**Вариант 12**

1. Описать структуру с именем NOTE, содержащую следующие поля :

• фамилия, имя;

• номер телефона;

• день рождения(массив из трех чисел).

2. Написать программу реализующую список(в виде класса) структур, выполняющую следующие действия :

• ввод с клавиатуры данных в массив, состоящий из восьми элементов типа NOTE; записи должны быть упорядочены по датам дней рождения;

• вывод па экран информации о человеке, номер телефона которого введен с клавиатуры;

• если такого нет, выдать на дисплей соответствующее сообщение.

**Решение**

Программа реализована с помощью класса «massiv», находящегося в файлах «massiv.h» и «massiv.cpp». Структура находится в заголовочном файле «Header.h», там же находится набор основных подключаемых библиотек. Исполнительный файл «2laba\_1.cpp» собирает все файлы и регулирует основную работу программы.

**Header.h**

#include <iostream>

#include <string>

using namespace std;

struct NOTE

{

string name = "name";

string phone = "+79003003030";

int date[3] = { 31,12,2000 };

void printInfo() {

cout << "name: " << name << endl;

cout << "phone: " << phone << endl;

cout << "date: " << date[0] << "." << date[1] << "." << date[2] << endl;

}

};

**massiv.h**

#include "Header.h"

class massiv

{

struct elementInfo

{

NOTE el;

elementInfo\* next, \* prev;

};

elementInfo \* head, \* end;

int count;

public:

massiv();

~massiv();

int getCount();

bool Del(int pos);

bool delAll();

bool Insert(NOTE x);

int Print();

int whoIsOlder(NOTE one, NOTE two);

int findPlace(NOTE x);

bool findNumber(string num);

};

**massiv.cpp**

#include "massiv.h"

massiv::massiv() //конструктор обнуляет список

{

head = 0;

end = 0;

count = 0;

}

massiv::~massiv() //деструктор очищает список

{

delAll();

}

int massiv::getCount() //получить кол-во элементов в списке

{

return count;

}

bool massiv::Del(int pos) //удаление элемента по индексу

{

if ((pos <= count) && (pos > 0)) { //проверка на правильность индекса

int i = 1;

elementInfo\* forDel = head; //перебор до нужной позиции от головы

while (i < pos) {

forDel = forDel->next;

i++;

}

elementInfo\* prevDel = forDel->prev; //сохраняем предыдущий

elementInfo\* nextDel = forDel->next; // сохраняем следующий

if (pos == 1) head = nextDel; else prevDel->next = nextDel; // определяем ссылку на следующий элемент

if (pos == count) end = prevDel; else nextDel->prev = prevDel; // определяем ссылку на предыдущий

delete forDel; // удаляем элемент

count--;

}

return true;

}

bool massiv::delAll() //очистить весь список

{

while(count != 0) {

Del(1);

}

return true;

}

bool massiv::Insert(NOTE x) // вставка элемента

{

elementInfo\* add = new elementInfo; // создаем новый элемент

if (count == 0) { //в случае пустого списка

add->next = NULL; //обнуление ссылок

add->prev = NULL;

add->el = x; //заполнение данных

head = end = add; //элемент - голова и хвост

count = 1;

return true;

}

//в случае не пустого списка

int k = findPlace(x); //определяем место вставки элемента

if (k == 1) { //если новый элемент - голова

add->prev = NULL; //предыдущего нет

add->el = x; //заполнение данных

add->next = head; // следующий - бывшая голова

head->prev = add; // предыдущий элемент бывшей головы - новый элемент

head = add; // новая голова

count++;

return true;

}

if (k > count) { // если элемент последний

add->next = NULL; // следующего нет

add->el = x; // заполнение данных

add->prev = end; // предыдущий - старый конец

end->next = add; // следующий у старого хвоста - новый элемент

end = add; // новый конец

count++;

return true;

}

//вставка в другое место списка

add->el = x; //заполнение данных

elementInfo\* ins = head; // находим элемент перед которым вставляем

int i = 1;

while (i < k) {

ins = ins->next;

i++;

}

elementInfo\* prevIns = ins->prev; //элемент перед вставляемым

prevIns->next = add; // настройка связей

add->next = ins;

add->prev = prevIns;

ins->prev = add;

count++;

return true;

}

int massiv::Print() //печать всех элементов

{

elementInfo\* pr = head;

for (int i = 1; i <= count; i++) {

cout << i << ". " << endl;

pr->el.printInfo();

pr = pr->next;

}

return 0;

}

int massiv::whoIsOlder(NOTE one, NOTE two) //определение кто из двух старше

{

for (int i = 2; i >= 0; i--) {

if (one.date[i] != two.date[i]) {

if (one.date[i] > two.date[i]) return 2;

else return 1;

}

}

return 0;

}

int massiv::findPlace(NOTE x) //поиск места по возрасту в списке

{

int i = 1;

elementInfo \* forFind = head;

for (i = 1; i <= count; i++) {

if (whoIsOlder(forFind->el, x) == 2) return i;

forFind = forFind->next;

}

return i;

}

bool massiv::findNumber(string num) // найти человека по номеру телефона

{

elementInfo\* forFind = head;

for (int i = 1; i <= count; i++) {

if (forFind->el.phone == num) {

forFind->el.printInfo();

return true;

}

forFind = forFind->next;

}

return false;

}

**2laba\_1.cpp**

#include "Header.h"

#include "massiv.h"

int main()

{

massiv list; //создаем список

int k = 1;

while (k != 0) {

cout << endl;

cout << "1 - print list; 2 - delete element; " << endl;

cout << "3 - insert elements; 4 - find number; 0 - exit. " << endl;

cin >> k;

switch (k)

{

case 1: { //печать списка

list.Print();

break;

}

case 2: { //удалить элемент по позиции введенной с клавиатуры

int j;

cout << "Enter position "; cin >> j;

list.Del(j);

break;

}

case 3: { //вставить любое количество элементов

cout << "How much elements?" << endl;

int n;

cin >> n;

NOTE ex;

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

cout << i + 1 << ". " << endl;

cout << "name: "; cin >> ex.name;

cout << "phone: "; cin >> ex.phone;

cout << "year: "; cin >> ex.date[2];

cout << "month: "; cin >> ex.date[1];

cout << "day: "; cin >> ex.date[0];

list.Insert(ex);

}

break;

}

case 4: { // найти человека по номеру телефона

string s;

cout << "Enter number: "; cin >> s;

if (list.findNumber(s) == false) cout << "not found" << endl;

break;

}

case 0: { // выйти из программы

k = 0;

break;

}

default:

break;

}

}

}