

Программирование

Корсков А. В.

24 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 библиотеки 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 функции `convert_UI`. Она считывает введенные 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 количество миль. Далее находится количество футов без миль. На следующем шаге вычисляется количество ярдов. И на последнем шаге находится количество 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 виртуальная машина с операционной системой Debian 8.1, в которой установлен QtCreator 3.5.0. В конце использовалась утилита `srrcheck` 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 количество футов, равное 9139. Ожидается, что результатом работы программы будет количество футов равное 1, количество ярдов равное 1046 и количество 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 `srrcheck`, 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)
```

```

9 {
10     puts("1. Перевод футов в мили, ярды и футы.");
11     puts("2. Проверка допустимости треугольника.");
12     puts("3. Удаление из числа нечётных чисел.");
13     puts("4. Проверить является ли двумерный массив латинским
        квадратом.");
14     puts("5. Удаление повторяющихся слов.");
15     int choice;
16     scanf("%d", &choice);
17     switch(choice){
18         case 1:
19             convert_UI();
20             break;
21         case 2:
22             check_UI();
23             break;
24         case 3:
25             removal_UI();
26             break;
27         case 4:
28             square_UI();
29         case 5:
30             removing_words_UI();
31     }
32     return 0;
33 }

```

```

1 #include <stdio.h>
2 #include "convert.h"
3
4 void convert_UI(){
5     puts("Введите количество футов.");
6     struct measure_units st1;
7     scanf("%d", &st1.ft);
8     convert(&st1);
9     printf("Количество миль: %d, количество ярдов: %d колли
        чество футов: %d \n", st1.m, st1.yd, st1.ft);
10 }

```

```

1 #include <stdio.h>
2 #include "convert.h"
3
4 void convert(struct measure_units *arg){
5     arg->m = arg->ft / 6000;
6     arg->ft = arg->ft - arg->m * 6000;
7     arg->yd = arg->ft / 3;
8     arg->ft = arg->ft - arg->yd * 3;
9 }

```

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 алгоритма были использованы функции стандартной библиотеки 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 функции `check_UI`. Она считывает введённые 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 равнобедренным. Если это условие выполняется, проверяем будет ли данный треугольник равносторонним.

1.2.4 Описание тестового стенда и методики тестирования

Для решения задачи использовалась виртуальная машина с операционной системой Debian 8.1, в которой установлен QtCreator 3.5.0. В конце использовалась утилита `crrcheck` 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 `srcheck`, 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`.

```
1 #include <stdio.h>
2 #include "check.h"
3
4 void check_UI(){
5     puts("Введите три стороны треугольника.");
6     int a,b,c,result_check;
7     scanf("%d%d%d", &a, &b, &c);
8     result_check = check(a,b,c);
9     /// Почему здесь не используются епит?
10    switch(result_check){
11        case 0:
12            printf("Данные некорректны.\n");
13            break;
14        case 1:
15            printf("Данный треугольник возможен.\n");
16            break;
17        case 2:
18            printf("Данный треугольник является равнобедренны
19                    м.\n");
20            break;
21        case 3:
22            printf("Данный треугольник является равносторонни
23                    м.\n");
24            break;
25        case 4:
26            printf("Данный треугольник не возможен.\n");
27    }
```

```
1 #include <stdio.h>
2 #include "check.h"
```

```

3
4 int check(int a, int b, int c){
5     enum types_of_triangles {incorrectly, possible, isosceles
6         , equilateral, impossible};
7     if (a <= 0 || b <= 0 || c <= 0)
8         return incorrectly;
9     else{
10         if (a + b > c && b + c > a && a + c > b){
11             if (a == b && b == c){
12                 return equilateral;
13             }
14             if ((a == b) || (a == c) || (b == c)){
15                 return isosceles;
16             }
17             return possible;
18         }
19         else{
20             return impossible;
21         }
22 }

```

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 библиотеки для ввода и вывода информации.

2.1.3 Проектирование

В функции `main` вызывается функция `removal_UI`. Взаимодействие с пользователем реализовано в функции `removal_UI`. Она считывает введённые 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 виртуальная машина с операционной системой Debian 8.1, в которой установлен QtCreator 3.5.0. В конце использовалась утилита crrcheck 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 crrcheck, 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.

```
1 #include <stdio.h>
2 #include "removal.h"
3
4 void removal_UI(){
5     puts("Введите число.");
6     int number, result_removal;
7     scanf("%d", &number);
8     result_removal = removal(number);
9     printf("Результат: %d \n", result_removal);
10 }
```

```
1 #include <stdio.h>
2 #include "removal.h"
3
4 int removal(int number){
5     int result = 0, digit = 1;
6     while(number > 0){
7         int numeral = number % 10;
8         number = number / 10;
9         if (numeral % 2 == 0){
10             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 $n \times n$, 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 библиотеки 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 функции `square_UI`. Она считывает введённые 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 виртуальная машина с операционной системой Debian 8.1, в которой установлен QtCreator 3.5.0. В конце использовалась утилита crrcheck 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 crrcheck, 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++){
```

```

18         fscanf(iFile, "%d", &two_dim[i][j]);
19     }
20 }
21 fclose(iFile);
22 result_square = square(two_dim,n);
23 for (i = 0; i < n; i++){
24     free(two_dim[i]);
25 }
26 free(two_dim);
27 FILE * oFile;
28 oFile = fopen("output", "w");
29 if (result_square == 1){
30     fprintf(oFile, "Матрица является латинским квадратом.
31         ");
32 }
33 else{
34     fprintf(oFile, "Матрица не является латинским квадрат
35         ом.");
36 }
37 fclose(oFile);
38 }

```

```

1 int square(int** two_dim, int n){
2     int i,j,flag,l;
3     for (i = 0; i < n; i++){
4         for (j = 1; j < n+1; j++){
5             flag = 0;
6             for (l = 0; l < n; l++){
7                 if (two_dim[i][l] == j){
8                     flag = 1;
9                 }
10            }
11            if (flag == 0){
12                return flag;
13            }
14        }
15    }
16    for (i = 0; i < n; i++){
17        for (j = 1; j < n+1; j++){
18            flag = 0;
19            for (l = 0; l < n; l++){
20                if (two_dim[l][i] == j){
21                    flag = 1;
22                }
23            }
24            if (flag == 0){
25                return flag;
26            }
27        }
28    }

```

```
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 библиотеки 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 функции `removing_words_UI`. Она считывает введённые 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 виртуальная 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.

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include "removing_words.h"
4
5 void removing_words_UI() {
6     FILE * iFile;
7     iFile = fopen("input_removing_words", "r");
8     char string[100], result[100] = "";
9     fgets(string, 100, iFile);
10    fclose(iFile);
11    removing_words(string, result);
12    FILE * oFile;
13    oFile = fopen("output_removing_words", "w");
14    fputs(result, oFile);
15    fclose(oFile);
16 }
```

```
1 #include "removing_words.h"
2 #include <string.h>
3 #include <stdlib.h>
4 #include <stdio.h>
5
6 void removing_words(char string[], char *result) {
```



```

7   int n,i = 0;
8   n = strlen(string);
9   char word[100];
10  while(i <= n){
11      int l=0;
12      while ((i < n) && (string[i] != ' ')){
13          word[l] = string[i];
14          i++;
15          l++;
16      }
17      word[l]='\0';
18      int j = i + 1;
19      int k = 0;
20      int flag = 1;
21      for (j=i+1;j <= i + l; j++){
22          if (word[k] != string[j]){
23              flag = 0;
24          }
25          k++;
26      }
27      if (flag == 0){
28          strcat(result, word);
29      }
30      i++;
31  }
32 }

```

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 $\sqrt{x_1^2 + y_1^2}$.

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 функции `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 виртуальная машина с операционной системой Debian 8.1, в которой установлен QtCreator 3.5.0. В конце использовалась утилита `srpcheck` 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 `srpcheck`, 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 <iostream>
2 #include "vectors.h"
3 #include "addition.h"
4 #include "subtraction.h"
5 #include "module.h"
6 #include "multiply.h"
7 #include "scalar_product.h"
8
9 using namespace std;
10
11 void work_with_vectors() {
12     int choice;
13     cin >> choice;
14     my_vector vec1(2,2);
```

```

15     switch (choice){
16         case 1:
17             addition();
18         break;
19         case 2:
20             subtraction();
21         break;
22         case 3:
23             try{
24                 vec1.just_exception();
25             }
26             catch(TestException& e){
27                 cout << e.get_error() << endl;
28             }
29         break;
30         case 4:
31             scalar_product();
32         break;
33         case 5:
34             multiply();
35         break;
36         case 6:
37             module();
38         break;
39     }
40 }

```

```

1  #include "vectors.h"
2  #include <iostream>
3  #include <math.h>
4
5  using namespace std;
6
7  void my_vector::operator +=(my_vector arg_sum)
8  {
9      x += arg_sum.x;
10     y += arg_sum.y;
11 }
12
13 void my_vector::operator -=(my_vector arg_sub)
14 {
15     x -= arg_sub.x;
16     y -= arg_sub.y;
17 }
18
19 int my_vector::scalar_product(my_vector arg1) const{
20     return arg1.x * x + arg1.y * y;
21 }
22

```

```

23 void my_vector::multiply(int z){
24     x *= z;
25     y *= z;
26 }
27
28 double my_vector::module() const{
29     return sqrt(x * x + y * y);
30 }
31
32 int my_vector::get_x() const{
33     return x;
34 }
35 int my_vector::get_y() const{
36     return y;
37 }
38 void my_vector::set_x(int xx){
39     x = xx;
40 }
41 void my_vector::set_y(int yy){
42     y = yy;
43 }
44 void my_vector::just_exception() const{
45     throw TestException();
46 }

```

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

5.1.7 Задача 1

```

1  #include "convert_class.h"
2
3  int convert_class::get_ft() const{
4      return ft;
5  }
6
7  int convert_class::get_m() const{
8      return m;
9  }
10
11 int convert_class::get_yd() const{
12     return yd;
13 }
14
15 void convert_class::set_ft(int arg_ft){
16     ft = arg_ft;
17     while (ft >= 3){
18         set_yd(yd + 1);

```

```

19     }
20 }
21
22 void convert_class::set_m(int arg_m){
23     m = arg_m;
24 }
25
26 void convert_class::set_yd(int arg_yd){
27     yd = arg_yd;
28     while (yd >= 2000){
29         set_m(m + 1);
30     }
31 }
32
33 void convert_class::ft_in_m(){
34     m = ft / 6000;
35     ft = ft - m * 6000;
36 }
37
38 void convert_class::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 int check_class::check_triangles() const
4 {
5     enum types_of_triangles {incorrectly, possible, isosceles
6     , equilateral, impossible};
7     if (a <= 0 || b <= 0 || c <= 0)
8         return incorrectly;
9     else{
10         if (a + b > c && b + c > a && a + c > b){
11             if (a == b && b == c){
12                 return equilateral;
13             }
14             if ((a == b) || (a == c) || (b == c)){
15                 return isosceles;
16             }
17             return possible;
18         }
19         else{
20             return impossible;
21         }
22     }
23 }

```

```

21     }
22 }
23
24 void check_class::set_a(int arg_a)
25 {
26     a = arg_a;
27 }
28
29 void check_class::set_b(int arg_b)
30 {
31     b = arg_b;
32 }
33
34 void check_class::set_c(int arg_c)
35 {
36     c = arg_c;
37 }
38
39 int check_class::get_a() const
40 {
41     return a;
42 }
43
44 int check_class::get_b() const
45 {
46     return b;
47 }
48
49 int check_class::get_c() const
50 {
51     return c;
52 }

```

5.1.9 Задача 3

```

1 #include "removal_class.h"
2
3 int removal_class::removal(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  square_class::square_class(int arg_n)
7  {
8      n = arg_n;
9      two_dim.resize(n);
10     for (int i = 0; i < n; ++i){
11         two_dim[i].resize(n);
12     }
13 }
14
15 int square_class::get_n() const
16 {
17     return n;
18 }
19
20 bool square_class::check_latin_square()
21 {
22     int i,j,l;
23     bool flag;
24     for (i = 0; i < n; i++){
25         for (j = 1; j < n+1; j++){
26             flag = false;
27             for (l = 0; l < n; l++){
28                 if (two_dim[i][l] == j){
29                     flag = true;
30                 }
31             }
32             if (flag == false){
33                 return flag;
34             }
35         }
36     }
37     for (i = 0; i < n; i++){
38         for (j = 1; j < n+1; j++){
39             flag = false;
40             for (l = 0; l < n; l++){
41                 if (two_dim[l][i] == j){
42                     flag = true;
```



```

43         }
44     }
45     if (flag == false){
46         return flag;
47     }
48 }
49 }
50 return flag;
51 }
52
53 void square_class::set_member_two_dim(int arg_i, int arg_j,
54 int arg)
55 {
56     if (arg_i > n || arg_j > n){
57         throw WrongAdressException(arg_i, arg_j);
58     }
59     two_dim[arg_i][arg_j] = arg;

```

5.1.11 Задача 5

```

1 #include "removing_words_class.h"
2 #include <string>
3
4 using namespace std;
5
6
7 string removing_words_class::removing_words(string str)
8 {
9     int found = 0, found_end = 0;
10    string word, sub_str, result;
11    sub_str = str;
12    while(found_end >= 0){
13        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 }

```