



МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение  
высшего образования

«МИРЭА – Российский технологический университет»

**РТУ МИРЭА**

---

Институт Информационных технологий

Кафедра Математического обеспечения и стандартизации информационных  
технологий

## **ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ № 6**

**по дисциплине**

**«Структуры и алгоритмы обработки данных»**

**Тема: «Рекурсивные функции»**

Выполнил студент группы ИКБО-18-22

Ракитин В.А.

Принял преподаватель

Филатов А.С.

Лабораторная работа выполнена

«\_\_»\_\_\_\_\_202\_\_ г.

*(подпись студента)*

«Зачтено»

«\_\_»\_\_\_\_\_202\_\_ г.

*(подпись руководителя)*

Москва 2023

## **1. Цель работы**

Получить знания и практические навыки по разработке и реализации рекурсивных процессов.

## **2. Постановка задачи**

Разработать и протестировать рекурсивные функции в соответствии с задачами варианта. Составить отчет.

Требования к выполнению первой задачи варианта:

- приведите итерационный алгоритм решения задачи
- реализуйте алгоритм в виде функции и отладьте его
- определите теоретическую сложность алгоритма
- реализуйте и отладьте рекурсивную функцию решения задачи
- определите глубину рекурсии, изменяя исходные данные
- определите сложность рекурсивного алгоритма, используя метод подстановки и дерево рекурсии
- приведите для одного из значений схему рекурсивных вызовов
- разработайте программу, демонстрирующую выполнение обеих функций, и покажите результаты тестирования.

Требования к выполнению второй задачи варианта:

- рекурсивную функцию для обработки списковой структуры согласно варианту. Информационная часть узла – простого типа – целого;
- определите глубину рекурсии
- определите теоретическую сложность алгоритма
- разработайте программу, демонстрирующую работу функций и покажите результаты тестов.

Вариант №6. Условие задания:

Упражнение 1	Сколько квадратов можно отрезать от прямоугольника со сторонами <i>a</i> и <i>b</i> .
Упражнение 2	Удаление связного стека.

### 3. Решение

Рекурсивные функции - это функции, которые вызывают сами себя. Рекурсия может быть использована для решения задач, которые могут быть разбиты на более мелкие подзадачи. Рекурсивная функция должна иметь базовый случай, который приводит к остановке рекурсии, и рекурсивный случай, который вызывает функцию снова с измененными параметрами. Рекурсивные функции могут быть опасными, если они не останавливаются, так что нужно быть осторожным при их использовании.

Для решения первого упражнения была написана рекурсивная функция `deleteSquares`, которая рекурсивно удаляет квадраты из прямоугольника. На вход функция получает два целочисленных значения: длину и ширину прямоугольника. Далее функция отрезает от заданного прямоугольника квадраты и функция вызывает саму себя.

Глубина данной рекурсивной функции зависит от разницы между длиной и шириной прямоугольника. Если длина и ширина равны, то функция вернет 1 и рекурсия завершится. В противном случае, функция будет рекурсивно вызываться для нового прямоугольника, полученного путем отрезания квадрата от большего прямоугольника, пока одна из сторон не станет равной другой.

Сложность данной рекурсивной функции равна  $O(\log N)$ , где  $N$  - максимальная сторона прямоугольника. Это происходит из-за того, что на каждой рекурсивной итерации мы отрезаем квадрат, который составляет не более чем половину от максимальной стороны оригинального

прямоугольника. Как только стороны прямоугольника станут равными и функция вернет 1, перебор остановится.

```
// Рекурсивная функция для подсчета количества квадратов.  
//Сложность данной рекурсивной функции равна  $O(\log N)$ , где N - максимальная  
сторона прямоугольника.  
int deleteSquares(int lenght, int width) {  
    int c;  
    if (lenght == width) { //Если длина равна ширине, то это квадрат  
        return 1;  
    }  
    else if (lenght < width) { // Если длина меньше ширины,  
        return deleteSquares(width, lenght); //то меняем стороны местами  
    }  
    else { //Если длина больше ширины, то отрезаем квадрат и рекурсивно  
        вызываем функцию  
        return 1 + deleteSquares(lenght - width, width);  
    }  
}
```

Схема рекурсивных вызовов показана на рисунке 1.

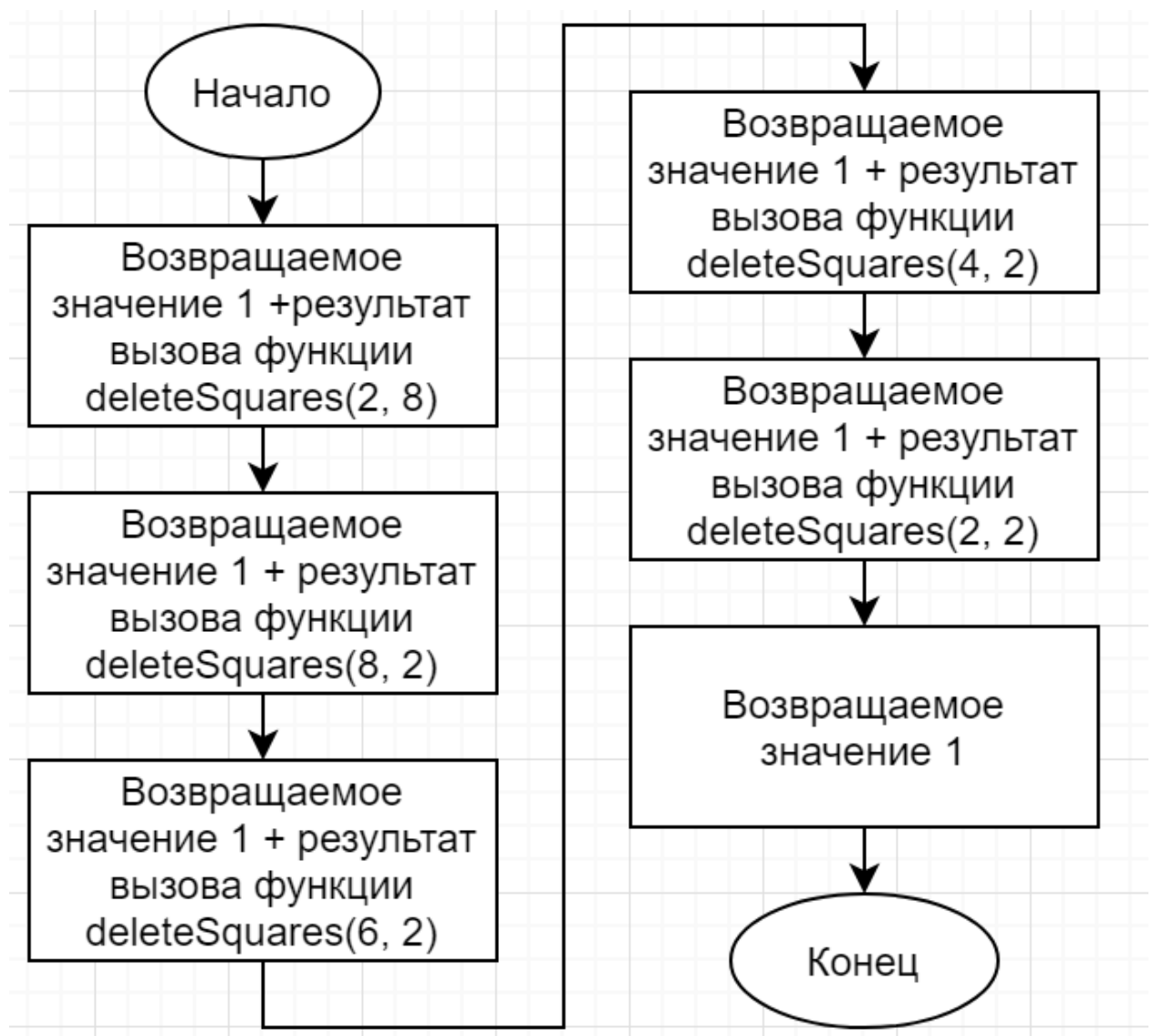


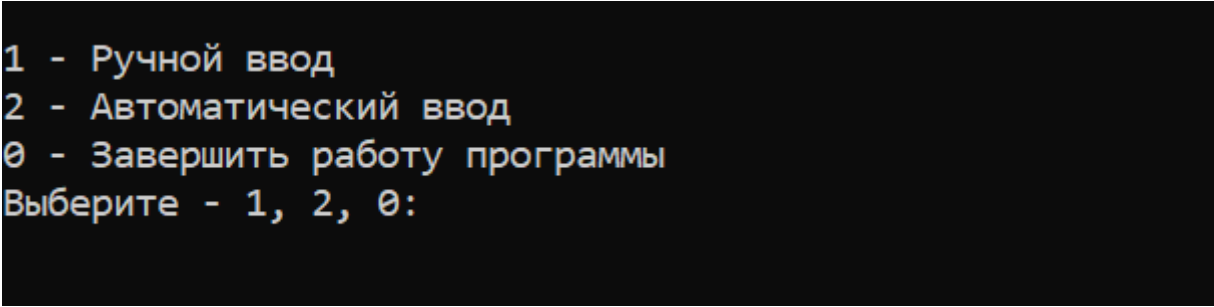
Рисунок 1. Схема рекурсивных вызовов

Для решения второго упражнения была написана функция `deleteList`, которая рекурсивно удаляет элемент из стека. На вход функция получает элемент стека. Далее функция удаляет этот элемент из стека и функция вызывает саму себя.

Глубина рекурсии будет равна количеству элементов в списке. Сложность рекурсии будет линейной.

```
// Рекурсивная функция удаления всех элементов односвязного списка
// сложность данной рекурсивной функции будет линейной, O(N)
void deleteList(Node*& head) {
    if (head == nullptr) {
        return;
    }
    deleteList(head->next);
    cout << "Удаляем элемент со значением " << head->data << endl;
    delete head;
    head = nullptr;
}
```

При запуске программы пользователь видит пользовательское меню, где пользователю надо выбрать тип ввода (ручной или автоматический) или завершить работу программы.



```
1 - Ручной ввод
2 - Автоматический ввод
0 - Завершить работу программы
Выберите - 1, 2, 0:
```

Рисунок 2. Интерфейс программы

#### 4. Тестирование

Протестируем программой выполнение упражнений. Ручным вводом введём некорректные длину и ширину прямоугольника и посмотрим реакцию программы на данный ввод. На рисунке 2 видно, что программа реагирует на

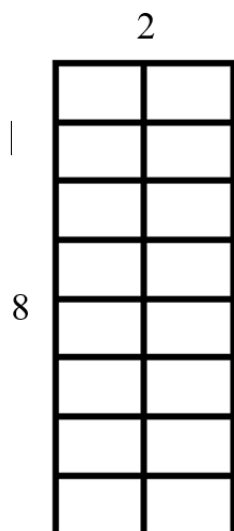
неверные значения и просит пользователя ввести новые значения длины и ширины.

```
1 - Ручной ввод
2 - Автоматический ввод
0 - Завершить работу программы
Выберите - 1, 2, 0: 1
Введите длину и ширину прямоугольника через пробел:
45 0
Вы ввели число, меньшее или равное нулю. Введите длину и ширину прямоугольника ещё раз:
-2 3
Вы ввели число, меньшее или равное нулю. Введите длину и ширину прямоугольника ещё раз:
-2 -5
Вы ввели число, меньшее или равное нулю. Введите длину и ширину прямоугольника ещё раз:
0 5
Вы ввели число, меньшее или равное нулю. Введите длину и ширину прямоугольника ещё раз:
34 -1
Вы ввели число, меньшее или равное нулю. Введите длину и ширину прямоугольника ещё раз:
_
```

Рисунок 3. Реакция программы на некорректный ввод

Протестируем программой выполнение упражнения. Автоматическим вводом введём размеры прямоугольника. В данном случае длина и ширина равны 2 и 8 соответственно. Далее программа удалила 4 квадрата размера 2x2. На рисунке 3 показан прямоугольник, заданный на входе программы и какие квадраты удалили.

Прямоугольник, который получила программа



Какие квадраты программа удаляет

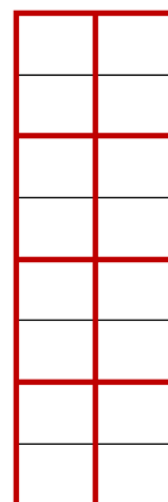


Рисунок 4. Иллюстрация работы программы

Далее в стек помещаются наши квадраты, а затем программа удаляет элементы стека и информирует пользователя об отсутствии элементов в стеке. На рисунке 4 программа вывела верный результат.

```
1 - Ручной ввод
2 - Автоматический ввод
0 - Завершить работу программы
Выберите - 1, 2, 0: 2

Стороны прямоугольника: 2 и 8
Количество квадратов, которые можно получить из этого прямоугольника: 4

Стек: 4 3 2 1
Удаляем элемент со значением 1
Удаляем элемент со значением 2
Удаляем элемент со значением 3
Удаляем элемент со значением 4
Стек успешно удалён!

1 - Ручной ввод
2 - Автоматический ввод
0 - Завершить работу программы
Выберите - 1, 2, 0: _
```

Рисунок 5. Решение программой упражнений

## 5. Вывод

В результате выполнения работы я:

1. Получил знания и практические навыки по разработке и реализации рекурсивных функций
2. Получил знания по вычислению сложности рекурсивных функций

## 6. Исходный код программы

```
#include <iostream>
using namespace std;

//Структура односвязного списка
struct Node {
    int data;
    Node* next;
    Node(int d) : data(d), next(nullptr) {}
};

// функция для добавления элемента в конец односвязного списка
void push_back(Node*& head, int data) {
    Node* newNode = new Node(data);
```





```

        if (length <= 0 or width <= 0) {
            cout << "Вы ввели число, меньшее или равное нулю.  
Введите длину и ширину прямоугольника ещё раз: " << endl;
        }
        break;
    case(2):
        length = rand() % 10 + 1;
        width = rand() % 10 + 1;
        break;
    case(0):
        return 0;
    default:
        break;
    }
    cout << endl << "Стороны прямоугольника: " << length << " и " <<
width << endl;
    int countSquares = deleteSquares(length, width);
    cout << "Количество квадратов, которые можно получить из этого  
прямоугольника: " << countSquares << endl;

    Node* head = nullptr;
    // создание списка из элементов, представляющих квадраты
    for (int i = 1; i <= countSquares; i++) {
        push_back(head, i);
    }

    // удаление списка с помощью рекурсивной функции
    cout << endl;
    cout << "Стек: ";
    printList(head);
    deleteList(head);
    cout << "Стек успешно удалён!" << endl;
}
}

```