Министерство образования и науки Российской Федерации

Волжский политехнический институт (филиал)

федерального государственного бюджетного образовательного учреждения высшего образования

«Волгоградский государственный технический университет»

(ВПИ (филиал) ВолгГТУ)

|  |  |
| --- | --- |
| Факультет | Инженерно-экономический факультет |
| Кафедра | Информатика и технология программирования |

|  |
| --- |
| **Реализация основных абстрактных типов данных и стандартные алгоритмы их обработки** |

|  |  |
| --- | --- |
| Студент | Малиновский Ярослав Андреевич |
|  | (фамилия, имя, отчество) |

|  |  |
| --- | --- |
| Руководитель | доц., Абрамова О.Ф. |
|  | (долж., фамилия и инициалы) |

Волжский, 2020

**Задание 1**

Описать структуру с именем FATT, с полями: фамилия, имя, должность, год поступления на работу. Сформировать односвязный список. Реализовать возможность удаления элемента по фамилии, вставки нового элемента в произвольное место списка.

#include <iostream>

#include <Windows.h>

#include <limits>

using namespace std;

unsigned int q = 0; // глобальная переменная - количество элементов в списке

struct fatt //описание структуры списка

{

string fam; // поле фамилии

string name; // поле имени

string post; // поле должности

unsigned int year; // поле года поступления

fatt \*next = NULL; //указатель на следующий элемент

} \*head, \*tail; //указатель на первый и последний элемент списка

// Добавление элемента

void Add(string fam, string name, string post, unsigned int year, int position)

{

fatt\* node = new fatt; //создание будущего нового элемента

node->fam = fam; // присвоение будущему элементу поля фамилии

node->name = name; // присвоение будущему элементу поля имени

node->post = post; // присвоение будущему элементу поля должности

node->year = year; // присвоение будущему элементу поля года поступления

if (head == NULL) //если список пуст

{

if (position == 1) // если элемент первый в списке

{

head = node; //определяется голова списка

tail = node; //определяется хвост списка

}

else // если позиция нового элемента не первая (в пустом списке нова позиция может быть только 1)

{

cout << "\nНельзя добавить элемент на данную позицию\n\n";

return;

}

}

else

{

if (position == 1) // если новый элемент первый в списке

{

node->next = head;

head = node;

}

else if (position == q + 1) // если новый элемент последний в списке

{

tail->next = node;

tail = node;

}

else if (position >= 1 && position <= q + 1) // если позиция не выходит за возможные пределы (напр., позиция 5, когда в списке 3 элемента)

{

fatt\* temp = new fatt;

temp = head;

for (int i = 1; i < position - 1; i++) // перебор до элемента под номером позиции - 1, т.к. список односвязный

temp = temp->next;

node->next = temp->next;

temp->next = node;

}

else // если позиция выходит за возможные пределы

{

cout << "\nНельзя добавить элемент на данную позицию\n\n";

return;

}

}

cout << "\nЭлемент добавлен...\n\n";

q++;

}

// Удаление элемента

void Del(string fam)

{

if (head == NULL)

cout << "\nСписок пуст\n\n";

else

{

fatt\* temp1 = new fatt;

temp1 = head;

if (head->fam == fam) // если удаляется 1-й элемент

{

head = head->next;

delete temp1;

cout << "\nЭлемент удален...\n\n";

q--;

}

else

{

for (int i = 1; i < q; i++) // перебор элементов для поиска по фамилии

{

if (temp1->next->fam == fam) // поиск элемента предыдущего перед удаляемым, т.к. список односвязный

{

if (temp1->next == tail) // если удаляемый элемент - хвост списка

{

temp1->next = tail->next;

delete tail;

tail = temp1;

}

else

{

fatt\* temp2 = new fatt;

temp2 = temp1->next;

temp1->next = temp2->next;

delete temp2;

}

cout << "\nЭлемент удален...\n\n";

q--;

return;

}

temp1 = temp1->next;

}

cout << "\nЭлемент не найден\n\n";

}

}

}

// Вывод списка

void Print()

{

if (head == NULL)

cout << "\nСписок пуст\n\n";

else

{

fatt\* temp = head;

cout << "\nЭлементы списка:\n";

for (int i = 0; i < q; i++)

{

cout << i + 1 << ". " << temp->fam << " " << temp->name << "\t\t" << temp->post << '\t' << temp->year << endl;

temp = temp->next;

}

cout << "\n";

}

}

// Главная функция

int main()

{

system("chcp 1251"); // смена кодировки консоли для коррекотного вывода кириллицы

Sleep(1000); // небольшая пауза перед выводом меню

string fam, name, post;

unsigned int year, position; // unsigned int, т.к. год и позиция - неотрицательные числа. Это позволит сместить границу предельных значений в положительную сторону

char x;

do

{

system("cls");

cout << "Количество элементов: " << q << endl;

cout << "1. Добавить элемент на позицию" << endl;

cout << "2. Удалить элемент по фамилии" << endl;

cout << "3. Вывести список" << endl;

cout << "0. Выйти" << endl;

cout << "\nНомер операции > ";

cin >> x;

switch (x)

{

case '0':

cout << endl;

break;

case '1':

cout << "\nФамилия > ";

cin >> fam;

cout << "Имя > ";

cin >> name;

cout << "Должность > ";

cin >> post;

cout << "Год поступления на работу > ";

cin >> year;

cout << "\nПозиция > ";

cin >> position;

Add(fam, name, post, year, position);

break;

case '2':

cout << "\nФамилия > ";

cin >> fam;

Del(fam);

break;

case '3':

Print();

break;

default:

cout << "\nОперации не существует\n\n";

cin.ignore((numeric\_limits<streamsize>::max)(), '\n');

}

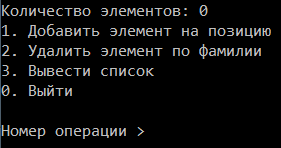
system("pause");

} while (x != '0');

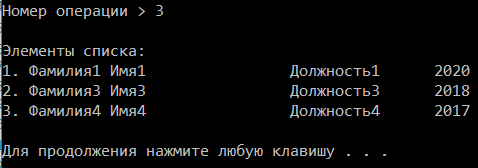
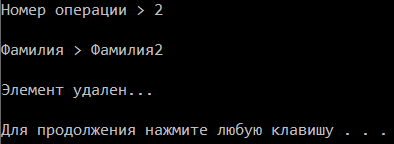
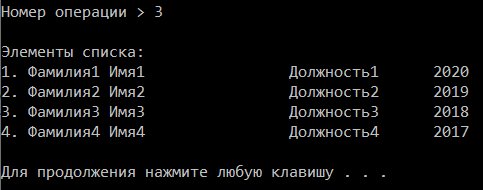
return 0;

}

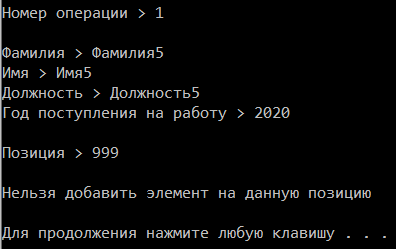
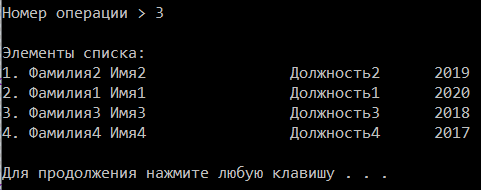
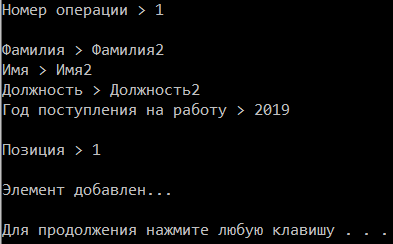
Результат работы программы:



Удаление элемента по фамилии:



Вставка нового элемента в произвольное место списка:



**Задание 2**

Построить хеш-таблицу содержащую последовательность из m=59 элементов размерности n=2. Элементы генерируются с помощью датчика случайных чисел. Хеш-функция – предпоследняя цифра квадрата ключа. Метод разрешения конфликта - квадратичные пробы.

#include <iostream>

#include <time.h>

using namespace std;

int main()

{

system("chcp 1251");

srand(time(0));

float koef, srprob;

const int m = 59;

bool inhash;

int z;

// Размероность хеш-таблицы

const int t = 2 \* m;

// Вывод элементов

unsigned int a[m];

cout << "\nЭлементы:\n\n";

for (int i = 0; i < m; i++)

{

a[i] = rand();

cout << "a[" << i + 1 << "] = " << a[i] << endl;

}

cout << endl;

// Хеш-таблица

// Число проб

int prob = 0;

// Хеширование

int\* b = new int[t] {};

for (int i = 0; i < m; i++)

{

for (int j = 0; ; j++)

{

prob++;

z = (a[i] % m + j \* j) % t;

if (!b[z])

{

b[z] = a[i];

break;

}

}

}

// Вывод таблицы

cout << "Таблица:\n";

for (int i = 0; i < t; i++)

{

cout << "\nb[" << i << "] = " << b[i];

}

// Подсчет среднего числа проб

srprob = (float)prob / m;

cout << "\n\nСреднее число проб = " << srprob;

// Подсчет коэфицента заполнения хеш-таблицы

int zapoln = 0;

for (int i = 0; i < t; i++)

{

if (b[i])

zapoln++;

}

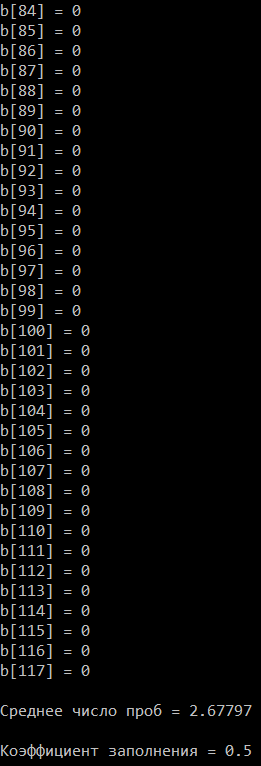
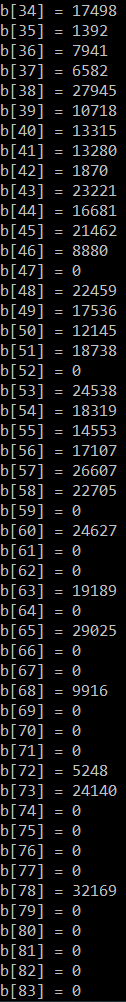
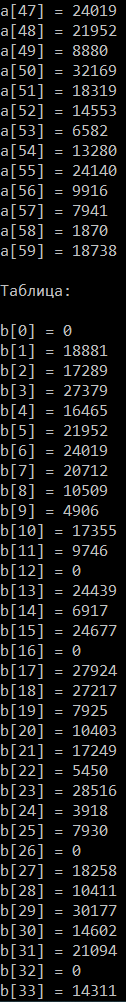
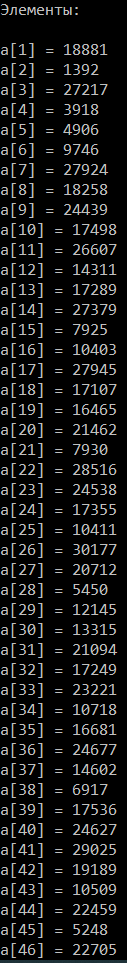
koef = (float)zapoln / t;

cout << "\n\nКоэффициент заполнения = " << koef << endl;

return 0;

}

Результат работы программы:



**Задание 3**

Разработать программу, формирующую динамическую структуру данных для хранения генеалогического дерева. Каждая вершина дерева должна содержать следующую информацию: имя и год рождения.

#include <iostream>

#include <time.h>

using namespace std;

unsigned int k = 0;

struct Node //Звено дерева

{

int x; //То, что записываем в дерево

Node\* l = NULL, \* r = NULL; //Это указатели на новые звенья

};

void show(Node\*& Tree, int i) //Функция обхода

{

if (Tree != NULL) //Пока не встретится пустое звено

{

show(Tree->r, i + 1); //Рекурсивная функция для вывода правого поддерева

for (int j = 0; j < i; j++)

cout << '\t';

cout << Tree->x << endl; //Отображаем значение

show(Tree->l, i + 1); //Рекурсивная функция для вывода левого поддерева

}

else

{

for (int j = 0; j < i; j++)

cout << '\t';

cout << "NULL" << endl;

}

}

void find(Node\*& Tree, int e) //Функция обхода

{

if (Tree != NULL) //Пока не встретится пустое звено

{

find(Tree->l, e); //Рекурсивная функция для вывода левого поддерева

find(Tree->r, e); //Рекурсивная функция для вывода правого поддерева

if (Tree->x == e)

k++;

}

}

// Добавили очистку памяти

void del(Node\*& Tree)

{

if (Tree != NULL) //Пока не встретится пустое звено

{

del(Tree->l); //Рекурсивная функция прохода по левому поддереву

del(Tree->r); //Рекурсивная функция для прохода по правому поддереву

delete Tree; //Убиваем конечный элемент дерева

Tree = NULL; //Делаем дерево пустым

}

}

void add\_node(int x, Node\*& MyTree) //Фукция добавления звена в дерево

{

if (MyTree == NULL) //То, о чем я в самом начале писал. Если дерева нет, то кладем семечко

{

MyTree = new Node; //Выделяем память под звено дерева

MyTree->x = x; //Записываем данные в звено

}

if (x < MyTree->x) //Если нововведенный элемент x меньше чем элемент x из семечка дерева, уходим влево

{

if (MyTree->l != NULL)

add\_node(x, MyTree->l); //При помощи рекурсии заталкиваем элемент на свободный участок

else //Если элемент получил свой участок, то

{

MyTree->l = new Node;//Выделяем память левому подзвену

MyTree->l->x = x; //Записываем в левое подзвено записываемый элемент

}

}

if (x > MyTree->x) //Если нововведенный элемент x больше чем элемент x из семечка дерева, уходим вправо

{

if (MyTree->r != NULL)

add\_node(x, MyTree->r); //При помощи рекурсии заталкиваем элемент на свободный участок

else //Если элемент получил свой участок, то

{

MyTree->r = new Node; //Выделяем память правому подзвену

MyTree->r->x = x; //Записываем в правое подзвено записываемый элемент

}

}

}

int main()

{

system("chcp 1251");

srand(time(NULL));

unsigned int q;

int e;

Node\* Tree = NULL; //Указатель на дерево, делаем пустым, т.к. дерева еще нет

cout << "Количество элементов: ";

cin >> q;

for (int i = 0; i < q; i++)

add\_node(rand() % 100, Tree);

show(Tree, 0); //Вывод на экран дерева

cout << "\nКорень: " << Tree->x << endl;

cout << "Значение элемента E: ";

cin >> e;

find(Tree, e);

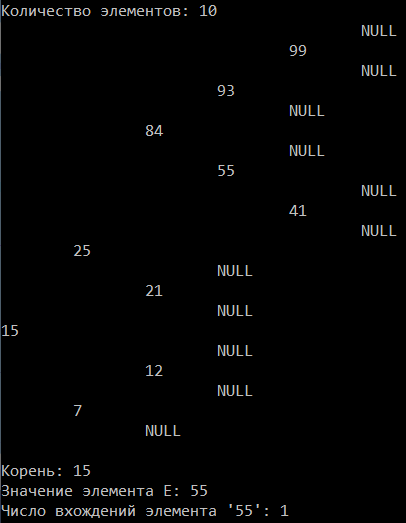
cout << "Число вхождений элемента '" << e << "': " << k << endl;

del(Tree); //Чистка памяти

return 0;

}

Результат работы программы:



Дерево:

**Задание 4**

Проверьте, содержит ли граф, заданный с помощью списков инцидентности, вершину, в которую входят дуги от всех остальных вершин графа, но из которой не исходит ни одна дуга.

#include <iostream>

#include <algorithm>

using namespace std;

bool init(unsigned int \*mas, int q)

{

cout << "Заполнение списка инцидентности. Укажите вершины, в которые входят ребра из указанной вершины:" << endl;

for (int i = 0; i < q; i++)

{

cout << i + 1 << " -> ";

cin >> mas[i];

for (int j = mas[i]; j; j /= 10)

if (j % 10 < 1 || j % 10 > q)

{

cout << "Указанной вершины не существует. Выход..." << endl;

return false;

}

}

for (int i = 0; i < q - 1; i++)

if (mas[i] == 0)

for (int j = i + 1; j < q; j++)

if (mas[j] == 0)

{

cout << "Подобных вершин нет1. Выход..." << endl;

return false;

}

return true;

}

void check(unsigned int\* mas, int q)

{

int n = 0, m = 1;

bool\* flags = new bool[q] {}, flagswap = false;

if (mas[0] == 0)

{

flagswap = !flagswap;

swap(mas[0], mas[1]);

}

for (int a = mas[0]; a; a /= 10)

{

for (int i = 0; i < q - 1; i++)

flags[i] = false;

int b = a % 10;

for (int i = 1; i < q; i++)

{

if (!mas[i])

flags[i - 1] = true;

else

{

for (int c = mas[i]; c; c /= 10)

{

int d = c % 10;

if (b == d)

{

flags[i - 1] = true;

break;

}

}

if (!flags[i - 1])

break;

}

}

for (int i = 0; i < q - 1; i++)

{

if (flags[i])

{

if (i == q - 2)

{

n += b \* m;

m \*= 10;

}

}

else

break;

}

}

if (flagswap)

{

flagswap = !flagswap;

swap(mas[0], mas[1]);

}

cout << "Вершины, в которую входят дуги от всех остальных вершин графа, но из которой не исходит ни одна дуга:" << endl;

if (n)

for (int i = n; i; i /= 10)

{

if (!mas[i % 10 - 1])

cout << i % 10 << endl;

}

else

{

cout << n << endl;

cout << "Подобных вершин нет2. Выход..." << endl;

}

}

int main()

{

system("chcp 1251");

unsigned int q;

cout << "Количество вершин: ";

cin >> q;

if (!(q < 2))

{

unsigned int\* mas = new unsigned int [q] {};

if (init(mas, q))

check(mas, q);

delete[] mas;

}

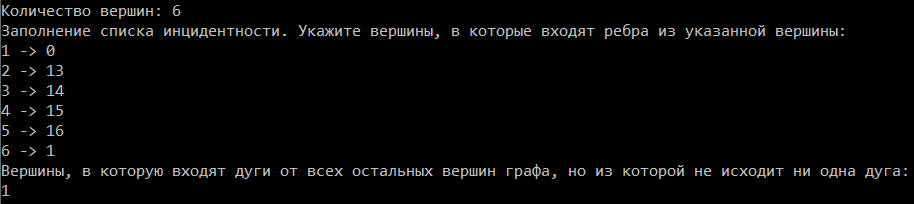
else

cout << "Количество вершин должно быть >= 2. Выход...";

return 0;

}

Результат работы программы:



Граф:

**Задание 5**

Формирование, печать, удаление элементов и преобразование файлов оформить в виде подпрограмм. Предусмотреть сообщения об ошибках при открытии файла и выполнении операций ввода/вывода.

Структура "Человек":

* фамилия, имя, отчество;
* домашний адрес;
* номер телефона;
* возраст.

Удалить все элементы с заданным возрастом.

#include <iostream>

#include <fstream>

#include <Windows.h>

#include <string>

#include <vector>

using namespace std;

void addPerson(fstream& file)

{

if (file.is\_open())

{

file.clear();

file.seekp(0, ios::end);

char str[60];

char s[] = "|                              |                                                          |                |         |";

cout << "ФИО: ";

cin.getline(str, 60);

for (int i = 0; str[i] != '\0'; i++)

s[i + 2] = str[i];

cout << "Домашний адрес: ";

cin.getline(str, 60);

for (int i = 0; str[i] != '\0'; i++)

s[i + 33] = str[i];

cout << "Номер телефона: ";

cin.getline(str, 60);

for (int i = 0; str[i] != '\0'; i++)

s[i + 92] = str[i];

cout << "Возраст: ";

cin.getline(str, 60);

for (int i = 0; str[i] != '\0'; i++)

s[i + 109] = str[i];

file << s << endl;

file << "----------------------------------------------------------------------------------------------------------------------" << endl;

cout << "\nЭлемент добавлен!" << endl;

}

else

cout << "Не получилось открыть файл!";

}

void clearFile(fstream& file, string name)

{

if (file.is\_open())

{

file.close();

file.open(name, ios::in | ios::out | ios::trunc);

cout << "Файл очищен!" << endl;

}

else

cout << "Не получилось открыть файл!";

}

void deleteAge(fstream& file, string name)

{

if (file.is\_open())

{

string str, age;

int count = 0;

file.clear();

file.seekg(0, ios::beg);

vector<string> curList;

while (getline(file, str))

{

cout << str << endl;

curList.push\_back(str);

}

file.close();

cout << "\nВозраст: ";

getline(cin, age);

file.open(name, ios::in | ios::out | ios::trunc);

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

if (curList[i].find(age, 109) == 109)

{

count++;

i++;

}

else

file << curList[i] << '\n';

cout << "Удалено: " << count << " элементов!" << endl;

}

else

cout << "Не получилось открыть файл!";

}

void showFile(fstream& file)

{

if (file.is\_open())

{

file.clear();

file.seekg(0, ios::beg);

string str;

while (getline(file, str))

cout << str << endl;

}

else

cout << "Не получилось открыть файл!";

}

void showMenu(fstream& file, string name)

{

char x;

do

{

file.seekp(0, ios::end);

if (!file.tellp())

{

file << "----------------------------------------------------------------------------------------------------------------------" << endl;

file << "|     ФАМИЛИЯ ИМЯ ОТЧЕСТВО     |                      ДОМАШНИЙ АДРЕС                      | НОМЕР ТЕЛЕФОНА | ВОЗРАСТ |" << endl;

file << "----------------------------------------------------------------------------------------------------------------------" << endl;

}

system("cls");

cout << name << endl;

cout << "Меню:" << endl;

cout << "1. Добавить элемент" << endl;

cout << "2. Очистить файл" << endl;

cout << "3. Удалить элементы с заданным возрастом" << endl;

cout << "4. Просмотр файла" << endl;

cout << "0. Закрыть файл" << endl;

cout << "\n> ";

cin >> x;

cin.ignore((numeric\_limits<streamsize>::max)(), '\n');

cout << endl;

switch (x)

{

case '0':

cout << "Файл сохранен!" << endl;

break;

case '1':

addPerson(file);

cout << endl;

system("pause");

break;

case '2':

clearFile(file, name);

cout << endl;

system("pause");

break;

case '3':

deleteAge(file, name);

cout << endl;

system("pause");

break;

case '4':

showFile(file);

cout << endl;

system("pause");

break;

default:

cout << "Операции не существует\n";

}

} while (x != '0');

}

void createFile()

{

string name;

cout << "Имя файла: ";

getline(cin, name);

if (name == "list.txt")

{

cout << "\nНевозможно содать файл с таким именем!" << endl;

return;

}

fstream list(name, ios::in);

if (list.is\_open())

{

char x;

list.close();

cout << "\nФайл уже существует! Заменить? y / n\t" << endl;

cin >> x;

if (x == 'y')

{

list.open(name, ios::in | ios::out | ios::trunc);

showMenu(list, name);

list.close();

}

}

else

{

list.open("list.txt", ios::out | ios::app);

list << name << endl;

list.close();

list.open(name, ios::in | ios::out | ios::app);

showMenu(list, name);

list.close();

}

}

void openFile()

{

fstream list("list.txt", ios::in);

if (list.is\_open())

{

string str, name;

bool foundFlag = false;

cout << "Список файлов:" << endl;

while (getline(list, str))

cout << " " << str << endl;

list.close();

cout << "\nИмя файла: ";

getline(cin, name);

list.open(name, ios::in);

if (list.is\_open())

{

list.close();

list.open(name, ios:: in | ios:: out | ios::app);

showMenu(list, name);

list.close();

}

else

cout << "\nФайла не существует!" << endl;

}

else

cout << "Не получилось открыть список файлов!" << endl;

}

void deleteFile()

{

fstream list;

list.open("list.txt", ios::in);

if (list.is\_open())

{

string str, name;

vector<string> curList;

bool foundFlag = false;

cout << "Список файлов:" << endl;

while (getline(list, str))

{

cout << " " << str << endl;

curList.push\_back(str);

}

list.close();

cout << "\nИмя файла: ";

getline(cin, name);

list.open("list.txt", ios::out | ios::trunc);

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

if (curList[i] == name)

{

foundFlag = true;

remove(name.c\_str());

cout << "\nФайл удален!" << endl;

continue;

}

else

list << curList[i] << '\n';

if (!foundFlag)

cout << "\nФайла не существует!" << endl;

list.close();

}

else

cout << "Не получилось открыть список файлов!" << endl;

}

int main()

{

system("chcp 1251"); // смена кодировки консоли для коррекотного вывода кириллицы

Sleep(1000); // небольшая пауза перед выводом меню

char x;

fstream list("list.txt", ios::out | ios::app);

list.close();

do

{

system("cls");

cout << "Меню:" << endl;

cout << "1. Создать файл" << endl;

cout << "2. Открыть файл" << endl;

cout << "3. Удалить файл" << endl;

cout << "0. Выход" << endl;

cout << "\n> ";

cin >> x;

cin.ignore((numeric\_limits<streamsize>::max)(), '\n');

cout << endl;

switch (x)

{

case '0':

cout << "Bye-Bye!" << endl;

break;

case '1':

createFile();

break;

case '2':

openFile();

break;

case '3':

deleteFile();

break;

default:

cout << "Операции не существует\n";

}

cout << endl;

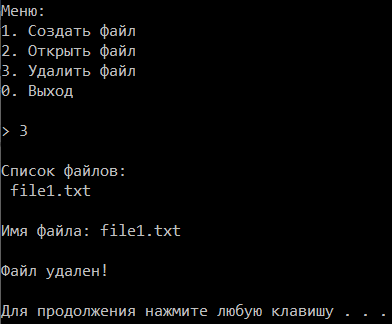
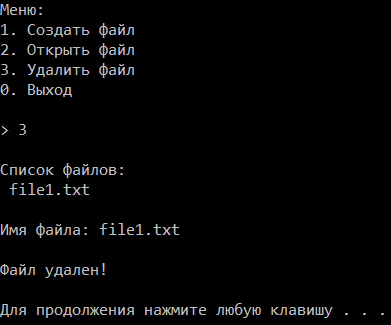
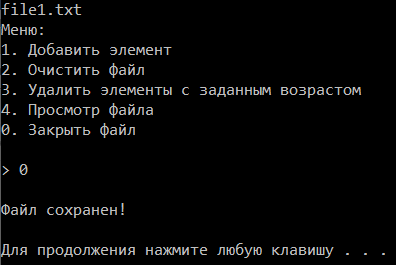
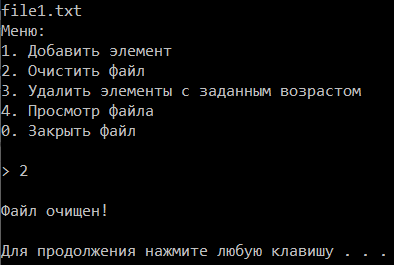
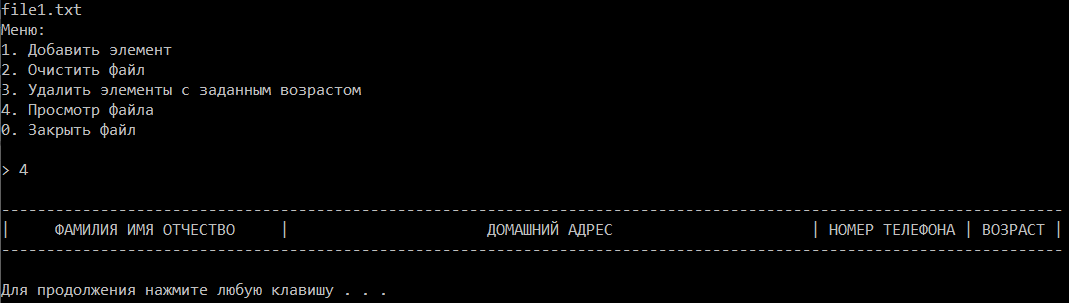
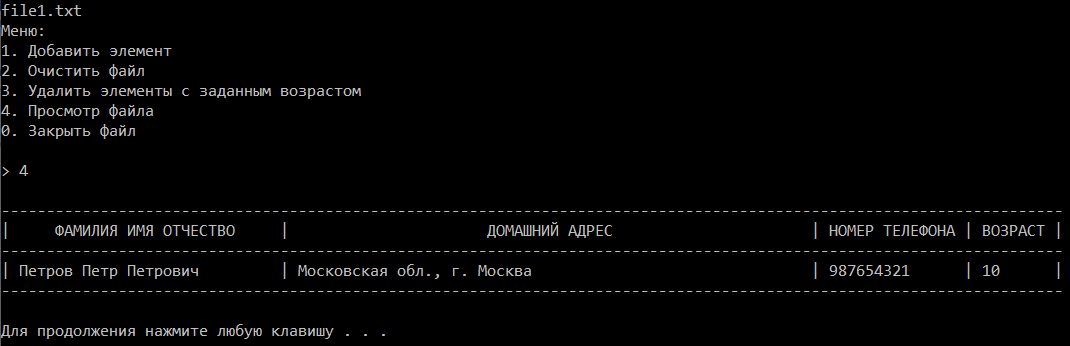
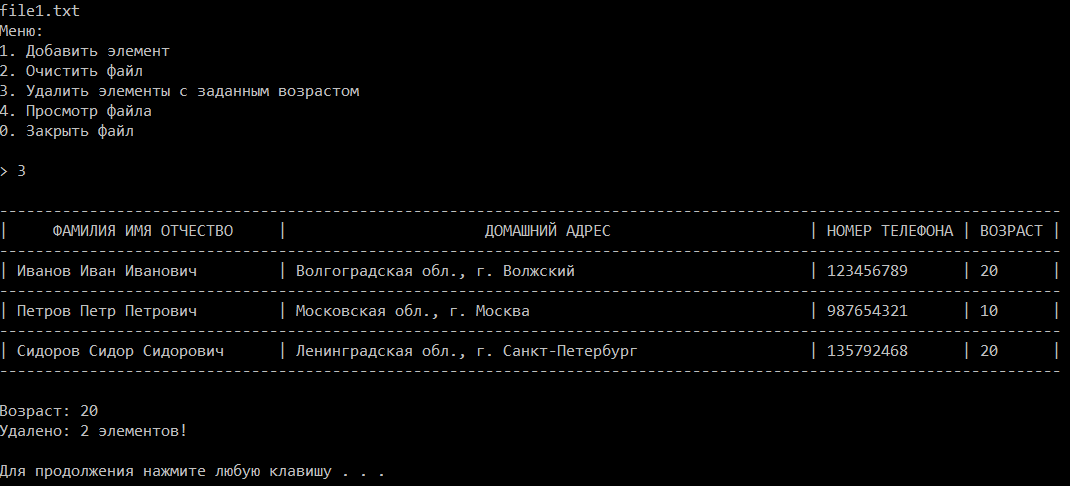
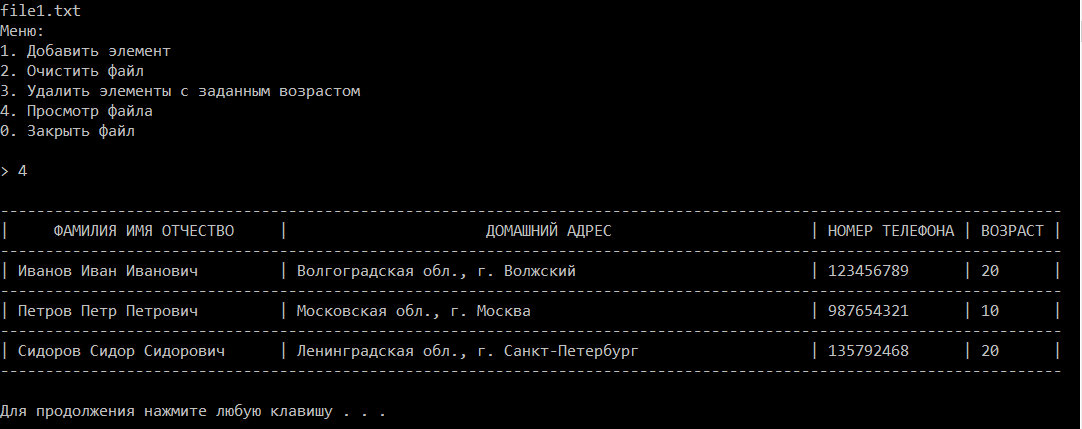
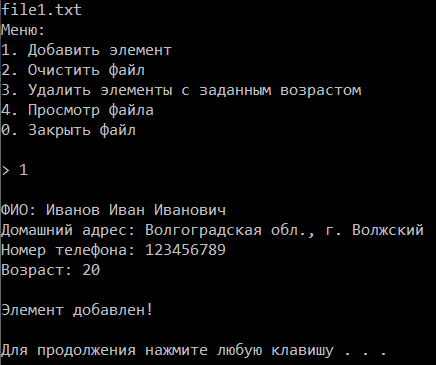
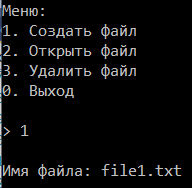
system("pause");

} while (x != '0');

return 0;

}

Результат работы программы:



Файл:

