МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МОЭВМ

КУРСОВАЯ РАБОТА

по дисциплине «Алгоритмы и структуры данных»

Тема: Хеш-таблицы с цепочками

Студентка гр. 9383 _____ Лапина А.А.

Преподаватель _____ Попова Е.В.

Санкт-Петербург

2020

ЗАДАНИЕ

НА КУРСОВУЮ РАБОТУ

Студентка Лапина А.А.

Группа 9383

Тема работы: Хеш-таблица с цепочками

Исходные данные:

Хеш-таблица с цепочками, операции вставки и исключение, язык программирования C++

Содержание пояснительной записки:

«Содержание», «Формальная постановка задачи», «Основные теоретические сведения», «Описание алгоритмов», «Описание структуры данных и функций», «Тестирование», «Исходный код программы», «Заключение»

Предполагаемый объем пояснительной записки:

Не менее 15 страниц.

Дата выдачи задания: 4.11.2020

Дата сдачи реферата: 21.12.2020

Дата защиты реферата: 21.12.2020

Студентка _____ Лапина А.А.

Преподаватель _____ Попова Е.В.

АННОТАЦИЯ

В данной курсовой работе главной задачей была демонстрация функций вставки и исключения элемента в такой структуре данных, как хеш-таблица с методом решения коллизий — цепочки. Была написана программа на языке программирования C++, на вход программе подаются элементы хэш-таблицы, с помощью которых она заполняется. Затем выполняется вставка и удаление элементов, которые вводит пользователь. После каждой функции выводится новая хэш-таблица на экран.

SUMMARY

In this course work, the main task was to demonstrate the functions of inserting and excluding an element in such a data structure as a hash table with a collision solution method - a chain. A program was written in the C ++ programming language, the elements of a hash table are fed to the program, with which it is filled. It then inserts and removes items that the user enters. After each function, a new hash table is displayed on the screen.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Цель работы: демонстрация вставки и исключения в хэш-таблицу с цепочками.

Чтобы выполнить поставленную цель необходимо решить следующие **задачи:**

1. Изучение теоретических сведений по изучаемой структуре данных (хеш-таблицам).
2. Изучение теоретических сведений о функциях вставки и исключения.
3. Написание программы на языке C++, реализующую заданную структуру данных и необходимые функции.
4. Тестирование программы.
5. Отладка программы.

1.ФОРМАЛЬНАЯ ПОСТАНОВКА ЗАДАЧИ

Вариант: 23

Хеш-таблицы с цепочками – вставка и исключение. Демонстрация

"Демонстрация" - визуализация структур данных, алгоритмов, действий. Демонстрация должна быть подробной и понятной (в том числе сопровождаться пояснениями), чтобы программу можно было использовать в обучении для объяснения используемой структуры данных и выполняемых с нею действий.

2. ОСНОВНЫЕ ТЕОРЕТИЧЕСКИЕ ПОЛОЖЕНИЯ

2.1. Хеш-таблица

Хеш-таблица — это структура данных, реализующая интерфейс ассоциативного массива, а именно, она позволяет хранить пары (ключ, значение) и выполнять три операции: операцию добавления новой пары, операцию поиска и операцию удаления пары по ключу.

Число хранимых элементов, делённое на размер массива H (число возможных значений хеш-функции), называется **коэффициентом заполнения хеш-таблицы** и является важным параметром, от которого зависит среднее время выполнения операций. Формула: k = n / N, где n – число хранимых элементов, N – размер массива, k – коэффициент заполнения хэш-таблицы.

Операции в хэш-таблице.

Выполнение операции в хеш-таблице начинается с вычисления хеш-функции от ключа. Хеш-код i = h(key) играет роль индекса в массиве H, а зная индекс, мы можем выполнить требующуюся операцию (добавление, удаление или поиск).

2.2 Хеш-функция

Хеш-функция - это математическая или иная функция, которая для строки произвольной длины вычисляет некоторое целое значение или некоторую другую строку фиксированной длины.

В общем случае нет однозначного соответствия между хеш-кодом (выходными данными) и исходными (входными) данными. Возвращаемые хеш-функцией значения (выходные данные) менее разнообразны, чем значения входного массива (входные данные). Случай, при котором хеш-функция преобразует более чем один массив входных данных в одинаковые сводки, называется «**коллизией**». Вероятность возникновения коллизий используется для оценки качества хеш-функций.

Другими словами, **коллизия (конфликт)** – событие, когда для разных ключей Ki ≠Kj функция расстановки H даёт один и тот же адрес H (Ki) = H (Kj).

2.3 Метод цепочек

Метод цепочек — метод, где каждая ячейка массива H является указателем на цепочку пар ключ-значение, соответствующих одному и тому же хеш-значению ключа. Коллизии просто приводят к тому, что появляются цепочки длиной более одного элемента.

Существуют следующие подходы к реализации:

1) Прямой метод цепочек, когда элементы списка находятся в самой хеш-таблице

2) Метод, когда элементы списка размещаются в области переполнения (в таблице символов), а хеш-таблица представляет собой таблицу индексов.

3) Метод цепочек, использующий таблицу индексов и списки.

В данной работе использован 3 метод реализации хэш-таблиц.

Метод цепочек, использующий таблицу индексов и списки.

Таблица символов реализована в виду отдельных цепочек, «присоединённая» каждая к своему индексу. Каждая ячейка i массива H содержит указатель на начало списка всех элементов, хеш-код которых равен i, либо указывает на их отсутствие. Коллизии приводят к тому, что появляются списки размером больше одного элемента.

Время работы поиска в наихудшем случае пропорционально длине списка, так как все n ключей захешировались в одну и ту же ячейку (создав список длиной n) время поиска будет равно O(n) плюс время вычисления хеш-функции. Удаление элемента может быть выполнено за O(1), как и вставка, при использовании двухсвязного списка.

Преимущества:

1) Не требует перемещения элементов, т.к. каждый элемент таблицы индексов всегда указывает на начало цепочки.

2) Легко обрабатывать переполнение таблицы символов, т.к. ограничена таблица индексов, а элементы можно добавлять в список.

Недостатки:

При неудачном выборе N — числа, отвечающего за размер таблицы индексов (то есть количество индексов), будут использованы не все индексы. Например, если использовать хэш-функцию как на рисунке 1 $h(x)=\left(\sum_{i=0}^{len\,s} ASCII(s[i])\right) mod\, N$ для N=10 используются только 4 индекса 1, 2, 3, 5, но получается 3 цепочки.

На рисунке 1 изображено представление хэш-таблицы, реализованной методом цепочек для функции $h(x)=\left(\sum_{i=0}^{len\,s} ASCII(s[i])\right) mod\, 11$ (Остаток от деления на 11 суммы ASCII-кодов символов, входящих в строку). Где i — это счетчик для прохода по строке от нулевого элемента да последнего элемента строки (len s — длина строки, то есть с его помощью узнаем номер последнего символа). Данная хэш-функция была рассмотрела на лекционном занятии. В данной работе реализуем ее с помощью программного кода.

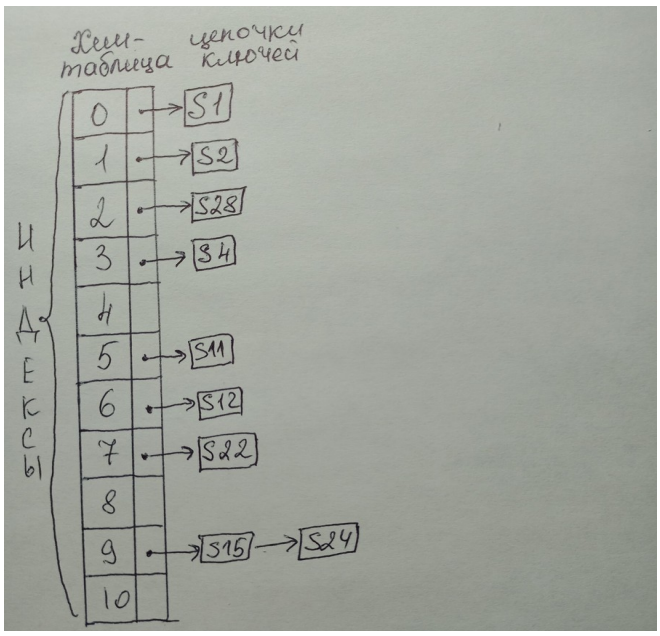

Рисунок 1 — Хэш-таблица, реализованная методом цепочек

3. ОПИСАНИЕ АЛГОРИТМОВ

3.1. Алгоритм вставки (с учётом уникальности)

1. Вычисляем хэш (индекс) для переданного элемента.

2. Обращаемся к цепочке по этому индексу, проверяем есть ли уже такой элемент в цепочке:

1) Если есть, то выводим сообщение о том, что данный элемент уже существует и вставка выполнена не будет. Выводим старую таблицу (без изменений).

2) Если такого элемента нет, то вставляем его в цепочку (если элементов нет, то вставляем в качестве первого, а если там уже есть элементы, то вставляем в конец цепочки). Выводим новую хеш-таблицу.

3.2. Алгоритм удаления

1. Вычисляем хэш (индекс) для переданного элемента.

2. Проверяем, есть ли такой элемент по индексу:

1) Если нет — выводим сообщение, что такого элемента в таблице нет, следовательно удалять нечего. Выводим старую таблицу (без изменений).

2) Если да — удаляем элемент. Выводим новую хэш-таблицу.

4. ОПИСАНИЕ СТРУКТУРЫ ДАННЫХ И ФУНКЦИЙ

4.1. Структура данных

Таблица представляет из себя массив. Который имеет четкую заранее заданную размерность и равен N (N – число от деления, на которое будет считаться остаток). Именно такой размер массив имеет, так как является таблицей индексов, где индексы – все возможные возвращаемые хеш-функцией значения. По индексам в свою очередь хранятся цепочки, реализованные на базе массива.

4.2. Функции программы

1) Разработан класс Elem. Каждый элемент хранит поля key-ключ, value — значение, Elem* next — ссылка на следующий элемент, Elem* pred — ссылка на предыдущий;

2) start() - заполняет ключи значением «NULL», чтобы понимать лежит ли ключ по данному индексу или нет;

3) print() - печатает хэш-таблицу;

4) func() - это хэш — функция, с помощью которой будем вычислять индекс ключа;

5) insert() - функция вставки элемента. Применяем алгоритм вставки: вычисляем индекс элемента, с помощью func() и обращаемся по полученному из нее индексу. Смотрим, лежит ли такой ключ там или нет. Если да, то не вставляем, выводим старую хэш-таблицу. Если нет, то вставляем элемент в цепочку.

6) deletion() - функция для удаления элемента. Применяем алгоритм удаления: вычисляем индекс элемента, с помощью func() и обращаемся по полученному из нее индексу. Смотрим, есть ли в цепочке такой элемент или нет. Если есть, то удаляем и выводим новую хэш-таблицу. Если такого элемента нет, то выводим сообщение о том, что удаление невозможно по причине отсутствия элемента в таблице и выводим старую хэш-таблицу.

7) main() — используем вышеперечисленные функции. Создаем таблицу индексов и символов. Заполняем хэш-таблицу, с помощью ключей, который вводит пользователь (для данной программы «!» - является концом ввода ключей), печатаем хэш-таблицу, выполняем демонстрацию вставки и удаления элементы, выводим хэш-таблицы, с помощью вышеописанных функций.

5. ТЕСТИРОВАНИЕ

5.1. Тестирование 1

Входные данные:

S1 S2 S4 S11 S12 S15 S22 S24 S28 !

(Считывание элементов до символа «!»)

S14 //элемент для вставки

S15 //элемент для исключения

Выходные данные изображены на рисунке 2.

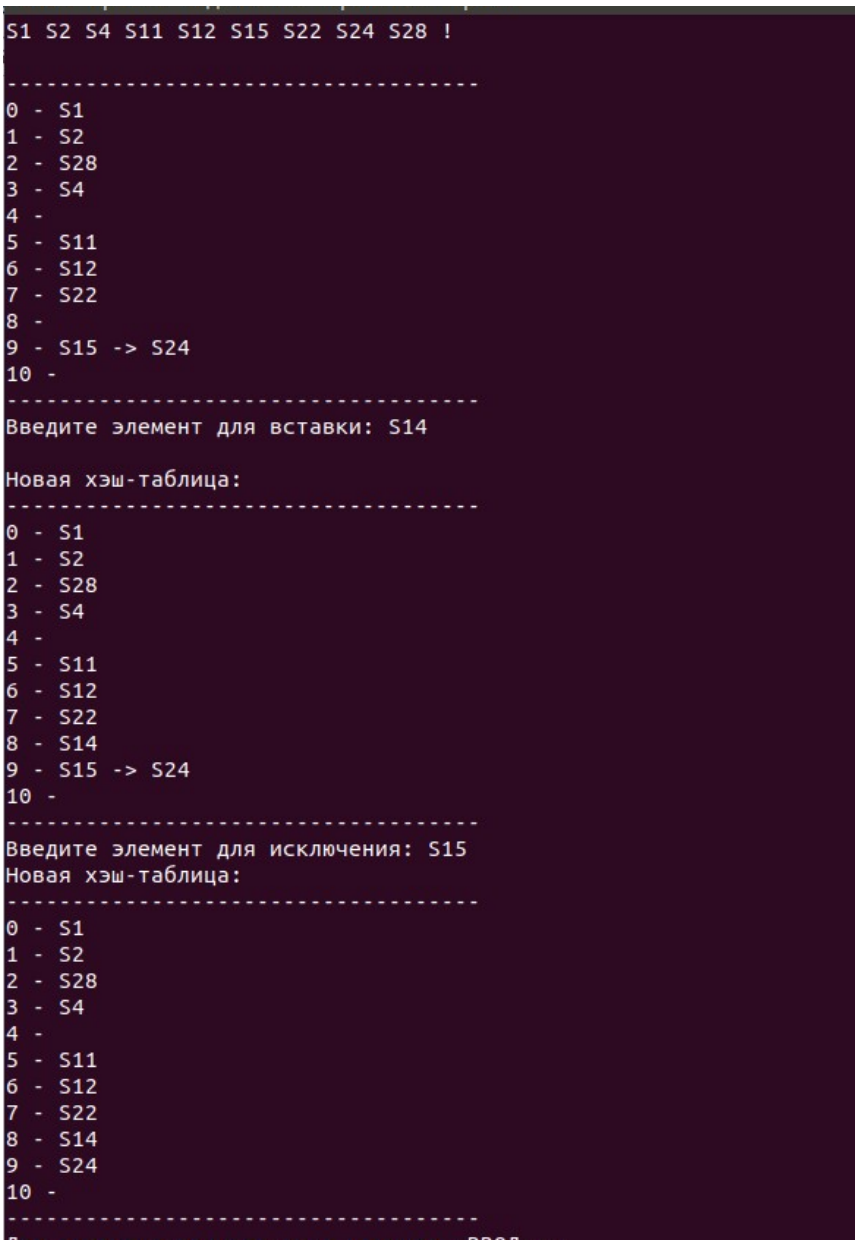

Рисунок 2 — Демонстрация работы программы с входными данными №1

5.2. Тестирование 2

Входные данные: t1 t2 t5 t12 t13 t15 t22 t24 t25 t29 !

(Считывание элементов до символа «!»)

t33 //элемент для вставки

t40 //элемент для исключения (так как такого элемента нет, выведется сообщение об этом и старая хэш-таблица)

Выходные данные изображены на рисунке 3.

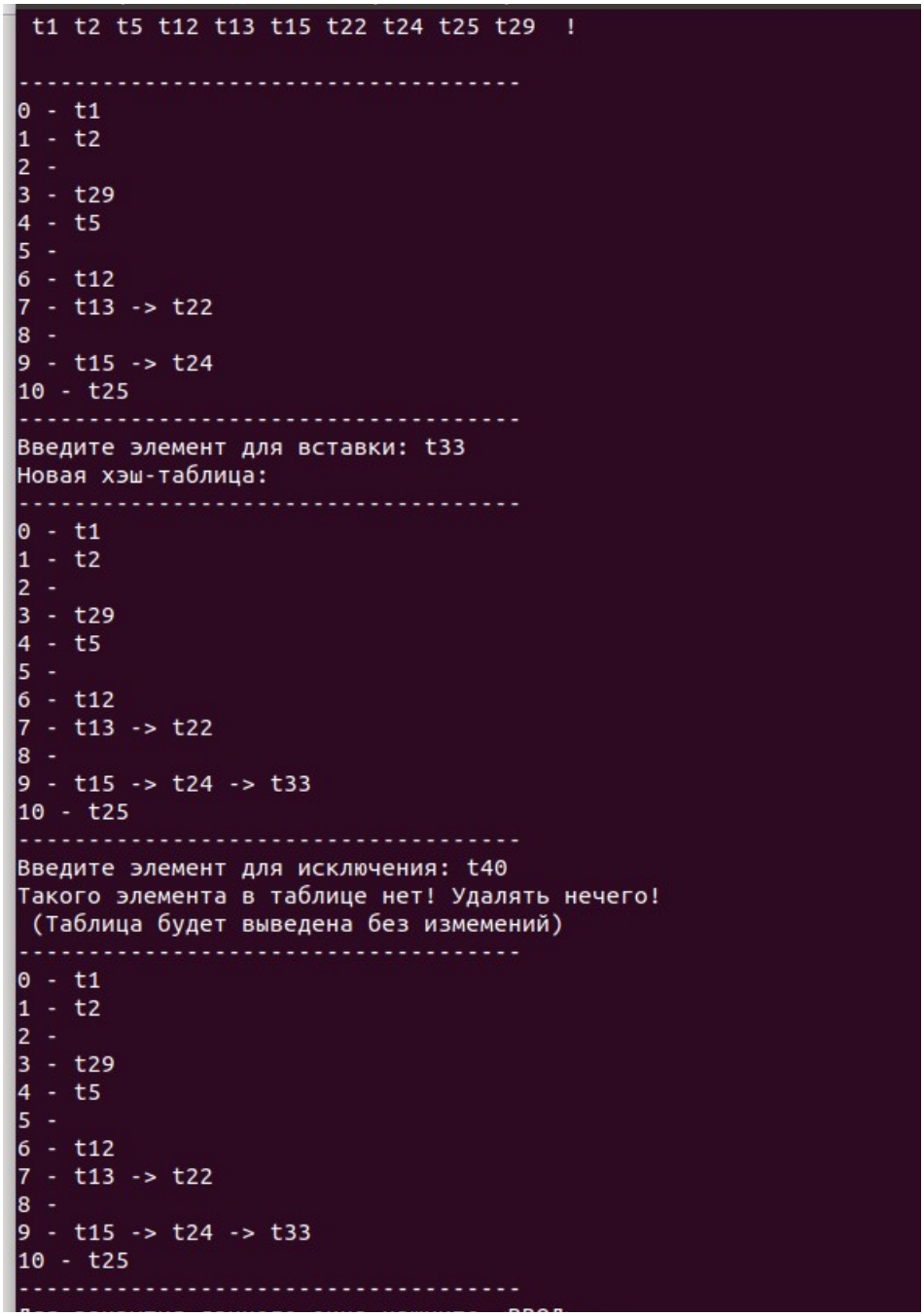

Рисунок 3 — Демонстрация работы программы с входными данными №2

5.3. Тестирование 3

Входные данные: we rrege vf jpjrpojg bvjkp fn jgb !

(Считывание элементов до символа «!»)

vf //элемент для вставки (так как такой элемент уже есть, выведется сообщение об этом и старая хэш-таблица)

vf //элемент для исключения

Выходные данные изображены на рисунке 4.

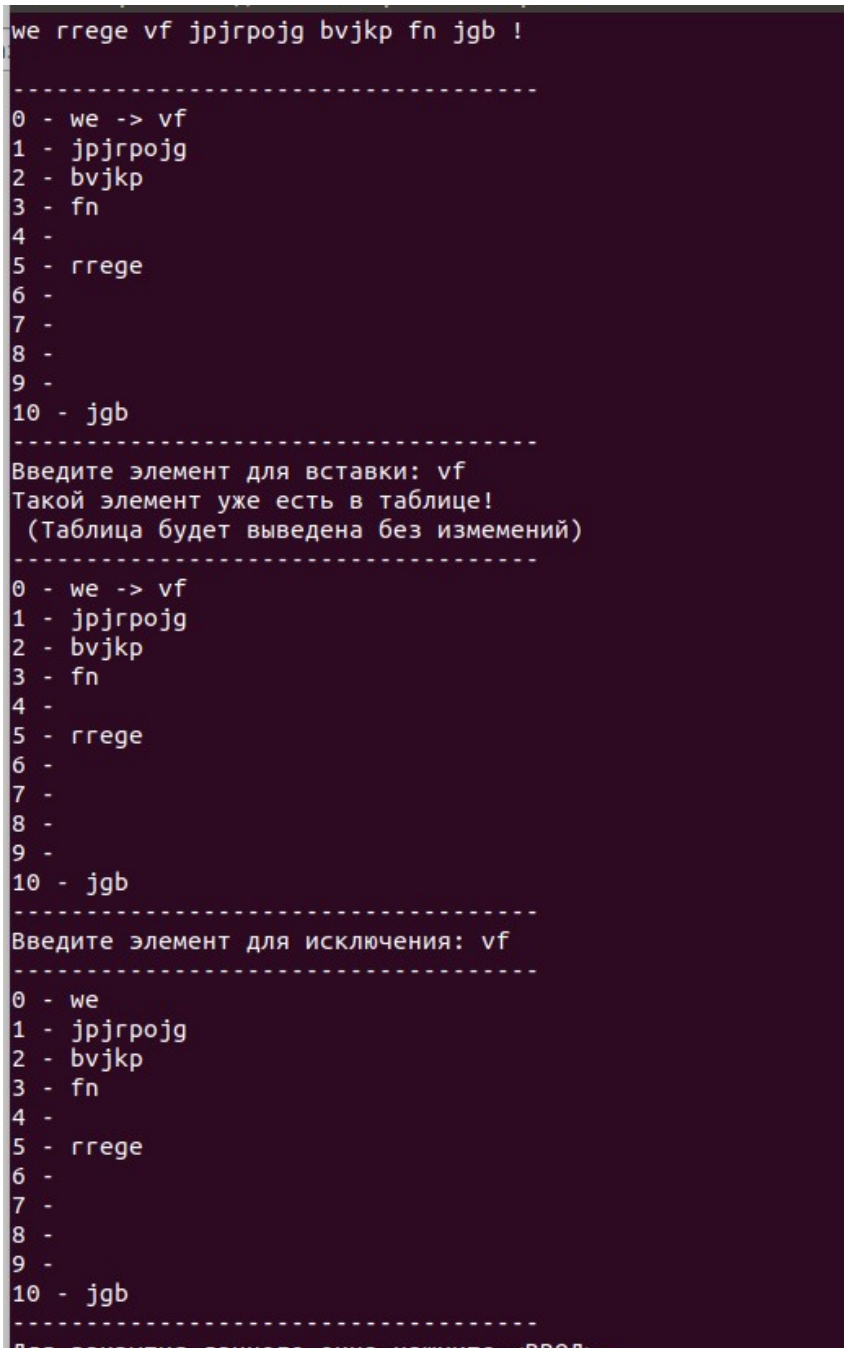

Рисунок 4 — Демонстрация работы программы с входными данными №3

5.4. Тестирование 4

Входные данные: qwe fdbfd qwert bnfdk hnl mnh vhid wfewf egw !

(Считывание элементов до символа «!»)

s235 //элемент для вставки

wfewf //элемент для исключения

Выходные данные изображены на рисунке 5.

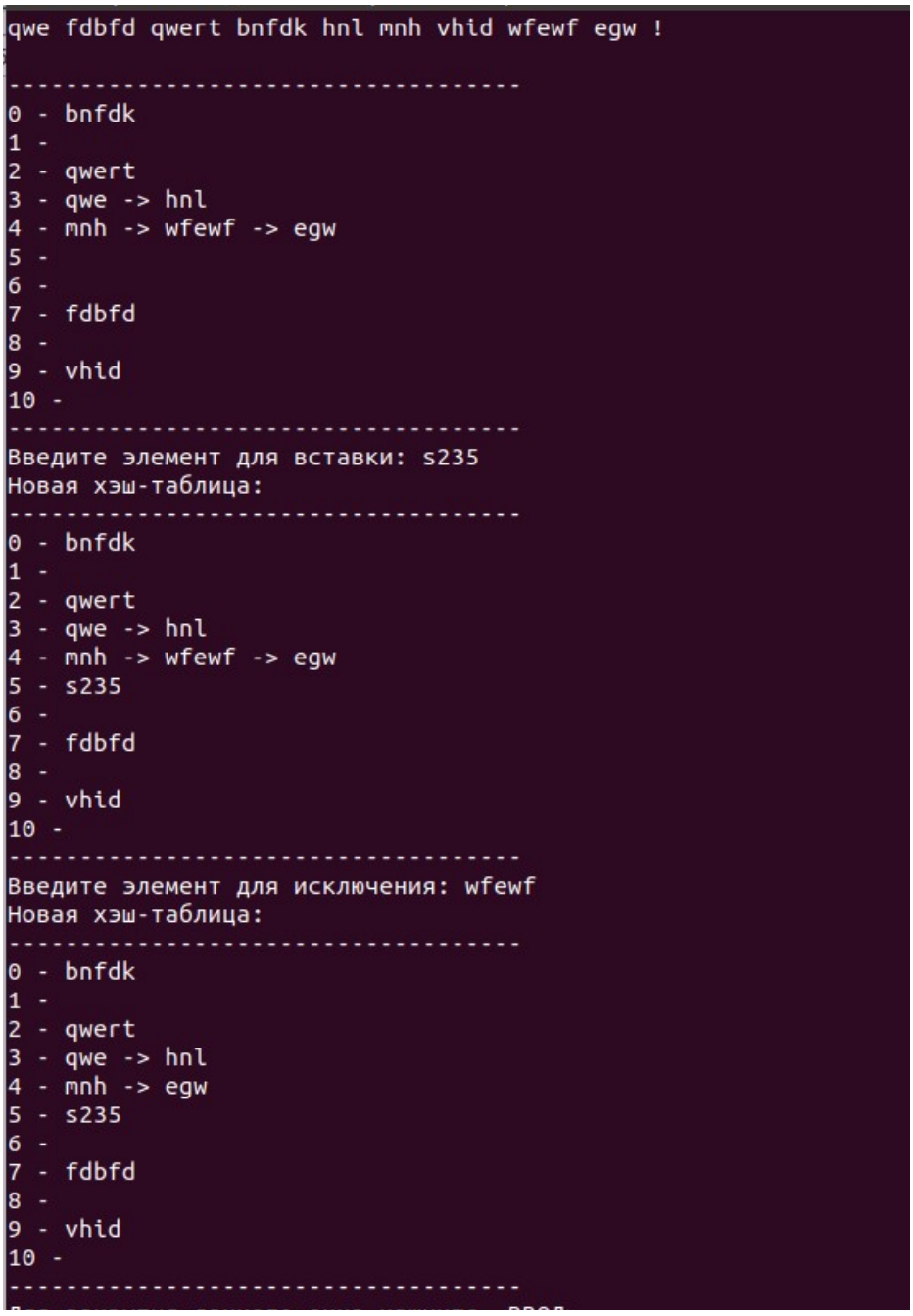

Рисунок 5 — Демонстрация работы программы с входными данными №4

ЗАКЛЮЧЕНИЕ

Была выполнена основная цель курсовой работы — продемонстрирован алгоритм вставки и исключения в хэш-таблицу с цепочками.

Для реализации данной цели были выполнены следующие **задачи:**

1. Изучены теоретические сведения по изучаемой структуре данных (хеш-таблицам).
2. Изучены теоретических сведения о функциях вставки и исключения.
3. Написана программа на языке C++, реализующую заданную структуру данных и необходимые функции.
4. Протестирована программа.
5. Выполнена отладка программы.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Описание электронного ресурса // Википедия. URL: https://ru.wikipedia.org/wiki/Хеш-таблица (дата обращения: 25.11.2020).
2. Описание электронного ресурса // Хабр. URL: https://habr.com/ru/post/509220/ (дата обращения: 25.11.2020)
3. Описание электронного ресурса // Algolist. URL: https://algolist.manual.ru/ds/s_has.php (дата обращения: 25.11.2020)
4. Описание электронного ресурса // Викиконспекты. URL: https://neerc.ifmo.ru/wiki/index.php?title=Хеш-таблица (дата обращения: 25.11.2020)
5. Описание электронного ресурса // Floppyy. URL: https://floppyy.ru/2016/08/24/hash-tables/ (дата обращения: 25.11.2020)
6. Описание электронного ресурса // K-press. URL: http://www.kpress.ru/cs/2000/4/bintree_htm/hash.asp (дата обращения: 25.11.2020)
7. Описание электронного ресурса // Github. URL: https://github.com/ilyamoskalev/HashTable (дата обращения: 19.12.2020)
8. Описание электронного ресурса // Genius. UML: http://genius.pstu.ru/file.php/1/pupils_works_2017/MuhinaAlisa.pdf (дата обращения: 19.12.2020)
9. Описание электронного ресурса // Викиконспекты. URL: https://neerc.ifmo.ru/wiki/index.php?title=Разрешение_коллизий (дата обращения: 19.12.2020)
10. Описание электронного ресурса // Хабр. URL: https://habr.com/ru/post/93226/ (дата обращения: 19.12.2020)
11. Описание электронного ресурса // Cryptoperson. URL: https://cryptoperson.ru/cryptography/chto-takoe-hjesh-kod-i-hjesh-funkcijaprakticheskoe-primenenie-obzor-populjarnyh-algoritmov (дата обращения: 19.12.2020)

ПРИЛОЖЕНИЕ А

РАЗРАБОТАННЫЙ КОД

Название файла main.cpp

```
#include <iostream>
#include <cstring>
#define N 11

using namespace std;

class Elem{
public:
   string key[10];
   int value[10];
   Elem* next;
   Elem* pred;

   void start(){
   for (int i = 0; i < 10; i++){
         key[i] = "NULL";
   }
   }

 };

void print(Elem hash[N]){
    cout << "-----------------------------------\n";
    for (int i = 0; i<N; i++){
          cout<<i<<" - ";
          int j =0;
          while (hash[i].key[j]!="NULL"){
          cout<<hash[i].key[j];
          if (hash[i].key[j+1]!="NULL")
              cout<<" -> ";
          j++;
          }
          cout<<"\n";
    }
    cout << "-----------------------------------\n";
}

int func(string s){
    int i = 0;
    int res = 0;
    while (s[i]!='\0'){
        res = (int)s[i]+res;
        i++;
    }
    res = res % N;
    return res;
}

void insert(Elem hash[N]){
```

```
    cout << "Введите элемент для вставки: ";
    string find_key;
    cin >> find_key;
    int f = func(find_key);
    int j = 0;
    int check = 0;
    while (hash[f].key[j]!="NULL"){
        if (hash[f].key[j]==find_key)
            check = 1;
        j++;
    }
    if (check==0){
        hash[f].key[j] = find_key;
        cout<<"Новая хэш-таблица:\n";
    }
    else {
        cout<<"Такой элемент уже есть в таблице!\n (Таблица будет выведена без
измемений)\n";
    }
    print(hash);
}

void deletion(Elem hash[N]){
    cout << "Введите элемент для исключения: ";
    string find_key;
    cin >> find_key;
    int f = func(find_key);
    int j = 0;
    int E = -1;
    while (hash[f].key[j]!="NULL"){
        if (hash[f].key[j]==find_key)
            E=j;
        j++;
    }
    if (E<0)
        cout << "Такого элемента в таблице нет! Удалять нечего!\n (Таблица будет
выведена без измемений)\n";
    else {
        if (hash[f].key[E+1]=="NULL")
            hash[f].key[E] = "NULL";
        else{
            while(hash[f].key[E]!="NULL"){
                hash[f].key[E] = hash[f].key[E+1];
                E++;
            }
            cout<<"Новая хэш-таблица:\n";
        }
    }

    print(hash);
}

int main()
{
    Elem hash[N];
    int i = 0;
```

```
    for(i = 0; i<N; i++){
        hash[i].start();
    }
    string s;
    i =0;
    cin>>s;
    int j = 0;
    while(s != "!"){
        i = func(s);
        j = 0;
        if (hash[i].key[j]=="NULL"){
            hash[i].key[j] = s;
        }
        else{
            while (hash[i].key[j]!="NULL")
                j++;
            hash[i].key[j] = s;
        }
        cin>>s;

    }
    cout << "\n";

    print(hash);
    insert(hash);
    deletion(hash);
    return 0;
}
```