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 данных: список»**

**Выполнил:** студент группы 381706-1

Голубева А. С.

\_\_\_\_\_\_\_\_\_\_\_ Подпись

**Научный руководитель:**

аспирант каф. МОСТ ИИТММ

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Лебедев И.Г.

Нижний Новгород

2018.

Содержание

[1. Введение 3](#_Toc388124)

[2. Постановка задачи 4](#_Toc388125)

[3. Руководство пользователя 5](#_Toc388126)

[4. Руководство программиста 6](#_Toc388127)

[4.1 Описание структуры программы 6](#_Toc388128)

[4.2 Описание структур данных 6](#_Toc388129)

[4.3 Описание алгоритмов 7](#_Toc388130)

[5. Заключение 9](#_Toc388131)

[6. Литература 10](#_Toc388132)

# **Введение**

Списком называется упорядоченное множество, состоящее из переменного числа элементов, к которым применимы операции включения, исключения. Список, отражающий отношения соседства между элементами, называется линейным.

Длина списка равна числу элементов, содержащихся в списке, список нулевой длины называется пустым списком. Списки представляют собой способ организации структуры данных, при которой элементы некоторого типа образуют цепочку. Для связывания элементов в списке используют систему указателей.

Однонаправленный (односвязный) список – это структура данных, представляющая собой последовательность элементов, в каждом из которых хранится значение и указатель на следующий элемент списка. В последнем элементе указатель на следующий элемент равен NULL.



Рисунок 1. Односвязный список

# **Постановка задачи**

Выполнение работы предполагает решение следующих задач:

1. Реализация класса элемента списка TElem.
2. Реализация класса списка TList.
3. Реализация класса TException для обработки исключений.
4. Реализация тестов и обеспечение их работоспособности.
5. Пример использования и обеспечение его работоспособности.

# **Руководство пользователя**

Рассмотрим пример работы программы. После запуска программы пользователю предлагается положить элементы в начало списка, требуется ввести количество элементов и сами элементы. Затем на экран выводится сам список и предлагается положить элементы в конец списка. Далее демонстрируется работа методов, позволяющих забрать элемент из начала и конца списка.

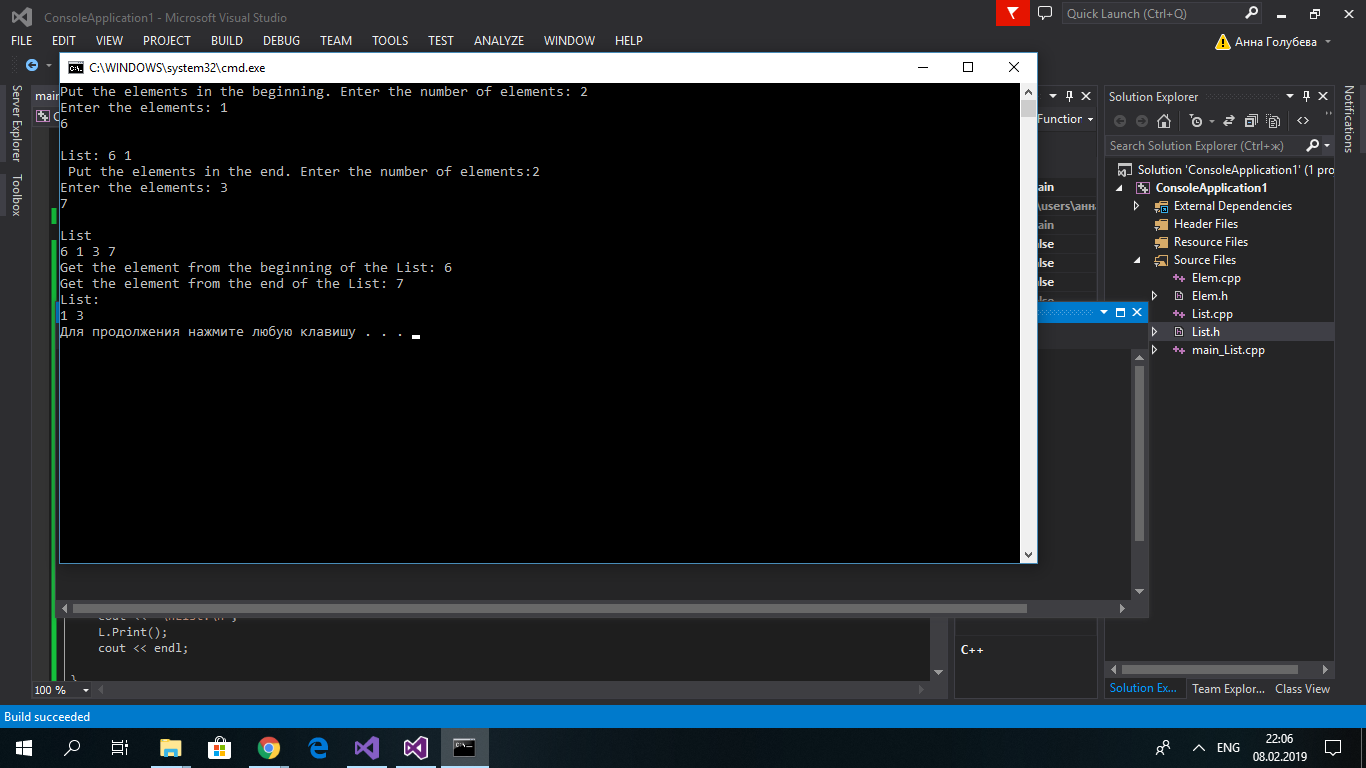


Рисунок 2. Пример работы программы

# **Руководство программиста**

## **Описание структуры программы**

Программа состоит из модулей:

* Модуль ListLib – содержит заголовочные файлы файлы List.h и Elem.h, в которых описаны и реализованы классы TList и TElem, и файлы List.cpp, Elem.cpp
* Модуль List – содержит файл main\_List.cpp, в котором реализован пример использования класса TList;
* Модуль ListTest – содержит файл test\_List.cpp, в котором реализованы тесты для классов TElem и Tlist.
* Модуль Exception – содержит заголовочный файл Exception.h, в котором реализован класс исключений, и файл Exception.cpp.

## **Описание структур данных**

**Класс TElem**

Класс TElem является шаблонным.

**public:**

Поля:

T data; - переменная, хранящая элемент списка

TElem<T>\* next; - указатель на следующий элемент списка

Методы:

TElem(T \_data = 0, TElem <T>\* \_next = 0) - конструктор.

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

virtual ~TElem() - деструктор.

TElem\* GetNext() - метод, возвращающий указатель на следующий элемент.

T GetData() - метод, возвращающий значение элемента.

void SetData(T \_data) - метод, позволяющий установить значение элемента.

void SetNext(TElem <T>\* \_next) – метод, позволяющий установить указатель на следующий элемент списка.

**Класс TList**

Класс TList списка является шаблонным.

**public:**

Поля:

TElem <T>\* begin – указатель на начало списка.

int size – число элементов в списке.

Методы:

TList() - конструктор .

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

virtual ~TList() - деструктор.

void Print() – печать списка на экран.

void Put(int n, T A) - положить элемент на позицию n.

T Get (int n) - взять элемент из списка с номером n.

void PutBegin(T A) – метод, позволяющий положить элемент в начало списка.

void PutEnd(T A) – метод, позволяющий положить элемент в конец списка.

T GetBegin() –метод, позволяющий взять элемент с удалением из начала.

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

bool IsEmpty() - проверка списка на пустоту.

bool IsFull()- проверка полон ли список.

## **Описание алгоритмов**

*Добавление элемента в начало списка.*

При добавлении элемента в начало списка мы создаем указатель на объект класса TElem, далее выделяем память под объект этого класса и создаем элемент списка. Указатель на начало списка переопределяем на созданный элемент списка. Алгоритм работает за линейное время, сложность O(n).

*Удаление элемента из начала списка.*

При удалении элемента из начала списка (если он не пуст) создаем указательна объект класса TElem, которому присваиваем значение текущего начала списка. Создаем временную переменную*,* в которую записываем значение первого элемента. Начало списка устанавливаем на следующий за удаляемым элемент. Удаляем указатель, указывающий на старое начало списка. Алгоритм работает за линейное время, сложность O(n).

*Добавление элемента в конец списка.*

Проверяем есть ли элементы в списке. Если есть, то создаем указатель на объект класса TElem, в него записываем значение начала списка. В цикле ищем последний элемент*.* Затем выделяем память под новый элемент списка и создаем его. Устанавливаем для найденного последнего элемента указатель на созданный элемент. Если в списке нет элементов, то создаем элемент и указателю на начало списка присваиваем значение, указывающее на созданный элемент.

*Удаление элемента из конца списка.*

Проверяем список на пустоту, если список не пуст проверяем больше ли одного элемента в списке. Если указатель на следующий за первым элемент равен 0, то возвращаем первый элемент и обнуляем указатель на начало списка. В случае, когда элементов больше одного, создаем указатель \*a на объект класса TElem. Ищем в цикле предпоследний элемент списка. Создаем указатель \*bна объект класса TElem. В него записываем указатель на последний элемент списка. Получаем данные из этого элемента. Удаляем указатель \*b. Для \*aустанавливаем указатель, на следующий элемент 0.

# **Заключение**

В результате лабораторной работы был реализована структура хранения данных – односвязный список с использованием указателей.

Успешно выполнены тесты, проверяющие работоспособность методов классов TList и TElem. Также был приведен пример работы со списком.

# **Литература**

* 1. Национальный открытый университет «Интуит». Лекция 30: Динамические структуры данных: однонаправленные и двунаправленные списки

URL: <https://www.intuit.ru/studies/courses/648/504/lecture/11456>

(Дата обращения: 06.02.2019)

* 1. Лабораторный практикум. Составители: Барышева И.В., Мееров И.Б., Сысоев А.В., Шестакова Н.В. Под редакцией Гергеля В.П. Учебно-методическое пособие. – Нижний Новгород: Нижегородский госуниверситет, 2017. – 105с.

URL: <http://www.unn.ru/books/met_files/Pract_ADS.pdf>

(Дата обращения: 06.02.2019)