**Министерство образования Республики Беларусь**

**Учреждение образования**

**«Брестский государственный технический университет»**

**Кафедра ИИТ**

**Лабораторная работа №12**

**По дисциплине ОАиП за II семестр**

**Тема: «Бинарные файлы»**

**Выполнил:**

Студент 1-го курса

группы ИИ-18

Туз И. С.

**Проверил:**

Михно

Брест 2020

**Вариант 17**

**Задание 1**

Необходимо разработать следующие функции:

1. Создание файла.

2. Добавление элемента в конец файла.

3. Вывод содержимого файла на экран.

4. Удаление элемента из файла:

* элемента с заданным номером;
* элемента с заданным ключом.

5. Упорядочивание элементов в файле по выбранному признаку.

«Здание»:

адрес; тип здания; количество этажей; количество квартир; срок эксплуатации;срок до капитального ремонта (25 лет - срок эксплуатации).

Код программы:

#include <iostream>

#include<string>

#include<vector>

#include<fstream>

using namespace std;

struct Building

{

string adress;

string type;

int stores;

int flats;

int time1;

int time2;

};

Building in();

string decToBe(Building build);

string toBe(int num);

int find(vector<Building>data);

void out(string path);

void infile(string path,string line);

void deleting(vector<Building>data, int k);

void sort(vector<Building>data);

int main()

{

setlocale(0, "");

bool t = true;

string path = "ForFilesLab";

int del;

while (t)

{

cout << "Выберите:" << endl;

cout << "1-Ввод данных" << endl;

cout << "2-Добавление одного элемента" << endl;

cout << "3-Вывод содержимого бинарного файла на экран" << endl;

cout << "4-Удаление элемента" << endl;

cout << "5-Сортировка данных" << endl;

cout << "6-Выход" << endl;

int N;

int n = 0;

vector<Building>data;

string str;

int menu;

cin >> menu;

switch (menu)

{

case 1://ввод данных

cout << "Скалько структур записать?" << endl;

cin >> N;

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

{

data.push\_back(in());

}

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

{

str+= decToBe(data[i]);

}

infile(path, str);

break;

case 2://добавление одного элемента

data.push\_back(in());

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

{

str+= decToBe(data[i]);

}

infile(path, str);

break;

case 3://вывод содержимого на экран

out(path);

system("pause");

system("cls");

break;

case 4://удаление элемента

cout << "Выберите:" << endl;

cout << "1-удаление структуры по номеру" << endl;

cout << "2-удаление структуры по ключу" << endl;

int q;

cin >> q;

switch (q)

{

case 1:

cout << "Введите номер структуры" << endl;

del;

cin >> del;

del--;

deleting(data, del);

break;

case 2:

del = find(data);

deleting(data, del);

break;

default:

cout << "ERROR!!!" << endl;

break;

}

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

{

str= decToBe(data[i]);

}

infile(path, str);

break;

case 5://сортировка

sort(data);

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

{

str = decToBe(data[i]);

}

infile(path, str);

break;

case 6:

t = false;

break;

default:

cout << "ERROR!!!" << endl;

break;

}

}

system("pause");

return 0;

}

string decToBe(Building build)

{

string becode;

for (int i = 0; i < build.adress.length(); i++)

{

string temp = toBe((int)build.adress[i]);

becode += temp;

}

for (int i = 0; i < build.type.length(); i++)

{

string temp = toBe((int)build.adress[i]);

becode += temp;

}

becode += toBe(build.stores);

becode += toBe(build.flats);

becode += toBe(build.time1);

becode += toBe(build.time2);

return becode;

}

string toBe(int n)

{

string result;

do

{

result += ('0' + (n % 2));

n = n / 2;

} while (n > 0);

reverse(result.begin(), result.end());

return result;

}

Building in()

{

Building\* st = new Building;

cout << "Введите адрес здания"<<endl;

cin >> st->adress;

cout << "Введите тип здания" << endl;

cin >> st->type;

cout << "Введите количество этажей" << endl;

cin >> st->stores;

cout << "Введите количество квартир" << endl;

cin >> st->flats;

cout << "Введите срок эксплуатации здания" << endl;

int t1, t2;

cin >> t1;

t1=st->time1;

t2 = 25 - t1;

t2= st->time2;

system("cls");

return \*st;

}

int find(vector<Building>data)

{

cout << "Выберете параметр для поиска" << endl;

cout << "1 - Адрес" << endl;

cout << "2 - Тип здания" << endl;

cout << "3 - Количество этажей" << endl;

cout << "4 - Количество квартир" << endl;

cout << "5 - Срок эксплуатации" << endl;

cout << "6 - Срок до капитального ремонта" << endl;

int f;

cin >> f;

int find2, findi;

string finds;

switch (f)

{

case 1:

cin >> finds;

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

{

if (data[i].adress == finds)

{

find2 = i;

break;

}

}

case 2:

cin >> finds;

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

{

if (data[i].type == finds)

{

find2 = i;

break;

}

}

case 3:

cin >> findi;

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

{

if (data[i].stores == findi)

{

find2 = i;

break;

}

}

case 4:

cin >> findi;

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

{

if (data[i].flats == findi)

{

find2 = i;

break;

}

}

case 5:

cin >> findi;

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

{

if (data[i].time1 == findi)

{

find2 = i;

break;

}

}

case 6:

cin >> findi;

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

{

if (data[i].time2 == findi)

{

find2 = i;

break;

}

}

default:

cout << "ERROR!!!" << endl;

break;

}

return find2;

}

void deleting(vector<Building>data, int k)

{

for (int i = 1; i < data.size(); i++)

{

data[k + i - 1].adress= data[k + i].adress;

data[k + i - 1].type, data[k + i].type;

data[k + i - 1].stores= data[k + i].stores;

data[k + i - 1].flats= data[k + i].flats;

data[k + i - 1].time1= data[k + i].time1;

data[k + i - 1].time2 = data[k + i].time2;

}

}

void sort(vector<Building>data)

{

Building temp;

cout << "Выберете параметр для сортировки" << endl;

cout << "1 - Адрес" << endl;

cout << "2 - Тип здания" << endl;

cout << "3 - Количество этажей" << endl;

cout << "4 - Количество квартир" << endl;

cout << "5 - Срок эксплуатации" << endl;

cout << "6 - Срок до капитального ремонта" << endl;

int q;

cin >> q;

switch (q)

{

case 1:

for (int k = 0; k < data.size(); k++)

{

for (int i = 0; i < data.size() - 1; i++)

{

if ((int)data[i].adress[0] > (int)data[i + 1].adress[0])

{

temp = data[i];

data[i] = data[i + 1];

data[i + 1] = temp;

}

}

}

break;

case 2:

for (int k = 0; k < data.size(); k++)

{

for (int i = 0; i < data.size() - 1; i++)

{

if ((int)data[i].type[0] > (int)data[i + 1].type[0])

{

temp = data[i];

data[i] = data[i + 1];

data[i + 1] = temp;

}

}

}

break;

case 3:

for (int k = 0; k < data.size(); k++)

{

for (int i = 0; i < data.size() - 1; i++)

{

if (data[i].stores > data[i + 1].stores)

{

temp = data[i];

data[i] = data[i + 1];

data[i + 1] = temp;

}

}

}

break;

case 4:

for (int k = 0; k < data.size(); k++)

{

for (int i = 0; i < data.size() - 1; i++)

{

if (data[i].flats > data[i + 1].flats)

{

temp = data[i];

data[i] = data[i + 1];

data[i + 1] = temp;

}

}

}

break;

case 5:

for (int k = 0; k < data.size(); k++)

{

for (int i = 0; i < data.size() - 1; i++)

{

if (data[i].time1 > data[i + 1].time1)

{

temp = data[i];

data[i] = data[i + 1];

data[i + 1] = temp;

}

}

}

break;

case 6:

for (int k = 0; k < data.size(); k++)

{

for (int i = 0; i < data.size() - 1; i++)

{

if (data[i].time2 > data[i + 1].time2)

{

temp = data[i];

data[i] = data[i + 1];

data[i + 1] = temp;

}

}

}

break;

default:

cout << "ERROR!!!" << endl;

break;

}

}

void out(string path)

{

ifstream file;

file.open(path);

if (file.is\_open())

{

string str;

while (getline(file, str))

cout << str << endl;

}

else cout << "ERORR!!!" << endl;

file.close();

}

void infile(string path, string line)

{

ofstream file;

file.open(path);

if (file.is\_open())

{

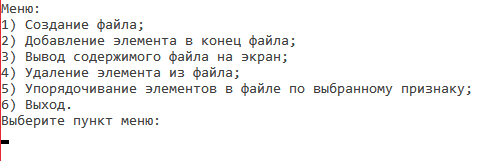
file << line;

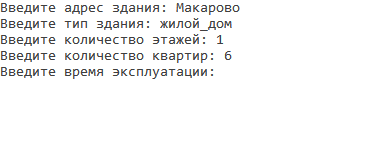
}

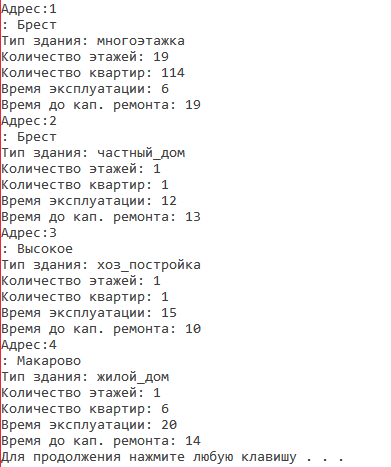
else cout << "ERORR!!!" << endl;

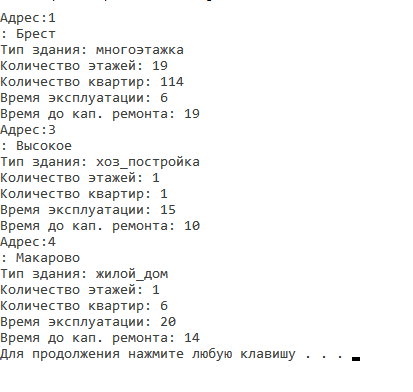
file.close();

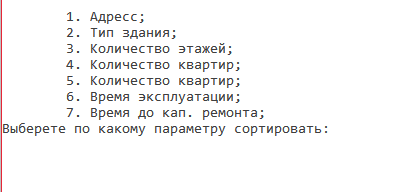
}











**Задание 2**

В первом файле хранится k матриц, во втором - l мат­риц размерности n×m. Поменять местами все нечетные (по порядковому номеру в файле)матрицы из первого и второго файлов (до конца меньшего из файлов). Вывести

на экран содержимое первого и второго файлов.

Код программы:

#include<fstream>

#include <iostream>

#include<vector>

#include<cstring>

#include<string>

using namespace std;

void swap(string\* al, string\* ak,int m,int l)

{

for (int i = 0; i < l \* m; i += (m \* 2))

{

for (int j = 0; j < m; j++)

{

string temp;

temp = ak[i + j];

ak[i + j] = al[i + j];

al[i + j] = temp;

}

}

}

int main()

{

setlocale(0, "");

ofstream fileL;

ofstream fileK;

string pathL = "D://fileL.txt";

string pathK = "D://fileK.txt";

fileL.open(pathL);

fileK.open(pathK);

if (!fileL.is\_open() || !fileK.is\_open())

{

cout << "Error!" << endl;

}

else

{

int l, k, m, n;

cout << "Введите l(количество матриц в первом файле):"; cin >> l;

cout << "\nВведите k(количество матриц во втором файле):"; cin >> k;

cout << "\nВведите m и n(размеры матриц):"; cin >> m >> n;

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

{

for (int j = 0; j < m; j++)

{

for (int r = 0; r < n; r++)

{

int a = 1;

fileK << a << " ";

}

fileK << '\n';

}

}

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

{

for (int j = 0; j < m; j++)

{

for (int r = 0; r < n; r++)

{

int a = 0;

fileL << a << " ";

}

fileL << '\n';

}

}

fileL.close();

fileK.close();

ifstream fl;

ifstream fk;

fl.open(pathL);

fk.open(pathK);

string\* al = new string[m \* l];

string\* ak = new string[m \* k];

string str;

for (int i = 0; getline(fl, str); i++)

{

al[i] = str;

}

for (int i = 0; getline(fk, str); i++)

{

ak[i] = str;

}

fk.close();

fl.close();

if (k > l)

swap(al, ak, m, l);

else

swap(al, ak, m, l);

ofstream filel;

ofstream filek;

filel.open(pathL);

filek.open(pathK);

for (int i = 0; i < m \* l; i++)

{

filel << al[i] << '\n';

}

for (int i = 0; i < m \* k; i++)

{

filek << ak[i] << '\n';

}

fl.close();

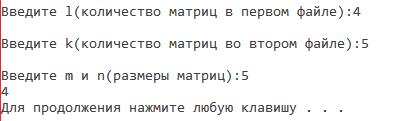
fk.close();

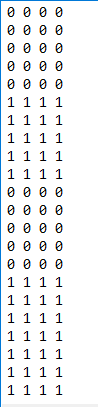
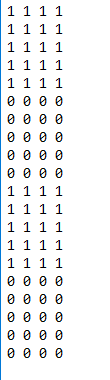
}

system("pause");

return 0;

}





**Вывод:** познакомился с темой «Бинарные файлы».