Белорусский государственный технологический университет

Факультет информационных технологий

Кафедра программной инженерии

 Лабораторная работа №13

По дисциплине «Основы алгоритмизации и программирования»

На тему « **Хеш-таблицы c открытой адресацией** »

Выполнил:

Студент 1 курса 10 группы

Мандрик Алексей Иванович

Преподаватель: асс. Андронова М.В.

Минск, 2024

В соответствии со своим вариантом построить ***хеш-таблицы*** с ***открытой*** адресацией разного размера, например, 16, 32 или 32, 64, 128 с коллизиями. В таблице **h**'**(key)** −значение хеш-функции, приведшее к коллизии.

Исследовать время поиска в хеш-таблицах. В приложении Excel построить соответствующие графики.

**Вариант 8:**

****

**Код программы:**

**Main.cpp**

#include "Hash.h"

#include <iostream>

#include <string>

#include <iomanip>

#include <stdio.h>

using namespace std;

// Структура AAA для элементов хеш-таблицы

struct AAA

{

int key; // ключ элемента

string keyString; // ключ в виде строки

const char\* mas; // указатель на строку

AAA(int k, const char z[])

{

key = k; mas = z;

}

AAA() {}

};

//-------------------------------

// Функция для извлечения ключа из элемента AAA

int key(void\* d)

{

AAA\* f = (AAA\*)d;

return f->key;

}

//-------------------------------

// Функция для вывода элемента AAA

void AAA\_print(void\* d)

{

cout << " ключ " << ((AAA\*)d)->keyString << " - " << ((AAA\*)d)->mas << endl;

}

//-------------------------------

// Функция для генерации ключа из строки

int getkeyInt(string k)

{

// Переменная для нового ключа

int key = 0;

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

key += (int)k[i];

}

return key;

}

//-------------------------------

int main(int argc, char\* argv[]) {

setlocale(LC\_ALL, "Rus"); // Установка русской локали

int siz = 10, choice;

string k;

cout << "Введите размер хеш-таблицы" << endl; cin >> siz;

Object H = create(siz, key); // Создание хеш-таблицы

for (;;)

{

cout << "1 - вывод хеш-таблицы" << endl;

cout << "2 - добавление элемента" << endl;

cout << "3 - удаление элемента" << endl;

cout << "4 - поиск элемента" << endl;

cout << "0 - выход" << endl;

cout << "сделайте выбор" << endl;

cin >> choice;

switch (choice)

{

case 0: exit(0); // Выход из программы

case 1: H.scan(AAA\_print); break; // Вывод хеш-таблицы

case 2: {

AAA\* a = new AAA;

char\* str = new char[20];

cout << "введите ключ" << endl;

getchar(); // Очистка буфера ввода

getline(cin, k); // Считывание ключа

int key = 0;

a->key = getkeyInt(k); // Получение числового ключа

a->keyString = k; // Сохранение ключа в виде строки

cout << "введите строку" << endl;

cin.getline(str, 20); // Считывание строки

a->mas = str; // Сохранение строки в элементе

if (H.N == H.size)

cout << "Таблица заполнена" << endl;

else

H.insert(a); // Вставка элемента в хеш-таблицу

} break;

case 3: { cout << "введите ключ для удаления" << endl;

getchar(); // Очистка буфера ввода

getline(cin, k); // Считывание ключа

H.deleteByKey(getkeyInt(k)); // Удаление элемента по ключу

} break;

case 4: { cout << "введите ключ для поиска" << endl;

getchar(); // Очистка буфера ввода

getline(cin, k); // Считывание ключа

if (H.search(getkeyInt(k)) == NULL)

cout << "Элемент не найден" << endl;

else

AAA\_print(H.search(getkeyInt(k))); // Поиск и вывод элемента

} break;

}

}

return 0;

}

**Hash.cpp**

#include "Hash.h"

#include <iostream>

// Хеш-функция

int HashFunction(int key, int size, int p) {

double key2 = 5 \* ((0.6180339887499 \* key) - int((0.6180339887499 \* key)));

return (p + key) % size;

}

//-------------------------------

// Функция для вычисления следующего хеша

int Next\_hash(int hash, int size, int p) {

return (hash + 5 \* p + 3 \* p \* p) % size;

}

//-------------------------------

// Создание объекта хеш-таблицы

Object create(int size, int(\*getkey)(void\*)) {

return \*(new Object(size, getkey));

}

//-------------------------------

// Конструктор объекта хеш-таблицы

Object::Object(int size, int(\*getkey)(void\*)) {

N = 0;

this->size = size;

this->getKey = getkey;

this->data = new void\* [size];

for (int i = 0; i < size; ++i)

data[i] = NULL;

}

//-------------------------------

// Вставка элемента в хеш-таблицу

bool Object::insert(void\* d) {

bool b = false;

if (N != size)

for (int i = 0, t = getKey(d), j = HashFunction(t, size, 0);

i != size && !b; j = Next\_hash(j, size, ++i))

if (data[j] == NULL || data[j] == DEL) {

data[j] = d;

N++;

b = true;

}

return b;

}

//-------------------------------

// Поиск индекса элемента по ключу

int Object::searchInd(int key) {

int t = -1;

bool b = false;

if (N != 0)

for (int i = 0, j = HashFunction(key, size, 0); data[j] != NULL && i != size && !b; j = HashFunction(key, size, ++i))

if (data[j] != DEL)

if (getKey(data[j]) == key) {

t = j; b = true;

}

return t;

}

//-------------------------------

// Поиск элемента по ключу

void\* Object::search(int key) {

int t = searchInd(key);

return(t >= 0) ? (data[t]) : (NULL);

}

//-------------------------------

// Удаление элемента по ключу

void\* Object::deleteByKey(int key) {

int i = searchInd(key);

void\* t = data[i];

if (t != NULL) {

data[i] = DEL;

N--;

}

return t;

}

//-------------------------------

// Удаление элемента по значению

bool Object::deleteByValue(void\* d) {

return(deleteByKey(getKey(d)) != NULL);

}

//-------------------------------

// Просмотр элементов хеш-таблицы

void Object::scan(void(\*f)(void\*)) {

for (int i = 0; i < this->size; i++) {

std::cout << " Элемент " << i;

if ((this->data)[i] == NULL)

std::cout << " пусто" << std::endl;

else

if ((this->data)[i] == DEL)

std::cout << " удален" << std::endl;

else

f((this->data)[i]);

}

}

**Hash.h**

#pragma once

// Определение символа удаления

#define HASHDEL (void\*) -1

// Структура хеш-таблицы

struct Object

{

void\*\* data; // Массив указателей

Object(int, int(\*)(void\*)); // Конструктор

int size;

int N; // Количество элементов

int(\*getKey)(void\*); // Функция для получения ключа

bool insert(void\*); // Вставка элемента

int searchInd(int key); // Поиск индекса элемента по ключу

void\* search(int key); // Поиск элемента по ключу

void\* deleteByKey(int key); // Удаление элемента по ключу

bool deleteByValue(void\*); // Удаление элемента по значению

void scan(void(\*f)(void\*)); // Проход по всем элементам

};

// Установка значения символа удаления

static void\* DEL = (void\*)HASHDEL;

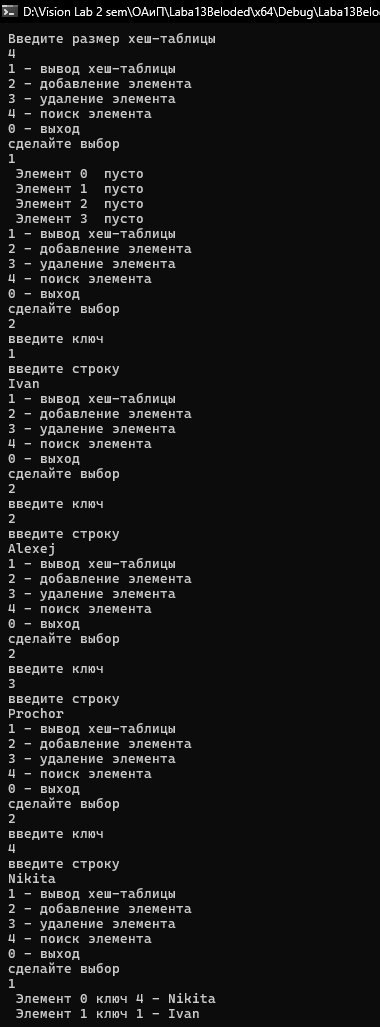
// Создание хеш-таблицы

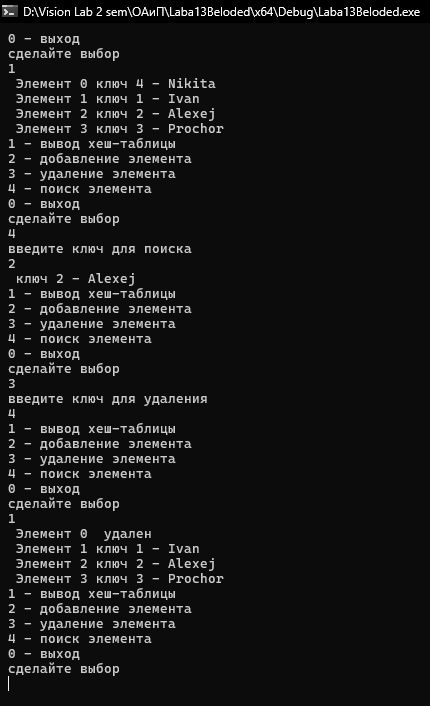
Object create(int size, int(\*getkey)(void\*));

// Очистка символа удаления

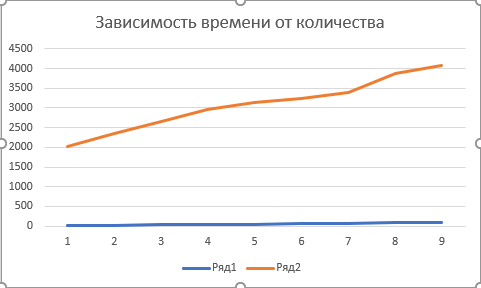
#undef HASHDEL

**Результат выполнения:**





**Диаграмма**



**Дополнительные задания:**

**Вариант 2 и по совместительству 7))))**

****

****

**Код программы:**

#include <iostream>

#include <string>

#include <cstdlib>

#include <vector>

const int TABLE\_SIZE = 128;

// Структура для узла хеш-таблицы

struct HashNode {

std::string key; // Ключ

int value; // Значение

bool is\_deleted; // Флаг удаления

// Конструктор

HashNode() : key(""), value(-1), is\_deleted(false) {}

};

// Класс хеш-таблицы

class HashTable {

private:

std::vector<HashNode> table; // Вектор узлов

public:

// Конструктор

HashTable() : table(TABLE\_SIZE) {}

// Хеш-функция

static unsigned int hash(const std::string& key, unsigned int p = 1000000007, unsigned int a = 31, unsigned int b = 17, unsigned int m = 1000) {

unsigned int hash = 0;

for (char c : key) {

hash = ((hash \* a + c) % p + b) % m;

}

return hash;

}

// Вставка элемента в хеш-таблицу

void insert(std::string key, int value) {

// Вычисляем хеш-код

unsigned int index = hash(key) % TABLE\_SIZE;

// Открытая адресация

while (!table[index].key.empty() && !table[index].is\_deleted) {

index = (index + 1) % TABLE\_SIZE;

}

// Вставляем элемент

table[index].key = key;

table[index].value = value;

table[index].is\_deleted = false;

}

// Получение значения элемента по ключу

int get(std::string key) {

unsigned int index = hash(key) % TABLE\_SIZE;

for (int i = 0; i < TABLE\_SIZE; ++i) {

if (table[index].key == key && !table[index].is\_deleted) {

return table[index].value;

}

index = (index + 1) % TABLE\_SIZE;

}

return -1; // Возвращаем -1, если элемент не найден

}

// Удаление элемента по ключу

void remove(std::string key) {

unsigned int index = hash(key) % TABLE\_SIZE;

for (int i = 0; i < TABLE\_SIZE; ++i) {

if (table[index].key == key && !table[index].is\_deleted) {

table[index].is\_deleted = true;

return;

}

index = (index + 1) % TABLE\_SIZE;

}

}

// Вывод хеш-таблицы

void print() {

for (int i = 0; i < TABLE\_SIZE; ++i) {

std::cout << i << ": ";

if (!table[i].key.empty() && !table[i].is\_deleted) {

std::cout << "[" << table[i].key << ", " << table[i].value << "] ";

}

std::cout << std::endl;

}

}

};

// Точка входа в программу

int main() {

setlocale(LC\_ALL, "Rus"); // Установка русской локали

HashTable table; // Создание хеш-таблицы

int choice = 0;

while (choice != 5) {

std::cout << "1. Вставить элемент в хеш-таблицу" << std::endl;

std::cout << "2. Найти элемент в хеш-таблице" << std::endl;

std::cout << "3. Удалить элемент из хеш-таблицы" << std::endl;

std::cout << "4. Распечатать хеш-таблицу" << std::endl;

std::cout << "5. Выйти из программы" << std::endl;

std::cout << "Введите номер команды: ";

std::cin >> choice;

switch (choice) {

case 1: {

std::string key;

std::cout << "Введите ключ: ";

std::cin >> key;

int value;

std::cout << "Введите значение: ";

std::cin >> value;

table.insert(key, value);

std::cout << "Элемент добавлен в хеш-таблицу" << std::endl;

break;

}

case 2: {

std::string key;

std::cout << "Введите ключ: ";

std::cin >> key;

int value = table.get(key);

if (value == -1) {

std::cout << "Элемент не найден" << std::endl;

}

else {

std::cout << "Значение элемента: " << value << std::endl;

}

break;

}

case 3: {

std::string key;

std::cout << "Введите ключ: ";

std::cin >> key;

table.remove(key);

std::cout << "Элемент удален из хеш-таблицы" << std::endl;

break;

}

case 4: {

std::cout << "Хеш-таблица: " << std::endl;

table.print();

break;

}

case 5: {

std::cout << "До свидания!" << std::endl;

break;

}

default: {

std::cout << "Некорректный выбор, попробуйте еще раз" << std::endl;

break;

}

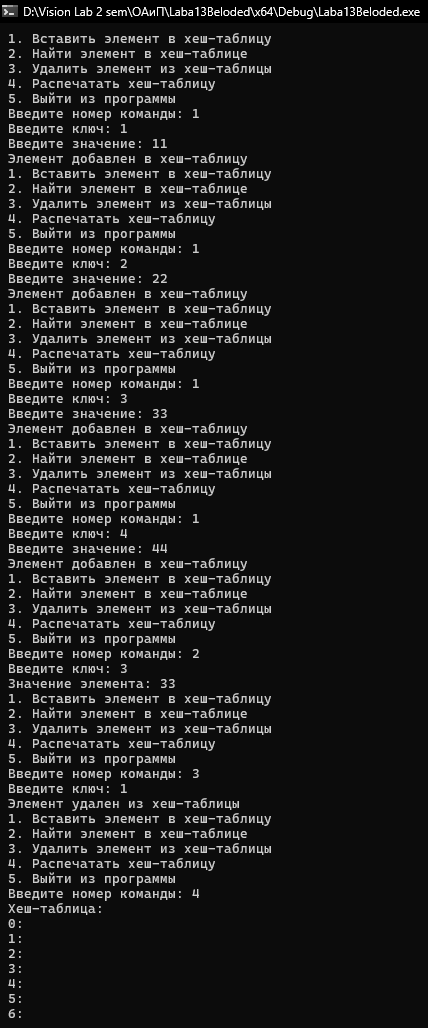
}

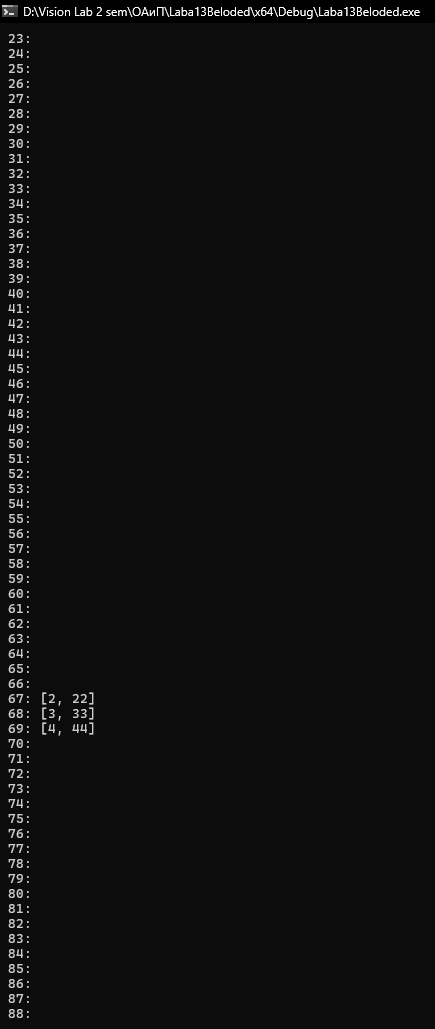
}

return 0;

}

**Результат выполнения:**

****



**Вариант 11:**



**Код программы:**

#include <iostream>

#include <string>

#include <cstdlib>

#include <vector>

const int TABLE\_SIZE = 128; // Размер хеш-таблицы

// Структура для узла хеш-таблицы

struct HashNode {

std::string key; // Ключ

int value; // Значение

bool is\_deleted; // Флаг удаления

// Конструктор

HashNode() : key(""), value(-1), is\_deleted(false) {}

};

// Класс хеш-таблицы

class HashTable {

private:

std::vector<HashNode> table; // Вектор узлов

public:

// Конструктор

HashTable() : table(TABLE\_SIZE) {}

// Хеш-функция

static unsigned int hash(const std::string& key, unsigned int p = 1000000007, unsigned int a = 31, unsigned int b = 17, unsigned int m = 1000) {

unsigned int hash = 0;

for (char c : key) {

hash = ((hash \* a + c) % p + b) % m;

}

return hash;

}

// Вставка элемента в хеш-таблицу

void insert(std::string key, int value) {

// Вычисляем хеш-код

unsigned int index = hash(key) % TABLE\_SIZE;

// Открытая адресация

while (!table[index].key.empty() && !table[index].is\_deleted) {

index = (index + 1) % TABLE\_SIZE;

}

// Вставляем элемент

table[index].key = key;

table[index].value = value;

table[index].is\_deleted = false;

}

// Получение значения элемента по ключу

int get(std::string key) {

unsigned int index = hash(key) % TABLE\_SIZE;

for (int i = 0; i < TABLE\_SIZE; ++i) {

if (table[index].key == key && !table[index].is\_deleted) {

return table[index].value;

}

index = (index + 1) % TABLE\_SIZE;

}

return -1; // Возвращаем -1, если элемент не найден

}

// Удаление элемента по ключу

void remove(std::string key) {

unsigned int index = hash(key) % TABLE\_SIZE;

for (int i = 0; i < TABLE\_SIZE; ++i) {

if (table[index].key == key && !table[index].is\_deleted) {

table[index].is\_deleted = true;

return;

}

index = (index + 1) % TABLE\_SIZE;

}

}

// Вывод хеш-таблицы

void print() {

for (int i = 0; i < TABLE\_SIZE; ++i) {

std::cout << i << ": ";

if (!table[i].key.empty() && !table[i].is\_deleted) {

std::cout << "[" << table[i].key << ", " << table[i].value << "] ";

}

std::cout << std::endl;

}

}

};

int main() {

setlocale(LC\_ALL, "Rus"); // Установка русской локали

HashTable table; // Создание хеш-таблицы

int choice = 0;

while (choice != 5) {

std::cout << "1. Вставить элемент в хеш-таблицу" << std::endl;

std::cout << "2. Найти элемент в хеш-таблице" << std::endl;

std::cout << "3. Удалить элемент из хеш-таблицы" << std::endl;

std::cout << "4. Распечатать хеш-таблицу" << std::endl;

std::cout << "5. Выйти из программы" << std::endl;

std::cout << "Введите номер команды: ";

std::cin >> choice;

switch (choice) {

case 1: {

std::string key;

std::cout << "Введите ключ: ";

std::cin >> key;

int value;

std::cout << "Введите значение: ";

std::cin >> value;

table.insert(key, value);

std::cout << "Элемент добавлен в хеш-таблицу" << std::endl;

break;

}

case 2: {

std::string key;

std::cout << "Введите ключ: ";

std::cin >> key;

int value = table.get(key);

if (value == -1) {

std::cout << "Элемент не найден" << std::endl;

}

else {

std::cout << "Значение элемента: " << value << std::endl;

}

break;

}

case 3: {

std::string key;

std::cout << "Введите ключ: ";

std::cin >> key;

table.remove(key);

std::cout << "Элемент удален из хеш-таблицы" << std::endl;

break;

}

case 4: {

std::cout << "Хеш-таблица: " << std::endl;

table.print();

break;

}

case 5: {

std::cout << "До свидания!" << std::endl;

break;

}

default: {

std::cout << "Некорректный выбор, попробуйте еще раз" << std::endl;

break;

}

}

}

return 0;

}

**Результат выполнения:**



