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:

Кулешов Артём

Студент 1 курса 8 группы

Преподаватель: асс. Андронова М.В.

Минск, 2024

Вариант №5

**Прайс-лист**. Создать хеш-таблицу со следующими полями: стоимость товара, название товара. Ключ – стоимость товара.

#include <iostream>

#include <vector>

#include <list>

#include <string>

#include <chrono>

using namespace std;

#define INT\_MAX 32767

struct Notebook {

int key; // Цена

string full\_name; // Название товара

};

// Функция хеширования

int hashFunction(int key, int size) {

return key % size; // Возвращает индекс на основе ключа

}

// Добавление нового элемента в хеш-таблицу

void Add(vector<list<Notebook>>& notebookArray, int key, string& full\_name) {

int ind = hashFunction(key, notebookArray.size());

notebookArray[ind].push\_back({ key, full\_name });

}

// Отображение содержимого хеш-таблицы

void Display(vector<list<Notebook>>& notebookArray) {

cout << "Price-list" << endl;

for (int i = 0; i < notebookArray.size(); ++i) {

cout << "Element " << i << ": ";

for (auto& element : notebookArray[i]) {

cout << "Price: " << element.key << ". Product: " << element.full\_name << " ";

}

cout << endl;

}

}

// Удаление элемента из хеш-таблицы по ключу

void Del(vector<list<Notebook>>& notebookArray, int key) {

int ind = hashFunction(key, notebookArray.size());

for (auto pointer = notebookArray[ind].begin(); pointer != notebookArray[ind].end(); ++pointer) {

if (pointer->key == key) {

notebookArray[ind].erase(pointer);

return;

}

}

}

// Поиск элемента в хеш-таблице по ключу

void Search(vector<list<Notebook>>& notebookArray, int key) {

int ind = hashFunction(key, notebookArray.size());

for (auto& element : notebookArray[ind]) {

if (element.key == key) {

cout << "Price: " << element.key << ". Product: " << element.full\_name << endl;

return;

}

}

cout << "Price " << key << " not found." << endl;

}

int main() {

int choice;

int size;

cout << "Enter size: "; // Запрос размера хеш-таблицы

cin >> size;

vector<list<Notebook>> notebookArray(size);

auto start\_time = chrono::steady\_clock::now(); // Время начала поиска

auto end\_time = chrono::steady\_clock::now(); // Время окончания поиска

for (;;) {

cout << "1. Add" << endl;

cout << "2. Display" << endl;

cout << "3. Del" << endl;

cout << "4. Search" << endl;

cout << "5. Exit" << endl;

cin >> choice;

switch (choice) {

case 1: {

int key;

string full\_name;

cout << "Price: "; // Запрос цены товара

cin >> key;

cout << "Product: "; // Запрос названия товара

cin.ignore();

getline(cin, full\_name);

Add(notebookArray, key, full\_name); // Добавление товара

break;

}

case 2:

Display(notebookArray); // Отображение всех товаров

break;

case 3: {

int key;

cout << "Price to del: "; // Запрос цены товара для удаления

cin >> key;

Del(notebookArray, key); // Удаление товара

break;

}

case 4: {

int key;

cout << "Price to search: "; // Запрос цены товара для поиска

cin >> key;

start\_time = chrono::steady\_clock::now(); // Время начала поиска

Search(notebookArray, key); // Поиск товара

end\_time = chrono::steady\_clock::now(); // Время окончания поиска

cout << "Time of searching: " << chrono::duration\_cast<chrono::nanoseconds>(end\_time - start\_time).count() << " nanoseconds" << endl;

break;

}

case 5:

return 0; // Завершение программы

default:

cout << "Error." << endl; // Сообщение об ошибке

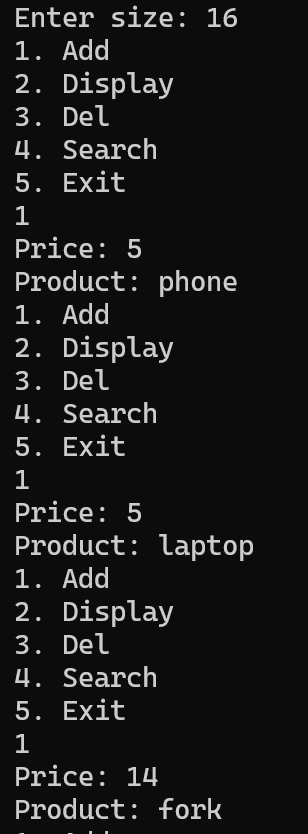
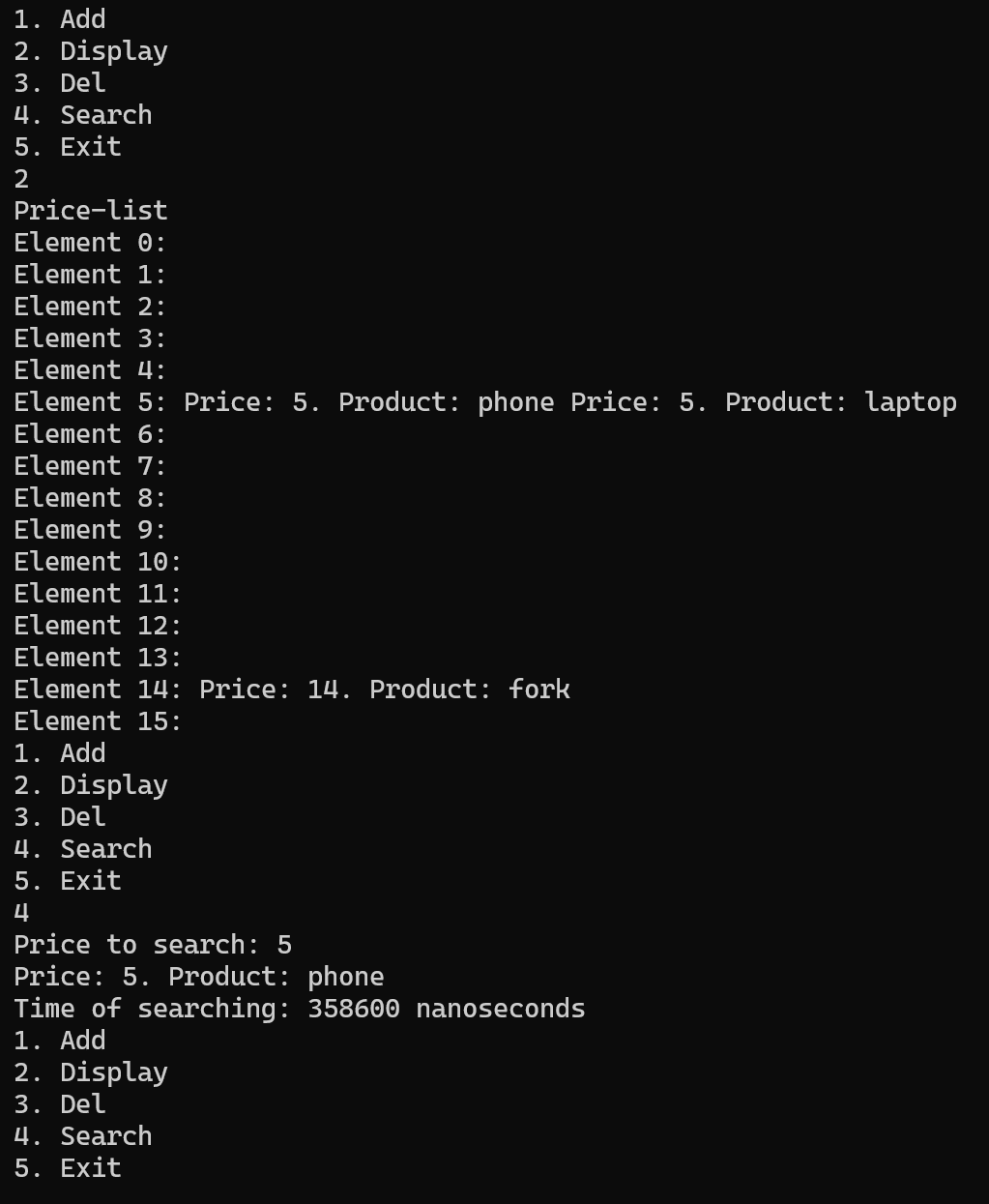
break;

}

}

return 0;

}

6. Дополнительные задания.

1. Составить хеш-таблицу, содержащую буквы и количество их вхождений во введенной строке. Вывести таблицу на экран. Осуществить поиск введенной буквы в хеш-таблице.

#include<iostream>

#include<vector>

#include<list>

#include<string>

#include<cctype>

using namespace std;

int main() {

setlocale(LC\_CTYPE, "rus");

string input\_string;

cout << "Введите строку: ";

getline(cin, input\_string);

vector<list<pair<char, int>>>hash\_table(26); // Используем 26 корзин для хранения букв от 'a' до 'z'

// Заполнение хеш-таблицы с подсчетом количества вхождений каждой буквы

for (char c : input\_string) {

if (isalpha(c)) {

char lowercase\_c = tolower(c);

int index = lowercase\_c - 'a'; // Вычисляем индекс корзины

bool found = false;

for (auto& pair : hash\_table[index]) {

if (pair.first == lowercase\_c) {

pair.second++;

found = true;

break;

}

}

if (!found) {

hash\_table[index].push\_back(make\_pair(lowercase\_c, 1));

}

}

}

// Вывод хеш-таблицы

cout << "Хеш-таблица (буква : количество вхождений):" << endl;

for (int i = 0; i < 26; ++i) {

for (const auto& pair : hash\_table[i]) {

cout << pair.first << " : " << pair.second << endl;

}

}

// Поиск буквы в хеш-таблице

char search\_letter;

cout << "Введите букву для поиска: ";

cin >> search\_letter;

search\_letter = tolower(search\_letter);

int index = search\_letter - 'a';

bool found = false;

for (const auto& pair : hash\_table[index]) {

if (pair.first == search\_letter) {

cout << "Буква '" << search\_letter << "' найдена. Количество вхождений: " << pair.second << endl;

found = true;

break;

}

}

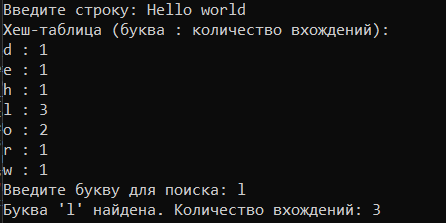
if (!found) {

cout << "Буква '" << search\_letter << "' ненайдена." << endl;

}

return 0;

}



3. Построить хеш-таблицу для зарезервированных слов, используемого языка программирования (не менее 20 слов), содержащую HELP для каждого слова. Выдать на экран подсказку по введенному слову. Добавить подсказку по вновь введенному слову, используя при необходимости реструктуризацию таблицы. Сравнить эффективность добавления ключа в таблицу или ее реструктуризацию для различной степени заполненности таблицы.

#include<iostream>

#include<vector>

#include<list>

#include<string>

#include<cctype>

using namespace std;

struct Entry{

string word;

string help = "HELP";

};

int hashFunction(const string& word) {

return tolower(word[0]) - 'a'; // Простейшая хеш-функция, использующая первую букву слова

}

void addWord(vector<list<Entry>>& hashTable, const string& word) {

int index = hashFunction(word);

hashTable[index].push\_back({ word });

}

int main() {

setlocale(LC\_CTYPE, "rus");

vector<list<Entry>>hashTable(26); // 26 корзин для хранения слов от 'a' до 'z'

// Заполнение хеш-таблицы зарезервированными словами и подсказками

addWord(hashTable, "auto");

addWord(hashTable, "break");

addWord(hashTable, "case");

// Добавьте другие зарезервированные слова по аналогии

string input\_word;

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

while (cin >> input\_word) {

if (input\_word == "exit") {

break;

}

int index = hashFunction(input\_word);

bool found = false;

for (const auto& entry : hashTable[index]) {

if (entry.word == input\_word) {

cout << "Подсказка для '" << input\_word << "': " << entry.help << endl;

found = true;

break;

}

}

if (!found) {

cout << "Слово '" << input\_word << "' не найдено. Добавьте подсказку: ";

string new\_help;

cin >> new\_help;

addWord(hashTable, input\_word);

cout << "Слово '" << input\_word << "' добавлено в таблицу с подсказкой '" << new\_help << "'" << endl;

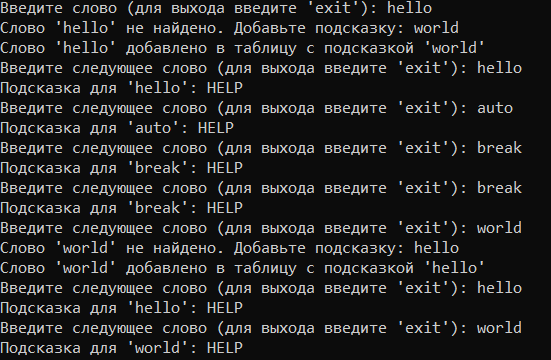
}

cout << "Введите следующее слово (для выхода введите 'exit'): ";

}

return 0;

}



4. В текстовом файле содержатся целые числа. Построить хеш-таблицу из чисел файла. Осуществить поиск введенного целого числа в хеш-таблице. Сравнить результаты количества сравнений при различном наборе данных в файле.

#include<iostream>

#include<fstream>

#include<vector>

#include<list>

using namespace std;

int hashFunction(int value, int tableSize) {

return value% tableSize; // Простейшая хеш-функция, использующая остаток от деления на размер таблицы

}

void addNumber(vector<list<int>>& hashTable, int value) {

int index = hashFunction(value, hashTable.size());

hashTable[index].push\_back(value);

}

bool searchNumber(const vector<list<int>>& hashTable, int value) {

int index = hashFunction(value, hashTable.size());

for (const auto& num : hashTable[index]) {

if (num == value) {

return true;

}

}

return false;

}

void printHashTable(const vector<list<int>>& hashTable) {

for (int i = 0; i < hashTable.size(); ++i) {

cout << "Хеш " << i << ": ";

for (const auto& num : hashTable[i]) {

cout << num << " ";

}

cout << endl;

}

}

int main() {

setlocale(LC\_CTYPE, "rus");

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

vector<list<int>>hashTable(tableSize);

ifstream inputFile("numbers.txt");

int number;

while (inputFile >> number) {

addNumber(hashTable, number);

}

inputFile.close();

cout << "Хеш-таблица после добавления чисел:" << endl;

printHashTable(hashTable);

int searchValue;

cout << "Введите число для поиска в хеш-таблице: ";

cin >> searchValue;

if (searchNumber(hashTable, searchValue)) {

cout << "Число " << searchValue << " найдено в хеш-таблице." << endl;

}

else {

cout << "Число " << searchValue << " не найдено в хеш-таблице." << endl;

}

return 0;

}

