

ГУАП  
КАФЕДРА №43

ОТЧЁТ  
ЗАЩИЩЁН С ОЦЕНКОЙ  
ПРЕПОДАВАТЕЛЬ

\_\_\_\_\_  
должность, уч. степень, звание

\_\_\_\_\_  
подпись, дата

\_\_\_\_\_  
инициалы, фамилия

**Лабораторная работа №4**

**Хеширование данных**

по дисциплине: Структуры и алгоритмы обработки данных

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. Z8432K

22.12.2019

А.С. ЛУНЕВ

\_\_\_\_\_  
подпись, дата

\_\_\_\_\_  
инициалы, фамилия

## 1. Цель работы

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

## 2. 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;

### Вариант №1:

№ вар.	Формат ключа	Количество сегментов	Метод хеширования (разрешения коллизий)
1	цццБцц	1500	Открытое хеширование

Где «ц» – это цифра 0...9; «Б» – это большая буква латиницы A...Z.

## 3. Описание созданных функций

Для реализации задания были использованы следующие функции:

**Имя:** hashFunc.

**Назначение:** хеш-функция.

**Входные данные:**

- **k** – переменная типа Key.

**Выходные данные:**

- вычисленный хеш-функции

**Побочный эффект:** отсутствует.

**Имя:** print.

**Назначение:** вывод в консоль хеш-функции.

**Входные данные:**

- ht – переменная типа HashTable.

**Выходные данные:**

**Побочный эффект:** отсутствует.

**Имя:** find.

**Назначение:** поиск ключа в хеш-таблице.

**Входные данные:**

- ht – переменная типа HashTable;
- key – переменная типа Key;

**Выходные данные:**

**Побочный эффект:** отсутствует.

**Имя:** insert.

**Назначение:** добавление значение в хеш-таблицу.

**Входные данные:**

- ht – переменная типа HashTable;
- key – переменная типа Key;

**Выходные данные:**

**Побочный эффект:** отсутствует.

**Имя:** remove.

**Назначение:** удаление значения из хеш-таблицы.

**Входные данные:**

- ht – переменная типа HashTable;
- key – переменная типа Key;

**Выходные данные:**

**Побочный эффект:** отсутствует.

**Имя:** stat.

**Назначение:** сбор и запись статистики распределения хешей в файл.

**Входные данные:**

- ht – переменная типа HashTable;
- key – переменная типа Key;

**Выходные данные:**

**Побочный эффект:** отсутствует.

**Имя:** mainmenu.

**Назначение:** печать пунктов меню в консоль.

**Входные данные:**

**Выходные данные:**

**Побочный эффект:** отсутствует.

Так же была использована структура Key с конструкторами

**Имя:** Key.

**Назначение:** генерация ключа.

**Входные данные:**

- s – константная переменная;

**Выходные данные:**

**Побочный эффект:** отсутствует.

**Имя:** operator.

**Назначение:** сравнение двух ключей.

**Входные данные:**

- k2 – переменная типа Key;

**Выходные данные:**

**Побочный эффект:** отсутствует.

**Имя:** operator.

**Назначение:** сравнение двух ключей.

**Входные данные:**

- k2 – переменная типа Key;

**Выходные данные:**

**Побочный эффект:** отсутствует.

**Имя:** gen.

**Назначение:** генерация случайных ключей.

**Входные данные:**

**Выходные данные:**

**Побочный эффект:** отсутствует.

## 4. Листинг программы

---

```
1. #include <iostream>
2. #include <fstream>
3. #include <list>
4. #include <vector>

5. using namespace std;

6. const int N_SEG = 12; // 1500 кол-во сегментов
7. const int KEY_LEN = 7; // длина ключа

8. struct Key
9. {
10.     char key[KEY_LEN+1];

11.     Key() {}

12.     Key(const char *s)
13.     {
14.         for(int i=0; i<KEY_LEN+1; i++)
15.             key[i] = s[i];
16.     }

17.     // Оператор для сравнения ключей посимвольно
18.     bool operator==(Key k2)
19.     {
20.         for(int i=0; i<KEY_LEN; i++)
21.             if(this->key[i] != k2.key[i])
22.                 return false;
23.         return true;
24.     }

25.     void gen()
26.     {
27.         for(int i=0; i<3; i++)
28.             key[i] = '0' + rand()%10;
29.         key[3] = 'A' + rand()%26;
30.         for(int i=4; i<6; i++)
31.             key[i] = '0' + rand()%10;
32.         key[6] = 0; // для разделения ключей
33.     }
34. };

35. typedef vector<list<Key>> HashTable;
36. //using HashTable = vector<list<Keys>>;
```

```

37.     int hashFunc(Key k)
38.     {
39.         int sum = 0;
40.         for(int i=0; i<KEY_LEN; i++)
41.             sum += k.key[i] * k.key[i];

42.         return sum % N_SEG;
43.     }

44.     void print(const HashTable &ht)
45.     {
46.         cout << "Hash Table:" << endl;
47.         for(int i=0; i<ht.size(); i++)
48.         {
49.             cout << i << "\t";
50.             auto keys = ht[i];
51.             for(auto it=keys.begin(); it!=keys.end(); ++it)
52.             {
53.                 cout << it->key << " ";
54.             }
55.             cout << endl;
56.         }
57.     }

58.     bool find(const HashTable &ht, Key key)
59.     {
60.         int i = hashFunc(key); // номер сегмента в таблице
61.         auto keys = ht[i]; // список в i-ом сегменте хеш-таблицы

62.         // перебираю все ключи в списке
63.         for(auto it=keys.begin(); it!=keys.end(); ++it)
64.         {
65.             if( *it == key)
66.                 return true;
67.         }
68.         return false;
69.     }

70.     void insert(HashTable &ht, Key key)
71.     {
72.         if(!find(ht, key))
73.         {
74.             int i = hashFunc(key); // номер сегмента в таблице
75.             // cout << "insert: i=" << i << endl;
76.             ht[i].push_back(key); // добавляем ключ в конец списка в i-ом сегменте
77.         }
78.     }

79.     void remove(HashTable &ht, Key key)

```

```

80.     {
81.         int i = hashFunc(key); // номер сегмента в таблице
82.         ht[i].remove(key); // удаление элемента из списка
83.     }

84.     void stat(const HashTable &ht)
85.     {
86.         ofstream fout("stat.txt");
87.         for (int i = 0; i < ht.size(); i++)
88.         {
89.             auto keys = ht[i];
90.             fout << i << "\t" << keys.size() << endl;
91.         }
92.         fout.close();
93.     }

94.     void mainmenu()
95.     {
96.         cout << "1. Generated hash table" << endl;
97.         cout << "2. Print hash table" << endl;
98.         cout << "3. Delete key" << endl;
99.         cout << "4. Find key" << endl;
100.        cout << "5. Print statistics hash table" << endl;
101.        cout << "9. Exit menu" << endl;
102.    }

103.    int main()
104.    {
105.        int choose = 0;
106.        HashTable hashTable(N_SEG);

107.        do
108.        {
109.            mainmenu();

110.            cout << "Choose item menu: ";
111.            cin >> choose;

112.            switch (choose)
113.            {

114.            case 1:
115.            {
116.                for (int i = 0; i < 30; i++)
117.                {
118.                    Key k;
119.                    k.gen();
120.                    insert(hashTable, k);
121.                }

```

```
122.     break;
123. }

124.     case 2:
125.     {
126.         print(hashTable);
127.         break;
128.     }

129.     case 3:
130.     {
131.         cout << "Input key to delete: ";
132.         char buf[7];
133.         cin >> buf;
134.         // проверка ключа
135.         Key k(buf);
136.         remove(hashTable, k);
137.         print(hashTable);
138.         break;
139.     }

140.     case 4:
141.     {
142.         cout << "Input key to find: ";
143.         char buf[7];
144.         cin >> buf;
145.         // проверка ключа
146.         Key k(buf);
147.         bool res = find(hashTable, Key(buf)); // true, false
148.         cout << std::boolalpha << res << endl;
149.         break;
150.     }

151.     case 5:
152.     {
153.         stat(hashTable);
154.         break;
155.     }
156. }
157. } while (choose != 9);

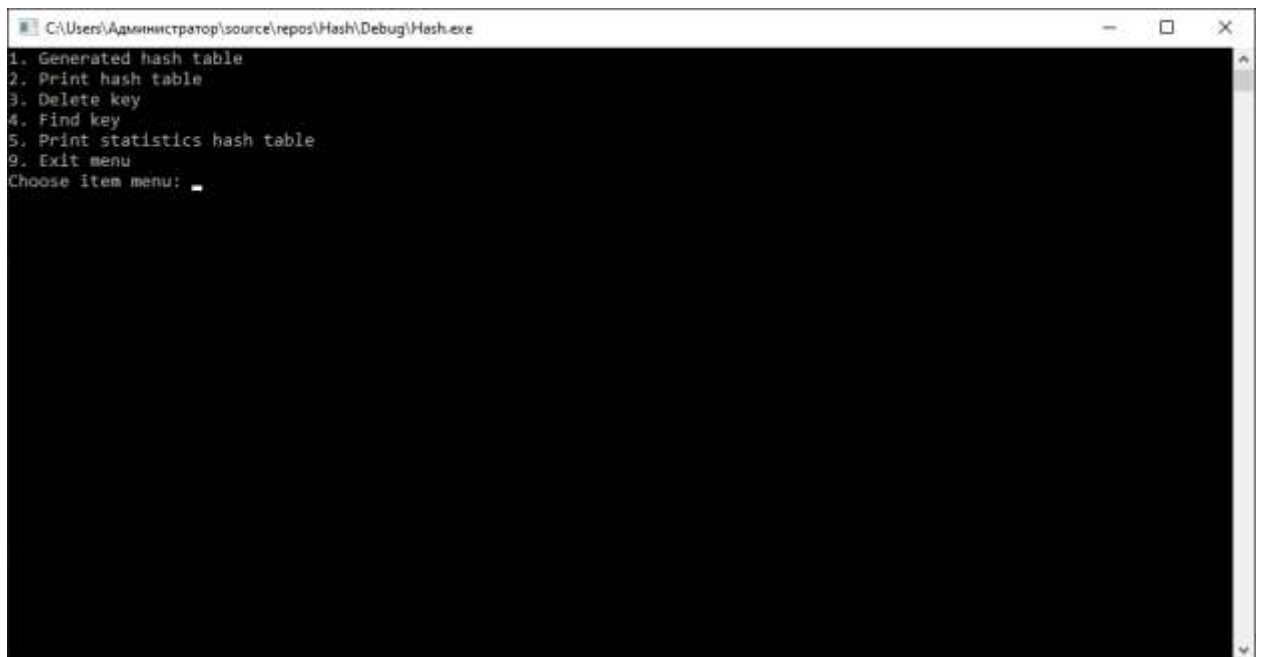
158.     system("pause");
159.     return 0;
160. }
```

---



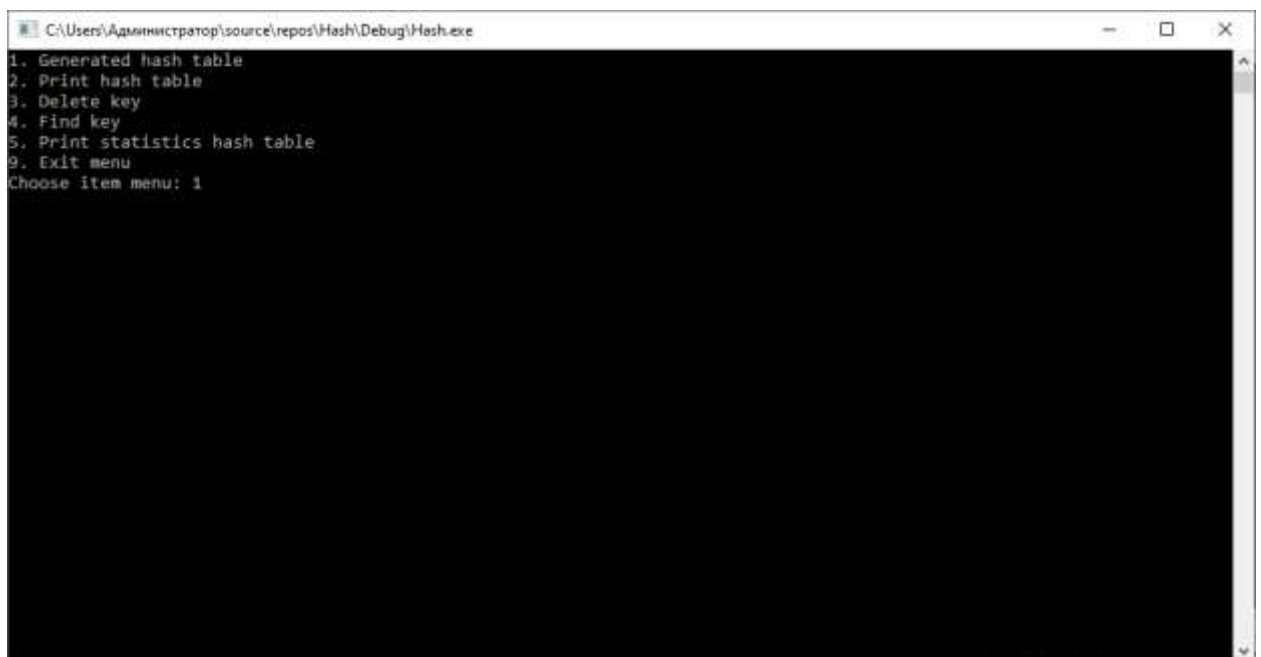
## 5. Пример выполнения программы

Ниже приведён пример выполнения программы



```
C:\Users\Администратор\source\repos\Hash\Debug\Hash.exe
1. Generated hash table
2. Print hash table
3. Delete key
4. Find key
5. Print statistics hash table
9. Exit menu
Choose item menu: _
```

Рис 1. Пример выполнения программы



```
C:\Users\Администратор\source\repos\Hash\Debug\Hash.exe
1. Generated hash table
2. Print hash table
3. Delete key
4. Find key
5. Print statistics hash table
9. Exit menu
Choose item menu: 1
```

Рис 2. Пример выполнения программы

```
C:\Users\Администратор\source\repos\Hash\Debug\Hash.exe
5. Print statistics hash table
9. Exit menu
Choose item menu: 1
1. Generated hash table
2. Print hash table
3. Delete key
4. Find key
5. Print statistics hash table
9. Exit menu
Choose item menu: 2
Hash Table:
0      171X52 793R98
1      777P33 599I18 618H84
2      650W86 266E38
3      882Y55 097Z72 478J51
4      761S23 761U92 601U57
5
6      389N44 606S18 496N78 012R09
7      090K61 829F35 984Y76 541000
8      174G94 387G27 146V71
9      233K11
10     795031 024C65 361P42 201A40
11     221W85
1. Generated hash table
2. Print hash table
3. Delete key
4. Find key
5. Print statistics hash table
9. Exit menu
Choose item menu: 
```

Рис 3. Пример выполнения программы

```
C:\Users\Администратор\source\repos\Hash\Debug\Hash.exe
9. Exit menu
Choose item menu: 1
1. Generated hash table
2. Print hash table
3. Delete key
4. Find key
5. Print statistics hash table
9. Exit menu
Choose item menu: 2
Hash Table:
0      171X52 793R98
1      777P33 599I18 618H84
2      650W86 266E38
3      882Y55 097Z72 478J51
4      761S23 761U92 601U57
5
6      389N44 606S18 496N78 012R09
7      090K61 829F35 984Y76 541000
8      174G94 387G27 146V71
9      233K11
10     795031 024C65 361P42 201A40
11     221W85
1. Generated hash table
2. Print hash table
3. Delete key
4. Find key
5. Print statistics hash table
9. Exit menu
Choose item menu: 3
Input key to delete: 221W85
```

Рис 4. Пример выполнения программы

```
C:\Users\Администратор\source\repos\Hash\Debug\Hash.exe
10      795031 024C65 361P42 201A40
11      221W85
1. Generated hash table
2. Print hash table
3. Delete key
4. Find key
5. Print statistics hash table
9. Exit menu
Choose item menu: 3
Input key to delete: 221W85
Hash Table:
0      171X52 793R08
1      777P33 599I18 618H84
2      650W86 266E38
3      882V55 097Z72 478J51
4      761523 761U92 601U57
5
6      389N44 606S18 496N78 012R09
7      090K61 829F35 984Y76 541000
8      174G94 387G27 146Y71
9      233K11
10     795031 024C65 361P42 201A40
11
1. Generated hash table
2. Print hash table
3. Delete key
4. Find key
5. Print statistics hash table
9. Exit menu
Choose item menu:
```

Рис 5. Пример выполнения программы

```
C:\Users\Администратор\source\repos\Hash\Debug\Hash.exe
Input key to delete: 221W85
Hash Table:
0      171X52 793R08
1      777P33 599I18 618H84
2      650W86 266E38
3      882V55 097Z72 478J51
4      761523 761U92 601U57
5
6      389N44 606S18 496N78 012R09
7      090K61 829F35 984Y76 541000
8      174G94 387G27 146Y71
9      233K11
10     795031 024C65 361P42 201A40
11
1. Generated hash table
2. Print hash table
3. Delete key
4. Find key
5. Print statistics hash table
9. Exit menu
Choose item menu: 4
Input key to find: 761523
true
1. Generated hash table
2. Print hash table
3. Delete key
4. Find key
5. Print statistics hash table
9. Exit menu
Choose item menu:
```

Рис 6. Пример выполнения программы

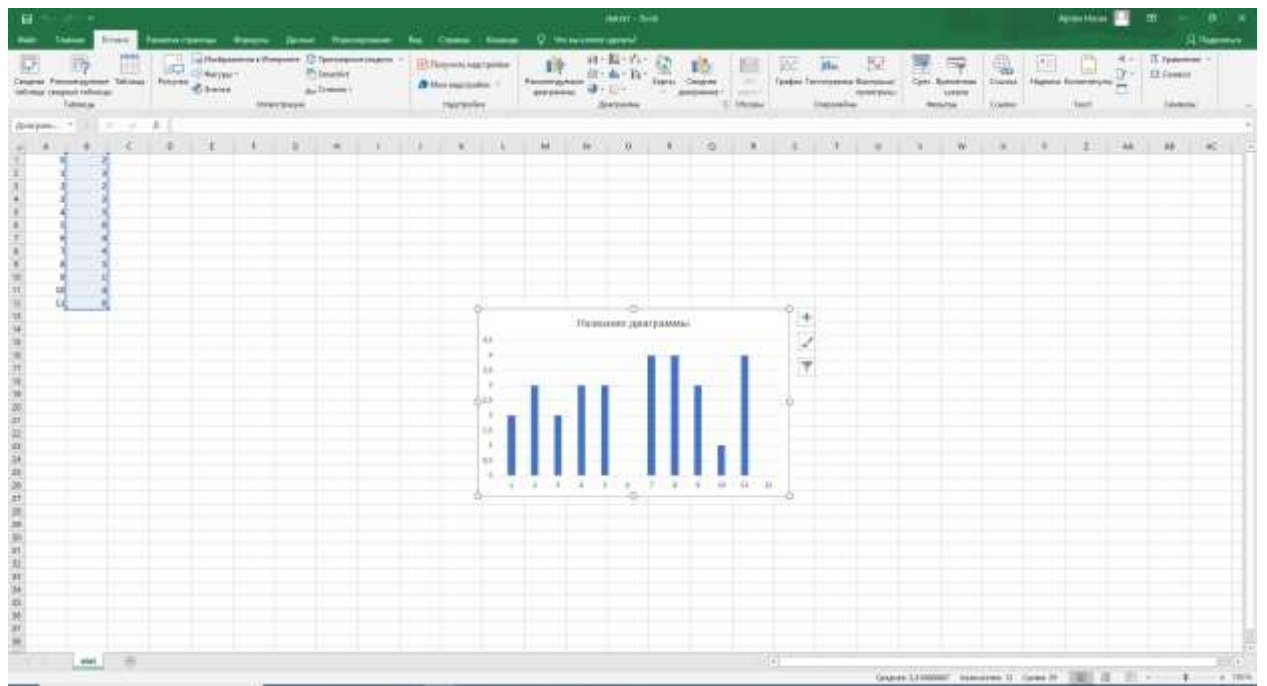


Рис 7. Пример выполнения программы

## **6. Анализ результатов и выводы**

Из достоинств можно выделить следующее:

Программа выполняет поставленную задачу и работает без ошибок

Из недостатков можно выделить следующее:

В качестве ответа на найден ли ключ в хеш-таблице выводится true – что означает найдено и false – что означает не найдено.