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**

**ЛАБОРАТОРНАЯ РАБОТА №6**

**«Список на массивах»**

**Выполнил:** студент группы 381903-3

Стеценко Данил Михайлович

**Руководитель:**

Ассистент кафедры МОСТ

Лебедев Илья Геннадьевич

Нижний Новгород

2020

**Содержание**

[1.Введение 3](#_30j0zll)

[2. Цели и задачи 5](#_1fob9te)

[2.1. Используемые инструменты 5](#_3znysh7)

[3. Руководство пользователя 7](#_2et92p0)

[4. Руководство программиста 13](#_tyjcwt)

[4.1. Описание структуры программы 13](#_3dy6vkm)

[4.2. Описание функций и процедур, их алгоритмов 13](#_1t3h5sf)

[5. Эксперименты 15](#_4d34og8)

[6. Заключение 16](#_2s8eyo1)

[7. Литература 17](#_17dp8vu)

[8. Приложения 18](#_3rdcrjn)

[8.1. Приложение 1:Класс](#_lnxbz9) Arraylist [18](#_lnxbz9)

[8.2. Приложение 2:Код программы тестирования и экспериментов](#_44sinio) 20

[8.3. Приложение 3:Тесты для класса](#_2jxsxqh) 22

# 1.Введение

Списки являются чрезвычайно гибкой структурой, так как их легко сделать большими или меньшими, и их элементы доступны для вставки или удаления в любой позиции списка. Списки также можно объединять или разбивать на меньшие списки. Списки регулярно используются в приложениях, например в программах информационного поиска, трансляторах программных языков или при моделировании различных процессов.

В математике список представляет собой последовательность элементов определенного типа ( elementtype ). Представим список в виде последовательности элементов, разделенных запятыми: a1, а2, ..., аn, где n≥ 0 и всё ai имеют тип elementtype. Количество элементов n - длина списка. Если n ≥ 1, то а1 называется первым элементом, а аn—последним элементом списка. В случае n = 0 имеем пустой список, который не содержит элементов.

Важное свойство списка заключается в том, что его элементы можно линейно упорядочить в соответствии с их позицией в списке. Говорим, что элемент ai предшествует ai+1 для і = 1, 2, ..., n — 1 и ai следует за ai-1 для і =2, 3, .., n. Также будем говорить, что элемент аi имеет позицию і.

Реализация списков посредством массивов

При реализации списков с помощью массивов элементы списка располагаются в смежных ячейках массива. Это представление позволяет легко просматривать содержимое списка и вставлять новые элементы в его конец. Но вставка нового элемента в середину списка требует перемещения всех, последующих элементов на одну позицию к концу массива, чтобы освободить место для нового элемента. Удаление элемента также требует перемещения элементов, чтобы закрыть освободившуюся ячейку.

# 2. Цели и задачи

В рамках лабораторной работы ставится задача эффективной реализации структуры данных – линейный односвязный список на массивах и выполнение основных операций над ним:

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

В процессе выполнения лабораторной работы требуется использовать систему контроля версий [Git](https://git-scm.com/book/ru/v2) и фрэймворк для разработки автоматических тестов [Google Test](https://github.com/google/googletest).

Выполнение работы предполагает решение следующих задач:

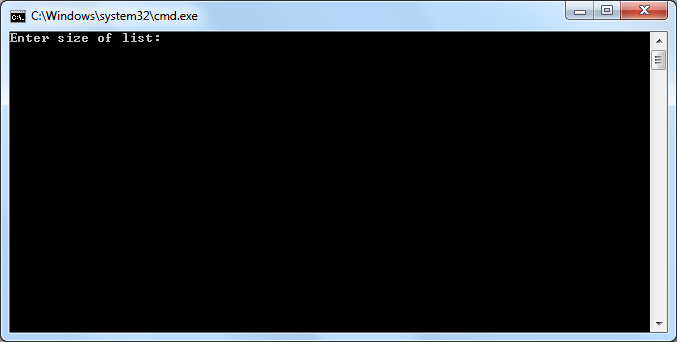
1. Реализация методов шаблонного класса Arraylist согласно заданному интерфейсу.
2. Обеспечение работоспособности тестов и примера использования.
3. Реализация заготовок тестов, покрывающих все методы Arraylist.
4. Модификация примера использования в тестовое приложение, позволяющее задавать списки и осуществлять основные операции над ними.

## 2.1. Используемые инструменты

* Система контроля версий [Git](https://git-scm.com/book/ru/v2). Рекомендуется использовать один из следующих клиентов на выбор студента:
  + [Git](https://git-scm.com/downloads)
  + [GitHub Desktop](https://desktop.github.com/)
* Фреймворк для написания автоматических тестов [Google Test](https://github.com/google/googletest).
* Среда разработки Microsoft Visual Studio (2008 или старше).
* Опционально. Утилита [CMake](http://www.cmake.org/) для генерации проектов по сборке исходных кодов. Может быть использована для генерации решения для среды разработки, отличной от Microsoft Visual Studio 2008 или 2010.

# 3. Руководство пользователя

Запускаем программу из файла sample\_tarrlist.cpp (Рис.1):



# 

# 

# 

# 

# 

# 4. Руководство программиста

Разработка системы вычисления проводились в среде “Microsoft Visual Studio 2010”.

В данной работе будет использовано 4 класса:

* Класс «Список» (Arraylist), реализованный с использованием массивов и использующий класс очередь.

## 4.1. Описание структуры программы

Модульная структура программы:

1. Arraylist.h– модуль с классом Arraylist, в котором определен интерфейс шаблонного класса Список и реализация его методов.
2. test\_main.cpp, test\_tarrlist.cpp – модуль с функциями тестирования для созданных классов. Содержит 24 теста для класса Arraylist.

## 4.2. Описание функций и процедур, их алгоритмов

Рассмотрим реализацию методов шаблонного класса TArrList:

template <class ArrListType>

class TArrList

1. TArrList(int \_size = 10) – конструктор класса, принимающий размер списка. По умолчанию создается список размера 10.

2. TArrList(TArrList<ArrListType> &List) - конструктор копирования.

Принимает ссылку на объект класса TArrList.

3. ~TArrList() - деструктор. Освобождает выделенную память.

4. void Put(int n, ArrListType elem) - метод, позволяющий добавить новый элемент в список на определенную позицию.

5. void PutStart(ArrListType elem) - метод, позволяющий добавить новый элемент в начало списка.

6. void PutEnd(ArrListType elem) - метод, позволяющий добавить новый элемент в конец списка.

7. ArrListType Get(int n) - метод изъятия элемента на определенной позиции из списка с удалением.

8. ArrListType GetStart() - метод изъятия элемента из начала списка с удалением.

9. ArrListType GetEnd() - метод изъятия элемента с конца списка с удалением.

10. bool IsFull() - метод проверки списка на полноту.

11. bool IsEmpty() - метод проверки списка на пустоту.

12. void Print() - метод отображения текущих элементов списка.

# 5. Эксперименты

В качестве примера рассмотрим операцию добавления элемента в конец и в начало списка для класса Arraylist.

Теоретическая сложность выполнения алгоритмов O(1).

Мы провели измерение скорости добавления элемента в конец и в начало списка при разном количестве элементов: 10, 100, …, 1000000. Ниже вы можете увидеть таблицу зависимости времени выполнения операции от количества элементов списка. (Таблица.1)

По приведенным данным можно сделать вывод, что практическая сложность выполнения алгоритма равна теоретической.

|  |  |  |
| --- | --- | --- |
| Кол-во элементов | метод PutEnd() | метод PutStart() |
| 10 | 0 | 0 |
| 100 | 0 | 0 |
| 1000 | 0 | 0 |
| 10000 | 0 | 0 |
| 100000 | 0 | 0 |

Таблица.1. Зависимость времени выполнения операций от размера списка.

# 6. Заключение

В результате лабораторной работы была разработана структура данных – односвязный линейный список на массивах, а также освоены такие инструменты разработки программного обеспечения, как система контроля версий [Git](https://git-scm.com/book/ru/v2) и фрэймворк для разработки автоматических тестов [Google Test](https://github.com/google/googletest).

Созданный шаблонный класс был протестированы с использованием Google Tests, а также были проведены эксперименты для сравнения теоретической и практической сложности выполнения операций на методе класса.

# 7. Литература

1. Лабораторный практикум: Учебно-методическое пособие / Мееров И.Б. [и др.] – Нижний Новгород: Нижегородский госуниверситет , 2017. – 105с.
2. Тестирование с использованием Google Test

([http://neerc.ifmo.ru/wiki/index.php?title=Тестирование\_с\_использованием\_Google\_Test#.D0.A4.D1.83.D0.BD.D0.BA.D1.86.D0.B8.D1.8F\_main.28.29](http://neerc.ifmo.ru/wiki/index.php?title=%D0%A2%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D1%81_%D0%B8%D1%81%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5%D0%BC_Google_Test#.D0.A4.D1.83.D0.BD.D0.BA.D1.86.D0.B8.D1.8F_main.28.29))

1. Касперски К. Техника оптимизации программ. Эффективное использование памяти. — СПб.: БХВ-Петербург, 2003. — 464 с.: ил.
2. <http://bookwu.net/book_algoritmy-i-struktury-dannyh_1245/11_2.1.-abstraktnyj-tip-dannyh-spisokquot>
3. Д. Кнут. Искусство программирования. (3-е издание) Т.1.

# 8. Приложения

## 8.1. Приложение 1:Класс Arraylist

|  |
| --- |
| **tarrlist.h** |
| #ifndef \_ARRAY\_LIST\_  #define \_ARRAY\_LIST\_  #include <iostream>  using namespace std;  template <class T1>  class TArrayListIterator;  template <class T>  class TArrayList  {  protected:  T\* data;  int\* links;  int size;  int count;  int root;  public:  TArrayList(int \_size = 1);  TArrayList(TArrayList<T>& \_v);  ~TArrayList();  TArrayList<T>& operator =(TArrayList<T>& \_v);  void InsFirst(T d);  void InsLast(T d);  void Ins(TArrayListIterator<T>& e, T d);  bool IsFull(void) const;  bool IsEmpty(void) const;  TArrayListIterator<T> Begin();  TArrayListIterator<T> End();  void DelFirst();  void DelLast();  T GetFirst();  T GetLast();  void Del(TArrayListIterator<T>& e);  int GetCount();  template <class T1>  friend ostream& operator<< (ostream& ostr, const TArrayList<T1>& A);  template <class T1>  friend istream& operator >> (istream& istr, TArrayList<T1>& A);  template <class T1>  friend class TArrayListIterator;  };  template <class T>  class TArrayListIterator  {  protected:  TArrayList<T>& list;  int index;  public:  TArrayListIterator(TArrayList<T>& \_list, int \_index = -2);  TArrayListIterator(TArrayListIterator<T>& \_v);  ~TArrayListIterator();  bool IsGoNext();  void GoNext();  bool operator == (const TArrayListIterator<T>& \_v);  TArrayListIterator<T>& operator ++ ();  TArrayListIterator<T>& operator =(const TArrayListIterator<T>& \_v);  T GetData();  T GetIndex();  };  template <class T1>  ostream& operator<< (ostream& ostr, const TArrayList<T1>& A)  {  int i = A.root;  while (A.links[i] != -1)  {  ostr << A.data[i];  i = A.links[i];  }  return ostr;  }  template <class T1>  istream& operator >> (istream& istr, TArrayList<T1>& A)  {  int count;  istr >> count;  for (int i = 0; i < count; i++)  {  T1 d;  istr >> d;  A.InsLast(d);  }  return istr;  }  template<class T>  inline TArrayList<T>::TArrayList(int \_size)  {  if (\_size <= 0)  {  throw logic\_error("Error");  }  size = \_size;  data = new T[size];  links = new int[size];  count = 0;  for (int i = 0; i < size; i++)  {  links[i] = -2;  }  root = -1;  }  template <class T>  TArrayList<T>::TArrayList(TArrayList<T>& \_v)  {  count = \_v.count;  size = \_v.size;  root = \_v.root;  data = new T[size];  links = new int[size];  for (int i = 0; i < size; i++)  {  links[i] = \_v.links[i];  data[i] = \_v.data[i];  }  }  template <class T>  TArrayList<T>::~TArrayList()  {  if (data != 0)  {  delete[] data;  delete[] links;  data = 0;  links = 0;  count = 0;  size = 0;  root = -1;  }  }  template <class T>  TArrayList<T>& TArrayList<T>::operator =(TArrayList<T>& \_v)  {  if (this == &\_v)  return \*this;  if (size != \_v.size)  {  delete[] data;  delete[] links;  data = new T[\_v.size];  links = new int[\_v.size];  }  count = \_v.count;  size = \_v.size;  root = \_v.root;  for (int i = 0; i < size; i++)  {  links[i] = \_v.links[i];  data[i] = \_v.data[i];  }  return \*this;  }  template<class T>  inline void TArrayList<T>::InsFirst(T d)  {  if (this->IsFull())  {  throw logic\_error("Error");  }  int i = 0;  for (i = 0; i < size; i++)  {  if (links[i] == -2)  {  break;  }  }  data[i] = d;  links[i] = root;  root = i;  count++;  }  template<class T>  inline void TArrayList<T>::InsLast(T d)  {  if (this->IsFull())  {  throw logic\_error("Error");  }  if (this->IsEmpty())  {  root = 0;  data[0] = d;  links[0] = -1;  }  else  {  int i = 0;  for (i = 0; i < size; i++)  {  if (links[i] == -2)  {  break;  }  }  int end = root;  while (links[end] != -1)  {  end = links[end];  }  data[i] = d;  links[i] = -1;  links[end] = i;  }  count++;  }  template<class T>  inline void TArrayList<T>::Ins(TArrayListIterator<T>& e, T d)  {  if (this->IsFull())  {  throw logic\_error("Error");  }  if (this->IsEmpty())  {  root = 0;  data[0] = d;  links[0] = -1;  }  else  {  int temp = e.GetIndex();  if (links[temp] == -2)  {  int end = root;  while (links[end] != -1)  {  end = links[end];  }  data[temp] = d;  links[temp] = -1;  links[end] = temp;  }  }  count++;  }  template<class T>  inline bool TArrayList<T>::IsFull(void) const  {  return count >= size;  }  template<class T>  inline bool TArrayList<T>::IsEmpty(void) const  {  return count == 0;  }  template<class T>  inline void TArrayList<T>::DelFirst()  {  if (this->IsEmpty())  {  throw logic\_error("Error");  }  int i = root;  root = links[root];  links[i] = -2;  count--;  }  template<class T>  inline void TArrayList<T>::DelLast()  {  if (this->IsEmpty())  {  throw logic\_error("Error");  }  if (links[root] == -1)  {  links[root] = -2;  root = -1;  }  else  {  int pEnd = root;  int end = links[root];  while (links[end] != -1)  {  pEnd = end;  end = links[end];  }  links[pEnd] = -1;  links[end] = -2;  }  count--;  }  template<class T>  inline T TArrayList<T>::GetFirst()  {  if (this->IsEmpty())  {  throw logic\_error("Error");  }  return data[root];  }  template<class T>  inline T TArrayList<T>::GetLast()  {  if (this->IsEmpty())  {  throw logic\_error("Error");  }  int end = root;  while (links[end] != -1)  {  end = links[end];  }  return data[end];  }  template<class T>  inline void TArrayList<T>::Del(TArrayListIterator<T>& e)  {  if (this->IsEmpty())  {  throw logic\_error("Error");  }  if (links[root] == -1)  {  links[root] = -2;  root = -1;  }  else  {  int pEnd = e.GetIndex();  int end = links[root];  while (links[end] != -1)  {  pEnd = end;  end = links[end];  }  links[pEnd] = -1;  links[end] = -2;  }  count--;  }  template<class T>  inline int TArrayList<T>::GetCount()  {  return count;  }  #endif  template<class T>  inline TArrayListIterator<T> TArrayList<T>::Begin()  {  return TArrayListIterator<T>(\*this, this->root);  }  template<class T>  inline TArrayListIterator<T> TArrayList<T>::End()  {  return TArrayListIterator<T>(\*this, -1);  }  template<class T>  inline TArrayListIterator<T>::TArrayListIterator(TArrayList<T>& \_list, int \_index) : list(\_list)  {  this->index = \_index;  }  template<class T>  inline TArrayListIterator<T>::TArrayListIterator(TArrayListIterator<T>& \_v) : list(\_v.list), index(\_v.index)  {  }  template<class T>  inline TArrayListIterator<T>::~TArrayListIterator()  {  index = -2;  }  template<class T>  inline bool TArrayListIterator<T>::IsGoNext()  {  return (index >= 0);  }  template<class T>  inline void TArrayListIterator<T>::GoNext()  {  if (IsGoNext())  {  index = list.links[index];  }  }  template<class T>  inline bool TArrayListIterator<T>::operator==(const TArrayListIterator<T>& \_v)  {  return (index == \_v.index && \*list == \*(\_v.list));  }  template<class T>  inline TArrayListIterator<T>& TArrayListIterator<T>::operator++()  {  GoNext();  return \*this;  }  template<class T>  inline TArrayListIterator<T>& TArrayListIterator<T>::operator=(const TArrayListIterator<T>& \_v)  {  list = \_v.list;  index = \_v.index;  }  template<class T>  inline T TArrayListIterator<T>::GetData()  {  if (index < 0)  {  throw logic\_error("Error");  }  return list.data[index];  }  template<class T>  inline T TArrayListIterator<T>::GetIndex()  {  if (index < 0)  {  throw logic\_error("Error");  }  return index;  } |

## 8.2. Приложение 2:Тесты для класса

|  |
| --- |
| **test\_main.cpp** |
| #include <gtest.h>  int main(int argc, char \*\*argv)  {  ::testing::InitGoogleTest(&argc, argv);  return RUN\_ALL\_TESTS();  } |

|  |
| --- |
| **test\_TArraylist.cpp** |
| #include "../gtest/gtest.h"  #include "../Arraylist/Arraylist.h"  TEST(TArrayList, can\_create\_Arraylist)  {  ASSERT\_NO\_THROW(TArrayList<int> temp(10));  }  TEST(TArrayList, can\_create\_Arraylist\_Iterator)  {  TArrayList<int> temp(10);  ASSERT\_NO\_THROW(TArrayListIterator<int> i(temp));  }  TEST(TArrayList, can\_copy\_Arraylist)  {  TArrayList<int> temp(10);  ASSERT\_NO\_THROW(TArrayList<int> temp2(temp));  }  TEST(TArrayList, can\_copy\_Arraylist\_Iterator)  {  TArrayList<int> temp(10);  TArrayListIterator<int> i(temp);  ASSERT\_NO\_THROW(TArrayListIterator<int> j = i);  }  TEST(TArrayList, can\_insert\_elem\_first)  {  TArrayList<int> temp(10);  ASSERT\_NO\_THROW(temp.InsFirst(1));  }  TEST(TArrayList, cant\_insert\_elem\_isFull)  {  TArrayList<int> temp(1);  temp.InsFirst(1);  ASSERT\_ANY\_THROW(temp.InsFirst(2));  }  TEST(TArrayList, can\_insert\_elem\_last)  {  TArrayList<int> temp(10);  ASSERT\_NO\_THROW(temp.InsLast(1));  }  TEST(TArrayList, can\_insert\_through\_iterator)  {  TArrayList<char> temp(10);  temp.InsFirst('a');  temp.InsLast('b');  temp.InsFirst('c');  TArrayListIterator<char> i(temp, 1);  ASSERT\_NO\_THROW(temp.Ins(i, 'd'));  }  TEST(TArrayList, check\_isFull)  {  TArrayList<char> temp(1);  temp.InsFirst('a');  EXPECT\_EQ(temp.IsFull(), true);  }  TEST(TArrayList, check\_isEmpty)  {  TArrayList<char> temp(1);  temp.InsFirst('a');  EXPECT\_EQ(temp.IsEmpty(), false);  }  TEST(TArrayList, can\_del\_first)  {  TArrayList<char> temp(1);  temp.InsFirst('a');  ASSERT\_NO\_THROW(temp.DelFirst());  }  TEST(TArrayList, can\_del\_last)  {  TArrayList<char> temp(1);  temp.InsFirst('a');  ASSERT\_NO\_THROW(temp.DelLast());  }  TEST(TArrayList, can\_Get\_first)  {  TArrayList<char> temp(1);  temp.InsFirst('a');  EXPECT\_EQ(temp.GetFirst(), 'a');  }  TEST(TArrayList, can\_Get\_last)  {  TArrayList<char> temp(1);  temp.InsFirst('a');  EXPECT\_EQ(temp.GetLast(), 'a');  }  TEST(TArrayList, can\_del\_iterator)  {  TArrayList<char> temp(1);  temp.InsFirst('a');  TArrayListIterator<char> i(temp, 1);  ASSERT\_NO\_THROW(temp.Del(i));  }  TEST(TArrayList, can\_get\_count)  {  TArrayList<char> temp(1);  temp.InsFirst('a');  EXPECT\_EQ(temp.GetCount(), 1);  }  TEST(TArrayList, can\_GetData\_iterator)  {  TArrayList<char> temp(1);  temp.InsFirst('a');  TArrayListIterator<char> i(temp, 1);  ASSERT\_NO\_THROW(i.GetData());  }  TEST(TArrayList, can\_GetIndex\_iterator)  {  TArrayList<char> temp(1);  temp.InsFirst('a');  TArrayListIterator<char> i(temp, 1);  ASSERT\_NO\_THROW(i.GetIndex());  } |