Белорусский государственный технологический университет

Факультет информационных технологий

Кафедра программной инженерии

 Лабораторная работа 14

По дисциплине «Основы алгоритмизации и программирования»

На тему «Хеш-таблицы с цепочками»

 Выполнила:

Студентка1 курса 6 группы

Литвинчук Дарья Валерьевна

Преподаватель: асс. Андронова М.В.

2024, Минск

**Вариант 10**

  
#include <iostream>

#include <string>

#include <cstdlib>

#include <ctime>

#include <Windows.h>

#include <climits>

using namespace std;

struct University {

int year;

string name;

string rector;

};

// Реализация хеширования на основе исключающего ИЛИ

class XORHash {

public:

int hash(const string& str, int size) {

int hash = 0;

for (char ch : str) {

hash ^= ch;

}

return hash % size;

}

};

class HashTable {

private:

static const int TABLE\_SIZE = 100; // Размер хеш-таблицы

XORHash hashFunction;

University\* table[TABLE\_SIZE];

public:

HashTable() {

// Инициализируем каждый элемент таблицы

for (int i = 0; i < TABLE\_SIZE; ++i) {

table[i] = NULL;

}

}

void insert(int year, const string& name, const string& rector) {

int index = hashFunction.hash(name, TABLE\_SIZE);

if (table[index] == NULL) {

table[index] = new University;

table[index]->year = year;

table[index]->name = name;

table[index]->rector = rector;

}

else {

cout << "Коллизия обработана. Для университета " << name << " выбран индекс " << index << endl;

// Можно использовать линейное или квадратичное пробирование для разрешения коллизий

}

}

University\* search(const string& name) {

int index = hashFunction.hash(name, TABLE\_SIZE);

if (table[index] != NULL && table[index]->name == name) {

return table[index];

}

else {

return NULL;

}

}

void display() {

for (int i = 0; i < TABLE\_SIZE; ++i) {

if (table[i] != NULL) {

cout << "Индекс " << i << ": Год создания: " << table[i]->year << ", Название: " << table[i]->name << ", Ректор: " << table[i]->rector << endl;

}

}

}

void remove(const string& name) {

int index = hashFunction.hash(name, TABLE\_SIZE);

if (table[index] != NULL && table[index]->name == name) {

delete table[index];

table[index] = NULL;

}

else {

cout << "Университет не найден" << endl;

}

}

};

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

HashTable universityCatalog;

int choice;

int year;

string name;

string rector;

while (true) {

cout << "Выберите действие:" << endl;

cout << "1. Вывести каталог университетов" << endl;

cout << "2. Добавить университет в каталог" << endl;

cout << "3. Удалить университет из каталога" << endl;

cout << "4. Найти университет по названию" << endl;

cout << "5. Исследовать время поиска информации" << endl;

cout << "0. Выход" << endl;

cout << "Ваш выбор: ";

cin >> choice;

switch (choice) {

case 1:

cout << "Текущий каталог университетов:" << endl;

universityCatalog.display();

break;

case 2:

cout << "Введите год создания университета: ";

cin >> year;

cout << "Введите название университета: ";

cin.ignore(); // Очищаем буфер после cin

getline(cin, name);

cout << "Введите ФИО ректора: ";

getline(cin, rector);

universityCatalog.insert(year, name, rector);

cout << "Университет добавлен." << endl;

break;

case 3:

cout << "Введите название университета для удаления: ";

cin.ignore(); // Очищаем буфер после cin

getline(cin, name);

universityCatalog.remove(name);

break;

case 4:

cout << "Введите название университета для поиска: ";

cin.ignore(); // Очищаем буфер после cin

getline(cin, name);

{

University\* result = universityCatalog.search(name);

if (result != NULL) {

cout << "Университет найден: Год создания: " << result->year << ", Название: " << result->name << ", Ректор: " << result->rector << endl;

}

else {

cout << "Университет не найден" << endl;

}

}

break;

case 5:

{

cout << "Введите название университета для тестирования времени поиска: ";

cin.ignore(); // Очищаем буфер после cin

getline(cin, name);

clock\_t start = clock();

universityCatalog.search(name);

clock\_t end = clock();

cout << "Время поиска: " << static\_cast<double>(end - start) / CLOCKS\_PER\_SEC << " секунд" << endl;

}

break;

case 0:

cout << "Выход из программы." << endl;

return 0;

default:

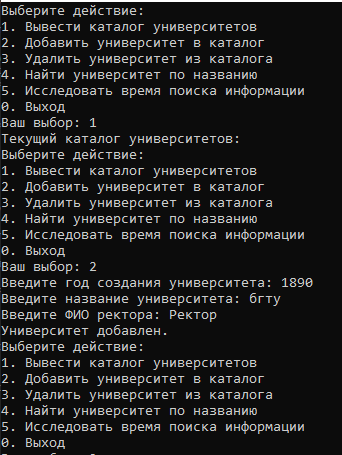
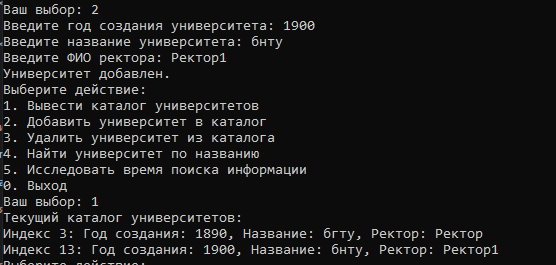
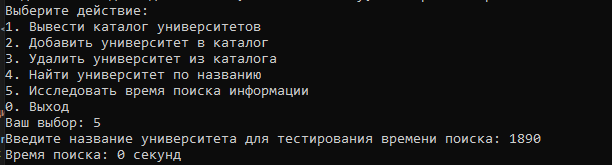
cout << "Неверный ввод. Попробуйте снова." << endl;

}

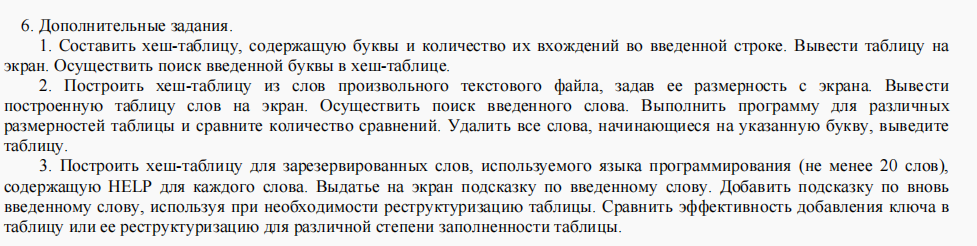
}

return 0;

}

**Дополнительные задания:**



#include <iostream>

#include <unordered\_map>

#include <string>

#include <Windows.h>

using namespace std;

// Функция для создания хеш-таблицы, содержащей буквы и количество их вхождений

void createHashTable(unordered\_map<char, int>& hashTable, const string& input) {

for (char ch : input) {

if (isalpha(ch)) { // Проверяем, является ли символ буквой

hashTable[ch]++; // Увеличиваем счетчик вхождений буквы в хеш-таблице

}

}

}

// Функция для вывода хеш-таблицы на экран

void printHashTable(const unordered\_map<char, int>& hashTable) {

for (const auto& entry : hashTable) {

cout << entry.first << ": " << entry.second << endl; // Выводим букву и количество ее вхождений

}

}

// Функция для поиска буквы в хеш-таблице

void searchLetter(const unordered\_map<char, int>& hashTable, char letter) {

auto it = hashTable.find(letter); // Ищем букву в хеш-таблице

if (it != hashTable.end()) {

cout << "Буква '" << letter << "' найдена с частотой: " << it->second << endl;

}

else {

cout << "Буква '" << letter << "' не найдена в хеш-таблице." << endl;

}

}

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

// Запрашиваем у пользователя ввод строки

string input;

cout << "Введите строку: ";

getline(cin, input);

// Создаем хеш-таблицу для хранения букв и их частоты

unordered\_map<char, int> hashTable;

// Заполняем хеш-таблицу на основе введенной строки

createHashTable(hashTable, input);

// Выводим построенную хеш-таблицу на экран

cout << "Построенная хеш-таблица:" << endl;

printHashTable(hashTable);

// Осуществляем поиск буквы, введенной пользователем

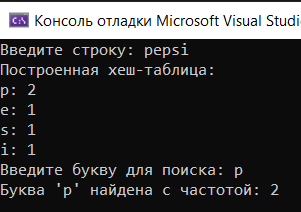
char letter;

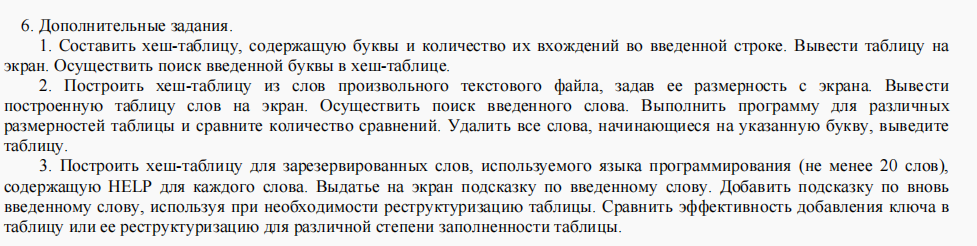
cout << "Введите букву для поиска: ";

cin >> letter;

searchLetter(hashTable, letter);

}





#include <iostream>

#include <fstream>

#include <unordered\_map>

#include <string>

using namespace std;

// Функция для создания хеш-таблицы из слов текстового файла

void createHashTable(unordered\_map<string, int>& hashTable, const string& filename) {

ifstream file(filename); // Открываем файл для чтения

if (!file.is\_open()) {

cerr << "Не удалось открыть файл." << endl;

return;

}

string word;

while (file >> word) {

hashTable[word]++; // Увеличиваем счетчик вхождений слова в хеш-таблице

}

file.close(); // Закрываем файл

}

// Функция для вывода хеш-таблицы на экран

void printHashTable(const unordered\_map<string, int>& hashTable) {

for (const auto& entry : hashTable) {

cout << entry.first << ": " << entry.second << endl; // Выводим слово и количество его вхождений

}

}

// Функция для поиска слова в хеш-таблице

void searchWord(const unordered\_map<string, int>& hashTable, const string& word) {

auto it = hashTable.find(word); // Ищем слово в хеш-таблице

if (it != hashTable.end()) {

cout << "Слово '" << word << "' найдено с частотой: " << it->second << endl;

}

else {

cout << "Слово '" << word << "' не найдено в хеш-таблице." << endl;

}

}

// Функция для удаления всех слов, начинающихся на указанную букву

void deleteWordsStartingWith(unordered\_map<string, int>& hashTable, char letter) {

for (auto it = hashTable.begin(); it != hashTable.end(); ) {

if (it->first[0] == letter) {

it = hashTable.erase(it); // Удаляем слово и получаем итератор на следующий элемент

}

else {

++it; // Переходим к следующему элементу

}

}

}

int main() {

// Задаем размерность таблицы (не используется напрямую, так как unordered\_map управляет размером сам)

int tableSize;

cout << "Введите предполагаемое количество различных слов в файле (размерность хеш-таблицы): ";

cin >> tableSize;

// Создаем хеш-таблицу для хранения слов и их частоты

unordered\_map<string, int> hashTable;

hashTable.reserve(tableSize); // Резервируем место в хеш-таблице для повышения эффективности

// Считываем слова из файла и создаем хеш-таблицу

string filename;

cout << "Введите имя файла: ";

cin >> filename;

createHashTable(hashTable, filename);

// Выводим построенную хеш-таблицу на экран

cout << "Построенная хеш-таблица:" << endl;

printHashTable(hashTable);

// Осуществляем поиск слова, введенного пользователем

string word;

cout << "Введите слово для поиска: ";

cin >> word;

searchWord(hashTable, word);

// Удаляем все слова, начинающиеся на указанную букву

char letter;

cout << "Введите букву для удаления всех слов, начинающихся на нее: ";

cin >> letter;

deleteWordsStartingWith(hashTable, letter);

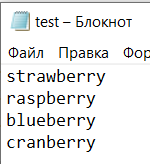
// Выводим хеш-таблицу после удаления слов

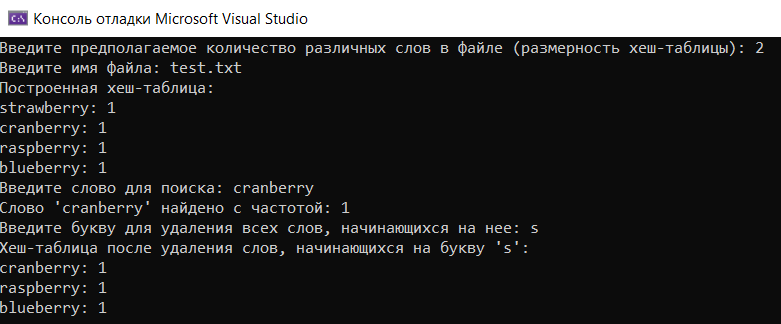
cout << "Хеш-таблица после удаления слов, начинающихся на букву '" << letter << "':" << endl;

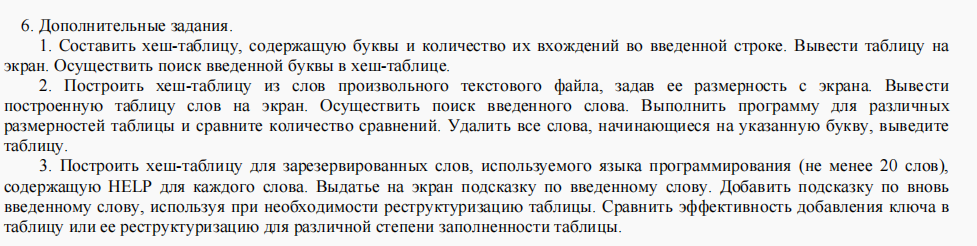
printHashTable(hashTable);

return 0;

}







#include <iostream>

#include <unordered\_map>

#include <string>

#include <Windows.h>

using namespace std;

// Функция для создания и инициализации хеш-таблицы зарезервированных слов с подсказками

unordered\_map<string, string> createReservedWordsMap() {

unordered\_map<string, string> reservedWords;

// Заполнение таблицы зарезервированными словами и их описанием (подсказками)

reservedWords["if"] = "Условно выполняет блок кода.";

reservedWords["else"] = "Выполняет блок кода, если условие в операторе if ложно.";

reservedWords["for"] = "Итерирует по диапазону.";

reservedWords["while"] = "Повторно выполняет блок кода, пока условие истинно.";

reservedWords["do"] = "Выполняет блок кода один раз, а затем повторяет его выполнение, пока условие истинно.";

reservedWords["break"] = "Прерывает выполнение ближайшего цикла или оператора switch.";

reservedWords["continue"] = "Пропускает остаток текущей итерации цикла и переходит к следующей итерации.";

reservedWords["switch"] = "Выбирает блок кода для выполнения из множества вариантов.";

reservedWords["case"] = "Определяет блок кода в операторе switch.";

reservedWords["default"] = "Определяет блок кода по умолчанию в операторе switch, если ни один case не совпал.";

reservedWords["return"] = "Завершает выполнение функции и, при необходимости, возвращает значение.";

reservedWords["void"] = "Указывает, что функция не возвращает значение.";

reservedWords["int"] = "Объявляет переменную целочисленного типа.";

reservedWords["float"] = "Объявляет переменную типа с плавающей запятой.";

reservedWords["double"] = "Объявляет переменную типа с двойной точностью.";

reservedWords["char"] = "Объявляет переменную символьного типа.";

reservedWords["bool"] = "Объявляет переменную логического типа.";

reservedWords["struct"] = "Определяет новый тип структуры.";

reservedWords["typedef"] = "Создает псевдоним для существующего типа данных.";

reservedWords["sizeof"] = "Возвращает размер переменной или типа данных.";

return reservedWords; // Возвращаем заполненную хеш-таблицу

}

// Функция для вывода подсказки по слову

void displayHelp(const unordered\_map<string, string>& reservedWords, const string& word) {

auto it = reservedWords.find(word); // Ищем слово в хеш-таблице

if (it != reservedWords.end()) {

// Если слово найдено, выводим подсказку

cout << "Подсказка для '" << word << "': " << it->second << endl;

}

else {

// Если слово не найдено, сообщаем об этом

cout << "Слово '" << word << "' не найдено среди зарезервированных слов." << endl;

}

}

// Функция для добавления нового слова с подсказкой

void addWord(unordered\_map<string, string>& reservedWords, const string& word, const string& help) {

reservedWords[word] = help; // Добавляем новое слово и подсказку в хеш-таблицу

cout << "Слово '" << word << "' добавлено с подсказкой: " << help << endl; // Сообщаем о добавлении

}

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

// Создаем и инициализируем хеш-таблицу зарезервированных слов

unordered\_map<string, string> reservedWords = createReservedWordsMap();

string input;

while (true) {

// Запрашиваем ввод слова у пользователя

cout << "Введите зарезервированное слово (или 'exit' для выхода): ";

cin >> input;

if (input == "exit") break; // Если введено 'exit', выходим из цикла

auto it = reservedWords.find(input); // Ищем введенное слово в хеш-таблице

if (it != reservedWords.end()) {

// Если слово найдено, выводим подсказку

displayHelp(reservedWords, input);

}

else {

// Если слово не найдено, предлагаем пользователю добавить его

cout << "Слово не найдено. Хотите добавить его? (да/нет): ";

string response;

cin >> response;

if (response == "да") {

// Если пользователь соглашается, запрашиваем подсказку для нового слова

cout << "Введите подсказку для слова '" << input << "': ";

string help;

cin.ignore(); // Игнорируем оставшийся '\n' в потоке ввода

getline(cin, help); // Считываем строку с подсказкой

addWord(reservedWords, input, help); // Добавляем новое слово и подсказку в хеш-таблицу

}

}

}

return 0;

}

