## 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:

Студент 1 курса 8 группы

Рауба Арсений

Преподаватель: асс. Андронова М.В.

Минск, 2024

Вариант 9: **Паспорт**. Реализовать хеш-таблицу со следующими полями: номер паспорта, имя клиента. Ключ – номер паспорта

#include <iostream>

#include <cstring>

#include <cmath>

#include <conio.h>

using namespace std;

struct passport {

int number;

char name[50];

passport& operator=(const passport& x) {

strcpy\_s(this->name, sizeof(this->name), x.name); // Используем strcpy\_s для безопасности

this->number = x.number;

return \*this;

}

void add() {

cout << "Введите имя владельца паспорта:\n";

cin.ignore(); // Очищаем буфер ввода от предыдущих символов

cin.getline(name, 50);

cout << "Введите номер паспорта\n";

cin >> number;

}

};

struct list {

list\* next;

passport that;

list() {

this->next = nullptr;

}

list\* last() {

list\* temp = this;

while (temp->next != nullptr) {

temp = temp->next;

}

return temp;

}

};

class Hash {

unsigned int func;

list mass[8];

public:

Hash() {

func = static\_cast<unsigned int>(pow(2, 32) - 1);

}

void insert() {

passport x;

x.add();

int address = (x.number ^ func) % 8;

list\* where = mass[address].last();

if (where->that.number != 0) { // если узел уже содержит данные, создаем новый узел

where->next = new list;

where = where->next;

}

where->that = x;

}

void search(int x) {

int address = (x ^ func) % 8;

list\* current = &mass[address];

while (current != nullptr) {

if (current->that.number == x) {

cout << "Имя владельца: " << current->that.name << endl;

cout << "Номер паспорта: " << current->that.number << endl;

return;

}

current = current->next;

}

cout << "Паспорт с номером " << x << " не найден." << endl;

}

};

int main() {

setlocale(LC\_ALL, "ru");

Hash get;

char input;

do {

system("cls");

cout << "Выберите функцию:\n";

cout << "1. Добавить новый паспорт\n";

cout << "2. Поиск паспорта\n";

cout << "3. Выход (ESC)\n";

input = \_getch();

if (input == '1') {

get.insert();

}

else if (input == '2') {

int number;

cout << "Введите номер паспорта для поиска:\n";

cin >> number;

get.search(number);

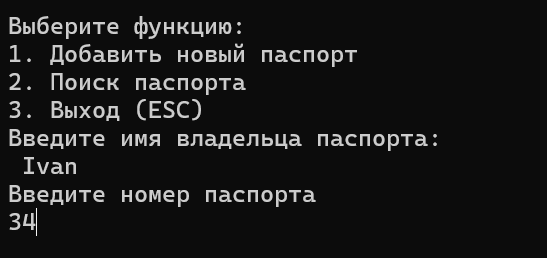
system("pause");

}

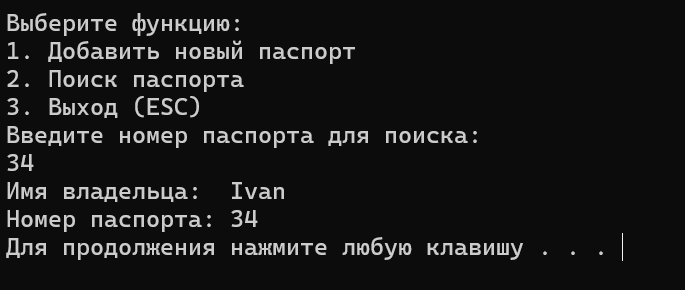
} while (input != 27 && input != '3'); // 27 - код клавиши ESC

return 0;

}



Ввод нового паспорта



Поиск паспорта

**Доп. Задания**

****

#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;

}

}

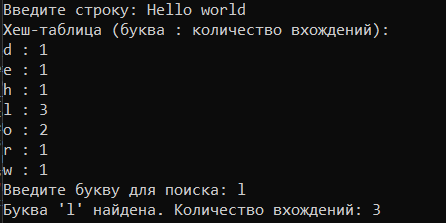
if (!found) {

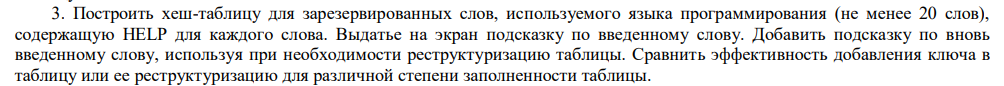
cout << "Буква '" << search\_letter << "' не найдена." << endl;

}

return 0;

}

****

****

#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;

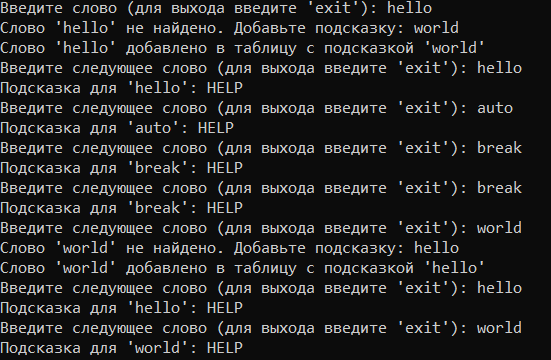
}

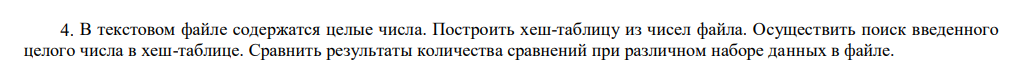
cout << "Введите следующее слово (для выхода введите 'exit'): ";

}

return 0;

}

****

****

#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;

}

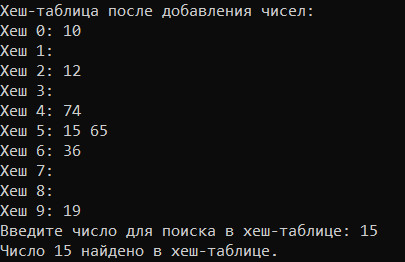
else {

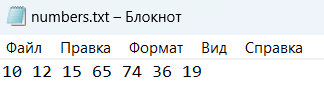
cout << "Число " << searchValue << " не найдено в хеш-таблице." << endl;

}

return 0;

}

****

****