Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

Пермский национально исследовательский политехнический университет

Электротехнический факультет

Кафедра информационных технологий и автоматизированных систем

**Отчет**

**Тема: Хеш таблица(Метод цепочек)**

Семестр: 2

Выполнил студент ИВТ-22-2Б:

Ромашихин Иван Вячеславович

Проверил доцент кафедры ИТАС:

Полякова Ольга Андреевна

Постановка задачи:

Написать реализацию хеш таблицы с использованием метод

Анализ решения:

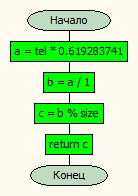
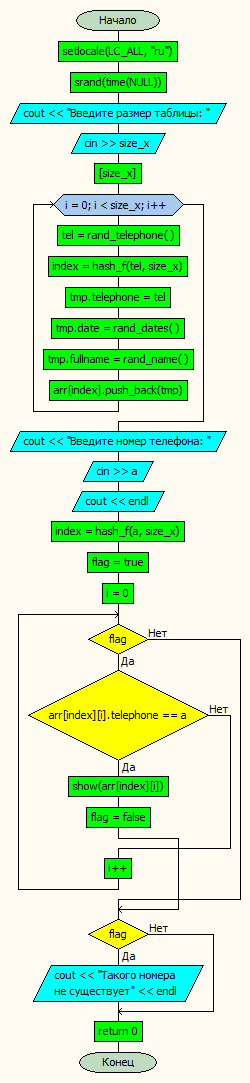
Для решения данной задачи сначала надо реализовать односвязный список, а затем создать массив односвязных списков. Каждый объект должен содержать:

* ФИО
* Номер телефона
* Дату рождения

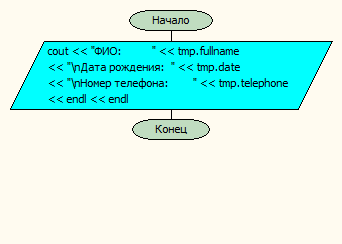
Номер телефона, обрабатываемый хеш-функцией, будет являться индексом объекта в массиве, а односвязный список поможет решить проблему коллизий.

Блок-схема:

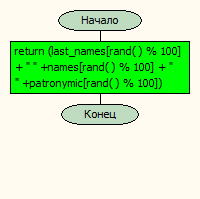
Main: hash\_f:



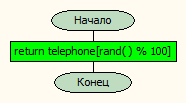
Show:



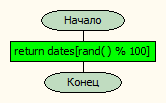
Rand\_name:



Rand\_telephone:



Rand\_date:



Код:

#include <iostream>

#include <string.h>

#include <time.h>

using namespace std;

int size\_x;

class human

{

public:

int telephone = NULL;

string fullname = "NULL";

string date = "NULL";

};

template<typename T>

class List

{

public:

List();

~List();

//удаление первого элемента в списке

void pop\_front();

//добавление элемента в конец списка

void push\_back(T data);

// очистить список

void clear();

// получить количество елементов в списке

int GetSize() { return Size; }

// перегруженный оператор []

T& operator[](const int index);

//добавление элемента в начало списка

void push\_front(T data);

//добавление элемента в список по указанному индексу

void insert(T data, int index);

//удаление элемента в списке по указанному индексу

void removeAt(int index);

//удаление последнего элемента в списке

void pop\_back();

private:

template<typename T>

class Node

{

public:

Node\* pNext;

T data;

Node(T data = T(), Node\* pNext = nullptr)

{

this->data = data;

this->pNext = pNext;

}

};

int Size;

Node<T>\* head;

};

template<typename T>

List<T>::List()

{

Size = 0;

head = nullptr;

}

template<typename T>

List<T>::~List()

{

clear();

}

template<typename T>

void List<T>::pop\_front()

{

Node<T>\* temp = head;

head = head->pNext;

delete temp;

Size--;

}

template<typename T>

void List<T>::push\_back(T data)

{

if (head == nullptr)

{

head = new Node<T>(data);

}

else

{

Node<T>\* current = this->head;

while (current->pNext != nullptr)

{

current = current->pNext;

}

current->pNext = new Node<T>(data);

}

Size++;

}

template<typename T>

void List<T>::clear()

{

while (Size)

{

pop\_front();

}

}

template<typename T>

T& List<T>::operator[](const int index)

{

int counter = 0;

Node<T>\* current = this->head;

while (current != nullptr)

{

if (counter == index)

{

return current->data;

}

current = current->pNext;

counter++;

}

}

template<typename T>

void List<T>::push\_front(T data)

{

head = new Node<T>(data, head);

Size++;

}

template<typename T>

void List<T>::insert(T data, int index)

{

if (index == 0)

{

push\_front(data);

}

else

{

Node<T>\* previous = this->head;

for (int i = 0; i < index - 1; i++)

{

previous = previous->pNext;

}

Node<T>\* newNode = new Node<T>(data, previous->pNext);

previous->pNext = newNode;

Size++;

}

}

template<typename T>

void List<T>::removeAt(int index)

{

if (index == 0)

{

pop\_front();

}

else

{

Node<T>\* previous = this->head;

for (int i = 0; i < index - 1; i++)

{

previous = previous->pNext;

}

Node<T>\* toDelete = previous->pNext;

previous->pNext = toDelete->pNext;

delete toDelete;

Size--;

}

}

template<typename T>

void List<T>::pop\_back()

{

removeAt(Size - 1);

}

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 last\_names[100] = { "Ипатов", "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 patronymic[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 dates[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","28.06.2021","22.02.2022","04.08.2021","27.10.2021","11.03.2022","23.07.2021","30.11.2021","19.10.2021","11.10.2021","03.09.2021","05.11.2021","12.05.2021","14.10.2021","25.02.2022","27.08.2021","20.05.2021","08.03.2022","12.01.2022","30.07.2021","23.12.2021","29.11.2021","19.11.2021","27.05.2021","11.02.2022","31.12.2021","16.11.2021","21.07.2021","26.11.2021","17.05.2021","15.06.2021","27.01.2022","19.04.2021","30.04.2021","04.02.2022","11.05.2021","22.06.2021","01.04.2022","17.03.2022","16.04.2021","21.01.2022","25.06.2021","16.06.2021","10.09.2021","17.02.2022","07.05.2021","29.06.2021","21.09.2021","12.08.2021","22.10.2021","04.01.2022","16.08.2021","02.07.2021","14.01.2022","04.10.2021","03.05.2021","25.08.2021","20.09.2021","23.09.2021","13.04.2021","05.07.2021","13.12.2021","22.12.2021","18.08.2021","22.03.2022","04.03.2022","11.06.2021","01.06.2021","28.01.2022","26.05.2021","17.12.2021","08.04.2022","31.05.2021" };

int 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, 578823, 6553748, 12335366, 986876845, 3257518, 5535355, 32453564, 1532368, 1253978, 216786552, 05645236, 4124506, 1268934, 2150464, 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, 21353548, 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 (last\_names[rand() % 100] + " " + names[rand() % 100] + " " + patronymic[rand() % 100]);

}

string rand\_dates()

{

return dates[rand() % 100];

}

int rand\_telephone()

{

return telephone[rand() % 100];

}

int hash\_f(int tel, int size)

{

double a = tel \* 0.619283741;

int b = a / 1;

int c = b % size;

return c;

}

void show(const human tmp)

{

cout << "ФИО: " << tmp.fullname

<< "\nДата рождения: " << tmp.date

<< "\nНомер телефона: " << tmp.telephone << endl << endl;

}

int main()

{

setlocale(LC\_ALL, "ru");

srand(time(NULL));

cout << "Введите размер таблицы: ";

cin >> size\_x;

List<human>\* arr;

arr = new List<human>[size\_x];

for (int i = 0; i < size\_x; i++)

{

int tel = rand\_telephone();

int index = hash\_f(tel, size\_x);

human tmp;

tmp.telephone = tel;

tmp.date = rand\_dates();

tmp.fullname = rand\_name();

arr[index].push\_back(tmp);

}

int a;

cout << "Введите номер телефона: ";

cin >> a;

cout << endl;

int index = hash\_f(a, size\_x);

bool flag = true;

int i = 0;

while (flag)

{

if (arr[index][i].telephone == a)

{

show(arr[index][i]);

flag = false;

break;

}

i++;

}

if (flag)

{

cout << "Такого номера не существует" << endl;

}

delete[] arr;

return 0;

}

Результат работы программы:

