Звіт

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

Лабораторна робота №3. Потоки

Тема. Робота з потоками: потокове введення / виведення на консоль та у файл, рядки типу string, stringstream.

Мета: отримати знання про основи роботи з потоковим введенням / виведенням на мові С++, роботу з файлами та рядками типу string.

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

Змінити попередню лабораторну роботу для роботи з потоками і строками.

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).

`std::string 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).

`std::string 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 std::string& Title)` – зміна значення поля title змінної класу CCountry(метод класу CCountry).

`CCountry()` – конструктор класу CCountry.

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

`CCountry(const std::string, const int&, 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).

`void find_to_str_by_file (const std::string& str)` – додавання об’єкту класу CCountry до масиву в класі CMetod за допомогою строки з інформацією про об’єкт(метод класу CMetod).

`void read_from_file(const std::string& name)` – заповнення масиву об’єктів класу CCountry інформація про які буде зчитана з файлу(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).

`void write_to_file (const std::string& name) const` – запис у файл інформації про об’єкти класу CCountry що є в масиві(метод класу CMetod).

2.4 Опис функцій

`void menu()` – функція меню.

3 Текст програми

Лабораторная работа №3.cpp

```
#pragma once
// Лабораторная работа №3.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>
#include <string>
#include <fstream>
#include <sstream>
class CCountry
{
private:
    std::string title;
    int population_density;
    int number_of_cities;
    int population;
    int area;
```

```
    int unical_index;
public:
    CCountry();
    CCountry(const std::string, const int&, const int&, const int&, const int&);
    ~CCountry();
    CCountry(const CCountry&);
    const std::string getTitle() const;
    int getPopulation_density() const;
    int getNumber_of_cities() const;
    int getPopulation() const;
    int getArea() const;
    int getUnical_index() const;
    void setTitle(const std::string&);
    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 std::string 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 std::string& 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)
{
	int i = 0;
	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 std::string Title, const int& Number_of_cities, const int& Population,
const int& Area, const int& Unical_index)
{
	int i = 0;
	title = Title;
	number_of_cities = Number_of_cities;
	population = Population;
	area = Area;
	population_density = Area / Population;
	unical_index = Unical_index;
	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;
    void find_to_str_by_file(const std::string str);
    std::string get_str_by_file(const int& index) const;
    void write_to_file(const std::string name);
    void read_from_file(const std::string name);
};
```

CMetod.cpp

```
#include "CMetod.h"
void CMetod::add_el(const CCountry& Country)
{
	if (next_i == 0)
	{
		countries = new CCountry[1];
		countries[next_i] = Country;
		next_i++;
	}
	else
	{
		copy = new CCountry[next_i + 1];
		for (int i = 0; i < next_i; i++)
		{
			copy[i] = countries[i];
		}
		delete[] countries;
		countries = new CCountry[next_i + 1];
		for (int i = 0; i < next_i; i++)
		{
			countries[i] = copy[i];
		}
		delete[] copy;
		countries[next_i] = Country;
		next_i++;
	}
}
void CMetod::remove_el(const int& index)
{
	if (next_i == 1)
	{
		delete[] countries;
		next_i--;
	}
	else
	{
		copy = new CCountry[next_i - 1];
		for (int i = 0; i < index; i++)
		{
			copy[i] = countries[i];
		}
		for (int i = index, j = index + 1; i < (next_i - 1), j < next_i; i++, j++)
		{
			copy[i] = countries[j];
```

```
			}
			delete[] countries;
			countries = new CCountry[next_i - 1];
			for (int i = 0; i < next_i - 1; i++)
			{
				countries[i] = copy[i];
			}
			delete[] copy;
			next_i--;
		}
}
void CMetod::del_all()
{
		if (next_i != 0)
		{
			delete[] countries;
			next_i = 0;
		}

}
void CMetod::get_to_Screen(const int& index) const
{
		std::cout << "Title " << "Number_of_cities " << "Population " << "Area " << "Unical_index "
<< "\n";
		std::cout << get_str_by_file(index) << "\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);
		}
}
std::string CMetod::get_str_by_file(const int& index) const
{
		std::stringstream ss;
	ss << countries[index].getTitle() << " " << countries[index].getNumber_of_cities() << " " <<
countries[index].getPopulation() << " " << countries[index].getArea() << " " <<
countries[index].getUnical_index();
		return ss.str();
}
void CMetod::find_to_str_by_file(const std::string str)
{
```

```
	int i = str.find(" ");
	std::string Title = str.substr(0, i);
	int i2 = str.find(" ", i + 1);
	std::string temp = str.substr(i + 1, i2 - i);
	std::stringstream s;
	s << temp;
	int Number_of_cities;
	s >> Number_of_cities;
	int i3 = str.find(" ", i2 + 1);
	s.clear();
	temp = str.substr(i2 + 1, i3 - i2);
	s << temp;
	int Population;
	s >> Population;
	int i4 = str.find(" ", i3 + 1);
	s.clear();
	temp = str.substr(i3 + 1, i4 - i3);
	s << temp;
	int Area;
	s >> Area;
	int i5 = str.find(" ", i4 + 1);
	s.clear();
	temp = str.substr(i4 + 1, i5 - i4);
	s << temp;
	int Unical_index;
	s >> Unical_index;
	int i6 = str.find(" ", i5 + 1);
	s.clear();
	temp = str.substr(i5 + 1, i6 - i5);
	s << temp;
	CCountry secondcountry(Title, Number_of_cities, Population, Area, Unical_index);
	add_el(secondcountry);
}
void CMetod::write_to_file(const std::string name)
{
	std::ofstream fout("text.txt");
	std::string s;
	for (int i = 0; i < next_i; i++)
	{
		s = get_str_by_file(i);
		fout << s;
		if (i != next_i - 1)
		{
			fout << "\n";
		}
	}
	fout.close();
}
void CMetod::read_from_file(const std::string name)
{
	del_all();
	std::ifstream fin("text.txt");
	char* check;
	while (!fin.eof())
	{
		check = new char[100];
		fin.getline(check, 100);
		find_to_str_by_file(check);
		delete[] check;
	}
	fin.close();
}
```

menu.h

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


void menu();
```

menu.cpp

```cpp
#include "menu.h"
void menu()
{
    setlocale(LC_ALL, "Russian");
    int n = 0, temp_i;
    CMetod dir;
    CCountry firstcountry1("страна1", 143, 45745656, 47342362, 1);
    dir.add_el(firstcountry1);
    CCountry firstcountry2("страна2", 156, 38567454, 68457458, 0);
    dir.add_el(firstcountry2);
    CCountry firstcountry3("страна3", 167, 46357625, 98686453, 1);
    dir.add_el(firstcountry3);
    CCountry firstcountry4("страна4", 179, 78567583, 68457458, 0);
    dir.add_el(firstcountry4);
    while (n != 8)
    {
        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 - записать данные а файл.-_-_-_-_-_-_-_-_-_-_-_-" << "\n";
        std::cout << "_-7 - считать данные из файла.-_-_-_-_-_-_-_-_-_-_-_" << "\n";
        std::cout << "_-8 - завершить работу программы.-_-_-_-_-_-_-_-_-_-" << "\n";
        std::cin >> n;
        if (n == 1)
        {
            CCountry firstcountry5("страна6", 323, 93645665, 78767464, 1);
            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" << "\n";
        }
        else if (n == 4)
        {
            dir.print_all();
        }
        else if (n == 5)
        {
            dir.find_to_population_density();
        }
        else if (n == 6)
        {
            dir.write_to_file("text.txt");
        }
        else if (n == 7)
        {
            dir.read_from_file("text.txt");
        }
    }
    dir.del_all();
}
```

tests.cpp

```
#pragma once
#include <iostream>
#include "menu.h"
#define _CRTDBG_MAP_ALLOC
int main()
{
	setlocale(LC_ALL, "Russian");
	CCountry firstcountry6("страна0", 200, 2000000, 10000000, 1);
	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";
	test_metod.write_to_file("text.txt");
	test_metod.read_from_file("text.txt");
	std::cout << "Если перед этим сообщение на экран не выводились новые числа то методы
write_to_file и read_from_file работают корректно." << "\n";
	if (_CrtDumpMemoryLeaks())
	{
		std::cout << "Утечка памяти обнаружена." << "\n";
	}
	else
	{
		std::cout << "Утечка памяти не обнаружена." << "\n";
	}
	int t;
	std::cin >> t;
}
```

text.txt

```
страна2 156 38567454 68457458 0
страна3 167 46357625 98686453 1
страна4 179 78567583 68457458 0
страна6 323 93645665 78767464 1
```

4. Результати роботи програми

Результати роботи програми:

```
Файл создан при помощи конструктора с аргументами.
Файл создан при помощи конструктора по умолчанию.
Файл создан при помощи конструктора по умолчанию.
Файл создан при помощи конструктора по умолчанию.
Файл уничтожен при помощи деструктора по умолчанию.
Файл уничтожен при помощи деструктора по умолчанию.
Файл создан при помощи конструктора по умолчанию.
Файл создан при помощи конструктора по умолчанию.
Файл создан при помощи конструктора по умолчанию.
Файл уничтожен при помощи деструктора по умолчанию.
Файл уничтожен при помощи деструктора по умолчанию.
Файл уничтожен при помощи деструктора по умолчанию.
Файл создан при помощи конструктора с аргументами.
Файл создан при помощи конструктора по умолчанию.
Файл создан при помощи конструктора по умолчанию.
Файл создан при помощи конструктора по умолчанию.
Файл создан при помощи конструктора по умолчанию.
Файл уничтожен при помощи деструктора по умолчанию.
Файл уничтожен при помощи деструктора по умолчанию.
Файл уничтожен при помощи деструктора по умолчанию.
Файл создан при помощи конструктора по умолчанию.
Файл создан при помощи конструктора по умолчанию.
Файл создан при помощи конструктора по умолчанию.
Файл создан при помощи конструктора по умолчанию.
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.
_-_-_-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 - записать данные а файл.-_-_-_-_-_-_-_-_-_-_-_-
_-7 - считать данные из файла.-_-_-_-_-_-_-_-_-_-_-_
_-8 - завершить работу программы.-_-_-_-_-_-_-_-_-_-
8
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.
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.
Файл создан при помощи конструктора по умолчанию.
Title Number_of_cities Population Area Unical_index
страна0 200 2000000 10000000 1
Если перед этим сообщением на экран вывелась информация о то файле методы add_el, print_all и get_to_Screen работают корректно.
Файл уничтожен при помощи деструктора по умолчанию.
Если перед этим сообщение на экран не выводились новые числа то методы find_to_population_density, del_all и remove_el работают корректно.
Файл создан при помощи конструктора с аргументами.
Файл создан при помощи конструктора по умолчанию.
Файл уничтожен при помощи деструктора по умолчанию.
Если перед этим сообщение на экран не выводились новые числа то методы write_to_file и read_from_file работают корректно.
Утечка памяти не обнаружена.
```

5. Висновки

При виконанні даної лабораторної роботи було набуто практичного досвіду роботи з потоками, строками та файлами.

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