Звіт

Автор: Капелька Я.І. КІТ-119а Дата: 19 травня 2020

Лабораторна робота №2. Перевантаження методів.

Тема. Класи. Конструктори та деструктори. Перевантаження методів.

Мета: отримати базові знання про класи, конструктори та деструктори.

Дослідити механізм створення та видалення об’єктів.

1. Завдання до роботи **Індивідуальне завдання**:

Визначити, яка країна має найменшу щільність населення.

2. Опис класів, змінних, методів та функцій

2.1 Опис класів

Базовий клас: CCountry

Клас, що має в собі масив базового класу та методи для роботи з ним: CMetod

2.2 Опис змінних

`int number_of_cities` – поле класу CCountry(кількість міст.).

`int population` – поле класу CCountry(популяція).

`int area` – поле класу CCountry(площа).

`int uncial_index` – поле класу CCountry(унікальний індекс).

`int population_density` – поле класу CCountry(щільність населення).

`const char* title` – поле класу CCountry(назва країни).

`int next_i` – поле класу CMetod(номер наступного файлу у директорії).

`int new_i` – поле класу CMetod(індекс наступного файлу у директорії).

`CCountry* countries` – поле класу CMetod(масив елементів класу

CCountry).

`CCountry* copy` – поле класу CMetod(показчик на клас CCountry, використовується для правильної роботи деяких методів).

2.3 Опис методів

`int getNumber_of_cities () const` – отримання значення поля number_of_cities змінної класу CCountry(метод класу CCountry).

`int getPopulation () const` – отримання значення поля population змінної класу CCountry(метод класу CCountry).

`int getArea () const` – отримання значення поля area змінної класу CCountry(метод класу CCountry).

`int getUnical_index () const` – отримання значення поля unical_index змінної класу CCountry(метод класу CCountry).

`int getPopulation_density () const` – отримання значення поля population_density змінної класу CCountry(метод класу CCountry).

`const char* getTitle () const` – отримання значення поля title змінної класу CCountry(метод класу CCountry).

`void setNumber_of_cities (const int &Number_of_cities)` – зміна значення поля number_of_cities змінної класу CCountry(метод класу CCountry).

`void setPopulation (const int &Population)` – зміна значення поля population змінної класу CCountry(метод класу CCountry).

`void setArea (const int &Area)` – зміна значення поля area змінної класу CCountry(метод класу CCountry).

`void setUnical_index (const int& Unical_index)` – зміна значення поля unical_index змінної класу CCountry(метод класу CCountry).

`void setPopulation_density (const int& Population_density)` – зміна значення поля population_density змінної класу CCountry(метод класу CCountry).

`void setTitle (const char* Title)` – зміна значення поля title змінної класу CCountry(метод класу CCountry).

`CCountry()` – конструктор класу CCountry.

`CCountry(const CCountry&)` – конструктор копіювання класу CCountry.

`CCountry(const char*, const int&, const int&, const int&)` – конструктор з параметрами класу CCountry.

`~CCountry()` – деструктор класу CCountry.

`void add_el(const CCountry & CCountry)` – додавання об’єкту класу CCountry до масиву в класі CMetod(метод класу CMetod).

`void remove_el(const int &index)` – видалення об’єкту класу CCountry з масиву в класі CMetod(метод класу CMetod).

`void del_all()` – видалення усіх об’єктів класу CCountry з масиву в класі CMetod(метод класу CMetod).

`CCountry find_to_index(const int& index) const` – отримання об’єкту класу CCountry з масиву в класі CMetod(метод класу CMetod).

`void get_to_Screen(const int &index) const` – виведення об’єкту класу CCountry з масиву в класі CMetod на екран(метод класу CMetod).

`void print_all() const` – виведення усіх об’єктів класу CCountry з масиву в класі CMetod на екран(метод класу CMetod).

`void find_to_population_density() const` –визначення, яка країна має найменшу щільність населення в об’єкті класу CMetod(метод класу CMetod).

2.4 Опис функцій

`void menu()` – функція меню.

3 Текст програми

Лабораторная работа №2.cpp

```
#pragma once
// Лабораторная работа №2.cpp : Этот файл содержит функцию "main". Здесь начинается и заканчивается выполнение программы.
//

#include <iostream>
#include "menu.h"
#define _CRTDBG_MAP_ALLOC

int main()
{
    menu();
    if (_CrtDumpMemoryLeaks())
    {
        std::cout << "Утечка памяти обнаружена." << "\n";
    }
    else
    {
        std::cout << "Утечка памяти не обнаружена." << "\n";
    }
}
```

CCountry.h

```
#pragma once
#include <iostream>
class CCountry
{
private:
    const char* title;
    int population_density;
    int number_of_cities;
    int population;
    int area;
    int unical_index;
public:
    CCountry();
    CCountry(const char*, const int&, const int&, const int&);
    ~CCountry();
    CCountry(const CCountry&);
    const char* getTitle() const;
    int getPopulation_density() const;
    int getNumber_of_cities() const;
    int getPopulation() const;
    int getArea() const;
    int getUnical_index() const;
    void setTitle(const char*);
    void setPopulation_density(const int&);
    void setNumber_of_cities(const int&);
    void setPopulation(const int&);
    void setArea(const int&);
    void setUnical_index(const int&);
};
```

CCountry.cpp

```
#include "CCountry.h"
const char* CCountry::getTitle() const { return title; }
int CCountry::getPopulation_density() const { return population_density; }
int CCountry::getNumber_of_cities() const { return number_of_cities; }
int CCountry::getPopulation() const { return  population; }
int CCountry::getArea() const { return area; }
int CCountry::getUnical_index() const { return unical_index; }
void CCountry::setTitle(const char* Title) { title = Title; }
void CCountry::setPopulation_density(const int& Population_density) { population_density =
Population_density; }
void CCountry::setNumber_of_cities(const int& Number_of_cities) { number_of_cities =
Number_of_cities; }
void CCountry::setPopulation(const int& Population) { population = Population; }
void CCountry::setArea(const int& Area) { area = Area; }
void CCountry::setUnical_index(const int& Unical_index) { unical_index = Unical_index; }
CCountry::CCountry()
{
	title = "CCountry";
	population_density = 1000;
	number_of_cities = 100;
	population = 1000000;
	area = 10000000;
	unical_index = 0;
	std::cout << "Файл создан при помощи конструктора по умолчанию." << "\n";
}
CCountry::CCountry(const CCountry& CCountry):title(CCountry.getTitle()),
population_density(CCountry.getPopulation_density()),
number_of_cities(CCountry.getNumber_of_cities()), population(CCountry.getPopulation()),
area(CCountry.getArea()), unical_index(CCountry.getUnical_index()){}
CCountry::CCountry(const char* Title, const int& Number_of_cities, const int& Population, const int&
Area):title(Title), number_of_cities(Number_of_cities), population(Population),area(Area),
population_density(Area / Population)
{
	std::cout << "Файл создан при помощи конструктора с аргументами." << "\n";
}
CCountry::~CCountry()
{
	std::cout << "Файл уничтожен при помощи деструктора по умолчанию." << "\n";
}
```

CMetod.h

```
#include "CCountry.h"
class CMetod
{
private:
    CCountry* countries;
    CCountry* copy;
    int next_i = 0;
    int new_i = 1;
public:
    void add_el(const CCountry& CCountry);
    void remove_el(const int& index);
    void del_all();
    void get_to_Screen(const int& index) const;
    CCountry find_to_index(const int& index) const;
    void print_all() const;
    void find_to_population_density() const;
};
```

CMetod.cpp

```
#include "CMetod.h"
void CMetod::add_el(const CCountry& Country)
{
	if (next_i == 0)
	{
		countries = (CCountry*)malloc(sizeof(CCountry));
		countries[next_i] = Country;
		next_i++;
	}
	else
```

```
	{
		copy = (CCountry*)malloc(sizeof(CCountry) * (next_i + 1));
		for (int i = 0; i < next_i; i++)
		{
			copy[i] = countries[i];
		}
		free(countries);
		countries = (CCountry*)malloc(sizeof(CCountry) * (next_i + 1));
		for (int i = 0; i < next_i; i++)
		{
			countries[i] = copy[i];
		}
		free(copy);
		countries[next_i] = Country;
		next_i++;
	}
}
void CMetod::remove_el(const int& index)
{
	if (next_i == 1)
	{
		free(countries);
		next_i--;
	}
	else
	{
		copy = (CCountry*)malloc(sizeof(CCountry) * (next_i - 1));
		for (int i = 0; i < index; i++)
		{
			copy[i] = countries[i];
		}
		for (int i = index + 1; i < next_i; i++)
		{
			copy[i - 1] = countries[i];
		}
		free(countries);
		countries = (CCountry*)malloc(sizeof(CCountry) * (next_i - 1));
		for (int i = 0; i < next_i; i++)
		{
			countries[i] = copy[i];
		}
		free(copy);
		next_i--;
	}
}
void CMetod::del_all()
{
	free(countries);
	next_i = 0;
}
void CMetod::get_to_Screen(const int& index) const
{
	std::cout << "Number_of_cities " << "Population " << "Area " << "\n";
	std::cout << countries[index].getNumber_of_cities() << "               " <<
countries[index].getPopulation() << "    " << countries[index].getArea() << "\n";
}
CCountry CMetod::find_to_index(const int& index) const
{
	for (int i = 0; i < next_i; i++)
	{
		if (countries[i].getUnical_index() == index)
		{
			return countries[i];
		}
	}
}
void CMetod::print_all() const
{
	for (int i = 0; i < next_i; i++)
```

```
	{
		get_to_Screen(i);
	}
}
void CMetod::find_to_population_density() const
{
	float min = countries[0].getPopulation_density();
	for (int i = 0; i < next_i; i++)
	{
		if (min > countries[i].getPopulation_density())
		{
			min = countries[i].getPopulation_density();
		}
	}
	for (int i = 0; i < next_i; i++)
	{
		if (countries[i].getPopulation_density() == min)
			get_to_Screen(i);
	}
}
```

menu.h

```
#pragma once
#include "CMetod.h"


void menu();
```

menu.cpp

```
#include "menu.h"
void menu()
{
    setlocale(LC_ALL, "Russian");
    int n = 0, temp_i;
    CMetod dir;
    CCountry firstcountry1("1-я страна", 143, 45745656, 47342362);
    dir.add_el(firstcountry1);
    CCountry firstcountry2("2-я страна", 156, 38567454, 68457458);
    dir.add_el(firstcountry2);
    CCountry firstcountry3("3-я страна", 167, 46357625, 98686453);
    dir.add_el(firstcountry3);
    CCountry firstcountry4("4-я страна", 179, 78567583, 68457458);
    dir.add_el(firstcountry4);
    while (n != 6)
    {
        std::cout << "_-_-_-Выберите желаемую опцию:_-_-_-_-_-_-_-_-_-_-_-_-" << "\n";
        std::cout << "_-1 - добавить элемент в список.-_-_-_-_-_-_-_-_-_-_-_" << "\n";
        std::cout << "_-2 - получить элемент в списке по индексу.-_-_-_-_-" << "\n";
        std::cout << "_-3 - удалить элемент из списка.-_-_-_-_-_-_-_-_-_-_-_" << "\n";
        std::cout << "_-4 - показать все элементы списка.-_-_-_-_-_-_-_-_-_-" << "\n";
        std::cout << "_-5 - найти наименьшую плотность населения страны.-_" << "\n";
        std::cout << "_-6 - завершить работу программы.-_-_-_-_-_-_-" << "\n";
        std::cin >> n;
        if (n == 1)
        {
            CCountry firstcountry5("5-я страна", 323, 93645665, 78767464);
            dir.add_el(firstcountry5);
            std::cout << "Страна добавлена." << "\n";
        }
        else if (n == 2)
        {
            std::cout << "Введите индекс нового элемента: ";
            std::cin >> temp_i;
            dir.find_to_index(temp_i);
        }
        else if (n == 3)
        {
            std::cout << "Введите номер удаляемого элемента (нумерация начинается с 1): ";
            std::cin >> temp_i;
            dir.remove_el(temp_i - 1);
            std::cout << "Страна удалена";
```

```
        }
        else if (n == 4)
        {
            dir.print_all();
        }
        else if (n == 5)
        {
            dir.find_to_population_density();
        }
    }
    dir.del_all();
}
```

test.cpp

```
#pragma once
#include <iostream>
#include "menu.h"
#define _CRTDBG_MAP_ALLOC
int main()
{
	setlocale(LC_ALL, "Russian");
	CCountry firstcountry6("Тест страна", 200, 2000000, 10000000);
	if ((firstcountry6.getNumber_of_cities() == 200) && (firstcountry6.getPopulation() ==
2000000) && (firstcountry6.getArea() == 10000000))
	{
		std::cout << "Первый тест на работу геттеров и сеттеров базового класса пройден
успешно." << "\n";
	}
	else
	{
		std::cout << "Первый тест на работу геттеров и сеттеров базового класса провален." <<
"\n";
	}
	CMetod test_metod;
	test_metod.add_el(firstcountry6);
	test_metod.print_all();
	std::cout << "Если перед этим сообщением на экран вывелась информация о то файле методы
add_el, print_all и get_to_Screen работают корректно." << "\n";
	test_metod.del_all();
	test_metod.find_to_population_density();
	test_metod.print_all();
	std::cout << "Если перед этим сообщение на экран не выводились новые числа то методы
find_to_population_density, del_all и remove_el работают корректно." << "\n";

	if (_CrtDumpMemoryLeaks())
	{
		std::cout << "Утечка памяти обнаружена." << "\n";
	}
	else
	{
		std::cout << "Утечка памяти не обнаружена." << "\n";
	}
	int t;
	std::cin >>
}
```

4. Результати роботи програми

Результати роботи програми:

```
Файл создан при помощи конструктора с аргументами.
Файл создан при помощи конструктора с аргументами.
Файл создан при помощи конструктора с аргументами.
Файл создан при помощи конструктора с аргументами.
_-_-_-Выберите желаемую опцию:_-_-_-_-_-_-_-_-_-_-_-_-_-
_-1 - добавить элемент в список.-_-_-_-_-_-_-_-_-_-_-_-_
_-2 - получить элемент в списке по индексу.-_-_-_-_-
_-3 - удалить элемент из списка.-_-_-_-_-_-_-_-_-_-_-_-
_-4 - показать все элементы списка.-_-_-_-_-_-_-_-_-_-
_-5 - найти наименьшую плотность населения страны.-_
_-6 - завершить работу программы.-_-_-_-_-_-_-
4
Number_of_cities Population Area
143              45745656   47342362
Number_of_cities Population Area
156              38567454   68457458
Number_of_cities Population Area
167              46357625   98686453
Number_of_cities Population Area
179              78567583   68457458
_-_-_-Выберите желаемую опцию:_-_-_-_-_-_-_-_-_-_-_-_-_
_-1 - добавить элемент в список.-_-_-_-_-_-_-_-_-_-_-_-_
_-2 - получить элемент в списке по индексу.-_-_-_-_-
_-3 - удалить элемент из списка.-_-_-_-_-_-_-_-_-_-_-_-
_-4 - показать все элементы списка.-_-_-_-_-_-_-_-_-_-
_-5 - найти наименьшую плотность населения страны.-_
_-6 - завершить работу программы.-_-_-_-_-_-_-
5
Number_of_cities Population Area
179              78567583   68457458
_-_-_-Выберите желаемую опцию:_-_-_-_-_-_-_-_-_-_-_-_-_
_-1 - добавить элемент в список.-_-_-_-_-_-_-_-_-_-_-_-_
_-2 - получить элемент в списке по индексу.-_-_-_-_-
_-3 - удалить элемент из списка.-_-_-_-_-_-_-_-_-_-_-_-
_-4 - показать все элементы списка.-_-_-_-_-_-_-_-_-_-
_-5 - найти наименьшую плотность населения страны.-_
_-6 - завершить работу программы.-_-_-_-_-_-_-
6
Файл уничтожен при помощи деструктора по умолчанию.
Файл уничтожен при помощи деструктора по умолчанию.
Файл уничтожен при помощи деструктора по умолчанию.
Файл уничтожен при помощи деструктора по умолчанию.
Утечка памяти не обнаружена.
```

Результати тестів:

```
Файл создан при помощи конструктора с аргументами.
Первый тест на работу геттеров и сеттеров базового класса пройден успешно.
Number_of_cities Population Area
200              2000000   10000000
Если перед этим сообщением на экран вывелась информация о то файле методы add_el, print_all и get_to_Screen работают корректно.
Если перед этим сообщение на экран не выводились новые числа то методы find_to_population_density, del_all и remove_el работают корректно.
Утечка памяти не обнаружена.
```

5. Висновки

При виконанні даної лабораторної роботи було набуто практичного досвіду роботи з класами та їх конструкторами та деструкторами.

Програма протестована, витоків пам’яті немає, виконується без помилок.