Министерство образования Республики Беларусь

Учреждение образования

“Брестский государственный университет”

Кафедра ИИТ

Лабораторная работа №2

По дисциплине “Языки программирования”

Вариант №8

Выполнил:

Крук Д.Н. (ПО-7,2)

Проверил:

Дряпко. А. В.

Дата выполнения:

16.10.21

Брест 2021

**Цель:**

Получить практические навыки создания иерархии классов и использования статических компонентов класса.

**Задание:**

1. Определить иерархию классов (в соответствии с вариантом).

2. Определить в классе статическую компоненту - указатель на нача-

ло связанного списка объектов и статическую функцию для просмотра

списка.

3. Реализовать классы.

4. Написать демонстрационную программу, в которой создаются

объекты различных классов и помещаются в список, после чего список

просматривается.

5. Сделать соответствующие методы не виртуальными и посмотреть,

что будет.

6. Реализовать вариант, когда объект добавляется в список при соз-

дании, т.е. в конструкторе (смотри пункт 6 следующего раздела).

**Иерархия классов:**

Место, область, город, мегаполис

**Код программы:**

**Файл place.h:**

#ifndef UNTITLED1\_PLACE\_H  
#define UNTITLED1\_PLACE\_H  
#include <iostream>  
  
using namespace std;  
  
class place {  
public:  
 virtual void Add() = 0;  
 virtual void Show() = 0;  
 void setName(char \*name);  
 void setSquare(float square);  
 void setPopulations(int populations);  
  
protected:  
 char \*name{};  
 float square;  
 int populations;  
};  
#endif

**Файл place.cpp:**

#include "place.h"  
  
void place::setName(char \*name) {  
 place::name = name;  
}  
  
void place::setSquare(float square) {  
 place::square = square;  
}  
  
void place::setPopulations(int populations) {  
 place::populations = populations;  
}

**Файл region.h:**

#ifndef UNTITLED1\_REGION\_H  
#define UNTITLED1\_REGION\_H  
#include "place.h"  
#include <iostream>  
  
using namespace std;  
  
class region : public place {  
protected:  
 static region \*head ;  
 region \*next;  
public:  
 region();  
 region(region &);  
 region(char \*, float, int);  
 void Show() override;  
 void Add() override;  
 static void Show\_list();  
 ~region();  
};  
  
#endif //UNTITLED1\_REGION\_H

**Файл region.cpp:**

##include "region.h"  
  
region::region() {  
 name = new char[100];  
 square = 0;  
 populations = 0;  
 next = nullptr;  
}  
  
region::region(region &copy) {  
 name = copy.name;  
 square = copy.square;  
 populations = copy.populations;  
 next = nullptr;  
}  
  
region::region(char \*new\_name, float new\_square, int new\_populations) {  
 name = new\_name;  
 square = new\_square;  
 populations = new\_populations;  
 next = nullptr;  
 Add();  
}  
  
region::~region() noexcept {  
 delete[] name;  
}  
  
void region::Add() {  
 if (head == nullptr || head == this)  
 head = this;  
 else {  
 region \*counter = head;  
 while (counter->next && counter->next != this)  
 counter = counter->next;  
 counter->next = this;  
 }  
}  
  
void region::Show() {  
 region \*counter = head;  
 while (counter) {  
 cout << "Название: " << counter->name << endl << "Площадь области: " << counter->square << endl  
 << "Население: " << counter->populations  
 << endl;  
 counter = counter->next;  
 }  
 this->next = nullptr;  
}  
  
void region::Show\_list() {  
 region \*counter = head;  
 while (counter) {  
 cout << "Название: " << counter->name << endl << "Площадь области: " << counter->square << endl  
 << "Население: " << counter->populations  
 << endl << endl;  
 counter = counter->next;  
 }  
}

**Файл town.h:**

#ifndef UNTITLED1\_TOWN\_H  
#define UNTITLED1\_TOWN\_H  
#include "place.h"  
  
class town : public place{  
protected:  
 int year;  
 static town \*head;  
 town \*next;  
public:  
 town();  
 town(town &);  
 town(char \*, float, int, int);  
 void Show() override;  
 void Add() override;  
 void setYear(int year);  
 static void Show\_list();  
 ~town();  
};  
  
#endif //UNTITLED1\_TOWN\_H

**Файл town.cpp:**

#include "town.h"  
  
town::town() {  
 name = new char[100];  
 square = 0;  
 populations = 0;  
 year = 0;  
 next = nullptr;  
}  
  
town::town(town &copy) {  
 name = copy.name;  
 square = copy.square;  
 populations = copy.populations;  
 year = copy.year;  
 next = nullptr;  
}  
  
town::town(char \*new\_name, float new\_square, int new\_populations, int new\_year) {  
 name = new\_name;  
 square = new\_square;  
 populations = new\_populations;  
 year = new\_year;  
 next = nullptr;  
}  
  
town::~town() noexcept {  
 delete[] name;  
}  
  
void town::Add() {  
 if (head == nullptr || head == this)  
 head = this;  
 else {  
 town \*counter = head;  
 while (counter->next && counter->next != this)  
 counter = counter->next;  
 counter->next = this;  
 }  
}  
  
void town::Show() {  
 town \*counter = head;  
 while (counter) {  
 cout << "Название: " << counter->name << endl << "Площадь области: " << counter->square << endl  
 << "Населенние: " << counter->populations << endl << "Год основания: " << year << endl << endl;  
 counter = counter->next;  
 }  
}  
  
void town::Show\_list() {  
 town \*counter = head;  
 while (counter) {  
 cout << "Название: " << counter->name << endl << "Площадь области: " << counter->square << endl  
 << "Населенние: " << counter->populations << endl << "Год основания: " << counter->year << endl << endl;  
 counter = counter->next;  
 }  
}  
  
void town::setYear(int year) {  
 town::year = year;  
}

**Файл metropolis.h:**

#ifndef UNTITLED1\_METROPOLIS\_H  
#define UNTITLED1\_METROPOLIS\_H  
#include "town.h"  
  
class metropolis : public town {  
protected:  
 float gdp;  
 static metropolis \*head;  
 metropolis \*next;  
public:  
 metropolis();  
 metropolis(metropolis &);  
 metropolis(char \*, float, int, int, float);  
 void Show() override;  
 void Add() override;  
 void setGdp(float gdp);  
 static void Show\_list();  
 ~metropolis();  
};  
  
#endif

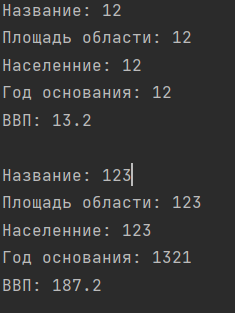
**Файл metropolis.cpp:**

#include "metropolis.h"  
  
metropolis::metropolis() {  
 name = new char[100];  
 square = 0;  
 populations = 0;  
 year = 0;  
 gdp = 0;  
 next = nullptr;  
}  
  
metropolis::metropolis(metropolis &copy) {  
 name = copy.name;  
 square = copy.square;  
 populations = copy.populations;  
 gdp = copy.gdp;  
 \*this = copy;  
 next = nullptr;  
  
}  
  
metropolis::metropolis(char \*new\_name, float new\_square, int new\_populations, int new\_year, float new\_gdp) {  
 name = new\_name;  
 square = new\_square;  
 populations = new\_populations;  
 year = new\_year;  
 gdp = new\_gdp;  
 next = nullptr;  
 Add();  
}  
  
metropolis::~metropolis() noexcept {  
 delete[] name;  
}  
  
void metropolis::Add() {  
 if (head == nullptr || head == this)  
 head = this;  
 else {  
 metropolis \*counter = head;  
 while (counter->next && counter->next != this)  
 counter = counter->next;  
 counter->next = this;  
 }  
}  
  
void metropolis::Show() {  
 metropolis \*counter = head;  
 while (counter) {  
 cout << "Название: " << counter->name << endl << "Площадь области: " << counter->square << endl  
 << "Населенние: " << counter->populations << endl << "Год основания: " << counter->year << endl  
 << "ВВП: " << counter->gdp << endl << endl;  
 counter = counter->next;  
 }  
}  
  
void metropolis::Show\_list() {  
 metropolis \*counter = head;  
 while (counter) {  
 cout << "Название: " << counter->name << endl << "Площадь области: " << counter->square << endl  
 << "Населенние: " << counter->populations << endl << "ВВП: " << counter->gdp << endl << endl;  
 counter = counter->next;  
 }  
}  
  
void metropolis::setGdp(float gdp) {  
 metropolis::gdp = gdp;  
}

**Файл main.cpp:**

#include <windows.h>  
#include "region.h"  
#include "metropolis.h"  
  
town\* town::head = nullptr;  
region\* region::head = nullptr;  
metropolis\* metropolis::head = nullptr;  
  
  
int main() {  
 SetConsoleCP(CP\_UTF8);  
 SetConsoleOutputCP(CP\_UTF8);  
 metropolis a1("12", 12, 12, 12, 13.2);  
 metropolis a2("123", 123, 123, 1321, 187.2);  
 town s1("12", 123, 123, 32);  
 s1.Add();  
 town\* f = &a2;  
 f->Show();  
 return 0;  
}

**Результат работы программы:**



**Вывод:**

Получил практические навыки создания иерархии классов и использования статических компонентов класса.