Министерство высшего образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

**«Пермский национальный исследовательский политехнический университет» (ПНИПУ)**

Электротехнический факультет

Кафедра «Информационные технологии и автоматизированные системы»

ОТЧЁТ

о лабораторной работе

Тема: «Реализация Хэш-таблицы»

Выполнил

Студент группы РИС-22-2б

Прядеин И.А.

Проверил доцент кафедры

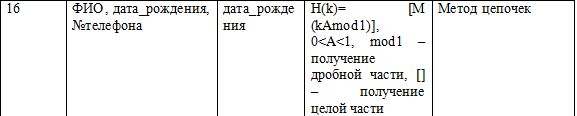
ИТАС

Полякова О. А.

Пермь 2023

**Постановка задачи**

Реализовать Хэш-таблицу с использованием языка программирования C++.



**Исходный код программы:**

#include <iostream>

#include <string>

#include <Windows.h>

#include <ctime>

#include <fstream>

using namespace std;

string Names[5] = { "Андрей", "Алексей", "Тимур", "Петр", "Иван" };

string Surnames[5] = { "Иванов", "Петров", "Андреев", "Алексеев", "Казаков" };

string Patronymics[5] = { "Андреевич", "Иванович", "Петрович", "Алексеевич", "Тимурович" };

string Dates[5] = { "30.12.2003", "08.01.2004", "05.05.2004", "14.03.2002", "15.06.2003" };

string phones[5] = { "89455433464", "85345423423", "89835432365", "89452341296", "89665439653" };

struct Human {

string first\_name = "NULL";

string last\_name = "NULL";

string patronymic = "NULL";

string date\_of\_birth = "NULL";

string phone\_number = "NULL";

Human\* next = nullptr;

void SetHuman(string first\_name, string last\_name, string patronymic, string date\_of\_birth, string phone\_number) {

this->first\_name = first\_name;

this->last\_name = last\_name;

this->patronymic = patronymic;

this->date\_of\_birth = date\_of\_birth;

this->phone\_number = phone\_number;

}

void OutputHuman() {

cout << "Name:\t" << this->last\_name << ' ' << this->first\_name << ' ' << this->patronymic << endl;

cout << "Date\_of\_birth:\t" << this->date\_of\_birth << endl;

cout << "Phone:\t" << this->phone\_number << endl;

}

void CopyHuman(Human\* copy) {

this->first\_name = copy->first\_name;

this->last\_name = copy->last\_name;

this->patronymic = copy->patronymic;

this->date\_of\_birth = copy->date\_of\_birth;

this->phone\_number = copy->phone\_number;

this->next = nullptr;

}

};

struct List {

int size = 0;

Human\* head = nullptr;

Human\* tail = nullptr;

};

bool ListIsEmpty(List\* list);

void PushBack(List\* list);

List\* CreateList(int size);

void ShowList(List\* list);

void RemoveElement(List\* list, int n);

bool ListIsEmpty(List\* list) {

return list == nullptr || list->size == 0;

}

void PushBack(List\* list) {

Human\* p = new Human;

string name = Names[rand() % 5];

string surname = Surnames[rand() % 5];

string patr = Patronymics[rand() % 5];

string date = Dates[rand() % 5];

string phone = phones[rand() % 5];

p->SetHuman(name, surname, patr, date, phone);

p->next = nullptr;

if (ListIsEmpty(list)) {

list->head = p;

list->tail = p;

}

else if (list->size == 1) {

list->tail->next = p;

list->tail = p;

list->head->next = list->tail;

}

else {

list->tail->next = p;

list->tail = p;

}

list->size++;

}

List\* CreateList(int size) {

List\* list = new List;

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

PushBack(list);

return list;

}

void ShowList(List\* list) {

if (ListIsEmpty(list)) cout << "Список пуст!\n";

else {

int i = 1;

Human\* temp = list->head;

cout << "---------------------------\n";

cout << "\tСписок\n";

while (i <= list->size) {

temp->OutputHuman();

temp = temp->next;

++i;

}

cout << "---------------------------\n";

}

}

void RemoveElement(List\* list, int n) {

Human\* temp = list->head;

int i = 1;

if (i == n) {

list->head = temp->next;

delete temp;

list->size--;

return;

}

while (i + 1 != n) {

i++;

}

temp->next = temp->next->next;

delete temp->next;

list->size--;

return;

}

int hash\_func(string pep, int size) {

float num;

num = stoi(pep.substr(0, 2)) + stoi(pep.substr(3, 2)) + stoi(pep.substr(6, 4));

num \*= 0.414141;

num -= (int)num;

num \*= size;

return (int)num;

}

Human\*\* open\_address\_method(List\* l) {

int size = l->size, pos, collisions\_count = 0;

Human\*\* table = new Human \* [size];

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

table[i] = new Human;

}

Human\* pep = l->head;

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

pos = hash\_func(pep->date\_of\_birth, size);

if (table[pos]->date\_of\_birth == "NULL") {

table[pos] = pep;

pep = pep->next;

}

else {

while (table[pos]->date\_of\_birth != "NULL") {

pos++;

collisions\_count++;

if (pos >= size) pos = pos - size;

}

table[pos] = pep;

pep = pep->next;

}

}

cout << "Кол-во коллизий = " << collisions\_count << endl;

return table;

}

Human\*\* chain\_method(List\* l) {

int size = l->size, pos, collisions\_count = 0;

Human\*\* table = new Human \* [size];

for (int i = 0; i < size; i++) table[i] = new Human;

Human\* curr = l->head;

Human\* pep = new Human, \* r;

pep->CopyHuman(curr);

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

pos = hash\_func(curr->date\_of\_birth, size);

if (table[pos]->date\_of\_birth == "NULL") {

table[pos] = pep;

}

else {

collisions\_count++;

r = table[pos];

pep->next = r;

table[pos] = pep;

}

pep = new Human;

if (curr->next != nullptr) {

curr = curr->next;

pep->CopyHuman(curr);

}

}

cout << "Кол-во коллизий = " << collisions\_count << endl;

return table;

}

void print\_hash(Human\*\* table, int size, int type) {

if (type == 1) for (int i = 0; i < size; i++) {

table[i]->OutputHuman();

cout << "-----------" << endl;

}

else {

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

cout << i << ":\n";

Human\* pep = table[i];

do {

pep->OutputHuman();

pep = pep->next;

cout << endl;

} while (pep != nullptr);

cout << "-----------" << endl;

}

}

}

void ShowMenu() {

cout << "\t\tМеню\n";

cout << "Введите 0 для завершения работы программы.\n";

cout << "Введите 1 для создания списка.\n";

cout << "Введите 2 для просмотра списка.\n";

cout << "Введите 3, чтобы удалить список.\n";

cout << "Введите 4 для метода открытой адрессации.\n";

cout << "Введите 5 для метода цепочек.\n";

}

void menu() {

int switch\_n, size\_list, number, count;

string key;

List\* list = new List;

Human\*\* table;

Human\* pep;

string date; int pos, counter = 0;

bool flag;

do {

system("pause");

system("cls");

ShowMenu();

cin >> switch\_n;

switch (switch\_n) {

case 0:

cout << "Работа завершена";

break;

case 1:

cout << "Введите размер списка: ";

cin >> size\_list;

while (size\_list < 1) {

cout << "Размер списка должен быть от одного и более элементов! Введите размер: ";

cin >> size\_list;

}

list = CreateList(size\_list);

break;

case 2:

ShowList(list);

break;

case 3:

while (!ListIsEmpty(list)) {

RemoveElement(list, 1);

}

cout << "Список удалён\n";

break;

case 4:

counter = 0;

table = open\_address\_method(list);

print\_hash(table, list->size, 1);

break;

case 5:

flag = false;

table = chain\_method(list);

print\_hash(table, list->size, 2);

break;

}

} while (switch\_n != 0);

}

int main() {

srand(time(NULL));

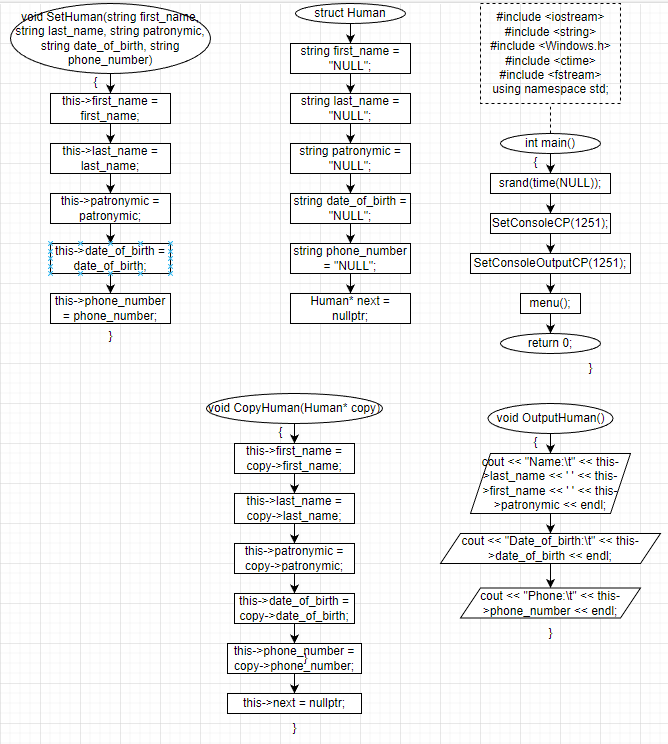
SetConsoleCP(1251); SetConsoleOutputCP(1251);

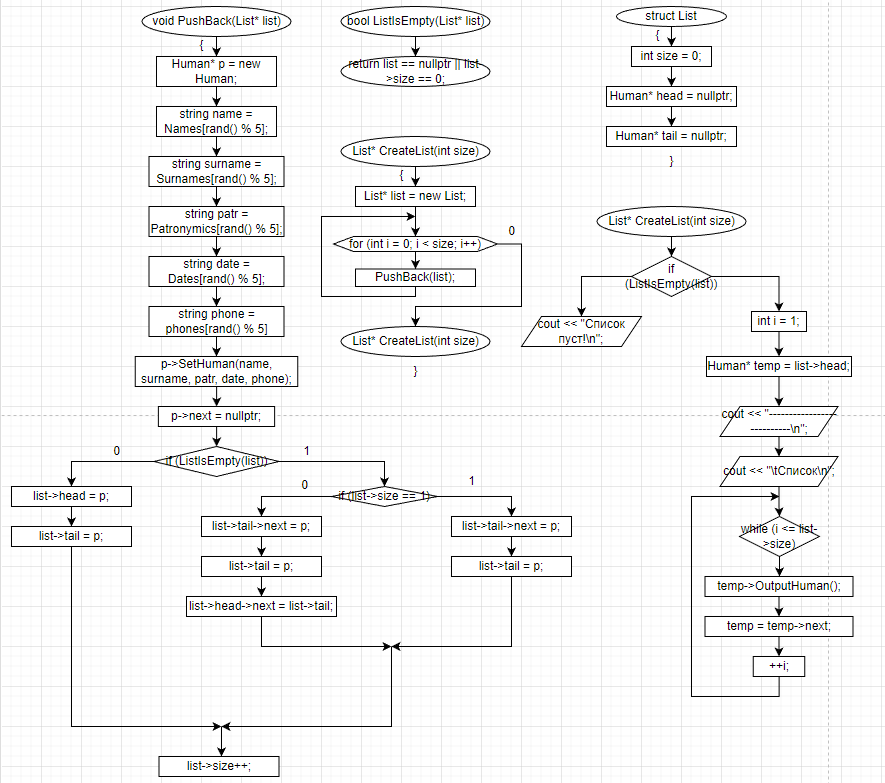
menu();

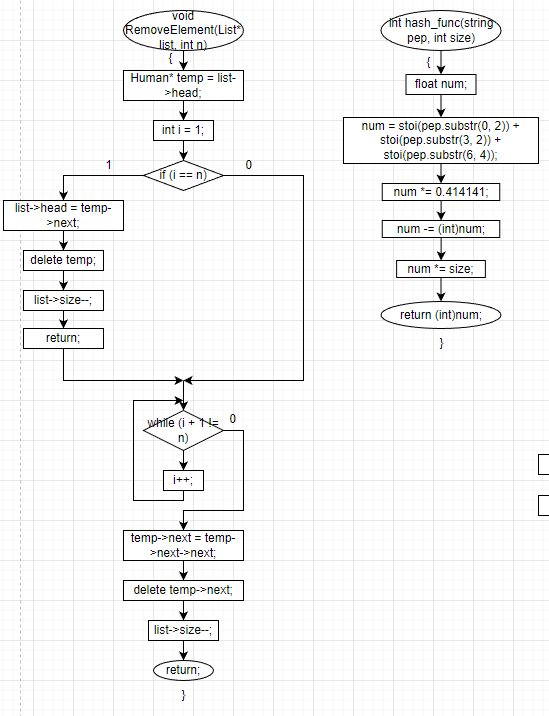
return 0;

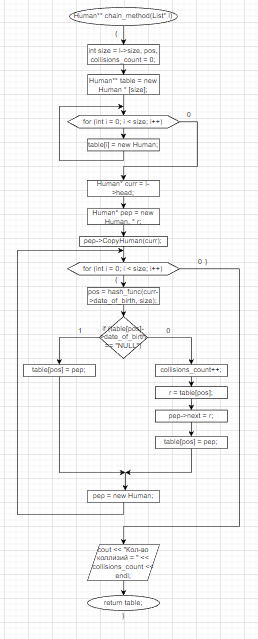
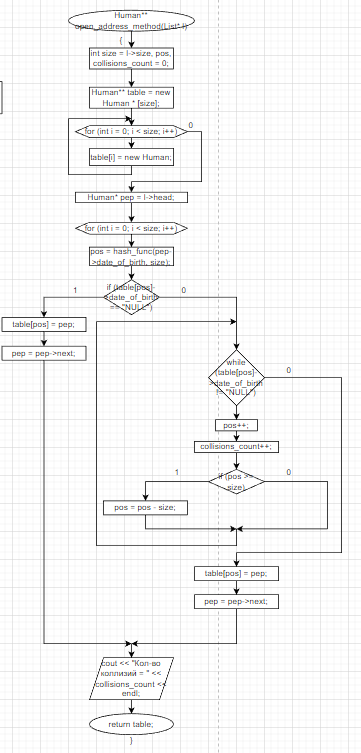
}

**Блок схема**









**Скриншоты результатов выполнения программы**

