

**УНИВЕРСИТЕТ ПО БИБЛИОТЕКОЗНАНИЕ И ИНФОРМАЦИОННИ ТЕХНОЛОГИИ  
  
КАТЕДРА „КОМПЮТЪРНИ НАУКИ“  
СПЕЦИАЛНОСТ „КОМПЮТЪРНИ НАУКИ“**

**КУРСОВА РАБОТА**

**на тема:**

**AP BitRenamer – “Преименувач” на много файлове   
задача по Визуални среди за програмиране**

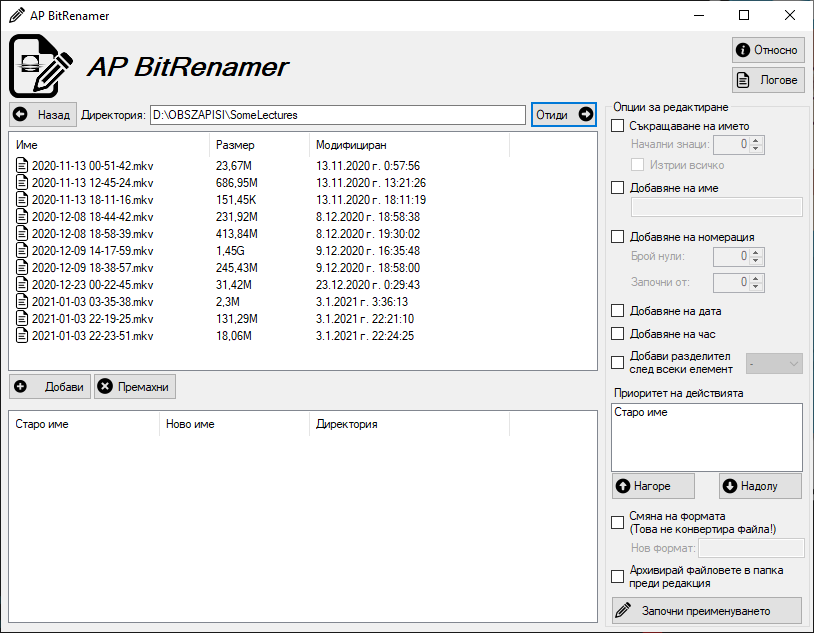
**Дипломант:**Антонио Петков  
Задочно обучение, 2 курс  
Фак. Ном. : 135КНЗ

Научен ръководител:  
проф. д-р Иван Иванов

СОФИЯ  
­­­­­­­­2023г.

**ПРЕДСТАВЯНЕ:**

Когато потребител се опита да преименува няколко файла в Windows Explorer(или File Explorer), Windows ги преименува в избраното име и поредния номер в скоби, например File.doc, File (1).doc, File (2).doc. Ако потребителят се опитва да промени сгрешен формат накрая(например от JFIF да се смени на JPEG), програмата Windows се обърква и ги преименува по този начин – File.jpeg.jfif, File.jpeg (2).jfif, File.jpeg (3).jfif.



Целта на тази програма е да улесни и забърза работата на потребителя в преименуването на голямо количество файлове, като предлага различни опции за именуване. В опциите са включени:

* Съкращаване на старото име
* Пълно премахване на старото име
* Добавяне на ново име
* Добавяне на поредна номерация
* Добавяне на датата и/или часът на модификация на файла
* Смяна на формата (не конвертира файла, само му сменя формата в името)

Идеята и вдъхновението за тази програма са взети от програмата [Advanced Renamer](https://www.advancedrenamer.com/) и функцията [PowerRenamer](https://learn.microsoft.com/en-us/windows/powertoys/powerrename) от [PowerToys](https://learn.microsoft.com/en-us/windows/powertoys/) колекцията от Microsoft

**СЪДЪРЖАНИЕ:**

1. Действие с програмата
2. Програмна логика и функциизз

2.1. Главни променливи

2.2. Функции в програмата

2.2.1. CreateHeadersAndFillListView()

2.2.2. MakeHradersSelectedFilesList()

2.2.3. PaintListView(string root)

2.2.4. PaintSelectedFilesView()

2.2.5. UpdateNewNames()

2.2.6. BeginRenaming()

2.2.7. WriteToLog(string message)

2.3. Проверки, валидация и обработка на грешки в програмата

2.3.1. Обработка на грешки при навигиране на папките

2.3.2. Обработка на грешки при добавяне на файлове в списъка

2.3.3. Ограничаване на входните данни за създаване на името

2.3.4. Проверка за потенциални дубликати след редакция.

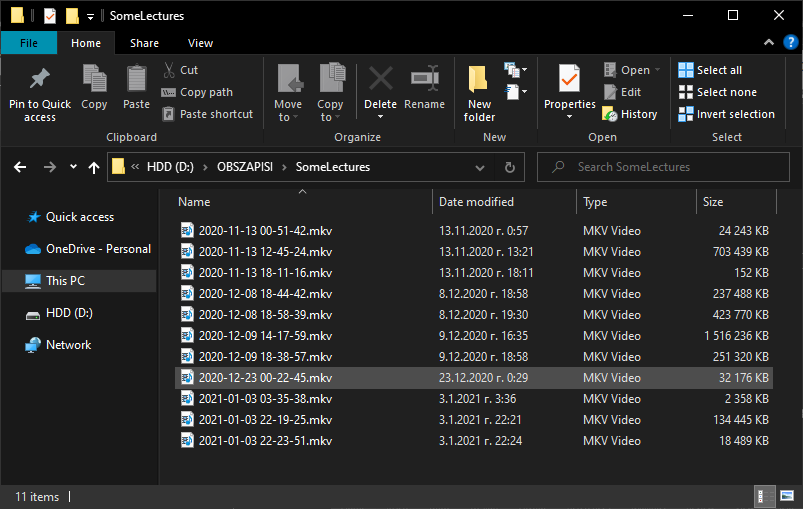
2.3.5. Грешки при архивиране на файловете

2.3.6. Грешки при преименуването на файловете

2.4. Генерално изпълнение на програмата

1. Код
2. **Действие с програмата**

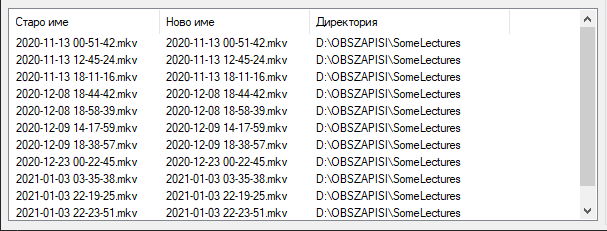
Да предположим, че потребител има голямо количество видео файлове за преименуване.



* 1. **. Избор на файлове**

При стартиране, програмата отваря потребителската папка (в повечето случай C:\Users\Потребител). Потребителя може да отиде до папката, където са файловете за редакция, като отваря папките вътре в изгледа, или директно въведе директорията горе.

След това, потребителят маркира всички файлове, които ще редактира и натиска на бутона Добави.

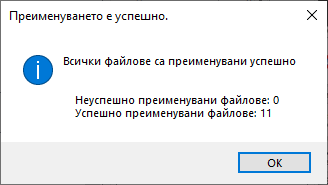
Всички файлове се появяват в списъка.

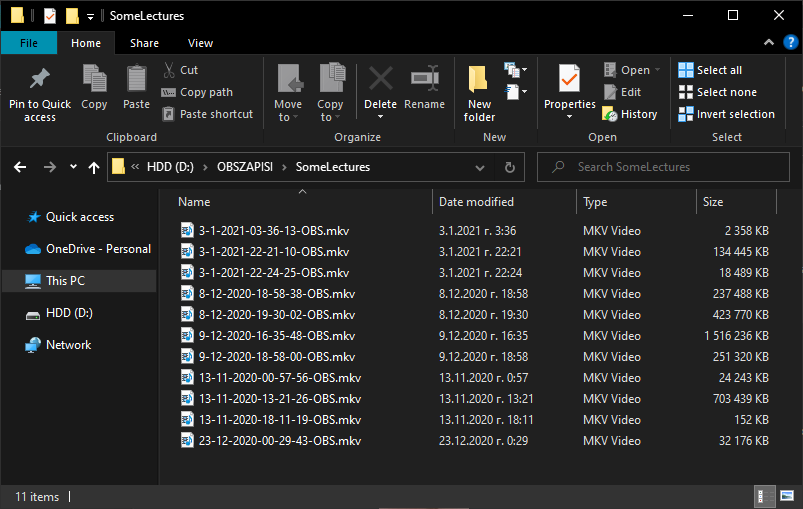
Потребителят решава да премахне старото име и да сложи ново име OBS, както и да добави дата. Маркира опциите „Съкращаване на името“, „Изтрий всичко“, „Добавяне на ново име“ и „Добавяне на дата“ от опциите за редактиране встрани. Това създава няколко дубликата и програмата не позволява предприемането на преименуване. Потребителят ще трябва да добави още някой елемент, за да направи имената различни.

Ако потребителят добави и час от менюто встрани, вече ще може да преименува файловете. Обаче имената ще бъдат нечетливи и преименуването става безсмислено.

Потребителят избира опцията „Добави разделител след всеки елемент“. Сега името е по-четливо.

Потребителят решава да премести новото име да отиде накрая. Това го прави като избира новото име от приоритетите и натиска на бутона надолу, докато не отиде до последното място. Потребителят започва преименуването като натиска на бутона „Започни преименуването“.

   
Всичко е готово…

  
И файловете са преименувани

1. **Логика на програмата**

**2.1. Главни променливи**

Клас FileBlock – Специално предназначен клас за по-лесно управление на избраните файлове, съдържа свойствата:  
- string FileName – оригиналното име на файла  
- string NewFileName – новото, редактирано име на файла  
- string Location – Показва директорията, в която се намира файлът  
- DateTime DateCreated – датата, на която файлът е модифициран

List<FileBlock> selectedFiles – Списък с файловете за редактиране

string userroot – потребителската директория, изчислена от ОС

string currentdir – променлива за директорията, която listViewFilesAndFolders показва

bool direrror – Използва се като флаг за неуспешно отворена директория.

**2.2. Функции в програмата**

**2.2.1. CreateHeadersAndFillListView()** – Тази функция има за цел да генерира различните детайли за всеки елемент. В този случай, генерира 3 колони –име на файла, размер на файла и дата на модифициране.  
  
Код за CreateHeadersAndFillListView():

private void CreateHeadersAndFillListView()

{

ColumnHeader colHead;

colHead = new ColumnHeader();

colHead.Text = "Име";

colHead.Width = 200;

this.listViewFilesAndFolders.Columns.Add(colHead);

colHead = new ColumnHeader();

colHead.Text = "Размер";

colHead.Width = 100;

this.listViewFilesAndFolders.Columns.Add(colHead);

colHead = new ColumnHeader();

colHead.Text = "Модифициран";

colHead.Width = 200;

this.listViewFilesAndFolders.Columns.Add(colHead);

}

**2.2.2. MakeHradersSelectedFilesList()** – Аналог на CreateHeadersAndFillListView(). Тук колоните са старото име на файла, новото име на файла и директорията, в която файлът се намира.

Код за MakeHradersSelectedFilesList():

private void MakeHeadersSelectedFilesList()

{

ColumnHeader colHead;

colHead = new ColumnHeader();

colHead.Text = "Старо име";

colHead.Width = 150;

this.listViewSelectedFiles.Columns.Add(colHead);

colHead = new ColumnHeader();

colHead.Text = "Ново име";

colHead.Width = 150;

this.listViewSelectedFiles.Columns.Add(colHead);

colHead = new ColumnHeader();

colHead.Text = "Директория";

colHead.Width = 200;

this.listViewSelectedFiles.Columns.Add(colHead);

}

**2.2.3. PaintListView(string root)** – Тази функция запълва елемента listViewFilesAndFolders с елементи от директорията, посочена от променливата root

Код за PaintListView(string root):  
private void PaintListView(string root)

{

listViewFilesAndFolders.SmallImageList = imageListSmall;

direrror = false;

try

{

ListViewItem lvi;

ListViewItem.ListViewSubItem lvsi;

if (root.CompareTo("") == 0) return;

DirectoryInfo dir = new DirectoryInfo(root);

DirectoryInfo[] dirs = dir.GetDirectories();

FileInfo[] files = dir.GetFiles();

this.listViewFilesAndFolders.Items.Clear();

this.textBoxCurDir.Text = root;

this.listViewFilesAndFolders.BeginUpdate();

foreach (DirectoryInfo direc in dirs)

{

lvi = new ListViewItem();

lvi.Text = direc.Name;

lvi.ImageIndex = 0;

lvi.Tag = direc.FullName;

lvsi = new ListViewItem.ListViewSubItem();

lvsi.Text = "";

lvi.SubItems.Add(lvsi);

lvsi = new ListViewItem.ListViewSubItem();

lvsi.Text = direc.LastAccessTime.ToString();

lvi.SubItems.Add(lvsi);

this.listViewFilesAndFolders.Items.Add(lvi);

}

foreach (FileInfo file in files)

{

lvi = new ListViewItem();

lvi.Text = file.Name;

lvi.ImageIndex = 1;

lvi.Tag = file.FullName;

lvsi = new ListViewItem.ListViewSubItem();

string size = file.Length.ToString();

lvsi.Text = ConvertToReadableBytes(size); ;

lvi.SubItems.Add(lvsi);

lvsi = new ListViewItem.ListViewSubItem();

lvsi.Text = file.LastWriteTime.ToString();

lvi.SubItems.Add(lvsi);

this.listViewFilesAndFolders.Items.Add(lvi);

lvsi = new ListViewItem.ListViewSubItem();

lvsi.Text = file.DirectoryName;

lvi.SubItems.Add(lvsi);

}

this.listViewFilesAndFolders.EndUpdate();

}

catch (System.Exception err)

{

WriteToLog("Грешка в програмата! " + err.Message);

MessageBox.Show("Грешка в програмата! " + err.Message, "Грешка в програмата!", MessageBoxButtons.OK, MessageBoxIcon.Error);

labelErrorReport.Text = "Грешка в програмата! " + err.Message;

direrror = true;

}

}

**2.2.4. PaintSelectedFilesView()** – Аналог на PaintListView(). Тук обаче запълва елемента listViewSelectedFiles с данни от листа selectedFiles.

Код за PaintSelectedFilesView():  
private void PaintSelectedFilesView()

{

try

{

ListViewItem lvi;

ListViewItem.ListViewSubItem lvsi;

this.listViewSelectedFiles.Items.Clear();

this.listViewSelectedFiles.BeginUpdate();

foreach (FileBlock file in selectedFiles)

{

lvi = new ListViewItem();

lvi.Text = file.FileName;

lvi.ImageIndex = 1;

lvi.Tag = file.Location + file.FileName;

lvsi = new ListViewItem.ListViewSubItem();

lvsi.Text = file.NewFileName;

lvi.SubItems.Add(lvsi);

lvsi = new ListViewItem.ListViewSubItem();

lvsi.Text = file.Location;

lvi.SubItems.Add(lvsi);

this.listViewSelectedFiles.Items.Add(lvi);

}

this.listViewSelectedFiles.EndUpdate();

}

catch (System.Exception err)

{

WriteToLog("Грешка в програмата! " + err.Message);

MessageBox.Show("Грешка в програмата! " + err.Message, "Грешка в програмата!", MessageBoxButtons.OK, MessageBoxIcon.Error);

labelErrorReport.Text = "Грешка в програмата! " + err.Message;

}

}

**2.2.5. UpdateNewNames()** – Тази функция е извиквана всеки път, когато се добави или премахне файл, и когато се избере или промени дадена опция за конструиране на името. Целта на тази функция е динамично да обновява новото име за избраните файлове в листа selectedFiles.

Код за UpdateNewNames():  
private void UpdateNewNames()

{

try

{

decimal counter = numericUpDownNumStart.Value;

//Файл от списъка с избрани файлове

foreach (FileBlock item in selectedFiles)

{

//опит за запазване на формата

string[] oldfullname = item.FileName.ToString().Split('.');

int nameparts = oldfullname.Length;

//конструиране на старото име

string oldname = "";

for (int i = 0; i < nameparts - 1; i++)

{

if (i != 0) oldname += ".";

oldname += oldfullname[i];

}

//стария формат

string oldextention = "." + oldfullname[nameparts - 1];

string newname = "";

//начало на действие по приоритет

foreach (string action in this.listBoxNamePriorities.Items)

{

//Добавяне на разделител

if (newname != "" && this.checkBoxDividerAdd.Checked)

{

if (comboBoxDivider.SelectedIndex == 0) newname += "-";

if (comboBoxDivider.SelectedIndex == 1) newname += " ";

if (comboBoxDivider.SelectedIndex == 2) newname += "\_";

}

//Съкращаване или оставяне на старото име

if (action == "Старо име")

{

if (checkBoxDeleteLastSymbols.Checked)

{

for (int i = 0; i < numericUpDownSymDelNum.Value; i++)

{

if (i <= oldname.Length - 1) newname += oldname[i];

}

}

else newname = oldname;

}

//Добавяне на номерация

else if (action == "Номер")

{

for (int i = counter.ToString().Length; i < numericUpDownNumZeroes.Value; i++)

{

newname += "0";

}

newname += counter;

}

//Добавяне на дата

else if (action == "Дата")

{

newname += item.DateCreated.Day.ToString() + "-" + item.DateCreated.Month.ToString() + "-" + item.DateCreated.Year.ToString();

}

//Добавяне на час

else if (action == "Час")

{

string hour;

string minute;

string second;

if (item.DateCreated.Hour < 10) hour = "0" + item.DateCreated.Hour.ToString();

else hour = item.DateCreated.Hour.ToString();

if (item.DateCreated.Minute < 10) minute = "0" + item.DateCreated.Minute.ToString();

else minute = item.DateCreated.Minute.ToString();

if (item.DateCreated.Second < 10) second = "0" + item.DateCreated.Second.ToString();

else second = item.DateCreated.Second.ToString();

newname += hour + "-" + minute + "-" + second;

}

//Добавяне на дата

else if (action == "Ново име") newname += textBoxAddName.Text;

}

//смяна на формата (без конвертиране)

if (checkBoxExtChange.Checked)

{

string newextention = textBoxExtChange.Text;

if (newextention != "") newextention = "." + newextention;

item.NewFileName = newname + newextention;

}

else

{

item.NewFileName = newname + oldextention;

}

counter++;

}

//Проверка за дублирани файлове и валидиране

var keyscheck = new HashSet<string>();

foreach (FileBlock file in selectedFiles)

{

if (!keyscheck.Add(file.NewFileName + "," + file.Location))

{

buttonStartRename.Enabled = false;

labelErrorReport.Text = "Намерени потенциални дубликати! Моля, сменете метода на преименуване!";

break;

}

labelErrorReport.Text = "";

buttonStartRename.Enabled = true;

}

PaintSelectedFilesView();

}

catch (Exception err)

{

WriteToLog("Грешка в програмата! " + err.Message);

MessageBox.Show("Грешка в програмата! " + err.Message, "Грешка в програмата!", MessageBoxButtons.OK, MessageBoxIcon.Error);

labelErrorReport.Text = "Грешка :" + err.Message;

}

}

**2.2.6. BeginRenaming()** – Това е алгоритъмът отговорен за преименуването на всички файлове. Започва с искане за потвърждение от потребителя. При потвърждаване се започва с архивиране на файловете, ако е избрано, и преименуването.

Код за buttonStartRename\_Click(object sender, EventArgs e):  
private void BeginRenaming()

{

//Потвърждение от потребителя

DialogResult result = MessageBox.Show("ВНИМАНИЕ?\r\n\nАко сте включили архивирането преди преименуване, съществуващите файлове може да бъдат презаписани.\r\n\nСигурни ли сте?", "ВНИМАНИЕ!", MessageBoxButtons.YesNo, MessageBoxIcon.Warning, MessageBoxDefaultButton.Button2);

//Ако потвърди

if (result == DialogResult.Yes)

{

try

{

int errcount = 0; // брояч на грешки

//Архивиране на файловете

if (checkBoxBackupFiles.Checked)

{

//безкраен цикъл за нов опит

while (true)

{

WriteToLog("----Започване на бекъп преди преименуване...---- ");

foreach (FileBlock file in selectedFiles)

{

try

{

string fullfilename = file.Location + @"\" + file.FileName; //Взима се пълната локация

System.IO.FileInfo filebuffer = new FileInfo(fullfilename); //Взима се информация за файла

string backupdir = filebuffer.DirectoryName + "\\BitRenamer-Backup\\" + filebuffer.Name; //Създава се дестинация

Directory.CreateDirectory($@"{filebuffer.Directory}\BitRenamer-backup"); // Създава папка, ако такава липсва

filebuffer.CopyTo(backupdir, true); //Копиране

}

catch (Exception err)

{

WriteToLog($"Файлът {file.FileName} не бе архивиран! {err.Message}");

errcount++;

}

}

//Избор за следващ опит, позволява на потребителя да затвори някой файл, ако го е забравил отворен в програма

if (errcount != 0)

{

string dialogtextRetry = "Някои файлове не бяха успешно архивирани.\r\nТова е възможно, ако някои от избраните файлове се използват в момента или са недостъпни.\r\nБихте ли опитали отново? Можете да продължите с преименуването, като натиенете Игнорирай.";

string dialogtitleRetry = "ВНИМАНИЕ!";

DialogResult resultretry = MessageBox.Show(dialogtextRetry, dialogtitleRetry, MessageBoxButtons.AbortRetryIgnore, MessageBoxIcon.Question);

if (resultretry == DialogResult.Abort) throw new Exception("Отказ от потребителя след неуспешно архивиране.");//отказ

if (resultretry == DialogResult.Ignore) break;//игнориране на предупреждението

// Опитва отново, ако се натисне retry

}

else break; //успешен бекъп

}

}

//започване на преименуването

errcount = 0; //ресетва се брояча

WriteToLog("----Започване на преименуването...---- ");

foreach (FileBlock file in selectedFiles)

{

try

{

string fullfilename = file.Location + @"\" + file.FileName; //Взима се пълната локация

System.IO.FileInfo filebuffer = new FileInfo(fullfilename); //Взима се инфомация за файла

string newfullfilename = filebuffer.Directory + "\\" + file.NewFileName; //Прави се дестинация

filebuffer.MoveTo(newfullfilename);//Преименуваме файла

}

//запис в лога при неуспех

catch (Exception err)

{

WriteToLog($"Файлът {file.FileName} не бе преименуван! {err.Message}");

errcount++;

}

}

//Частичен успех

if (errcount != 0)

{

string dialogtextReady = "Някои файлове не бяха успешно преименувани.\r\n" +

"Това е възможно, ако някои от избраните файлове се използват в момента, или са недостъпни.\r\n" +

"Проверете лога за повече подробности\r\n\n\n" +

$" Неуспешно преименувани файлове: {errcount}\r\n" +

$" Успешно преименувани файлове: {selectedFiles.Count - errcount}";

string dialogtitleReady = "Преименуването е частично успешно.";

MessageBox.Show(dialogtextReady, dialogtitleReady, MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

}

//Успешно преименуване

else

{

string dialogtextReady =

"Всички файлове са преименувани успешно\r\n\n\n" +

$" Неуспешно преименувани файлове: {errcount}\r\n" +

$" Успешно преименувани файлове: {selectedFiles.Count - errcount}";

string dialogtitleReady = "Преименуването е успешно.";

MessageBox.Show(dialogtextReady, dialogtitleReady, MessageBoxButtons.OK, MessageBoxIcon.Asterisk);

}

//При частичен успех или пълен успех, списъка се изчиства и се обновява изгледа с папките

listViewSelectedFiles.Items.Clear();

selectedFiles.Clear();

PaintListView(currentdir);

}

//Запис в лога при критична грешка, било то отказ от потребителя или грешка от програмата

catch (Exception err)

{

WriteToLog($"Преименуването е неуспешно! {err.Message}");

MessageBox.Show("Преименуването е неуспешно! Проверете лога за повече подробности. \r\nГрешка: " + err.Message, "Преименуването e неуспешно!", MessageBoxButtons.OK, MessageBoxIcon.Error);

labelErrorReport.Text = "Преименуването е неуспешно! Проверете лога за подробности!";

}

}

}

**2.2.7. WriteToLog(string message)** – Тази функция записва съобщението message с часа, в което това съобщение е пуснато. Всичко се записва във файла BitRenamerlog.txt, който се намира на същото място, където е и програмата. Използва се за записване на всички грешки и най-вече неуспешно архивирани и преименувани файлове.

Код за WriteToLog(string message):  
 private void WriteToLog(string message)

{

try

{

string logline = DateTime.Now.ToString() + " " + message + "\r\n";

string programpath = Path.GetDirectoryName(Assembly.GetExecutingAssembly().Location);

File.AppendAllText(programpath + "\\BitRenamerlog.txt", logline);

}

catch (Exception err)

{

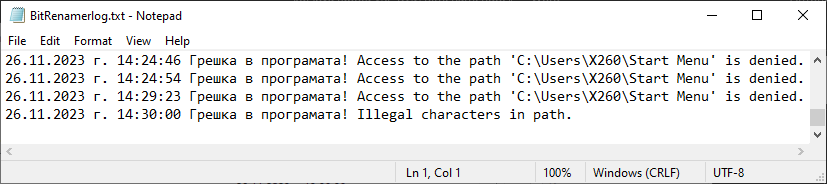
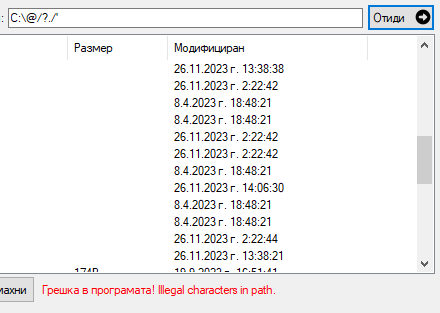
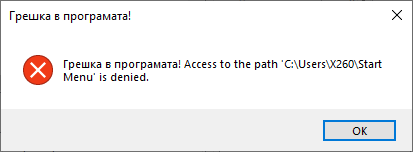
MessageBox.Show("Неуспешен запис в лога! \r\nГрешка:" + err.Message, "Неуспешен запис в лога!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

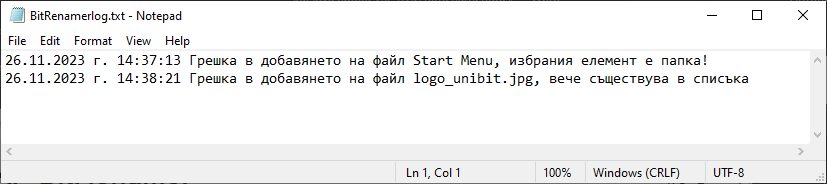
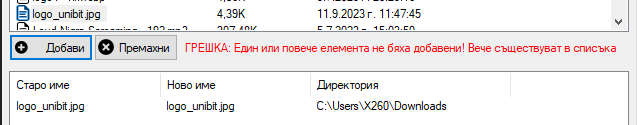
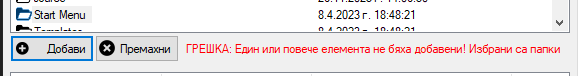
}

**2.3.** **Проверки, валидация и обработка на грешки в програмата**  
За да работи програмата безотказно, без нежелани ефекти, се използва валидация, проверка и обработка на грешката.

**2.3.1. Обработка на грешки при навигиране на папките**  
Когато потребител въведе невалидна директория в textBoxCurDir или се появи грешка при отварянето/извеждането на директорията(например забранен достъп), то тя бива обработена и показана като съобщение на отделен прозорец и до бутоните за добавяне/премахване, както и се записва в лог файла на програмата.



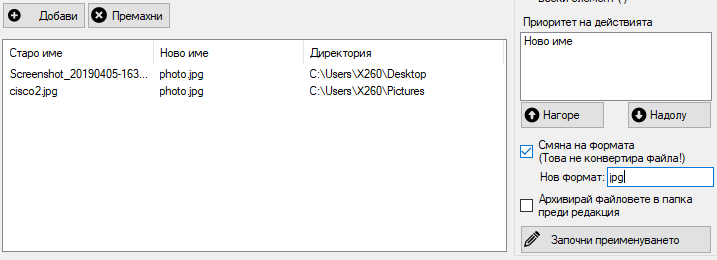
**2.3.2. Обработка на грешки при добавяне на файлове в списъка**   
В програмата може да се появи грешка при зареждане на информация от файла, или при добавяне на вече съществуващ файл в списъка или папка. При добавяне на съществуващ файл в списъка или папка, се изписва грешката до бутоните за добавяне и премахване на файл, и се записва в лога.

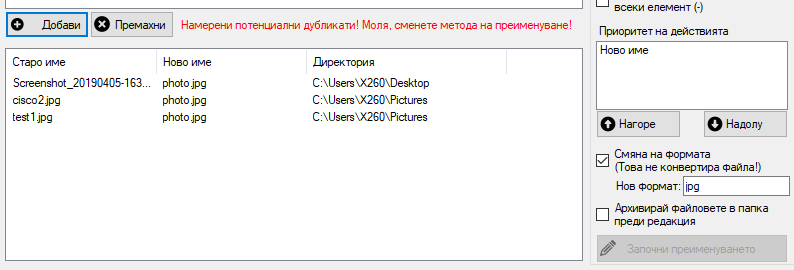


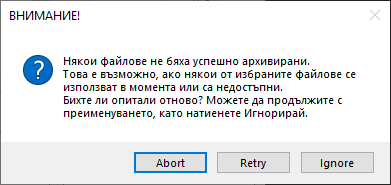
**2.3.3. Ограничаване на входните данни за създаване на името**  
За да не се допусне въвеждане на неправилни данни, които могат да причинят проблеми при преименуването на файла, са вложени следните ограничения при създаването на ново име на файла.

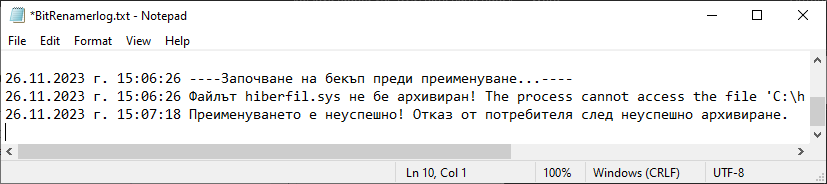
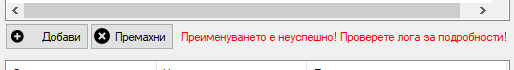
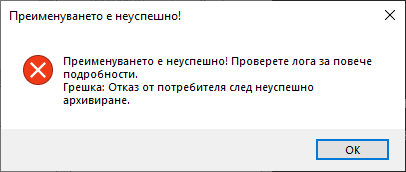
- Съкращаване на първите знаци (numericUpDownSymDelNum): 0-256  
- Ново име (textBoxAddName): разрешени са само букви, цифри, точка, интервал, долна черта, тире и е позволено изтриването с Backspace  
- Брой нули при номерация (numericUpDownNumZeroes): 0-6  
- Начално число за номерация (numericUpDownNumStart): 0-999999  
- Нов формат на файла (textBoxExtChange): разрешени са само букви, цифри, тире, долна черта и изтриване с Backspace

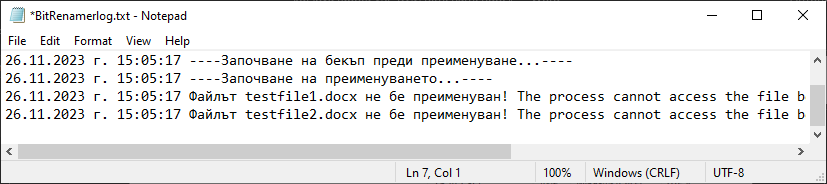
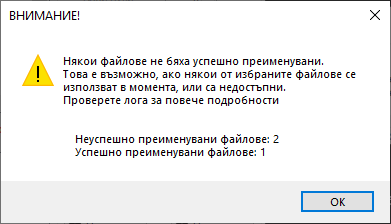
**2.3.4. Проверка за потенциални дубликати след редакция.**Програмата използва HashSet, за да прецени дали има дубликати. След като всички нови имена за избраните файлове са сглобени, се прави цикъл, в който се добавят в HashSet новото име на файла и неговата директория слепени. Ако се появи грешка в добавянето(тя става когато вече има такъв Hash в HashSet-а), се изписва грешка до бутоните за добавяне и премахване и се блокира бутона за старт на преименуването. Ако не намери дубликати, то бутонът се освобождава, ако е бил блокиран преди това. Опресняването става при всяко добавяне/премахване на файл, както и при добавяне и смяна на елемент от името.





**2.3.5. Грешки при архивиране на файловете**  
Възможно е, когато се направи опит да се архивира файл, да се появи грешка, било то заради вече несъществуващ файл или забранен достъп, най-често, защото файлът е отворен в друга програма. Когато в процеса на архивиране такъв файл е намерен, се добавя грешката в лога и се вдига броячът errcount. След това, ако има грешки, изскача съобщение, което подканва потребителя да реши дали да продължи с редактирането, дали да опита отново или да се откаже.   


При нов опит, цикълът продължава да се изпълнява, при игнориране се излиза от цикъла, а при отказ на редакцията, се хвърля Exception “Отказ от потребителя” и се спира процеса.  


**2.3.6. Грешки при преименуването на файловете**  
Също както при архивирането, е възможно да се появи грешка, когато файлът е недостъпен. В този случай, грешката се записва отново в лога. Накрая ще се изпише различно съобщение от стандартното и ще се покаже броят неуспешни файлове.  


**2.4. Генерално изпълнение на програмата**

Със стартирането на програмата се зареждат променливите за цялата програма.  
public partial class Form1 : Form

{

private List<FileBlock> selectedFiles = new List<FileBlock>(); // Лист с избраните файлове

private System.Collections.Specialized.StringCollection folderCol;

private string userroot = Environment.GetFolderPath(Environment.SpecialFolder.UserProfile); //Директорията на потребителя

string currentdir; //Променлива за сегашната директория

bool direrror = false; //Променлива при грешка с директорията

След това се зарежда формата, задават се настройките по подразбиране и се зарежда потребителската директория в listViewFilesAndFolders.  
 public Form1()

{

InitializeComponent();

listBoxNamePriorities.Items.Add("Старо име");

comboBoxDivider.SelectedIndex = 0;

folderCol = new System.Collections.Specialized.StringCollection();

CreateHeadersAndFillListView();

MakeHeadersSelectedFilesList();

folderCol.Add(@"C:\");

folderCol.Add(@"C:\Users");

currentdir = userroot;

PaintListView(userroot);

folderCol.Add(userroot);

PaintSelectedFilesView();

}

Потребителят може да отвори папка вътре в елемента listViewFilesAndFolders и това става с действието listViewFilesAndFolders\_ItemsActivate

private void listViewFilesAndFolders\_ItemActivate(object sender, EventArgs e)

{

System.Windows.Forms.ListView lw = (System.Windows.Forms.ListView)sender;

string filename = lw.SelectedItems[0].Tag.ToString();

if (lw.SelectedItems[0].ImageIndex != 0)

{

try

{

System.Diagnostics.Process.Start(filename);

}

catch

{

return;

}

}

else

{

PaintListView(filename);

if (!direrror)

{

folderCol.Add(filename);

currentdir = filename;

}

}

}

След като потребителят избере няколко файла, той трябва да натисне бутона „Добави“, който изпълнява тази функция, която добавя файловете в листата selectedFiles, пълна с елементи от тип FileBlock.  
private void buttonAddFile\_Click(object sender, EventArgs e)

{

labelErrorReport.Text = "";

try

{

foreach (System.Windows.Forms.ListViewItem file in listViewFilesAndFolders.SelectedItems)

{

FileInfo fileInfoBuffer = new FileInfo(file.Tag.ToString());

if (file.ImageIndex != 0)

{

FileBlock filebuffer = new FileBlock(fileInfoBuffer.Name, fileInfoBuffer.Directory.ToString(), fileInfoBuffer.LastWriteTime);

//при празен списък

if (selectedFiles.Count == 0)

{

selectedFiles.Add(filebuffer);

}

else

{

int duplicate = 0;

foreach (FileBlock filechecked in selectedFiles.ToList())

{

//файлът вече е в списъка

if (filebuffer.FileName == filechecked.FileName && filebuffer.Location == filechecked.Location)

{

WriteToLog($"Грешка в добавянето на файл {filebuffer.FileName}, вече съществува в списъка");

labelErrorReport.Text = "ГРЕШКА: Един или повече елемента не бяха добавени! Вече съществуват в списъка";

duplicate++;

}

}

//ако не е намерен дубликат, файла се добавя

if (duplicate == 0) selectedFiles.Add(filebuffer);

}

}

//избрана е папка и се принтира грешка до бутоните

else

{

WriteToLog($"Грешка в добавянето на файл {file.Text}, избрания елемент е папка!");

labelErrorReport.Text = "ГРЕШКА: Един или повече елемента не бяха добавени! Избрани са папки";

}

}

UpdateNewNames();

PaintSelectedFilesView();

}

catch (System.Exception err)

{

WriteToLog("Грешка при добавянето на файлове! " + err.Message);

MessageBox.Show("Грешка при добавянето на файлове! " + err.Message, "Грешка при добавянето на файлове!", MessageBoxButtons.OK, MessageBoxIcon.Error);

labelErrorReport.Text = "Грешка при добавянето на файлове! " + err.Message;

}

}

След това, ако иска да премахне файл, той трябва да избере файловете от listViewSelectedFiles и да посочи бутона „Премахни“.  
private void buttonRemoveFile\_Click(object sender, EventArgs e)

{

try

{

labelErrorReport.Text = "";

//ако са избрани повече от един файл, се добавя променливата adjustйment, тъй като може да се объркат индексите

int adjustment = 0;

foreach (System.Windows.Forms.ListViewItem file in listViewSelectedFiles.SelectedItems)

{

selectedFiles.RemoveAt(file.Index - adjustment);

adjustment++;

}

PaintSelectedFilesView();

UpdateNewNames();

}

catch (Exception err)

{

WriteToLog("Грешка в програмата! " + err.Message);

labelErrorReport.Text = "Грешка: " + err.Message;

}

}

Когато е готов с избора на файловете, той може да избере опциите, които се намират встрани, когато маркира някоя опция с тикче, в зависимост от какво предлага опцията, определените елементи се включват и се добавят в списъка с приоритети listBoxNamePriorities. Даден е примера с опцията „Съкращаване на името“. За останалите, действията са приблизително същите и могат да се видят в кода по-долу.  
private void checkBoxDeleteLastSymbols\_CheckedChanged(object sender, EventArgs e)

{

if (checkBoxDeleteLastSymbols.Checked && checkBoxDeleteAllSymbols.Checked)

{

numericUpDownSymDelNum.Enabled = false;

labelSymDelNum.Enabled = false;

checkBoxDeleteAllSymbols.Enabled = true;

listBoxNamePriorities.Items.Remove("Старо име");

}

else if (checkBoxDeleteLastSymbols.Checked && !checkBoxDeleteAllSymbols.Checked)

{

numericUpDownSymDelNum.Enabled = true;

labelSymDelNum.Enabled = true;

checkBoxDeleteAllSymbols.Enabled = true;

}

else

{

numericUpDownSymDelNum.Enabled = false;

labelSymDelNum.Enabled = false;

checkBoxDeleteAllSymbols.Enabled = false;

if (checkBoxDeleteAllSymbols.Checked) listBoxNamePriorities.Items.Add("Старо име");

}

UpdateNewNames();

}

private void numericUpDownSymDelNum\_ValueChanged(object sender, EventArgs e)

{

UpdateNewNames();

}

Когато се въвеждат данни в опциите „Ново име“ и „Смяна на формат“, има проверка за въведения символ. Показан е примера за функцията „Ново име“. За другата функция е почти същото, само че с по-малко приети символи.  
private void textBoxAddName\_KeyPress(object sender, KeyPressEventArgs e)

{

if (((e.KeyChar < 48 || (e.KeyChar > 57 && e.KeyChar < 65) || (e.KeyChar > 90 & e.KeyChar < 97) || e.KeyChar > 122)) && e.KeyChar != 8 && e.KeyChar != 32 && e.KeyChar != 45 && e.KeyChar != 46 && e.KeyChar != 95) e.Handled = true;

}

Програмата позволява и смяна на приоритета на различните елементи, разместването е съвсем лесно с този код. Разликата между двете действия е знакът при смяна на индекса.  
private void buttonPriorityUp\_Click(object sender, EventArgs e)

{

try

{

//ако не е избрано нищо, не изпълнява кода, предотвратява грешки

if (listBoxNamePriorities.SelectedItem != null)

{

int oldindex = listBoxNamePriorities.SelectedIndex;

string item = listBoxNamePriorities.SelectedItem.ToString();

if (oldindex - 1 >= 0)

{

listBoxNamePriorities.Items.RemoveAt(oldindex);

listBoxNamePriorities.Items.Insert(oldindex - 1, item);

listBoxNamePriorities.SelectedIndex = oldindex - 1;

}

}

UpdateNewNames();

}

catch (Exception err)

{

WriteToLog("Грешка в програмата! " + err.Message);

labelErrorReport.Text = "Грешка: " + err.Message;

}

}През цялото време, каквото и да се променя, се извиква функцията UpdateNewNames, която е описана по-горе. Ако се намерят потенциални дубликати, функцията изписва грешка и блокира възможността за преименуване на файловете. Когато всичко е готово и се натисне бутона „Започни преименуването“, започва процеса по архивиране и преименуване. Извиква се функцията BeginRenaming(), която е описана по-горе.

След приключване, списъкът с избрани файлове се изчиства и се обновява директорията, в която се намира в момента.

**3. Код**

Главен код(Form1.cs):  
using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Reflection;

using System.Windows.Forms;

namespace VSP\_135KNZ\_MyProject

{

//AP BITRENAMER

//От Антонио Петков, 2. курс, специалност КН, Фак. Ном. 135КНЗ

//Тази програма използва класа FileBlock.cs

//Повече информация вижте в документа към програмата

public partial class Form1 : Form

{

private List<FileBlock> selectedFiles = new List<FileBlock>(); // Лист с избраните файлове

private System.Collections.Specialized.StringCollection folderCol;

private string userroot = Environment.GetFolderPath(Environment.SpecialFolder.UserProfile); //Директорията на потребителя

string currentdir; //Променлива за сегашната директория

bool direrror = false; //Променлива при грешка с директорията

public Form1()

{

InitializeComponent();

listBoxNamePriorities.Items.Add("Старо име");

comboBoxDivider.SelectedIndex = 0;

folderCol = new System.Collections.Specialized.StringCollection();

CreateHeadersAndFillListView();

MakeHeadersSelectedFilesList();

folderCol.Add(@"C:\");

folderCol.Add(@"C:\Users");

currentdir = userroot;

PaintListView(userroot);

folderCol.Add(userroot);

PaintSelectedFilesView();

}

//действия по обновяването на интерфейса и валидация

//Съкращаване на името

private void checkBoxDeleteLastSymbols\_CheckedChanged(object sender, EventArgs e)

{

if (checkBoxDeleteLastSymbols.Checked && checkBoxDeleteAllSymbols.Checked)

{

numericUpDownSymDelNum.Enabled = false;

labelSymDelNum.Enabled = false;

checkBoxDeleteAllSymbols.Enabled = true;

listBoxNamePriorities.Items.Remove("Старо име");

}

else if (checkBoxDeleteLastSymbols.Checked && !checkBoxDeleteAllSymbols.Checked)

{

numericUpDownSymDelNum.Enabled = true;

labelSymDelNum.Enabled = true;

checkBoxDeleteAllSymbols.Enabled = true;

}

else

{

numericUpDownSymDelNum.Enabled = false;

labelSymDelNum.Enabled = false;

checkBoxDeleteAllSymbols.Enabled = false;

if (checkBoxDeleteAllSymbols.Checked) listBoxNamePriorities.Items.Add("Старо име");

}

UpdateNewNames();

}

//Изтриване на цялото име(без формата)

private void checkBoxDeleteAllSymbols\_CheckedChanged(object sender, EventArgs e)

{

if (checkBoxDeleteAllSymbols.Checked)

{

numericUpDownSymDelNum.Enabled = false;

labelSymDelNum.Enabled = false;

listBoxNamePriorities.Items.Remove("Старо име");

}

else

{

numericUpDownSymDelNum.Enabled = true;

labelSymDelNum.Enabled = true;

listBoxNamePriorities.Items.Add("Старо име");

}

UpdateNewNames();

}

//Добавяне на ново име

private void checkBoxAddName\_CheckedChanged(object sender, EventArgs e)

{

if (checkBoxAddName.Checked)

{

textBoxAddName.Enabled = true;

listBoxNamePriorities.Items.Add("Ново име");

}

else

{

textBoxAddName.Enabled = false;

listBoxNamePriorities.Items.Remove("Ново име");

}

UpdateNewNames();

}

//Добавяне на номерация в името

private void checkBoxAddNum\_CheckedChanged(object sender, EventArgs e)

{

if (checkBoxAddNum.Checked)

{

numericUpDownNumZeroes.Enabled = true;

labelNumZeroes.Enabled = true;

numericUpDownNumStart.Enabled = true;

labelNumStart.Enabled = true;

listBoxNamePriorities.Items.Add("Номер");

}

else

{

numericUpDownNumZeroes.Enabled = false;

labelNumZeroes.Enabled = false;

numericUpDownNumStart.Enabled = false;

labelNumStart.Enabled = false;

listBoxNamePriorities.Items.Remove("Номер");

}

UpdateNewNames();

}

//Добавяне на дата към името

private void checkBoxAddDate\_CheckedChanged(object sender, EventArgs e)

{

if (checkBoxAddDate.Checked)

{

listBoxNamePriorities.Items.Add("Дата");

}

else

{

listBoxNamePriorities.Items.Remove("Дата");

}

UpdateNewNames();

}

//Добавяне на час към името

private void checkBoxAddTime\_CheckedChanged(object sender, EventArgs e)

{

if (checkBoxAddTime.Checked)

{

listBoxNamePriorities.Items.Add("Час");

}

else

{

listBoxNamePriorities.Items.Remove("Час");

}

UpdateNewNames();

}

//Смяна на формата(без конвертиране)

private void checkBoxExtChange\_CheckedChanged(object sender, EventArgs e)

{

if (checkBoxExtChange.Checked)

{

textBoxExtChange.Enabled = true;

labelExtChange.Enabled = true;

}

else

{

textBoxExtChange.Enabled = false;

labelExtChange.Enabled = false;

}

UpdateNewNames();

}

//Добавяне на разделител между елементите

private void checkBoxDividerAdd\_CheckedChanged(object sender, EventArgs e)

{

if (checkBoxDividerAdd.Checked) comboBoxDivider.Enabled = true;

else comboBoxDivider.Enabled = false;

UpdateNewNames();

}

//Отваря формата 'Относно'

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

{

FormAbout form = new FormAbout();

form.ShowDialog();

}

//Действия на програмата

//Отваря избрания файл или папка, ако е папка, тя се отваря в програмата

private void listViewFilesAndFolders\_ItemActivate(object sender, EventArgs e)

{

System.Windows.Forms.ListView lw = (System.Windows.Forms.ListView)sender;

string filename = lw.SelectedItems[0].Tag.ToString();

if (lw.SelectedItems[0].ImageIndex != 0)

{

try

{

System.Diagnostics.Process.Start(filename);

}

catch

{

return;

}

}

else

{

PaintListView(filename);

if (!direrror)

{

folderCol.Add(filename);

currentdir = filename;

}

}

}

// Променя имената на избраните файлове и ги записва в техните елементи от типа FileBlock

private void UpdateNewNames()

{

try

{

decimal counter = numericUpDownNumStart.Value;

//Файл от списъка с избрани файлове

foreach (FileBlock item in selectedFiles)

{

//опит за запазване на формата

string[] oldfullname = item.FileName.ToString().Split('.');

int nameparts = oldfullname.Length;

//конструиране на старото име

string oldname = "";

for (int i = 0; i < nameparts - 1; i++)

{

if (i != 0) oldname += ".";

oldname += oldfullname[i];

}

//стария формат

string oldextention = "." + oldfullname[nameparts - 1];

string newname = "";

//начало на действие по приоритет

foreach (string action in this.listBoxNamePriorities.Items)

{

//Добавяне на разделител

if (newname != "" && this.checkBoxDividerAdd.Checked)

{

if (comboBoxDivider.SelectedIndex == 0) newname += "-";

if (comboBoxDivider.SelectedIndex == 1) newname += " ";

if (comboBoxDivider.SelectedIndex == 2) newname += "\_";

}

//Съкращаване или оставяне на старото име

if (action == "Старо име")

{

if (checkBoxDeleteLastSymbols.Checked)

{

for (int i = 0; i < numericUpDownSymDelNum.Value; i++)

{

if (i <= oldname.Length - 1) newname += oldname[i];

}

}

else newname = oldname;

}

//Добавяне на номерация

else if (action == "Номер")

{

for (int i = counter.ToString().Length; i < numericUpDownNumZeroes.Value; i++)

{

newname += "0";

}

newname += counter;

}

//Добавяне на дата

else if (action == "Дата")

{

newname += item.DateCreated.Day.ToString() + "-" + item.DateCreated.Month.ToString() + "-" + item.DateCreated.Year.ToString();

}

//Добавяне на час

else if (action == "Час")

{

string hour;

string minute;

string second;

if (item.DateCreated.Hour < 10) hour = "0" + item.DateCreated.Hour.ToString();

else hour = item.DateCreated.Hour.ToString();

if (item.DateCreated.Minute < 10) minute = "0" + item.DateCreated.Minute.ToString();

else minute = item.DateCreated.Minute.ToString();

if (item.DateCreated.Second < 10) second = "0" + item.DateCreated.Second.ToString();

else second = item.DateCreated.Second.ToString();

newname += hour + "-" + minute + "-" + second;

}

//Добавяне на дата

else if (action == "Ново име") newname += textBoxAddName.Text;

}

//смяна на формата (без конвертиране)

if (checkBoxExtChange.Checked)

{

string newextention = textBoxExtChange.Text;

if (newextention != "") newextention = "." + newextention;

item.NewFileName = newname + newextention;

}

else

{

item.NewFileName = newname + oldextention;

}

counter++;

}

//Проверка за дублирани файлове и валидиране

var keyscheck = new HashSet<string>();

foreach (FileBlock file in selectedFiles)

{

if (!keyscheck.Add(file.NewFileName + "," + file.Location))

{

buttonStartRename.Enabled = false;

labelErrorReport.Text = "Намерени потенциални дубликати! Моля, сменете метода на преименуване!";

break;

}

labelErrorReport.Text = "";

buttonStartRename.Enabled = true;

}

PaintSelectedFilesView();

}

catch (Exception err)

{

WriteToLog("Грешка в програмата! " + err.Message);

MessageBox.Show("Грешка в програмата! " + err.Message, "Грешка в програмата!", MessageBoxButtons.OK, MessageBoxIcon.Error);

labelErrorReport.Text = "Грешка :" + err.Message;

}

}

//Връщане назад с една директория

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

{

if (folderCol.Count > 1)

{

PaintListView(folderCol[folderCol.Count - 2].ToString());

currentdir = folderCol[folderCol.Count - 2].ToString();

folderCol.RemoveAt(folderCol.Count - 1);

}

else

{

PaintListView(folderCol[0].ToString());

}

}

//Отиване на посочената директория, при грешка изписва грешката и не прави нищо

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

{

labelErrorReport.Text = "";

PaintListView(textBoxCurDir.Text);

currentdir = textBoxCurDir.Text;

folderCol.Add(textBoxCurDir.Text);

}

//Конвентира дадения файлов размер в четлив формат за потребителя, например 1048576 ще изведе 1M - 1 мегабайт

private string ConvertToReadableBytes(string size)

{

try

{

//взима размера на файла

double sizenum = double.Parse(size);

//гигабайт

if (sizenum > 1073741824)

{

sizenum /= 1073741824;

sizenum = Math.Round(sizenum, 2);

string result = sizenum.ToString() + "G";

return result;

}

//мегабайт

else if (sizenum > 1048576)

{

sizenum /= 1048576;

sizenum = Math.Round(sizenum, 2);

string result = sizenum.ToString() + "M";

return result;

}

//килобайт

else if (sizenum > 1024)

{

sizenum /= 1024;

sizenum = Math.Round(sizenum, 2);

string result = sizenum.ToString() + "K";

return result;

}

//байт

else

{

string result = sizenum.ToString() + "B";

return result;

}

}

//ако стане грешка, било то твърде голямо число или грешка в изчислението, то връща за размер N/A

catch

{

WriteToLog("Грешка в изчисляването на размера!");

return "N/A";

}

}

//Добавя се файла към списъка за преименуване, ако е папка или вече добавен, не се добавя и се изписва грешка до бутона

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

{

labelErrorReport.Text = "";

try

{

foreach (System.Windows.Forms.ListViewItem file in listViewFilesAndFolders.SelectedItems)

{

FileInfo fileInfoBuffer = new FileInfo(file.Tag.ToString());

if (file.ImageIndex != 0)

{

FileBlock filebuffer = new FileBlock(fileInfoBuffer.Name, fileInfoBuffer.Directory.ToString(), fileInfoBuffer.LastWriteTime);

//при празен списък

if (selectedFiles.Count == 0)

{

selectedFiles.Add(filebuffer);

}

else

{

int duplicate = 0;

foreach (FileBlock filechecked in selectedFiles.ToList())

{

//файлът вече е в списъка

if (filebuffer.FileName == filechecked.FileName && filebuffer.Location == filechecked.Location)

{

WriteToLog($"Грешка в добавянето на файл {filebuffer.FileName}, вече съществува в списъка");

labelErrorReport.Text = "ГРЕШКА: Един или повече елемента не бяха добавени! Вече съществуват в списъка";

duplicate++;

}

}

//ако не е намерен дубликат, файла се добавя

if (duplicate == 0) selectedFiles.Add(filebuffer);

}

}

//избрана е папка и се принтира грешка до бутоните

else

{

WriteToLog($"Грешка в добавянето на файл {file.Text}, избрания елемент е папка!");

labelErrorReport.Text = "ГРЕШКА: Един или повече елемента не бяха добавени! Избрани са папки";

}

}

UpdateNewNames();

PaintSelectedFilesView();

}

catch (System.Exception err)

{

WriteToLog("Грешка при добавянето на файлове! " + err.Message);

MessageBox.Show("Грешка при добавянето на файлове! " + err.Message, "Грешка при добавянето на файлове!", MessageBoxButtons.OK, MessageBoxIcon.Error);

labelErrorReport.Text = "Грешка при добавянето на файлове! " + err.Message;

}

}

//Създава изгледа на формата за файловия мениджър

private void CreateHeadersAndFillListView()

{

ColumnHeader colHead;

colHead = new ColumnHeader();

colHead.Text = "Име";

colHead.Width = 200;

this.listViewFilesAndFolders.Columns.Add(colHead);

colHead = new ColumnHeader();

colHead.Text = "Размер";

colHead.Width = 100;

this.listViewFilesAndFolders.Columns.Add(colHead);

colHead = new ColumnHeader();

colHead.Text = "Модифициран";

colHead.Width = 200;

this.listViewFilesAndFolders.Columns.Add(colHead);

}

//Създава изгледа за избраните файлове за преименуване

private void MakeHeadersSelectedFilesList()

{

ColumnHeader colHead;

colHead = new ColumnHeader();

colHead.Text = "Старо име";

colHead.Width = 150;

this.listViewSelectedFiles.Columns.Add(colHead);

colHead = new ColumnHeader();

colHead.Text = "Ново име";

colHead.Width = 150;

this.listViewSelectedFiles.Columns.Add(colHead);

colHead = new ColumnHeader();

colHead.Text = "Директория";

colHead.Width = 200;

this.listViewSelectedFiles.Columns.Add(colHead);

}

//Обновява ListView за файловия мениджър

private void PaintListView(string root)

{

listViewFilesAndFolders.SmallImageList = imageListSmall;

direrror = false;

try

{

ListViewItem lvi;

ListViewItem.ListViewSubItem lvsi;

if (root.CompareTo("") == 0) return;

DirectoryInfo dir = new DirectoryInfo(root);

DirectoryInfo[] dirs = dir.GetDirectories();

FileInfo[] files = dir.GetFiles();

this.listViewFilesAndFolders.Items.Clear();

this.textBoxCurDir.Text = root;

this.listViewFilesAndFolders.BeginUpdate();

foreach (DirectoryInfo direc in dirs)

{

lvi = new ListViewItem();

lvi.Text = direc.Name;

lvi.ImageIndex = 0;

lvi.Tag = direc.FullName;

lvsi = new ListViewItem.ListViewSubItem();

lvsi.Text = "";

lvi.SubItems.Add(lvsi);

lvsi = new ListViewItem.ListViewSubItem();

lvsi.Text = direc.LastAccessTime.ToString();

lvi.SubItems.Add(lvsi);

this.listViewFilesAndFolders.Items.Add(lvi);

}

foreach (FileInfo file in files)

{

lvi = new ListViewItem();

lvi.Text = file.Name;

lvi.ImageIndex = 1;

lvi.Tag = file.FullName;

lvsi = new ListViewItem.ListViewSubItem();

string size = file.Length.ToString();

lvsi.Text = ConvertToReadableBytes(size); ;

lvi.SubItems.Add(lvsi);

lvsi = new ListViewItem.ListViewSubItem();

lvsi.Text = file.LastWriteTime.ToString();

lvi.SubItems.Add(lvsi);

this.listViewFilesAndFolders.Items.Add(lvi);

lvsi = new ListViewItem.ListViewSubItem();

lvsi.Text = file.DirectoryName;

lvi.SubItems.Add(lvsi);

}

this.listViewFilesAndFolders.EndUpdate();

}

catch (System.Exception err)

{

WriteToLog("Грешка в програмата! " + err.Message);

MessageBox.Show("Грешка в програмата! " + err.Message, "Грешка в програмата!", MessageBoxButtons.OK, MessageBoxIcon.Error);

labelErrorReport.Text = "Грешка в програмата! " + err.Message;

direrror = true;

;

}

}

//Обновява ListView за избраните файлове за преименуване

private void PaintSelectedFilesView()

{

try

{

ListViewItem lvi;

ListViewItem.ListViewSubItem lvsi;

this.listViewSelectedFiles.Items.Clear();

this.listViewSelectedFiles.BeginUpdate();

foreach (FileBlock file in selectedFiles)

{

lvi = new ListViewItem();

lvi.Text = file.FileName;

lvi.ImageIndex = 1;

lvi.Tag = file.Location + file.FileName;

lvsi = new ListViewItem.ListViewSubItem();

lvsi.Text = file.NewFileName;

lvi.SubItems.Add(lvsi);

lvsi = new ListViewItem.ListViewSubItem();

lvsi.Text = file.Location;

lvi.SubItems.Add(lvsi);

this.listViewSelectedFiles.Items.Add(lvi);

}

this.listViewSelectedFiles.EndUpdate();

}

catch (System.Exception err)

{

WriteToLog("Грешка в програмата! " + err.Message);

MessageBox.Show("Грешка в програмата! " + err.Message, "Грешка в програмата!", MessageBoxButtons.OK, MessageBoxIcon.Error);

labelErrorReport.Text = "Грешка в програмата! " + err.Message;

}

}

//Премахва файловете от списъка за преименуване

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

{

try

{

labelErrorReport.Text = "";

//ако са избрани повече от един файл, се добавя променливата adjustment, тъй като може да се объркат индексите

int adjustment = 0;

foreach (System.Windows.Forms.ListViewItem file in listViewSelectedFiles.SelectedItems)

{

selectedFiles.RemoveAt(file.Index - adjustment);

adjustment++;

}

PaintSelectedFilesView();

UpdateNewNames();

}

catch (Exception err)

{

WriteToLog("Грешка в програмата! " + err.Message);

labelErrorReport.Text = "Грешка: " + err.Message;

}

}

//Промяна на приоритета за метода към по-висок

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

{

try

{

//ако не е избрано нищо, не изпълнява кода, предотвратява грешки

if (listBoxNamePriorities.SelectedItem != null)

{

int oldindex = listBoxNamePriorities.SelectedIndex;

string item = listBoxNamePriorities.SelectedItem.ToString();

if (oldindex - 1 >= 0)

{

listBoxNamePriorities.Items.RemoveAt(oldindex);

listBoxNamePriorities.Items.Insert(oldindex - 1, item);

listBoxNamePriorities.SelectedIndex = oldindex - 1;

}

}

UpdateNewNames();

}

catch (Exception err)

{

WriteToLog("Грешка в програмата! " + err.Message);

labelErrorReport.Text = "Грешка: " + err.Message;

}

}

//Промяна на приоритета за метода към по-нисък

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

{

try

{

//ако не е избрано нищо, не изпълнява кода, предотвратява грешки

if (listBoxNamePriorities.SelectedItem != null)

{

int oldindex = listBoxNamePriorities.SelectedIndex;

string item = listBoxNamePriorities.SelectedItem.ToString();

if (oldindex+1 < listBoxNamePriorities.Items.Count)

{

listBoxNamePriorities.Items.RemoveAt(oldindex);

listBoxNamePriorities.Items.Insert(oldindex + 1, item);

listBoxNamePriorities.SelectedIndex = oldindex + 1;

}

}

UpdateNewNames();

}

catch (Exception err)

{

WriteToLog("Грешка в програмата! " + err.Message);

labelErrorReport.Text = "Грешка: " + err.Message;

}

}

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

{

BeginRenaming();

}

//Метод за преименуване и архивиране

private void BeginRenaming()

{

//Потвърждение от потребителя

DialogResult result = MessageBox.Show("ВНИМАНИЕ?\r\n\nАко сте включили архивирането преди преименуване, съществуващите файлове може да бъдат презаписани.\r\n\nСигурни ли сте?", "ВНИМАНИЕ!", MessageBoxButtons.YesNo, MessageBoxIcon.Warning, MessageBoxDefaultButton.Button2);

//Ако потвърди

if (result == DialogResult.Yes)

{

try

{

int errcount = 0; // брояч на грешки

//Архивиране на файловете

if (checkBoxBackupFiles.Checked)

{

//безкраен цикъл за нов опит

while (true)

{

WriteToLog("----Започване на бекъп преди преименуване...---- ");

foreach (FileBlock file in selectedFiles)

{

try

{

string fullfilename = file.Location + @"\" + file.FileName; //Взима се пълната локация

System.IO.FileInfo filebuffer = new FileInfo(fullfilename); //Взима се информация за файла

string backupdir = filebuffer.DirectoryName + "\\BitRenamer-Backup\\" + filebuffer.Name; //Създава се дестинация

Directory.CreateDirectory($@"{filebuffer.Directory}\BitRenamer-backup"); // Създава папка, ако такава липсва

filebuffer.CopyTo(backupdir, true); //Копиране

}

catch (Exception err)

{

WriteToLog($"Файлът {file.FileName} не бе архивиран! {err.Message}");

errcount++;

}

}

//Избор за следващ опит, позволява на потребителя да затвори някой файл, ако го е забравил отворен в програма

if (errcount != 0)

{

string dialogtextRetry = "Някои файлове не бяха успешно архивирани.\r\nТова е възможно, ако някои от избраните файлове се използват в момента или са недостъпни.\r\nБихте ли опитали отново? Можете да продължите с преименуването, като натиенете Игнорирай.";

string dialogtitleRetry = "ВНИМАНИЕ!";

DialogResult resultretry = MessageBox.Show(dialogtextRetry, dialogtitleRetry, MessageBoxButtons.AbortRetryIgnore, MessageBoxIcon.Question);

if (resultretry == DialogResult.Abort) throw new Exception("Отказ от потребителя след неуспешно архивиране.");//отказ

if (resultretry == DialogResult.Ignore) break;//игнориране на предупреждението

// Опитва отново, ако се натисне retry

}

else break; //успешен бекъп

}

}

//започване на преименуването

errcount = 0; //ресетва се брояча

WriteToLog("----Започване на преименуването...---- ");

foreach (FileBlock file in selectedFiles)

{

try

{

string fullfilename = file.Location + @"\" + file.FileName; //Взима се пълната локация

System.IO.FileInfo filebuffer = new FileInfo(fullfilename); //Взима се инфомация за файла

string newfullfilename = filebuffer.Directory + "\\" + file.NewFileName; //Прави се дестинация

filebuffer.MoveTo(newfullfilename);//Преименуваме файла

}

//запис в лога при неуспех

catch (Exception err)

{

WriteToLog($"Файлът {file.FileName} не бе преименуван! {err.Message}");

errcount++;

}

}

//Частичен успех

if (errcount != 0)

{

string dialogtextReady = "Някои файлове не бяха успешно преименувани.\r\n" +

"Това е възможно, ако някои от избраните файлове се използват в момента, или са недостъпни.\r\n" +

"Проверете лога за повече подробности\r\n\n\n" +

$" Неуспешно преименувани файлове: {errcount}\r\n" +

$" Успешно преименувани файлове: {selectedFiles.Count - errcount}";

string dialogtitleReady = "Преименуването е частично успешно.";

MessageBox.Show(dialogtextReady, dialogtitleReady, MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

}

//Успешно преименуване

else

{

string dialogtextReady =

"Всички файлове са преименувани успешно\r\n\n\n" +

$" Неуспешно преименувани файлове: {errcount}\r\n" +

$" Успешно преименувани файлове: {selectedFiles.Count - errcount}";

string dialogtitleReady = "Преименуването е успешно.";

MessageBox.Show(dialogtextReady, dialogtitleReady, MessageBoxButtons.OK, MessageBoxIcon.Asterisk);

}

//При частичен успех или пълен успех, списъка се изчиства и се обновява изгледа с папките

listViewSelectedFiles.Items.Clear();

selectedFiles.Clear();

PaintListView(currentdir);

}

//Запис в лога при критична грешка, било то отказ от потребителя или грешка от програмата

catch (Exception err)

{

WriteToLog($"Преименуването е неуспешно! {err.Message}");

MessageBox.Show("Преименуването е неуспешно! Проверете лога за повече подробности. \r\nГрешка: " + err.Message, "Преименуването e неуспешно!", MessageBoxButtons.OK, MessageBoxIcon.Error);

labelErrorReport.Text = "Преименуването е неуспешно! Проверете лога за подробности!";

}

}

}

//Логване на всички грешки

private void WriteToLog(string message)

{

try

{

string logline = DateTime.Now.ToString() + " " + message + "\r\n";

string programpath = Path.GetDirectoryName(Assembly.GetExecutingAssembly().Location);

File.AppendAllText(programpath + "\\BitRenamerlog.txt", logline);

}

catch (Exception err)

{

MessageBox.Show("Неуспешен запис в лога! \r\nГрешка:" + err.Message, "Неуспешен запис в лога!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

//отваряне на логовете

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

{

try

{

string programpath = Path.GetDirectoryName(Assembly.GetExecutingAssembly().Location);

System.Diagnostics.Process.Start(programpath + "\\BitRenamerlog.txt");

}

catch (Exception err)

{

MessageBox.Show("Лог файлът не може да се отвори! \r\nГрешка:" + err.Message, "Лог файлът не може да се отвори!", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

}

}

//Позволява въвеждане само на букви, цифри, долна черта, тире и позволява изтриване

private void textBoxExtChange\_KeyPress(object sender, KeyPressEventArgs e)

{

if (((e.KeyChar < 48 || (e.KeyChar > 57 && e.KeyChar < 65) || (e.KeyChar > 90 & e.KeyChar < 97) || e.KeyChar > 122)) && e.KeyChar != 8 && e.KeyChar != 45 && e.KeyChar != 95) e.Handled = true;

}

//Позволява въвеждане само на букви, цифри, долна черта, тире, точка, спейс и позволява изтриване

private void textBoxAddName\_KeyPress(object sender, KeyPressEventArgs e)

{

if (((e.KeyChar < 48 || (e.KeyChar > 57 && e.KeyChar < 65) || (e.KeyChar > 90 & e.KeyChar < 97) || e.KeyChar > 122)) && e.KeyChar != 8 && e.KeyChar != 32 && e.KeyChar != 45 && e.KeyChar != 46 && e.KeyChar != 95) e.Handled = true;

}

//Всички методи, които само извикват функцията UpdateNewNames()

private void comboBoxDivider\_SelectedIndexChanged(object sender, EventArgs e)

{

UpdateNewNames();

}

private void textBoxAddName\_TextChanged(object sender, EventArgs e)

{

UpdateNewNames();

}

private void numericUpDownSymDelNum\_ValueChanged(object sender, EventArgs e)

{

UpdateNewNames();

}

private void numericUpDownNumZeroes\_ValueChanged(object sender, EventArgs e)

{

UpdateNewNames();

}

private void numericUpDownNumStart\_ValueChanged(object sender, EventArgs e)

{

UpdateNewNames();

}

private void textBoxExtChange\_TextChanged(object sender, EventArgs e)

{

UpdateNewNames();

}

}

}

Клас FileBlock:

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace VSP\_135KNZ\_MyProject{

//клас FileBlock за програмата AP BITRENAMER

//Служи за по-лесно съхранение на данните, нужни при преименуване на файловете

//Повече информация вижте в документа към програмата

internal class FileBlock

{

// Полета

private string filelocation;//Локация на файла

private string filename;//Оригинално име на файла

private string newfilename;//Ново преименувано име на файла

private DateTime datecreated;//Дата на модифициране на файла(използва се при добавяне на дата/час към името)

// Свойства

public string Location//Локация на файла

{

get { return filelocation; }

set { filelocation = value; }

}

public string FileName

{

get { return filename; }//Оригинално име на файла

set { filename = value; }

}

public string NewFileName//Ново преименувано име на файла

{

get { return newfilename; }

set { newfilename = value; }

}

public DateTime DateCreated//Дата на модифициране на файла(използва се при добавяне на дата/час към името)

{

get { return datecreated; }

set { datecreated = value; }

}

// конструктор

public FileBlock (string name, string location, DateTime date)

{

this.FileName = name;//Оригинално име на файла

this.NewFileName = name;//Ново преименувано име на файла

this.Location = location;//Локация на файла

this.DateCreated = date;//Дата на модифициране на файла(използва се при добавяне на дата/час към името)

}

}

}