

**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**  
**«Структуры данных: реализация односвязного списка на массивах»**

Выполнил:  
студент группы 361706-1  
Резанцев Сергей Алексеевич  
\_\_\_\_\_ Подпись

Научный руководитель:  
ассистент каф. МОСТ ИИТММ  
\_\_\_\_\_ Лебедев И. Г.

Нижний Новгород  
2018 г.

<b>Содержание</b>	
<b>Введение</b>	<b>3</b>
<b>Постановка задачи</b>	<b>4</b>
<b>Руководство пользователя</b>	<b>5</b>
<b>Руководство программиста</b>	<b>6</b>
Описание структуры программы	6
Описание структуры данных	6
Описание алгоритмов	7
<b>Заключение</b>	<b>9</b>
<b>Литература</b>	<b>10</b>

# 1. Введение

Связный список — базовая динамическая структура данных в информатике, состоящая из узлов, каждый из которых содержит как собственно данные, так и одну или две ссылки («связки») на следующий и/или предыдущий узел списка.

Принципиальным преимуществом перед массивом является структурная гибкость: порядок элементов связного списка может не совпадать с порядком расположения элементов данных в памяти компьютера, а порядок обхода списка всегда явно задаётся его внутренними связями.

Для создания такого списка используется два массива одинакового размера: содержательный массив и массив индексов. Эти массивы связаны 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.

Естественно, что для работы со списком одних только ссылок между элементами недостаточно — надо знать, например, где расположен первый элемент списка, и отличать последний элемент от всех остальных (ведь за ним уже никаких элементов нет). Для этих целей можно было бы ввести две переменные, которые содержали бы индексы начала и конца списка.

Воображаемый  
список

Массив узлов  
списка (Elems)

Массив ссылок,  
порядок следования  
узлов (Refs)

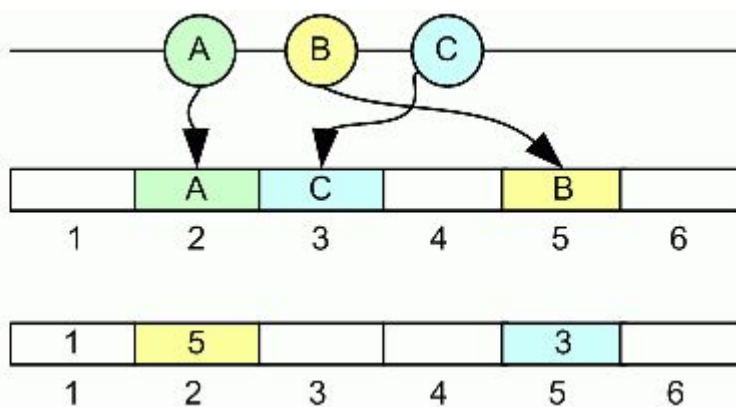


Рисунок 1. Представление списка

## 2. Постановка задачи

Реализовать класс TAList для реализации линейного односвязного списка. Поля классов должны быть закрыты.

В классах обязательно должны присутствовать методы:

- 1) добавление элемента в начало, в конец и текущего
- 2) удаление элемента в начале, в конце и текущего

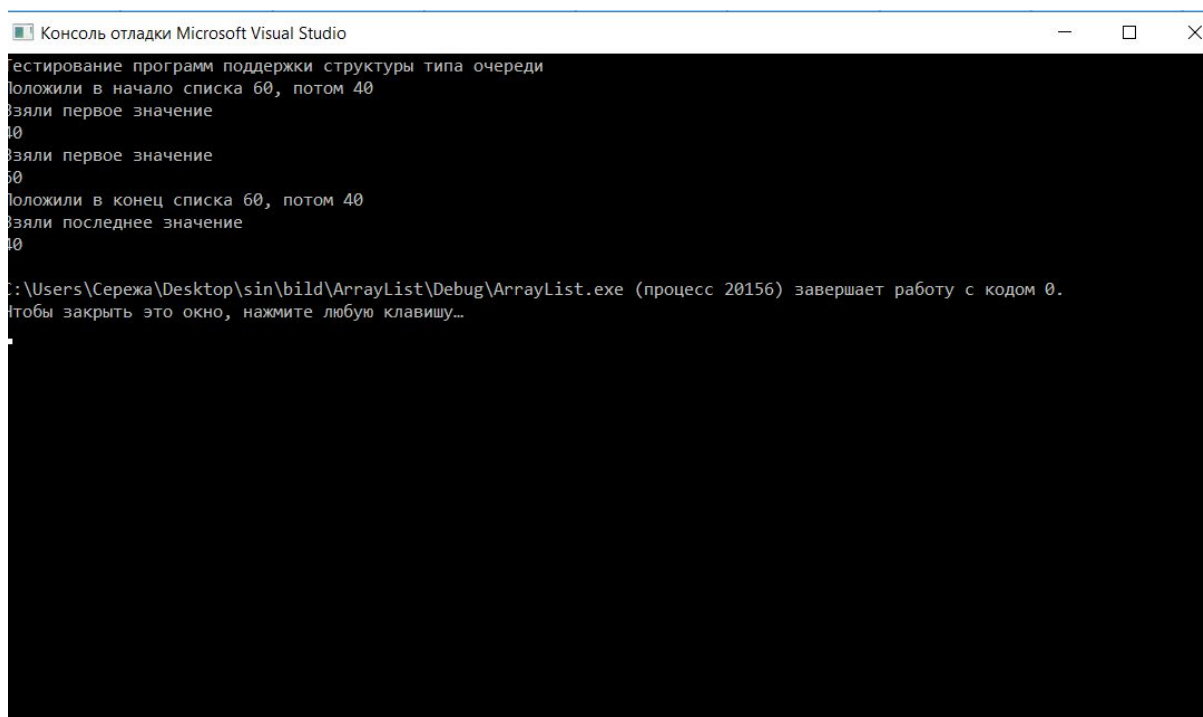
Должны быть реализованы конструкторы: копирования и инициализатор.

Предоставить пример использования и обеспечить работоспособность тестов, покрывающих все методы класса TAList.

### 3. Руководство пользователя

Чтобы начать работу с программой запустите приложение ArrayList.

На экране появится следующее:



```
Консоль отладки Microsoft Visual Studio
тестирование программ поддержки структуры типа очереди
Положили в начало списка 60, потом 40
Взяли первое значение
60
Взяли первое значение
60
Положили в конец списка 60, потом 40
Взяли последнее значение
40

C:\Users\Сергея\Desktop\sin\bild\ArrayList\Debug\ArrayList.exe (процесс 20156) завершает работу с кодом 0.
Чтобы закрыть это окно, нажмите любую клавишу...
```

Рисунок 2. Результат программы, выведенный на консоль.

Затем программа завершится.

## **4. Руководство программиста**

### **4.1. Описание структуры программы**

Для реализации алгоритмов будут использованы классы TStack и TQueue и TAlisT.

Лабораторная работа состоит из следующих модулей:

StackLib

Библиотека, содержащая заголовочный файл TStack.h, в котором содержится класс TStack и реализация его методов, и файл TStack.cpp

QueueLib

Библиотека, содержащая заголовочный файл TQueue.h, в котором содержится класс TQueue - наследник TStack, и реализация его методов, и файл TQueue.cpp

AlisTLib

Библиотека, содержащая заголовочный файл TAlisT.h, в котором содержится класс TAlisT, и реализация его методов, и файл TAlisT.cpp

ArrayList

Пример использования программы.

test

В файле test\_alisT.cpp прописаны тесты, покрывающие каждый метод класса TAlisT.

### **4.2. Описание структуры данных**

Класс TList является шаблонным. В классе 8 полей:

T\* mas – указатель на область памяти для хранения элементов списка.

int \*nextIndex - указатель на область памяти для хранения индексов, указывающих на следующий элемент списка.

int \*prevIndex - указатель на область памяти для хранения индексов, указывающих на предыдущий элемент списка.

int size – максимальный размер списка.

int count – текущее количество элементов в списке.

int start - индекс первого элемента списка.

int end - индекс последнего элемента списка.

TQueue <int> freeElem - Очередь свободных ячеек в массиве mas.

И реализованы следующие функции:

TAList(int \_size = 10) - конструктор по умолчанию.

TAList(TAList<T> &A) - конструктор копирования.

~ TAList() – деструктор.

void PutStart(T elem) – метод, позволяющий добавить элемент в начало списка.

void PutEnd(T elem) – метод, позволяющий добавить элемент в конец списка.

void Put(int n, T elem) - добавить промежуточный элемент на позицию n в списке

T Get(int n) - извлечь из списка элемент на позиции n.

T GetStart() – метод, позволяющий получить с удалением элемент из начала списка.

T GetEnd() – метод, позволяющий получить с удалением элемент из конца списка.

bool IsFull() - проверка списка на полноту.

bool IsEmpty() – проверка списка на пустоту.

void Print() – вывод элементов списка на консоль.

### 4.3. Описание алгоритмов

В данном разделе не будут рассматриваться тривиальные методы.

1)Добавление звена списка в начало и в конец.

Сначала проверяем не заполнен ли список. 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 nextIndex[i] = start. Если, перед добавлением, список не был пуст, то предыдущим для первого элемента списка делаем

только что добавленный элемент, то есть:  $\text{prevIndex}[\text{start}] = i$ . Если же список был пуст, то определяем, что добавленный элемент является и последним элементом в списке. Затем индекс  $\text{start}$  переопределяем на только что добавленный элемент:  $\text{start} = i$ . Увеличиваем количество элементов в списке  $\text{count}++$ .

Для добавления элемента в конец списка, рассуждения аналогичны, с поправкой на то, что добавляем в конец списка.

## 2) Удаление звена списка из начала и из конца.

Сначала проверяем не пуст ли список. Если он пуст, то бросаем исключение. Если нет, то в переменную  $\text{elem}$  записываем элемент в начале списка  $\text{mas}[\text{start}]$ . Затем переопределяем индекс первого элемента списка, он будет равен индексу элемента следующим за первым:  $\text{newstart} = \text{nextIndex}[\text{start}]$ . Затем в очередь свободных позиций  $\text{freeElem}$ , добавляем освободившуюся после изъятия первого элемента свободную ячейку  $\text{start}$ . Идентифицируем пустыми соответствующие ячейки массивов  $\text{nextIndex}$  и  $\text{prevIndex}$ :  $\text{nextIndex}[\text{start}] = \text{prevIndex}[\text{start}] = -2$ . Если после извлечения первого элемента, список не пуст, то предыдущего элемента для нового первого элемента не существует:  $\text{prevIndex}[\text{newstart}] = -1$ . Затем индекс  $\text{start}$  переопределяем на  $\text{newstart}$ :  $\text{start} = \text{newstart}$ . Уменьшаем количество элементов в списке  $\text{count}--$ .

Для извлечения элемента из конца списка, рассуждения аналогичны.



## 5. Заключение

В ходе выполнения лабораторной работы была разработана библиотека, реализующая шаблонный класс списка на массивах. Она позволяет при работе со списком на массивах выполнять базовые операции извлечения/добавление элементов списка.

Предоставлено описание примера работы со списком на массивах в разделе «Руководство пользователя».

Также разработаны и доведены до успешного выполнения тесты, проверяющие корректность методов класса TAList.

## 6. Литература

1. Васильев А.Н. Самоучитель С++ с примерами и задачами. -СПб.: Наука и Техника, 2016. -480с.
2. Т. А. Павловская С/С++ Программирование на языке высокого уровня. - СПб.:Питер, 2011. - 461 с.
- 3.Крапенко С. Н. и др. Методы объектно-ориентированного программирования. <http://e-learning.unn.ru/course/view.php?id=251>.
- 4.Страуструп. Б. Курс «Язык программирование С++ для профессионалов» <http://www.intuit.ru/studies/courses/98/98/info>
- 5.Гергель В.П. Методические материалы по курсу “Методы программирования 2”:  
[<http://www.itmm.unn.ru/files/2018/10/Primer-1.1.-Struktury-hraneniya-mnozhestva.pdf>], 2015.
6. <https://rstdn.org/article/alg/list.xml>