Министерство образования и науки РФ

Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и кибербезопасности

Высшая школа «Компьютерных технологий и информационных систем»

ОТЧЕТ

по дисциплине ««Теория и технология программирования»

**Лабораторная работа № 8**

**Выполнил:**

Cтудент гр. 5130902/40002 Г.Ю. Рюмин

**Проверил**

Ст. преподаватель А.М. Журавская

Санкт-Петербург

2025 г.

**1. Цель работы:**

Цель задания – овладеть методами хеширования\*.

Понять разницу между хешированием, шифрованием и кодированием.

**2. Задание:**

Составить программу для поиска по хешам данных. Хеширование проводить в соответствии с индивидуальными заданиями. В модуле поиска, предусмотреть реализацию обработки случая, при котором хэш-коды различных данных совпадают.

В некоторых заданиях слово «хеширование» и производные от него употреблены не совсем корректно, поняв разницу между хешированием, шифрованием и кодировкой вы сможете корректно определить, где допущены неточности, это не влияет на условие в целом, но важно для понимания и ответы на вопросы. Например, вариант 6 – предложенный вариант преобразования информации не удовлетворяет определению хеширования.

База данных – это могут просто данные из блокнота, формат .txt, проверку на ввод корректного формата файла по желанию. То же относится и к вариантам, где «дан текст» и синонимичные фразы.

В конце данного документа есть ссылки на материалы, но вы можете пользоваться своими материалами.

|  |  |
| --- | --- |
| 27. | В ресторанном меню есть множество блюд. Набор продуктов ограничен несколькими компонентами (например, говядина, сыр, укроп, картофель, морковь и.т.д). Есть продукты, которые определяют свойства блюда (вегетарианское, кошерное, безлактозное, халяльное). Требуется реализовать алгоритм по хэшированию по 3 компонентам и свойству и поиск блюд с введенными свойствами. Например, найти не кошерное блюдо с морковью, мясом и огурцами. |

## **Код программы**

#include <iostream>

#include <string>

#include <vector>

#include <unordered\_map>

#include <algorithm>

#include <sstream>

#include <cctype>

#include <locale>

#ifdef \_WIN32

#include <windows.h>

#endif

using namespace std;

void initLocalization() {

#ifdef \_WIN32

SetConsoleOutputCP(CP\_UTF8);

SetConsoleCP(CP\_UTF8);

#endif

locale::global(locale(""));

cout.imbue(locale());

}

const vector<string> VALID\_COMPONENTS = {

"морковь", "огурцы", "помидоры", "говядина",

"картофель", "лук", "сыр", "укроп", "масло"

};

const vector<string> VALID\_PROPERTIES = {

"вегетарианское", "кошерное", "безлактозное", "халяльное", "некошерное"

};

struct Dish {

string name;

vector<string> components;

bool isVegetarian;

bool isKosher;

bool isLactoseFree;

bool isHalal;

bool operator==(const Dish& other) const {

return name == other.name;

}

};

class MenuDatabase {

private:

unordered\_map<size\_t, vector<Dish>> componentHashTable;

unordered\_map<size\_t, vector<Dish>> propertyHashTable;

size\_t getHash(const string& s) {

return hash<string>{}(s);

}

public:

void addDish(const Dish& dish, const string& property) {

for (const auto& component : dish.components) {

size\_t componentHash = getHash(component);

componentHashTable[componentHash].push\_back(dish);

}

size\_t propertyHash = getHash(property);

propertyHashTable[propertyHash].push\_back(dish);

}

vector<Dish> searchDishes(const vector<string>& components, const string& property) {

vector<Dish> result;

if (components.empty()) return result;

vector<size\_t> componentHashes;

for (const auto& component : components) {

componentHashes.push\_back(getHash(component));

}

size\_t propertyHash = getHash(property);

unordered\_map<size\_t, vector<Dish>> temp;

for (auto hash : componentHashes) {

if (componentHashTable.find(hash) == componentHashTable.end()) return {};

temp[hash] = componentHashTable[hash];

}

vector<Dish> commonDishes;

bool first = true;

for (const auto& pair : temp) {

if (first) {

commonDishes = pair.second;

first = false;

}

else {

vector<Dish> intersection;

for (const auto& dish : commonDishes) {

if (find(pair.second.begin(), pair.second.end(), dish) != pair.second.end()) {

intersection.push\_back(dish);

}

}

commonDishes = intersection;

}

}

if (propertyHashTable.find(propertyHash) == propertyHashTable.end()) return {};

const auto& propertyDishes = propertyHashTable[propertyHash];

for (const auto& dish : commonDishes) {

if (find(propertyDishes.begin(), propertyDishes.end(), dish) != propertyDishes.end()) {

result.push\_back(dish);

}

}

return result;

}

};

template<typename T>

void printNumberedList(const vector<T>& items) {

for (size\_t i = 0; i < items.size(); ++i) {

cout << " " << (i + 1) << ". " << items[i] << endl;

}

}

vector<string> inputComponents() {

while (true) {

cout << "Выберите компоненты (1-" << VALID\_COMPONENTS.size() << "):" << endl;

for (size\_t i = 0; i < VALID\_COMPONENTS.size(); ++i) {

cout << " " << (i + 1) << ". " << VALID\_COMPONENTS[i] << endl;

}

cout << "Введите номера через запятую (1-3 компонента): ";

string input;

getline(cin, input);

input.erase(remove\_if(input.begin(), input.end(), ::isspace), input.end());

vector<int> choices;

stringstream ss(input);

string token;

bool hasError = false;

while (getline(ss, token, ',')) {

if (token.empty()) {

cout << "Ошибка: пустой элемент!" << endl;

hasError = true;

break;

}

if (token.find\_first\_not\_of("0123456789") != string::npos) {

cout << "Ошибка: '" << token << "' не является числом!" << endl;

hasError = true;

break;

}

try {

int num = stoi(token);

if (num < 1 || num > VALID\_COMPONENTS.size()) {

cout << "Ошибка: номер " << num << " вне диапазона!" << endl;

hasError = true;

break;

}

choices.push\_back(num - 1);

}

catch (...) {

cout << "Ошибка: недопустимый ввод!" << endl;

hasError = true;

break;

}

}

if (hasError) {

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

continue;

}

if (choices.empty() || choices.size() > 3) {

cout << "Ошибка: выберите от 1 до 3 компонентов!" << endl << endl;

continue;

}

sort(choices.begin(), choices.end());

choices.erase(unique(choices.begin(), choices.end()), choices.end());

vector<string> components;

for (int idx : choices) {

components.push\_back(VALID\_COMPONENTS[idx]);

}

return components;

}

}

string inputProperty() {

while (true) {

cout << "Выберите свойство:" << endl;

for (size\_t i = 0; i < VALID\_PROPERTIES.size(); ++i) {

cout << " " << (i + 1) << ". " << VALID\_PROPERTIES[i] << endl;

}

cout << "Введите номер: ";

string input;

getline(cin, input);

input.erase(remove\_if(input.begin(), input.end(), ::isspace), input.end());

if (input.empty() || input.find\_first\_not\_of("0123456789") != string::npos) {

cout << "Ошибка: введите число!" << endl << endl;

continue;

}

try {

int num = stoi(input);

if (num < 1 || num > VALID\_PROPERTIES.size()) {

cout << "Ошибка: недопустимый номер!" << endl << endl;

continue;

}

return VALID\_PROPERTIES[num - 1];

}

catch (...) {

cout << "Ошибка: недопустимый ввод!" << endl << endl;

}

}

}

int main() {

initLocalization();

MenuDatabase db;

Dish dish1{ "Овощной салат", {"ab", "ab", "помидоры"}, true, true, true, true };

db.addDish(dish1, "вегетарианское");

Dish dish2{"Винегрет", {"картофель", "морковь", "огурцы"}, true, true, true, true};

db.addDish(dish2, "вегетарианское");

Dish dish3{ "Жаркое", {"говядина", "картофель", "лук"}, false, true, true, true };

db.addDish(dish3, "некошерное");

Dish dish4{ "Гуляш", {"говядина", "лук", "помидоры"}, false, true, true, true };

db.addDish(dish4, "некошерное");

Dish dish5{ "Сырная тарелка", {"сыр", "укроп", "масло"}, true, false, false, true };

db.addDish(dish5, "кошерное");

Dish dish6{ "Гренки", {"сыр", "помидоры", "лук"}, true, false, false, true };

db.addDish(dish6, "безлактозное");

Dish dish7{ "Рулетики из огурцов", {"огурцы", "сыр", "укроп"}, true, true, false, true };

db.addDish(dish7, "вегетарианское");

Dish dish8{ "Фаршированные помидоры", {"помидоры", "сыр", "лук"}, true, true, false, true };

db.addDish(dish8, "вегетарианское");

Dish dish9{ "Картофель фри", {"картофель", "масло", "укроп"}, true, true, true, true };

db.addDish(dish9, "вегетарианское");

Dish dish10{ "Тушёные овощи", {"морковь", "лук", "масло"}, true, true, true, true };

db.addDish(dish10, "вегетарианское");

Dish dish11{ "Овощной суп", {"картофель", "морковь", "укроп"}, true, true, true, true };

db.addDish(dish11, "вегетарианское");

Dish dish12{ "Харчо", {"говядина", "рис", "помидоры"}, false, true, true, true };

db.addDish(dish12, "некошерное");

Dish dish13{ "Морковные оладьи", {"морковь", "масло", "лук"}, true, true, true, true };

db.addDish(dish13, "вегетарианское");

Dish dish14{ "Сырный суп", {"сыр", "картофель", "укроп"}, true, false, false, true };

db.addDish(dish14, "кошерное");

Dish dish15{ "Мясная запеканка", {"говядина", "сыр", "лук"}, false, false, false, true };

db.addDish(dish15, "халяльное");

while (true) {

vector<string> components = inputComponents();

string property = inputProperty();

vector<Dish> results = db.searchDishes(components, property);

if (results.empty()) {

cout << "Блюд по вашему запросу не найдено." << endl;

}

else {

cout << endl << "Найдены блюда:" << endl;

for (const auto& dish : results) {

cout << "-> " << dish.name << " (";

for (const auto& c : dish.components) cout << c << " ";

cout << ")" << endl;

}

}

cout << endl << "Продолжить поиск? (y/n): ";

char choice;

cin >> choice;

cin.ignore();

if (tolower(choice) != 'y') break;

}

return 0;

}

**Пример работы программы**

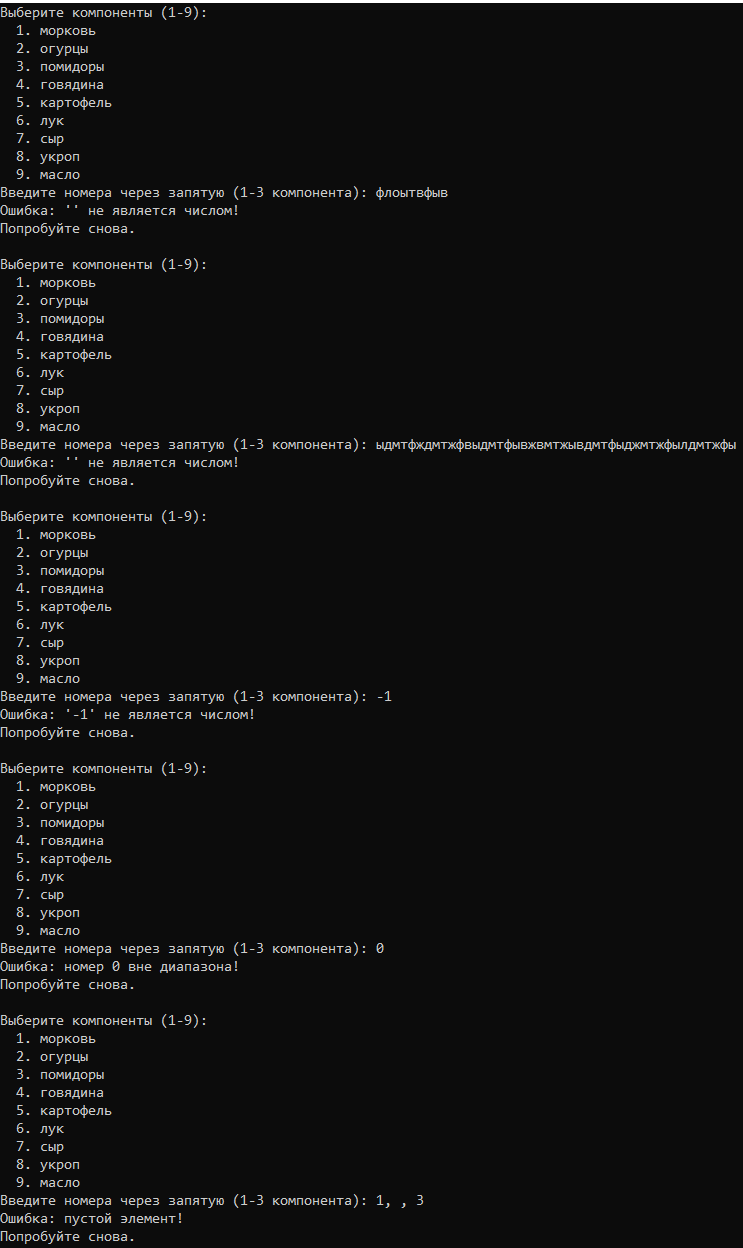


Рисунок 1 – Пример работы программы с некорректным вводом

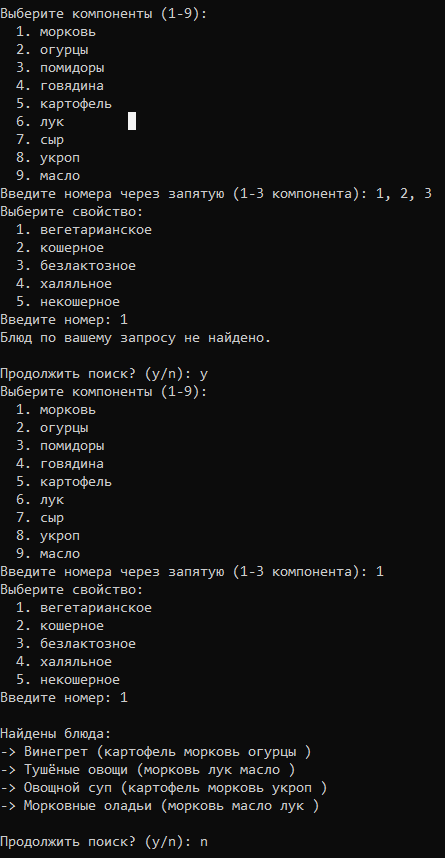
**

Рисунок 2 ­ Пример работы программы с корректным вводом

## **Вывод**

В ходе выполнения лабораторной работы была реализована система хеширования для поиска блюд в ресторанном меню по заданным компонентам и свойствам. Работа позволила закрепить понимание принципов хеширования, его отличий от шифрования и кодирования, а также продемонстрировала практическое применение методов обработки коллизий. Была создана структура данных, обеспечивающая эффективный поиск комбинаций продуктов и характеристик блюд, что подтвердило возможность использования хеш-функций для быстрого доступа к информации в условиях множественных параметров фильтрации.