**ЛАБОРАТОРНА РОБОТА № 12**

**Робота з файлами у мові C#.**

***Мета:*** навчитися застосовувати бібліотеки мови C# для вико-нання операцій над текстовими та двійковими файлами; навчитися використо-вувати серіалізацію та десеріалізацію об’єктів.

**Хід роботи:**

**Завдання:**

1. Створити рішення з назвою «oop-lab12».

2. У рішенні має бути два проекти:

- віконний додаток;

- бібліотека класів.

3. У віконному додатку реалізувати програму, яка дозволить:

- вибирати користувачу довільні файли для подальшої їх обробки;

- шифруватиме вибрані файли;

- архівуватиме вибрані файли;

- запаковуватиме вибрані файли;

- розпаковуватиме вибрані файли.

Обов’язкові вимоги до інтерфейсу програми:

- можливість вибору файлів за допомогою діалогу OpenFileDialog;

- відображення списку вибраних файлів у формі;

- можливість вибору файлів шляхом Drag&Drop (перетягування файлів та папок у вікно програми);

- можливість вибору дії:

- шифрування;

- розшифрування;

- архівування;

- розархівування.

- меню для доступу до можливостей програми, створене за допомогою компонента MenuStrip.

4. Закомітити рішення у репозиторій «oop-lab12» і надати доступ викла-дачам.

Лістинг програми (Form):

using System;

using System.Windows.Forms;

using System.IO;

using System.Text.RegularExpressions;

using ClassLibrary;

namespace WindowsFormsApp

{

public partial class Form1 : Form

{

Regex fileRegex = new Regex(@"^.+\..+$");

public Form1()

{

InitializeComponent();

}

private string GetAllFiles(string[] path)

{

string result = null;

foreach (var file in path)

{

if (fileRegex.IsMatch(file))

{

result += file + Environment.NewLine;

}

else

{

foreach (FileInfo item in new DirectoryInfo(file).GetFiles("\*", SearchOption.AllDirectories))

{

result += item.FullName + Environment.NewLine;

}

}

}

return result;

}

private void textBoxListFiles\_DragDrop(object sender, DragEventArgs e)

{

string[] files = (string[])e.Data.GetData(DataFormats.FileDrop);

textBoxListFiles.Text = GetAllFiles(files);

}

private void textBoxListFiles\_DragEnter(object sender, DragEventArgs e)

{

if (e.Data.GetDataPresent(DataFormats.FileDrop, false) == true)

{

e.Effect = DragDropEffects.All;

}

}

private string[] getFilePath()

{

return textBoxListFiles.Text.Split(new char[] { '\n' }, StringSplitOptions.RemoveEmptyEntries);

}

private void toolStripMenuItemEncrypt\_Click(object sender, EventArgs e)

{

FileManager fileManager = new FileManager(getFilePath());

fileManager.EncryptFile();

}

private void toolStripMenuItemDecrypt\_Click(object sender, EventArgs e)

{

FileManager fileManager = new FileManager(getFilePath());

fileManager.DecryptFile();

}

private void toolStripMenuItemArchive\_Click(object sender, EventArgs e)

{

FileManager fileManager = new FileManager(getFilePath());

fileManager.CompressFile();

}

private void toolStripMenuItemUnarchive\_Click(object sender, EventArgs e)

{

FileManager fileManager = new FileManager(getFilePath());

fileManager.DecompressFile();

}

private void buttonChooseFiles\_Click(object sender, EventArgs e)

{

using (OpenFileDialog fileDialog = new OpenFileDialog())

{

fileDialog.InitialDirectory = "c:\\";

fileDialog.Filter = "txt files (\*.txt\*)|\*.txt|All files (\*.\*)|\*.\*";

fileDialog.FilterIndex = 2;

if (fileDialog.ShowDialog() == DialogResult.OK)

{

textBoxListFiles.Text += fileDialog.FileName + Environment.NewLine;

}

}

}

private void buttonClearFiles\_Click(object sender, EventArgs e)

{

textBoxListFiles.Text = null;

}

}

}

Лістинг бібліотеки ClassLibrary

FileManager

using System;

using System.Text.RegularExpressions;

using System.Security.Cryptography;

using System.IO;

using System.IO.Compression;

namespace ClassLibrary

{

public class FileManager

{

public FileInfo[] Files { get; set; }

public FileManager(FileInfo[] files)

{

Files = files;

}

public FileManager(string[] Path)

{

for (int i = 0; i < Path.Length; i++)

Path[i] = Regex.Replace(Path[i], @"\r", @"");

Files = new FileInfo[Path.Length];

for (int i = 0; i < Path.Length; i++)

{

if (Path[i] != null)

Files[i] = new FileInfo(Path[i]);

}

}

public void EncryptFile(string folderDestination = @"C:\Users\laptop\Desktop\Files\Зашифрованные файлы")

{

folderDestination = createFolder(folderDestination);

try

{

foreach (var fileSource in Files)

{

using (FileStream originalFile = fileSource.OpenRead())

{

string fileDestination = folderDestination + @"\" + fileSource.Name + ".crypto";

using (FileStream newFile = File.Create(fileDestination))

{

DESCryptoServiceProvider rmCrypto = new DESCryptoServiceProvider();

using (CryptoStream cryptStream = new CryptoStream(newFile, rmCrypto.CreateEncryptor(), CryptoStreamMode.Write))

{

originalFile.CopyTo(cryptStream);

}

}

}

}

}

catch (Exception e)

{

Console.WriteLine(e);

}

}

public void DecryptFile(string folderDestination = @"C:\Users\laptop\Desktop\Files\Расшифрованные файлы")

{

folderDestination = createFolder(folderDestination);

try

{

foreach (var fileSource in Files)

{

if (Regex.IsMatch(fileSource.Name, @"^.+\.crypto$"))

{

using (FileStream cryptedFile = fileSource.OpenRead())

{

string fileDestination = folderDestination + @"\" + fileSource.Name;

fileDestination = Regex.Replace(fileDestination, @"(.+)\.crypto", "$1");

using (FileStream newFile = File.Create(fileDestination))

{

DESCryptoServiceProvider rmCrypto = new DESCryptoServiceProvider();

using (CryptoStream cryptStream = new CryptoStream(cryptedFile, rmCrypto.CreateDecryptor(), CryptoStreamMode.Read))

{

byte[] data = new byte[1000];

cryptStream.Read(data, 0, data.Length);

Console.WriteLine(data);

cryptStream.CopyTo(newFile);

}

}

}

}

}

}

catch (Exception e)

{

Console.WriteLine(e);

}

}

private string createFolder(string folderDestination)

{

int i = 1;

do

{

if (!Directory.Exists(folderDestination))

break;

else if (i == 1)

{

folderDestination += "(1)";

i++;

}

else

{

folderDestination = Regex.Replace(folderDestination, @"^(.+)\(\d+\)$", $"$1({i++})");

Console.WriteLine(folderDestination);

}

} while (true);

Directory.CreateDirectory(folderDestination);

return folderDestination;

}

public void CompressFile(string folderDestination = @"C:\Users\laptop\Desktop\Files\Сжатые файлы")

{

folderDestination = createFolder(folderDestination);

try

{

foreach (var fileSource in Files)

{

using (FileStream originalFile = fileSource.OpenRead())

{

string fileDestination = folderDestination + @"\" + fileSource.Name + ".gzar";

using (FileStream newFile = File.Create(fileDestination))

{

using (GZipStream compressStream = new GZipStream(newFile, CompressionMode.Compress))

{

originalFile.CopyTo(compressStream);

}

}

}

}

}

catch (Exception e)

{

Console.WriteLine(e);

}

}

public void DecompressFile(string folderDestination = @"C:\Users\laptop\Desktop\Files\Распакованные файлы")

{

folderDestination = createFolder(folderDestination);

try

{

foreach (var fileSource in Files)

{

if (Regex.IsMatch(fileSource.Name, @"^.+\.gzar$"))

{

using (FileStream compressedFile = fileSource.OpenRead())

{

string fileDestination = folderDestination + @"\" + fileSource.Name;

fileDestination = Regex.Replace(fileDestination, @"(.+)\.gzar", "$1");

using (FileStream newFile = File.Create(fileDestination))

{

using (GZipStream decompressionStream = new GZipStream(compressedFile, CompressionMode.Decompress))

{

decompressionStream.CopyTo(newFile);

}

}

}

}

}

}

catch (Exception e)

{

Console.WriteLine(e);

}

}

}

}

Результат виконання програми:

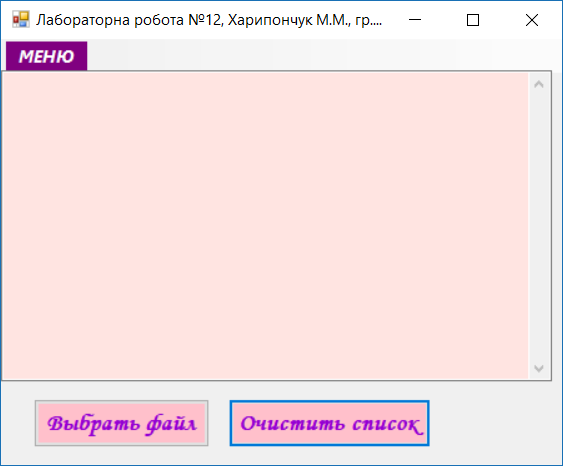


Рис. 1. Вигляд програми при відкритті

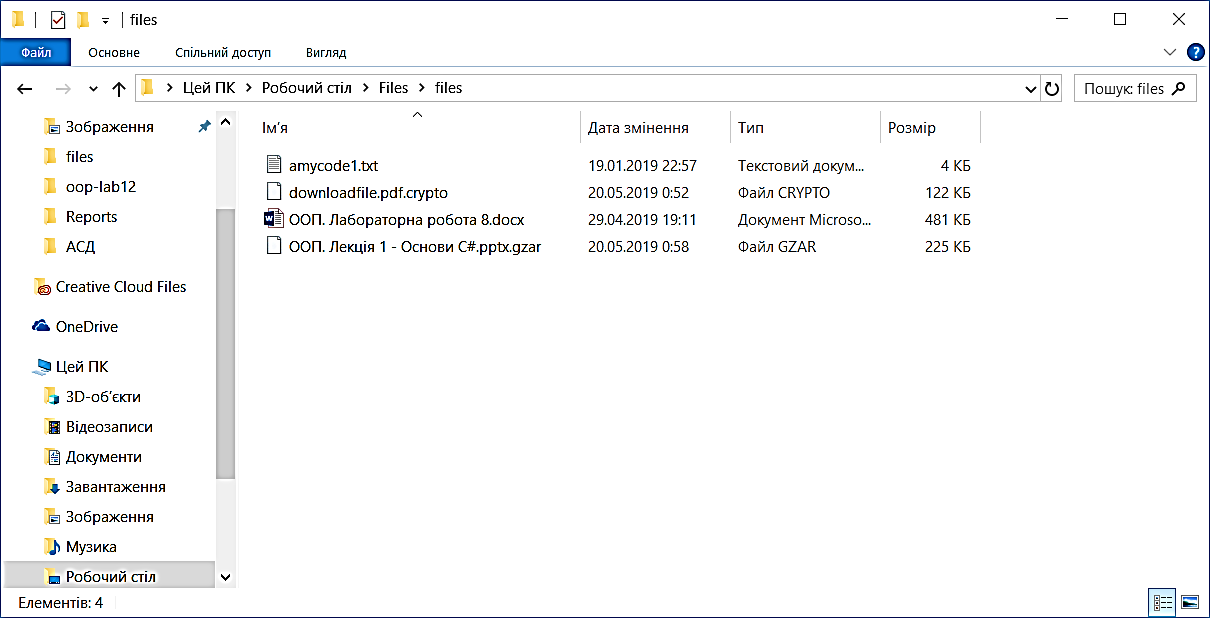


Рис. 2. Файли, які обираються

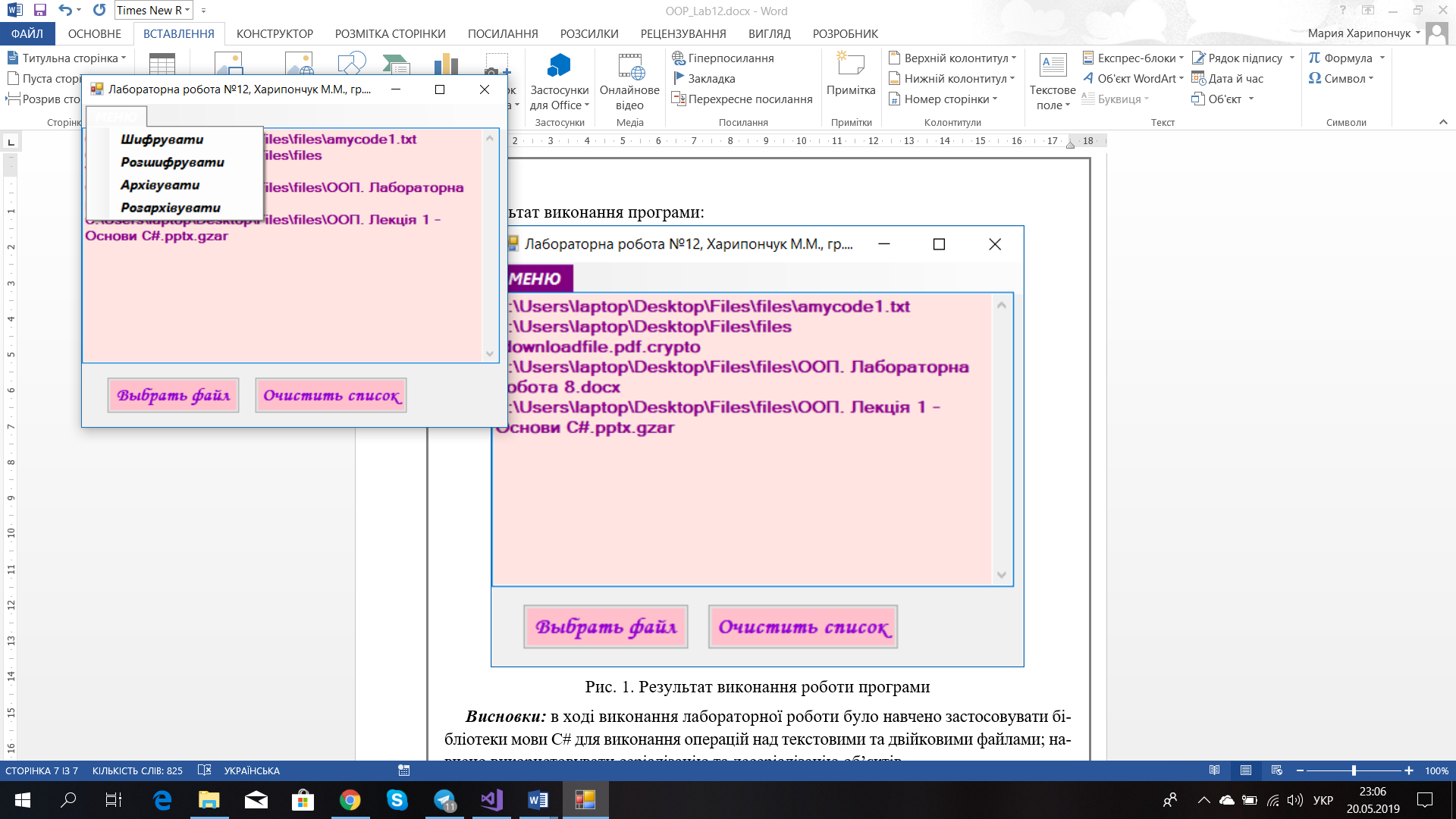


Рис. 3. Кнопка МЕНЮ на панелі програми

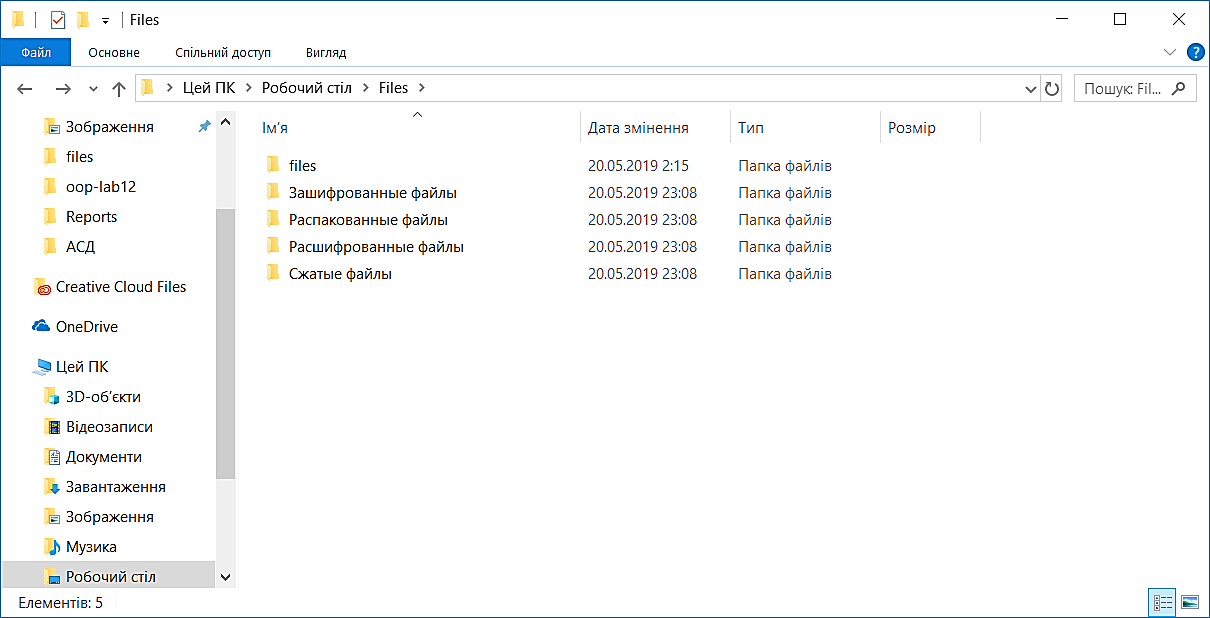


Рис. 4. Результат виконання роботи програми

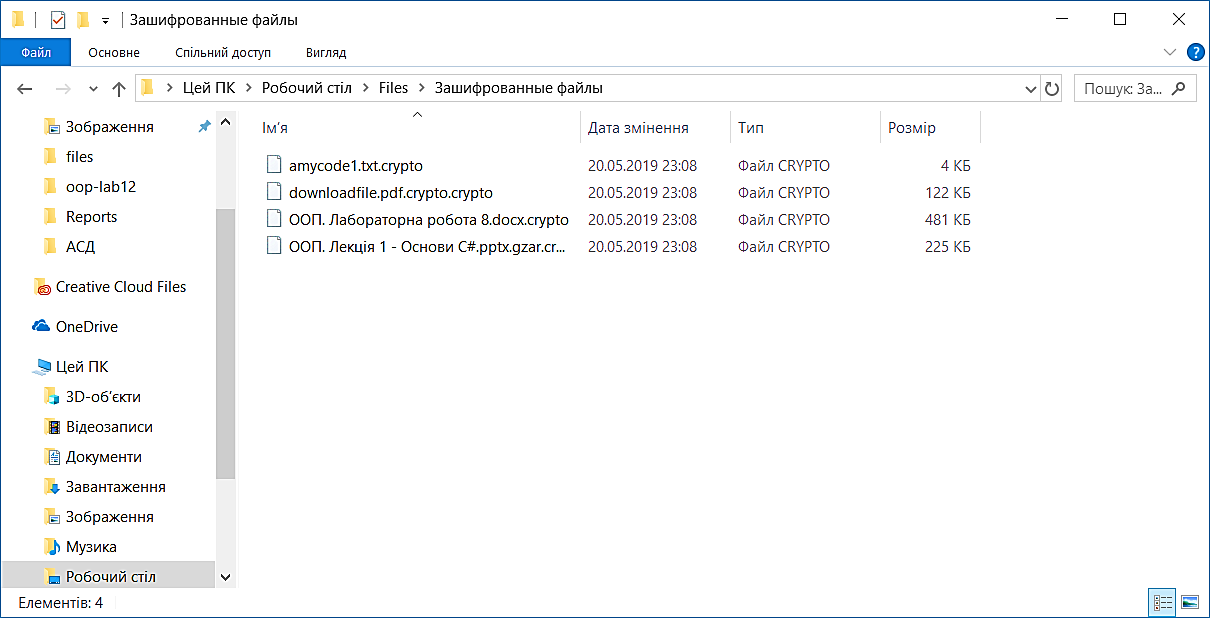


Рис. 5. Результат шифрування

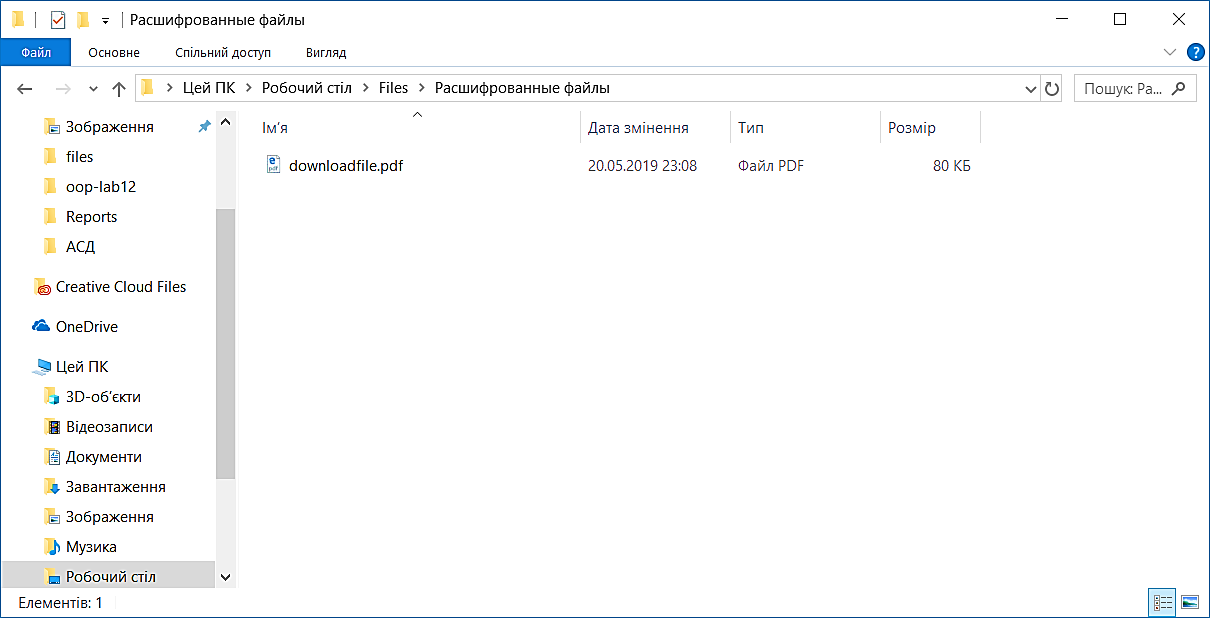


Рис. 6. Результат розшифрування

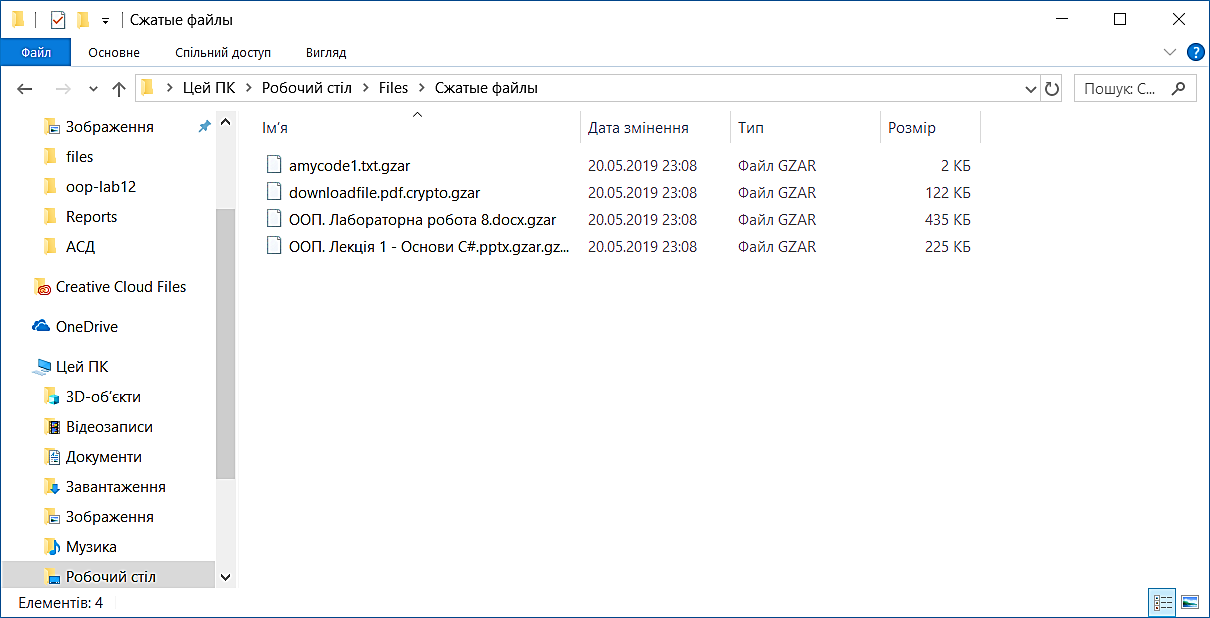


Рис. 7. Результат архівування

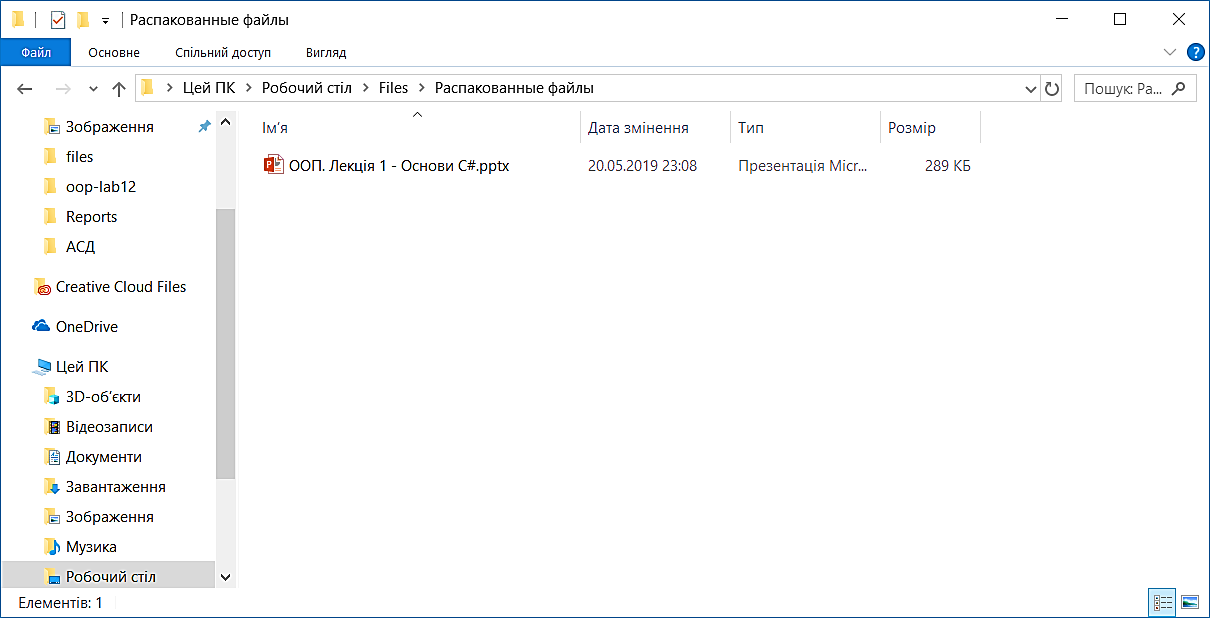


Рис. 8. Результат розархівування

***Висновки:*** в ході виконання лабораторної роботи було навчено застосовувати бібліотеки мови C# для виконання операцій над текстовими та двійковими файлами; навчено використовувати серіалізацію та десеріалізацію об’єктів.