Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего образования

**«Пермский национальный исследовательский**

**политехнический университет»**

Электротехнический факультет

Кафедра «Информационные технологии и автоматизированные системы»

направление подготовки: 09.03.01 - «Информатика и вычислительная техника»

**Отчет по Лабораторной работе Хеш-таблицы**

**по дисциплине**

**«Основы алгоритмизации и программирования»**

Выполнил студент гр. ИВТ-21-1б

Ахунов Руслан Булатович

Проверил:

Ст. Преподаватель кафедры ИТАС

Яруллин Денис Владимирович

(оценка) (подпись)

г. Пермь - 2022

**Постановка задачи:**

1) Создать динамический массив из записей (в соответствии с вариантом),

содержащий не менее 100 элементов. Для заполнения элементов массива

использовать ДСЧ.

2) Выполнить поиск элемента в массиве по ключу в соответствии с вариантом. Для

поиска использовать хеш-таблицу.

3) Подсчитать количество коллизий при размере хеш-таблицы 40, 75 и 90

элементов.

Вариант 2

ФИО, №счета, сумма

№счета - ключ

H(k)= [M (kAmod1)], 0<A<1, mod1 – получение дробной части, [] – получение целой части

Метод открытой

адресации

**Анализ задачи:**

struct human

{

    string fullname = "Null";

    string num = "Null";

    string sum = "Null";

};

struct hash\_table

{

    human\* arr;

    hash\_table(int size)

    {

        arr = new human[size];

    }

    void add(human tmp, int size);

    void findIndex(string num, int size);

};

Структура human – Содержит в себе 3 поля под ФИО, номер счета и сумму.

Структура hash\_table – объявляется массив значения типа human, так же есть конструктор, который заполняется этот массив.

2 функции:

add – добавить новый элемент, передается значения типа human, а так же размер хеш-таблицы

findIndex – функция для нахождения человека в хеш-таблицы.

string rand\_name()

{

    return (postname[rand() % 100] + " " + names[rand() % 100] + " " + patr[rand() % 100]);

}

string rand\_num()

{

    return num[rand() % 100];

}

string rand\_sum()

{

    return sum[rand() % 100];

}

3 функция для рандомизации человека. Его ФИО, номера счета и суммы.

human create\_human()

{

    human tmp;

    tmp.fullname = rand\_name();

    tmp.num = rand\_num();

    tmp.sum = rand\_sum();

    return tmp;

}

void fill\_human(human\* arr, int count)

{

    for (int i = 0; i < count; i++)

    {

        arr[i] = create\_human();

    }

}

Функция create\_human – как понятно из названия создает нового человека, используя 3 предыдущие функции для рандомизации. Возвращается значения типа human.

Функция fill\_human – заполняет массив значениями типа human.

void show(const human tmp)

{

    cout << "ФИО:          " << tmp.fullname

         << "\nНомер счета:  " << tmp.num

         << "\nСумма:        " << tmp.sum << endl << endl;

}

void showTable(hash\_table\* table, int size)

{

    for (int i = 0; i < size; i++)

    {

        show(table->arr[i]);

    }

}

show – выводит в консоль значения трех полей: fullname, num, sum.

showTable – выводит на экран значения из хеш-таблицы.

int hash\_f(string str, int size)

{

    long double a = (0.68 \* stoi(str));

    double a1 = size \* (a - static\_cast<int>(a));

    return static\_cast<int>(a1);

}

Хеш-функция для нахождения значения индекса для хеш-таблицы. Формула была взята из условия задания.

void hash\_table::add(human tmp, int size)

{

    int index = hash\_f(tmp.num, size);

    int hash = index;

    if (arr[index].fullname == "Null")

    {

        arr[index] = tmp;

        return;

    }

    else

    {

        while (index < size)

        {

            if (arr[index].fullname == "Null")

            {

                arr[index] = tmp;

                return;

            }

            index++;

            collis++;

        }

        if (index >= size)

        {

            index = 0;

            collis++;

            while (index < hash)

            {

                if (arr[index].fullname == "Null")

                {

                    arr[index] = tmp;

                    return;

                }

                index++;

                collis++;

            }

        }

    }

}

Функция для добавления нового человека в хеш-таблицу. Так как по условию задачи нужно использовать метод открытой адресации, то мы проходим по всей таблице(массиву), пока не найдем свободного места, в этой время увеличивается значения счетчика коллизий.

void hash\_table::findIndex(string num, int size)

{

    int hash = hash\_f(num, size);

    int index = hash;

    while (arr[index].num != num && index < size)

    {

        index++;

    }

    if (index >= size)

    {

        index = 0;

        while (arr[index].num != num && index < hash)

        {

            index++;

        }

        if (index >= hash)

        {

            cout << "Такого человека нет";

        }

        else

        {

            cout << "Такой человек есть по индексу: " << index << endl << endl;

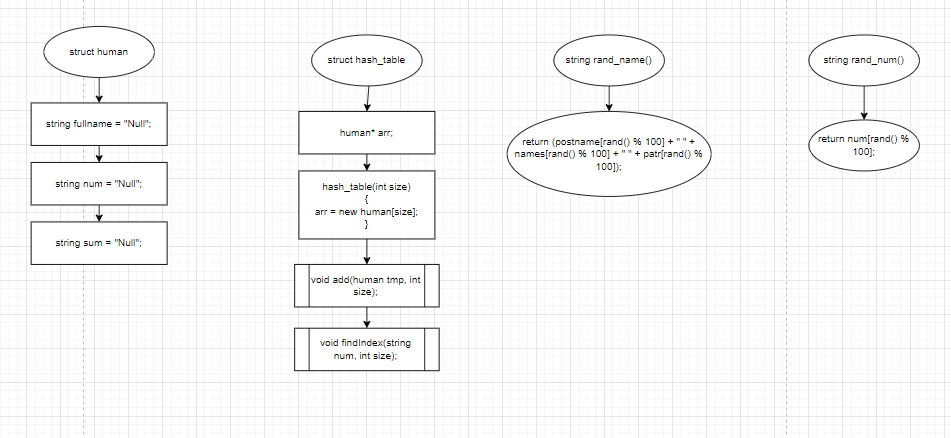
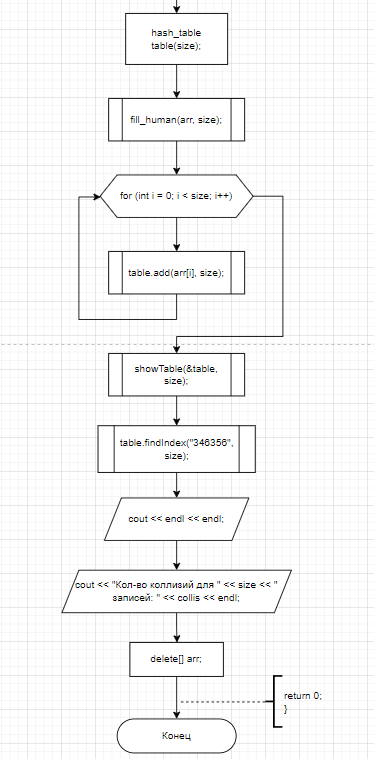
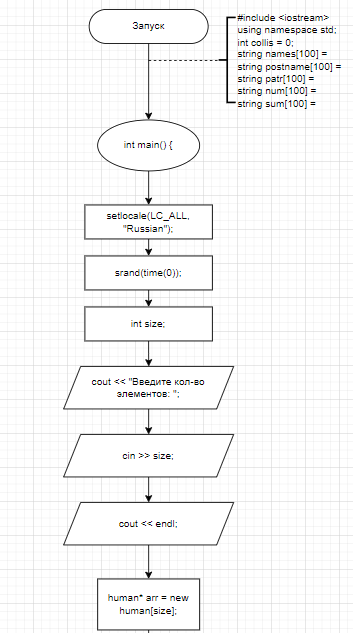
        }

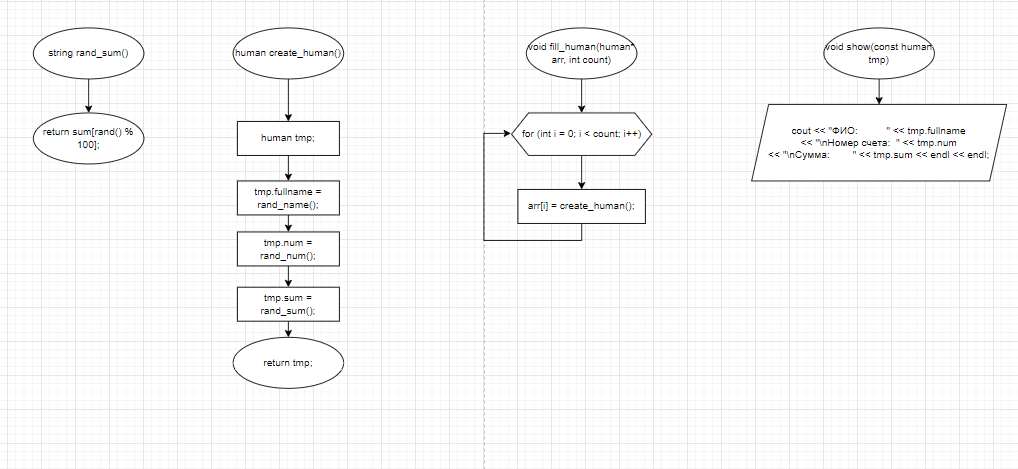
    }

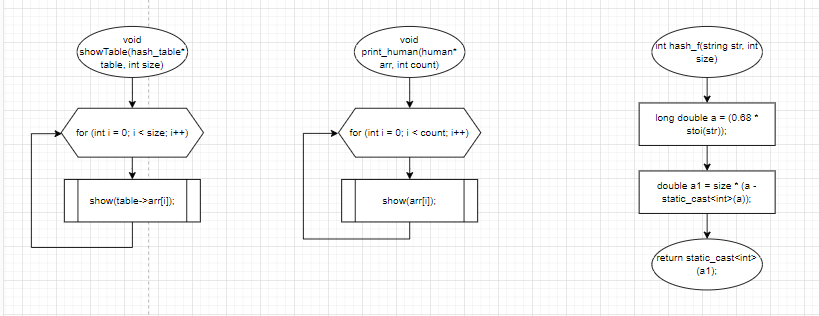
}

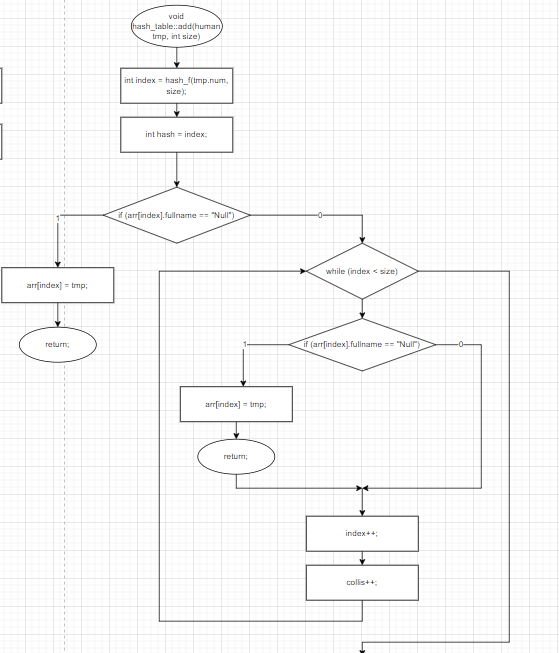
Функция поиска человека, все значения сперва уходят в хеш-функцию, чтобы получить индекс, и дальше идет проверка по номеру счета, подходит ли он или нет.

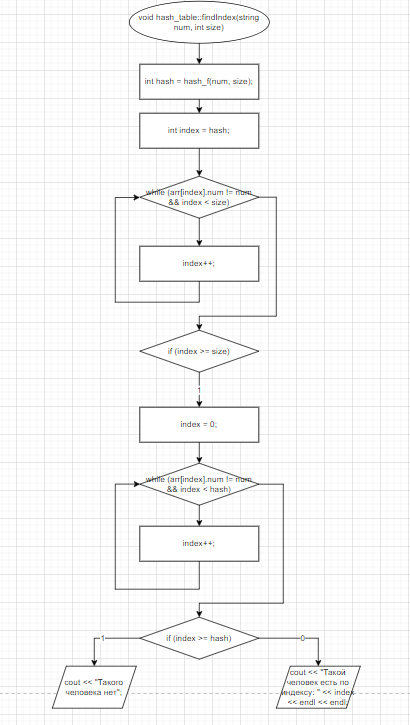
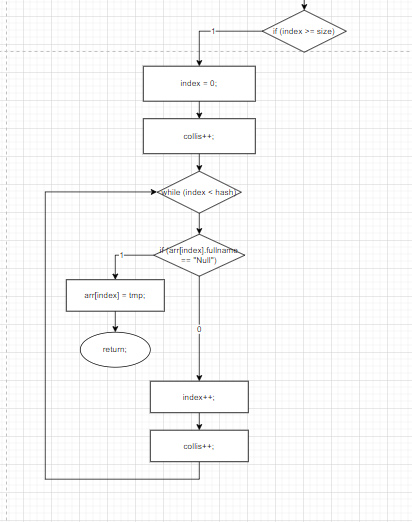
**Блок-схема**

****

****

****

****

****

**Код программы**

#include <iostream>

#include <string>

using namespace std;

int collis = 0;

struct human

{

    string fullname = "Null";

    string num = "Null";

    string sum = "Null";

};

struct hash\_table

{

    human\* arr;

    hash\_table(int size)

    {

        arr = new human[size];

    }

    void add(human tmp, int size);

    void findIndex(string num, int size);

};

string names[100] = { "Руслан", "Алексей", "Пол", "Артем", "Егор", "Александр", "Дмитрий", "Максим", "Сергей", "Андрей", "Илья", "Кирилл", "Михаил", "Матвей", "Роман", "Иван", "Ильяс", "Арсенний", "Денис", "Евгений", "Даниил", "Тимофей", "Владислав", "Павел", "Марк", "Константин", "Тимур", "Олег", "Ярослав", "Антон", "Николай", "Глеб", "Данил", "Савелий", "Вадим", "Степан", "Юрий", "Богдан", "Артур", "Семен", "Макар", "Лев", "Виктор", "Елисей", "Виталий", "Вячеслав", "Захар", "Мирон", "Дамир", "Георгий", "Давид", "Платон", "Анатолий", "Григорий", "Демид", "Данила", "Станислав", "Василий", "Федор", "Родион" , "Леонид", "Одиссей", "Валерий", "Святослав", "Борис", "Эдуард", "Марат", "Герман", "Даниэль", "Петр", "Амир", "Всеволод", "Мирослав", "Гордей", "Артемий", "Эмиль", "Назар", "Савва", "Ян", "Рустам", "Ингат", "Влад", "Альберт", "Тамерлан", "Айдар", "Роберт", "Марсель", "Ильдар", "Самир", "Тихон" , "Рамиль", "Ринат", "Радмир", "Филипп", "Арсен", "Ростислав", "Святогор", "Яромир", "Алдуин", "Пастернак" };

string postname[100] = { "Ланнистер", "Баратеон", "Талли", "Аррены", "Грейджой", "Бронн", "Мартелл", "Атрейдес", "Харконнен", "Гильдеец", "Смирнов", "Иванов", "Кузнецов", "Соколов", "Попов", "Лебедев", "Козлов", "Новиков", "Морозов", "Петров", "Волков", "Соловьев", "Васильев", "Зайцев", "Павлов", "Семенов", "Голубев", "Виноградов", "Богданов", "Влолбев", "Федоров", "Михайлов", "Беляев", "Тарасов", "Белов", "Комаров", "Тарасов", "Белов", "Комаров", "Орлов", "Киселев", "Андреев", "Макаров", "Гладиатор", "Ильин", "Лазарев", "Медведев", "Ершов", "Никитин", "Соболев", "Рябов", "Поляков", "Цветков", "Данилов", "Жуков", "Фролов", "Журавльев", "Драконорожденный", "Егоров", "Петухов" , "Тимофеев", "Власов", "Калорв", "Лапин", "Сазонов", "Гордеев", "Уваров", "Брагин", "Якушев", "Зыков", "Шарапов", "Рожков", "Самоснов", "Лихачев", "Щукин", "Шаров", "Сафонов", "Птеродактель", "Устинов", "Исаев", "Капустин", "Кириллов", "Рогов", "Князев", "Кулагин", "Логинов", "Савин", "Туров", "Блинов", "Королев" , "Пестов", "Меркушев", "Гущин", "Буров", "Галкин", "Коновалов", "Громов", "Мельников", "Давыдов", "Русаков" };

string patr[100] = { "Витальев", "Иванов", "Сергеев", "Владимиров", "Алексеев", "Мизаров", "Степанов", "Каладиев", "Булатович", "Лексеевич", "Джиравоич", "Лопарович", "Хорашивич", "Шолораович", "Корупович", "Щокурепич", "Защеливич", "Степонвич", "Жиданович", "Жолкавович", "Долгичвич", "Зоглаоррвич", "Ларенвич", "Апотьлиавич", "Логравич", "Хологрвич", "Хорекунивич", "Экуравоич", "Федорович", "Болкивич", "Ждаровкич", "Шорокывич", "Героувич", "Носорович", "Гордунвич", "Цекрович", "Шитохович", "Дорекфич", "Жирович", "Эколович", "Кардинович", "Можетвич", "Твичевич", "Ютубович", "Тровович", "Дискордович", "Скапович", "Вкович", "Карандошович", "Стрелкович", "Усталович", "Щеголович", "Викингович", "Ероваич", "Евроводич", "Бытович", "Жетович", "Цукович", "Парович", "Мирович" , "Часович", "Владимриович", "Оакович", "Лапович", "Ушович", "Хокунович", "Шоуканович", "Скорпирович", "Вальгалович", "Асасинович", "Котович", "Лопухович", "Автомобиволич", "Ипотивович", "Сахарович", "Можетович", "Хорошкович", "Началович", "Жароквоич", "Нулевич", "Можектович", "Ктокович", "Собакович", "Радорович", "Флешович", "Хартович", "Резонович", "Сезонович", "Ансерович" , "Коверович", "Пикович", "Глазович", "Асович", "Легионвич", "Шадоувич", "Энович", "Пиполович", "Медневич", "Концович" };

string num[100] = { "5127", "34653464", "4565", "1237", "3334", "2355", "45568", "664563", "345346", "5654", "34546", "3363453", "23537", "983445", "344645", "4575568", "346356", "457678", "56674", "236457", "456354", "347567", "347457", "2345345", "265467457", "13645754", "236455", "24845", "246457", "2546", "2673", "24634", "2543", "2637", "234754", "457", "347566", "2354745", "34463", "3475", "346568", "34546", "54734", "457546", "45678", "2379", "456857", "546", "34647465", "34768", "7865", "57665", "585764", "5465", "56665", "76643", "6746", "78463", "34353", "56435" , "3476", "56653", "6574354", "67465", "235435", "546564", "65746", "4552", "57457", "523225", "3483", "9795", "8375", "385", "091123", "887234", "3587", "85289", "892394", "2375", "2235", "5457", "3464", "34767", "76798", "565689", "2035", "723259", "89925", "88732" , "8985", "59679", "239679", "45780870", "467078", "455685", "59", "45787", "982325", "2398095" };

string sum[100] = { "5412", "2134", "5625", "7896", "5624", "5135", "6548", "1345", "4866", "5135", "6489", "5126", "5256", "5412", "5689", "6987", "1236", "5469", "1267", "8569", "4536", "6884", "2315", "3651", "2152", "3256", "3216", "11363", "5964", "2356", "5153", "5984", "1235", "9523", "6548", "1236", "9845", "3218", "5555", "3264", "1538", "1253", "2152", "0236", "4106", "0234", "2150", "3215", "5410", "5962", "12654", "1562", "15951", "2315", "123548", "12315", "51261", "21354", "1554", "56156" , "15615", "15618", "51512", "5132", "6594", "2626", "5191", "2165", "12315", "1231", "1548", "3548", "15215", "56548", "12315", "1533", "15648", "5153", "02148", "351505", "5105", "5156", "8456", "9875", "3654", "486521", "215654", "54612", "156548", "123545" , "12354", "35198", "1564", "1534", "8645", "4864", "4864", "56454", "51526", "6554" };

string rand\_name()

{

    return (postname[rand() % 100] + " " + names[rand() % 100] + " " + patr[rand() % 100]);

}

string rand\_num()

{

    return num[rand() % 100];

}

string rand\_sum()

{

    return sum[rand() % 100];

}

human create\_human()

{

    human tmp;

    tmp.fullname = rand\_name();

    tmp.num = rand\_num();

    tmp.sum = rand\_sum();

    return tmp;

}

void fill\_human(human\* arr, int count)

{

    for (int i = 0; i < count; i++)

    {

        arr[i] = create\_human();

    }

}

void show(const human tmp)

{

    cout << "ФИО:          " << tmp.fullname

         << "\nНомер счета:  " << tmp.num

         << "\nСумма:        " << tmp.sum << endl << endl;

}

void showTable(hash\_table\* table, int size)

{

    for (int i = 0; i < size; i++)

    {

        show(table->arr[i]);

    }

}

void print\_human(human\* arr, int count)

{

    for (int i = 0; i < count; i++)

    {

        show(arr[i]);

    }

}

int hash\_f(string str, int size)

{

    long double a = (0.68 \* stoi(str));

    double a1 = size \* (a - static\_cast<int>(a));

    return static\_cast<int>(a1);

}

void hash\_table::add(human tmp, int size)

{

    int index = hash\_f(tmp.num, size);

    int hash = index;

    if (arr[index].fullname == "Null")

    {

        arr[index] = tmp;

        return;

    }

    else

    {

        while (index < size)

        {

            if (arr[index].fullname == "Null")

            {

                arr[index] = tmp;

                return;

            }

            index++;

            collis++;

        }

        if (index >= size)

        {

            index = 0;

            collis++;

            while (index < hash)

            {

                if (arr[index].fullname == "Null")

                {

                    arr[index] = tmp;

                    return;

                }

                index++;

                collis++;

            }

        }

    }

}

void hash\_table::findIndex(string num, int size)

{

    int hash = hash\_f(num, size);

    int index = hash;

    while (arr[index].num != num && index < size)

    {

        index++;

    }

    if (index >= size)

    {

        index = 0;

        while (arr[index].num != num && index < hash)

        {

            index++;

        }

        if (index >= hash)

        {

            cout << "Такого человека нет";

        }

        else

        {

            cout << "Такой человек есть по индексу: " << index << endl << endl;

        }

    }

}

int main()

{

    srand(time(0));

    setlocale(LC\_ALL, "Russian");

    int size;

    cout << "Введите кол-во элементов: ";

    cin >> size;

    cout << endl;

    human\* arr = new human[size];

    hash\_table table(size);

    fill\_human(arr, size);

    for (int i = 0; i < size; i++)

    {

        table.add(arr[i], size);

    }

    showTable(&table, size);

    table.findIndex("346356", size);

    cout << endl << endl;

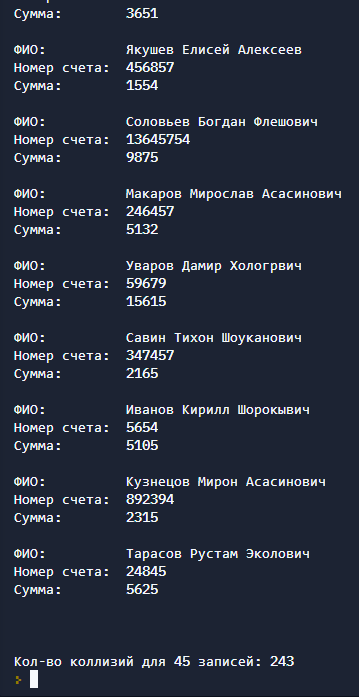
    cout << "Кол-во коллизий для " << size << " записей: " << collis << endl;

    delete[] arr;

    return 0;

}

**Вывод программы**

****