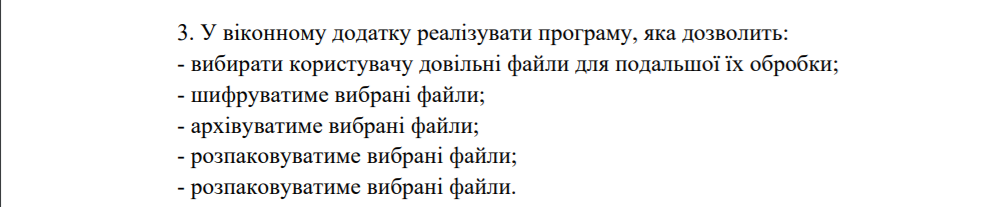
**ЛАБОРАТОРНА РОБОТА № 14**

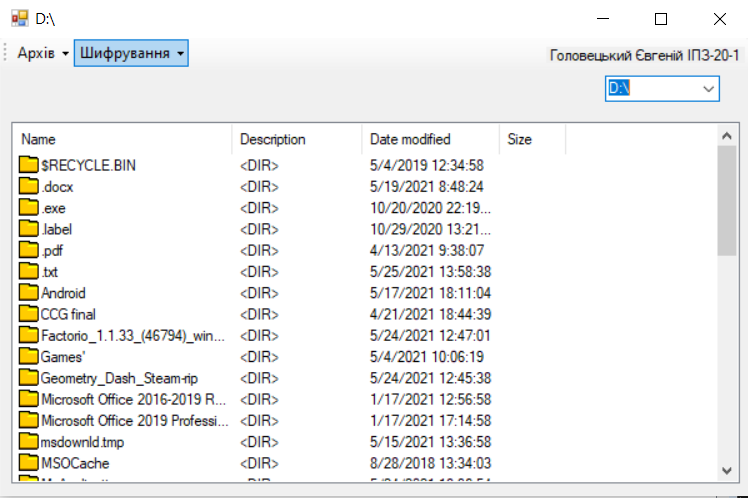
**Тема:** **Робота з файлами у мові C#.**

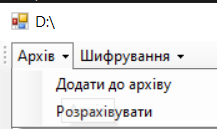
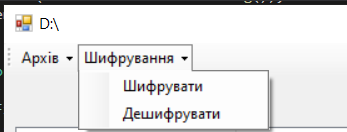
***Мета:*** навчитися застосовувати бібліотеки мови C# для виконання операцій над текстовими та двійковими файлами; навчитися використовувати серіалізацію та десеріалізацію об’єктів.

**Хід роботи:**

***Завдання 1.***



***Результат виконання:***

******

*** ***

***Form1.cs***

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.ComponentModel;  using System.Data;  using System.Diagnostics;  using System.Drawing;  using System.IO;  using System.IO.Compression;  using System.Linq;  using System.Text;  using System.Text.RegularExpressions;  using System.Threading.Tasks;  using System.Windows.Forms;  using WindowsFormsApp1.Additional;  namespace WindowsFormsApp1  {  public partial class Form1 : Form  {  ImageList imageList1;  DirectoryInfo currenDirectory;  Crypto crypt = new Crypto("a");  WorkZip workZip = new WorkZip();  public Form1()  {  InitializeComponent();  imageList1 = new ImageList();  listView1.SmallImageList = imageList1;  }  private void Form1\_Load(object sender, EventArgs e)  {  Icon dirIcon = new Icon("../../Assets/folder.ico");  imageList1.Images.Add("dir", dirIcon);  foreach (DriveInfo drive in DriveInfo.GetDrives())  {  comboBox1.Items.Add(drive.Name);  }  }  private void listView1\_MouseDoubleClick(object sender, MouseEventArgs e)  {  string currDriver = comboBox1.SelectedItem.ToString();  ListViewItem item = listView1.SelectedItems[0];  if (new DirectoryInfo(currenDirectory.FullName + "/" + item.Text).Exists)  {  listView1.Items.Clear();  currenDirectory = new DirectoryInfo(currenDirectory.FullName + "/" + item.Text);  this.Text = currenDirectory.FullName;  GetFilesAndDirs(currenDirectory);  }  else  {  currenDirectory = new DirectoryInfo(currenDirectory.FullName);  }  }  private void comboBox1\_SelectedValueChanged(object sender, EventArgs e)  {  listView1.Items.Clear();  string currDriver = comboBox1.SelectedItem.ToString();  currenDirectory = new DirectoryInfo(currDriver);  this.Text = currenDirectory.FullName;  DriveInfo drive = new DriveInfo(currDriver);  ListViewItem item = new ListViewItem();  DirectoryInfo dirs = new DirectoryInfo(currDriver);  GetFilesAndDirs(dirs);  }  private void GetFilesAndDirs(DirectoryInfo directoryInfo)  {    listView1.BeginUpdate();  listView1.Items.Clear();  try  {  ListViewItem item = new ListViewItem();  foreach (DirectoryInfo dir in directoryInfo.GetDirectories())  {  item = new ListViewItem(dir.Name, 1);  item.ImageKey = "dir";  item.SubItems.Add("<DIR>");  item.SubItems.Add(dir.LastWriteTime.ToString());  listView1.Items.Add(item);  }  foreach (FileInfo file in directoryInfo.GetFiles())  {  Icon iconForFile = SystemIcons.WinLogo;  item = new ListViewItem(file.Name, 1);  if (!imageList1.Images.ContainsKey(file.Extension))  {  iconForFile = Icon.ExtractAssociatedIcon(file.FullName);  imageList1.Images.Add(file.FullName + file.Extension, iconForFile);  }  item.ImageKey = file.FullName + file.Extension;  item.SubItems.Add(file.Extension);  item.SubItems.Add(file.LastWriteTime.ToString());  item.SubItems.Add(file.Length.ToString());  listView1.Items.Add(item);  }  }  catch (Exception)  {  }  listView1.EndUpdate();  }  private void toolStripMenuCipher\_Click(object sender, EventArgs e)  {  ListViewItem item = new ListViewItem();  item = listView1.SelectedItems[0];  crypt.Encrypt(currenDirectory.FullName + "/" + item.Text, currenDirectory.FullName + "/" + item.Text + ".crypt");    }  private void toolStripMenuDeСipher\_Click(object sender, EventArgs e)  {  ListViewItem item = new ListViewItem();  item = listView1.SelectedItems[0];  if (item.Text.Contains(".crypt"))  {  crypt.Decrypt(currenDirectory.FullName + "/" + item.Text, currenDirectory.FullName + "/" + item.Text.Replace(".crypt", ""));  }    }  private void toolStripAddToArchive\_Click(object sender, EventArgs e)  {  DirectoryInfo dir = Directory.CreateDirectory(currenDirectory.FullName + "/" + "TEMPDIRZIP");  foreach (ListViewItem item in listView1.SelectedItems)  {  File.Move(currenDirectory.FullName+"/"+item.Text, dir.FullName+ "/" + item.Text);  }  workZip.CreateSample(currenDirectory.FullName + "/" + "NewZip.zip", null,dir.FullName);  if (dir.Exists)  {  dir.Delete(true);  }      }  private void toolStripDeArchive\_Click(object sender, EventArgs e)  {  ListViewItem item = new ListViewItem();  item = listView1.SelectedItems[0];  if (item.Text.Contains(".zip"))  {  this.Text = currenDirectory.FullName + item.Text;  workZip.ExtractZipFile(currenDirectory.FullName + item.Text, null, currenDirectory.FullName);  }    }      }  } |

***Crypto.cs***

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.IO;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace WindowsFormsApp1.Additional  {  class Crypto  {  private readonly int[] \_key;  private readonly int[] \_inversedKey;  public Crypto(string key)  {  var indexPairs = key  .Select((chr, idx1) => new { chr, idx1 })  .OrderBy(arg => arg.chr)  .Select((arg, idx2) =>  new  {  arg.idx1,  idx2  })  .ToArray();  \_key = indexPairs  .OrderBy(arg => arg.idx1)  .Select(arg => arg.idx2)  .ToArray();  \_inversedKey = indexPairs  .OrderBy(arg => arg.idx2)  .Select(arg => arg.idx1)  .ToArray();  }  public void Encrypt(string sourceFile, string destinationFile)  {  EncryptDecrypt(sourceFile, destinationFile, \_key);  }  public void Decrypt(string sourceFile, string destinationFile)  {  EncryptDecrypt(sourceFile, destinationFile, \_inversedKey);  }  private static void EncryptDecrypt(string sourceFile, string destinationFile, int[] key)  {  var keyLength = key.Length;  var buffer1 = new byte[keyLength];  var buffer2 = new byte[keyLength];  using (var source = new FileStream(sourceFile, FileMode.Open))  using (var destination = new FileStream(destinationFile, FileMode.OpenOrCreate))  {  while (true)  {  var read = source.Read(buffer1, 0, keyLength);  if (read == 0)  {  return;  }  else if (read < keyLength)  {  for (int i = read; i < keyLength; i++)  {  buffer1[i] = (byte)' ';  }  }  for (var i = 0; i < keyLength; i++)  {  var idx = i / keyLength \* keyLength + key[i % keyLength];  buffer2[idx] = buffer1[i];  }  destination.Write(buffer2, 0, keyLength);  }  }  }  }  } |

***WorkZip.cs***

|  |
| --- |
| using ICSharpCode.SharpZipLib.Core;  using ICSharpCode.SharpZipLib.Zip;  using System;  using System.Collections.Generic;  using System.IO;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace WindowsFormsApp1.Additional  {  class WorkZip  {  public void CreateSample(string outPathname, string password, string folderName)  {  using (FileStream fsOut = File.Create(outPathname))  using (var zipStream = new ZipOutputStream(fsOut))  {  zipStream.SetLevel(3);  zipStream.Password = password;  int folderOffset = folderName.Length + (folderName.EndsWith("\\") ? 0 : 1);  CompressFolder(folderName, zipStream, folderOffset);  }  }  public void ExtractZipFile(string archivePath, string password, string outFolder)  {  using (var fsInput = File.OpenRead(archivePath))  using (var zf = new ZipFile(fsInput))  {  if (!String.IsNullOrEmpty(password))  {  zf.Password = password;  }  foreach (ZipEntry zipEntry in zf)  {  if (!zipEntry.IsFile)  {  continue;  }  String entryFileName = zipEntry.Name;  var fullZipToPath = Path.Combine(outFolder, entryFileName);  var directoryName = Path.GetDirectoryName(fullZipToPath);  if (directoryName.Length > 0)  {  Directory.CreateDirectory(directoryName);  }  var buffer = new byte[4096];  using (var zipStream = zf.GetInputStream(zipEntry))  using (Stream fsOutput = File.Create(fullZipToPath))  {  StreamUtils.Copy(zipStream, fsOutput, buffer);  }  }  }  }  private void CompressFolder(string path, ZipOutputStream zipStream, int folderOffset)  {  var files = Directory.GetFiles(path);  foreach (var filename in files)  {  var fi = new FileInfo(filename);  var entryName = filename.Substring(folderOffset);  entryName = ZipEntry.CleanName(entryName);  var newEntry = new ZipEntry(entryName);  newEntry.DateTime = fi.LastWriteTime;  newEntry.Size = fi.Length;  zipStream.PutNextEntry(newEntry);  var buffer = new byte[4096];  using (FileStream fsInput = File.OpenRead(filename))  {  StreamUtils.Copy(fsInput, zipStream, buffer);  }  zipStream.CloseEntry();  }  var folders = Directory.GetDirectories(path);  foreach (var folder in folders)  {  CompressFolder(folder, zipStream, folderOffset);  }  }  }  } |

***Висновки:*** в ході виконання лабораторної роботи було ознайомлено зроботою з файлами. Досліджено та отримано практичні навики щодо створення архівів та шифрування.