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 10 группы

Мандрик Алексей Иванович

Преподаватель: асс. Андронова М.В.

Минск, 2024

В соответствии со своим вариантом разработать проект для условия из таблицы, представленной ниже. Построить ***хеш-таблицы с цепочками*** разного размера с коллизиями.

Для вариантов с 1 по 8 вычисление хеш-функции произвести по методу универсального хеширования. При вычислении хеш-функции для вариантов с 9 по 16 использовать алгоритм на основе исключающего ИЛИ для поля строки данных.

Исследовать время поиска информации.

**Вариант 8:**

****

**Код программы:**

#include <iostream>

#include <string>

#include <vector>

#include <chrono>

using namespace std;

// Структура для хранения информации о рейсе

struct FlightInfo {

string destinationAirport; // Аэропорт назначения

string flightNumber; // Номер рейса

};

// Класс Хеш-таблица

class HashTable {

public:

// Конструктор с указанием размера таблицы

HashTable(int size) {

table.resize(size);

}

// Метод для добавления информации о рейсе

void addFlight(FlightInfo flight) {

int index = hashFunction(flight.flightNumber); // Вычисление индекса для хранения по номеру рейса

table[index].push\_back(flight); // Добавление вектора рейсов по индексу

}

// Метод для получения информации о рейсе по его номеру

vector<FlightInfo> getFlights(string flightNumber) {

int index = hashFunction(flightNumber); // Вычисление индекса для поиска по номеру рейса

return table[index]; // Возвращаем вектор рейсов по данному индексу

}

// Метод для вывода содержимого таблицы

void printTable() {

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

if (table[i].size() > 0) {

for (int j = 0; j < table[i].size(); j++) {

cout << "\t" << table[i][j].destinationAirport << " - " << table[i][j].flightNumber << endl; // Вывод информации о рейсе

}

}

}

}

// Метод для удаления информации о рейсе

void deleteFlight(string flightNumber) {

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

if (table[i].size() > 0) {

for (int j = 0; j < table[i].size(); j++) {

if (table[i][j].flightNumber == flightNumber) { // Если номер рейса найден

table[i].erase(table[i].begin() + j); // Удаляем соответствующий рейс

break;

}

}

}

}

}

// Метод для очистки таблицы

void clearTable() {

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

table[i].clear(); // Очищаем все элементы вектора

}

}

private:

vector<vector<FlightInfo>> table; // Хеш-таблица, где каждый элемент - вектор рейсов

// Простейшая хеш-функция, основанная на суммировании кодов символов в номере рейса

int hashFunction(string flightNumber) {

int sum = 0;

for (char c : flightNumber) {

sum += c;

}

return sum % table.size(); // Возвращаем остаток от деления суммы на размер таблицы

}

};

int main() {

setlocale(LC\_ALL, "Rus");

HashTable table(128); // Создаем хеш-таблицу размером 128

int choice = 0;

while (choice != 6) { // Меню выбора действий

cout << "Меню:\n";

cout << "1. Добавить новый рейс\n";

cout << "2. Поиск рейсов по номеру\n";

cout << "3. Удалить рейс\n";

cout << "4. Очистить таблицу\n";

cout << "5. Вывести таблицу\n";

cout << "6. Выход\n";

cout << "Введите ваш выбор: ";

cin >> choice;

switch (choice) {

case 1: {

string destinationAirport, flightNumber;

cout << "Введите аэропорт назначения: ";

cin >> destinationAirport;

cout << "Введите номер рейса: ";

cin >> flightNumber;

table.addFlight({ destinationAirport, flightNumber }); // Добавление нового рейса

break;

}

case 2: {

string flightNumber;

cout << "Введите номер рейса: ";

cin >> flightNumber;

auto begin = std::chrono::steady\_clock::now(); // Начало отсчета времени

vector<FlightInfo> flights = table.getFlights(flightNumber); // Получение информации о рейсе по его номеру

auto end = std::chrono::steady\_clock::now(); // Конец отсчета времени

auto elapsed\_ms = std::chrono::duration\_cast<std::chrono::nanoseconds>(end - begin); // Вычисление времени выполнения

std::cout << "Время: " << elapsed\_ms.count() << " наносекунд\n";

if (flights.size() == 0) {

cout << "Рейс с номером " << flightNumber << " не найден" << endl;

}

else {

cout << "Информация о рейсе " << flightNumber << ":\n";

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

cout << "\t" << flights[i].destinationAirport << " - " << flights[i].flightNumber << endl;

}

}

break;

}

case 3: {

string flightNumber;

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

cin >> flightNumber;

table.deleteFlight(flightNumber); // Удаление рейса

break;

}

case 4: {

table.clearTable(); // Очистка таблицы

cout << "Таблица очищена" << endl;

break;

}

case 5: {

table.printTable(); // Вывод таблицы

break;

}

default: {

exit(1);

}

}

}

return 0;

}

**Результат выполнения:**

****

**Дополнительные задания:**

**Вариант 2:**

****

**Код программы:**

#include <iostream>

#include <string>

#include <vector>

#include <chrono>

using namespace std;

struct EmailAddress {

string address; // адрес электронной почты

string surname; // фамилия

int year; // год

};

class HashTable {

public:

HashTable(int size) {

table.resize(size);

}

void addEmail(EmailAddress email) { // метод для добавления электронного адреса

int index = hashFunction(email.year); // индекс для хеширования

table[index].push\_back(email); // добавление в хеш-таблицу

}

vector<EmailAddress> getEmails(int year) { // метод для получения электронных адресов по году

int index = hashFunction(year); // индекс для хеширования

return table[index]; // возвращаем адреса

}

void printTable() { // вывод хеш-таблицы

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

if (table[i].size() > 0) {

for (int j = 0; j < table[i].size(); j++) {

cout << "\t" << table[i][j].address << " - " << table[i][j].surname << endl; // выводим адреса

}

}

}

}

void deleteEmail(string address) { // метод для удаления электронного адреса

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

if (table[i].size() > 0) {

for (int j = 0; j < table[i].size(); j++) {

if (table[i][j].address == address) { // если адрес найден

table[i].erase(table[i].begin() + j); // удаляем

break;

}

}

}

}

}

void clearTable() { // метод для очистки таблицы

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

table[i].clear();

}

}

private:

vector<vector<EmailAddress>> table; // хеш-таблица для хранения адресов

int hashFunction(int year) { // хеш-функция

return year % table.size();

}

};

int main() {

setlocale(LC\_ALL, "Rus");

HashTable table(128); // создание хеш-таблицы размером 128

int choice = 0;

while (choice != 6) { // цикл для меню

cout << "Меню:\n";

cout << "1. Добавить новый адрес электронной почты\n";

cout << "2. Поиск адресов электронной почты по году\n";

cout << "3. Удалить адрес электронной почты\n";

cout << "4. Очистить таблицу\n";

cout << "5. Вывести таблицу\n";

cout << "6. Выход\n";

cout << "Введите ваш выбор: ";

cin >> choice;

switch (choice) {

case 1: {

string address, surname;

int year;

cout << "Введите адрес электронной почты: ";

cin >> address;

cout << "Введите фамилию: ";

cin >> surname;

cout << "Введите год: ";

cin >> year;

table.addEmail({ address, surname, year }); // добавление адреса

break;

}

case 2: {

double startTime, endTime;

int year;

cout << "Введите год: ";

cin >> year;

auto begin = std::chrono::steady\_clock::now();

vector<EmailAddress> emails = table.getEmails(year); // получение адресов по году

auto end = std::chrono::steady\_clock::now();

auto elapsed\_ms = std::chrono::duration\_cast<std::chrono::nanoseconds>(end - begin);

std::cout << "Время: " << elapsed\_ms.count() << " наносекунд\n";

if (emails.size() == 0) {

cout << "Адреса электронной почты не найдены для года " << year << endl;

}

else {

cout << "Адреса электронной почты для года " << year << ":\n";

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

cout << "\t" << emails[i].address << " - " << emails[i].surname << endl;

}

}

break;

}

case 3: {

string address;

cout << "Введите адрес электронной почты для удаления: ";

cin >> address;

table.deleteEmail(address); // удаление адреса

break;

}

case 4: {

table.clearTable(); // очистка таблицы

cout << "Таблица очищена" << endl;

break;

}

case 5: {

table.printTable(); // вывод таблицы

break;

}

default: {

exit(1);

}

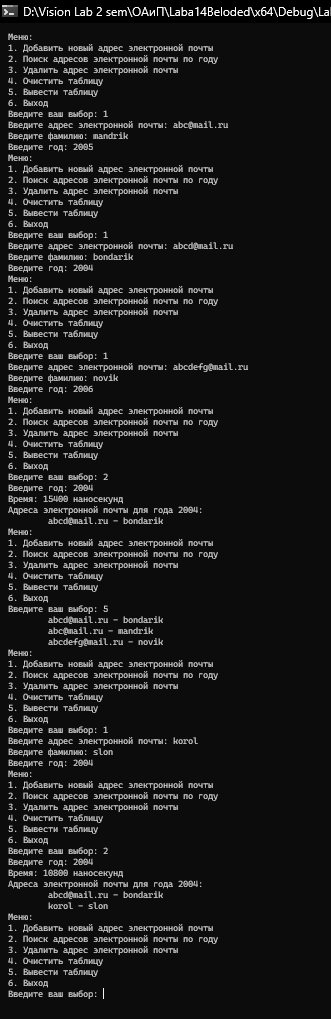
}

}

return 0;

}

**Результат выполнения:**

****

**Вариант 3:**

****

**Код программы:**

#include <iostream>

#include <string>

#include <vector>

using namespace std;

// Структура для хранения информации о школе

struct Shool {

string surname; // Фамилия

int num; // Номер школы

};

// Класс Хеш-таблица

class HashTable {

public:

// Конструктор с указанием размера таблицы

HashTable(int size) {

table.resize(size);

}

// Метод для добавления информации о школе

void addSchool(Shool school) {

int index = hashFunction(school.num); // Вычисление индекса для хранения по номеру школы

table[index].push\_back(school); // Добавление вектора школ по индексу

}

// Метод для получения информации о школах по их номеру

vector<Shool> getSchools(int num) {

int index = hashFunction(num); // Вычисление индекса для поиска по номеру школы

return table[index]; // Возвращаем вектор школ по данному индексу

}

// Метод для вывода содержимого таблицы

void printTable() {

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

if (table[i].size() > 0) {

cout << "Год " << i << ":\n"; // Выводим год

for (int j = 0; j < table[i].size(); j++) {

cout << "\t" << table[i][j].surname << " - " << table[i][j].surname << endl; // Выводим информацию о школе

}

}

}

}

// Метод для удаления информации о школе

void deleteSchool(string surname) {

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

if (table[i].size() > 0) {

for (int j = 0; j < table[i].size(); j++) {

if (table[i][j].surname == surname) { // Если фамилия найдена

table[i].erase(table[i].begin() + j); // Удаляем соответствующую школу

break;

}

}

}

}

}

// Метод для очистки таблицы

void clearTable() {

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

table[i].clear(); // Очищаем все элементы вектора

}

}

private:

vector<vector<Shool>> table; // Хеш-таблица, где каждый элемент - вектор школ

// Простейшая хеш-функция, основанная на остатке от деления номера школы на размер таблицы

int hashFunction(int num) {

return num % table.size();

}

};

int main() {

setlocale(LC\_ALL, "Rus");

HashTable table(10); // Создаем хеш-таблицу размером 10

int choice = 0;

while (choice != 6) { // Меню выбора действий

cout << "Меню:\n";

cout << "1. Добавить новую школу\n";

cout << "2. Поиск фамилий по номеру школы\n";

cout << "3. Удалить фамилию\n";

cout << "4. Очистить таблицу\n";

cout << "5. Вывести таблицу\n";

cout << "6. Выход\n";

cout << "Введите ваш выбор: ";

cin >> choice;

switch (choice) {

case 1: {

string surname;

int num;

cout << "Введите фамилию: ";

cin >> surname;

cout << "Введите номер школы: ";

cin >> num;

table.addSchool({ surname, num }); // Добавление новой школы

break;

}

case 2: {

int num;

cout << "Введите номер школы: ";

cin >> num;

vector<Shool> schools = table.getSchools(num); // Получение информации о школах по их номеру

if (schools.size() == 0) {

cout << "Нет адресов школ для номера " << num << endl;

}

else {

cout << "Адреса школ для номера " << num << ":\n";

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

cout << "\t" << schools[i].surname << endl;

}

}

break;

}

case 3: {

string surname;

cout << "Введите фамилию для удаления: ";

cin >> surname;

table.deleteSchool(surname); // Удаление школы

break;

}

case 4: {

table.clearTable(); // Очистка таблицы

cout << "Таблица очищена" << endl;

break;

}

case 5: {

table.printTable(); // Вывод таблицы

break;

}

default: {

exit(1);

}

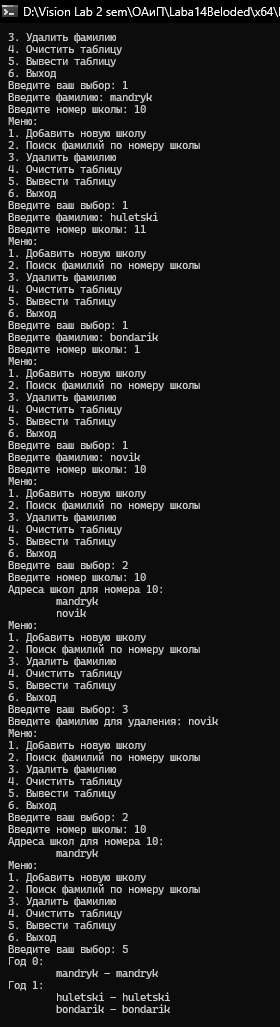
}

}

return 0;

}

**Результат выполнения:**



**Вариант 4:**



**Код программы:**

#include <iostream>

#include <string>

#include <vector>

using namespace std;

// Структура для хранения информации о книге

struct Book {

string name; // Название книги

int num; // Номер книги

};

// Класс Хеш-таблица

class HashTable {

public:

// Конструктор с указанием размера таблицы

HashTable(int size) {

table.resize(size);

}

// Метод для добавления информации о книге

void addBook(Book book) {

int index = hashFunction(book.num); // Вычисление индекса для хранения по номеру книги

table[index].push\_back(book); // Добавление вектора книг по индексу

}

// Метод для получения информации о книгах по их номеру

vector<Book> getBooks(int num) {

int index = hashFunction(num); // Вычисление индекса для поиска по номеру книги

return table[index]; // Возвращаем вектор книг по данному индексу

}

// Метод для вывода содержимого таблицы

void printTable() {

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

if (table[i].size() > 0) {

cout << "Год " << i << ":\n"; // Выводим год

for (int j = 0; j < table[i].size(); j++) {

cout << "\t" << table[i][j].name << " - " << table[i][j].name << endl; // Выводим информацию о книге

}

}

}

}

// Метод для удаления информации о книге

void deleteBook(string name) {

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

if (table[i].size() > 0) {

for (int j = 0; j < table[i].size(); j++) {

if (table[i][j].name == name) { // Если название найдено

table[i].erase(table[i].begin() + j); // Удаляем соответствующую книгу

break;

}

}

}

}

}

// Метод для очистки таблицы

void clearTable() {

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

table[i].clear(); // Очищаем все элементы вектора

}

}

private:

vector<vector<Book>> table; // Хеш-таблица, где каждый элемент - вектор книг

// Простейшая хеш-функция, основанная на остатке от деления номера книги на размер таблицы

int hashFunction(int num) {

return num % table.size();

}

};

int main() {

HashTable table(10); // Создаем хеш-таблицу размером 10

setlocale(LC\_ALL, "Rus");

int choice = 0;

while (choice != 6) { // Меню выбора действий

cout << "Меню:\n";

cout << "1. Добавить новую книгу\n";

cout << "2. Поиск названий по номеру книги\n";

cout << "3. Удалить название\n";

cout << "4. Очистить таблицу\n";

cout << "5. Вывести таблицу\n";

cout << "6. Выход\n";

cout << "Введите ваш выбор: ";

cin >> choice;

switch (choice) {

case 1: {

string name;

int num;

cout << "Введите название: ";

cin >> name;

cout << "Введите номер книги: ";

cin >> num;

table.addBook({ name, num }); // Добавление новой книги

break;

}

case 2: {

int num;

cout << "Введите номер книги: ";

cin >> num;

vector<Book> books = table.getBooks(num); // Получение информации о книгах по их номеру

if (books.size() == 0) {

cout << "Нет адресов книг для номера " << num << endl;

}

else {

cout << "Адреса книг для номера " << num << ":\n";

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

cout << "\t" << books[i].name << endl;

}

}

break;

}

case 3: {

string name;

cout << "Введите название для удаления: ";

cin >> name;

table.deleteBook(name); // Удаление книги

break;

}

case 4: {

table.clearTable(); // Очистка таблицы

cout << "Таблица очищена" << endl;

break;

}

case 5: {

table.printTable(); // Вывод таблицы

break;

}

default: {

exit(1);

}

}

}

return 0;

}

**Результат выполнения:**

