МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний аерокосмічний університет ім. М.Є. Жуковського

«Харківський авіаційний інститут»

Факультет радіоелектроніки, комп’ютерних систем та інфокомунікацій

Кафедра комп’ютерних систем, мереж і кібербезпеки (503)

Лабораторна робота № *7*

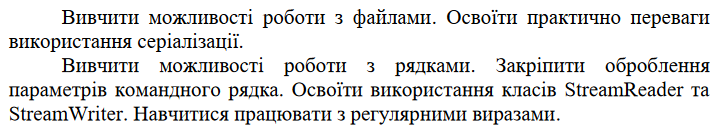
|  |  |
| --- | --- |
|  | *Робота з файлами та серіалізація. Робота з текстом та регулярними виразами* |
|  | (назва лабораторної роботи) |
| з дисципліни | *Технології програмування* |
|  | (шифр)  ХАІ**.**503**.**525a**.**03О**.**123-Комп'ютерна інженерія**,** ПЗ №9629619 |

|  |  |  |
| --- | --- | --- |
| Виконав студент гр. | 525а | *Литвиненко А.В.* |
| 11.12.2022 | (№ групи) | (П.І.Б.) |
| (підпис, дата) |  |  |
| Перевірив | канд. техн. наук, доцент | |
|  |  | *Є. В. Бабешенко* |
| (підпис, дата) |  | (П.І.Б.) |

Харків – 2022

**Тема роботи:** робота з файлами та серіалізація. Робота з текстом та регулярними виразами.

**Мета роботи**:

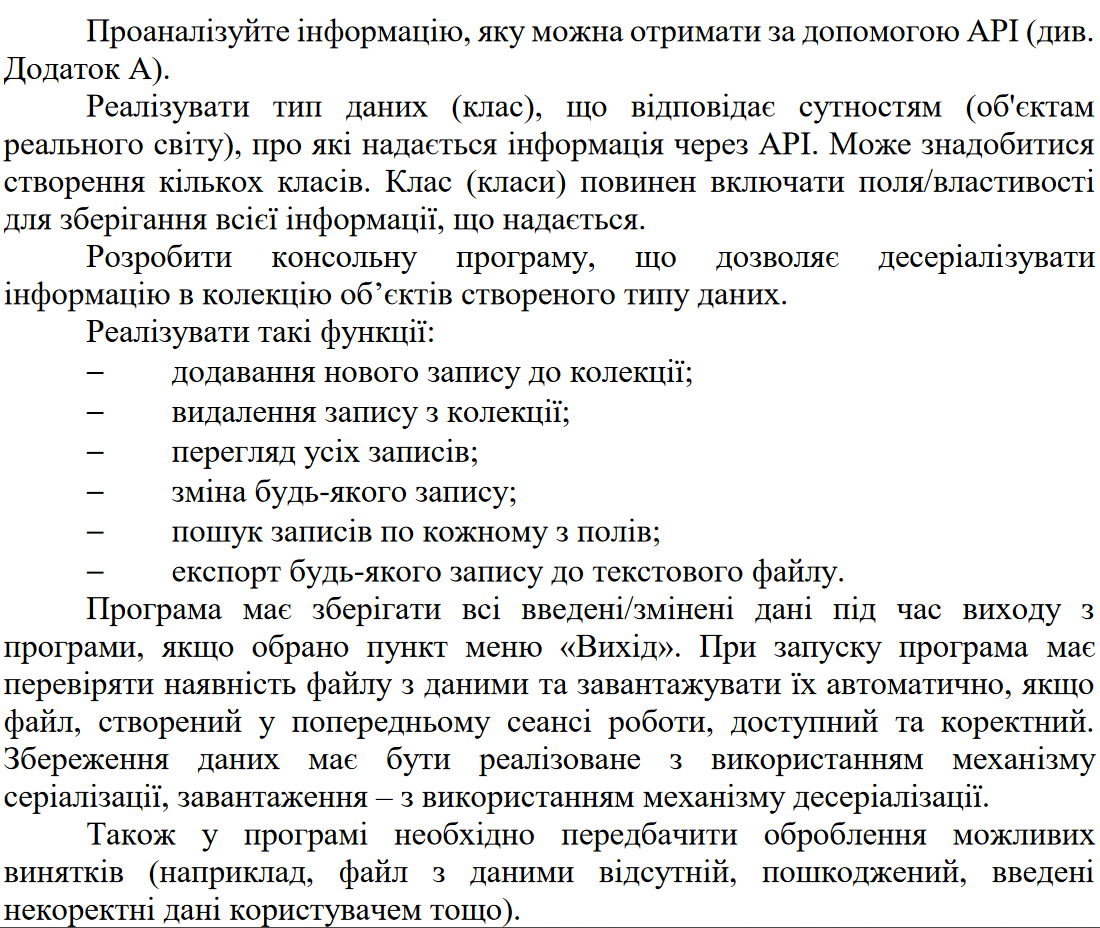


**Варіант 5**

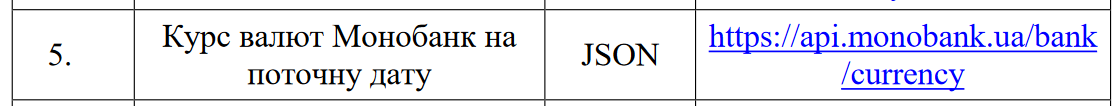
**Задача 1**

**Частина 1**. Постановка завдання

**Умова:**



**Умова з додатка:**



<https://api.monobank.ua/bank/currency> - лінк на файл

**Частина 2**. Схема класів

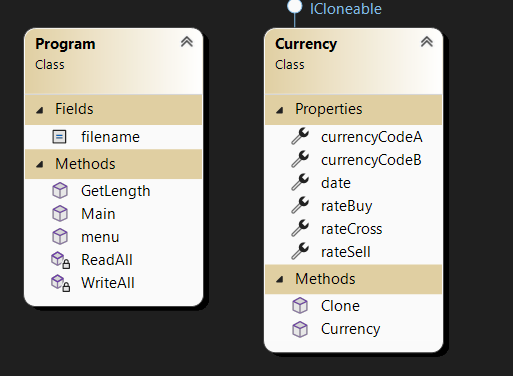


Рисунок 1 - Схема класів

**Частина 3**. Текст програми

Відповідно до розробленого алгоритму в середовищі Microsoft Visual Studio була написана програма, яка наведена нижче.

**Program.cs**

using Newtonsoft.Json;

using System.IO.Enumeration;

using System.Net.Http.Json;

using System.Reflection.Metadata.Ecma335;

using System.Runtime.Serialization.Json;

namespace Task\_1

{

internal class Program

{

public const string filename = "currency.json";

static void WriteAll(List<Currency> instances, string file)

{

string serializedUsers = JsonConvert.SerializeObject(instances);

File.WriteAllText(file, serializedUsers);

}

static List<Currency> ReadAll()

{

string json = File.ReadAllText(filename);

List<Currency> instances = JsonConvert.DeserializeObject<List<Currency>>(json)!;

return instances;

}

static public int GetLength(Currency[] arr)

{

int i = 0;

while (arr[i] != null)

{

i++;

}

return i;

}

static public void menu()

{

Console.WriteLine("\t========== Menu ==========");

Console.WriteLine("\t0. This menu"); // +

Console.WriteLine("\t1. Show content"); // +

Console.WriteLine("\t2. Add record"); // +

Console.WriteLine("\t3. Delete record"); // +

Console.WriteLine("\t4. Change record"); // +

Console.WriteLine("\t5. Find record"); // +

Console.WriteLine("\t6. Export to another file");

Console.WriteLine("\t7. Save and exit");

Console.WriteLine("\t8. Exit (without saving)"); // +

Console.WriteLine("\t==========================");

}

public static void Main(string[] args)

{

List<Currency> array = ReadAll();

// Infinity cycle or MENU

int op = 0;

int i = 0;

char yn;

int user\_value;

int field\_id = 0;

string value = "";

Currency t = new Currency(0,0,0,0,0,0);

while (true)

{

switch (op)

{

// Prints menu

case 0:

menu();

break;

// Show structured file content

case 1:

i = 0;

foreach(Currency c in array)

{

Console.WriteLine(String.Format("==========={0}===========", i));

Console.WriteLine("CurrencyCodeA = " + c.currencyCodeA.ToString());

Console.WriteLine("CurrencyCodeB = " + c.currencyCodeB.ToString());

Console.WriteLine("Date = " + c.date.ToString());

if (c.rateCross == 0)

{

Console.WriteLine("RateBuy = " + c.rateBuy.ToString());

Console.WriteLine("RateSell = " + c.rateSell.ToString());

}

else

{

Console.WriteLine("RateCross = " + c.rateCross.ToString());

}

Console.WriteLine("============================\n");

i++;

}

Console.WriteLine("Totally: " + i.ToString());

break;

// Add record to array

case 2:

Console.WriteLine("\t[Add a record]");

Console.WriteLine("If you want to skip a field, enter: 0");

Console.Write("currencyCodeA = ");

t.currencyCodeA = int.Parse(Console.ReadLine());

Console.Write("currencyCodeB = ");

t.currencyCodeB = int.Parse(Console.ReadLine());

Console.Write("date = ");

t.date = int.Parse(Console.ReadLine());

Console.Write("rateBuy = ");

t.rateBuy = double.Parse(Console.ReadLine());

Console.Write("rateSell = ");

t.rateSell = double.Parse(Console.ReadLine());

Console.Write("rateCross = ");

t.rateCross = double.Parse(Console.ReadLine());

array.Add((Currency) t.Clone());

Console.WriteLine("[OK]");

break;

// Remove record in array

case 3:

Console.Write("[Enter a record index] >> ");

user\_value = int.Parse(Console.ReadLine());

// numbers = numbers.Where((val, idx) => idx != numIndex).ToArray();

array.Remove(array[user\_value]);

Console.WriteLine("[OK]");

break;

// Change record in array

case 4:

Console.Write("[Enter a record index] >> ");

user\_value = int.Parse(Console.ReadLine());

t = array[user\_value];

// Print object information

Console.WriteLine("===== [ INFO ] =====");

Console.WriteLine("1. CurrencyCodeA = " + t.currencyCodeA.ToString());

Console.WriteLine("2. CurrencyCodeB = " + t.currencyCodeB.ToString());

Console.WriteLine("3. Date = " + t.date.ToString());

if (t.rateCross == 0)

{

Console.WriteLine("4. RateBuy = " + t.rateBuy.ToString());

Console.WriteLine("5. RateSell = " + t.rateSell.ToString());

}

else

{

Console.WriteLine("4. RateCross = " + t.rateCross.ToString());

}

Console.WriteLine("=====================\n");

Console.Write("[Field's number] >> ");

field\_id = int.Parse(Console.ReadLine());

Console.Write("[Field's value] >> ");

value = Console.ReadLine();

switch (field\_id)

{

case 1:

t.currencyCodeA = int.Parse(value);

break;

case 2:

t.currencyCodeB = int.Parse(value);

break;

case 3:

t.date = int.Parse(value);

break;

case 4:

if (t.rateBuy == 0)

{

t.rateCross = double.Parse(value);

}

else

{

t.rateBuy = double.Parse(value);

}

break;

case 5:

if (t.rateSell == 0)

{

break;

}

else

{

t.rateSell = double.Parse(value);

}

break;

default:

Console.WriteLine("[ERR] This field doesn't exist!");

break;

}

Console.WriteLine("[OK]");

break;

// Find record in array

case 5:

Console.WriteLine("Fields to search:");

Console.WriteLine("\t1. currencyCodeA");

Console.WriteLine("\t2. currencyCodeB");

Console.WriteLine("\t3. date");

Console.WriteLine("\t4. rateBuy");

Console.WriteLine("\t5. rateSell");

Console.WriteLine("\t6. rateCross");

Console.Write("[Field's number] >> ");

field\_id = int.Parse(Console.ReadLine());

Console.Write("[Value to find] >> ");

value = Console.ReadLine();

bool flag = true;

i = 0;

foreach(Currency c in array)

{

if (!flag)

{

break;

}

switch (field\_id)

{

// currencyCodeA

case 1:

if (c.currencyCodeA == int.Parse(value))

{

flag = false;

}

break;

// currencyCodeB

case 2:

if (c.currencyCodeB == int.Parse(value))

{

flag = false;

}

break;

// date

case 3:

if (c.date == int.Parse(value))

{

flag = false;

}

break;

// rateBuy

case 4:

if (c.rateBuy == double.Parse(value))

{

flag = false;

}

break;

// rateSell

case 5:

if (c.rateSell == double.Parse(value))

{

flag = false;

}

break;

// rateCross

case 6:

if (c.rateCross == double.Parse(value))

{

flag = false;

}

break;

default:

Console.WriteLine("[ERR] Invalid field's number!");

break;

}

if (!flag)

{

Console.WriteLine();

Console.WriteLine(">>> [Found!] <<<<");

Console.WriteLine("Index: " + i.ToString());

Console.WriteLine("CurrencyCodeA = " + t.currencyCodeA.ToString());

Console.WriteLine("CurrencyCodeB = " + t.currencyCodeB.ToString());

Console.WriteLine("Date = " + t.date.ToString());

if (t.rateCross == 0)

{

Console.WriteLine("RateBuy = " + t.rateBuy.ToString());

Console.WriteLine("RateSell = " + t.rateSell.ToString());

}

else

{

Console.WriteLine("RateCross = " + t.rateCross.ToString());

}

Console.WriteLine("\n");

}

i++;

}

if (flag)

{

Console.WriteLine("[ERR] 404 not found\n");

}

break;

// Export file content to another file

case 6:

Console.Write("Enteer filename to export: ");

value = Console.ReadLine();

WriteAll(array,value);

Console.WriteLine("OK");

break;

// Save data and exit

case 7:

// saving user data to currency.json

WriteAll(array, filename);

Console.WriteLine("Ok");

return;

// Just exit without saving the file

case 8:

Console.Write("[INFO] You may lost of your data. Are you sure? (y/N): ");

yn = Console.ReadLine()[0];

if (yn == 'y')

{

return;

}

else

{

Console.WriteLine("[INFO] Your data was saved from destruction! :)");

}

break;

default:

Console.WriteLine("[INFO] Invalid operation!");

break;

}

Console.Write("[OP] >> ");

try

{

op = int.Parse(Console.ReadLine());

}

catch (FormatException)

{

Console.WriteLine("[ERR] Invalid format!");

op = 0;

}

}

return;

}

}

}

**Class1.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Runtime.Serialization.Json;

using System.Runtime.Serialization;

using System.Text.Json.Serialization;

using System.ComponentModel;

namespace Task\_1

{

class Currency : ICloneable

{

public int currencyCodeA { get; set; }

public int currencyCodeB { get; set; }

public int date { get; set; }

public double rateBuy { get; set; }

public double rateSell { get; set; }

public double rateCross { get; set; }

public Currency(int a, int b, int d, double buy, double sell, double cross)

{

currencyCodeA = a;

currencyCodeB = b;

date = d;

rateBuy = buy;

rateSell = sell;

rateCross = cross;

}

public object Clone()

{

return this.MemberwiseClone();

}

}

}

**Частина 4**. Тестування

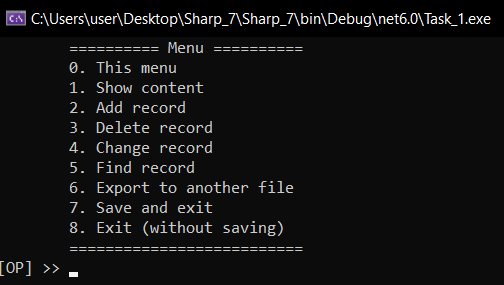


Рисунок 2 – Меню програми

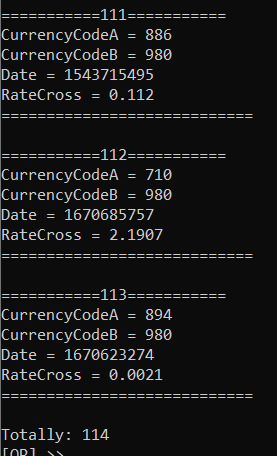


Рисунок 3 – Показ усіх значень

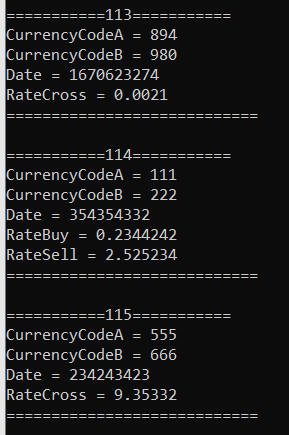
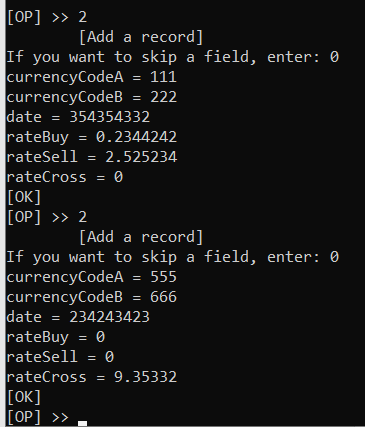


Рисунок 4 – Додавання значень та перевірка

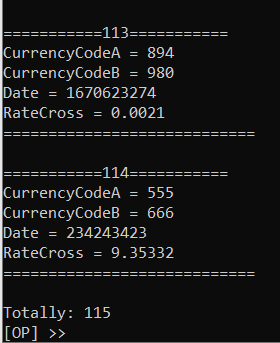
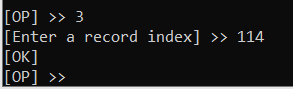


Рисунок 5 – Видалення значення з індексом 114

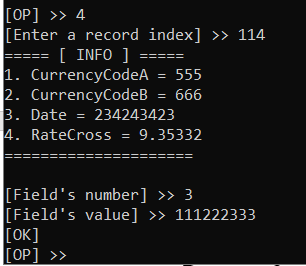


Рисунок 6 – Зміна значення date елемента з індексом 114

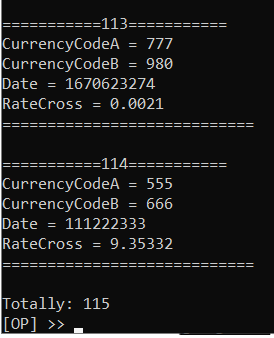
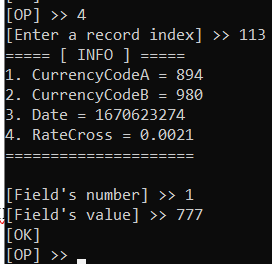


Рисунок 7 – Зміна значення currencyCodeA елемента з індексом 113

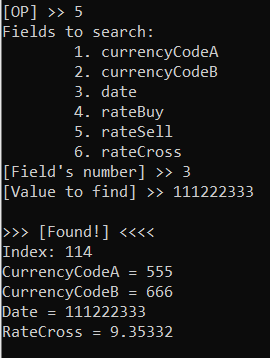
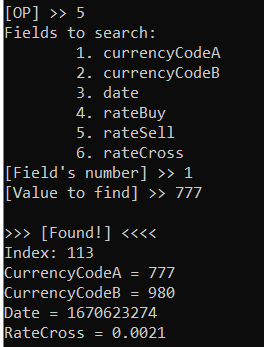


Рисунок 8 – Пошук за значенням

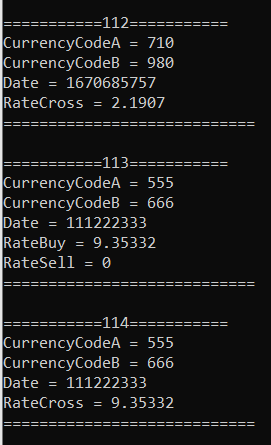


Рисунок 9 – Як виглядають останні елементи після змін

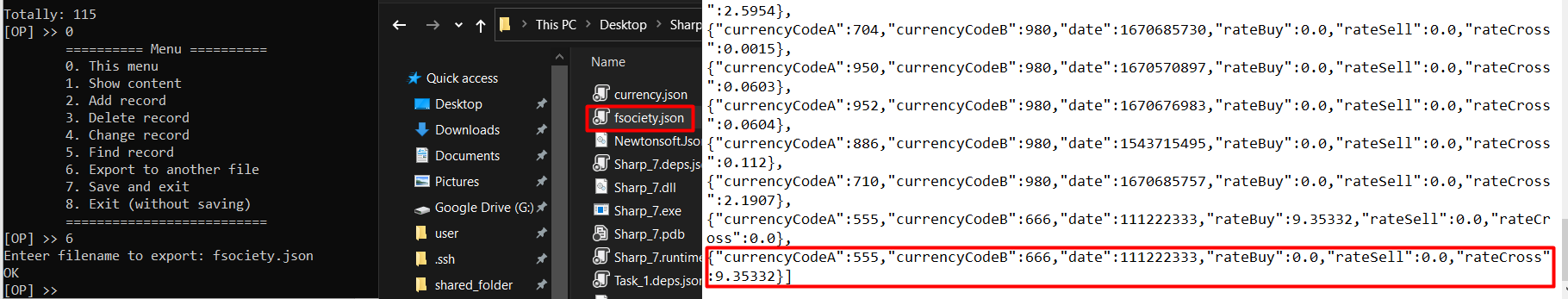


Рисунок 10 – Експорт в інший файл

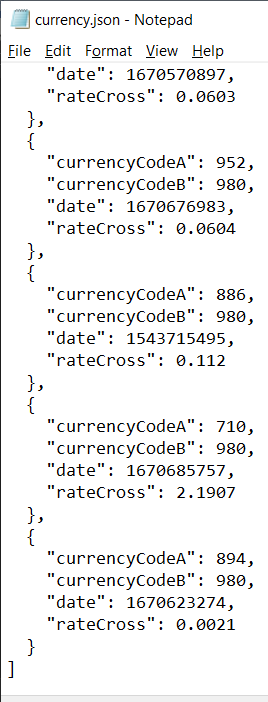


Рисунок 11 – Файл до збереження

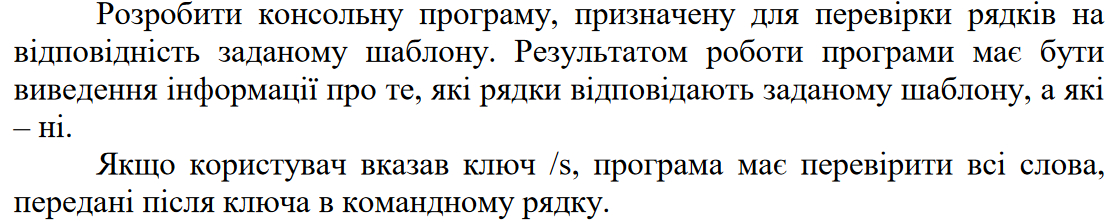


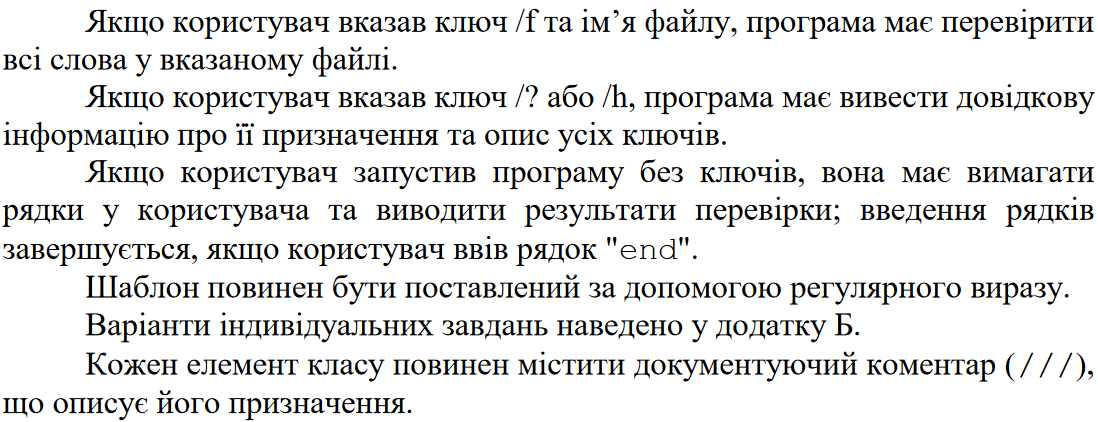
Рисунок 12 – Файл після збереження

**Задача 2**

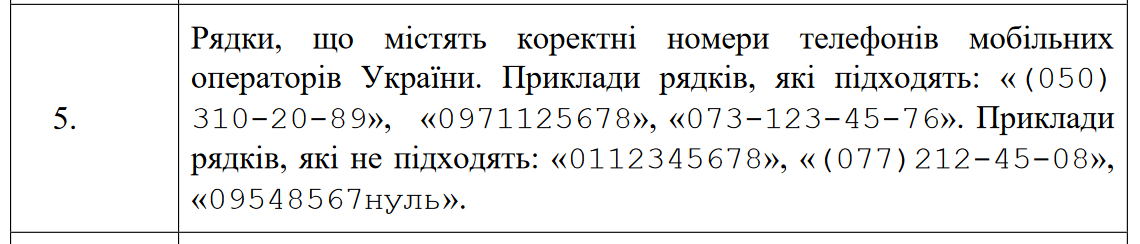
**Частина 1**. Постановка завдання

**Умова:**





**Умова з додатка:**



**Частина 2**. Схема класів

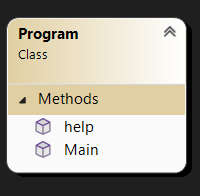


Рисунок 13 - Схема класів

**Частина 3**. Розробка тестів

Таблиця 3 – Тестові набори

|  |  |  |
| --- | --- | --- |
| № | Вхідні дані | Очікуваний результат |
| 1 | (050) 310-20-89 | OK |
| 2 | 0971125678 | OK |
| 3 | 073-123-45-76 | OK |
| 4 | 0112345678 | BAD |
| 5 | (077)212-45-08 | BAD |
| 6 | 09548567нуль | BAD |

**Частина 4**. Текст програми

**Program.cs**

using System.Text.RegularExpressions;

namespace Task\_2

{

internal class Program

{

static public void help()

{

Console.WriteLine("Task 2 made by Andrii Lytvynenko (C) 2022" +

"\n\t - /s -- search in a list of words" +

"\n\t - /f -- search in a list of files" +

"\n\t - /? /h -- this help message");

}

public static void Main(string[] args)

{

string pattern = @"(\(050\)\d{3}-\d{2}-\d{2})|(097\d{7})|(073-\d{3}-\d{2}-\d{2})";

string argument;

string tmp = "";

try

{

argument = args[0];

}

catch (IndexOutOfRangeException)

{

Console.WriteLine("[ERR] No command and values!");

return;

}

List<string> values = args.ToList();

values = values.GetRange(1, values.Count-1);

if(values.Count == 0 && !(argument == "/?" || argument == "/h"))

{

// user must enter values

Console.WriteLine("Enter values. If you want to exit, you'll enter 'end' key :)");

while(true)

{

tmp = Console.ReadLine()!;

if(tmp == "end")

{

break;

}

values.Add(tmp);

}

}

Console.WriteLine("\n======= Regex check begins! =======\n");

switch (argument)

{

case "/s":

foreach(string word in values)

{

if(Regex.IsMatch(word, pattern))

{

Console.WriteLine(String.Format("'{0}' is good! :)", word));

}

else

{

Console.WriteLine(String.Format("'{0}' is BAAAAAAD! >:(", word));

}

}

break;

case "/f":

foreach(string file in values)

{

try

{

string[] text = File.ReadAllLines(file);

foreach (string word in text)

{

if (Regex.IsMatch(word, pattern))

{

Console.WriteLine(String.Format("'{0}' is good! :)", word));

}

else

{

Console.WriteLine(String.Format("'{0}' is BAAAAAAD! >:(", word));

}

}

}

catch (FileNotFoundException)

{

Console.WriteLine(String.Format("[ERR] '{0}' not found!", file));

}

catch (ArgumentException)

{

Console.WriteLine("[ERR] Empty filename!");

}

}

break;

case "/?":

case "/h":

help();

break;

}

return;

}

}

}

**Частина 5**. Тестування

Результати тестування наведені в таблиці 4.

Таблиця 4 – Результати тестування

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Номер тесту | Вхідні дані | Очікуваний результат | Отриманий результат | Результат тестування |
| 1 | (050) 310-20-89 | OK | OK | OK |
| 2 | 0971125678 | OK | OK | OK |
| 3 | 073-123-45-76 | OK | OK | OK |
| 4 | 0112345678 | BAD | BAD | OK |
| 5 | (077)212-45-08 | BAD | BAD | OK |
| 6 | 09548567нуль | BAD | BAD | OK |

**Скриншот тестування:**

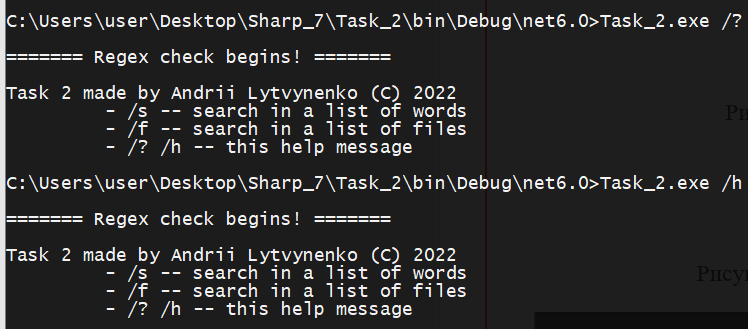


Рисунок 14 – Довідка

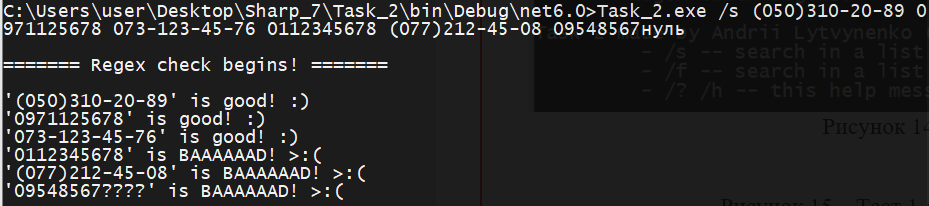


Рисунок 15 – Тест 1-6 з командного рядка

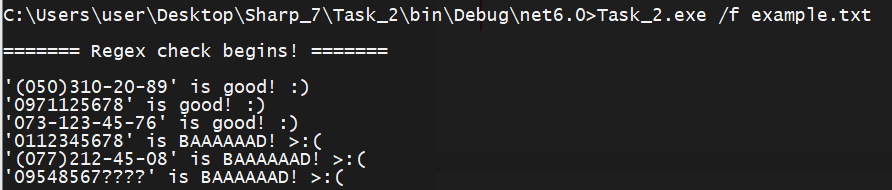


Рисунок 16 – Тест 1-6 з файлу

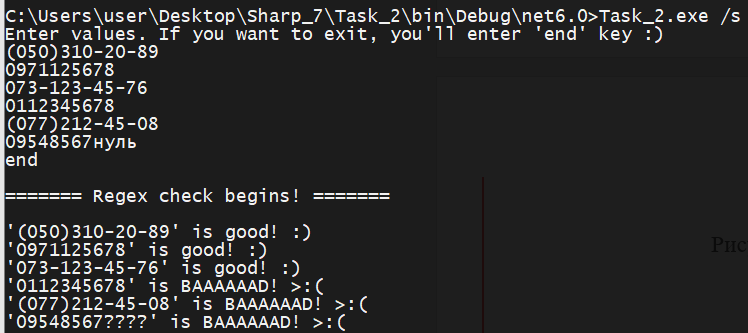


Рисунок 17 – Тест 1-6 з введення у програмі

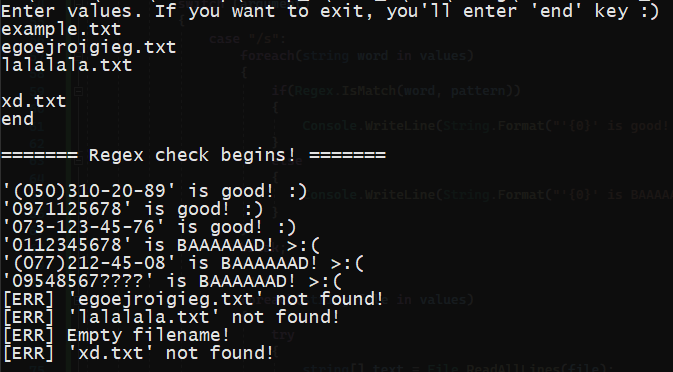


Рисунок 18 – Тест 1-6 з декількох файлів файлу

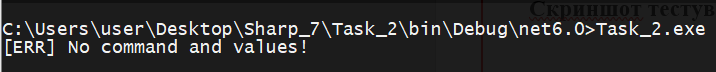


Рисунок 19 – Перехоплення помилки

**Висновки**

Під час цієї лабораторної роботи я навчився користуватися файлами, зумів їх відкривати, читати, перезаписувати та закривати. Також закріпив практичні переваги використання серіалізації. У другій частині я покращив свої знання з використання рядків, закріпив оброблення параметрів командного рядка і навчився користуватися регулярними виразами.