МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ТЕРНОПІЛЬСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ

ФАКУЛЬТЕТ КОМП’ЮТЕРНИХ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра комп’ютерних наук

**КОМПЛЕКСНО-ПРАКТИЧНЕ**

**ІНДИВІДУАЛЬНЕ ЗАВДАННЯ**

**з навчальної дисципліни “Конструювання програмного забезпечення”**

**Виконав:** ст.гр. ПЗС-31

Гончарук Д. Ю.

**Перевірив:** Сирник О.Й.

ТНЕУ– 2015

**Код програми**

using System;

using System.Drawing;

using System.Collections;

using System.ComponentModel;

using System.Windows.Forms;

using System.Data;

using System.IO;

using System.Management;

using System.Globalization;

namespace My\_Explorer

{

// Author: Paul Li

// Create Date: 8/1/2002

public class Explorer : System.Windows.Forms.Form

{

private System.Windows.Forms.Splitter splitter1;

private System.Windows.Forms.MainMenu mainMenu1;

private System.Windows.Forms.MenuItem menuItem1;

private System.Windows.Forms.MenuItem menuItem2;

private System.Windows.Forms.MenuItem menuItem3;

private System.Windows.Forms.MenuItem menuItem4;

private System.Windows.Forms.TreeView tvFolders;

private System.Windows.Forms.ListView lvFiles;

private System.Windows.Forms.ImageList m\_imageListTreeView;

private System.ComponentModel.IContainer components;

public Explorer()

{

//

// Required for Windows Form Designer support

//

InitializeComponent();

// Populate TreeView with Drive list

PopulateDriveList();

}

/// <summary>

/// Clean up any resources being used.

/// </summary>

protected override void Dispose( bool disposing )

{

if( disposing )

{

if (components != null)

{

components.Dispose();

}

}

base.Dispose( disposing );

}

#region Windows Form Designer generated code

/// <summary>

/// Required method for Designer support - do not modify

/// the contents of this method with the code editor.

/// </summary>

private void InitializeComponent()

{

this.components = new System.ComponentModel.Container();

System.Resources.ResourceManager resources = new System.Resources.ResourceManager(typeof(Explorer));

this.tvFolders = new System.Windows.Forms.TreeView();

this.m\_imageListTreeView = new System.Windows.Forms.ImageList(this.components);

this.splitter1 = new System.Windows.Forms.Splitter();

this.lvFiles = new System.Windows.Forms.ListView();

this.mainMenu1 = new System.Windows.Forms.MainMenu();

this.menuItem1 = new System.Windows.Forms.MenuItem();

this.menuItem2 = new System.Windows.Forms.MenuItem();

this.menuItem3 = new System.Windows.Forms.MenuItem();

this.menuItem4 = new System.Windows.Forms.MenuItem();

this.SuspendLayout();

//

// tvFolders

//

this.tvFolders.Dock = System.Windows.Forms.DockStyle.Left;

this.tvFolders.ImageList = this.m\_imageListTreeView;

this.tvFolders.Name = "tvFolders";

this.tvFolders.Size = new System.Drawing.Size(168, 357);

this.tvFolders.TabIndex = 2;

this.tvFolders.AfterSelect += new System.Windows.Forms.TreeViewEventHandler(this.tvFolders\_AfterSelect);

//

// m\_imageListTreeView

//

this.m\_imageListTreeView.ColorDepth = System.Windows.Forms.ColorDepth.Depth8Bit;

this.m\_imageListTreeView.ImageSize = new System.Drawing.Size(16, 16);

this.m\_imageListTreeView.ImageStream = ((System.Windows.Forms.ImageListStreamer)(resources.GetObject("m\_imageListTreeView.ImageStream")));

this.m\_imageListTreeView.TransparentColor = System.Drawing.Color.Transparent;

//

// splitter1

//

this.splitter1.Location = new System.Drawing.Point(168, 0);

this.splitter1.Name = "splitter1";

this.splitter1.Size = new System.Drawing.Size(3, 357);

this.splitter1.TabIndex = 3;

this.splitter1.TabStop = false;

//

// lvFiles

//

this.lvFiles.Dock = System.Windows.Forms.DockStyle.Fill;

this.lvFiles.Location = new System.Drawing.Point(171, 0);

this.lvFiles.Name = "lvFiles";

this.lvFiles.Size = new System.Drawing.Size(429, 357);

this.lvFiles.TabIndex = 4;

this.lvFiles.View = System.Windows.Forms.View.Details;

//

// mainMenu1

//

this.mainMenu1.MenuItems.AddRange(new System.Windows.Forms.MenuItem[] {

this.menuItem1,

this.menuItem3});

//

// menuItem1

//

this.menuItem1.Index = 0;

this.menuItem1.MenuItems.AddRange(new System.Windows.Forms.MenuItem[] {

this.menuItem2});

this.menuItem1.Text = "&File";

//

// menuItem2

//

this.menuItem2.Index = 0;

this.menuItem2.Text = "&Close";

this.menuItem2.Click += new System.EventHandler(this.menuItem2\_Click);

//

// menuItem3

//

this.menuItem3.Index = 1;

this.menuItem3.MenuItems.AddRange(new System.Windows.Forms.MenuItem[] {

this.menuItem4});

this.menuItem3.Text = "&Help";

//

// menuItem4

//

this.menuItem4.Index = 0;

this.menuItem4.Text = "&About";

//

// Explorer

//

this.AutoScaleBaseSize = new System.Drawing.Size(5, 13);

this.ClientSize = new System.Drawing.Size(600, 357);

this.Controls.AddRange(new System.Windows.Forms.Control[] {

this.lvFiles,

this.splitter1,

this.tvFolders});

this.Icon = ((System.Drawing.Icon)(resources.GetObject("$this.Icon")));

this.Menu = this.mainMenu1;

this.Name = "Explorer";

this.Text = "My Window Explorer";

this.ResumeLayout(false);

}

#endregion

/// <summary>

/// The main entry point for the application.

/// </summary>

[STAThread]

static void Main()

{

Application.Run(new Explorer());

}

//This procedure populate the TreeView with the Drive list

private void PopulateDriveList()

{

TreeNode nodeTreeNode;

int imageIndex = 0;

int selectIndex = 0;

const int Removable = 2;

const int LocalDisk = 3;

const int Network = 4;

const int CD = 5;

//const int RAMDrive = 6;

this.Cursor = Cursors.WaitCursor;

//clear TreeView

tvFolders.Nodes.Clear();

nodeTreeNode = new TreeNode("My Computer",0,0);

tvFolders.Nodes.Add(nodeTreeNode);

//set node collection

TreeNodeCollection nodeCollection = nodeTreeNode.Nodes;

//Get Drive list

ManagementObjectCollection queryCollection = getDrives();

foreach ( ManagementObject mo in queryCollection)

{

switch (int.Parse( mo["DriveType"].ToString()))

{

case Removable: //removable drives

imageIndex = 5;

selectIndex = 5;

break;

case LocalDisk: //Local drives

imageIndex = 6;

selectIndex = 6;

break;

case CD: //CD rom drives

imageIndex = 7;

selectIndex = 7;

break;

case Network: //Network drives

imageIndex = 8;

selectIndex = 8;

break;

default: //defalut to folder

imageIndex = 2;

selectIndex = 3;

break;

}

//create new drive node

nodeTreeNode = new TreeNode(mo["Name"].ToString() + "\\" ,imageIndex,selectIndex);

//add new node

nodeCollection.Add(nodeTreeNode);

}

//Init files ListView

InitListView();

this.Cursor = Cursors.Default;

}

private void tvFolders\_AfterSelect(object sender, System.Windows.Forms.TreeViewEventArgs e)

{

//Populate folders and files when a folder is selected

this.Cursor = Cursors.WaitCursor;

//get current selected drive or folder

TreeNode nodeCurrent = e.Node;

//clear all sub-folders

nodeCurrent.Nodes.Clear();

if (nodeCurrent.SelectedImageIndex == 0)

{

//Selected My Computer - repopulate drive list

PopulateDriveList();

}

else

{

//populate sub-folders and folder files

PopulateDirectory(nodeCurrent, nodeCurrent.Nodes);

}

this.Cursor = Cursors.Default;

}

protected void InitListView()

{

//init ListView control

lvFiles.Clear(); //clear control

//create column header for ListView

lvFiles.Columns.Add("Name",150,System.Windows.Forms.HorizontalAlignment.Left);

lvFiles.Columns.Add("Size",75, System.Windows.Forms.HorizontalAlignment.Right);

lvFiles.Columns.Add("Created", 140, System.Windows.Forms.HorizontalAlignment.Left);

lvFiles.Columns.Add("Modified", 140, System.Windows.Forms.HorizontalAlignment.Left);

}

protected void PopulateDirectory(TreeNode nodeCurrent, TreeNodeCollection nodeCurrentCollection)

{

TreeNode nodeDir;

int imageIndex = 2; //unselected image index

int selectIndex = 3; //selected image index

if (nodeCurrent.SelectedImageIndex != 0)

{

//populate treeview with folders

try

{

//check path

if(Directory.Exists(getFullPath(nodeCurrent.FullPath)) == false)

{

MessageBox.Show("Directory or path " + nodeCurrent.ToString() + " does not exist.");

}

else

{

//populate files

PopulateFiles(nodeCurrent);

string[] stringDirectories = Directory.GetDirectories(getFullPath(nodeCurrent.FullPath));

string stringFullPath = "";

string stringPathName = "";

//loop throught all directories

foreach (string stringDir in stringDirectories)

{

stringFullPath = stringDir;

stringPathName = GetPathName(stringFullPath);

//create node for directories

nodeDir = new TreeNode(stringPathName.ToString(),imageIndex,selectIndex);

nodeCurrentCollection.Add(nodeDir);

}

}

}

catch (IOException e)

{

MessageBox.Show("Error: Drive not ready or directory does not exist.");

}

catch (UnauthorizedAccessException e)

{

MessageBox.Show("Error: Drive or directory access denided.");

}

catch (Exception e)

{

MessageBox.Show("Error: " + e);

}

}

}

protected string GetPathName(string stringPath)

{

//Get Name of folder

string[] stringSplit = stringPath.Split('\\');

int \_maxIndex = stringSplit.Length;

return stringSplit[\_maxIndex-1];

}

protected void PopulateFiles(TreeNode nodeCurrent)

{

//Populate listview with files

string[] lvData = new string[4];

//clear list

InitListView();

if (nodeCurrent.SelectedImageIndex != 0)

{

//check path

if(Directory.Exists((string) getFullPath(nodeCurrent.FullPath)) == false)

{

MessageBox.Show("Directory or path " + nodeCurrent.ToString() + " does not exist.");

}

else

{

try

{

string[] stringFiles = Directory.GetFiles(getFullPath(nodeCurrent.FullPath));

string stringFileName = "";

DateTime dtCreateDate, dtModifyDate;

Int64 lFileSize = 0;

//loop throught all files

foreach (string stringFile in stringFiles)

{

stringFileName = stringFile;

FileInfo objFileSize = new FileInfo(stringFileName);

lFileSize = objFileSize.Length;

dtCreateDate = objFileSize.CreationTime; //GetCreationTime(stringFileName);

dtModifyDate = objFileSize.LastWriteTime; //GetLastWriteTime(stringFileName);

//create listview data

lvData[0] = GetPathName(stringFileName);

lvData[1] = formatSize(lFileSize);

//check if file is in local current day light saving time

if (TimeZone.CurrentTimeZone.IsDaylightSavingTime(dtCreateDate) == false)

{

//not in day light saving time adjust time

lvData[2] = formatDate(dtCreateDate.AddHours(1));

}

else

{

//is in day light saving time adjust time

lvData[2] = formatDate(dtCreateDate);

}

//check if file is in local current day light saving time

if (TimeZone.CurrentTimeZone.IsDaylightSavingTime(dtModifyDate) == false)

{

//not in day light saving time adjust time

lvData[3] = formatDate(dtModifyDate.AddHours(1));

}

else

{

//not in day light saving time adjust time

lvData[3] = formatDate(dtModifyDate);

}

//Create actual list item

ListViewItem lvItem = new ListViewItem(lvData,0);

lvFiles.Items.Add(lvItem);

}

}

catch (IOException e)

{

MessageBox.Show("Error: Drive not ready or directory does not exist.");

}

catch (UnauthorizedAccessException e)

{

MessageBox.Show("Error: Drive or directory access denided.");

}

catch (Exception e)

{

MessageBox.Show("Error: " + e);

}

}

}

}

protected string getFullPath(string stringPath)

{

//Get Full path

string stringParse = "";

//remove My Computer from path.

stringParse = stringPath.Replace("My Computer\\", "");

return stringParse;

}

protected ManagementObjectCollection getDrives()

{

//get drive collection

ManagementObjectSearcher query = new ManagementObjectSearcher("SELECT \* From Win32\_LogicalDisk ");

ManagementObjectCollection queryCollection = query.Get();

return queryCollection;

}

protected string formatDate(DateTime dtDate)

{

//Get date and time in short format

string stringDate = "";

stringDate = dtDate.ToShortDateString().ToString() + " " + dtDate.ToShortTimeString().ToString();

return stringDate;

}

protected string formatSize(Int64 lSize)

{

//Format number to KB

string stringSize = "";

NumberFormatInfo myNfi = new NumberFormatInfo();

Int64 lKBSize = 0;

if (lSize < 1024 )

{

if (lSize == 0)

{

//zero byte

stringSize = "0";

}

else

{

//less than 1K but not zero byte

stringSize = "1";

}

}

else

{

//convert to KB

lKBSize = lSize / 1024;

//format number with default format

stringSize = lKBSize.ToString("n",myNfi);

//remove decimal

stringSize = stringSize.Replace(".00", "");

}

return stringSize + " KB";

}

private void menuItem2\_Click(object sender, System.EventArgs e)

{

//quit application

this.Close();

}

}

}

**Тестування програми**

Програма Проводник аналогічна програмі Диспечер Файлов Windows 3.xx.

Вона призначена для роботи з дисками, папками та іншими об’єктами.

Подібно як і Norton Commander програма Проводник дозволяє проглядати

вмістиме дисків, каталогів на дисках, копіювати, перейменовувати,

переміщувати і знищувати файли і каталоги, проводити пошук, друк файлів,

та ін

Для запуску програми необхідно відкрити головне меню і вибрати

Программы - Проводник. Ця програма є обов(язковою у меню Программы. На

робочому столі з(явиться головне вікно цієї програми.

На лівій панелі вікна показано тільки дерево каталогів (папок) і може

відображатися вміст папок робочого столу, дисків і папок, розміщених на

них. Найменший елемент відображення - папка. Інші типи елементів -

документи, прграми, ярлики вибраної папки на лівій панелі -

відображаються тільки на правій панелі. Таким чином, права панель завжди

відображає вміст відкритої папки. Перед виконням операції на правій

панелі має бути видимим початковий об(єкт, а на лівій - місце

призначення (папка, диск або робочий стіл).

Копіювання об(єкта виконують таким чином:

на правій панелі вибрати об(єкт, який треба копіювати (клацнути по

значку вказівки миші);

натиснути клавішу Ctrl і, не відпускаючи її, сумістити вказівку миші з

піктограмою вибраного об’єкта та натиснути ліву кнопку миші;

не відпускаючи ліву кнопку миші, а також клавішу Ctrl, перетягнути копію

піктограми (з іменем) вибраного об’єкта на ліву панель і сумістити її з

піктограмою папки, де знаходиться місце призначення,та відпустити ліву

кнопку миші;

відкрити папку і перевірити, чи знаходиться об’єкт на новому місці

Якщо проходить операція переміщення, то замість клавіші Ctrl необхідно

натиснути і утримувати клавішу Alt.

Допускається виконання операцій з групами об’єктів.Для цього їх

необхідно вибрати (відмітити).Щоб виділити одну папку, досить клацнути

один раз мишкою по її назві чи розсташованій зліва від імені піктограму

(для цього можна також використовувати клавіші переміщення курсора). Для

виділення декількох папок існують такі способи:

найуніверсальніший спосіб ( спочатку виділити перший файл, а потім

натиснути клавішу Ctrl, і, тримаючи її, клацати мишею імена потрібних

файлів;

Щоб виділити групу файлів, розсташованих на панелі підряд, треба

спочатку виділити першу папку з групи, а потім натиснути клавішу Shift

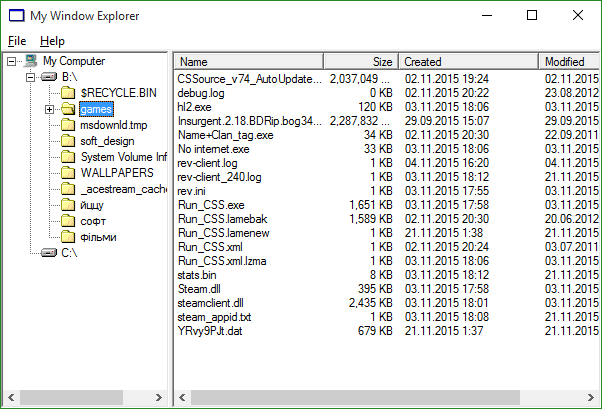
і, утримуючи її, клацнути мишкою ім(я послідньої папки з групи

щоб добавити до вже виділених папок групу папок, розсташованих на панелі

підряд, потрібно натиснути клавішу Ctrl, потім, не відпускаючи її,

клацнути ім(я першої папки з групи, потім ще натиснути і клавішу Shift і

утримуючи дві клавіші, клацнути ім(я послідньої папки з групи.



**Список використаних джерел**:

1. С#: Пер. С англ./Дейтел Х.,Дейтел П., Листфилд Дж., Нието Т., Йегер Ш., Златкина М. – СПб.: БХВ-Петербург, 2006. – 1056 с.:ил.
2. Джеффри Рихтер - CLR via C#. Программирование на платформе Microsoft .NET Framework 2.0 на языке С#. Мастер-класс./Пер. с англ. — М.: Издательство «Русская Редакция»; СПб.: Питер, 2007. — 656 стр. : ил.
3. Кристиан Нейгел, Билл Ивьен, Джей Глинн, Карли Уотсон, Морган Скиннер - С# 2008 и платформа .Net 3.5 для профессионалов.
4. [Електронний ресурус]. – Режим доступу: http://www.codeabbey.com/