МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

**АДЫГЕЙСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

Инженерно-физический факультет

Кафедра автоматизированных систем обработки информации и управления

Отчет по практике  
Программная реализация структуры данных *Хеш-таблицы*

2 курс, группа 2ИВТ АСОИУ

Выполнил:  
В. Р. Хайрулин   
«06.06» 2025 г.

Руководитель:  
 С. В. Теплоухов  
«06.06» 2025 г.

Майкоп, 2025 г.

# Введение

## Текстовая формулировка задачи (Вариант 6)

Реализовать консольное приложение для работы с хеш-таблицей. Программа должна обеспечивать операции вставки, поиска и удаления элементов, а также отображение содержимого таблицы.

## Теория метода

Хеш-таблица — это структура данных, представляющая собой массив с необычной адресацией, задаваемой хеш-функцией. Среднее время поиска элемента составляет , а в наихудшем случае — , что зависит от количества коллизий.

Основные свойства хеш-таблицы:

* Хеш-функция преобразует ключ в индекс массива, например, путем деления ключа на размер таблицы и взятия остатка.
* Для разрешения коллизий используется метод связывания, где элементы с одинаковым хешем хранятся в линейных списках.
* Хеш-функция должна быть быстрой и обеспечивать равномерное распределение ключей для минимизации коллизий.

Хеширование полезно для хранения большого диапазона значений в ограниченной памяти с быстрым доступом. Оно широко применяется в базах данных и компиляторах для таблиц идентификаторов. При вставке элемента ключ хешируется, и элемент добавляется в начало соответствующего списка. Поиск и удаление выполняются аналогично через хеш-функцию и линейный поиск в списке. Размер таблицы должен быть достаточным для минимизации коллизий.

# Ход работы

## Выбор средств для разработки

Для реализации хеш-таблицы я выбрал стек технологий, состоящий из:

* **C++** — основной язык разработки, обеспечивающий высокую производительность и гибкость;
* **Шаблоны (templates)** — для создания универсального списка, используемого в хеш-таблице.

Целью было создать консольное приложение с интерактивным интерфейсом для управления хеш-таблицей.

## Код приложения

В основе приложения лежит реализация хеш-таблицы с использованием пользовательского шаблонного списка, разделенного на несколько файлов. Ниже приведены фрагменты кода:

* **Файл list.h:** Определение шаблонного класса list.
* #pragma once  
  #include <iostream>  
  using namespace std;  
    
  template<class T>  
  class list {  
  public:  
   list();  
   void pop\_front();  
   void pop\_back();  
   void removeAt(int index);  
   void clear();  
   void push\_back(T data);  
   void push\_front(T data);  
   void insert(T data, int index);  
   void print();  
   int Get\_size();  
   T& operator [](const int index);  
   bool contains(T data);  
   void remove\_value(T data);  
   ~list();  
    
  private:  
   template<class T>  
   struct Node {  
   Node\* ptr\_next;  
   T data;  
   Node(T data = T(), Node\* ptr\_next = nullptr) {  
   this->data = data;  
   this->ptr\_next = ptr\_next;  
   }  
   };  
   int Size;  
   Node<T>\* head;  
  };
* **Файл list.cpp:** Реализация методов класса list.
* template<class T>  
  void list<T>::push\_front(T data) {  
   Node<T>\* current = new Node<T>(data);  
   current->ptr\_next = head;  
   head = current;  
   Size++;  
  }  
    
  template<class T>  
  void list<T>::print() {  
   Node<T>\* current = head;  
   while (current) {  
   cout << current->data << " ";  
   current = current->ptr\_next;  
   }  
  }  
    
  template<class T>  
  list<T>::~list() {  
   clear();  
  }  
    
  template class list<int>; // Явное указание шаблона
* **Файл Hash\_Table.h:** Определение класса Hash\_Table.
* #pragma once  
  #include "list.h"  
  #include <iostream>  
  using namespace std;  
    
  class Hash\_Table  
  {  
  private:  
   static const int SIZE = 8;  
   list<int> table[SIZE];  
    
   int hash(int key) {  
   return key % SIZE;  
   }  
  public:  
   void insert(int key);  
   void remove(int key);  
   void search(int key);  
   void display();  
  };
* **Файл Hash\_Table.cpp:** Реализация методов класса Hash\_Table.
* void Hash\_Table::insert(int key) {  
   int index = hash(key);  
   if (!table[index].contains(key)) {  
   table[index].push\_front(key);  
   cout << "Элемент " << key << " добавлен.\n";  
   }  
   else {  
   cout << "Элемент уже существует.\n";  
   }  
  }  
    
  void Hash\_Table::display() {  
   for (int i = 0; i < SIZE; i++) {  
   cout << "[" << i << "]: ";  
   table[i].print();  
   cout << endl;  
   }  
  }
* **Файл Source.cpp:** Точка входа с интерактивным меню.
* int main() {  
   setlocale(LC\_ALL, "Russian");  
   Hash\_Table ht;  
   int choice, value;  
    
   do {  
   cout << "\nМеню:\n";  
   cout << "1. Вставить\n2. Удалить\n3. Поиск\n4. Печать\n0. Выход\n";  
   cout << "Выбор: ";  
   cin >> choice;  
    
   switch (choice) {  
   case 1:  
   cout << "Введите значение: ";  
   cin >> value;  
   ht.insert(value);  
   break;  
   case 4:  
   ht.display();  
   break;  
   case 0:  
   cout << "Выход.\n";  
   break;  
   }  
   } while (choice != 0);  
    
   return 0;  
  }

## Ключевые фрагменты кода

**Реализация списка (list):**

template<class T>  
void list<T>::push\_front(T data) {  
 Node<T>\* current = new Node<T>(data);  
 current->ptr\_next = head;  
 head = current;  
 Size++;  
}

**Реализация хеш-таблицы (Hash\_Table):**

void Hash\_Table::insert(int key) {  
 int index = hash(key);  
 if (!table[index].contains(key)) {  
 table[index].push\_front(key);  
 cout << "Элемент " << key << " добавлен.\n";  
 }  
 else {  
 cout << "Элемент уже существует.\n";  
 }  
}

**Реализация поиска (Hash\_Table):**

void Hash\_Table::search(int key) {  
 int index = hash(key);  
 if (table[index].contains(key)) {  
 cout << "Элемент найден в бакете " << index << ".\n";  
 }  
 else {  
 cout << "Элемент не найден.\n";  
 }  
}

**Отображение таблицы (Hash\_Table):**

void Hash\_Table::display() {  
 for (int i = 0; i < SIZE; i++) {  
 cout << "[" << i << "]: ";  
 table[i].print();  
 cout << endl;  
 }  
}

## Описание архитектуры приложения

* list.h и list.cpp — содержат реализацию шаблонного класса list для управления связными списками.
* Hash\_Table.h и Hash\_Table.cpp — содержат определение и реализацию класса Hash\_Table с хеш-функцией и операциями над таблицей.
* Source.cpp — главный файл с точкой входа и интерактивным меню для управления хеш-таблицей.

# Скриншоты программы

На следующих изображениях представлены примеры вывода консольного приложения:



|  |
| --- |
|  |

Пример работы программы

# Источники

9 Кормен Т.Х., Лейзерсон Ч.Е., Ривест Р.Л., Штайн К. Алгоритмы: построение и анализ. Москва: Вильямс, 2009 г. Струстрап Б. Программирование: принципы и практика с использованием C++. Москва: Вильямс, 2013 г. Документация C++ на cppreference.com. <https://en.cppreference.com>, 2025 г.

# Доступ к исходному коду и результатам работы

Репозиторий: https://github.com/PiVo-vrh/Hash-Table\