1.3 Проектирование

В ходе проектирования было решено выделить 4 функций, 3 из которых отвечают за логику, а оставшаяся– за взаимодействие с пользователем.

1. **Логика**

- `int comparing_transport_areey_main(int **matrix1, int **matrix2, int a)`

  Эта функция осуществляет проверку транспортируется ли матрица по главной диагонали. Если да то она возвращает 1, если нет то она возвращает 0.

- `int comparing_transport_areey_secondary_diagonal(int**matrix1, int**mat`

  Эта функция осуществляет проверку транспортируется ли матрица по побочной диагонали. Если да то она возвращает 1, если нет то она возвращает 0.

- `int are_matrixes_transposable(int** matrix1, int** matrix2, int a)`

  Эта функция осуществляет проверку полученых данных от функций `int comparing_transport_areey_main(int **matrix1, int **matrix2` и `int comparing_transport_areey_secondary_diagonal(int**matrix1, int**ma` спомощью оператор ветвления **if**, который брабатывает полученые значения. Если да то она возвращает 1, если нет то она возвращает 0.

2. Взаимодействие с пользователем

- `void matrix()`

  Эта функция отвечает за взаимодействие с пользователем при запуске приложения в интерактивном режиме. Она содержит один аргумент – размер масива. Также выделяет динамическую память и и заполняет её числами с файла.

3.1.4 Описание тестового стенда и методики тестирования

Интегрированная среда разработки: Qt Creator 3.5.0 (opensource)
Компилятор: GCC 4.9.1 20140922 (Red Hat 4.9.1-10)
Операционная система: Debian GNU/Linux 8 (jessie) 32-бита (version 3.14.1)
Утилита cppcheck: 1.67
Утилита valgrind: valgrind-3.10.0

На всех стадиях разработки приложения проходило автоматическое тестирование с помощью модульных тестов *Qt*, основанных на библиотеке *QTestLib*.

Аналогично, на всех стадиях разработки приложения проводился динамический анализ утилитой *valgrind*. На финальной стадии был проведён статический анализ с помощью утилиты *cppcheck*.

Интегрированная среда разработки: Qt Creator 3.5.0 (opensource)
Компилятор: GCC 4.9.1 20140922 (Red Hat 4.9.1-10)
Операционная система: Debian Windows 8.1 64-бита

На всех стадиях разработки приложения проходило автоматическо етестирование. Осуществлялось посредством модульных тестов *Qt*, основанных на библиотеке *QTestLib*.

3.1.5 Тестовый план и результаты тестирования

1. **Модульные тесты *Qt***

   I тест

   Входные данные:
   Первая матрица 1 2 3 4 Вторая матрица 1 3 2 4
   Выходные данные: 1
   Результат: Тест успешно пройден

3.1.6 Выводы

В ходе выполнения работы автор получил опыт в обработке матрицы и в работе с файлами.

Листинги

issituated.c)

```
#include "main.h"
#include "issituated_logic.h"

void issituated()
{

    struct poligon plot;
    struct poligon house1;
    struct poligon house2;
    printf("Print land x, y:");
    scanf("%d%d", &plot.length, &plot.width);
```

```
12      printf("Print the coordinates of house 1:");
13      scanf("%d%d", &house1.length, &house1.width);
14      printf("Print the coordinates of house 2:");
15      scanf("%d%d", &house2.length, &house2.width);
16
17      if (calculation(plot, house1, house2) == 1)
18      {
19          printf("There is enough space for who houses\n");
20      }
21      else
22      {
23          printf("There is not enough space for who houses\
                n");
24      }
25  }
```

issituated_logic.c

```
1   #include "issituated_logic.h"
2
3   int calculation(struct poligon plot, struct poligon
        house1, struct poligon house2)
4   {
5
6       if (((plot.length >= (house1.length + house2.length))
7            && (plot.width >= house2.width)
8            && (plot.width >= house1.width))
9               || ((plot.width >= (house2.width + house1.
                    width))
10                   && (plot.length >= house1.length)
11                   && (plot.length >= house2.length)))
12      {
13          return 1;
14      }
15      else
16      {
17          return 0;
18      }
19  }
```

Глава 4

Строки

4.1 Задание 1. Поиск слов по ключевому слову

4.1.1 Задание

Задан набор ключевых слов, а также текст, в котором хранится длинный список названий книг. Выбрать названия, содержащие хотя бы одно из заданных ключевых слов.

4.1.2 Теоритические сведения

При разработке приложения были задействованы следующие конструкции языка: оператор выбора **switch**, оператор ветвления **if**, оператор цикла со счётчиком **for**, оператор цикла с предусловием **while** – и были использованы функции стандартной библиотеки *fopen*, *fclose*, *fgets*, *fputs* и *puts*, определённые в заголовочном файле *stdio.h*; *atoi*, *calloc*, *free*, определённые в *stdlib.h*; *strlen*, *memset* и *strcat*, определённые в *string.h*.

Так как формат ввода текста с файла не дан автор решил что каждое название кники будет начинаться с новой строки, после проверки строки на наличие ключевого слова, будет выводится строка в которой находится ключевое слово.

4.1.3 Проектирование

В ходе проектирования было решено выделить 3 функций, 2 из которых отвечают за логику, а оставшаяся – за взаимодействие с пользователем.

1. **Логика**

   - `void poisk(char *write_string, char *keyword, FILE *open_file))`
     Эта функция открывает файл, проходит по тексту, лежащему в нем, и сравнивает ключевое слово со строкой. Если в строке есть ключевое слово, то запускается функция `print_book(write_string, first_o`

   - `int print_book(char *write_string, char *first_occurrence_of_write_strin`
     Эта функция выводит строку в которой лежит ключевое слово. И возвращает переменную `first_occurrence_of_write_string`

2. **Взаимодействие с пользователем**

   - `void string_book()`
     Эта функция выделяет память стороке и ключевому слову, а также заполняет их.

4.1.4 Выводы

В ходе работы я получил опыт в обработке строк, а также укрепил навык работы с файлами.

Листинги

`string_book.c`

```c
#include "main.h"
#include "string_book.h"

void string_book()
{
    char *write_string;
    char *keyword;

    write_string = (char*)malloc(1000*sizeof(char));
    keyword = (char*)malloc(100*sizeof(char));

    FILE *open_file = fopen("string_book.txt" , "r");
        //Название книги вводить в столбик
    if (open_file == NULL)
    {
        printf("Error open file\n");
    }
```

```
17      else
18      {
19          printf("Print keyword\n");
20
21          gets(keyword);
22          gets(keyword);
23
24          poisk(write_string, keyword, open_file);
25          fclose(open_file);
26      }
27
28      free(write_string);
29      free(keyword);
30  }
```

string_book_logic.c

```
1   #include "string_book.h"
2
3   void poisk(char *write_string, char *keyword, FILE *
        open_file)
4   {
5       char *first_occurrence_of_write_string;
6       first_occurrence_of_write_string = 0;
7       while (fgets(write_string, 100, open_file) &&
            first_occurrence_of_write_string == 0)
8       {
9           first_occurrence_of_write_string = strstr (
                write_string, keyword);
10
11          if (first_occurrence_of_write_string != 0)
12          {
13              first_occurrence_of_write_string = print_book
                    (write_string,
                    first_occurrence_of_write_string);
14          }
15      }
16  }
17
18  int print_book(char *write_string, char *
        first_occurrence_of_write_string)
19  {
20      puts(write_string);
21      first_occurrence_of_write_string = 0;
22      return (first_occurrence_of_write_string);
23  }
```

Глава 5

Введение в классы C++

5.1 Задание 1. Инкапсуляция. Линейный список

5.1.1 Задание

Реализовать класс ЛИНЕЙНЫЙ СПИСОК (целых чисел). Требуемые методы: конструктор, деструктор, поиск элемента, удаление элемента

5.1.2 Теоритические сведения

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

5.1.3 Проектирование

В ходе проектирования было решено выделить 2 класса, 1 из которых отвечают за логику, а другой – за поиск ошибок.

1. **Логика**. `class List`
   (a) `Поля`
      i. `int* list`
      ii. `int size`
      iii. `int i`
      iv. `const int sizeIncrement = 6`
   (b) `Методы`

i. `void allocateMoreMemory()` Этот метод выделяет дополнительную память.

ii. `void allocateMoreMemory()` Конструктор. В этом методе динамически выделяется память размером `size = 2`.

iii. `List(int size = 2)()` Деструктор. Освобождает выделенную память. Уничтожает объект.

iv. `~List()` Этот метод выводит значения в консоль.

v. `void put(int number)` Этот метод удоляет элемент.

vi. `void erase(int position)` Это метод осуществляет поиск элемента слева на право.

vii. `int find(int number) const` Это метод осуществляет поиск элемента права на лево.

viii. `class NoItemException` Этот класс ишат исключение.

5.1.4 Описание тестового стенда и методики тестирования

Интегрированная среда разработки: Qt Creator 3.5.0 (opensource)
Компилятор: GCC 4.9.1 20140922 (Red Hat 4.9.1-10)
Операционная система: Debian Windows 8.1 64-бита

На всех стадиях разработки приложения проходило основанных на библиотеке *QTestLib*.

Аналогично, на всех стадиях разработки приложения проводился динамический анализ утилитой *valgrind*.

На финальной стадии был проведён статический анализ с помощью утилиты *cppcheck*.

5.1.5 Тестовый план и результаты тестирования

1. **Модульные тесты *Qt***

   Модульными тестами была протестиована работоспособность методов. Все требуемые методы - добавление элемента, копирование объекта, индексирование по ключу, конструктор и деструктор - работают.

2. **Статический анализ *cppcheck***

   Утилита *cppcheck* не выдала предупреждений.

3. **Динамический анализ *valgrind***

   Утилита *valgrind* не выявила проблем.

5.1.6 Выводы

Автор получил опыт работы в языке C++, познакомился с инкапсуляцией, а также научился обрабатывать исключительные ситуации.

Листинги

list.cpp

```
1  #include "list.h"
2
3  List::List(int size) : size(size)
4  {
5      list = new int[size];
6      i = -1;
7  }
8
9  List::~List()
10 {
11     delete[] list;
12 }
13
14 void List::allocateMoreMemory()
15 {
16     int* t = new int[size += sizeIncrement];
17     for (int i = 0; i <= this->i; ++i)
18     {
19         t[i] = list[i];
20     }
21     delete[] list;
22     list = t;
23 }
24
25 void List::put(int number)
26 {
27     if (i == size - 1)
28     {
29         allocateMoreMemory();
30     }
31     list[++i] = number;
32 }
33
34 void List::erase(int position)
35 {
```

```
36      if (position > i)
37      {
38          throw BeyondTheLimitException(position);
39      }
40      for (int i = position; i <= --this->i; ++i)
41      {
42          list[i] = list[i + 1];
43      }
44      int* t = new int[size];
45      for (int i = 0; i <= this->i; ++i)
46      {
47          t[i] = list[i];
48      }
49      delete[] list;
50      list = t;
51  }
52
53  int List::find(int number) const
54  {
55      for (int i = 0; i <= this->i; ++i)
56      {
57          if (list[i] == number)
58              return i;
59      }
60      throw NoItemException(number);
61  }
62
63  int List::rfind(int number) const
64  {
65      for (int i = this->i; i >= 0; --i)
66      {
67          if (list[i] == number)
68              return i;
69      }
70      throw NoItemException(number);
71  }
```

list.h

```
1   #ifndef LIST_H
2   #define LIST_H
3
4   #include <exception>
5
6   class NoItemException
7   {
8       int number;
9   public:
10      NoItemException(int number) : number(number){}
```

```
11      int getError()
12      {
13          return number;
14      }
15  };
16
17  class BeyondTheLimitException
18  {
19      int i;
20  public:
21      BeyondTheLimitException(int i) : i(i){}
22      int getError()
23      {
24          return i;
25      }
26  };
27
28  class List
29  {
30      int* list;
31      int size;
32      int i;
33      const int sizeIncrement = 6;
34      void allocateMoreMemory();
35
36  public:
37      List(int size = 2);
38      ~List();
39      void put(int number);
40      void erase(int position);
41      int find(int number) const;
42      int rfind(int number) const;
43  };
44
45  #endif // LIST_H
```

Глава 6

Приложения

Листинги вклад в банке

matrix_ui.c

```
#include "main.h"
#include "matrix_logic.h"

void matrix()
{
    int a;
    puts("Enter matrix dimension");
    scanf("%d", &a);
    int **matrix1, **matrix2;
    int i, j;
    matrix1 = (int**)malloc(a*sizeof(int*));
    for(i = 0; i < a; i++)
    {
        matrix1[i] = (int*)malloc(a*sizeof(int));
    }

    matrix2 = (int**)malloc(a*sizeof(int*));
    for(i = 0; i < a; i++)
    {
        matrix2[i] = (int*)malloc(a*sizeof(int));
    }

    FILE *file_matrix1 = fopen ("matrix1.txt", "r");
    for (i = 0; i < a; i++)
    {
        for (j = 0; j < a; j++)
        {
            fscanf(file_matrix1, "%d ", &matrix1[i][j]);
        }
        fscanf(file_matrix1, "\n");
```

```
    }
    fclose(file_matrix1);

    FILE *file_matrix2 = fopen ("matrix2.txt", "r");
    for (i = 0; i < a; i++)
    {
        for (j = 0; j < a; j++)
        {
            fscanf(file_matrix2, "%d ", &matrix2[i][j]);
        }
        fscanf(file_matrix2, "\n");
    }
    fclose(file_matrix2);

    if (are_matrixes_transposable(matrix1, matrix2, a))
    {
        puts("YES!!!");
    }
    else
    {
        puts("NO!!!");
    }
    for(i = 0; i < a; i++)
    {
        free(matrix1[i]);
        free(matrix2[i]);
    }
    free(matrix1);
    free(matrix2);
}
```

matrix_logic.c

```
int comparing_transport_areey_main(int **matrix1, int **
    matrix2, int a)
{
    int i , j, result = 1;
    int b = 0;
    for (i = 0; i < a; i++)
    {
        b++;
        for(j = 0; j < b; j++)
        {
            if (matrix1[i][j] != matrix2[j][i])
            {
                result = 0;
            }
        }
    }
```

```
16      return result;
17  }
18
19  int comparing_transport_areey_secondary_diagonal(int**
        matrix1, int**matrix2, int  a)
20  {
21      int i, j, result = 1;
22      int b = a;
23      for (i = 0; i < a - 1; i++)
24      {
25          b--;
26          for(j = 0; j < b; j++)
27          {
28              if (matrix1[i][j] != matrix2[a-j-1][a-i-1])
29              {
30                  result = 0;
31              }
32          }
33      }
34      return result;
35  }
36  int are_matrixes_transposable(int** matrix1, int**
        matrix2, int a)
37  {
38      if (comparing_transport_areey_main(matrix1, matrix2,
            a) ||
39              comparing_transport_areey_secondary_diagonal(
                    matrix1, matrix2, a))
40      {
41          return 1;
42      }
43      else
44      {
45          return 0;
46      }
47  }
```

matrix_logic.h

```
1  #ifndef MATRIX
2  #define MATRIX
3
4  #include <stdio.h>
5  #include <stdlib.h>
6  #include <string.h>
7  #include <malloc.h>
8
9  #ifdef __cplusplus
10
```

```
11 extern "C" {
12
13 #endif
14
15 int are_matrixes_transposable(int** matrix1, int**
       matrix2, int a);
16 int comparing_transport_areey_main(int**, int**, int);
17 int comparing_transport_areey_secondary_diagonal(int**,
       int**, int);
18
19 #ifdef __cplusplus
20 }
21 #endif
22
23 #endif // MATRIX
```

Листинги рамзмещение двух домов на участке

matrix_ui.c

```
1  #include "main.h"
2  #include "matrix_logic.h"
3
4  void matrix()
5  {
6      int a;
7      puts("Enter matrix dimension");
8      scanf("%d", &a);
9      int **matrix1, **matrix2;
10     int i, j;
11     matrix1 = (int**)malloc(a*sizeof(int*));
12     for(i = 0; i < a; i++)
13     {
14         matrix1[i] = (int*)malloc(a*sizeof(int));
15     }
16
17     matrix2 = (int**)malloc(a*sizeof(int*));
18     for(i = 0; i < a; i++)
19     {
20         matrix2[i] = (int*)malloc(a*sizeof(int));
21     }
22
23     FILE *file_matrix1 = fopen ("matrix1.txt", "r");
24     for (i = 0; i < a; i++)
25     {
26         for (j = 0; j < a; j++)
27         {
28             fscanf(file_matrix1, "%d ", &matrix1[i][j]);
```

```
        }
        fscanf(file_matrix1, "\n");
    }
    fclose(file_matrix1);

    FILE *file_matrix2 = fopen ("matrix2.txt", "r");
    for (i = 0; i < a; i++)
    {
        for (j = 0; j < a; j++)
        {
            fscanf(file_matrix2, "%d ", &matrix2[i][j]);
        }
        fscanf(file_matrix2, "\n");
    }
    fclose(file_matrix2);

    if (are_matrixes_transposable(matrix1, matrix2, a))
    {
        puts("YES!!!");
    }
    else
    {
        puts("NO!!!");
    }
    for(i = 0; i < a; i++)
    {
        free(matrix1[i]);
        free(matrix2[i]);
    }
    free(matrix1);
    free(matrix2);
}
```

matrix_logic.c

```
int comparing_transport_areey_main(int **matrix1, int **
    matrix2, int a)
{
    int i , j, result = 1;
    int b = 0;
    for (i = 0; i < a; i++)
    {
        b++;
        for(j = 0; j < b; j++)
        {
            if (matrix1[i][j] != matrix2[j][i])
            {
                result = 0;
            }
```

```
        }
    }
    return result;
}

int comparing_transport_areey_secondary_diagonal(int**
    matrix1, int**matrix2, int  a)
{
    int i, j, result = 1;
    int b = a;
    for (i = 0; i < a - 1; i++)
    {
        b--;
        for(j = 0; j < b; j++)
        {
            if (matrix1[i][j] != matrix2[a-j-1][a-i-1])
            {
                result = 0;
            }
        }
    }
    return result;
}
int are_matrixes_transposable(int** matrix1, int**
    matrix2, int a)
{
    if (comparing_transport_areey_main(matrix1, matrix2,
        a) ||
            comparing_transport_areey_secondary_diagonal(
                matrix1, matrix2, a))
    {
        return 1;
    }
    else
    {
        return 0;
    }
}
```

matrix_logic.h

```
#ifndef MATRIX
#define MATRIX

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <malloc.h>

```

```
9  #ifdef __cplusplus
10
11 extern "C" {
12
13 #endif
14
15 int are_matrixes_transposable(int** matrix1, int**
       matrix2, int a);
16 int comparing_transport_areey_main(int**, int**, int);
17 int comparing_transport_areey_secondary_diagonal(int**,
       int**, int);
18
19 #ifdef __cplusplus
20 }
21 #endif
22
23 #endif // MATRIX
```

Листинги циклов

matrix_ui.c

```
1  #include "main.h"
2  #include "matrix_logic.h"
3
4  void matrix()
5  {
6      int a;
7      puts("Enter matrix dimension");
8      scanf("%d", &a);
9      int **matrix1, **matrix2;
10     int i, j;
11     matrix1 = (int**)malloc(a*sizeof(int*));
12     for(i = 0; i < a; i++)
13     {
14         matrix1[i] = (int*)malloc(a*sizeof(int));
15     }
16
17     matrix2 = (int**)malloc(a*sizeof(int*));
18     for(i = 0; i < a; i++)
19     {
20         matrix2[i] = (int*)malloc(a*sizeof(int));
21     }
22
23     FILE *file_matrix1 = fopen ("matrix1.txt", "r");
24     for (i = 0; i < a; i++)
25     {
26         for (j = 0; j < a; j++)
```

```
27          {
28              fscanf(file_matrix1, "%d ", &matrix1[i][j]);
29          }
30          fscanf(file_matrix1, "\n");
31      }
32      fclose(file_matrix1);
33
34      FILE *file_matrix2 = fopen ("matrix2.txt", "r");
35      for (i = 0; i < a; i++)
36      {
37          for (j = 0; j < a; j++)
38          {
39              fscanf(file_matrix2, "%d ", &matrix2[i][j]);
40          }
41          fscanf(file_matrix2, "\n");
42      }
43      fclose(file_matrix2);
44
45      if (are_matrixes_transposable(matrix1, matrix2, a))
46      {
47          puts("YES!!!");
48      }
49      else
50      {
51          puts("NO!!!");
52      }
53      for(i = 0; i < a; i++)
54      {
55          free(matrix1[i]);
56          free(matrix2[i]);
57      }
58      free(matrix1);
59      free(matrix2);
60  }
```

matrix_logic.c

```
1  int comparing_transport_areey_main(int **matrix1, int **
       matrix2, int a)
2  {
3      int i , j, result = 1;
4      int b = 0;
5      for (i = 0; i < a; i++)
6      {
7          b++;
8          for(j = 0; j < b; j++)
9          {
10             if (matrix1[i][j] != matrix2[j][i])
11             {
```

```
                result = 0;
            }
        }
    }
    return result;
}

int comparing_transport_areey_secondary_diagonal(int**
    matrix1, int**matrix2, int  a)
{
    int i, j, result = 1;
    int b = a;
    for (i = 0; i < a - 1; i++)
    {
        b--;
        for(j = 0; j < b; j++)
        {
            if (matrix1[i][j] != matrix2[a-j-1][a-i-1])
            {
                result = 0;
            }
        }
    }
    return result;
}
int are_matrixes_transposable(int** matrix1, int**
    matrix2, int a)
{
    if (comparing_transport_areey_main(matrix1, matrix2,
        a) ||
            comparing_transport_areey_secondary_diagonal(
                matrix1, matrix2, a))
    {
        return 1;
    }
    else
    {
        return 0;
    }
}
```

matrix_logic.h

```
#ifndef MATRIX
#define MATRIX

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

```
7  #include <malloc.h>
8
9  #ifdef __cplusplus
10
11 extern "C" {
12
13 #endif
14
15 int are_matrixes_transposable(int** matrix1, int**
       matrix2, int a);
16 int comparing_transport_areey_main(int**, int**, int);
17 int comparing_transport_areey_secondary_diagonal(int**,
       int**, int);
18
19 #ifdef __cplusplus
20 }
21 #endif
22
23 #endif // MATRIX
```

Листинги матриц

issituated.c)

```
1  #include "main.h"
2  #include "issituated_logic.h"
3
4  void issituated()
5  {
6
7      struct poligon plot;
8      struct poligon house1;
9      struct poligon house2;
10     printf("Print land x, y:");
11     scanf("%d%d", &plot.length, &plot.width);
12     printf("Print the coordinates of house 1:");
13     scanf("%d%d", &house1.length, &house1.width);
14     printf("Print the coordinates of house 2:");
15     scanf("%d%d", &house2.length, &house2.width);
16
17     if (calculation(plot, house1, house2) == 1)
18     {
19         printf("There is enough space for who houses\n");
20     }
21     else
22     {
23         printf("There is not enough space for who houses\
               n");
```

```
    }
}
```

issituated_logic.c

```c
#include "issituated_logic.h"

int calculation(struct poligon plot, struct poligon
    house1, struct poligon house2)
{

    if (((plot.length >= (house1.length + house2.length))
         && (plot.width >= house2.width)
         && (plot.width >= house1.width))
            || ((plot.width >= (house2.width + house1.
                width))
                && (plot.length >= house1.length)
                && (plot.length >= house2.length)))
    {
        return 1;
    }
    else
    {
        return 0;
    }
}
```

issituated_logic.h

```c
#ifndef ISSITUATED_H
#define ISSITUATED_H

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <malloc.h>

#ifdef __cplusplus

extern "C" {

#endif


struct poligon{
    int length;
    int width;
    int plot;
```

```
20      int house1;
21      int house2;
22  };
23
24  void menu_issituated();
25  void issituated();
26  int calculation(struct poligon, struct poligon, struct
        poligon);
27
28  #ifdef __cplusplus
29  }
30  #endif
31
32  #endif // ISSITUATED_H
```

Листинги строк

string_book.c

```
1   #include "main.h"
2   #include "string_book.h"
3
4   void string_book()
5   {
6       char *write_string;
7       char *keyword;
8
9       write_string = (char*)malloc(1000*sizeof(char));
10      keyword = (char*)malloc(100*sizeof(char));
11
12      FILE *open_file = fopen("string_book.txt" , "r");
            //Название книги вводить в столбик
13      if (open_file == NULL)
14      {
15          printf("Error open file\n");
16      }
17      else
18      {
19          printf("Print keyword\n");
20
21          gets(keyword);
22          gets(keyword);
23
24          poisk(write_string, keyword, open_file);
25          fclose(open_file);
26      }
27
28      free(write_string);
```

```
29      free(keyword);
30  }
```

string_book_logic.c

```
1  #include "string_book.h"
2
3  void poisk(char *write_string, char *keyword, FILE *
       open_file)
4  {
5      char *first_occurrence_of_write_string;
6      first_occurrence_of_write_string = 0;
7      while (fgets(write_string, 100, open_file) &&
           first_occurrence_of_write_string == 0)
8      {
9          first_occurrence_of_write_string = strstr (
               write_string, keyword);
10
11         if (first_occurrence_of_write_string != 0)
12         {
13             first_occurrence_of_write_string = print_book
                   (write_string,
                   first_occurrence_of_write_string);
14         }
15     }
16 }
17
18 int print_book(char *write_string, char *
       first_occurrence_of_write_string)
19 {
20     puts(write_string);
21     first_occurrence_of_write_string = 0;
22     return (first_occurrence_of_write_string);
23 }
```

string_book.h

```
1  #ifndef STRING
2  #define STRING
3
4  #include <stdio.h>
5  #include <stdlib.h>
6  #include <string.h>
7  #include <malloc.h>
8
9  #ifdef __cplusplus
10
11 extern "C" {
```

```
12
13 #endif
14
15 void menu_string_book();
16 void string_book();
17 void poisk(char*, char*, FILE*);
18 void file(char*, char*);
19 int print_book(char*, char*);
20
21 #ifdef __cplusplus
22 }
23 #endif
24
25
26 #endif // STRING
```

Листинги

list.cpp

```
1  #include "list.h"
2
3  List::List(int size) : size(size)
4  {
5      list = new int[size];
6      i = -1;
7  }
8
9  List::~List()
10 {
11     delete[] list;
12 }
13
14 void List::allocateMoreMemory()
15 {
16     int* t = new int[size += sizeIncrement];
17     for (int i = 0; i <= this->i; ++i)
18     {
19         t[i] = list[i];
20     }
21     delete[] list;
22     list = t;
23 }
24
25 void List::put(int number)
26 {
27     if (i == size - 1)
28     {
```

```
29          allocateMoreMemory();
30      }
31      list[++i] = number;
32  }
33
34  void List::erase(int position)
35  {
36      if (position > i)
37      {
38          throw BeyondTheLimitException(position);
39      }
40      for (int i = position; i <= --this->i; ++i)
41      {
42          list[i] = list[i + 1];
43      }
44      int* t = new int[size];
45      for (int i = 0; i <= this->i; ++i)
46      {
47          t[i] = list[i];
48      }
49      delete[] list;
50      list = t;
51  }
52
53  int List::find(int number) const
54  {
55      for (int i = 0; i <= this->i; ++i)
56      {
57          if (list[i] == number)
58              return i;
59      }
60      throw NoItemException(number);
61  }
62
63  int List::rfind(int number) const
64  {
65      for (int i = this->i; i >= 0; --i)
66      {
67          if (list[i] == number)
68              return i;
69      }
70      throw NoItemException(number);
71  }
```

main.cpp

```
1  #include <iostream>
2  #include "list.h"
3
```

```
4  using namespace std;
5
6  int main()
7  {
8      List list;
9      list.put(5);
10     list.put(6);
11     list.put(7);
12     list.put(5);
13     cout << list.find(5) << endl;
14     cout << list.rfind(5) << endl;
15     list.erase(3);
16     cout << list.rfind(5) << endl;
17     try
18     {
19         //list.erase(5);
20         list.find(10);
21     }
22     catch (NoItemException& e)
23     {
24         cout << "There is no item like \"" << e.getError
               () << "\"" << endl;
25     }
26     catch (BeyondTheLimitException& e)
27     {
28         cout << "There is not item with position " << e.
               getError() << endl;
29     }
30
31     return 0;
32 }
```

list.h

```
1  #ifndef LIST_H
2  #define LIST_H
3
4  #include <exception>
5
6  class NoItemException
7  {
8      int number;
9  public:
10     NoItemException(int number) : number(number){}
11     int getError()
12     {
13         return number;
14     }
15 };
```

```
16
17 class BeyondTheLimitException
18 {
19     int i;
20 public:
21     BeyondTheLimitException(int i) : i(i){}
22     int getError()
23     {
24         return i;
25     }
26 };
27
28 class List
29 {
30     int* list;
31     int size;
32     int i;
33     const int sizeIncrement = 6;
34     void allocateMoreMemory();
35
36 public:
37     List(int size = 2);
38     ~List();
39     void put(int number);
40     void erase(int position);
41     int find(int number) const;
42     int rfind(int number) const;
43 };
44
45 #endif // LIST_H
```