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

КУРСОВАЯ РАБОТА

по дисциплине «Алгоритмы и структуры данных»

Тема: Демонстрация хеш – таблицы с цепочками

Студент гр. 9304 _____ Сорин А.В.

Преподаватель _____ Филатов А.Ю.

Санкт-Петербург

2020

ЗАДАНИЕ НА КУРСОВУЮ РАБОТУ

Студент Сорин А.В.

Группа 9304

Тема работы: демонстрация хеш – таблицы с цепочками

Исходные данные:
хеш – таблица с цепочками, язык программирования C++.

Содержание пояснительной записки:
«Содержание», «Задание», «Основные теоретические положения», «Описание структур данных и используемых функций», «Описание интерфейса пользователя», «Тестирование», «Заключение»

Предполагаемый объем пояснительной записки:
Не менее 17 страниц.

Дата выдачи задания: 23.11.2020

Дата сдачи реферата: 28.12.2020

Дата защиты реферата: 28.12.2020

Студент Сорин А.В.

Преподаватель Филатов А.Ю.

АННОТАЦИЯ

В данной работе была реализована программа, выполняющая построение хеш – таблицы с цепочками, а также вставку новых элементов и удаление старых по ключу.
Вставка и удаление сопровождаются пояснениями и демонстрацией таблицы.

SUMMARY

In this work, a program was implemented that builds a hesh – table with chains, as well as inserting new elements and deleting old ones by key. Insertion and deletion are accompanied by an explanation and demonstration of the table.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Цель работы

1. Реализация программы для демонстрации алгоритма вставки и удаления элементов из хеш – таблицы с цепочками, чтобы на каждом шаге пользователю давалось пояснение, что происходит в данный в момент и какой алгоритм используется.

2. Визуализация структур данных.

Задание

Вариант 23. Хеш – таблица с цепочками – вставка и исключение. Демонстрация. (Визуализация хеш – таблицы)

Предусмотреть возможность повторного выполнения с другим элементом.

Пояснение задания

Требуется: Продемонстрировать хеш – таблицу, вставку и удаление элементов с пояснениями. Демонстрация должна быть подробной и понятной.

1. ОСНОВНЫЕ ТЕОРЕТИЧЕСКИЕ ПОЛОЖЕНИЯ

1.1. Хеш – таблица с цепочками

Хеш – таблица с цепочками – это таблица для хранения данных, которая имеет следующую структуру:

Был реализован вариант с таблицей индексов. Все элемены хранятся в массиве в порядке добавления. Элементы, имеющие одинаковое число, возвращаемое хеш – функцией, связываются в цепочки:

- Номер первого элемента хранится в таблице индексов
- Номер любого другого элемента цепочки хранится в специальном поле предыдущего элемента

1.2. Вставка элементов

Получаем для ключа число из хеш – функции. Затем идем по цепочке пока не встретим последний элемент этой цепочки. Добавляем новый элемент в конец массива и сохраняем его номер в элементе, который был последним в цепочке. Если элементов цепочки не было, то добавляем элемент в конец таблицы и записываем его номер в таблицу индексов.

1.3. Удаление элементов

Получаем для ключа число из хеш – функции. Идем по цепочке, пока не встретим число с искомым ключем. Если элемент первый, то записываем в таблицу индексов вместо его номера номер следующего за ним элемента. Иначе записываем в его предыдущий элемент вместо его номера номер следующего за ним элемента. Затем надо уменьшить все числа, которые не меньше номера удаляемого элемента, в таблице индексов и в таблице элементов на 1.

2. ОПИСАНИЕ СТРУКТУР ДАННЫХ И ИСПОЛЬЗУЕМЫХ ФУНКЦИЙ

- Класс key_line – Реализация элемента таблицы данных
- Поле Name класса key_line – хранит ключ элемента. Ключ должен быть не больше заранее заданного размера.
- Поле Data класса key_line – хранит данные, связанные с ключем
- Поле Chain класса key_line – хранит номер следующего элемента в цепочке. Chain = -1 означает, что следующий элемент отсутствует
- 3 конструктора key_line класса key_line – создают элемент.
- Класс hesh_table – Реализация таблицы индексов и таблицы данных.
- Поле HeshTableSize класса hesh_table – максимальный размер таблицы индексов. Используется хеш – функцией.
- Поле KeySize класса hesh_table – Максимальный размер ключа. Нужен для вывода таблицы.
- Поле KeyTable класса hesh_table – таблица с данными. Реализована через стандартный контейнер std::vector
- Поле HeshTable класса hesh_table – таблица индексов. Реализована через умный указатель.
- Конструктор hesh_table класса hesh_table – создает пустую таблицу и заполняет начальные значения.
- Метод AddToHeshTable – добавляет элемент в таблицу. Для этого считается значение хеш – функции и элемент либо начинает новую цепочку, либо вставляется в уже имеющуюся.
- Метод DelFromHeshTable – удаляет элемент из таблицы. Для этого считается значение хеш – функции и ищет элемент с удаляемым ключем в нужной цепочке..
- Метод Ord – возвращает номер символа.
- Метод H – возвращает значение хеш – функции.

- Метод ReadKey – считывание ключа из потока.
- Функция GetLine – считывание линии из потока
- Функция operator<< – функция вывода хеш – таблицы на экран.

В данной реализации хеш – таблица реализована через таблицу индексов. Это очень удобно ведь это позволяет иметь фиксированную таблицу индексов, которая отражает количество возможных значений, возвращаемых хеш – функцией, и неограниченный динамический массив – таблицу с данными. При добавлении большого количества элементов в таком случае не придется перевыделять память, а можно просто добавлять элементы в конец, увеличивая размеры цепочек.

```
// класс - хеш-таблица
class hesh_table {
public:
    // размер таблицы индексов
    unsigned int HeshTableSize = 11;
    // максимальный размер ключа
    unsigned int KeySize = 8;
    // таблица с данными
    std::vector<key_line> KeyTable;
    // таблица индексов
    std::unique_ptr<int[]> HeshTable;
    // конструктор
    hesh_table(unsigned int HTS, unsigned int KS);
    // добавление элемента в хеш-таблицу
    int AddToHeshTable(std::string NewName, std::string NewData);
    // удаление элемента из хеш-таблицы
    int DelFromHeshTable(std::string DelName);
    // возвращает номер символа
    int Ord(char C);
    // хеш-функция
    int H(std::string Name);
    // считывание ключа
    void ReadKey(std::string& Key, std::stringstream& Stream);
};
```

Рисунок 1 – Класс хеш – таблицы

3. Описание интерфейса пользователя

Можно ввести enter для выхода из программы.

Чтобы добавить данные в таблицу или удалить из нее, надо ввести ключ и нажать enter.

Посте этого чтобы удалить элемент с введенным ключем, надо нажать enter.

Чтобы добавить элемент, надо ввести данные и нажать enter.

Ключ состоит из символов ‘0’…’9’, ‘a’…’z’ или ‘A’…’Z’.

Данные состоят из любых символов.

4. ТЕСТИРОВАНИЕ

4.1. Описание интерфейса.

```
1)Press enter to exit
2)To delete or add an element, enter the key and press enter
3)To delete an element, press enter after entering the key
3)To add an element, enter the data after entering the key. Press enter

The key consists of characters '0'...'9', 'a'...'z' or 'A'...'Z'
The data consists of any characters

1)
```

4.2. Добавление элемента.

```
Index table:

|-1|-1|-1|-1|-1|

Data table:
|chain|key|Data|


Hesh function number: 3
Index number 3: -1->0
Added element number: 0
Number of element meetings: 0

Index table:

|-1|-1|-1| 0|-1|

Data table:
|chain|key|Data|

|-1|       ex1|fhfhfhf|
```

Рисунок 2 – Дабавлен 1 элемент.

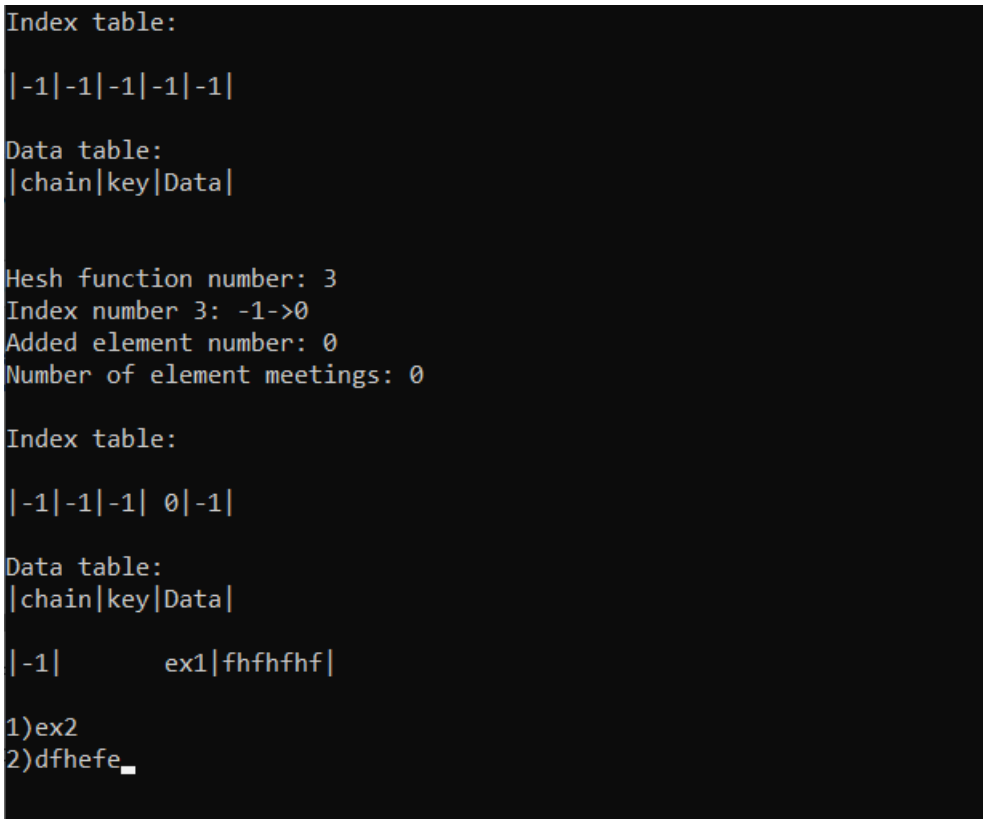

Рисунок 3 – Добавление 2 элемента

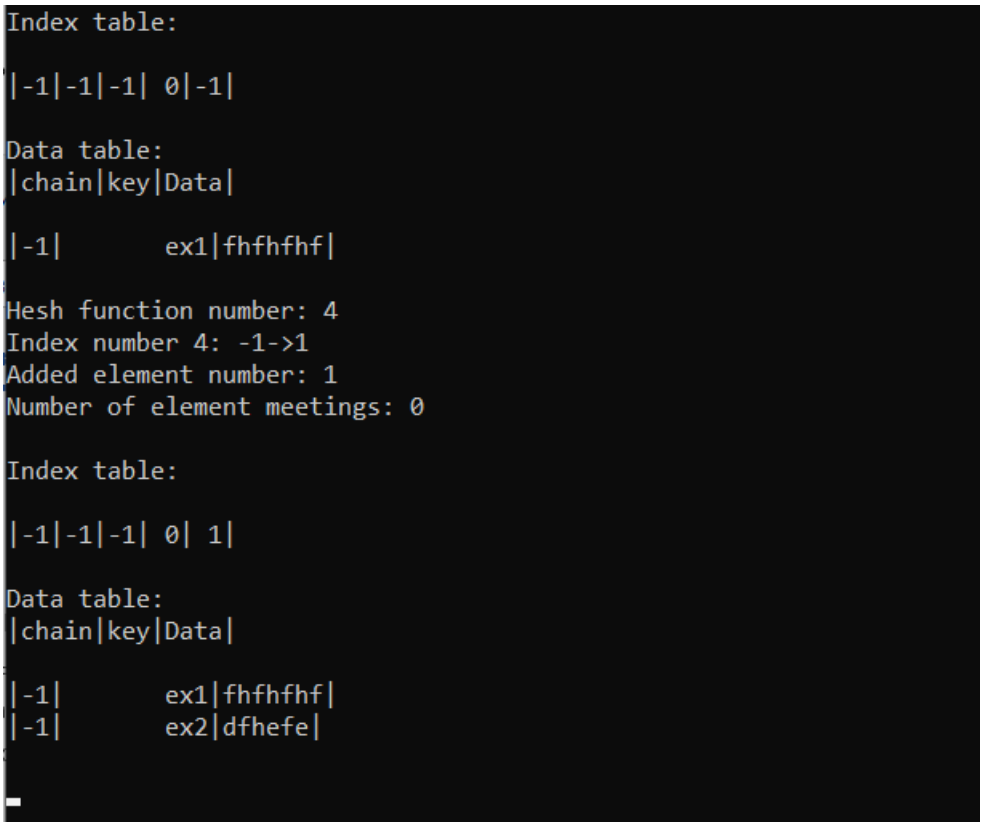

Рисунок 4 – Добавлен 2 элемент

```
Index table:

|-1|-1|-1| 0| 1|

Data table:
|chain|key|Data|

|-1|       ex1|fhfhfhf|
|-1|       ex2|dfhefe|

Hesh function number: 4
Element number: 1
Element chein: -1
Chain element number 1: -1->2
Added element number: 2
Number of element meetings: 0

Index table:

|-1|-1|-1| 0| 1|

Data table:
|chain|key|Data|

|-1|       ex1|fhfhfhf|
| 2|       ex2|dfhefe|
|-1|       ex7|num3|
```

Рисунок 5 – Добавлен 3 элемент

```
Index table:

|-1|-1|-1| 0| 1|

Data table:
|chain|key|Data|

|-1|       ex1|fhfhfhf|
| 2|       ex2|dfhefe|
|-1|       ex7|num3|

Hesh function number: 0
Index number 0: -1->3
Added element number: 3
Number of element meetings: 0

Index table:

| 3|-1|-1| 0| 1|

Data table:
|chain|key|Data|

|-1|       ex1|fhfhfhf|
| 2|       ex2|dfhefe|
|-1|       ex7|num3|
|-1|eeeexxxxx9|num4|
```

Рисунок 6 – Добавлен 4 элемент

4.3. Удаление элемента.

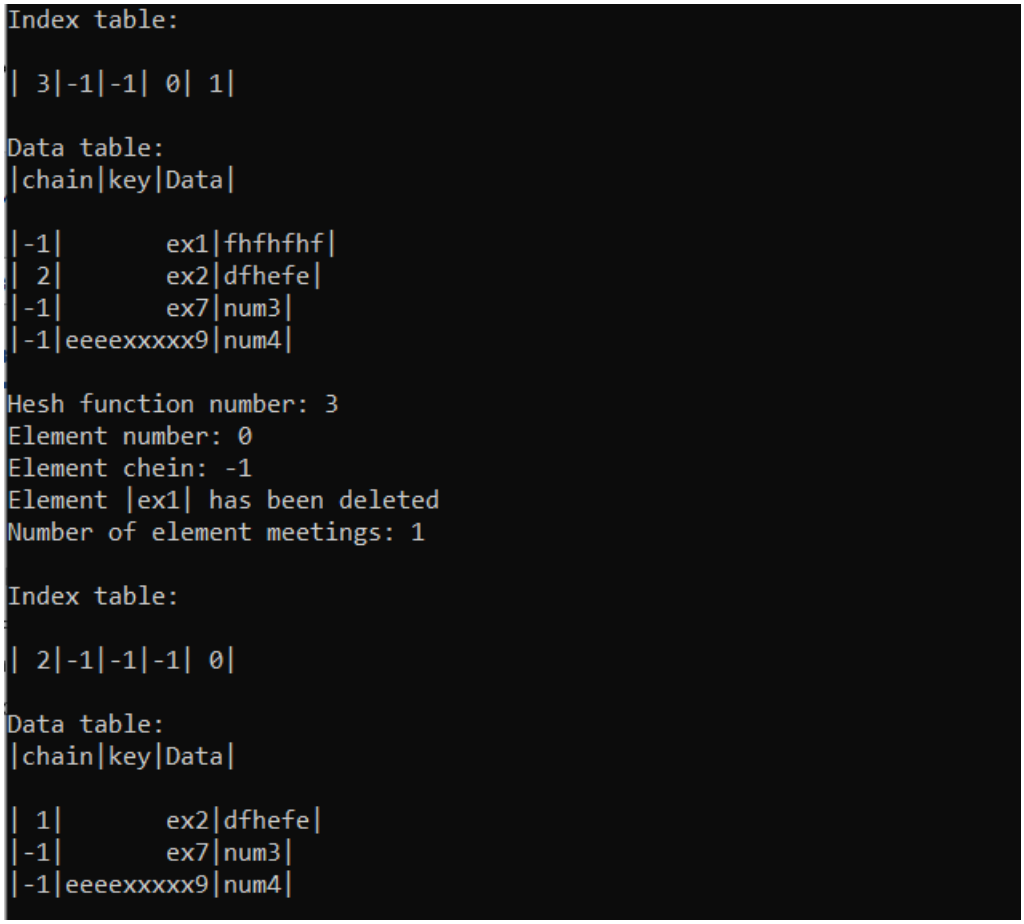

Рисунок 7 – Удален 1 элемент

```
Index table:

| 3|-1|-1| 0| 1|

Data table:
|chain|key|Data|

|-1|       ex1|fhfhfhf|
| 2|       ex2|dfhefe|
|-1|       ex7|num3|
|-1|eeeexxxxx9|num4|

Hesh function number: 3
Element number: 0
Element chein: -1
Element |ex1| has been deleted
Number of element meetings: 1

Index table:

| 2|-1|-1|-1| 0|

Data table:
|chain|key|Data|

| 1|       ex2|dfhefe|
|-1|       ex7|num3|
|-1|eeeexxxxx9|num4|

1)ex7
```

Рисунок 8 – Удаление 2 элемента

```
Index table:

| 2|-1|-1|-1| 0|

Data table:
|chain|key|Data|

| 1|       ex2|dfhefe|
|-1|       ex7|num3|
|-1|eeeexxxxx9|num4|

Hesh function number: 4
Element number: 0
Element chein: 1
Element number: 1
Element chein: -1
Element |ex7| has been deleted
Number of element meetings: 1

Index table:

| 1|-1|-1|-1| 0|

Data table:
|chain|key|Data|

|-1|       ex2|dfhefe|
|-1|eeeexxxxx9|num4|
```

Рисунок 9 – Удален 2 элемент

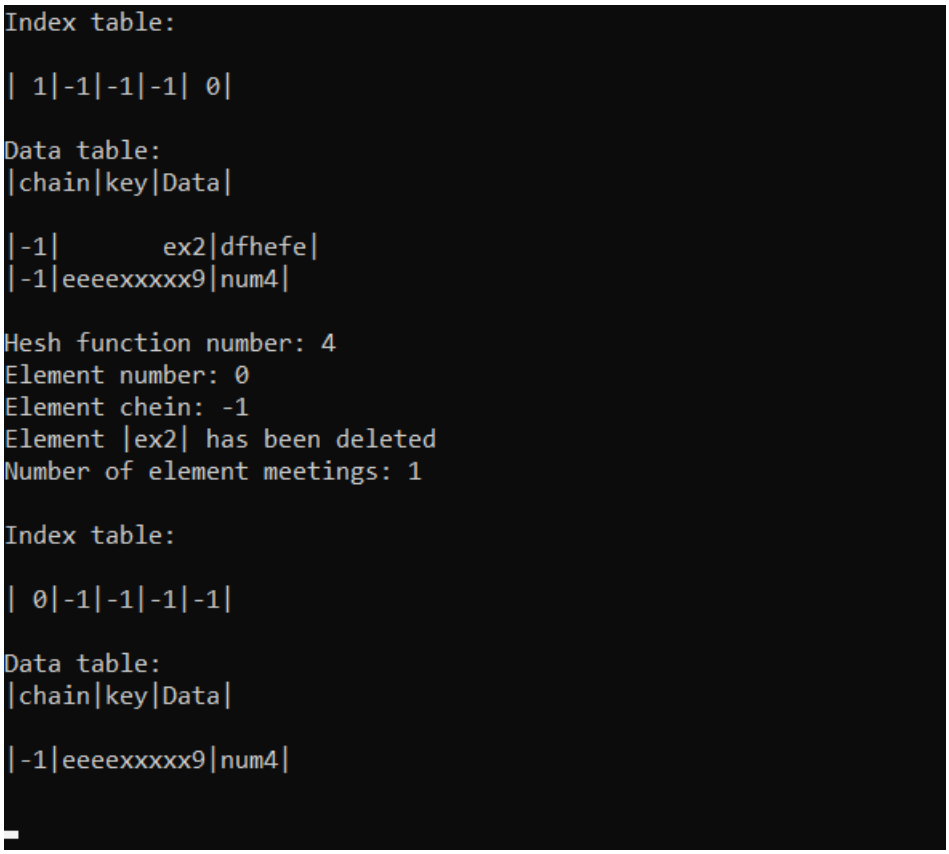

Рисунок 10 – Удален 1 элемент

4.4. Пример неправильного ввода.

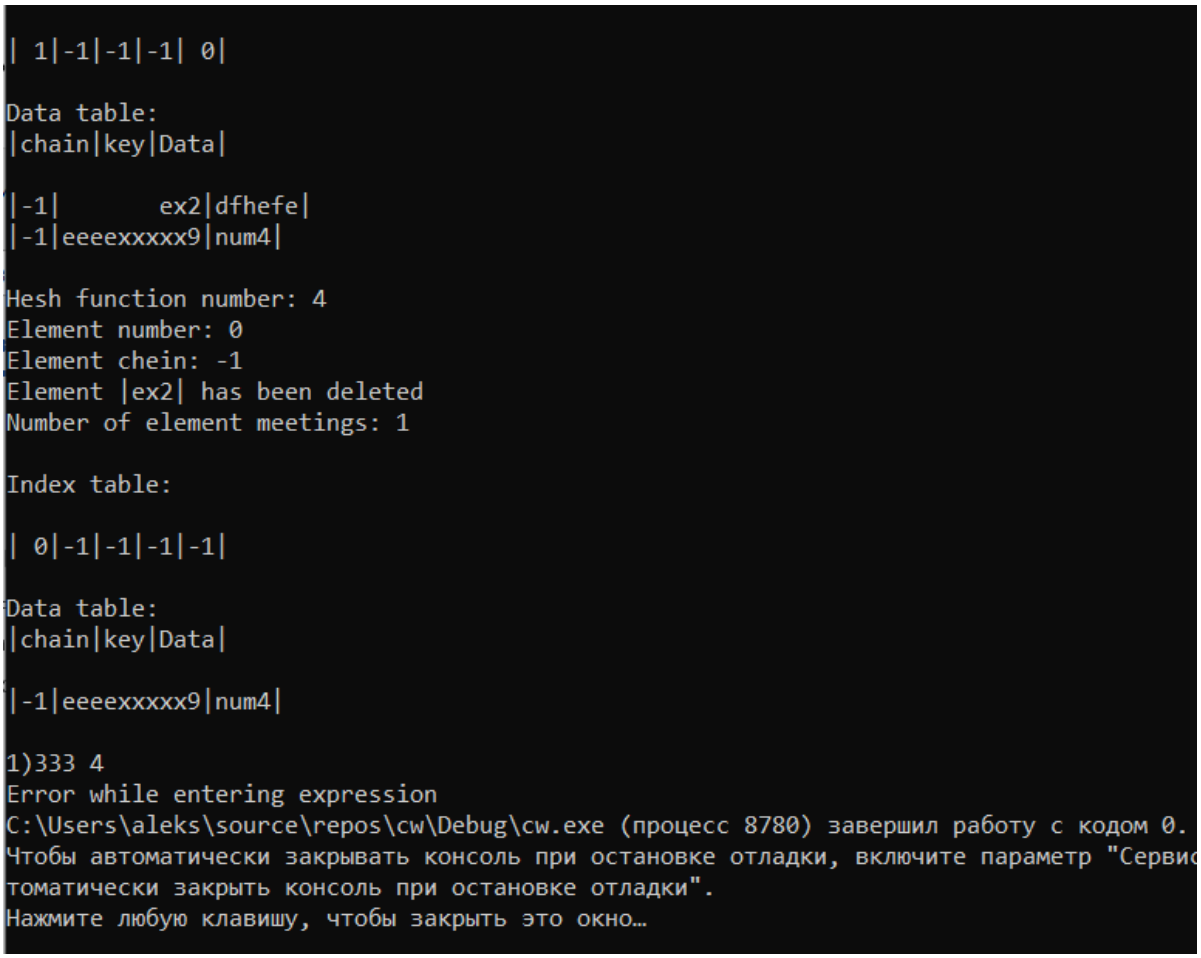

Рисунок 11 – Пример неправильного ввода

ЗАКЛЮЧЕНИЕ

Были изучена хеш – таблица с цепочками, функции для работы с ней: вставка и удаление элементов по ключу. Решена задача реализации хеш – таблицы с цепочками на C++, а также визуализации структур данных, которая сопровождается пояснениями.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД

Файл main.cpp:

```
#include <string>
#include "hesh.h"

// считывание линии из потока
std::stringstream GetLine(void) {
	std::string Str;
	if (!std::getline(std::cin, Str))
		throw std::runtime_error("Error while reading from stream");
	std::stringstream Stream(Str);
	return Stream;
}

// перегрузка оператора вывода для вывода хеш-таблицы
std::ostream& operator<<(std::ostream& os, hesh_table& HT) {
	// вывод таблицы индексов
	os << "Index table:\n\n";
	os << '|';
	for (int it = 0; it < HT.HeshTableSize; it++) {
		os.width(2);
		os << HT.HeshTable[it] << '|';
	}
	os << "\n\n";
	// вывод таблицы с данными и цепочками
	os << "Data table:\n|chain|key|Data|\n\n";
	for (int it = 0; it < HT.KeyTable.size(); it++) {
		os << '|';
```

```
            os.width(2);
            os << HT.KeyTable[it].Chain;
            os << '|';
            std::string s = HT.KeyTable[it].Name;
            os.width(HT.KeySize);
            os << s;
            os << '|';
            os << HT.KeyTable[it].Data;
            os << '|';
            os << '\n';
        }
        return os << '\n';
}

int main() {
        try
        {
            // размер таблицы индексов и максимальный размер ключа
            unsigned int HTS = 5, MKS = 10;

            // создание пустой таблицы
            hesh_table HT(HTS, MKS);
            int Res = 0;
            std::string Key;
            while (1) {
                Key = "";
                std::stringstream Stream = GetLine();
                char C = 0;
                if (Stream.get(C).eof())
```

```
                    break;
                if (C != '1')
                    throw std::invalid_argument("Error while entering
expression");
                if ((C = Stream.get()) != ')')
                    throw std::invalid_argument("Error while entering
expression");
                // считывание ключа
                HT.ReadKey(Key, Stream);
                // считывание данных
                std::stringstream Stream2 = GetLine();
                if (Stream2.get(C).eof()) {
                    // чистка экрана
                    std::cout << "\x1B[2J\x1B[H";
                    // вывод самой таблицы
                    std::cout << HT;
                    // удаление элемента из таблицы
                    Res = HT.DelFromHeshTable(Key);
                }
                else {
                    std::string Data;
                    if ((C) != '2')
                        throw std::invalid_argument("Error while entering
expression");
                    if ((C = Stream2.get()) != ')')
                        throw std::invalid_argument("Error while entering
expression");
                    Data = Stream2.str().erase(0, 2);
                    // чистка экрана
                    std::cout << "\x1B[2J\x1B[H";
                    // вывод самой таблицы
```

```
                        std::cout << HT;
                        // добавление элемента в таблицу
                        Res = HT.AddToHeshTable(Key, Data);
                }
                // информация о колличестве встреченных элементов
                std::cout << "Number of element meetings: " << Res << "\n\n";
                // вывод самой таблицы
                std::cout << HT;
        }

    }
    catch (const std::exception& Error) // отлавливание ошибок
    {
        // вывод информации об ошибке
        std::cout << Error.what();
    }
    return 0;
}
```

Файл hesh.cpp:

```
#include "hesh.h"

key_line::key_line() {

}
key_line::key_line(std::string NewName, std::string NewData) : Name(NewName),
Data(NewData) {

}
```

```
key_line::key_line(std::string NewName, std::string NewData, int NewChain) :
Name(NewName), Data(NewData), Chain(NewChain) {


}


hesh_table::hesh_table(unsigned int HTS, unsigned int KS) : HeshTableSize(HTS),
KeySize(KS) {
	// выделение памяти для таблицы индексов
	std::unique_ptr<int[]> Fi(new int[HeshTableSize]);
	HeshTable = std::move(Fi);
	for (int i = 0; i < HeshTableSize; i++)
		HeshTable[i] = -1;
}
int hesh_table::AddToHeshTable(std::string NewName, std::string NewData) {
	int N = H(NewName);
	// число хеш-функции
	std::cout << "Hesh function number: " << N << '\n';
	if (HeshTable[N] == -1) {
		// если не было элементов с таким числом
		key_line KL(NewName, NewData);
		KeyTable.push_back(KL);
		HeshTable[N] = KeyTable.size() - 1;
		std::cout << "Index number " << N << ": " << -1 << "->" << HeshTable[N]
<< '\n';
		std::cout << "Added element number: " << HeshTable[N] << '\n';
		return 0;
	}
	else {
		int Count = 0;


		int SaveInd = HeshTable[N];
```

```
                std::cout << "Element number: " << SaveInd << '\n';
                std::cout << "Element chein: " << KeyTable[SaveInd].Chain << '\n';
                if (KeyTable[SaveInd].Name == NewName) {
                        Count++;
                        std::cout << "Element meeting: count = " << Count << '\n';
                }
                // переход дальше по цепочке
                while (KeyTable[SaveInd].Chain != -1) {
                        SaveInd = KeyTable[SaveInd].Chain;
                        std::cout << "Element number: " << SaveInd << '\n';
                        std::cout << "Element chein: " << KeyTable[SaveInd].Chain << '\n';
                        if (KeyTable[SaveInd].Name == NewName) {
                                Count++;
                                std::cout << "Element meeting: count = " << Count << '\n';
                        }
                }
                // добавление элемента
                KeyTable[SaveInd].Chain = KeyTable.size();
                std::cout << "Chain element number " << SaveInd << ": " << -1 << "->" <<
KeyTable[SaveInd].Chain << '\n';
                std::cout << "Added element number: " << KeyTable[SaveInd].Chain << '\n';
                key_line KL(NewName, NewData);
                KeyTable.push_back(KL);
                return Count;
        }
}
int hesh_table::DelFromHeshTable(std::string DelName) {
        int N = H(DelName);
        // число хеш функции
        std::cout << "Hesh function number: " << N << '\n';
        if (HeshTable[N] == -1) {
```

```
            std::cout << "Element to be deleted is missing\n";
            return 0;
    }
    else {
            int SaveSaveInd = 0;

            int SaveInd = HeshTable[N];
            std::cout << "Element number: " << SaveInd << '\n';
            std::cout << "Element chein: " << KeyTable[SaveInd].Chain << '\n';
            // если удаляется первый элемент цепочки
            if (KeyTable[SaveInd].Name == DelName) {
                    HeshTable[N] = KeyTable[SaveInd].Chain;
                    KeyTable.erase(KeyTable.begin() + SaveInd);
                    for (int i = 0; i < HeshTableSize; i++)
                            if (HeshTable[i] >= SaveInd)
                                    HeshTable[i]--;
                    for (int i = 0; i < KeyTable.size(); i++)
                            if (KeyTable[i].Chain >= SaveInd)
                                    KeyTable[i].Chain--;
                    std::cout << "Element |" << DelName << "| has been deleted" << '\n';
                    return 1;
            }
            while (KeyTable[SaveInd].Chain != -1) {
                    SaveSaveInd = SaveInd;
                    SaveInd = KeyTable[SaveInd].Chain;
                    std::cout << "Element number: " << SaveInd << '\n';
                    std::cout << "Element chein: " << KeyTable[SaveInd].Chain << '\n';
                    // если удаляется не первый элемент цепочки
                    if (KeyTable[SaveInd].Name == DelName) {
                            KeyTable[SaveSaveInd].Chain = KeyTable[SaveInd].Chain;
```

```
                                KeyTable.erase(KeyTable.begin() + SaveInd);
                                for (int i = 0; i < HeshTableSize; i++)
                                        if (HeshTable[i] >= SaveInd)
                                                HeshTable[i]--;
                                for (int i = 0; i < KeyTable.size(); i++)
                                        if (KeyTable[i].Chain >= SaveInd)
                                                KeyTable[i].Chain--;
                                std::cout << "Element |" << DelName << "| has been deleted"
<< '\n';
                                return 1;
                        }
                }
                std::cout << "Element to be deleted is missing\n";
                return 0;
        }
}
int hesh_table::Ord(char C) {
        if (std::isdigit(C))
                return C - '0';
        else if (C >= 'a' && C <= 'z') {
                return C - 'a' + 10; // 'a'...'z' стоят после цифр
        }
        else if (C >= 'A' && C <= 'Z') {
                return C - 'A' + 10 + 'z' - 'a' + 1;// // 'A'...'Z' стоят после 'a'...'z'
        }
        else
                throw std::invalid_argument("Error while entering expression");
}
int hesh_table::H(std::string Name) {
        int Sum = 0;
        // подсчет числа для ключа
```

```
	for (int i = 0; i < Name.size(); i++)
		Sum += Ord(Name[i]);
	int Mod = HeshTableSize;
	// получение числа хеш-функции для ключа
	Sum %= Mod;
	return Sum;
}
void hesh_table::ReadKey(std::string& Key, std::stringstream& Stream) {
	char C = 0;
	if (!Stream.get(C))
		throw std::invalid_argument("Error while entering expression");
	for (int i = 0; (std::isdigit(C) || (C >= 'a' && C <= 'z') || (C >= 'A' && C <= 'Z')) &&
(i < KeySize); i++) {
		Key += C;
		// выход если ключ закончился
		if (Stream.get(C).eof())
			return;
		if (Stream.fail())
			throw std::invalid_argument("Error while entering expression");
	}
	throw std::invalid_argument("Error while entering expression");
}
```

Файл hesh.h:

```
#ifndef __HESH_H
#define __HESH_H

#include <iostream>
#include <stdexcept>
#include <sstream>
#include <string>
#include <vector>
```

```
#include <chrono>
#include <thread>

// класс - одна строчка из таблицы с данными
class key_line {
public:
// конструкторы
key_line();
key_line(std::string NewName, std::string NewData);
key_line(std::string NewName, std::string NewData, int NewChain);
// ключ
std::string Name;
// данные
std::string Data;
// чепочка
int Chain = -1;
};

// класс - хеш-таблица
class hesh_table {
public:
// размер таблицы индексов
unsigned int HeshTableSize = 11;
// максимальный размер ключа
unsigned int KeySize = 8;
// таблица с данными
std::vector<key_line> KeyTable;
// таблица индексов
std::unique_ptr<int[]> HeshTable;
// конструктор
hesh_table(unsigned int HTS, unsigned int KS);
// добавление элемента в хеш-таблицу
int AddToHeshTable(std::string NewName, std::string NewData);
```

```
// удаление элемента из хеш-таблицы
int DelFromHeshTable(std::string DelName);
// возвращает номер символа
int Ord(char C);
// хеш-функция
int H(std::string Name);
// считывание ключа
void ReadKey(std::string& Key, std::stringstream& Stream);
};

#endif // __HESH_H
```