ГУАП

КАФЕДРА № 43

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| старший преподаватель |  |  |  |  |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ  «ЛИНЕЙНЫЕ И ЦИКЛИЧЕСКИЕ СПИСКИ» |
|  |
| по дисциплине: СТРУКТУРЫ И АЛГОРИТМЫ ОБРАБОТКИ ДАННЫХ |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | 4131 |  |  |  | В.А. Алексеев |
|  |  |  | подпись, дата |  | инициалы, фамилия |

* 1. Цель работы

Целью работы является изучение методов хеширования данных и получение практических навыков реализации хеш-таблиц.

* 1. Задание на лабораторную работу

Составить хеш-функцию в соответствии с заданным вариантом и проанализировать ее. При необходимости доработать хеш-функцию. Используя полученную хеш-функцию разработать на языке программирования высокого уровня программу, которая должна выполнять следующие функции:

− создавать хеш-таблицу;

− добавлять элементы в хеш-таблицу;

− просматривать хеш-таблицу;

− искать элементы в хеш-таблице по номеру сегмента/по ключу;

− выгружать содержимое хеш-таблицы в файл для построения гистограммы в MS Excel, или в аналогичном подходящем ПО;

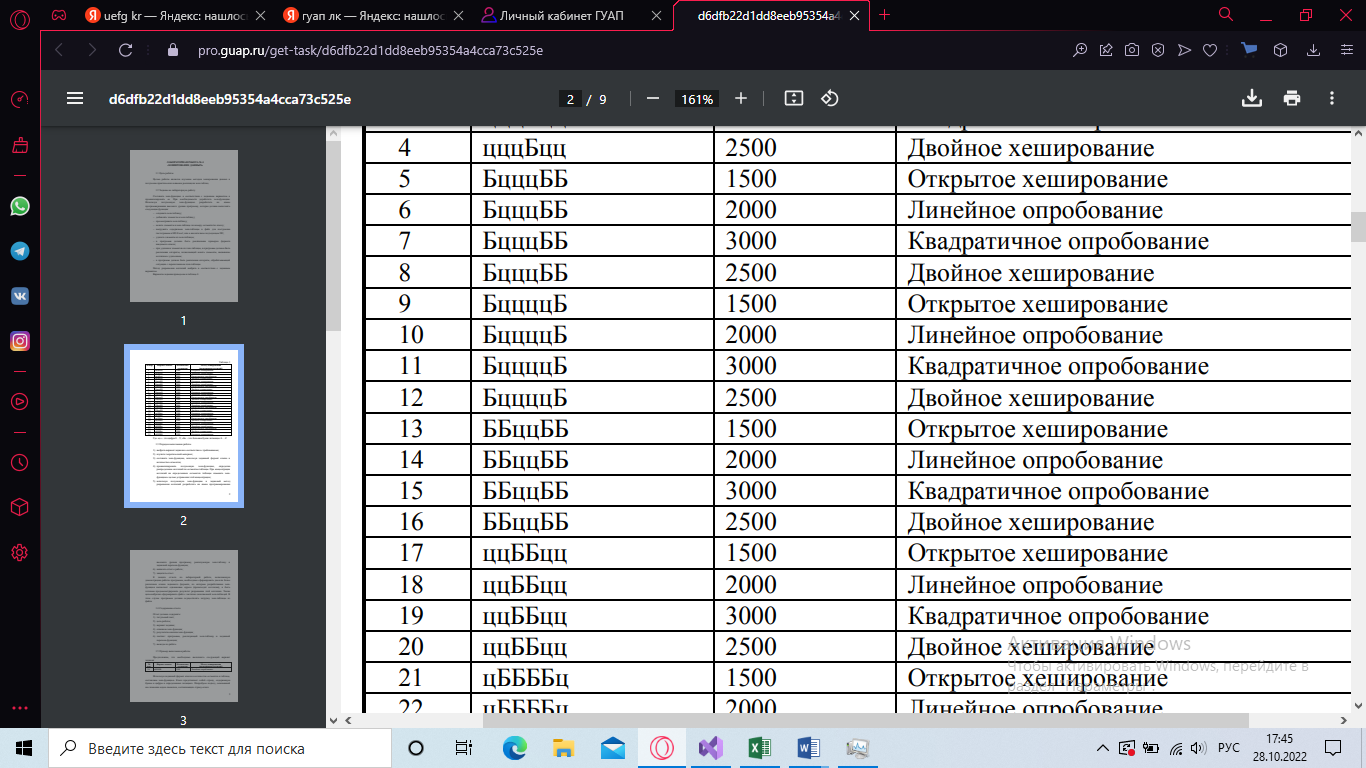
− удалять элементы из хеш-таблицы;

− в программе должна быть реализована проверка формата вводимого ключа;

− при удалении элементов из хэш-таблицы, в программе должен быть реализован алгоритм, позволяющий искать элементы, вызвавшие коллизию с удаленным;

− в программе должен быть реализован алгоритм, обрабатывающий ситуации с переполнением хэш-таблицы.

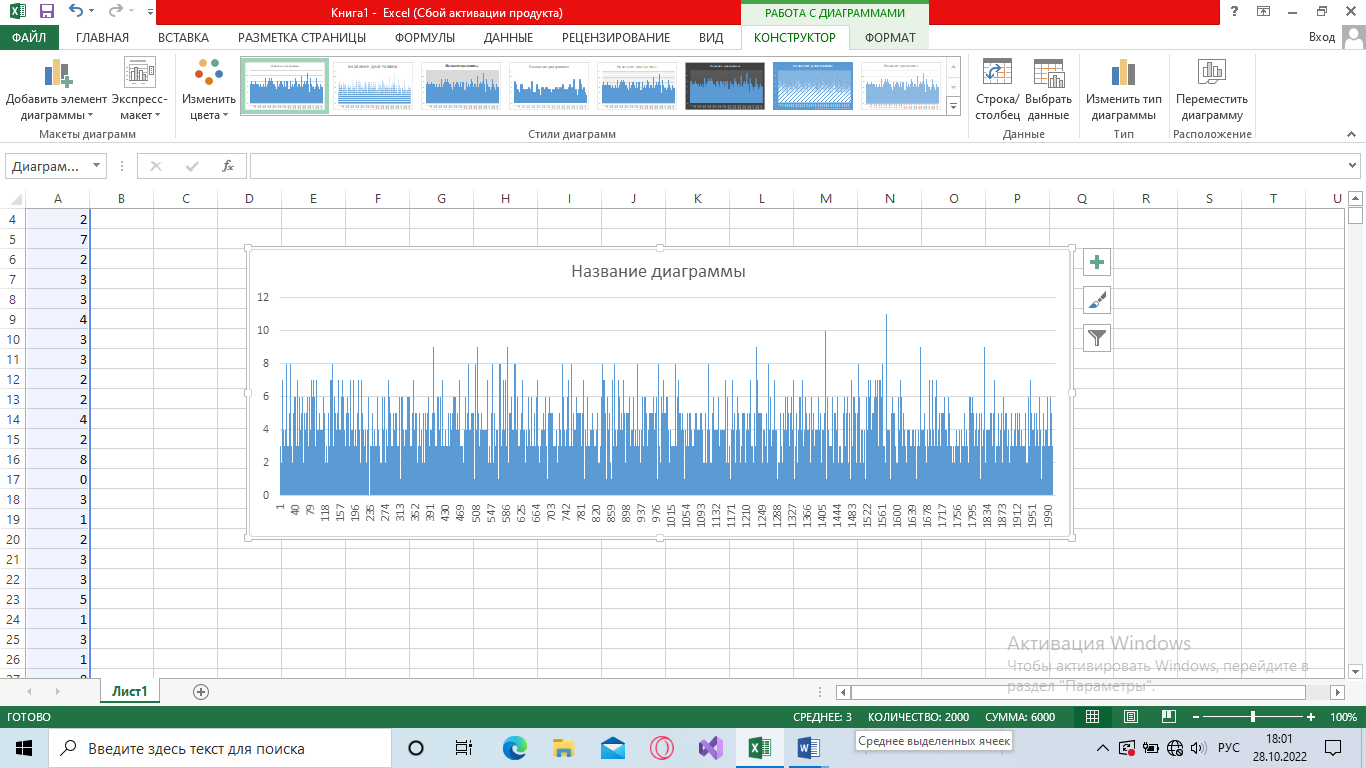
Вариант.



* + 1. Хеш функция

Создано 6000 ключей на 2000 сегментов хештаблицы

h = (c[0] \* c[0]\*7 + c[1] \* c[1]\*19 + c[2] \*c[2]\*2 + c[3] \* c[3] + c[4]\*c[4]\*13 + 17\*c[5] \* c[5]) % 2000;



* + 1. Листинг

#include<iostream>

#include<ctime>

#include<string>

using namespace std;

struct record {

string key = { "" };

int flag = 0;

int data;

};

int func(char k[]) {

int c[6];

for (int i = 0; i < 6; i++)

c[i] = int(k[i]);

return (c[0] \* c[0] \* 7 + c[1] \* c[1] \* 19 + c[2] \* c[2] \* 2 + c[3] \* c[3] + c[4] \* c[4] \* 13 + 17 \* c[5] \* c[5]) % 2000;

}

int create(record\* a, int sz) {

int ad[2000] = { 0 };

int bdk = 0;

srand(time(nullptr));

string\* keys = new string[2000];

for (int i = 0; i < 2000; i++) {

int f = 0;

int h;

char k[7] = { " " };

while (f == 0) {

k[0] = char((65 + rand() % (26)));

k[1] = char((65 + rand() % (26)));

k[2] = char((48 + rand() % (10)));

k[3] = char((48 + rand() % (10)));

k[4] = char((65 + rand() % (26)));

k[5] = char((65 + rand() % (26)));

h = func(k);

k[6] = '\0';

keys[i] = k;

if (i != 0) {

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

if (keys[j] == keys[i]) {

f = 0;

break;

}

else {

f = 1;

}

}

}

else {

f = 1;

}

}

int l = 0;

f = 1;

while (a[h].flag == 1) {

h = h + 3 \* l;

l++;

if (h > 1999) {

bdk++;

f = 0;

break;

}

}

if (f == 1) {

a[h].flag = 1;

a[h].key = k;

a[h].data = 8;

}

}

return bdk;

}

void print(record\* tab, int n) {

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

cout << i << " " << tab[i].key << " " << tab[i].flag << endl;

}

string inputcheck(int\* ad) {

char inputkey[7];

int f = 0;

while ((f == 0) || cin.fail()) {

cout << "Введите ключ вида ББццББ : ";

cin.clear();

cin.ignore(cin.rdbuf()->in\_avail());

cin.getline(inputkey, 7);

f = 1;

if (!cin.fail()) {

if (sizeof(inputkey) < 6) {

cout << "Мало символов";

f = 0;

}

else {

for (int i = 0; i < 6; i++) {

if (i == 2 || i == 3) {

if (int(inputkey[i]) < 48 || int(inputkey[i]) > 57) {

cout << i + 1 << " символ неверен.\n";

f = 0;

}

}

else {

if (int(inputkey[i]) < 65 || int(inputkey[i]) > 90) {

cout << i + 1 << " символ неверен.\n";

f = 0;

}

}

}

}

}

else {

f = 0;

cout << "Введено слишком много символов";

}

if (f == 0) {

cout << "Попробуйте снова.\n";

}

}

\*ad = func(inputkey);

return inputkey;

}

void add(record\* tab) {

int ad = 0;

string inputkey = inputcheck(&ad);

cout << "Начальный адрес " << ad << endl;

int l = 0;

while (tab[ad].flag == 1) {

ad += 3;

if (ad > 1999) {

if (l < 3) {

ad = ad % 1999;

l++;

}

else {

cout << "Не удалось добавить\n";

return;

}

}

}

cout << "Предыдущее значениe ключа: " << tab[ad].key << endl;

tab[ad].flag = 1;

tab[ad].key = inputkey;

tab[ad].data = 8;

cout << "Запись успешно добавлена. Адрес: " << ad << endl;

}

int found(record\* tab, string o, int\* col) {

int ad;

\*col = 0;

string inputkey;

if (o[0] == '1') {

inputkey = inputcheck(&ad);

}

else {

cout << "Введите адрес сегмента :";

cin >> ad;

if (tab[ad].flag == 1) {

cout << "Запись найдена";

return ad;

}

else {

cout << "Записи с таким ключом нет.\n";

return -1;

}

}

cout << "изначальный адрес " << ad << endl;

int l = 0;

while (tab[ad].flag == 1 || tab[ad].flag == 2) {

if (inputkey == tab[ad].key) {

cout << "Запись найдена";

return ad;

}

else {

(\*col)++;

ad += 3;

if (ad > 1999) {

if (l < 3) {

ad = ad % 1999;

l++;

}

else {

cout << "Записи с таким ключом нет.\n";

return -1;

}

}

}

}

cout << "Записи с таким ключом нет.\n";

return -1;

}

void del(record\* tab) {

cout << "Чтобы удалить запись по ключу введите 1, иначе будет выполнено удаление по сегменту : ";

int coliz;

string o;

cin.clear();

cin.ignore(cin.rdbuf()->in\_avail());

getline(cin, o);

int ad = found(tab, o, &coliz);

if (ad == -1) {

cout << "Удаление отменено.\n";

return;

}

else {

tab[ad].flag = 2;

tab[ad].data = 0;

tab[ad].key = {};

cout << ". Удаление по адресу " << ad << " успешно завершено.\n";

}

}

void main() {

setlocale(LC\_ALL, "rus");

record\* tab = new record[2000];

int bdk;

int p = 6;

while (p != 0) {

switch (p) {

case 0:

p = 0;

case 1:

cout << "Создание хеш таблицы из 2000 рандомных ключей.\n";

bdk = create(tab, 2000);

cout << "Ключей не добавлено : " << bdk << endl;

break;

case 2:

cout << "Добавление записи.\n";

add(tab);

break;

case 3: {

cout << "Поиск записи.\n";

cout << "Чтобы произвести поиск по ключу введите 1, иначе будет выполнен поиск по сегменту : ";

string o;

cin.clear();

cin.ignore(cin.rdbuf()->in\_avail());

getline(cin, o);

int col;

int ad = found(tab, o, &col);

if (ad != -1) {

cout << ": " << ad << " " << tab[ad].key << " " << tab[ad].flag << " (коллизии до зпс " << col << ")" << endl;

}

break; }

case 4:

cout << "Удаление записи.\n";

del(tab);

break;

case 5:

cout << "Вывод таблицы.\n";

print(tab, 2000);

break;

case 6:

cout << "Меню пользователя:\n";

cout << "0 - Выход\n1 - Создание рандомной таблицы\n2 - Добавление записи\n3 - Поиск записи\n4 - Удаление записи\n5 - Просмотр таблицы\n6 - Меню пользователя\n";

}

cout << "Введите номер команды : ";

cin >> p;

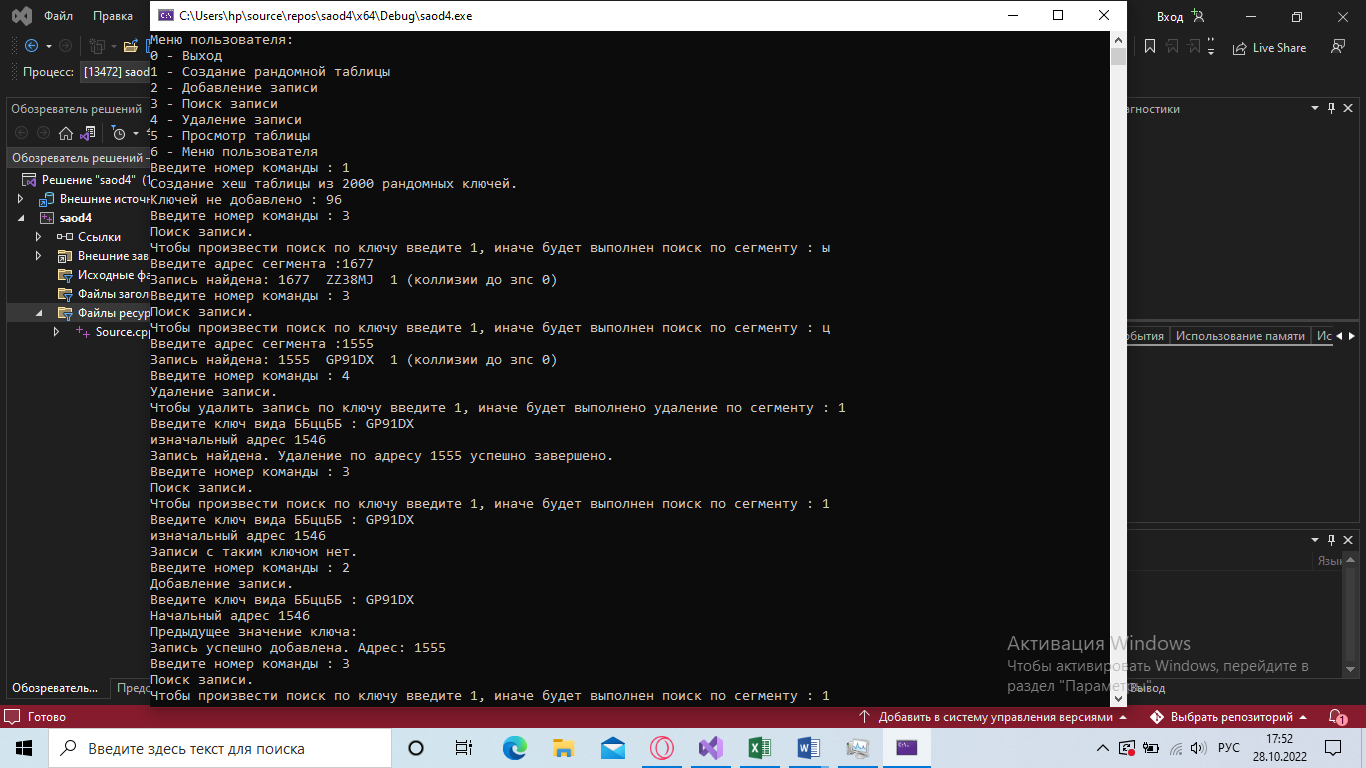
if (p == 0)

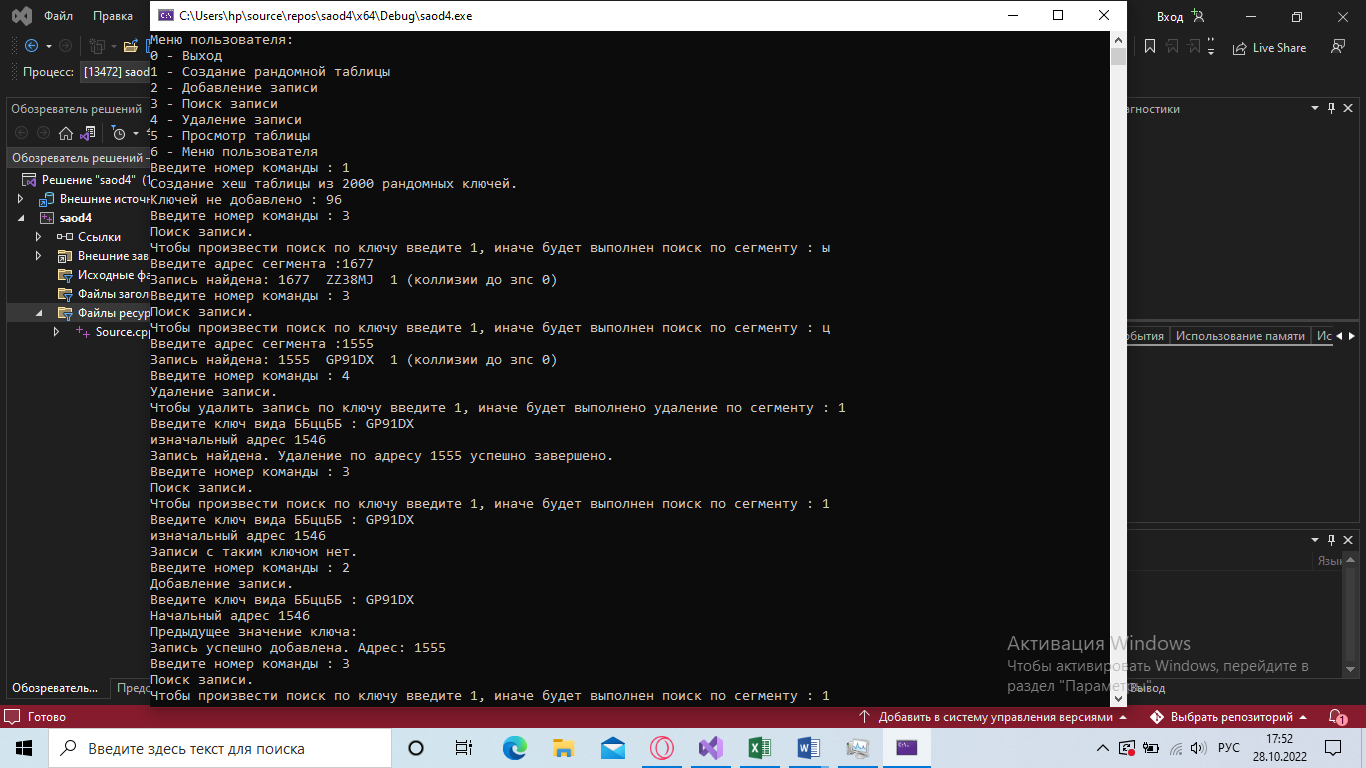
cout << "Программа завершена.";

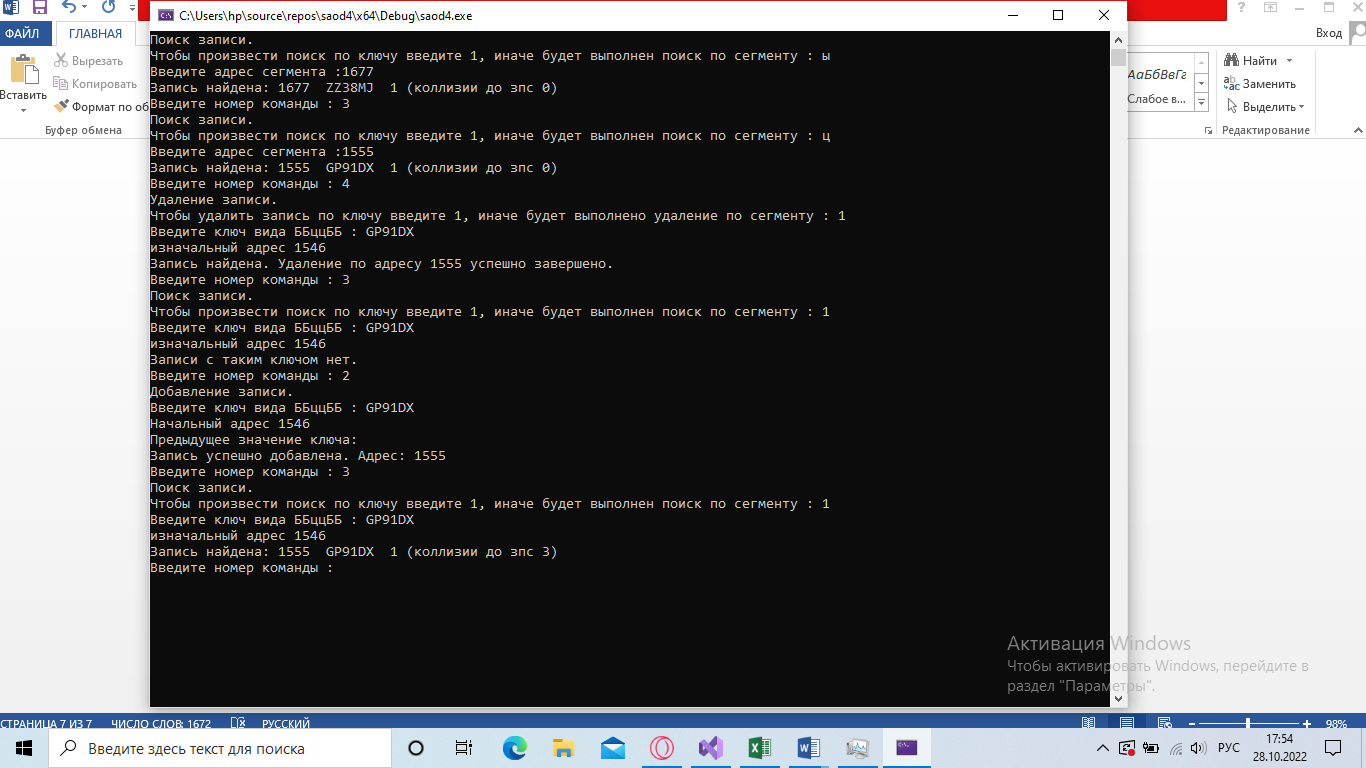
}

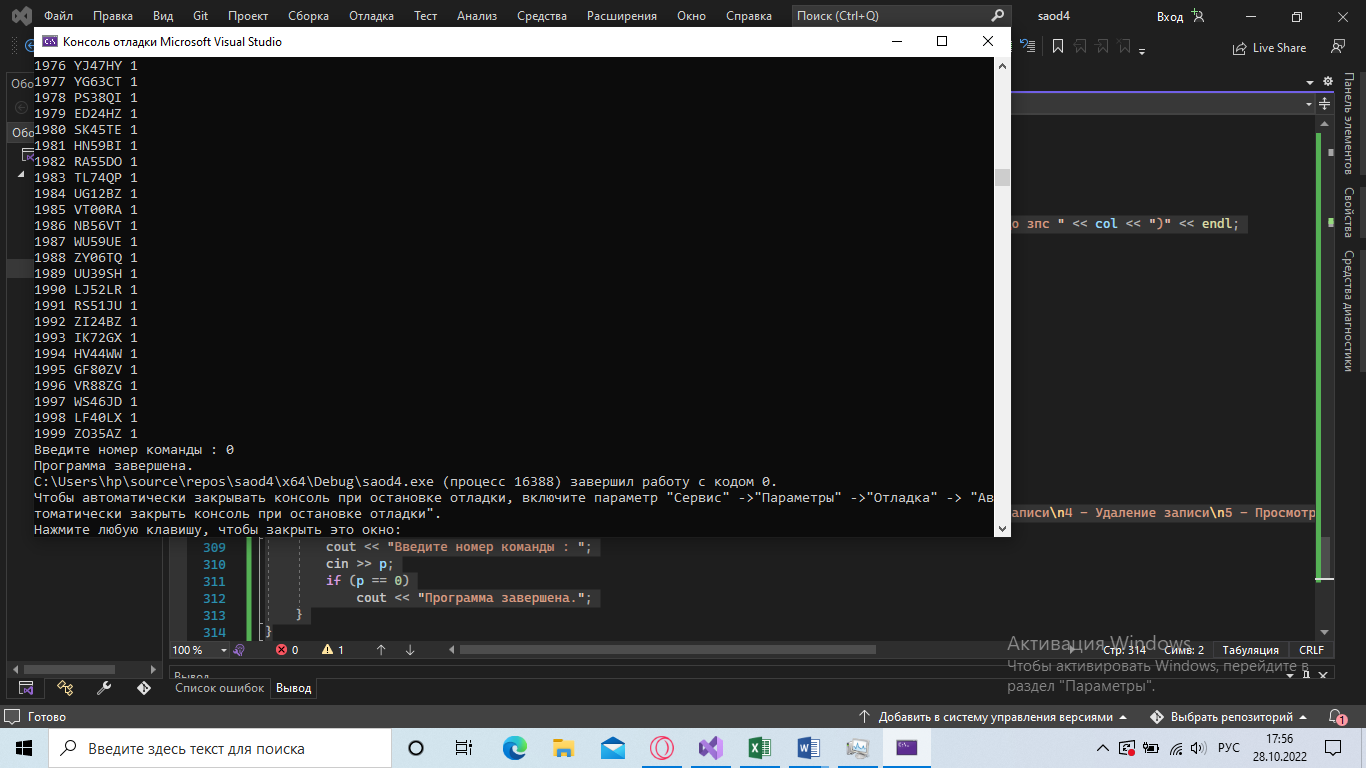
}

* 1. Работа программы









* 1. Вывод

Я изучил методы хеширования данных и получил практические навыки реализации хеш-таблиц.