**Практическая работа № 1-2**

**Разработка файлового менеджера (Explorer, Проводник)**

**с помощью WinForms**

Цель: Разработать функциональный двухпанельный файловый менеджер на языке программирования C#, способный выполнять основные операции работы с файловой системой, обеспечивая удобство просмотра и управления файлами и директориями.

В процессе выполнения задания вы будете осваивать следующие ключевые аспекты:

1. **Создание и Удаление файлов и директорий**: Приобретение навыков программного взаимодействия с файловой системой для создания, удаления и переименования файлов и директорий.

2. **Копирование и Перемещение файлов и директорий**: Разработка механизмов копирования и перемещения файлов, а также директорий внутри файловой системы.

3**. Работа с содержимым файлов**: Открытие и чтение содержимого файлов, а также освоение базовых операций записи в файл.

4. **Фильтрация файлов по расширению и дате**: Реализация функционала фильтрации файлов в зависимости от их расширения и даты создания/изменения.

5. **Рекурсивный обход директорий**: Разработка рекурсивных алгоритмов для обхода всех вложенных директорий и выполнения операций над их содержимым.

Ход работы:

**1. Создание WinForms проекта:** − В Visual Studio создайте новый проект типа Windows Forms App (.NET Framework или .NET Core, в зависимости от вашей среды разработки).

**2. Добавление компонентов для панелей:** − Разместите два объекта **ListView** на вашей форме. Это будут ваши панели, предназначенные для отображения файлов и директорий

**3. Добавление столбцов**: − Для каждого ListView добавьте столбцы для отображения информации о файлах и директориях (имя, размер, тип, дата создания, дата последнего изменения). − Вы можете это сделать в режиме дизайнера или программно в коде.

public partial class Simple\_explorer : Form

{

private bool ctrlKeyPressed = false;

private ListView lastSelectedListView;

private StringCollection clipboardPaths;

private readonly string mainDirectory = @"C:\";

**4. Переход внутрь выбранной директории и возврат**: − Для каждого ListView добавьте обработчики событий, которые реагируют на двойной клик по элементу. В обработчике события получите информацию о выбранном элементе (файле или директории)

public Simple\_explorer()

{

InitializeComponent();

this.Resize += Simple\_explorer\_Resize;

clipboardPaths = new StringCollection();

}

private void Listnumber\_one\_SelectedIndexChanged(object sender, EventArgs e)

{

if (ctrlKeyPressed)

{

**//если зажат ctrl то устанавливаем множественный выбор**

lastSelectedListView.MultiSelect = true;

}

else

{

**//если не зажат ctrl то устанавливаем одиночный выбор**

lastSelectedListView.MultiSelect = false;

}

int selectedIndex = lastSelectedListView.SelectedIndices.Count > 0 ? lastSelectedListView.SelectedIndices[0] : -1;

if (selectedIndex >= 0 && !ctrlKeyPressed)

{

ListViewItem selectedItem = lastSelectedListView.SelectedItems[0];

string selectedDirectory = selectedItem.Text;

**5. Заполнение ListView информацией о файлах и директориях**: − Используйте DirectoryInfo и FileInfo для получения информации о файлах и директориях. − Заполните соответствующий ListView полученными данными

**// выбираем первый столбец, он содержит имя директории**

string currentPath = pathEntry.Text;

string newPath = Path.Combine(currentPath, selectedDirectory);

pathEntry.Text = newPath;

ShowDirectoryContents(newPath);

}

}

private void Listnumber\_two\_SelectedIndexChanged(object sender, EventArgs e)

{

if (ctrlKeyPressed)

{

// ...

lastSelectedListView.MultiSelect = true;

}

else

{

// ...

lastSelectedListView.MultiSelect = false;

}

int selectedIndex = lastSelectedListView.SelectedIndices.Count > 0 ? lastSelectedListView.SelectedIndices[0] : -1;

if (selectedIndex >= 0 && !ctrlKeyPressed)

{

ListViewItem selectedItem = lastSelectedListView.SelectedItems[0];

string selectedDirectory = selectedItem.Text; // ...

string currentPath = pathEntry.Text;

string newPath = Path.Combine(currentPath, selectedDirectory);

pathEntry.Text = newPath;

ShowDirectoryContents(newPath);

}

}

private void simple\_explorer\_Load(object sender, EventArgs e)

{

int headerHeight = headerLabel.Height;

int navButtonHeight = navBarPanel.Height;

**6. Обновление содержимого панелей: −** Добавьте кнопки или другие элементы управления для обновления содержимого панелей. − Реализуйте обработчики событий для этих элементов, которые будут вызывать обновление ListView с новыми данными о файлах и директориях.

**// высчитываем отступ для двух ListView**

int desiredHeight = headerHeight + navButtonHeight;

**//выбираем текущий размер для ListView**

int newHeight = this.Height - desiredHeight;

**// устанавливаем позицию и размер для folderListView, 1/2**

folderListView.Size = new System.Drawing.Size(this.Width / 2 - 10, newHeight - 20);

folderListView.Location = new Point(0, headerHeight + navButtonHeight);

**//устанавливаем позицию и размер для twoFolderListView, 1/2**

twoFolderListView.Size = new System.Drawing.Size(this.Width / 2 - 10, newHeight - 20);

twoFolderListView.Location = new Point(this.Width / 2, headerHeight + navButtonHeight);

this.columnHeader1.Width = this.folderListView.Width;

this.columnHeader2.Width = this.twoFolderListView.Width;

folderListView.Enter += OnFolderListBoxClick;

twoFolderListView.Enter += OnTwoFolderListBoxClick;

folderListView.KeyDown += OnFolderListKeyDown;

folderListView.KeyUp += OnFolderListKeyUp;

**//устанавливаем все иконки который буду использовать**

var homeIcon = IconChar.Home;

var fileIcon = IconChar.Folder;

var prevIcon = IconChar.LeftLong;

var archIcon = IconChar.FileZipper;

var unpackIcon = IconChar.FileDownload;

var copyIcon = IconChar.Copy;

var pasteIcon = IconChar.Paste;

var deleteIcon = IconChar.Remove;

**// добавляем отступ для иконок относительно заголовка у кнопки**

var homeIconImage = AddPaddingToIcon(homeIcon, 15, Color.White, 0, FlipOrientation.Normal, 10);

var fileIconImage = AddPaddingToIcon(fileIcon, 15, Color.White, 0, FlipOrientation.Normal, 10);

var prevIconImage = AddPaddingToIcon(prevIcon, 15, Color.White, 0, FlipOrientation.Normal, 10);

var archiveIconImage = AddPaddingToIcon(archIcon, 15, Color.White, 0, FlipOrientation.Normal, 10);

var unpackIconImage = AddPaddingToIcon(unpackIcon, 15, Color.White, 0, FlipOrientation.Normal, 10);

var copyIconImage = AddPaddingToIcon(copyIcon, 15, Color.White, 0, FlipOrientation.Normal, 10);

var pasteIconImage = AddPaddingToIcon(pasteIcon, 15, Color.White, 0, FlipOrientation.Normal, 10);

var deleteIconImage = AddPaddingToIcon(deleteIcon, 15, Color.White, 0, FlipOrientation.Normal, 10);

**// Установка изображения для кнопки**

navBarHomeButton.Image = homeIconImage;

navBarFileButton.Image = fileIconImage;

navBarPrevButton.Image = prevIconImage;

navBarArchiveButton.Image = archiveIconImage;

navBarUnpackButton.Image = unpackIconImage;

navBarCopyButton.Image = copyIconImage;

navBarPasteButton.Image = pasteIconImage;

navBarDeleteButton.Image = deleteIconImage;

navBarUnpackButton.Text = "Unpack";

navBarFileButton.Text = "Dir";

navBarPrevButton.Text = "Cancel";

navBarArchiveButton.Text = "Achive";

navBarCopyButton.Text = "Copy";

navBarPasteButton.Text = "paste";

navBarDeleteButton.Text = "Delete";

pathEntry.Text = mainDirectory;

RenderListBoxContents(folderListView, mainDirectory);

RenderListBoxContents(twoFolderListView, mainDirectory);

**// устанавливаем направление**

navBarHomeButton.TextImageRelation = TextImageRelation.ImageBeforeText;

navBarFileButton.TextImageRelation = TextImageRelation.ImageBeforeText;

navBarPrevButton.TextImageRelation = TextImageRelation.ImageBeforeText;

navBarUnpackButton.TextImageRelation = TextImageRelation.ImageBeforeText;

navBarArchiveButton.TextImageRelation = TextImageRelation.ImageBeforeText;

navBarCopyButton.TextImageRelation = TextImageRelation.ImageBeforeText;

navBarPasteButton.TextImageRelation = TextImageRelation.ImageBeforeText;

navBarDeleteButton.TextImageRelation = TextImageRelation.ImageBeforeText;

}

private void OnFolderListBoxClick(object sender, EventArgs e)

{

lastSelectedListView = folderListView;

}

private void OnTwoFolderListBoxClick(object sender, EventArgs e)

{

lastSelectedListView = twoFolderListView;

}

private void OnHomeButtonClick(object sender, EventArgs e)

{

ShowDirectoryContents(mainDirectory);

}

private void OnBrowseButtonClick(object sender, EventArgs e)

{

BrowseDirectory(sender, e);

}

**// эвент нажатия кнопки назад**

private void OnPrevButtonClick(object sender, EventArgs e)

{

string currentPath = pathEntry.Text;

string parentDirectory = Directory.GetParent(currentPath)?.FullName;

if (!string.IsNullOrEmpty(parentDirectory))

{

pathEntry.Text = parentDirectory;

ShowDirectoryContents(parentDirectory);

}

}

private void OnNavBarCopyButtonClick(object sender, EventArgs e)

{

**// копируем в буфер обмена пути**

clipboardPaths.Clear();

foreach (var selectedItem in lastSelectedListView.SelectedItems)

{

string selectedText = ((ListViewItem)selectedItem).Text;

string fullPath = Path.Combine(pathEntry.Text, selectedText);

clipboardPaths.Add(fullPath);

}

MessageBox.Show("Выбранные файлы и папки скопированы в буфер обмена.");

}

private void OnNavBarPasteButtonClick(object sender, EventArgs e)

{

**// вставляем из буфера все файлы или папки**

string destinationDirectory = pathEntry.Text;

if (clipboardPaths.Count > 0)

{

foreach (string copiedPath in clipboardPaths)

{

try

{

if (File.Exists(copiedPath))

{

string destinationPath = Path.Combine(destinationDirectory, Path.GetFileName(copiedPath));

File.Copy(copiedPath, destinationPath, true);

}

else if (Directory.Exists(copiedPath))

{

string destinationPath = Path.Combine(destinationDirectory, Path.GetFileName(copiedPath));

CopyDirectory(copiedPath, destinationPath);

}

}

catch (Exception ex)

{

MessageBox.Show($"Error pasting {copiedPath}: {ex.Message}");

}

}

MessageBox.Show("Файлы и папки успешно вставлены.");

}

else

{

MessageBox.Show("Буфер обмена пуст.");

}

**// перезагрузка ListView**

ShowDirectoryContents(destinationDirectory);

}

**7. Создание и удаление файлов и директорий:** − Добавьте кнопки или другие элементы управления для создания и удаления файлов и директорий. − Реализуйте обработчики событий для этих элементов, в которых будет вызываться соответствующий код для создания и удаления.

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

{

if (lastSelectedListView.SelectedItems.Count > 0)

{

DialogResult result = MessageBox.Show("Вы уверены, что хотите удалить выбранные элементы?", "Удаление", MessageBoxButtons.YesNo, MessageBoxIcon.Warning);

if (result == DialogResult.Yes)

{

try

{

foreach (var selectedItem in lastSelectedListView.SelectedItems)

{

string selectedText = ((ListViewItem)selectedItem).Text;

string fullPath = Path.Combine(pathEntry.Text, selectedText);

if (File.Exists(fullPath))

{

File.Delete(fullPath);

}

else if (Directory.Exists(fullPath))

{

Directory.Delete(fullPath, true);

}

}

ShowDirectoryContents(pathEntry.Text);

MessageBox.Show("Удаление успешно выполнено!");

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка удаления: {ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

else

{

MessageBox.Show("Нет выбранных элементов для удаления.", "Предупреждение", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

}

private void OnNavBarArchiveButtonClick(object sender, EventArgs e)

{

if (lastSelectedListView.SelectedItems.Count > 0)

{

**// Предложите пользователю выбрать место назначения для zip-файла**

using (var saveFileDialog = new SaveFileDialog())

{

saveFileDialog.Filter = "Zip files (\*.zip)|\*.zip";

saveFileDialog.Title = "Сохранить архив";

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

{

using (var zipArchive = ZipFile.Open(saveFileDialog.FileName, ZipArchiveMode.Create))

{

foreach (var selectedItem in lastSelectedListView.SelectedItems)

{

string selectedText = ((ListViewItem)selectedItem).Text;

string fullPath = Path.Combine(pathEntry.Text, selectedText);

if (File.Exists(fullPath))

{

zipArchive.CreateEntryFromFile(fullPath, selectedText);

}

else if (Directory.Exists(fullPath))

{

{

foreach (var file in Directory.GetFiles(fullPath, "\*", SearchOption.AllDirectories))

{

string relativePath = file.Substring(pathEntry.Text.Length + 1);

zipArchive.CreateEntryFromFile(file, relativePath);

}

}

}

}

MessageBox.Show("Архивация успешно произведена!");

}

}

else

{

MessageBox.Show("Нет выбранных файлов для архивации.");

}

}

}

}

private void OnNavBarUnpackButtonClick(object sender, EventArgs e)

{

if (lastSelectedListView.SelectedItems.Count > 0)

{

using (var folderDialog = new FolderBrowserDialog())

{

folderDialog.Description = "Выберите папку для распаковки файлов";

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

{

string destinationFolder = folderDialog.SelectedPath;

foreach (var selectedItem in lastSelectedListView.SelectedItems)

{

// **процедура выбора каждого элемента в ListView**

string selectedText = ((ListViewItem)selectedItem).Text;

string fullPath = Path.Combine(pathEntry.Text, selectedText);

if (IsZipArchive(fullPath))

{

**// если это архив получаем полный путь**

string extractFolder = Path.Combine(destinationFolder, Path.GetFileNameWithoutExtension(selectedText));

try

{

ZipFile.ExtractToDirectory(fullPath, extractFolder);

}

catch (Exception ex)

{

MessageBox.Show($"Error extracting {selectedText}: {ex.Message}");

}

}

else

{

**//если не архив файл**

if (File.Exists(fullPath))

{

**8. Механизм копирования и перемещения файлов и директорий между панелями:** − Добавьте кнопки или другие элементы управления для копирования и перемещения файлов и директорий между панелями. − В обработчиках событий этих элементов вызывайте код для копирования и перемещения файлов и директорий. − При реализации механизма копирования и перемещения учтите, что вам потребуется обновить содержимое обеих панелей после выполнения этих операций.

**// копируем файл**

string destinationPath = Path.Combine(destinationFolder, selectedText);

File.Copy(fullPath, destinationPath, true);

}

else if (Directory.Exists(fullPath))

{

**// копируем директорию**

string destinationPath = Path.Combine(destinationFolder, selectedText);

CopyDirectory(fullPath, destinationPath);

}

}

}

MessageBox.Show("Распаковка успешно произведена!");

}

}

}

else

{

MessageBox.Show("Нет выбранных файлов для распаковки.");

}

}

**10. Обновление содержимого панелей после операций:** − После выполнения операций создания, удаления, копирования и перемещения, обновите содержимое обеих панелей, чтобы отразить изменения в файловой системе.

**11. Просмотр содержимого текстовых файлов:** − Добавьте элемент управления, например, TextBox или RichTextBox, для отображения содержимого текстового файла. − Предусмотрите возможность выбора текстового файла в одной из панелей, а затем отображения его содержимого в добавленном элементе управления.

**// метод копирования директории в виде проверки есть ли в каждой директории папки**

private void CopyDirectory(string sourcePath, string destinationPath)

{

Directory.CreateDirectory(destinationPath);

foreach (var file in Directory.GetFiles(sourcePath))

{

string fileName = Path.GetFileName(file);

string destinationFilePath = Path.Combine(destinationPath, fileName);

File.Copy(file, destinationFilePath);

}

foreach (var directory in Directory.GetDirectories(sourcePath))

{

string directoryName = Path.GetFileName(directory);

string destinationDirectoryPath = Path.Combine(destinationPath, directoryName);

CopyDirectory(directory, destinationDirectoryPath);

}

}

**// эвент двойного нажатия на элемент ListView**

private void OnListViewDoubleClick(object sender, EventArgs e)

{

if (lastSelectedListView != null)

{

int selectedIndex = lastSelectedListView.SelectedIndices.Count > 0 ? lastSelectedListView.SelectedIndices[0] : -1;

if (selectedIndex >= 0)

{

string selectedDirectory = lastSelectedListView.Items[selectedIndex].SubItems[0].Text;

if (lastSelectedListView.Items[selectedIndex].Tag is ListViewItem selectedItem)

{

selectedDirectory = selectedItem.Text;

}

string currentPath = pathEntry.Text;

string newPath = Path.Combine(currentPath, selectedDirectory);

pathEntry.Text = newPath;

ShowDirectoryContents(newPath);

}

}

}

private void Simple\_explorer\_Resize(object sender, EventArgs e)

{

**12. Редактирование текстовых файлов:** − Добавьте элементы управления, такие как кнопка "Редактировать" или "Сохранить", чтобы предоставить возможность редактирования содержимого текстового файла.

**13. Отдельное окно для редактирования:** − Если предпочтительно, предусмотрите возможность редактирования текстовых файлов в отдельном окне, используя, например, Form с TextBox или RichTextBox. − Предусмотрите кнопку "Сохранить" в этом отдельном окне для сохранения внесенных изменений.

**// Установка макс размера ox панели по размеру окна**

int maxPanelWidth = this.ClientSize.Width;

**// передача макс размера для установки скролла**

this.navBarPanel.MaximumSize = new System.Drawing.Size(maxPanelWidth, this.navBarPanel.Height);

this.navBarPanel.Width = this.ClientSize.Width;

this.headerLabel.Width = this.ClientSize.Width;

int headerHeight = headerLabel.Height;

int navButtonHeight = navBarPanel.Height;

**// вычисление общего отступа**

int desiredHeight = headerHeight + navButtonHeight;

**// вычисление размера с отступом**

int newHeight = this.Height - desiredHeight;

**// позиция и размер относительно отступа**

folderListView.Size = new System.Drawing.Size(this.Width / 2 - 10, newHeight - 20);

folderListView.Location = new Point(0, headerHeight + navButtonHeight);

**//позиция и размер относительно отступа**

twoFolderListView.Size = new System.Drawing.Size(this.Width / 2 - 10, newHeight - 20);

twoFolderListView.Location = new Point(this.Width / 2, headerHeight + navButtonHeight);

this.columnHeader1.Width = this.folderListView.Width;

this.columnHeader2.Width = this.twoFolderListView.Width;

}

private void RenderListBoxContents(ListView listView, string path)

{

listView.Items.Clear();

**// Очистить все элементы**

try

{

if (IsZipArchive(path))

{

AddFilesFromArchive(path);

}

else

{

**14. Обновление содержимого после редактирования:** − После завершения редактирования и сохранения, обновите содержимое соответствующей панели для отражения внесенных изменений

**// проверить есть ли дирректория**

if (!Directory.Exists(path))

{

MessageBox.Show("Invalid directory path or insufficient permissions.");

return;

}

string[] directories = Directory.GetDirectories(path);

string[] files = Directory.GetFiles(path);

**// добавить родительскую дирректорию**

if (path != mainDirectory)

{

string parentFolderName = Path.GetFileName(Path.GetDirectoryName(path));

Image folderIcon = GetIconImage(IconChar.Folder);

listView.Items.Add(new ListViewItem { Text = ".." + parentFolderName });

}

foreach (string directory in directories)

{

string folderName = Path.GetFileName(directory);

Image folderIcon = GetIconImage(IconChar.Folder);

listView.Items.Add(new ListViewItem { Text = folderName });

}

foreach (string file in files)

{

string fileName = Path.GetFileName(file);

Image fileIcon = GetIconImage(IconChar.File);

listView.Items.Add(new ListViewItem { Text = fileName });

}

}

}

catch (UnauthorizedAccessException)

{

MessageBox.Show("Ошибка доступа к дирректории или архиву.");

}

catch (Exception ex)

{

MessageBox.Show($"Error: {ex.Message}");

}

}

**17. Механизм поиска файлов в текущей директории и поддиректориях:** − Добавьте текстовое поле для ввода ключевого слова поиска. − Реализуйте обработчик события для кнопки "Поиск", в котором будет вызываться код для поиска файлов в текущей директории и её поддиректориях по указанному ключевому слову.

**// эвент навигации директорий**

private void BrowseDirectory(object sender, EventArgs e)

{

using (var folderDialog = new FolderBrowserDialog())

{

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

{

string directoryPath = folderDialog.SelectedPath;

pathEntry.Text = directoryPath;

ShowDirectoryContents(directoryPath);

}

}

}

**22. Важные атрибуты и информация**: − Включите в отображаемую информацию важные атрибуты, такие как размер файла, дата создания, атрибуты файла (только для файлов), путь и другие параметры, которые считаете важными.

**23. Динамическое обновление:** − Обеспечьте динамическое обновление информации на панели при изменении выбора файла или директории. Это можно сделать с использованием событий или регулярного опроса выбранного элемента

**// метод показа директорий**

private void ShowDirectoryContents(string path)

{

lastSelectedListView.Items.Clear();

try

{

if (IsZipArchive(path))

{

AddFilesFromArchive(path);

}

else

{

if (!Directory.Exists(path))

{

MessageBox.Show("Invalid directory path or insufficient permissions.");

return;

}

string[] directories = Directory.GetDirectories(path);

string[] files = Directory.GetFiles(path);

if (path != mainDirectory)

{

string parentFolderName = Path.GetFileName(Path.GetDirectoryName(path));

Image folderIcon = GetIconImage(IconChar.Folder);

lastSelectedListView.Items.Add(new ListViewItem { Text = ".." + parentFolderName});

}

foreach (string directory in directories)

{

string folderName = Path.GetFileName(directory);

Image folderIcon = GetIconImage(IconChar.Folder);

lastSelectedListView.Items.Add(new ListViewItem { Text = folderName});

}

foreach (string file in files)

{

string fileName = Path.GetFileName(file);

Image fileIcon = GetIconImage(IconChar.File);

lastSelectedListView.Items.Add(new ListViewItem { Text = fileName});

}

}

}

catch (UnauthorizedAccessException)

{

MessageBox.Show("Ошибка доступа к дирректории или архиву.");

}

catch (Exception ex)

{

MessageBox.Show($"Error: {ex.Message}");

}

}

private bool IsZipArchive(string path)

{

**// проверить , является ли текущий файл архивом**

return string.Equals(Path.GetExtension(path), ".zip", StringComparison.OrdinalIgnoreCase);

}

private void AddFilesFromArchive(string archivePath)

{

try

{

using (var archive = ZipFile.OpenRead(archivePath))

{

foreach (var entry in archive.Entries)

{

string entryName = Path.Combine(Path.GetFileNameWithoutExtension(archivePath), entry.FullName);

entryName = entryName.Replace(Path.DirectorySeparatorChar, Path.PathSeparator);

folderListView.Items.Add(new ListViewItem { Text = entryName});

twoFolderListView.Items.Add(new ListViewItem { Text = entryName});

}

}

}

catch (Exception ex)

{

MessageBox.Show($"Error reading archive: {ex.Message}");

}

}

private bool IsArchiveExtension(string extension)

{

string[] archiveExtensions = { ".zip", ".rar", ".7z" };

return archiveExtensions.Contains(extension, StringComparer.OrdinalIgnoreCase);

}

**// метод получения базовой иконки для FontAwesome**

private Image GetIconImage(IconChar icon)

{

int iconSize = 16;

Color iconColor = Color.Black;

**// добавить отступ к иконке**

var iconWithPadding = AddPaddingToIcon(icon, iconSize, iconColor, 0, FlipOrientation.Normal, 2);

return iconWithPadding;

}

**//эвент выбора индекса из текущего ListView**

private void OnFolderListKeyDown(object sender, KeyEventArgs e)

{

ctrlKeyPressed = e.Control;

}

private void OnFolderListKeyUp(object sender, KeyEventArgs e)

{

**// Сбросить состояние клавиши Ctrl**

ctrlKeyPressed = false;

}

**// Метод добавление отступа**

private Bitmap AddPaddingToIcon(IconChar icon, int size, Color color, double rotation, FlipOrientation flip, int padding)

{

**//получаем битмеп**

var originalIcon = icon.ToBitmap(IconFont.Auto, size, color, rotation, flip);

**// добавляем к битмепу отступ**

var iconWithPadding = new Bitmap(originalIcon.Width + padding, originalIcon.Height);

using (var graphics = Graphics.FromImage(iconWithPadding))

{

**// рендерим битмем вместе с прямоугольником**

graphics.DrawImage(originalIcon, new Rectangle(padding, 0, originalIcon.Width, originalIcon.Height));

}

return iconWithPadding;

}