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 цепочек»

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 узла
2. Создать структуру hashtab
3. Создать структуру client
4. Создать функцию для создания узла
5. Создать функцию для сохранения массива в файл
6. Создать функцию для чтения массива из файла
7. Создать функцию для создания массива
8. Создать функцию для добавления элемента
9. Создать функцию печати таблицы
10. Создать функцию удаления элемента
11. Создать главную функцию main

**Код программы**

#include <iostream>

#include <time.h>

#include <fstream>

#include <string>

#include <Windows.h>

using namespace std;

struct node {

char date[256] = "";

node\* next = nullptr;

};

struct hashtab {

int number = -1;

node\* head = nullptr;

};

struct client {

char FIO[256] = "";

char date[256] = "";

char passport[256] = "";

};

// Функция для создания узла списка

node\* createNode(char\* data) {

node\* newNode = new node;

strncpy(newNode->date, data, 256);

newNode->next = nullptr;

return newNode;

}

// Функция для сохранения массива в файл

void SaveArr(client\* arr, int n) {

client\* current = new client;

char FName[] = "array.txt"; //Путь к файлу

ofstream f1(FName, ios\_base::out | ios\_base::binary); // открываем файл для записи

for (int i = 0; i < n; i++) {

//current.FIO = arr[i].FIO;

strncpy(current->FIO, arr[i].FIO, 256);

strncpy(current->date, arr[i].date, 256);

strncpy(current->passport, arr[i].passport, 256);

f1.write((char\*)current, sizeof(client)); /\*записываем структуру в файл\*/

}

f1.close(); // закрываем файл

}

// Функция для чтения массива из файла

void LoadArr(client\* arr, int n) {

client current;

char FName[] = "array.txt"; //Путь к файлу

ifstream f2(FName); // открываем файл для чтения

if (!f2) {

cout << "Файл не существует\n\n";

f2.close();

}

for (int i = 0; i < n; i++) {

strncpy(arr[i].FIO, "", 256);

strncpy(arr[i].date, "", 256);

strncpy(arr[i].passport, "", 256);

}

f2.seekg(0, ios::end); //функция перехода в конец файла

long NumBytes = (long)f2.tellg(); //выяснение кол-ва байтов в файле

float NumStruct = NumBytes / sizeof(client); //выяснение кол-ва структур в файле

f2.seekg(0, ios::beg); //переход в начало файла

for (int i = 0; i < NumStruct; i++) {

f2.read((char\*)&current, sizeof(client)); // запись структуры из файла в переменную current

//cout << current2.FIO << " " << current2.account << " " << current2.summa << endl;

strncpy(arr[i].FIO, current.FIO, 256);

strncpy(arr[i].date, current.date, 256);

strncpy(arr[i].passport, current.passport, 256);

cout << arr[i].FIO << " " << arr[i].date << " " << arr[i].passport << endl;

}

f2.close();

}

int get\_rnd\_number(int max) {

return rand() % max;

}

string get\_rnd\_passport() {

string pass = to\_string(rand() % 90 + 10) + to\_string(rand() % 90 + 10) + " " + to\_string(rand() % 90 + 10) + to\_string(rand() % 90 + 10) + to\_string(rand() % 90 + 10);

return pass;

}

string get\_rnd\_date() {

string birth = to\_string(rand() % 27 + 1) + "." + to\_string(rand() % 12 + 1) + "." + to\_string(19 + rand() % 2) + to\_string(10 + rand() % 13);

return birth;

}

string get\_rnd\_fio() {

string name[10] = { "Александр", "Борис", "Виктор", "Григорий", "Дмитрий", "Евгений", "Иван", "Николай", "Михаил", "Олег" };

string patronymic[10] = { "Александрович", "Борисович", "Викторович", "Григорьевич", "Дмитриевич", "Евгеньевич", "Иванович", "Николаевич", "Михайлович", "Олегович" };

string surname[10] = { "Александров", "Борисов", "Викторов", "Григорьев", "Дмитриев", "Евгеньев", "Иванов", "Николаев", "Михайлов", "Олегов" };

string fio = name[get\_rnd\_number(9)] + " " + patronymic[get\_rnd\_number(9)] + " " + surname[get\_rnd\_number(9)];

return fio;

}

void makearr\_rnd(int n, client\* arr) {

string s1 = "";

string s2 = "";

string s3 = "";

char sarr1[256];

char sarr2[256];

char sarr3[256];

s1 = get\_rnd\_fio();

s2 = get\_rnd\_date();

s3 = get\_rnd\_passport();

strcpy(sarr1, s1.c\_str());

strcpy(sarr2, s2.c\_str());

strcpy(sarr3, s3.c\_str());

for (int i = 0; i < n - 5; i++) {//-5 - оставляеи 5 пустых строк, чтобы было, куда добавить

s1 = get\_rnd\_fio();

s2 = get\_rnd\_date();

s3 = get\_rnd\_passport();

strcpy(sarr1, s1.c\_str());

strcpy(sarr2, s2.c\_str());

strcpy(sarr3, s3.c\_str());

strncpy(arr[i].FIO, sarr1, 256);

strncpy(arr[i].date, sarr2, 256);

strncpy(arr[i].passport, sarr3, 256);

}

for (int j = 0; j < n - 5; j++) {

cout << arr[j].FIO << " " << arr[j].date << " " << arr[j].passport << endl;

}

cout << endl;

}

void addEl(client\* arr, int n) {

client input;

string temp;

string temp1;

string temp2;

string temp3;

cout << "Введите ФИО: ";

cin.ignore();

getline(cin, temp);

strcpy(input.FIO, temp.c\_str());

cout << "Введите дату рождения: ";

getline(cin, temp1);

strcpy(input.date, temp1.c\_str());

cout << "Введите № паспорта: ";

getline(cin, temp2);

strcpy(input.passport, temp2.c\_str());

if (arr == nullptr) {

cout << "Ошибка";

return;

}

else {

for (int i = 0; i < n; i++) {

temp3 = arr[i].FIO;

if (temp3 == "") {

arr[i] = input;

return;

}

}

cout << "В массиве нет места для новой записи" << endl;

}

}

int hash\_function(string str, int n) {

unsigned long i = 0;

for (int j = 0; str[j]; j++) {

i += str[j];

}

return i % n;

}

void hashta(client\* arr, int n, hashtab\* hashtable) {

int cnt = 0;

for (int i = 0; i < n; i++) {

hashtable[i].number = -1;

hashtable[i].head = nullptr;

}

for (int i = 0; i < n; i++) {

if (arr[i].date != "") { //строка массива не пуста

int key = hash\_function(arr[i].date, n);

bool found = false;

if (hashtable[key].number == -1) { //строка хэш-таблицы свободна

hashtable[key].number = key;

hashtable[key].head = createNode(arr[i].date);

found = true;

}

else { // строка хэш-таблицы заполнена - это коллизия

cnt++; //счетчик коллизий ++

node\* newNode = createNode(arr[i].date); //добавляем новую запись в начало списка

newNode->next = hashtable[key].head;

hashtable[key].head = newNode;

found = true;

}

}

}

for (int i = 0; i < n; i++) { // печать хэш-таблицы

cout << hashtable[i].number;

if (hashtable[i].head != nullptr) {

node\* current = hashtable[i].head;

do {

cout << ' ' << current->date << endl;

current = current->next;

} while (current != nullptr);

}

else {

cout << endl;

}

}

cout << endl;

cout << "счётчик коллизий: " << cnt << endl;

}

void hashFound(hashtab\* hashtable, int n, client\* arr) {

string a;

int k, i;

bool found = false;

cout << "введите дату рождения для поиска: "; cin.ignore(); getline(cin, a);

k = hash\_function(a, n);

if (hashtable[k].head != nullptr) { // в хэш-таблице есть запись с ключом k

node\* current = hashtable[k].head;

do {

if (current->date == a) {

found = true;

}

current = current->next;

} while (current != nullptr && !found);

}

if (found == 0) {

cout << "человек не найден" << endl;

}

else {

cout << "ответ найден в позиции: " << k << endl;

found = FALSE;

for (i = 0; i < n && !found; i++) {

if (arr[i].date == a) {

found = TRUE;

cout << arr[i].FIO << ' ' << arr[i].date << ' ' << arr[i].passport << endl;

}

}

}

}

void DelEL(client\* arr, int n) {

string data;

int pos;

bool found = 0;

int temp;

if (arr == nullptr) {

cout << "Ошибка";

return;

}

else {

cout << "введите дату рождения для удаления: ";

cin.ignore();

getline(cin, data);

pos = hash\_function(data, n);

}

if (data == arr[pos].date) {

found = 1;

strcpy(arr[pos].FIO, "");

strcpy(arr[pos].date, "");

strcpy(arr[pos].passport, "");

}

else {

temp = (pos + 1) % n;

while (temp != pos && !found) {

if (data == arr[temp].FIO) {

found = 1;

strcpy(arr[temp].FIO, "");

strcpy(arr[temp].date, "");

strcpy(arr[temp].passport, "");

}

temp = (temp + 1) % n;

}

}

if (found == 0) cout << "человек не найден" << endl;

else cout << "запись удалена" << endl;

}

int main()

{

int choice = -1;

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

srand((int)time(NULL));

int n = 50;

client\* arr = new client[n];

hashtab\* hashtable = new hashtab[n];

while (choice != 0) {

cout << "1 заполнение массива структур" << endl;

cout << "2 сохранение в файл" << endl;

cout << "3 загрузка из файла" << endl;

cout << "4 добавление элемента в массив" << endl;

cout << "5 создание и печать хэш таблицы" << endl;

cout << "6 поиск даты рождения в хэш таблице" << endl;

cout << "7 удаление заданной даты рождения" << endl;

cout << "8 печать массива" << endl;

cout << "0 выход из программы" << endl;

cout << "введите пункт меню: ";

cin >> choice;

cout << endl;

switch (choice) {

case 1:

makearr\_rnd(n, arr);

break;

case 2:

SaveArr(arr, n);

break;

case 3:

LoadArr(arr, n);

break;

case 4:

addEl(arr, n);

break;

case 5:

hashta(arr, n, hashtable);

break;

case 6:

hashFound(hashtable, n, arr);

break;

case 7:

DelEL(arr, n);

break;

case 8:

for (int j = 0; j < n; j++) cout << arr[j].FIO << " " << arr[j].date << " " << arr[j].passport << endl;

break;

case 0:

return 0;

}

}

return 0;

}