Министерство образования Республики Беларусь

Учреждение образования

«Белорусский государственный университет

информатики и радиоэлектроники»

Кафедра инженерной психологии и эргономики

­­­­­

Основы алгоритмизации и программирования

Отчет по лабораторной работе №15

«Хеширование»

Выполнил: Усов А.М.

Студент группы 310901

Преподаватель: Кабариха В. А.

Минск 2023

Цель: сформировать знания и умения по работе с подпрограммами, приобрести навыки написания программ с использованием хеш-функций.

Задание . Составить хеш-функцию в соответствии с заданным вариантом и

проанализировать ее. При необходимости доработать хеш-функцию.

Используя полученную хеш-функцию разработать на языке

программирования С++ программу, которая должна выполнять следующие

функции:

* создавать хеш-таблицу;
* добавлять элементы в хеш-таблицу;
* просматривать хеш-таблицу;
* искать элементы в хеш-таблице;
* удалять элементы из хеш-таблицы.

Описание хеш-функции

Хеш-функция основана на возведении суммы кодов символов ключа в

квадрат и извлечение из полученного квадрата нескольких средних цифр. При

этом коды символов умножаем на частное кода и произведения тройки на

порядковый номер символа в ключе (1ч6). Звучит убого, вот так выглядит

формула суммы:

где - код символа с индексом "i";

Возведенная в квадрат сумма колеблется от 7997584 до 22781529, а это

семизначное или восьмизначное число. Для адресации сегментов хештаблицы необходимо четырехзначное число, не превышающее 2000. Откинем

у квадрата суммы 2 первых и два последних разряда, так у нас получится

трехзначное или четырехзначное число. Для того, чтобы адрес не превысил

максимально допустимый адрес 1999, будем брать остаток от деления на 2000

до тех пор, пока он не попадет в нужный диапазон.

*Экспериментальный анализ хеш-функции*

Экспериментальное исследование проводится следующим образом:

формируются случайным образом ключи заданного формата в

количестве, превышающем количество сегментов хеш-таблицы в 2…3 раза;

для каждого сформированного ключа вычисляется хеш-функция, и

подсчитывается, сколько раз вычислялся адрес того или иного сегмента хештаблицы.

#include <iostream>

using namespace std;

class Node {

string data;

string key;

Node\* next;

public:

Node(string data) : data(data), next(nullptr) {}

Node(string data, string key) : data(data), key(key), next(nullptr) {}

string getData() { return data; }

string getKey() { return key; }

Node\* getNext() { return next; }

void setNext(Node\* next) { this->next = next; }

void setData(string data) { this->data = data; }

void setKey(string key) { this->key = key; }

};

class LinkedList {

Node\* head;

int size;

public:

LinkedList() : head(nullptr), size(0) {}

int getSize() { return size; }

void push(string data, string key) {

if (size == 0) {

head = new Node(data, key);

}

else {

Node\* current = head;

while (current->getNext() != nullptr) {

current = current->getNext();

}

current->setNext(new Node(data, key));

}

size++;

}

string get(string key) {

if (size == 0) {

cout << "List is empty";

return "";

}

else if (size == 1) {

return head->getData();

}

else {

Node\* current = head;

while (current) {

if (current->getKey() == key) {

return current->getData();

}

current = current->getNext();

}

cout << "Element not found";

return "";

}

}

Node\* get(int index) {

if (index < 0 || index >= size) {

cout << "Index out of range";

return nullptr;

}

else {

Node\* current = head;

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

current = current->getNext();

}

return current;

}

}

void deleteNode(string key) {

if (size == 0) {

cout << "List is empty";

return;

}

else if (size == 1) {

delete head;

head = nullptr;

size = 0;

return;

}

else {

Node\* current = head;

Node\* prev = nullptr;

while (current) {

if (current->getKey() == key) {

if (prev) {

prev->setNext(current->getNext());

}

else {

head = current->getNext();

}

delete current;

delete prev;

size--;

return;

}

prev = current;

current = current->getNext();

}

}

cout << "Element not found" << endl;

}

};

class HashTable {

LinkedList\* arr;

int size;

public:

HashTable(int size) : size(size) {

arr = new LinkedList[size];

}

void Add(string key, string data) {

int hash = getHash(key);

arr[hash].push(data, key);

}

int getHash(string data) {

int sum = 0;

for (int i = 0; i < data.length(); i++) {

sum += data[i] / 1;

}

return sum % size;

}

string get(string key) {

int hash = getHash(key);

return arr[hash].get(key);

}

void Delete(string key) {

int hash = getHash(key);

arr[hash].deleteNode(key);

}

void ViewHashTable() {

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

if (arr[i].getSize() != 0) {

for (int j = 0; j < arr[i].getSize(); j++) {

Node\* temp = arr[i].get(j);

cout << temp->getKey() << " \t:\t " << temp->getData() << endl;

}

}

}

}

};

int main() {

HashTable table(2000);

table.Add("123", "Hello");

table.Add("Petya", "Exlent");

table.Add("Vasya", "Good");

table.Add("Kolya", "Bad");

table.Add("Masha", "Nice");

table.ViewHashTable();

cout << "==========================" << endl;

table.Delete("Petya");

table.Delete("Vasya");

table.ViewHashTable();

cout << "==========================" << endl;

cout << table.get("123") << endl;

return 0;

}

# Результат работы программы представлен на рисунке 1.

# 

Рисунок 1 – Результат выполнения программы