Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра автоматизированных систем управления (АСУ)

**Хеш-таблицы**

**Отчет по лабораторной работе №5**

**По дисциплине**

**«Структуры и алгоритмы обработки данных в ЭВМ»**

Выполнил: студент гр. 439-1

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Зозуля Е.Д.

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_ 2021 г.

Проверил: ассистент каф. АСУ

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Яблонский Я.В.

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_ 2021 г.

Томск 2020

# Задание на лабораторную работу

Вариант 7.

# Написать программу, которая реализует метод открытого хеширования с хеш-функцией, основанной на методе деления с остатком. Данные, хранящиеся в файле занести в хеш-таблицу. Файл должен содержать не менее 15 целых чисел. Вывести построенную хеш-таблицу на экран (вместе с количеством выполненных проб). Организовать поиск данных в хеш-таблице. Результаты поиска данных вывести на экран. Также вывести количество проб, которые были затрачены при поиске.

# Алгоритм решения задачи

Выполняется обход с поиском элемента Е, где одновременно с поиском элемента ведётся подсчёт количества проб.

# Листинг программы

main.cpp

#include <iostream>

#include "HashTable.h"

#include <fstream>

#include <ctime>

using namespace std;

using namespace Mine;

int main()

{

setlocale(LC\_ALL, "RUS");

srand(static\_cast<unsigned int>(time(0)));

HashTable a;

ofstream Gen;

Gen.open("Input.txt");

for (int i = 0; i < 15; i++)

{

Gen << rand() % 100;

Gen << "\n";

}

Gen.close();

ifstream FileIn("Input.txt");

int rNum;

while (!FileIn.eof())

{

FileIn >> rNum;

a.push(rNum);

}

///считали числа из файла в хеш таблицу

a.print();

int searchValue;

cin >> searchValue;

a.itemSearch(searchValue);

cin >> searchValue;

a.itemSearch(searchValue);

return 0;

}

HashTable.h

#include "List.h"

namespace Mine

{

class HashTable

{

public:

HashTable();

void push (int value);

int itemSearch (int key) const;

void delItem (int key);

void print() const;

private:

int Hash(int value) const;

static const int LENGHT = 10;

List arr[LENGHT];

};

}

BinaryTree.cpp

#include "HashTable.h"

#include <iostream>

#include <cmath>

using namespace std;

namespace Mine

{

HashTable::HashTable()

{

}

void HashTable::push(int value)

{

arr[Hash(value)].insertionInEnd(value);

}

int HashTable::itemSearch (int key) const

{

cout<<"Ищем "<<key<<":";

int numOfTry=0;

int index = Hash(key);

ListNode \*curHead=arr[index].getHead();

if(curHead==NULL)

{

cout<<"ОШИБКА: искомого числа в таблице нет!\n";

throw "Искомого числа в таблице нет";

}

while ((curHead->next!=NULL)&&(curHead->info!=key))

{

numOfTry++;

curHead = curHead->next;

}

if(curHead->info!=key)

{

cout<<"ОШИБКА: искомого числа в таблице нет!\n";

throw "Искомого числа в таблице нет";

}

cout<<"\nИндексы:"<<index<<"; "<<numOfTry<<"\nКоличество проб:"<<numOfTry+1<<endl;

return index;

}

void HashTable::delItem (int key)

{

int secIndex=0;

int index = Hash(key);

ListNode \*curHead=arr[index].getHead();

if(curHead==NULL)

{

cout<<"ОШИБКА: удаляемого числа в таблице нет!\n";

throw "Удаляемого числа в таблице нет";

}

while ((curHead->next!=NULL)&&(curHead->info!=key))

{

secIndex++;

curHead = curHead->next;

}

if(curHead->info!=key)

{

cout<<"ОШИБКА: удаляемого числа в таблице нет!\n";

throw "Удаляемого числа в таблице нет";

}

arr[index].removeByIndex(secIndex);

}

void HashTable::print() const

{

int numOfTry=0;

ListNode \*curHead;

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

{

curHead = arr[i].getHead();

while (curHead!=NULL)

{

numOfTry++;

cout<<curHead->info<<"\t";

curHead=curHead->next;

}

cout<<endl;

}

cout<<"\nКоличество проб при печати: "<<numOfTry<<endl;

}

int HashTable::Hash(int value) const

{

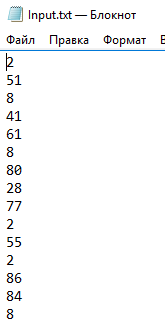
return value%10;

}

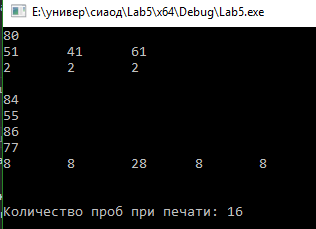
# }

# Пример решения

Скриншоты успешной работы программы, входные данные тестового примера, выходные данные на рисунке 4.1 и 4.2.



*Рисунок 4.1.- Входные данные*



*Рисунок 4.2.- Сформированная таблица*

# Вывод

Было изучено АТД Хеш-таблица и использовано для решения задачи.