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 8 группы

Лужецкий Владислав Константинович

Преподаватель: асс. Андронова М.В.

2024, Минск

Оглавление

[Основное задание(Вариант 6) 2](#_Toc165466615)

[Дополнительное задание 1(Вариант 7) 5](#_Toc165466616)

[Дополнительное задание 2(Вариант 1) 6](#_Toc165466617)

[Дополнительное задание 3(Вариант 15) 7](#_Toc165466618)

# Основное задание(Вариант 6)



#include "Hash.h"

#include <iostream>

int HashFunction(int key, int size, int p) //Хеш-функция

{

double A = 0.6180339887499; // Константа A для мультипликативного хеширования

double hash = size \* (key \* A - int(key \* A));

return (p + key) % size;

}

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

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

{

int A = 5; // Константа A для модульного хеширования

int B = 3; // Константа B для модульного хеширования

return (hash + A \* p + B \* p \* p) % size;

}

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

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

{

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

}

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

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;

}

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

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;

}

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

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;

}

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

void\* Object::search(int key)

{

int t = searchInd(key);

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

}

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

void\* Object::deleteByKey(int key)

{

int i = searchInd(key);

void\* t = data[i];

if (t != NULL)

{

data[i] = DEL;

N--;

}

return t;

}

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

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;

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

std::cout << " пусто" << std::endl;

else

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

std::cout << " удален" << std::endl;

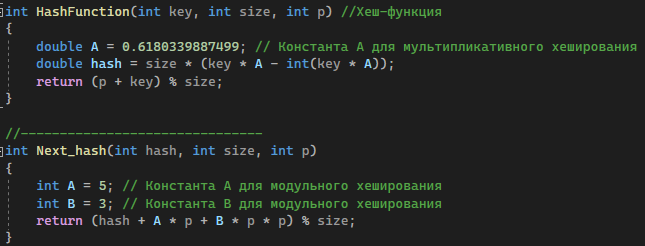
else

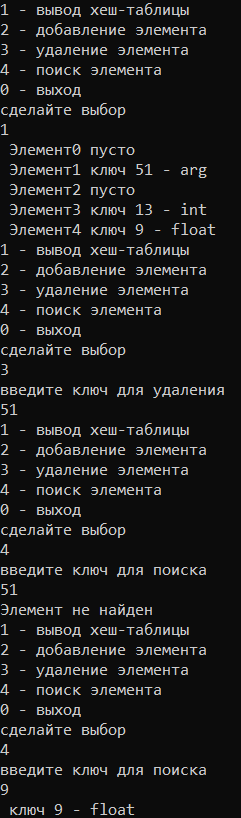
f((this->data)[i]);

}

}

//Это весь код программы. Дальше я буду кидать отдельные фрагменты кода, которые нужны для конкретного дополнительного задания. Ниже 6 вариант

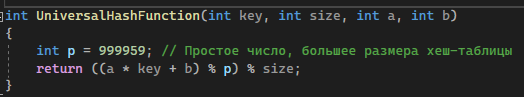


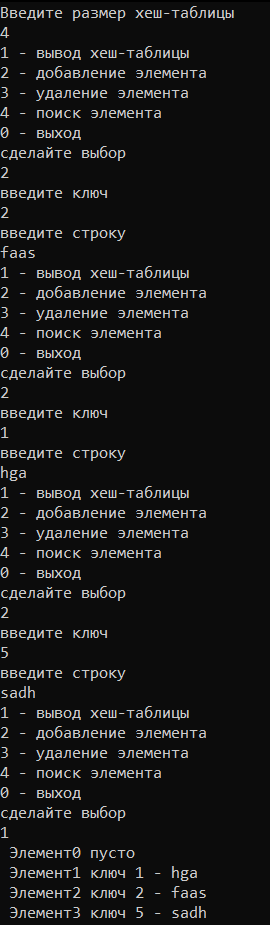


# Дополнительное задание 1(Вариант 7)



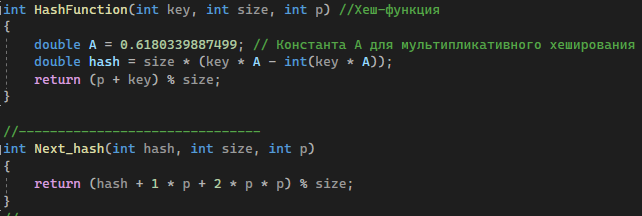
//Заменяем HashFunction и Next\_hash для более равномерного распределения ключей и снижения коллизий.

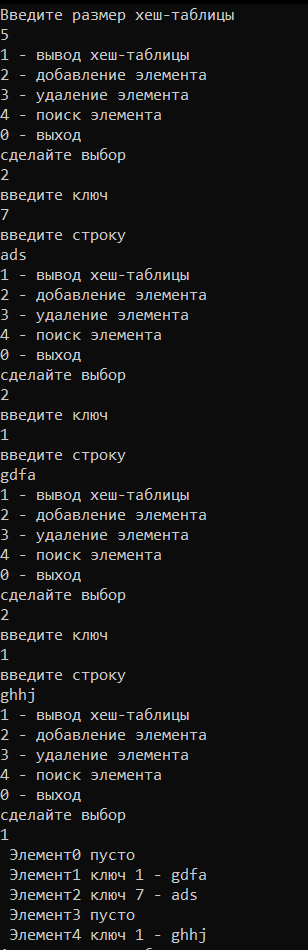




# Дополнительное задание 2(Вариант 1)







# Дополнительное задание 3(Вариант 15)

