**“ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ**

**УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ” (ТУСУР)**

Кафедра автоматизированных систем управления (АСУ)

Отчет по лабораторной работе №1

«Хэш-таблицы»

По дисциплине СиАОДвЭВМ

Выполнил студент гр. 438-3:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Канаев О.А.

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2020 г.

Проверил

Доктор технических наук,

доцент каф. АСУ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Горитов А.Н

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2020 г.

Томск 2020

**Цель работы:**

Получить первичные навыки методов обработки абстрактного типа данных: хэш-таблицы.

**Задание:**

7. Создать файл из 15 целых чисел. Написать программу, которая реализует метод открытого хеширования с хеш-функцией, основанной на методе деления с остатком. Занести данные, хранящиеся в файле в хеш-таблицу. Вывести построенную хеш-таблицу на экран. Организовать поиск данных в хеш-таблице. Результаты поиска данных вывести на экран. Также вывести количество проб, которые были затрачены при поиске.

**Теория:**

**Хеш-таблицей** называется структура данных, предназначенная для реализации ассоциативного массива, такого в котором адресация реализуется посредством хеш-функции. **Хеш-функция** – это функция, преобразующая ключ key в некоторый индекс i равный h(key), где h(key) – хеш-код key. Весь процесс получения индексов хеш-таблицы называется хешированием.

**Хэш-функции бывают:**

* Метод деления с остатком h(key) = key mod M, где M- размер таблицы
* Метод умножения h(key) = M(key\*A mod 1), где А ∈(0,1)
* Преобразование строки символов в целое число

**Требования к хэш-таблице:** 1) Легко вычисляемая; 2) Равномерное распределяется по таблице.

**Способы построения:** Открытое и Закрытое хеширование.

**Алгоритм:**

1. Создать хэш-таблицу
2. Считать из файла число
3. Определить хэш-код с помощью хеш-функции от этого числа
4. Занести число в таблицу
   1. Если ячейка таблицы свободна, то инициализировать линейный односвязный список.
   2. Иначе добавить в конец списка число (коллизия)
5. Найти требуемое число из таблицы

**Решение:**

#include <iostream>

#include <fstream>

#include <cstdio>

#include <iomanip>

#define ELEM\_NUM 15 //Количество элементов хэш-таблицы (задано индивидуально)

using namespace std;

struct list { //Структура списка

int data;

struct list \*ptr;

};

struct list \* addelem(list \*lst, int number); //Функция добавления элемента в список lst с определенной data = number

int hashf(int num); //Хэш-функция (по карточке)

void hesh\_search(struct list \*hash\_table); //Функция поиска числа в хэш-таблице

int main() {

setlocale(LC\_ALL, "Rus");

list hash\_table[ELEM\_NUM];

for (int i = 0; i < ELEM\_NUM; i++) {

hash\_table[i].data = 0;

hash\_table[i].ptr = NULL;

}

int num; //Считываемое число

int hash; //Хэш значение

ifstream input("input.txt");

if (!input.is\_open()) {

cout << "Ошибка открытия файла";

}

else

{

while (input.eof() != true)

{

input >> num;

hash = hashf(num);

if (hash\_table[hash].data == 0) { //Если список пуст, то пишем data

hash\_table[hash].data = num;

}

else { //Иначе добавляем элемент в конец списка

addelem(&hash\_table[hash], num);

}

}

input.close();

}

cout << "#hash " << setw(6) << left << "#\_1" //Аккуратный вывод хэш-таблицы

<< setw(6) << "#\_2"

<< setw(6) << "#\_3"

<< endl;

list \*lptr;

for (int i = 0; i < ELEM\_NUM; i++)

{

lptr = &hash\_table[i];

cout << setw(2) << i << "-> ";

if (lptr->ptr == NULL) cout << setw(6) << lptr->data << endl;

else {

while (lptr->ptr != NULL) {

cout << setw(6) << lptr->data;

lptr = lptr->ptr;

}

cout << setw(6) << lptr->data << endl;

}

}

hesh\_search(hash\_table); //Вызов функции поиска элемента

return 0;

}

struct list \* addelem(list \*lst, int number)

{

struct list \*temp, \*p = lst;

temp = (struct list\*)malloc(sizeof(list));

while (p->ptr != NULL) {

p = p->ptr;

}

p->ptr = temp;

temp->data = number;

temp->ptr = NULL;

return(temp);

}

int hashf(int num) {

return (num % ELEM\_NUM);

}

void hesh\_search(struct list \*hash\_table) {

int key, hash, count=0;

struct list \*lptr= hash\_table;

cout << "Введите число, которое котите найти: ";

cin >> key;

hash = hashf(key);

while (lptr[hash].data != key)

{

if (lptr[hash].ptr != NULL) lptr[hash] = \*lptr[hash].ptr;

else {

cout << "Введенного числа не существует!\n";

return;

}

count++;

}

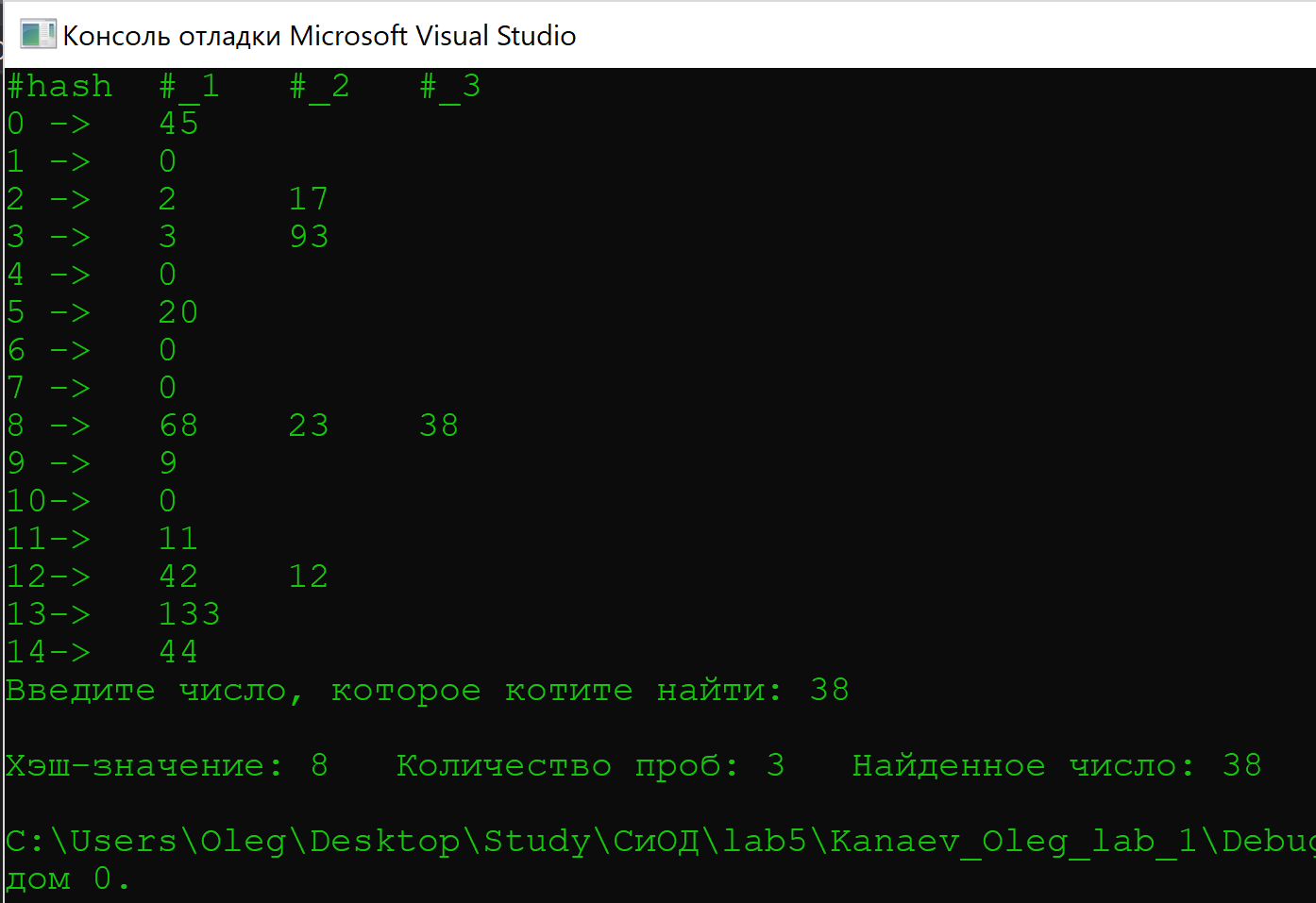
count++;

cout << "\nХэш-значение: " << hash << " Количество проб: " << count << " Найденное число: " << lptr[hash].data << endl;

return;

}

**Результат работы:**



**Вывод:**Научился использовать абстрактный тип данных, хэш-таблицы, с учетом плюсов и минусов. Плюсы открытого хеширования состоят в том, что количество элементов не ограниченно, а минусы: нужна дополнительная структура- список, нужно двигаться по всему списку при нахождении элемента