Беларускі дзяржаўны тэхналагічны ўніверсітэт

Факультэт інфармацыйных тэхналогій

Кафедра праграмнай інжэнерыі

Лабараторная работа №13

Па дысцыпліне «Асновы алгарытмізацыі і праграмавання»

На тэму «Хэш-табліцы»

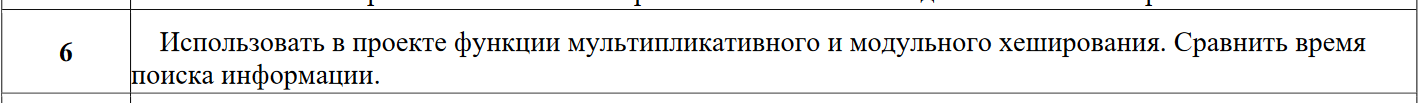
Выканала:

Студэнтка 1 курса 6 группы

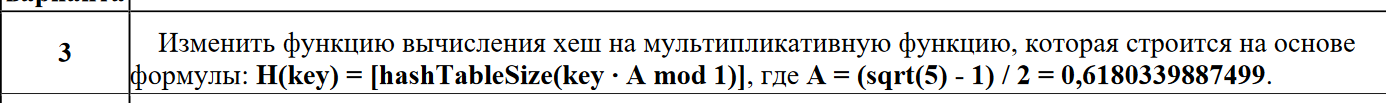
Жучкевіч Кацярына Сяргееўна

Выкладчык: асс. Андронава М.В.

Мінск, 2024



**+ 3 варыянт**

****

**Hash.h:**

#pragma once

#include <iostream>

#include <Windows.h>

#include <chrono>

#define HASHDEL (void\*) -1

struct Object //структура для хэш табліцы

{

void\*\* data;

int size;

int N;

int(\*getKey)(void\*);

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

bool insert(void\*);

int searchIndByMOD(int key);

int searchIndByMULT(int key);

void\* searchMOD(int key);

void\* searchMULT(int key);

void\* deleteByKey(int key);

bool deleteByValue(void\*);

void scan(void(\*f)(void\*));

};

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

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

#undef HASHDEL

**Hash.cpp:**

#include "Hash.h"

int HashFunctionMult(int key, int size, int p) { // Функцыя вылічэнні хэша з выкарыстаннем мультыплікатыўнага метада

double A = 0.6180339887499;

double hashValue = key \* A;

hashValue -= int(hashValue);

return (p + int(size \* hashValue)) % size; //павяртаем канчатковае значэнне хэша

}

//ГЭТА АДРАЗУ І 3 ВАРЫЯНТ

int HashFunctionMod(int key, int size, int p) //модульная хэш функцыя

{

return (p + key) % size;

}

int Next\_hash(int hash, int size, int p) //для павелічэння хэша

{

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

}

Object create(int size, int(\*getkey)(void\*))//функцыя для стварэння новага аб'екта

{

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

}

Object::Object(int size, int(\*getkey)(void\*)) //а тут ствараем сам аб'ект

{

N = 0;

this->size = size;

this->getKey = getkey;

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

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

data[i] = NULL;

}

bool Object::insert(void\* d) // функцыя для таго, каб уставіць элемент

{

bool b = false;

if (N != size)

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

i != size && !b; j = Next\_hash(j, size, ++i)) //шукаем месца для аб'екта

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

data[j] = d;

N++;

b = true;

}

return b; //вяртаем вынік

}

int Object::searchIndByMOD(int key) { // Функцыя для пошуку індэкса ў хэш-табліцы з дапамогай модульнага хэшыравання

int t = -1;

bool b = false;

if (N != 0)

for (int i = 0, j = HashFunctionMod(key, size, 0); data[j] != NULL && i != size && !b; j = HashFunctionMod(key, size, ++i)) //цыкл для пошуку

if (data[j] != DEL)

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

t = j; //захоўваем індэкс

b = true;

}

return t; // вяртаем

}

int Object::searchIndByMULT(int key) { // Функцыя для пошуку індэкса ў хэш-табліцы з дапамогай мультыплікатыўнага хэшыравання

int t = -1;

bool b = false;

if (N != 0)

for (int i = 0, j = HashFunctionMult(key, size, 0); data[j] != NULL && i != size && !b; j = HashFunctionMult(key, size, ++i)) // Цыкл для пошуку

if (data[j] != DEL)

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

t = j; // захоўваем індэкс

b = true;

}

return t; //вяртаем

}

void\* Object::searchMOD(int key) // функцыя для пошуку з дапамогай мод хэшыравання

{

int t = searchIndByMOD(key);

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

}

void\* Object::searchMULT(int key)// функцыя для пошуку з дапамогай мульт хэшыравання

{

int t = searchIndByMULT(key);

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

}

// Функцыя для выдалення элемента з хэш-табліцы па індэксу

void\* Object::deleteByKey(int key)

{

int i = searchIndByMOD(key);

void\* t = data[i];

if (t != NULL)

{

data[i] = DEL;

N--;

}

return t;

}

// Функцыя для выдалення элемента з хэш-табліцы па значэнні

bool Object::deleteByValue(void\* d)

{

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

}

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

{

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

{

std::cout << "Элемент" << i << '\t';

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

std::cout << "пуста" << std::endl;

else

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

std::cout << "выдалены" << std::endl;

else

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

}

}

**1.cpp**

#include "Hash.h"

using namespace std::chrono;

using namespace std;

struct AAA//структура для прадстаўлення элементаў хэш-табліцы.

{

int key;

char\* mas;

AAA(int k, char\* z)

{

key = k;

mas = z;

}

AAA() {}

};

int key(void\* d)//функцыя для атрымання ключа элемента AAA.

{

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

return f->key;

}

void AAA\_print(void\* d) // функцыя для вываду элемента

{

std::cout << " ключ " << ((AAA\*)d)->key << " - " << ((AAA\*)d)->mas << std::endl;

}

int main()

{

SetConsoleOutputCP(1251);

SetConsoleCP(1251);

int siz = 10, choice, k;

cout << "Увядзіце памер хэш-табліцы" << endl;

cin >> siz;

Object H = create(siz, key);

for (;;)

{

cout << "1 - вывад хэш-табліцы" << endl;

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

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

cout << "4 - пошук элемента модульным метадам" << endl;

cout << "5 - пошук элемента мультыплікатыйным метадам" << endl;

cout << "0 - выхад" << endl;

cout << "зрабіце выбар" << endl; cin >> 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];

cout << "увядзіце ключ" << endl;

cin >> k;

a->key = k;

cout << "увядзіце радок" << endl;

cin >> str;

a->mas = str;

if (H.N == H.size)

cout << "Табліца запоўнена" << endl;

else

H.insert(a);

} break;

case 3: {

cout << "увядзіце ключ для выдалення" << endl;

cin >> k;

H.deleteByKey(k);

} break;

case 4: {

cout << "увядзіце ключ для пошуку" << endl;

cin >> k;

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

cout << "Элемент не знойдзены" << endl;

else {

auto start = high\_resolution\_clock::now();

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

auto end = high\_resolution\_clock::now();

double time = duration\_cast<microseconds>(end - start).count();

cout << "\nЧас: " << time;

}

} break;

case 5: {

cout << "увядзіце ключ для пошуку" << endl;

cin >> k;

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

cout << "Элемент не знойдзены" << endl;

else {

auto start = high\_resolution\_clock::now();

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

auto end = high\_resolution\_clock::now();

double time = duration\_cast<microseconds>(end - start).count();

cout << "\nЧас: " << time;

}

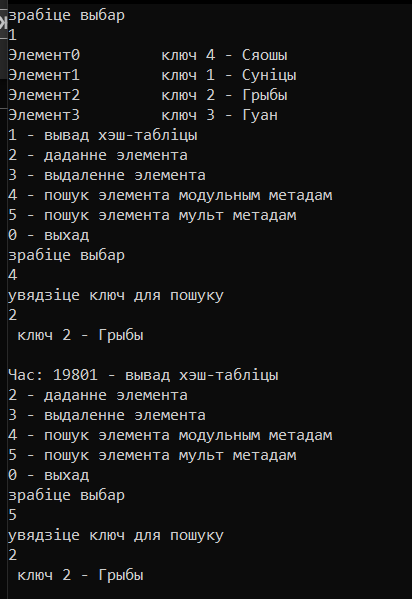
} break;

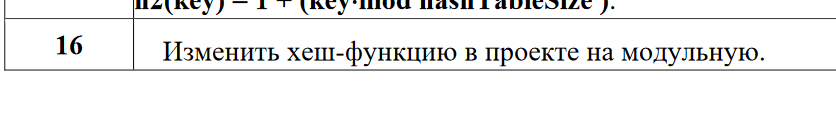
}

}

return 0;

}

**** (час крывы, таму што ад’ядноўваецца з пунктам, я толькі потым гэта заўважыла(( )



Hash.cpp

#include "Hash.h"

int HashFunctionMod(int key, int size, int p) //модульная хэш функцыя

{

return (p + key) % size;

}

Object::Object(int size, int(\*getkey)(void\*)) //а тут ствараем сам аб'ект

{

N = 0;

this->size = size;

this->getKey = getkey;

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

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

data[i] = NULL;

}

bool Object::insert(void\* d) // функцыя для таго, каб уставіць элемент

{

bool b = false;

if (N != size)

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

i != size && !b; j = Next\_hash(j, size, ++i)) //шукаем месца для аб'екта

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

data[j] = d;

N++;

b = true;

}

return b; //вяртаем вынік

}

int Object::searchIndByMOD(int key) { // Функцыя для пошуку індэкса ў хэш-табліцы з дапамогай модульнага хэшыравання

int t = -1;

bool b = false;

if (N != 0)

for (int i = 0, j = HashFunctionMod(key, size, 0); data[j] != NULL && i != size && !b; j = HashFunctionMod(key, size, ++i)) //цыкл для пошуку

if (data[j] != DEL)

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

t = j; //захоўваем індэкс

b = true;

}

return t; // вяртаем

}

void\* Object::searchMOD(int key) // функцыя для пошуку з дапамогай мод хэшыравання

{

int t = searchIndByMOD(key);

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

}

// Функцыя для выдалення элемента з хэш-табліцы па індэксу

void\* Object::deleteByKey(int key)

{

int i = searchIndByMOD(key);

void\* t = data[i];

if (t != NULL)

{

data[i] = DEL;

N--;

}

return t;

}

// Функцыя для выдалення элемента з хэш-табліцы па значэнні

bool Object::deleteByValue(void\* d)

{

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

}

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

{

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

{

std::cout << "Элемент" << i << '\t';

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

std::cout << "пуста" << std::endl;

else

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

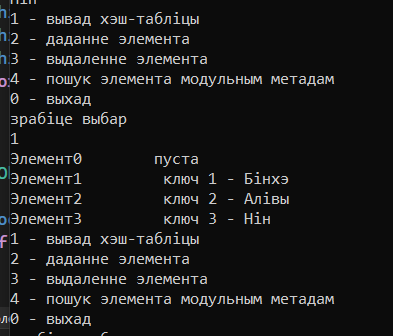
std::cout << "выдалены" << std::endl;

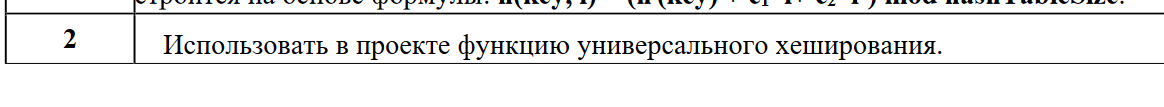
else

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

}

}





int HashUni(int key, int size, int p) //модульная хэш функцыя

{

const int p = 52;

int hashVal = 0;

int a = rand() % (p - 1) + 1;

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

int b = (key >> (8 \* i)) & 0xFF;

hashVal = (hashVal + (b \* a)) % p;

}

return hashVal % size;

}