# Техника безопасности на рабочем месте.

Перед тем, как включить компьютер, необходимо уделить внимание следующим действиям:

* Нужно убедиться, что в зоне досягаемости отсутствуют оголенные провода и различные шнуры;
* Нельзя начинать работу на технике с видимым повреждением;
* Предметы на столе не должны мешать обзору, пользованию мышкой и клавиатурой, поверхность экрана должна быть абсолютно чистой;
* Запрещается начинать работу в помещениях с повышенной влажностью;

При выполнении работы:

* Рекомендуется избегать расположения жидкостей рядом с модулями компьютера;
* Нельзя очищать поверхность компьютера от загрязнений, когда он включен;
* Недопустимо снимать корпус любой из составных частей ПК во время его работы;
* В помещении с компьютерами непозволительно курить или употреблять пищу на рабочем месте;
* При ощущении запаха гари, необходимо выключить ПК из сети;

Чтобы минимизировать негативные влияния при длительном нахождении за монитором, стоит знать:

* Расстояние между глазами пользователя и экраном составляет не менее полуметра;
* Клавиатура размещается за 20-30 сантиметров от края стола;
* Локти согнуты под прямым углом;
* Стул стоит таким образом, что спина немного упирается в его спинку;
* Ноги упираются в твердую поверхность, распрямлены вперед;

Содержание

[Техника безопасности на рабочем месте. 1](#_Toc167193887)

[Практическая работа № 1. Разработка файлового менеджера (Explorer, Проводник) с помощью WinForms 4](#_Toc167193888)

[1.1 Создание проекта 4](#_Toc167193889)

[1.2 Создание стилей 6](#_Toc167193890)

[1.3 Добавление функции копироваания 8](#_Toc167193891)

[1.4 Создание функции удаления 11](#_Toc167193892)

[Практическая работа №2 Разработка архиватора 12](#_Toc167193893)

[2.1 Создание функции архивации 12](#_Toc167193894)

[2.2 Создание функции для распаковки файлов 14](#_Toc167193895)

[2.3 Проверка метода директории 16](#_Toc167193896)

[2.4 Создаем окно и накладываем стили 19](#_Toc167193897)

[2.5 Проверяем наличие дириктории 21](#_Toc167193898)

[2.6 Проверка навигации в директории 23](#_Toc167193899)

[2.7 Создаем отображение директории 24](#_Toc167193900)

[2.8 Создаем проверку архива 26](#_Toc167193901)

[2.9 Создание иконки 28](#_Toc167193902)

[2.10 Создание эвента для индекса 29](#_Toc167193903)

[2.11 Создаем стили для битмена 31](#_Toc167193904)

[Практическая работа №3 Разработка библиотеки файловой БД 34](#_Toc167193905)

[3.1 XML смеха 34](#_Toc167193906)

Практическая работа № 1. Разработка файлового менеджера (Explorer, Проводник) с помощью WinForms

* 1. Создание проекта.

1) Настройка ListView для отображения имен файлов и папок. Установка свойства View в Details, чтобы отображать иконки, имена и другие сведения о файлах и папках.

2) Добавление обработчики событий для ListView, чтобы реагировать на выбор элементов и изменение текущего пути.

3) Реализуем метод ShowDirectoryContents, который будет отображать содержимое директории в ListView. Этот метод должен читать содержимое директории, определять тип каждого элемента (файл или папка) и добавлять соответствующие элементы в ListView



Рисунок 1 – Директория

* 1. Создание стилей

1) Создаем метод simple\_explorer\_Load

2) Вычисляются высоты заголовка и панели навигации, а также вычисляется желаемая высота для двух ListView. Размеры и позиции для каждого ListView устанавливаются таким образом, чтобы они занимали половину ширины формы и занимали оставшуюся высоту.

3) Устанавливается ширина столбцов в соответствии с шириной ListView.

4) Добавляются обработчики событий для Enter, KeyDown, и KeyUp для ListView

5) Используется метод AddPaddingToIcon для добавления отступов и цвета к иконкам, а затем устанавливаются эти иконки на кнопки на панели навигации.

6) Устанавливается текст и направление для кнопок на панели навигации.

7) Устанавливается начальный путь в TextBox и вызывается метод RenderListBoxContents для отображения содержимого директории в обоих ListView.

8) Устанавливается направление текста и иконки для кнопок на панели навигации.

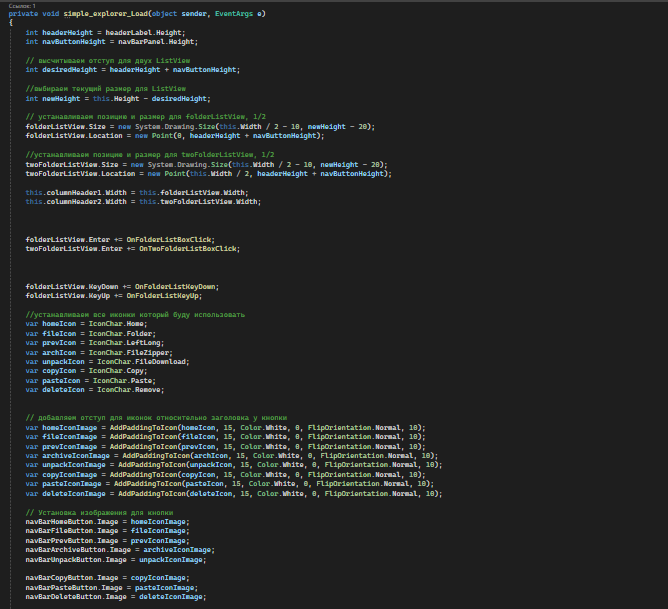


Рисунок 2.1 – Инициализация компонентов

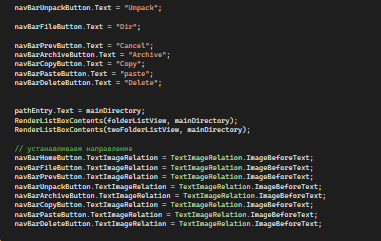


Рисунок 2.2 – Инициализация компонентов

* 1. Добавление функции копироваания

1) OnFolderListBoxClick и OnTwoFolderListBoxClick - при выборе элемента в ListView, сохраняется ссылка на выбранный ListView в lastSelectedListView

2) Обработчики событий для кнопок:

* OnHomeButtonClick - при нажатии на кнопку "Home", отображается содержимое основной директории.
* OnBrowseButtonClick - при нажатии на кнопку "Browse", вызывается метод BrowseDirectory для просмотра директории.
* OnPrevButtonClick - при нажатии на кнопку "Prev", переходит к родительской директории текущей директории.
* OnNavBarCopyButtonClick - при нажатии на кнопку "Copy", копирует выбранные файлы и папки в буфер обмена.
* OnNavBarPasteButtonClick - при нажатии на кнопку "Paste", вставляет файлы и папки из буфера обмена в текущую директорию.

3) В методе OnNavBarCopyButtonClick выбранные пути файлов и папок копируются в clipboardPaths

4) В методе OnNavBarPasteButtonClick файлы и папки из буфера обмена вставляются в текущую директорию. Если вставляется файл, он копируется, если папка - вызывается метод CopyDirectory для рекурсивного копирования

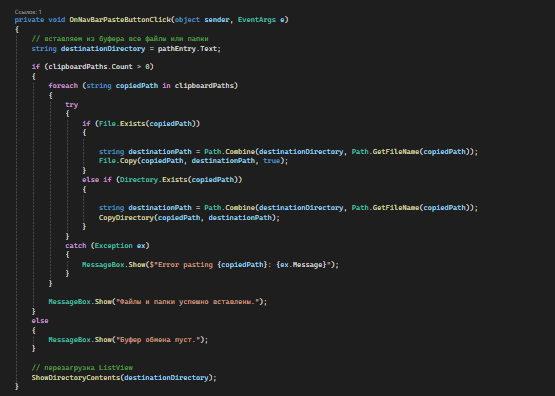
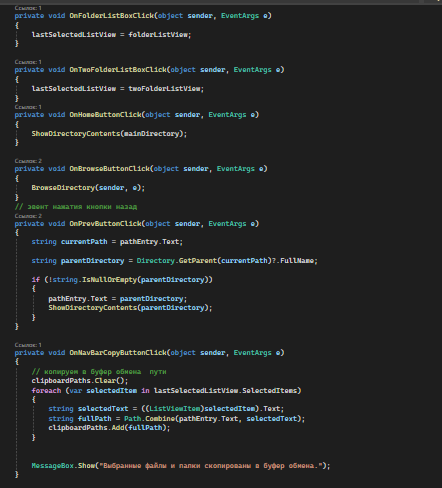


Рисунок 3 – Реализуем копирование

* 1. Создание функции удаления

В методе navBarDeleteButton\_Click выполняются следующие шаги:

1. Перед удалением проверяется, есть ли выбранные элементы в lastSelectedListView. Если выбранных элементов нет, выводится сообщение об отсутствии выбранных элементов для удаления.
2. Если есть выбранные элементы, появляется диалоговое окно с предупреждением о том, что выбранные элементы будут удалены. Пользователю предлагается выбрать "Да" или "Нет".
3. : Если пользователь выбрал "Да", происходит попытка удаления каждого выбранного элемента. Если элемент является файлом, он удаляется с помощью File.Delete, а если элемент является директорией, она удаляется с помощью Directory.Delete.
4. После успешного удаления элементов вызывается метод ShowDirectoryContents для обновления содержимого директории в ListView.

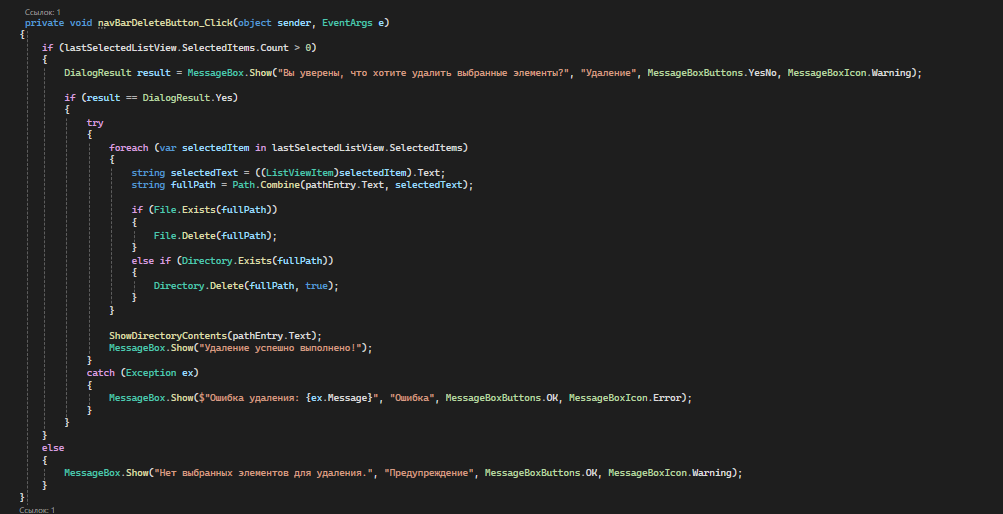


Рисунок 4 – Реализуем удаление

Практическая работа №2 Разработка архиватора

* 1. Создание функции архивации

В методе OnNavBarArchiveButtonClick выполняются следующие шаги:

1. Перед архивацией проверяется, есть ли выбранные элементы в lastSelectedListView. Если выбранных элементов нет, выводится сообщение об отсутствии выбранных файлов для архивации.
2. Используется SaveFileDialog для предложения пользователю выбрать место назначения для zip-файла. Фильтр файлов устанавливается на "Zip files (*.zip)|*.zip", чтобы пользователь мог видеть только zip-файлы.
3. Если пользователь выбрал место для сохранения, используется ZipFile.Open для создания нового zip-архива с указанным путем и режимом ZipArchiveMode.Create.
4. Для каждого выбранного элемента в lastSelectedListView происходит проверка на то, является ли элемент файлом или директорией. Если элемент является файлом, он добавляется в архив с помощью zipArchive.CreateEntryFromFile. Если элемент является директорией, все файлы в ней (включая файлы во вложенных директориях) добавляются в архив, используя рекурсивный обход директорий с помощью Directory.GetFiles.

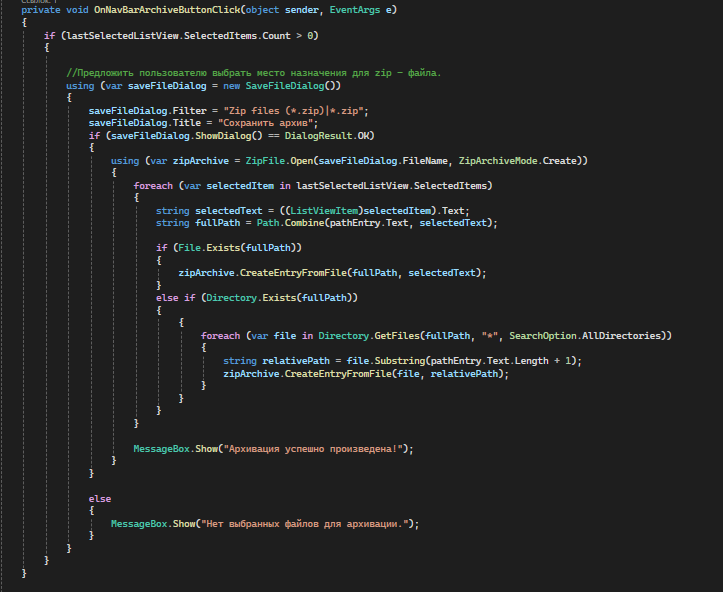


Рисунок 5 – Реализуем архивацию

* 1. Создание функции для распаковки файлов

В методе OnNavBarUnpackButtonClick выполняются следующие шаги:

1. Перед распаковкой проверяется, есть ли выбранные элементы в lastSelectedListView. Если выбранных элементов нет, выводится сообщение об отсутствии выбранных файлов для распаковки.
2. Используется FolderBrowserDialog для предложения пользователю выбрать папку, в которую будут распакованы файлы. Описание диалога устанавливается на "Выберите папку для распаковки файлов".
3. Если пользователь выбрал папку для распаковки, для каждого выбранного элемента в lastSelectedListView происходит проверка на то, является ли элемент zip-архивом. Если элемент является архивом, он распаковывается в выбранную папку с помощью ZipFile.ExtractToDirectory. Если элемент не является архивом, но файл или директория существует, он копируется в выбранную папку.



Рисунок 6 – Реализуем метод для распаковки файлов

* 1. Проверка метода директории

1. Реализация метода копирования директории:
   * Создание новой директории в место назначения с помощью Directory.CreateDirectory.
   * Копирование всех файлов из исходной директории в новую директорию с помощью цикла foreach и File.Copy.
   * Рекурсивное копирование всех поддиректорий с помощью вложенного цикла foreach и вызова CopyDirectory.
2. Реализация обработчика события двойного нажатия на элемент ListView:
   * Проверка наличия выбранного элемента в lastSelectedListView.
   * Извлечение индекса выбранного элемента и его имени.
   * Обновление текущего пути с помощью pathEntry.Text и отображение содержимого новой директории с помощью ShowDirectoryContents.
3. Использование метода копирования директории:
   * Вызов CopyDirectory для копирования директории при необходимости.
4. Подключение обработчика события двойного нажатия на элемент ListView:
   * Назначение метода OnListViewDoubleClick в качестве обработчика события двойного нажатия на элемент ListView.

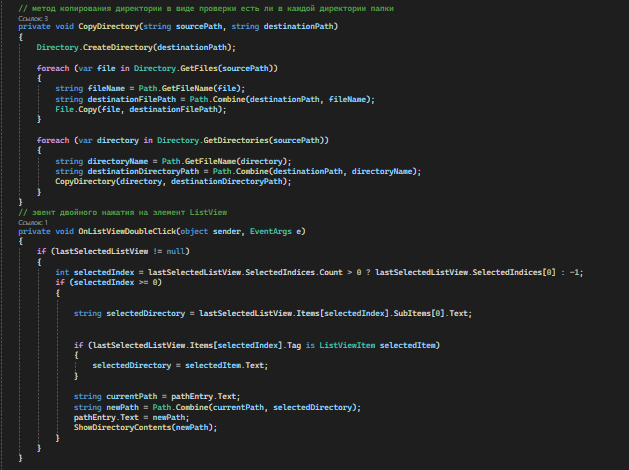


Рисунок 7 – Проверяем метод проверки директории и “’эвент двойного нажатия”

* 1. Создаем окно и накладываем стили

В методе Simple\_explorer\_Resize выполняются следующие шаги:

1. Установка максимального размера панели навигации: Размер панели навигации (navBarPanel) устанавливается равным ширине клиентской области окна.
2. Установка ширины заголовка и панели навигации: Ширина заголовка (headerLabel) и панели навигации (navBarPanel) также устанавливается равной ширине клиентской области окна.
3. Вычисление желаемой высоты: Вычисляется желаемая высота, которая включает в себя высоту заголовка и панели навигации.
4. Вычисление новой высоты: Вычисляется новая высота, которая равна разнице между высотой окна и желаемой высотой.
5. Установка размера и позиции для folderListView: Размер и позиция folderListView устанавливаются таким образом, чтобы они занимали половину ширины окна и занимали всю оставшуюся высоту, за вычетом отступов.
6. Установка размера и позиции для twoFolderListView: Размер и позиция twoFolderListView устанавливаются так же, как и для folderListView, но они начинаются с середины окна.
7. Установка ширины заголовков столбцов: Ширина заголовков столбцов (columnHeader1 и columnHeader2) устанавливается равной ширине соответствующих ListView.

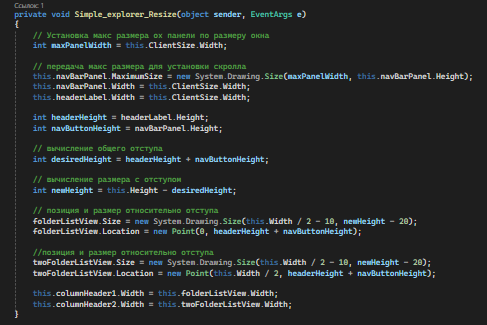


Рисунок 8 – Задаем размер окна и отступы

* 1. Проверяем наличие директории

В методе RenderListBoxContents выполняются следующие шаги:

1. Вызывается метод Clear для очистки всех элементов в ListView.
2. Если путь указывает на zip-архив, вызывается метод AddFilesFromArchive для добавления файлов из архива в ListView.
3. Если путь не указывает на zip-архив, проверяется наличие директории по указанному пути. Если директория не существует, выводится сообщение об ошибке.
4. Если директория существует, получаются списки директорий и файлов в указанной директории с помощью Directory.GetDirectories и Directory.GetFiles.
5. Если путь не совпадает с основной директорией, добавляется элемент ".." (родительская директория) в ListView.
6. Для каждой директории и файла в списках директорий и файлов создается новый элемент ListViewItem с именем файла или директории и добавляется в ListView.
7. Если возникает исключение UnauthorizedAccessException, выводится сообщение об ошибке доступа к директории или архиву. Если возникает другое исключение, выводится сообщение об общей ошибке с информацией об исключении.



Рисунок 9 – Создаем метод для проверки наличия директории

* 1. Проверка навигации в директории

В методе BrowseDirectory выполняются следующие шаги:

1. Инициализация диалога выбора директории: Создается экземпляр FolderBrowserDialog, который позволяет пользователю выбрать директорию.
2. Отображение диалога выбора директории: Вызывается метод ShowDialog, который отображает диалоговое окно для выбора директории. Если пользователь нажимает "ОК", результат диалога будет DialogResult.OK.
3. Получение выбранного пути директории: Если пользователь нажал "ОК", из диалога можно получить выбранный путь директории с помощью свойства SelectedPath.
4. Обновление текстового поля пути: Выбранный путь записывается в текстовое поле pathEntry.
5. Отображение содержимого директории: Вызывается метод ShowDirectoryContents, передавая в него выбранный путь директории. Этот метод отвечает за отображение файлов и поддиректорий выбранной директории.
6. Обработка исключений: Если возникает исключение, оно будет перехвачено и обработано внутри блока using, который автоматически закроет диалог.

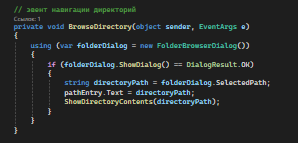


Рисунок 10 – Проверяем работу навигации в директории

* 1. Создаем отображение директории

В методе ShowDirectoryContents выполняются следующие шаги:

1. Очистка элементов ListView: Вызывается метод Clear для очистки всех элементов в ListView (lastSelectedListView).
2. Проверка на архив: Если путь указывает на zip-архив, вызывается метод AddFilesFromArchive для добавления файлов из архива в ListView.
3. Проверка наличия директории: Если путь не указывает на zip-архив, проверяется наличие директории по указанному пути. Если директория не существует, выводится сообщение об ошибке.
4. Получение списков директорий и файлов: Если директория существует, получаются списки директорий и файлов в указанной директории с помощью Directory.GetDirectories и Directory.GetFiles.
5. Добавление родительской директории: Если путь не совпадает с основной директорией, добавляется элемент ".." (родительская директория) в ListView.
6. Добавление директорий и файлов: Для каждой директории и файла в списках директорий и файлов создается новый элемент ListViewItem с именем файла или директории и добавляется в ListView.
7. Обработка исключений: Если возникает исключение UnauthorizedAccessException, выводится сообщение об ошибке доступа к директории или архиву. Если возникает другое исключение, выводится сообщение об общей ошибке с информацией об исключении.

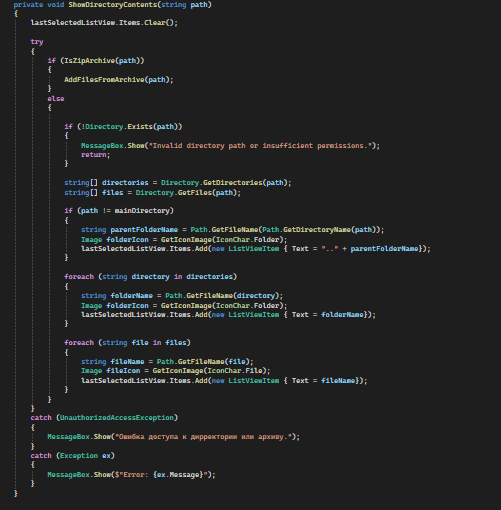


Рисунок 11 – Реализуем отображение директории

* 1. Создаем проверку архива

Этапы создания:

1. Реализация метода IsZipArchive:
   * Используется Path.GetExtension для получения расширения файла из пути.
   * Сравнивает полученное расширение с ".zip" с учетом регистра.
2. Реализация метода AddFilesFromArchive:
   * Открывает zip-архив с помощью ZipFile.OpenRead.
   * Итерирует по записям архива и добавляет их в ListView.
   * Обрабатывает исключения, выводя сообщение об ошибке.
3. Реализация метода IsArchiveExtension:
   * Определяет список известных расширений архивов.
   * Использует Contains для проверки, содержится ли расширение в списке известных расширений.
4. Использование методов в коде:
   * Вызывается IsZipArchive для проверки, является ли файл архивом.
   * Вызывается AddFilesFromArchive для добавления файлов из архива в ListView.
   * Вызывается IsArchiveExtension для проверки, является ли расширение архивом.

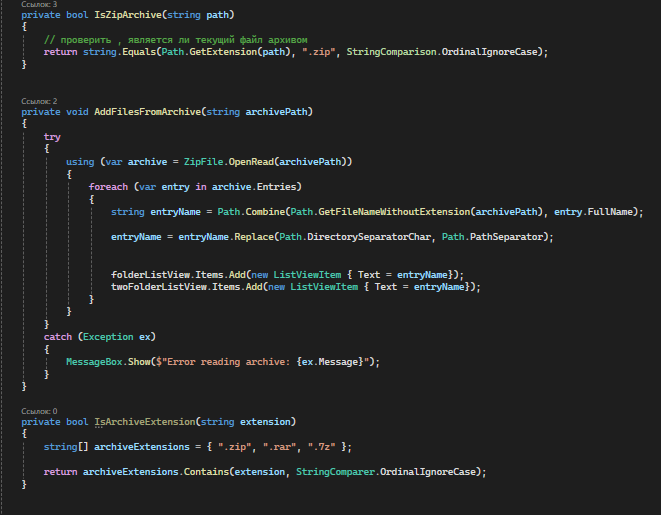


Рисунок 12 – Реализуем проверку файла на то является ли он архивом

* 1. Создание иконки

В методе GetIconImage выполняются следующие шаги:

1. Инициализация размера иконки и ее цвета: Размер иконки устанавливается равным 16, а цвет - черный.
2. Добавление отступа к иконке: Вызывается метод AddPaddingToIcon, который добавляет отступ к иконке. В качестве параметров передаются значения иконки, ее размера, цвета, отступа, ориентации и толщины линий.
3. Возврат иконки с отступом: Метод возвращает иконку с добавленным отступом.

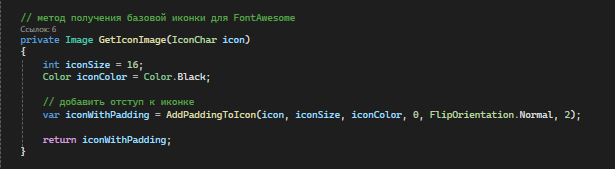


Рисунок 13 – Создаем иконку

* 1. Создание эвента для индекса

Этапы создания:

1. Реализация метода OnFolderListKeyDown:
   * Обрабатывает событие KeyDown для ListView.
   * Извлекает информацию о нажатой клавише из аргумента события KeyEventArgs.
   * Проверяет, была ли нажата клавиша Ctrl.
   * Устанавливает соответствующий флаг в зависимости от состояния клавиши Ctrl.
2. Реализация метода OnFolderListKeyUp:
   * Обрабатывает событие KeyUp для ListView.
   * Сбрасывает флаг ctrlKeyPressed, указывая, что клавиша Ctrl больше не нажата.
3. Подключение обработчиков событий:
   * Подключает методы OnFolderListKeyDown и OnFolderListKeyUp к событиям KeyDown и KeyUp соответственно для ListView.
4. Инициализация флага ctrlKeyPressed:
   * Объявляет переменную ctrlKeyPressed в классе, которая будет использоваться для отслеживания состояния клавиши Ctrl.
5. Использование флага ctrlKeyPressed:
   * В зависимости от значения флага ctrlKeyPressed, можно реализовать дополнительные функции, такие как выбор нескольких элементов или выполнение других действий при нажатой клавише Ctrl.

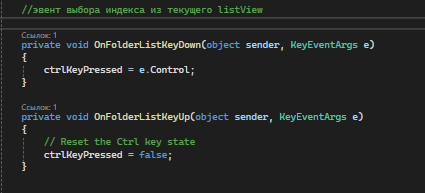


Рисунок 14 – Создаем эвент для выбора индекса

2.14 Создаем стили для битмена

В методе AddPaddingToIcon выполняются следующие шаги:

1. Получение исходной иконки в виде Bitmap: Используется метод ToBitmap для преобразования иконки в Bitmap с заданными параметрами размера, цвета, поворота и отражения.
2. Создание нового Bitmap с отступом: Создается новый Bitmap с шириной, равной ширине исходной иконки плюс отступ, и высотой, равной высоте исходной иконки.
3. Рендеринг иконки с отступом: Используется класс Graphics для рисования исходной иконки на новом Bitmap с отступом. Отступ добавляется слева, а высота и ширина остаются без изменений.
4. Возврат Bitmap с отступом: Возвращается Bitmap с добавленным отступом.

Этапы создания:

1. Преобразование иконки в Bitmap:
   * Используется метод ToBitmap для преобразования иконки в Bitmap.
2. Создание нового Bitmap с отступом:
   * Создается новый Bitmap с увеличенной шириной для добавления отступа.
3. Рендеринг иконки с отступом:
   * Используется класс Graphics для рисования исходной иконки на новом Bitmap с отступом.
4. Возврат Bitmap с отступом:
   * Возвращается Bitmap с добавленным отступом.
5. Обработка исключений:
   * Необходимо учитывать возможность исключений, которые могут возникнуть при работе с Bitmap и Graphics.
6. Использование метода AddPaddingToIcon:
   * Вызывается метод AddPaddingToIcon с необходимыми параметрами для добавления отступа к иконке.

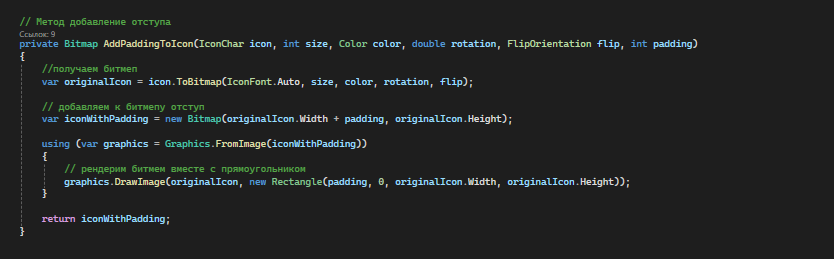


Рисунок 15 – Внешний вид битмепа

Практическая работа №3 Разработка библиотеки файловой БД

* 1. XML смеха

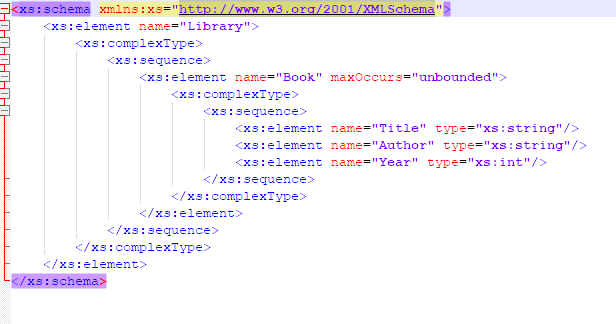


Рисунок 16 – XML схема