|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  |  |  | | МИНОБРНАУКИ РОССИИ | | | | Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | |   Институт информационных технологий |
| Кафедра вычислительной техники |
|  |

|  |  |
| --- | --- |
| **ОТЧЕТ ПО ЛАБОРАТОРНЫМ РАБОТАМ** | |
| **по дисциплине** | |
| «Структуры и алгоритмы обработки данных**»** | |
|  | |
| Выполнил студент группы ИКБО-01-17 | Загорулько К.А. |
| Принял | Асадова Ю.С. |

|  |  |  |
| --- | --- | --- |
| Лабораторные работы выполнены | «\_\_»\_\_\_\_\_\_\_201\_\_ г. | *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_* |
|  |  |  |
| «Зачтено» | «\_\_»\_\_\_\_\_\_\_201\_\_ г. | *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_* |

Москва 2018

|  |  |  |
| --- | --- | --- |
|  |  |  |
| МИНОБРНАУКИ РОССИИ | | |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | |

Институт информационных технологий

Кафедра вычислительной техники

Выполнено \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/К.А. Загорулько/

Зачтено \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Ю.С. Асадова/

**Задание на лабораторную работу №5**

Дисциплина: **«Структуры и алгоритмы обработки данных»**

Студент Загорулько Константин Александрович Шифр 17и0284

Группа ИКБО-01-17

**1. Тема**: «ПЕРЕГРУЗКА ФУНКЦИЙ».

**2. Срок сдачи студентом законченной работы:** \_\_\_\_/\_\_\_\_\_/2018г.

**3. Исходные данные:** Введённая матрица заданного размера и типа данных

**4. Задание:**

Упорядочить столбцы по убыванию первых элементов столбцов

**5. Содержание отчета:**

* титульный лист;
* задание;
* оглавление;
* введение;
* основные разделы отчета;
* заключение;
* использованные источники;

Руководитель работы Ю.С. Асадова \_\_\_\_\_\_\_\_ «\_\_\_» \_\_\_\_\_\_\_\_\_ 2018г.

подпись

Задание принял к исполнению К.А. Загорулько \_\_\_\_\_\_\_\_ «\_\_\_» \_\_\_\_\_\_\_\_\_ 2018г.

подпись

### Содержание

Введение................................................................................................ 4

1. Цель работы...................................................................................... 5

2. Решение............................................................................................. 5

3. Код программы................................................................................. 5

4. Примеры работы............................................................................... 8

Заключение........................................................................................... 10

Использованные источники................................................................ 11

### 

### Введение

В задании требуется создать двусвязный список групп факультетов. Каждая группа должна представляет собой односвязный список студентов. Односвязный список было решено реализовать собственными усилиями. Как двусвязный используется стандартный контейнер STL.

### Цель работы

Ознакомление со структурой данных – связный список, приобретение умений реализации типовых операций со связными списками, приобретение навыков использования связных списков при решении задач, требующих оптимизации ресурсов времени и памяти.

### 2. Задание

Создайте двусвязный список групп факультета. Каждая группа представляет собой односвязный список студентов.

Исходные данные:

Вариант №8

### 3. Код программы

|  |
| --- |
| #ifndef SINGLELIST\_H  #define SINGLELIST\_H  #include <assert.h>  #include <stddef.h>  #include <iostream>  #include <list>  #include <algorithm>  **using** **namespace** std;  **template** <**typename** T>  **struct** Cell {  T value;  Cell\* next;  };  **template** <**typename** T>  **class** **SingleList**  {  **private:**  Cell<T>\* begin;  Cell<T>\* end;  **int** len;  **public:**  SingleList();  **void** **push**(T value);  T **pop**();  **void** **insert**(**unsigned** **int** pos, T value);  string **get\_string**(string splitter = " ");  **void** **sort**();  **static** **void** **test**();  };  #endif // SINGLELIST\_H  #include "singlelist.h"  **template** <**typename** T>  SingleList<T>::SingleList()  {  len = **0**;  }  **template** <**typename** T>  T SingleList<T>::pop()  {  assert(len > **0**);  Cell<T>\* temp = begin;  Cell<T>\* temp2;  **while**(temp->next != NULL) {  temp2 = temp;  temp = temp->next;  **if**(temp2 == temp) {  temp->next = NULL;  }  }  len--;  **if**(len > **0**) {  temp2->next = NULL;  }  end = temp2;  **return** temp->value;  }  **template**<**typename** T>  **void** SingleList<T>::insert(**unsigned** **int** pos, T value)  {  Cell<T> \*temp;  Cell<T> \*newCell;  assert((**unsigned**)**this**->len >= pos);  **if**((**unsigned**)**this**->len == pos) {  push(value);  } **else** {  newCell = **new** Cell<T>;  newCell->value = value;  **if**(pos == **0**) {  newCell->next = begin;  begin = newCell;  } **else** {  temp = begin;  **for**(**unsigned** **int** i = **0**; i < pos-**1**; i++) {  temp = temp->next;  }  newCell->next = temp->next;  temp->next = newCell;  }  **this**->len++;  }  }  **template** <**typename** T>  **void** SingleList<T>::push(T value)  {  Cell<T>\* temp = **new** Cell<T>;  temp->next = NULL;  temp->value = value;  **if**(len == **0**) {  begin = temp;  end = temp;  }  end->next = temp;  end = temp;  len++;  }  **template** <**typename** T>  string SingleList<T>::get\_string(string splitter)  {  string result;  Cell<T>\* temp = begin;  result = "";  **while**(temp != end) {  result+= temp->value;  result+= splitter;  temp = temp->next;  }  result+=temp->value;  **return** result;  }  **template**<**typename** T>  **void** SingleList<T>::sort()  {  list<T> tmp;  **while**(**this**->len > **0**) {  T temp = **this**->pop();  tmp.push\_back(temp);  }  tmp.sort();  **while**(!tmp.empty()) {  **this**->push(tmp.front());  tmp.pop\_front();  }  }  **template**<**typename** T>  **void** SingleList<T>::test()  {  SingleList<string> students;  students.push("Ivan Petrov");  students.push("Vasilij Sidorov");  students.push("Andey Panov");  students.push("Kirill Emeljanov");  students.push("Avror Vorontsov");  students.push("Michael Afanas'ev");  students.push("Arsen Socolov");  students.push("Igor Isakov");  students.push("Stepan Vishnyakov");  students.push("Arsenij Tyrov");  students.push("Leonid Volokitin");  students.push("Afanasij Efimov");  cout << students.get\_string("**\n**");  cout << endl << endl;  students.sort();  cout << students.get\_string("**\n**");  }  **template** **class** **SingleList**<**int**>;  **template** **class** **SingleList**<string>;  #ifndef FACULTY\_H  #define FACULTY\_H  #include <list>  #include <string>  #include <algorithm>  #include <stddef.h>  #include <cstring>  #include "singlelist.h"  **class** **Faculty**  {  **private:**  list<pair<string, SingleList<string>>> data;  list<pair<string, SingleList<string>>>::iterator find(string group);  **public:**  Faculty();  **void** **push\_back**(string group, string name);  string **pop\_back**(string group);  string **get\_string**(string splitter = " ");  **bool** **insert\_afeter**(string after, string name);  **void** **sort**();  **static** **void** **test**();  };  #endif // FACULTY\_H  #include "faculty.h"  list<pair<string, SingleList<string>>>::iterator Faculty::find(string group)  {  **for**(list<pair<string, SingleList<string>>>::iterator it = data.begin();  it != data.end(); it++) {  **if**(it->first == group) {  **return** it;  }  }  **return** data.end();  }  Faculty::Faculty()  {  }  **void** Faculty::push\_back(string group, string name)  {  list<pair<string, SingleList<string>>>::iterator pos;  pos = **this**->find(group);  **if**(pos != data.end()) {  pos->second.push(name);  } **else** {  SingleList<string> \* list = **new** SingleList<string>;  list->push(name);  data.push\_back(make\_pair(group, \*list));  }  }  string Faculty::get\_string(string splitter)  {  string result;  result = "";  **for**(list<pair<string, SingleList<string>>>::iterator it = data.begin(); it != data.end(); it++){  result+=it->first + splitter + it->second.get\_string(splitter) + "**\n**";  }  **return** result;  }  **void** Faculty::sort()  {  data.sort([](**const** pair<string, SingleList<string>> &a,  **const** pair<string, SingleList<string>> &b) {  **return** a.first < b.first;  });  **for**(**auto** i = data.begin(); i != data.end(); ++i) {  i->second.sort();  }  }  **void** Faculty::test()  {  Faculty it;  it.push\_back("ikbo-01-17", "Zagorulko Konstanin");  it.push\_back("ikbo-03-17", "Vasilij Sidorov");  it.push\_back("ikbo-02-17", "Ivan Petrov");  it.push\_back("ikbo-01-17", "Andey Panov");  it.push\_back("ikbo-01-17", "Kirill Emeljanov");  it.push\_back("ivbo-01-17", "Avror Vorontsov");  it.push\_back("ivbo-02-17", "Michael Afanas'ev");  it.push\_back("ivbo-02-17", "Arsen Socolov");  it.push\_back("ivbo-01-17", "Igor Isakov");  it.push\_back("ikbo-01-17", "Stepan Vishnyakov");  it.push\_back("ikbo-01-17", "Arsenij Tyrov");  it.push\_back("ikbo-03-17", "Leonid Volokitin");  it.push\_back("ikbo-01-17", "Afanasij Efimov");  cout << it.get\_string("**\n** ") << endl;  it.sort();  cout << it.get\_string("**\n** ") << endl;  }  #include "find.h" // 1 практическая  #include "findname.h" // 1 практическая  #include "singlelist.h" // 1 лабораторная  #include "faculty.h" // 1 лабораторная  **int** **main**()  {  Faculty::test();  **return** **0**;  } |

### 4. Примеры работы

Примеры работы продемонстрирован на рисунке 1.

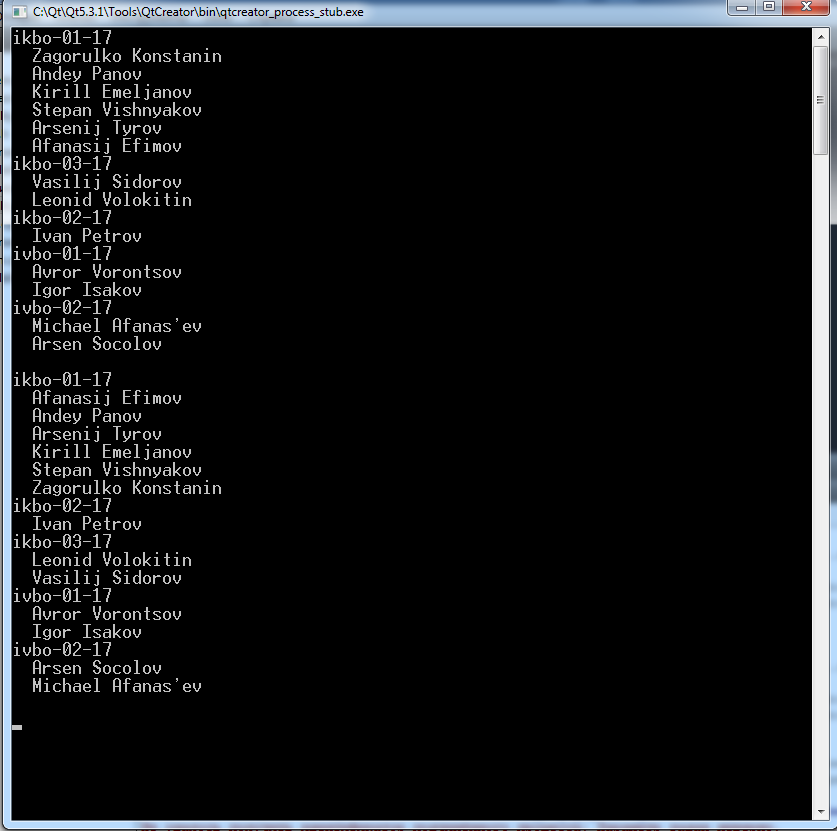


Рис. 1 – скриншот работы программы с типом char \*.

### Заключение

В процессе выполнения данной лабораторной работы были углублены знания в области стандартных контейнерах. Приобретён опыт создания собственного контейнера. Прописано несколько сложных сортировок.

### Использованные источники

1. Алгоритмы и структуры данных. Новая версия для Оберона + CD / Пер. с англ. Ткачев Ф. В. – М.: ДМК Пресс, 2010. – 272 с.: ил
2. Павловская Т.А., Щупак Ю.А. С/С++. Структурное программирование. Практикум. – СПб.: Питер, 2002.
3. Подбельский В.В. Язык С++; Учебное пособие. М.: Финансы и статистика, 1996.
4. Павловская Т.А. С/С++. Программирование на языке высокого уровня. – СПб.: Питер, 2002.
5. Дейтл Х.М., Дейтл П.Дж. Как программировать на С. – М.: Бином, 200