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

«Просматриваемые таблицы»

Выполнил:
студент группы 381706-1
Митягина Дарья Сергеевна
_____ Подпись

Проверил:
ассистент каф. МОСТ ИИТММ
Лебедев Илья Геннадьевич
_____ Подпись

Нижний Новгород

2019

Оглавление

1. Введение

В данной лабораторной работе под таблицей понимается набор элементов, состоящих из ключа и данных. Ключ – уникальный идентификатор элемента таблицы. Поля просматриваемой таблицы являются неупорядоченными, то есть порядок расположения элементов в таблице не важен. Просматриваемая таблица используется в тех ситуациях, когда редко используется поиск элементов.

В данной работе мы рассмотрим таблицы следующего вида: пользовательские данные являются шаблонным классом, а ключи элементов представляют собой некоторую строку.

2. Постановка задачи

Целью данной лабораторной работы является написание библиотеки для работы с просматриваемыми таблицами.

1. Создать класс элементов таблицы;
2. Создать класс для хранения таблиц и работы с ними;
3. Написать пример использования библиотеки;
4. Написать тесты для проверки правильности работы методов классов.

3. Руководство пользователя

Для работы пользователя была разработана небольшая программа, демонстрирующая основные возможности библиотеки. Программа расположена в файле *table.cpp (*модуль *ViewTable).*

Результатом работы программы является создание таблицы и добавление в нее новых элементов, их удаление.

4. Руководство программиста

4.1. Описание структуры программы

Три основных модуля программы:

1. *ViewTableLib* – библиотека для работы со списками. Состоит из файлов *ViewTableElem.h* и *ViewTable.h*, в которых находятся шаблонные классы элемента таблицы *TViewTableElem* и самой таблицы класс *TViewTable* соответственно.
2. *ViewTable*– пример использования.
3. *ViewTableTest* – тесты.

4.2 Описание структур данных

TViewTableElem – шаблонный класс, содержащий следующие поля и методы:

protected:

- *string key* – ключ;
- *T data* – данные;

public:

- *TViewTableElem()* – конструктор по умолчанию.
- *TViewTableElem(string k, T d)* - конструктор инициализатор;
- *TViewTableElem(TViewTableElem<T>& A)* – конструктор копирования;
- *TViewTableElem<T>& operator=(TViewTableElem<T>& A)* – оператор присваивания;
- *string GetKey()* – получить ключ элемента;
- *T GetData()* – получить данные;
- *void SetKey(string k)* – установить ключ в значение *k*;
- *void SetData(T d)* – установить данные в значение *d;*
- *bool operator==(TViewTableElem<T>& A)* – оператор сравнения.
- *bool operator>(TViewTableElem<T>& A)* – оператор сравнения;

- *bool operator< (TViewTableElem<T>& A)* - оператор сравнения;
- *friend ostream& operator<<(ostream& ostr, const TViewTableElem<T1> SE)* – оператор вывода;

T*ViewTable* – шаблонный класс, содержащий следующие поля и методы:

protected:

- *TViewTableElem<T>* mas* – массив элементов таблицы;
- *unsigned size* – размер таблицы;
- *unsigned count* – количество элементов в таблице;

public:

- *TViewTableElem<T> not_found* –означает несуществующее поле;
- *TViewTable (int n = 0)* – конструктор;
- *TViewTable (TViewTable<T>& A)* – конструктор копирования;
- *void Put(string k, T d)* – добавление поля с ключом *k* и данными *d*;
- *void Del(string k)* – удаление элемента с ключом *k*;
- *int GetCount()* – получить количество записей в таблице;
- *TViewTableElem<T>& Search(string k)* – поиск элемента с ключом *k*;
- *T& operator[](string k)* – оператор индексации;
- *~TViewTable ()* – деструктор;
- *friend ostream& operator<<(ostream& ostr, const TViewTable<T1> &A)* – оператор вывода.

4.3 Описание алгоритмов

Далее n - количество записей в таблице

- Поиск элемента:

Производится проверка всех ключей элементов таблицы на равенство с пришедшей в качестве параметра строкой. В случае успеха (элемент с таким ключом найден) возвращается ссылка на этот элемент. Иначе возвращается ссылка на элемент not_found.

Сложность алгоритма: O(n).

- Добавление элемента:

Производится проверка наличия в таблице свободного место. Если место есть, то элемент добавляется в конец таблицы. Иначе, бросается исключение.

Сложность алгоритма: O(1).

- Удаление элемента:

Если в таблице есть элементы, то ищем удаляемый элемент, затем, если искомый элемент найден, на его место ставим последний элемент таблицы. Если элемент не с данным ключом не найден или таблица пустая, то бросается исключение

Сложность алгоритма: O(n).

5. Заключение

В процессе работы над данной лабораторной работой удалось разработать библиотеку для работы с просматриваемыми таблицами.

Кроме того, было достигнуто более глубокое понимание принципов работы с данной структурой, тестирования работы программы с помощью Google Test.

6. Список литературы

1. Павловская Т. А. C/C++ Программирование на языке высокого уровня [Книга]. - СПб : Питер, 2003.
2. Страуструп Бьерн Язык программирования C++ Бином, 2004.
3. Лафоре Роберт Структуры данных и алгоритмы в Java [Книга]. - СПб : Питер, 2013. - 2 : стр. 704.