Белорусский государственный технологический университет

Факультет информационных технологий

Кафедра программной инженерии

 Лабораторная работа 14

По дисциплине «Основы алгоритмизации и программирования»

На тему «Хеш-таблицы c цепочками»

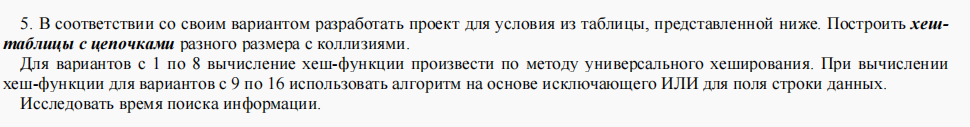
Выполнил:

Студент 1 курса 6 группы

Романов Игорь Вячеславович

Преподаватель: асс. Андронова М.В.

2024, Минск





#include <iostream>

#include <string>

#include <Windows.h>

using namespace std;

// Структура для хранения информации о жильцах

struct Resident {

int apartmentNumber; // Номер квартиры

string name; // Имя жильца

Resident\* next; // Указатель на следующего жильца в цепочке

Resident(int num, const string& resName) : apartmentNumber(num), name(resName), next(nullptr) {}

};

// Хеш-функция на основе исключающего ИЛИ

int hashFunction(int key, int tableSize) {

return key ^ ((key >> 16) ^ (key >> 8)) % tableSize; // Используется XOR для хеширования

}

// Функция для вставки нового жильца в хеш-таблицу

void insertResident(Resident\*\* table, int tableSize, int apartmentNumber, const string& name) {

int index = hashFunction(apartmentNumber, tableSize); // Получаем индекс для вставки

Resident\* newResident = new Resident(apartmentNumber, name); // Создаем нового жильца

newResident->next = table[index]; // Присваиваем новому жильцу следующий элемент в цепочке

table[index] = newResident; // Обновляем указатель на начало цепочки

}

// Функция для удаления жильца из хеш-таблицы

void removeResident(Resident\*\* table, int tableSize, int apartmentNumber) {

int index = hashFunction(apartmentNumber, tableSize); // Получаем индекс для поиска

Resident\* entry = table[index]; // Начинаем поиск с начала цепочки

Resident\* prev = nullptr; // Инициализируем указатель на предыдущий элемент

// Ищем нужного жильца или достигаем конца цепочки

while (entry != nullptr && entry->apartmentNumber != apartmentNumber) {

prev = entry; // Сохраняем текущий элемент как предыдущий

entry = entry->next; // Переходим к следующему элементу в цепочке

}

// Если жилец не найден, выводим сообщение и выходим из функции

if (entry == nullptr) {

cout << "Жилец не найден в квартире " << apartmentNumber << endl;

return;

}

// Если удаляемый жилец находится в начале списка

if (prev == nullptr) {

table[index] = entry->next; // Обновляем указатель на начало цепочки

}

else {

prev->next = entry->next; // Иначе пропускаем удаляемый элемент в цепочке

}

delete entry; // Освобождаем память, выделенную под удаляемого жильца

}

// Функция для отображения содержимого хеш-таблицы

void displayTable(Resident\*\* table, int tableSize) {

// Проходим по всем индексам таблицы

for (int i = 0; i < tableSize; ++i) {

cout << "Индекс " << i << ": ";

Resident\* entry = table[i]; // Начинаем с начала цепочки

// Пока не достигнут конец цепочки

while (entry != nullptr) {

cout << "[Квартира " << entry->apartmentNumber << ": " << entry->name << "] -> ";

entry = entry->next; // Переходим к следующему элементу в цепочке

}

cout << "nullptr" << endl; // Выводим nullptr, когда цепочка закончена

}

}

// Функция для отображения меню

void displayMenu() {

cout << "Выберите вариант:\n";

cout << "1. Добавить жильца\n";

cout << "2. Удалить жильца\n";

cout << "3. Показать хеш-таблицу\n";

cout << "4. Выйти\n";

cout << "Введите ваш выбор: ";

}

// Главная функция

int main() {

// Установка кодировки для работы с кириллицей в консоли Windows

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

const int tableSize = 10; // Размер хеш-таблицы

Resident\* table[tableSize] = { nullptr }; // Создание массива указателей на жильцов

// Цикл работы программы

while (true) {

displayMenu(); // Отображаем меню

int choice;

cin >> choice; // Получаем выбор пользователя

switch (choice) {

case 1: { // Добавить жильца

int apartmentNumber;

string name;

cout << "Введите номер квартиры: ";

cin >> apartmentNumber;

cout << "Введите имя жильца: ";

cin.ignore(); // Очистка буфера ввода

getline(cin, name); // Считываем имя с пробелами

insertResident(table, tableSize, apartmentNumber, name); // Вызываем функцию добавления

break;

}

case 2: { // Удалить жильца

int apartmentNumber;

cout << "Введите номер квартиры для удаления: ";

cin >> apartmentNumber;

removeResident(table, tableSize, apartmentNumber); // Вызываем функцию удаления

break;

}

case 3: // Показать хеш-таблицу

displayTable(table, tableSize); // Вызываем функцию отображения

break;

case 4: // Выйти из программы

cout << "Выход из программы.\n";

return 0;

default: // Некорректный ввод

cout << "Неверный выбор. Попробуйте еще раз.\n";

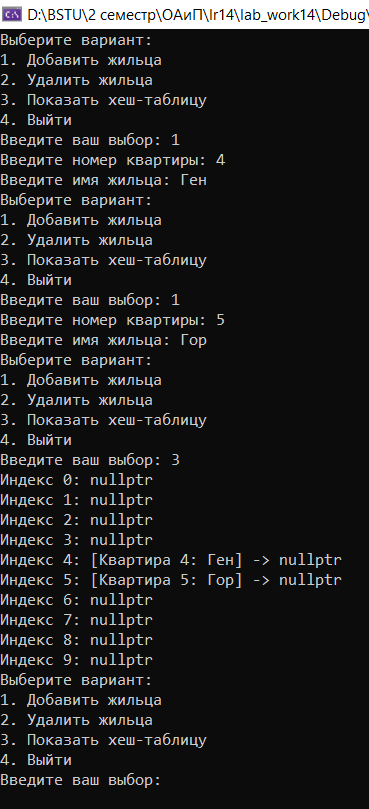
break;

}

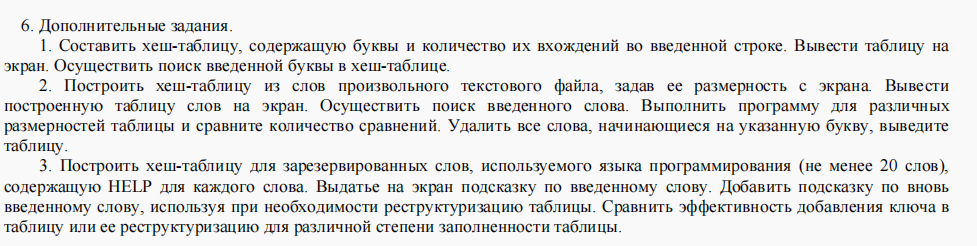
}

return 0;

}



Доп. 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;

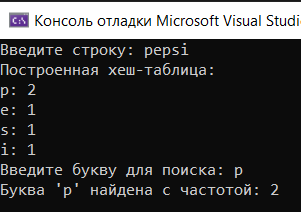
cout << "Введите букву для поиска: ";

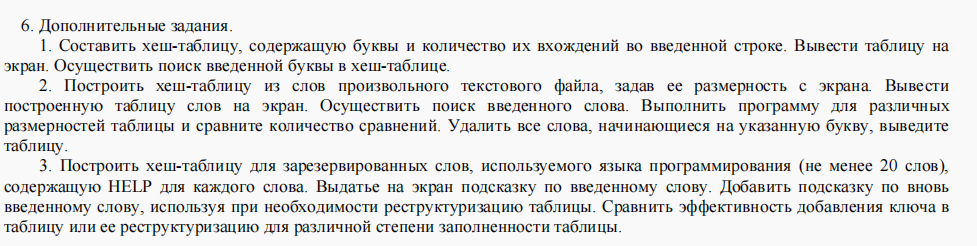
cin >> letter;

searchLetter(hashTable, letter);

return 0;

}





#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);

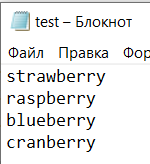
// Выводим хеш-таблицу после удаления слов

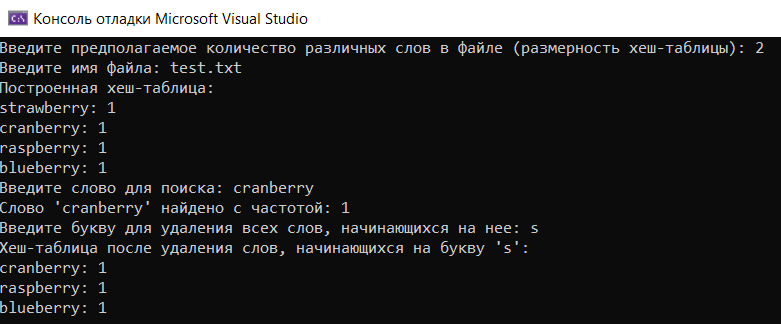
cout << "Хеш-таблица после удаления слов, начинающихся на букву '" << letter << "':" << endl;

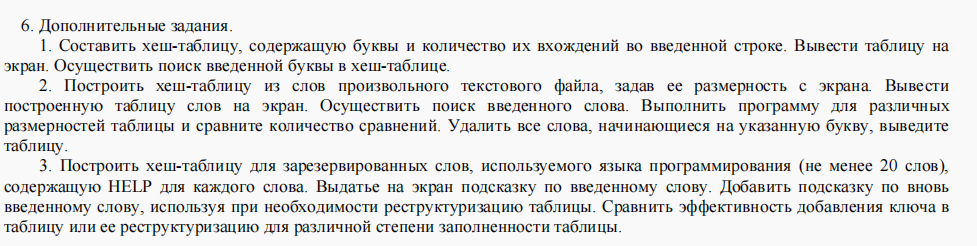
printHashTable(hashTable);

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;

}

