

# Звіт

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

## Лабораторна робота №4. Регулярні вирази

**Тема.** Регулярні вирази.

**Мета:** отримати знання про базові регулярні вирази та досвід роботи із застосування їх на практиці.

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

Змінити попередню лабораторну роботу для роботи з потоками і строками за допомогою регулярних виразів.

### 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: CMethod

#### 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` `unical_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 CMethod(номер наступного файлу у директорії).

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

`CCountry*` `countries` – поле класу CMethod(масив елементів класу CCountry).

`CCountry*` `copy` – поле класу CMethod (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`).

`void get_str_by_file (const int &index) const` – запис у рядок інформації про об'єкт класу `CCountry` (метод класу `CMetod`).

`bool check_str(const std::string& str) const` – перевірка рядка на відповідність формату зберігання даних про об'єкт класу `CCountry` (метод класу `CMetod`).

`void print_all_with_2_or_more_words() const` – виведення усіх об'єктів класу `CCountry` в назві яких є 2 або більше слів з масиву в класі `CMetod` на екран(метод класу `CMetod`).

## 2.4 Опис функцій

`void menu()` – функція меню.

## 3 Текст програми

Лабораторная работа №4.cpp

```
#pragma once
// Лабораторная работа №4.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>
#include <regex>
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&);
    ~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)
{
    int i = 0;
    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";
}
CMethod.h
#include "CCountry.h"
class CMethod
{
private:
    CCountry* countries;
    CCountry* copy;
    int next_i = 0;
    int new_i = 1;
public:
    void add_el(const CCountry& Country);
    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);
    bool check_str(const std::string& str) const;
    void print_all_with_2_or_more_words() const;
};
CMethod.cpp
#include "CMethod.h"
void CMethod::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 CMethod::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 CMethod::del_all()
{
    if (next_i != 0)
    {
        delete[] countries;
        next_i = 0;
    }
}

void CMethod::get_to_Screen(const int& index) const
{
    std::cout << "Title " << "Number_of_cities " << "Population " << "Area " << "\n";
    std::cout << get_str_by_file(index) << "\n";
}

CCountry CMethod::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 CMethod::print_all() const
{
    for (int i = 0; i < next_i; i++)
    {
        get_to_Screen(i);
    }
}

void CMethod::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 CMethod::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();
}

```

```

        return ss.str();
    }
    void CMetod::find_to_str_by_file(const std::string str)
    {
        if (check_str(str))
        {
            std::regex reg("_.+_");
            std::smatch smat;
            std::regex_search(str, smat, reg);
            int i = str.find("_");
            i = str.find("_", i + 1);
            std::regex reg_temp("_");
            std::string temp = smat[0];
            std::string Title = std::regex_replace(temp, reg_temp, "");
            int i2 = str.find(" ", i + 2);
            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;
            CCountry secondcountry(Title, Number_of_cities, Population, Area);
            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();
    }
}

```

```

bool CMetod::check_str(const std::string& str) const
{
    std::regex reg("[A-Za-zA-Яa-я0-9\\!,\\?\\\"\\/;\\'\\']*");
    if (!(std::regex_search(str, reg)))
    {
        return false;
    }
    std::regex reg_2("\\s{2,}");
    if (std::regex_search(str, reg_2))
    {
        return false;
    }
    std::regex reg_3("[\\!\\?:\\.\\,\\;]{2,}");
    if (std::regex_search(str, reg_3))
    {
        return false;
    }
    std::regex reg_4("[\\'\\\"]{2,}");
    if (std::regex_search(str, reg_4))
    {
        return false;
    }
    std::regex reg_5("^\\[A-ZА-Я]");
    if (!std::regex_search(str, reg_5))
    {
        return false;
    }
    return true;
}

void CMetod::print_all_with_2_or_more_words() const
{
    for (int i = 0; i < next_i; i++)
    {
        std::string str;
        str = get_str_by_file(i);
        std::regex reg("_+ .+_");
        if (std::regex_search(str, reg))
        {
            std::cout << i + 1 << " " << str << "\n";
        }
    }
}

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("Страна1", 143, 45745656, 47342362);
    dir.add_el(firstcountry1);
    CCountry firstcountry2("Страна2", 156, 38567454, 68457458);
    dir.add_el(firstcountry2);
    CCountry firstcountry3("Страна3", 167, 46357625, 98686453);
    dir.add_el(firstcountry3);
    CCountry firstcountry4("Страна4", 179, 78567583, 68457458);
    dir.add_el(firstcountry4);
    while (n != 9)
    {
        std::cout << "-----МЕНЮ-----" << "\n";
        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 << "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.read_from_file("text.txt");
test_metod.print_all();
std::cout << "Если перед этим сообщением на экран вывелась информация о файлах, то методы
read_from_file и get_str_by_file работают правильно." << "\n";
test_metod.add_el(firstcountry6);
test_metod.write_to_file("text.txt");
test_metod.read_from_file("text.txt");
test_metod.print_all();
std::cout << "Если перед этим сообщением на экран вывелась информация о всех предыдущих
файлах и одном новом, то метод write_to_file работает правильно." << "\n";
test_metod.print_all_with_2_or_more_words();
std::cout << "Если перед этим сообщением на экран вывелась информация о двух файлах с
названиями из двух слов, то метод вывода информации о файлах с названиями из двух слов работает
верно. " << "\n";
test_metod.read_from_file("Test.txt");
test_metod.print_all();
std::cout << "Если перед этим сообщением на экран вывелась информация о тестовом файле и
больше ни о каких других, то проверка данных о файле в методе получения файла из строки работает
верно." << "\n";
if (_CrtDumpMemoryLeaks())
{
    std::cout << "Утечка памяти обнаружена." << "\n";
}
else
{
    std::cout << "Утечка памяти не обнаружена." << "\n";
}
int t;
std::cin >> t;
}

```

```

text.txt
_Страна1_ 143 45745656 47342362
_Страна2_ 156 38567454 68457458
_Страна3_ 167 46357625 98686453
_Страна4_ 179 78567583 68457458

```

```

Test.txt
_Стра__на1_ 143 45745656 47342362
_Страна2!!!_ 156 38567454 68457458
_СТР3_ 167 46357625 98686453
_абра дабра_ 179 78567583 68457458
_Тест_ 131 57290422 97850932

```

#### 4. Результати роботи програми

Результати роботи програми:



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

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