Лабораторная работа 13

По дисциплине «Основы алгоритмизации и программирования»

На тему «Хеш-таблицы c открытой адресацией»

***Задание#1***



*Macros.h*

#include <iostream>

#include <vector>

#include <ctime>

#include <cmath>

#include <iomanip>

#include <stack>

#include <map>

#include <cstring>

#include <string>

#include <chrono>

#include <fstream>

#include <Windows.h>

#include <queue>

#define vi cout <<

#define vv cin >>

#define nl cout << "\n";

#define nw cout << "\t";

#define SCOCP1251 SetConsoleOutputCP(1251);

#define SCCP1251 SetConsoleCP(1251);

#define ret return

#define pause system("pause");

using namespace :: std;

*Hash.h*

#pragma once

#include "Macros.h"

#define HASHDEL (void\*) -1 // Макрос для определения значения удаленного элемента

// Структура объекта

struct Object {

void\*\* data; // Указатель на массив указателей на данные

int size; // Размер массива

int N; // Количество элементов в массиве

int(\*getKey)(void\*); // Указатель на функцию получения ключа из данных

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

bool insert(void\*); // Функция вставки элемента в массив

int searchInd(int key); // Функция поиска индекса элемента по ключу

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

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

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

void scan(void(\*f)(void\*)); // Функция сканирования элементов с применением функции к каждому

};

static void\* DEL = (void\*)HASHDEL; // Статическая переменная для обозначения удаленного элемента

Object create(int size, int(\*getkey)(void\*)); // Функция создания объекта

#undef HASHDEL // Удаление макроса для значения удаленного элемента

*Hash.cpp*

#include "Hash.h"

#include "Macros.h"

// Функция хеширования

int HashFunction(int key, int size) // Мультипликативная хеш-функция

{

double A = (sqrt(5) - 1) / 2; // Константа Кнута

double hashValue = size \* (key \* A - int(key \* A)); // Расчет значения хеша

ret static\_cast<int>(hashValue); // Возвращаем значение хеша в виде целого числа

}

// Функция для расчета следующего значения хеша

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

{

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

}

// Создание хеш-таблицы заданного размера

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

{

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

}

// Конструктор 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) // Инициализация таблицы пустыми ячейками

data[i] = NULL;

}

// Вставка элемента в таблицу

bool Object::insert(void\* d)

{

bool b = false;

if (N != size)

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

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

if (data[j] == NULL || data[j] == DEL) // Если ячейка пуста или удалена

{

data[j] = d; // Вставляем элемент

N++; // Увеличиваем количество заполненных ячеек

b = true;

}

ret b;

}

// Поиск ключа в таблице

int Object::searchInd(int key)

{

int t = -1;

bool b = false;

if (N != 0)

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

!= size && !b; j = HashFunction(key, size))

if (data[j] != DEL)

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

{

t = j; b = true;

}

ret t;

}

// Поиск элемента в таблице

void\* Object::search(int key)

{

int t = searchInd(key);

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

}

// Удаление элемента из таблицы по ключу

void\* Object::deleteByKey(int key)

{

int i = searchInd(key);

void\* t = data[i];

if (t != NULL)

{

data[i] = DEL;

N--;

}

ret t;

}

// Удаление элемента из таблицы по значению

bool Object::deleteByValue(void\* d)

{

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

}

// Проверка состояния ячейки (заполнена, пуста, удалена)

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

{

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

{

vi " Элемент" << i;

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

{

vi " пусто"; nl

}

else

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

{

vi " удален"; nl

}

else

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

}

}

*Task#1.cpp*

#include "Hash.h"

#include "Macros.h"

struct AAA

{

int key;

char\* mas;

AAA(int k, char\* z)

{

key = k; mas = z;

} AAA() {}

};

//Создание ключа

int key(void\* d)

{

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

ret f->key;

}

//Вывод ключа

void AAA\_print(void\* d)

{

vi " ключ " << ((AAA\*)d)->key << " - " << ((AAA\*)d)->mas; nl

}

//Главная фукнция с меню

int main()

{

SCOCP1251

SCCP1251

int siz = 10, choice, k;

vi "Введите размер хеш-таблицы"; nl

vv siz;

Object H = create(siz, key);

for (;;)

{

vi "1 - вывод хеш-таблицы"; nl

vi "2 - добавление элемента"; nl

vi "3 - удаление элемента"; nl

vi "4 - поиск элемента"; nl

vi "0 - выход"; nl

vi "сделайте выбор"; nl vv choice;

switch (choice)

{

case 0: exit(0);

case 1: H.scan(AAA\_print); break;

case 2: {

AAA\* a = new AAA;

char\* str = new char[20];

vi "введите ключ"; nl

vv k;

a->key = k; //Добавляем ключ в таблицу

vi "введите строку"; nl

vv str;

a->mas = str; //Добавляем строку по клч

if (H.N == H.size) //Проверяем, заполнена ли таблица

{

vi "Таблица заполнена"; nl

}

else //Иначе вставляем в нее ключ-элемент

H.insert(a);

} break;

case 3: {

vi "введите ключ для удаления"; nl

vv k;

H.deleteByKey(k);

} break;

case 4: {

vi "введите ключ для поиска"; nl

vv k;

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

{

vi "Элемент не найден"; nl

}

else

AAA\_print(H.search(k));

} break;

}

}

pause

ret 0;

}

 