Министерство образования Республики Беларусь

Учреждение образования «Полоцкий государственный университет»

Факультет информационных технологий

Кафедра технологий программирования

**Лабораторная работа №6**

**По дисциплине «Основы алгоритмизации и программирования»**

**Тема: «Модульное программирование»**

Выполнил: Шиковец Е.А.

Группа 21-ИТ-1

Проверила: Пантелейко А.Ф

Преподаватель-стажер кафедры ТП

Полоцк 2022

**Цель работы: изучить принципы построения модулей на языке С++, а также реализацию программ, включающих несколько модулей.**

**Вариант № 29**

Создайте новый модуль для меню.

Создайте новый модуль для функций ввода/вывода данных.

Создайте новый модуль для функций работы с файлами.

2. Перенесите пользовательские функции, реализованные в предыдущих лабораторных работах, в соответствующие модули.

3. Оформите заголовочные файлы для всех модулей.

4. Подключите модули друг к другу в соответствии с тем, каким образом они будут взаимодействовать.

5. Нарисуйте модульную схему вашего приложения.

**Main.cpp**

#include "main.h"

#include "menu.h"

using namespace std;

int main()

{

T\_inf\* tenants = new T\_inf[10];

menu(tenants);

system("CLS");

free(tenants); // освобождаем память

return 0;

}

**Output.cpp**

#include "output.h"

#include "main.h"

void output(struct T\_inf\* tenants, int pos) // вывод в формате списка

{

if (pos)

{

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

{

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

cout << " Street: " << tenants[i].street << " House: " << tenants[i].house\_num << " Apart: " << tenants[i].apart\_num << " Area: " << tenants[i].area;

cout << " Rooms: " << tenants[i].rooms << " Name: " << tenants[i].name << " Birth: " << tenants[i].birth\_date << " Reg: " << tenants[i].reg\_date;

cout << " Disch: " << tenants[i].disch\_date << " Rel: " << tenants[i].rel;

cout << endl;

}

}

else

cout << "There are no elements\n";

}

void table(struct T\_inf\* tenants, int pos) // вывод в табличном формате

{

if (pos)

{

cout << setw(10) << "Street" << setw(10) << "House" << setw(10) << "Apart" << setw(10) << "Area" << setw(10) << "Rooms" << setw(10) << "Name" << setw(10) << "Birth" << setw(10) << "Reg" << setw(10) << "Disch" << setw(10) << "Rel" << endl;

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

{

cout << setw(10) << tenants[i].street << setw(10) << tenants[i].house\_num << setw(10) << tenants[i].apart\_num << setw(10) << tenants[i].area << setw(10) << tenants[i].rooms << setw(10) << tenants[i].name << setw(10) << tenants[i].birth\_date << setw(10) << tenants[i].reg\_date << setw(10) << tenants[i].disch\_date << setw(10) << tenants[i].rel;

cout << endl;

}

}

else

cout << "There are no elements\n";

}

**File.cpp**

#include "file.h"

#include "main.h"

void new\_DB(struct T\_inf\* tenants, int pos, string path)

{

ofstream fout(path, ios\_base::out | ios\_base::trunc);

if (fout.is\_open())

{

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

{

fout << tenants[i].street << " " << tenants[i].house\_num << " " << tenants[i].apart\_num << " " << tenants[i].area << " " << tenants[i].rooms << " " << tenants[i].name << " " << tenants[i].birth\_date << " " << tenants[i].reg\_date << " " << tenants[i].disch\_date << " " << tenants[i].rel;

fout << endl;

}

fout.close();

cout << "Database was created\n";

}

else

{

cout << "Something went wrong";

}

}

void save\_DB(struct T\_inf\* tenants, int pos, string path)

{

ofstream fout(path, ios\_base::out | ios\_base::trunc);

if (fout.is\_open())

{

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

{

fout << tenants[i].street << " " << tenants[i].house\_num << " " << tenants[i].apart\_num << " " << tenants[i].area << " " << tenants[i].rooms << " " << tenants[i].name << " " << tenants[i].birth\_date << " " << tenants[i].reg\_date << " " << tenants[i].disch\_date << " " << tenants[i].rel;

fout << endl;

}

fout.close();

cout << "Database was saved\n";

}

else

{

cout << "Something went wrong\n";

}

}

void remove\_DB(struct T\_inf\* tenants, int pos, string path)

{

char buf[50];

strcpy\_s(buf, path.c\_str());

if (!remove(buf))

cout << "DB was deleted succesfully\n";

else

cout << "Something went wrong\n";

}

int DB\_length(string path)

{

int pos = 0;

ifstream fin(path);

if (!fin.is\_open())

cout << "File opening error\n";

else

{

string line;

while (getline(fin, line))

{

pos++;

}

fin.close();

}

return pos;

}

struct T\_inf\* load\_DB(struct T\_inf\* tenants, string path, int pos)

{

ifstream fin(path);

if (fin.is\_open())

{

tenants = (struct T\_inf\*)malloc(pos \* sizeof(struct T\_inf\*));

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

{

tenants[i].street = new char[20];

fin >> tenants[i].street;

tenants[i].house\_num = new char[20];

fin >> tenants[i].house\_num;

tenants[i].apart\_num = new char[20];

fin >> tenants[i].apart\_num;

tenants[i].area = new char[20];

fin >> tenants[i].area;

tenants[i].rooms = new char[20];

fin >> tenants[i].rooms;

tenants[i].name = new char[20];

fin >> tenants[i].name;

tenants[i].birth\_date = new char[20];

fin >> tenants[i].birth\_date;

tenants[i].reg\_date = new char[20];

fin >> tenants[i].reg\_date;

tenants[i].disch\_date = new char[20];

fin >> tenants[i].disch\_date;

tenants[i].rel = new char[20];

fin >> tenants[i].rel;

}

cout << pos << " tenants was loaded\n";

fin.close();

}

else

cout << "DB File opening error\n";

return tenants;

}

**Input.cpp**

#include "input.h"

#include "main.h"

T\_inf\* add\_tenant(struct T\_inf\* tenants, int pos)

{

system("CLS");

cout << "Size: " << pos << endl;

cout << "-- Adding a tenant --" << endl;

// информация о квартире

cout << "Street: ";

tenants[pos].street = new char[20];

cin >> tenants[pos].street;

cout << "House number: ";

tenants[pos].house\_num = new char[20];

cin >> tenants[pos].house\_num;

cout << "Apartment number: ";

tenants[pos].apart\_num = new char[20];

cin >> tenants[pos].apart\_num;

cout << "Apartment area: ";

tenants[pos].area = new char[20];

cin >> tenants[pos].area;

cout << "Rooms: ";

tenants[pos].rooms = new char[20];

cin >> tenants[pos].rooms;

// информация о жителе

cout << "Name: ";

tenants[pos].name = new char[20];

cin >> tenants[pos].name;

cout << "Date of birth: ";

tenants[pos].birth\_date = new char[20];

cin >> tenants[pos].birth\_date;

cout << "Regestration date: ";

tenants[pos].reg\_date = new char[20];

cin >> tenants[pos].reg\_date;

cout << "Discharge date: "; // дата выписки

tenants[pos].disch\_date = new char[20];

cin >> tenants[pos].disch\_date;

cout << "Relations: "; // отношение к владельцу квартиры

tenants[pos].rel = new char[20];

cin >> tenants[pos].rel;

system("CLS");

cout << tenants[pos].name << " was succesfuly added!\n\n";

return tenants; // возвращаем новый массив

}

T\_inf\* clear(struct T\_inf\* tenants, int pos)

{

if (pos)

{

memset(tenants, 0, sizeof(T\_inf));

cout << "Structure was cleared\n";

}

else

{

cout << "Structure is already empty\n";

}

return tenants;

}

**Menu.cpp**

#include "menu.h"

#include "main.h"

#include "input.h"

#include "file.h"

#include "output.h"

void menu(struct T\_inf\* tenants)

{

int n = -1, pos = 0;

string path = "DB.txt";

while (n != 0)

{

// cout << "-- Main menu --\n";

cout << "Input 1:\n";

cout << "Output 2:\n";

cout << "Table 3:\n";

cout << "New DB 4:\n";

cout << "Load DB 5:\n";

cout << "Save DB 6:\n";

cout << "Remove DB 7:\n";

cout << "Clear structure 8:\n";

cout << "To exit enter 0:\n";

cin >> n;

switch (n)

{

case 1:

{

system("CLS");

tenants = add\_tenant(tenants, pos);

pos++;

break;

}

case 2:

{

system("CLS");

output(tenants, pos);

break;

}

case 3:

{

system("CLS");

table(tenants, pos);

break;

}

case 4:

{

system("CLS");

new\_DB(tenants, pos, path);

break;

}

case 5:

{

system("CLS");

tenants = clear(tenants, pos);

pos = DB\_length(path);

tenants = load\_DB(tenants, path, pos);

break;

}

case 6:

{

system("CLS");

save\_DB(tenants, pos, path);

break;

}

case 7:

{

system("CLS");

remove\_DB(tenants, pos, path);

break;

}

case 8:

{

system("CLS");

tenants = clear(tenants, pos);

pos = 0;

break;

}

default:

{

cout << "Check your input";

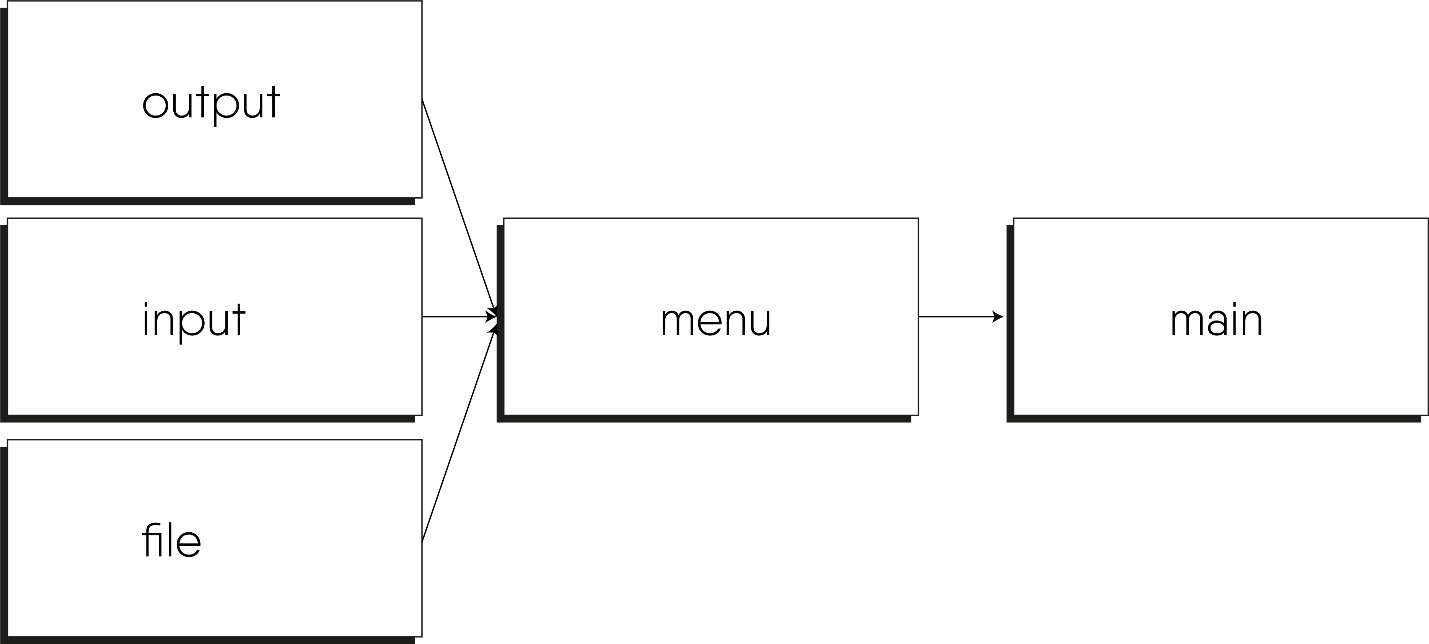
}

}

}

}

**Модульная схема приложения:**



**Вывод:**

Изучены принципы работы с модулями и функциями для работы с ними. Разработана модульная программа.