Программирование

Корсков А. В.

25 декабря 2015 г.

Глава 1

Основные конструкции языка

1.1 Задание 1

1.1.1 Задание

В морской миле 2000 ярдов или 6000 футов. Задано некоторое расстояние в футах, например, 9139 футов. Вывести то же расстояние в милях, ярдах и футах, например, 9139 = 1 миля 1046 ярдов и 1 фут.

1.1.2 Теоретические сведения

Для реализации данного алгоритмы были использованы функции стандартной библеотеки для ввода и вывода информации. Также для передачи значений из функции мы использовали указатели. В ходе доработки было принято решение использовать структуры.

Морская миля — единица измерения расстояния, применяемая в мореплавании и авиации. Ярд (англ. yard) — британская и американская единица измерения расстояния. Фут — единица измерения длины в английской системе мер. Мы знаем, что в 1 миле содержится 2000 ярдов или 6000 футов. А в 1 ярде содержится 3 фута.

1.1.3 Проектирование

В функции main вызывается функция convert_UI. Взаимодействие с пользователем реализовано в фунции convert_UI. Она считывает ввёдные пользователем значения из консоли, вызывает функцию convert, а после выполнения функции convert, выводит результат в консоль. Для реализации алгоритма мы выделили функцию convert. Функция принимает один указатель на структуру. Сначала в функции вычисляет-

ся колличество миль. Далее находится колличество футов без миль. На следующем шаге вычисляется колличество ярдов. И на последнем шаге находится колличество футов. Функция ничего не возвращает, так как для передачи значения мы пользуемся указателями.

1.1.4 Описание тестового стенда и методики тестирования

Для решения задачи использовалась вертуальная машина с операционной системой Debian 8.1, в которой установлен QtCreator 3.5.0. В конце использовалась утилита cppcheck 1.67. Пользователь может сам ввести его данные или воспользоваться автоматическими тестами, запустив их отдельно.

1.1.5 Тестовый план и результаты тестирования

В программе предусмотрены тесты, которые проверяют правильность исполнения программы. В тесте вызывается функция, ей передаётся колличество футов, равное 9139. Ожидается, что результатом работы программы будет колличество футов равное 1, колличество ярдов равное 1046 и колличество миль равное 1. Далее каждая из трёх переменных проверяется на соответствие с ожидаемым правильным результатом. При вызове теста ошибок не обнаружено. В конце программа была проверена утилитой cppcheck, которая не выдала замечаний.

1.1.6 Выводы

В ходе работы мы научились структурировать проект, разбивая задачи на мелкие подзадачи. Получили опыт создания тестов. Также мы научились работать с системой контроля версий и приобрели навыки работы с указателями. Также получили опыт работы со структурами.

Листинги

```
1 #include <stdio.h>
2 #include "convert_UI.h"
3 #include "check_UI.h"
4 #include "removal_UI.h"
5 #include "square_UI.h"
6 #include "removing_words_UI.h"
7
8 int main(void)
```

```
{
    puts("1. Перевод футов в мили, ярды и футы.");
    puts("2. Проверка допустимости треугольника.");
    puts("3. Удаление из числа нечётных чисел.");
    puts("4. Проверить является ли двумерный массив латинским
        квадратом.");
    puts("5. Удаление повторяющихся слов.");
    int choice;
    scanf("%d", &choice);
    switch(choice){
        case 1:
            convert_UI();
        break;
        case 2:
            check_UI();
        break;
        case 3:
            removal_UI();
        break;
        case 4:
            square_UI();
        case 5:
            removing_words_UI();
    }
    return 0;
}
```

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

void convert_UI(){
    puts("Введите количество футов.");
    struct measure_units st1;
    scanf("%d", &st1.ft);
    convert(&st1);
    printf("Колличество миль: %d, колличество ярдов: %d колли
       чество футов: %d \n", st1.m, st1.yd, st1.ft);
}
```

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

void convert(struct measure_units *arg){
    arg->m = arg->ft / 6000;
    arg->ft = arg->ft - arg->m * 6000;
    arg->yd = arg->ft / 3;
    arg->ft = arg->ft - arg->yd * 3;
}
```

1.2 Задание 2

1.2.1 Задание

Заданы три целых числа: a, b, c. Определить, могут ли они быть длинами сторон треугольника, и если да, определить, является ли он равнобедренным либо равносторонним.

1.2.2 Теоритические сведения

Треугольник существует только тогда, когда сумма любых двух его сторон больше третьей. У равнобедренного треугольника две стороны равны. У равностороннего все стороны равны. Для реализации данного алгоритмы были использованы функции стандартной библеотеки для ввода и вывода информации. Также использовалась конструкция if...else.

1.2.3 Проектирование

В функции main вызывается функция check_UI. Взаимодействие с пользователем реализовано в фунции check_UI. Она считывает ввёдные пользователем значения из консоли, вызывает функцию check, а после выполнения функции check, выводит результат в консоль. Для реализации алгоритма мы выделили функцию check. Функция принимает три переменные типа int. Каждая из этих трёх переменных задаёт одну из сторон треугольника. Сначала входные данные проверяются на корректность. Далее проверяем возможен ли данный треугольник или нет. Затем смотрим является ли данный треугольник равнобедренним. Если это условие выполняется, проверяем будет ли данный треугольник равносторонним.

1.2.4 Описание тестового стенда и методики тестирования

Для решения задачи использовалась вертуальная машина с операционной системой Debian 8.1, в которой установлен QtCreator 3.5.0. В конце использовалась утилита cppcheck 1.67. Пользователь может сам ввести его данные или воспользоваться автоматическими тестами, запустив их отдельно.

1.2.5 Тестовый план и результаты тестирования

В программе предусмотрены тесты, которые проверяют правильность исполнения программы. В тесте вызывается функция, ей передаётся три стороны треугольника, которые все равны 3. Ожидается, что результатом работы программы будет цифра 3, означающая, что треугольник равносторонний. Далее полученное число сверяется с ожидаемым правильным ответом. При вызове теста ошибок не обнаружено. В конце программа была проверена утилитой cppcheck, которая не выдала замечаний.

1.2.6 Выводы

В ходе работы мы научились работать с конструкцией if...else. Получили опыт создания вложенных операторов if...else.

```c
#include <stdio.h>
#include "check.h"

void check_UI(){
    puts("Введите три стороны треугольника.");
    int a,b,c,result_check;
    scanf("%d%d%d", &a, &b, &c);
    result_check = check(a,b,c);
    switch(result_check){
        case incorrectly:
            printf("Данные некорректны.\n");
        break;
        case possible:
            printf("Данный треугольник возможен.\n");
        break;
        case isosceles:
            printf("Данный треугольник является равнобедренны
                м.\n");
        break;
        case equilateral:
            printf("Данный треугольник является равносторонни
                м.\n");
        break;
        case impossible:
            printf("Данный треугольник не возможен.\n");
    }
}
```

```c
#include <stdio.h>
#include "check.h"

```

```
4  int check(int a, int b, int c){
5      if (a <= 0 || b <= 0 || c <= 0)
6          return incorrectly;
7      else{
8          if (a + b > c && b + c > a && a + c > b){
9              if (a == b && b == c){
10                 return equilateral;
11             }
12             if ((a == b) || (a == c) || (b == c)){
13                 return isosceles;
14             }
15             return possible;
16         }
17         else{
18             return impossible;
19         }
20     }
21 }
```

Глава 2

Циклы

2.1 Задание 1

2.1.1 Задание

Найти число, полученное из данного выбрасыванием нечётных цифр.

2.1.2 Теоритические сведения

Для решения данной задачи ипользовалась конструкция if..else. А также цикл с предусловием while и функции стандартной библеотеки для ввода и вывода информации.

2.1.3 Проектирование

В функции main вызывается функция removal_UI. Взаимодействие с пользователем реализовано в фунции removal_UI. Она считывает ввёдные пользователем значения из консоли, вызывает функцию removal, а после выполнения функции removal, выводит результат в консоль. Для реализации алгоритма мы выделили функцию removal. Функция принимает одно значение типа int. Это значение задаёт число, в котором нужно удалить цифры. Далее пока число не будет равняться 0 оно будет делиться на 10. От числа отрезается последняя цифра и если она чётная записать её в результат.

2.1.4 Описание тестового стенда и методики тестирования

Для решения задачи использовалась вертуальная машина с операционной системой Debian 8.1, в которой установлен QtCreator 3.5.0. В конце использовалась утилита cppcheck 1.67. Пользователь может сам ввести его данные или воспользоваться автоматическими тестами, запустив их отдельно.

2.1.5 Тестовый план и результаты тестирования

В программе предусмотрены тесты, которые проверяют правильность исполнения программы. В тесте вызывается функция, ей передаётся число равное 2597. Ожидается, что результатом работы программы будет цифра 2.Далее полученное число сверяется с ожидаемым правильным ответом. При вызове теста ошибок не обнаружено. В конце программа была проверена утилитой cppcheck, которая выдала стилистическое замечание:"The scope of the variable 'numeral' can be reduced.". Это замечание было устранено.

2.1.6 Выводы

В этом задании мы научились работать с циклом while.

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

void removal_UI(){
    puts("Введите число.");
    int number,result_removal;
    scanf("%d", &number);
    result_removal = removal(number);
    printf("Результат: %d \n", result_removal);
}
```

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

int removal(int number){
    int result = 0, digit = 1;
    while(number > 0){
        int numeral = number % 10;
        number = number / 10;
        if (numeral % 2 == 0){
            result = result + digit * numeral;
```

```
11              digit = digit * 10;
12          }
13      }
14      return result;
15  }
```

Глава 3

Матрицы

3.1 Задание 1

3.1.1 Задание

Латинским квадратом порядка n называется квадратная таблица размером n x n, каждая строка и каждый столбец которой содержат все числа от 1 до n. Проверить, является ли заданная целочисленная матрица латинским квадратом.

3.1.2 Теоритические сведения

Двумерный массив является латинским квадратом, если во всех его строках и столбцах есть все цифры от 1 до n.

Для решения данной задачи ипользовалась конструкция if..else, цикл for функции стандартной библеотеки для ввода и вывода информации. Также в задаче понадобилось использовать двумерный массив и выделять для него память.

3.1.3 Проектирование

В функции main вызывается функция square_UI. Взаимодействие с пользователем реализовано в фунции square_UI. Она считывает ввёдные пользователем значения из файла, вызывает функцию square, а после выполнения функции square, выводит результат в файл. Для реализации алгоритма мы выделили функцию square. Функция принимает одно значение типа int и указатель на двумерный массив. Далее мы проходим по всем столбцам и строкам и ищем все цифры от 1 до n, которое мы

передали функции square. Если какой-либо цифры нет мы выходим из функции с помощью переменной flag.

3.1.4 Описание тестового стенда и методики тестирования

Для решения задачи использовалась вертуальная машина с операционной системой Debian 8.1, в которой установлен QtCreator 3.5.0. В конце использовалась утилита cppcheck 1.67. Пользователь может сам ввести его данные или воспользоваться автоматическими тестами, запустив их отдельно.

3.1.5 Тестовый план и результаты тестирования

В программе предусмотрены тесты, которые проверяют правильность исполнения программы. В тесте вызывается функция, ей передаётся двумерный массив размером 2x2. Ожидается, что результатом работы программы будет цифра 1. Далее полученный результат сверяется с ожидаемым правильным ответом. При вызове теста ошибок не обнаружено. В конце программа была проверена утилитой cppcheck, которая не выдала замечаний.

3.1.6 Выводы

В этом задании мы научились создавать двумерный массив, выделять под него память, а также манипулировать с его содержимым.

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include "square.h"
4
5  void square_UI(){
6      int n,result_square;
7      FILE * iFile;
8      iFile = fopen("input", "r");
9      fscanf(iFile, "%d", &n);
10     int** two_dim;
11     two_dim = (int **) malloc(sizeof(int*) * n);
12     int i,j;
13     for (i = 0; i < n; i++){
14         two_dim[i] = (int *) malloc(sizeof(int) * n);
15     }
16     for (i = 0; i < n; i++){
17         for (j = 0; j < n; j++){
```

```
            fscanf(iFile, "%d", &two_dim[i][j]);
        }
    }
    fclose(iFile);
    result_square = square(two_dim,n);
    for (i = 0; i < n; i++){
        free(two_dim[i]);
    }
    free(two_dim);
    FILE * oFile;
    oFile = fopen("output", "w");
    if (result_square == 1){
        fprintf(oFile, "Матрица является латинским квадратом.
           ");
    }
    else{
        fprintf(oFile, "Матрица не является латинским квадрат
           ом.");
    }
    fclose(oFile);
}
```

```
int square(int** two_dim, int n){
    int i,j,flag,l;
    for (i = 0; i < n; i++){
        for (j = 1; j < n+1; j++){
            flag = 0;
            for (l = 0; l < n; l++){
                if (two_dim[i][l] == j){
                    flag = 1;
                }
            }
            if (flag == 0){
                return flag;
            }
        }
    }
    for (i = 0; i < n; i++){
        for (j = 1; j < n+1; j++){
            flag = 0;
            for (l = 0; l < n; l++){
                if (two_dim[l][i] == j){
                    flag = 1;
                }
            }
            if (flag == 0){
                return flag;
            }
        }
```

```
28      }
29      return 1;
30  }
```

Глава 4

Строки

4.1 Задание 1

4.1.1 Задание

Часто встречающаяся ошибка начинающих наборщиков – дважды записанное слово. Обнаружить и исправить такие ошибки в тексте.

4.1.2 Теоритические сведения

Для решения данной задачи ипользовалась конструкция if..else, циклы for, while и функции стандартной библеотеки для ввода и вывода информации. Также в задаче понадобилось использовать функции стандартной библиотеки для работы со строками.

4.1.3 Проектирование

В функции main вызывается функция removing_words_UI. Взаимодействие с пользователем реализовано в фунции removing_words_UI. Она считывает ввёдные пользователем значения из файла, вызывает функцию removing_words. Для реализации алгоритма мы выделили функцию removing_words. Функция принимает одно значение типа массив char. Мы отрезаем каждое слово до пробела и сравниваем со следующим словом и если слова различны записываем данное слово.

4.1.4 Описание тестового стенда и методики тестирования

Для решения задачи использовалась вертуальная машина с операционной системой Debian 8.1, в которой установлен QtCreator 3.5.0. В конце использовалась утилита cppcheck 1.67.

4.1.5 Тестовый план и результаты тестирования

Программе передаётся строка "ui ui test test". Ожидается, что результатом работы программы будет строка "ui test". При вызове теста ошибок не обнаружено. В конце программа была проверена утилитой cppcheck, которая выдала стилистические ошибки:"The scope of the variable 'j' can be reduced.", "The scope of the variable 'l' can be reduced.", "The scope of the variable 'k' can be reduced.", "The scope of the variable 'flag' can be reduced.", "Array index 'i' is used before limits check.". Все ошибки были исправлены.

4.1.6 Выводы

В этом задании мы научились работать со строками.

```
#include <stdio.h>
#include <stdlib.h>
#include "removing_words.h"

void removing_words_UI(){
    FILE * iFile;
    iFile = fopen("input_removing_words", "r");
    char string[100],result[100] = "";
    fgets(string, 100, iFile);
    fclose(iFile);
    removing_words(string,result);
    FILE * oFile;
    oFile = fopen("output_removing_words", "w");
    fputs(result,oFile);
    fclose(oFile);
}
```

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

void removing_words(char string[], char *result){
```

```
    int n,i = 0;
    n = strlen(string);
    char word[100];
    while(i <= n){
        int l=0;
        while ((i < n) && (string[i] != ' ')){
            word[l] = string[i];
            i++;
            l++;
        }
        word[l]='\0';
        int j = i + 1;
        int k = 0;
        int flag = 1;
        for (j=i+1;j <= i + l; j++){
            if (word[k] != string[j]){
                flag = 0;
            }
            k++;
        }
        if (flag == 0){
            strcat(result, word);
        }
        i++;
    }
}
```

Глава 5

Инкапсуляция

5.1 Задание 1

5.1.1 Задание

Реализовать класс ВЕКТОР (многомерный). Требуемые методы: конструктор, деструктор, сложение, вычитание, копирование, скалярное произведение, умножение на число, модуль.

5.1.2 Теоритические сведения

Два вектора имеют координаты x_1, y_1, x_2, y_2. В результате их сложения получится вектор с координатами $x_1 + x_2$ и $y_1 + y_2$. В результате их вычитания получится вектор с координатами $x_1 - x_2$ и $y_1 - y_2$. В результате их скалярного произведения получится число равное $x_1 * x_2 + y_1 * y_2$. В результате умножения вектора на число z получится вектор с координатами $x_1 * z$ и $y_1 * z$. В результате взятия модуля получится число равное $\sqrt{x_1 * x_1 + y_1 * y_1}$.

Для решения данной задачи мы создали специальный класс. Его методы будут задавать действия над векторами.

5.1.3 Проектирование

В функции main вызывается функция works_with_vectors. Взаимодействие с пользователем реализовано в фунции works_with_vectors. С помощью неё пользователь может производить действия над векторами. Для реализации алгоритма мы выделили класс vectors. Его методы будут задавать действия над векторами. А также будут два поля x и y задающие координаты.

5.1.4 Описание тестового стенда и методики тестирования

Для решения задачи использовалась вертуальная машина с операционной системой Debian 8.1, в которой установлен QtCreator 3.5.0. В конце использовалась утилита cppcheck 1.67.

5.1.5 Тестовый план и результаты тестирования

В программе предусмотрены тесты, которые проверяют правильность исполнения программы. В тесте проверяются все методы класса. Для теста сложения задаются два вектора с координатами (3,4) и (5,6). Ожидается, что в результате у первого вектора будут координаты (8,10). Для теста вычитания задаются два вектора с координатами (5,6) и (3,4). Ожидается, что в результате у первого вектора будут координаты (2,2). Для теста скалярного произведения задаются два вектора с координатами (3,4) и (5,6). Ожидается, что результатом выполнения программы будет число равное 39. Для теста умножения вектора на число задаётся вектор с координатами (3,4) и число 10. Ожидается, что в результате у первого вектора будут координаты (30,40). Для теста взятия модуля задаётся вектор с координатами (3,4). Ожидается, что результатом выполнения программы будет число равное 10. Все тесты проходят успешно. В конце программа была проверена утилитой cppcheck, которая не выдала замечаний.

5.1.6 Выводы

В этом задании мы научились создавать классы. Задавать методы и поля класса. А также производить действия над объектами одного класса.

```
1  #include "work_with_vectors.h"
2  #include <iostream>
3  #include "vectors.h"
4
5  void work_with_vectors::demonstration() const
6  {
7      addition();
8      subtraction();
9      scalar_product();
10     multiply();
11     module();
12     just_exception();
13 }
14
```

```
15 void work_with_vectors::addition() const
16 {
17     int x1,y1,x2,y2;
18     cout << "Введите координаты векторов." << endl;
19     cin >> x1 >> y1 >> x2 >> y2;
20     my_vector vec1(x1,y1), vec2(x2,y2);
21     vec1 += vec2;
22     cout << vec1.get_x() << ' ' << vec1.get_y() << endl;
23 }
24
25 void work_with_vectors::subtraction() const
26 {
27     int x1,y1,x2,y2;
28     cout << "Введите координаты векторов." << endl;
29     cin >> x1 >> y1 >> x2 >> y2;
30     my_vector vec1(x1,y1), vec2(x2,y2);
31     vec1 -= vec2;
32     cout << vec1.get_x() << ' ' << vec1.get_y() << endl;
33 }
34
35 void work_with_vectors::scalar_product() const
36 {
37     int x1,y1,x2,y2,result;
38     cout << "Введите координаты векторов." << endl;
39     cin >> x1 >> y1 >> x2 >> y2;
40     my_vector vec1(x1,y1), vec2(x2,y2);
41     result = vec1.scalar_product(vec2);
42     cout << result << endl;
43 }
44
45 void work_with_vectors::multiply() const
46 {
47     int x1,y1,number;
48     cout << "Введите координаты вектора и число." << endl;
49     cin >> x1 >> y1 >> number;
50     my_vector vec1(x1,y1);
51     vec1 *= number;
52     cout << vec1.get_x() << ' ' << vec1.get_y() << endl;
53 }
54
55 void work_with_vectors::module() const
56 {
57     int x1,y1;
58     double result;
59     cout << "Введите координаты вектора." << endl;
60     cin >> x1 >> y1;
61     my_vector vec1(x1,y1);
62     result = vec1.module();
63     cout << result << endl;
```

```
}

void work_with_vectors::just_exception() const
{
    int x1,y1;
    cout << "Введите координаты вектора." << endl;
    cin >> x1 >> y1;
    my_vector vec1(x1,y1);
    try{
        vec1.just_exception();
    }
    catch(TestException& e){
        cout << e.get_error() << endl;
    }
}
```

```
#include "vectors.h"
#include <iostream>
#include <math.h>

using namespace std;

void my_vector::operator +=(const my_vector arg_sum)
{
    x += arg_sum.x;
    y += arg_sum.y;
}

void my_vector::operator -=(const my_vector arg_sub)
{
    x -= arg_sub.x;
    y -= arg_sub.y;
}

int my_vector::scalar_product(const my_vector arg1) const{
    return arg1.x * x + arg1.y * y;
}

void my_vector::operator *=(const int arg_z)
{
    x *= arg_z;
    y *= arg_z;
}

double my_vector::module() const{
    return sqrt(x * x + y * y);
}

int my_vector::get_x() const{
```

```
34      return x;
35  }
36  int my_vector::get_y() const{
37      return y;
38  }
39  void my_vector::set_x(int xx){
40      x = xx;
41  }
42  void my_vector::set_y(int yy){
43      y = yy;
44  }
45  void my_vector::just_exception() const{
46      throw TestException();
47  }
```

Также в рамках задания были переписаны все программы, которые были сделаны на c, на c++.

5.1.7 Задача 1

```
1   #include "convert_class.h"
2
3   int imperial_system::get_ft() const{
4       return ft;
5   }
6
7   int imperial_system::get_m() const{
8       return m;
9   }
10
11  int imperial_system::get_yd() const{
12      return yd;
13  }
14
15  void imperial_system::set_ft(const int arg_ft){
16      ft = arg_ft;
17      while (ft >= 3){
18          set_yd(yd + 1);
19      }
20  }
21
22  void imperial_system::set_m(const int arg_m){
23      m = arg_m;
24  }
25
26  void imperial_system::set_yd(const int arg_yd){
27      yd = arg_yd;
28      while (yd >= 2000){
```

```
29          set_m(m + 1);
30      }
31  }
32
33  void imperial_system::ft_in_m(){
34      m = ft / 6000;
35      ft = ft - m * 6000;
36  }
37
38  void imperial_system::yd_in_ft(){
39      yd = ft / 3;
40      ft = ft - yd * 3;
41  }
```

5.1.8 Задача 2

```
1  #include "check_class.h"
2
3  types_of_triangles triangles::check_triangles() const
4  {
5      if (a <= 0 || b <= 0 || c <= 0)
6          return incorrectly;
7      else{
8          if (a + b > c && b + c > a && a + c > b){
9              if (a == b && b == c){
10                 return equilateral;
11             }
12             if ((a == b) || (a == c) || (b == c)){
13                 return isosceles;
14             }
15             return possible;
16         }
17         else{
18             return impossible;
19         }
20     }
21 }
22
23 int triangles::get_a() const
24 {
25     return a;
26 }
27
28 int triangles::get_b() const
29 {
30     return b;
31 }
```

```
32
33 int triangles::get_c() const
34 {
35     return c;
36 }
```

5.1.9 Задача 3

```
1  #include "removal_class.h"
2
3  int removal::removal_odd(int number){
4      int numeral, result = 0, digit = 1;
5      while(number > 0){
6          numeral = number % 10;
7          number = number / 10;
8          if (numeral % 2 == 0){
9              result = result + digit * numeral;
10             digit = digit * 10;
11         }
12     }
13     return result;
14 }
```

5.1.10 Задача 4

```
1  #include "square_class.h"
2  #include <exception>
3
4  using namespace std;
5
6  int latin_square::get_n() const
7  {
8      return n;
9  }
10
11 bool latin_square::check_latin_square() const
12 {
13         int i,j,l;
14         bool flag;
15         for (i = 0; i < n; i++){
16             for (j = 1; j < n+1; j++){
17                 flag = false;
18                 for (l = 0; l < n; l++){
19                     if (two_dim[i][l] == j){
20                         flag = true;
```

```
                    }
                }
                if (flag == false){
                    return flag;
                }
            }
        }
        for (i = 0; i < n; i++){
            for (j = 1; j < n+1; j++){
                flag = false;
                for (l = 0; l < n; l++){
                    if (two_dim[l][i] == j){
                        flag = true;
                    }
                }
                if (flag == false){
                    return flag;
                }
            }
        }
        return flag;
}

void latin_square::set_member_two_dim(const int arg_i, const
    int arg_j, const int arg)
{
    if (arg_i > n - 1 || arg_j > n - 1){
        throw WrongAdressException(arg_i, arg_j);
    }
    two_dim[arg_i][arg_j] = arg;
}
```

5.1.11 Задача 5

```
#include "removing_words_class.h"
#include <string>

using namespace std;


string removing_words::removing_repeating_words(string str)
{
    int found = 0,found_end = 0;
    string word, sub_str,result;
    sub_str = str;
    while(found_end >= 0){
        word = sub_str.substr(0, sub_str.find(' '));
```

```
14          sub_str.erase(0, sub_str.find(' ') + 1);
15          found = sub_str.find(word);
16          if (found != 0){
17              result = result + word + ' ';
18          }
19          found_end = sub_str.find(' ');
20      }
21      result += sub_str;
22      return result;
23  }
```