МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение

высшего образования

**«Нижегородский государственный университет им. Н.И. Лобачевского»**

**Национальный исследовательский университет**

**Институт информационных технологий, математики и механики**

**Кафедра математического обеспечения и суперкомпьютерных технологий**

**Отчет по учебной практике**

**«Структура хранения односвязного списка с использованием массивов»**

**Выполнил:** студент группы 381706-1

Полетуева Анастасия Николаевна

\_\_\_\_\_\_\_\_\_\_\_ Подпись

**Научный руководитель:**

ассистент каф. МОСТ ИИТММ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Лебедев И.Г

Нижний Новгород

2018.

Содержание

[1. Введение 3](#_Toc536801998)

[2. Постановка задачи 4](#_Toc536801999)

[3. Руководство пользователя 5](#_Toc536802000)

[4. Руководство программиста 6](#_Toc536802001)

[4.1. Описание структуры программы 6](#_Toc536802002)

[4.2. Описание структур данных 6](#_Toc536802003)

[4.3. Описание алгоритмов 7](#_Toc536802004)

[5. Заключение 9](#_Toc536802005)

# Введение

**Основная цель данной работы** – разработка динамической структуры данных - линейного односвязного списка с использованием массивов.

**Односвязный список** – это структура данных, состоящая из элементов одного типа, связанных между собой последовательно посредством указателей.

Для создания такого списка используется два массива одинакового размера: содержательный массив и массив индексов. Эти массивы связанны друг с другом по индексу. По фиксированному *i*, содержательный массив, в *i*-й ячейке, хранит значение элемента списка, а массив индексов, в *i*-й ячейке, содержит индекс следующего элемента списка в содержательном массиве.

Если список заполнен не полностью, то для пустых ячеек в содержательном массиве, в соответствующих ячейках в массиве индексов лежит значение «-2». Для последнего элемента списка, в соответствующей ячейке массива индексов лежит значение -1.

Как правило, стоит дополнительно хранить индекс первого элемента списка в содержательном массиве.

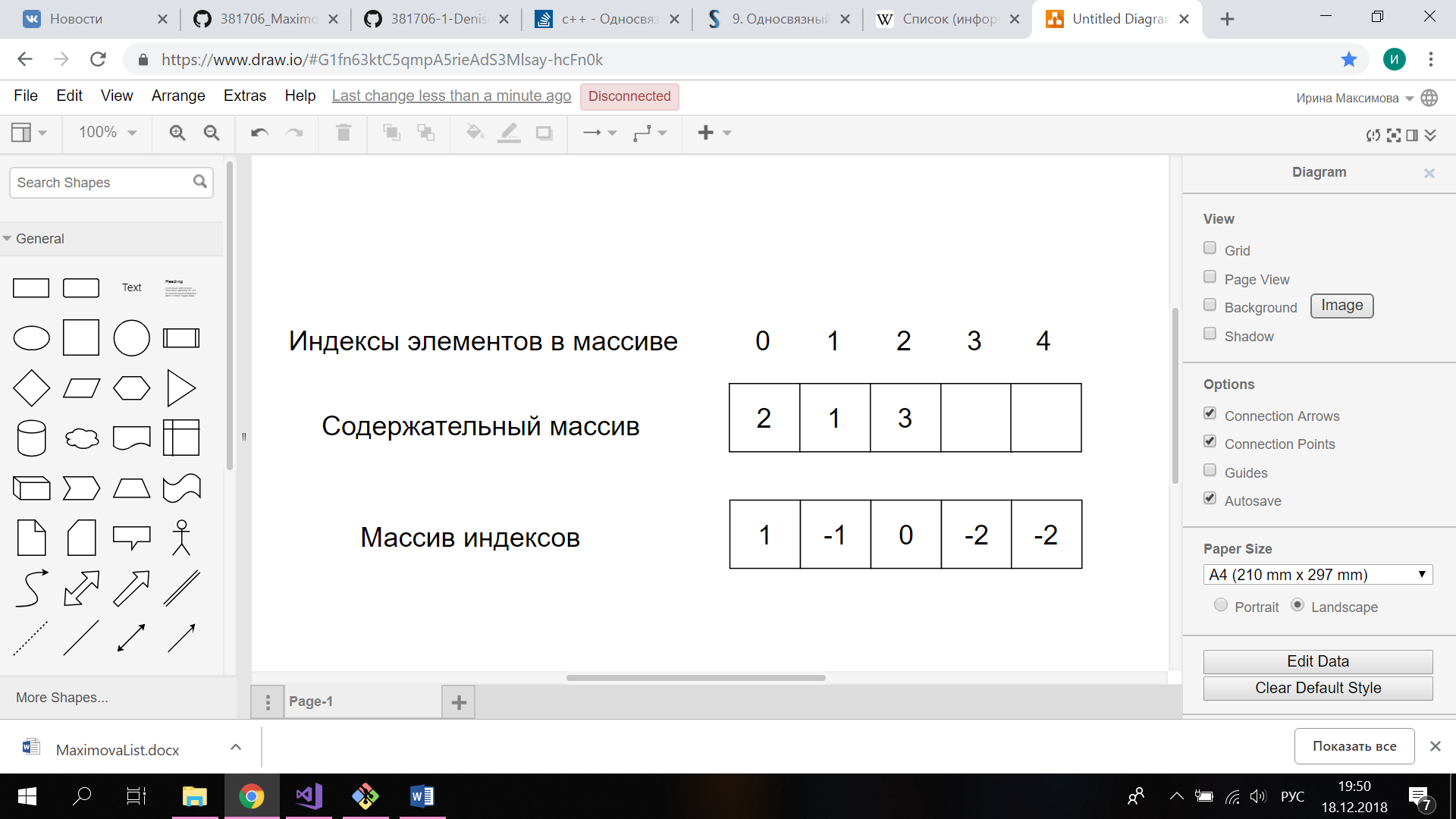


Рисунок 1 Представление списка на массивах в памяти компьютера

# Постановка задачи

Выполнение работы предполагает решение следующих задач:

1. Разработка и реализация класса списка – TArrList.
2. Пример программы, демонстрирующая работу класса TArrList.
3. Написание набора автоматических тестов с использованием Google C++ Testing Framework и проверка работоспособности методов классов.

Для работы со списком необходимо реализовать операции:

* добавления элемента в начало списка,
* добавления элемента в конец списка,
* добавление и извлечение промежуточного элемента списка,
* извлечения с удалением элемента из начала списка,
* извлечения с удалением элемента из конца списка,
* проверка списка на полноту/пустоту.

# Руководство пользователя

При запуске программы пользователя просят ввести размер будущего списка. После чего получившийся список выводится на экран.

После чего на экран будут выведены первый и последний элемент данного списка и обновленный список уже без этих элементов. Затем, пользователю предлагается изъять один элемент из списка, введя его индекс. Значение элемента с таким индексом будет выведено на экран.

# Руководство программиста

## Описание структуры программы

Программа состоит из следующих модулей:

* Модуль ArrListLib. Статическая библиотека. Включает в себя заголовочный файл ArrList.h, в котором описаны методы с реализаций шаблонного класса *TArrList.*
* Модуль ArrListTest. Набор тестов для класса TArrList. Включает в себя файл Arr*ListTest.cpp.* Разработаны они с помощью использования Google C++ Testing Framework.
* Модуль ArrList. Пример использования списка. Включает в себя файл с реализацией *main.cpp.*

## Описание структур данных

**Класс TArrList:**

T\* mas; - массив элементов списка

int \*nextInd; - массив индексов, указывающих на следующий элемент списка

int \*predInd; - массив индексов, указывающих на предыдущий элемент списка

int size; - Длина списка

int count; - количество элементов в списке

int start; - индекс первого элемента в списке

int end; - индекс последнего элемента в списке

TQueue<int> freeElem – очередь свободных ячеек в массиве elems

TArrList(int \_size = 10);

TArrList(TArrList<T> &A);

~TArrList();

void Put(int n, T elem); - добавить элемент в указанную позицию

void PutStart(T elem); - положить в начало

void PutEnd(T elem); - положить в конец

T Get(int n); - получить элемент из списка

T GetStart(); - получить первый элемент

T GetEnd(); - получить последний элемент

bool IsFull(); - проверка на полноту

bool IsEmpty(); - проверка на пустоту

void Print(); - вывод листа на экран

## Описание алгоритмов

**Добавление элемента списка в начало и в конец.**

Изначально идет проверка на полноту списка. Если он заполнен, то бросаем исключение. Если нет, то в очереди свободных позиций *freeElem*, берем первую свободную ячейку *i*. По полученному индексу в массив *mas* записываем значение, которое хотим положить в список. Определяем, что следующим для этого элемента, будет элемент с текущим индексом start, то есть *nextInd[i] = start*. Если, перед добавлением, список не был пуст, то предыдущим для первого элемента списка делаем только что добавленный элемент, то есть: *predInd[start] = i.* Если же список был пуст, то определяем, что добавленный элемент является и последним элементом в списке. Затем индекс *start* переопределяем на только что добавленный элемент: *start = i*. Увеличиваем количество элементов в списке *count++*.

Для добавления элемента в конец списка, рассуждения аналогичны.

**Удаление элемента списка из начала и из конца.**

Изначально идет проверка на пустоту списка. Если он пуст, то бросаем исключение. Если нет, то в переменную *mas* записываем элемент в начале списка mas*[start].* Затем переопределяем индекс первого элемента списка, он будет равен индексу элемента следующим за первым: *newstart = nextInd[start].* Затем в очередь свободных позиций *freeElem*, добавляем освободившуюся после изъятия первого элемента свободную ячейку *start*. Идентифицируем пустыми соответствующие ячейки массивов *nextInd* и *predInd*: *nextInd[start] = predInd[start]* = -2. Если после извлечения первого элемента, список не пуст, то предыдущего элемента для нового первого элемента не существует: *predInd[newstart] = -1*.Затем индекс *start* переопределяем на *newstart*: *start = newstart*. Уменьшаем количество элементов в списке *count--*.

Для извлечения элемента из конца списка, рассуждения аналогичны.

# Заключение

В ходе выполнения, был реализован шаблонный класс списка TArrList, описанный в специально разработанной библиотеке ArrListLib. Все это позволило нам проводить базовые операции со списками: добавление или изъятие элементов из него. Также были написаны автоматические тесты, реализованные с использованием Google C++ Testing Framework, и пример использования списка для пользователя.

1. **Литература**
2. Гергель В.П. Методические материалы по курсу «Методы программирования 2»: [http://www.itmm.unn.ru/files/2018/11/Primer-1.7.-Razrabotka-obshhego-predstavleniya-linejnogo-spiska.pdf], 2015.
3. Википедия: свободная электронная энциклопедия: на русском языке: [https://ru.wikipedia.org/wiki/Список\_(информатика)]