Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

Пермский национальный исследовательский политехнический университет

Электротехнический факультет

Кафедра информационных технологий и автоматизированных систем

**ОТЧЕТ**

**о работе по информатике**

Семестр: 2

На тему: «Поиск с помощью хэш-таблиц. Метод открытой адресации»

Выполнил студент ИВТ-22-2б:

Захаров Дмитрий Сергеевич

Проверил доцент кафедры ИТАС:

Полякова Ольга Андреевна

Пермь 2023

**Постановка задачи**

1. Создать динамический массив из записей (в соответствии с вариантом), содержащий не менее 100 элементов. Для заполнения элементов массива использовать ДСЧ.
2. Предусмотреть сохранение массива в файл и загрузку массива из файла.
3. Предусмотреть возможность добавления и удаления элементов из массива (файла).
4. Выполнить поиск элемента в массиве по ключу в соответствии с вариантом. Для поиска использовать хэш-таблицу.
5. Подсчитать количество коллизий при размере хэш-таблицы 40, 75 и 90 элементов.

**Алгоритм решения**

1. Создать структуру human
2. Создать функцию hash\_table
3. Создать массив с данными
4. Создать функцию заполнения динамического массива структурами
5. Создать функцию печати таблицы
6. Создать функцию добавления элемента в таблицу
7. Создать главную функцию main

**Код программы**

#include <iostream>

#include <string>

using namespace std;

int collis = 0;

struct human

{

string fullname = "Null";

string datas = "Null";

string telephone = "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] = { "Никита", "Иван", "Пол", "Артем", "Егор", "Рома", "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] = { "Иванов", "Баратеон", "Талли", "Инь", "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 datas[100] = { "14.03.2022","01.12.2021","07.08.2021","18.06.2021","16.02.2022",

"15.10.2021","30.08.2021","20.08.2021","24.08.2021","28.09.2021","28.10.2021","25.11.2021",

"10.06.2021","03.11.2021","04.04.2022","09.11.2021","25.05.2021","12.11.2021","26.01.2022",

"03.03.2022","21.03.2022","14.09.2021","08.12.2021","06.12.2021","07.09.2021","24.01.2022",

"20.04.2021","06.09.2021","28.062021","22.022022","04.08.2021","27.10.2021","11.03.2022",

"23.07.2021","30.11.2021","19.102021","11.102021","03.09.2021","05.11.2021","12.05.2021",

"14.10.2021","25.02.2022","27.082021","20.052021","08.03.2022","12.01.2022","30.07.2021",

"23.12.2021","29.11.2021","19.112021","27.052021","11.02.2022","31.12.2021","16.11.2021",

"21.07.2021","26.11.2021","17.052021","15.062021","27.01.2022","19.04.2021","30.04.2021",

"04.02.2022","11.05.2021","22.062021","01.042022","17.03.2022","16.04.2021","21.01.2022",

"25.06.2021","16.06.2021","10.092021","17.022022","07.05.2021","29.06.2021","21.09.2021",

"12.08.2021","22.10.2021","04.012022","16.082021","02.07.2021","14.01.2022","04.10.2021",

"03.05.2021","25.08.2021","20.092021","23.092021","13.04.2021","05.07.2021","13.12.2021",

"22.12.2021","18.08.2021","22.032022","04.032022","11.06.2021","01.06.2021","28.01.2022",

"26.05.2021","17.12.2021","08.042022","31.052021" };

string telephone[100] = { "5424212", "2422134", "5622425", "2427896", "51432624", "511435", "657648", "133242345",

"485666", "5143235", "6425389", "5176426", "5233156", "5413572", "5646789", "6942587", "1212536", "5464789",

"1264367", "856149", "4535236", "6897884", "231565", "364251", "2114252", "32345356", "3268716", "11335363",

"596242354", "235643645", "516588753", "598804234", "12314355", "9578823", "6553748", "12335366", "986876845",

"3257518", "5535355", "32453564", "1532368", "1253978", "216786552", "05645236", "4124506", "0268934",

"21504í64", "3245715", "54587910", "596356342", "126867654", "15255362", "15951", "2315", "123548", "1352315",

"564661261", "2180354", "157854", "561567876" , "15657515", "1561835", "51535612", "5135752", "6556494",

"2575626", "5136491", "2180765", "1256745315", "12353231", "1542458", "3558548", "152464515", "56548", "12315",

"157833", "156345348", "5153533", "021353548", "351505", "5105332", "5113556", "8413156", "98575775",

"3657554", "486521", "215654", "54612", "156548", "123545" , "15752354", "35194648", "1563534", "153352",

"8242325645", "4345864", "4864", "56454", "51526", "6554" };

string rand\_name()

{

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

}

string rand\_datas()

{

return datas[rand() % 100];

}

string rand\_telephone()

{

return telephone[rand() % 100];

}

human create\_human()

{

human tmp;

tmp.fullname = rand\_name();

tmp.datas = rand\_datas();

tmp.telephone = rand\_telephone();

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 << "ÔÈÎ: " << tmp.fullname

<< "\nÄàòà ðîæäåíèÿ: " << tmp.datas

<< "\nÍîìåð òåëåôîíà: " << tmp.telephone << 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 = (0.68245 \* 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.datas, 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].telephone != num && index < size)

{

index++;

}

if (index >= size)

{

index = 0;

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

{

index++;

}

if (index >= hash)

{

cout << "Òàêîãî ÷åëîâåêà íåò";

}

else

{

cout << "Òàêîé ÷åëîâåê åñòü ïî èíäåêñó: " << index << endl << endl;

}

}

}

int main()

{

srand(time(0));

setlocale(LC\_ALL, "Russian");

int size;

cout << "Ââåäèòå êîë-âî ýëåìåíòîâ: ";

cin >> size;

cout << endl;

human\* arr = new human[size];

hash\_table table(size);

(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 << "Êîë-âî êîëëèçèé äëÿ " << size << " çàïèñåé: " << collis << endl;

delete[] arr;

return 0;

}