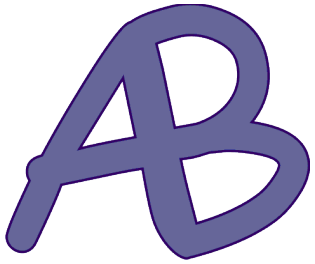
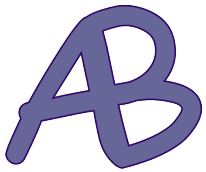


Wykład
Pliki tekstowe



Strumienie i Pliki

Strumienie i pliki

Strumienie są formą wymiany i transportu danych, obsługiwaną przez klasy przestrzeni *System.IO*.

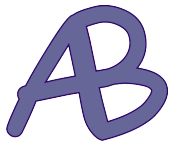
- ✓ Przy użyciu strumieni można komunikować się z konsolą oraz operować na danych znajdujących się w pamięci komputera, w plikach.
- ✓ Np., strumień może być plikiem, pamięcią operacyjną lub współdzielonym zasobem sieciowym.

Strumienie i pliki

Klasy służące do operowania na plikach i katalagach

Klasa	Opis
Directory	Służy do operowania na katalogach (przenoszenie, kopiowanie).
File	Klasa umożliwia tworzenie, usuwanie oraz przenoszenie plików.
Path	Służy do przetwarzania informacji o ścieżkach (do katalogów i plików)
DirectoryInfo	Podobna do klasy Directory. Stosujemy, jeżeli dokonujemy wielu działań na katalogach, gdyż nie wykonuje testów bezpieczeństwa.
FileInfo	Podobna do klasy File. Stosujemy, jeżeli dokonujemy wielu działań na plikach, gdyż nie wykonuje testów bezpieczeństwa.

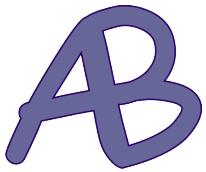
Strumienie i pliki



Przykładowe operacje na katalogu

W naszym przykładzie katalog „test”– sprawdzamy, czy katalog istnieje i tworzymy go gdy nie istniał.

```
if (folderBrowserDialog1.ShowDialog()==DialogResult.OK)
{
    if (!Directory.Exists(folderBrowserDialog1.SelectedPath+"test"))
    {
        Directory.CreateDirectory(folderBrowserDialog1.SelectedPath + "test");
    }
}
```



Klasa File

Strumienie i pliki

Tworzenie i usuwanie plików

```
File.CreateText("C:\\plik.txt");
```

Tworzy nowy plik gotowy do zapisu tekstu z kodowaniem UTF-8.

```
if (!File.Exists("C:\\plik.txt"))
{
    StreamWriter sw = File.CreateText("C:\\plik.txt");

    sw.WriteLine("Witaj świecie");
    sw.Close();
}
```

Aby zapisać tekst do pliku można skorzystać z klasy StreamWriter, której obiekt jest zwracany przez metodę CreateText():

```
File.Delete("C:\\plik.txt");
```

Kasowanie pliku

Strumienie i pliki

Kopiowanie i przenoszenie plików

```
string src = "C:\\test.txt";
string dst = "C:\\kopiatestu.txt";

if (!File.Exists(dst))
{
    File.Copy(src, dst);
}
```

Kopiowanie pliku pod nową nazwą

```
string src = "C:\\test.txt";
string dst = "D:\\test.txt";

if (!File.Exists(dst))
{
    File.Move(src, dst);
}
```

Przenoszenie pliku - w tym przykładzie z dysku c: na dysk d:

Strumienie i pliki

Odczytywanie plików tekstowych za pomocą klasy File

```
textBox1.Text = File.ReadAllText(sciezka);
```

Wczytanie całego pliku (w postaci jednego stringa

```
String[] linie =File.ReadAllLines(sciezka);
List<string> list = new List<string>(linie);
```

Wczytanie całej zawartości pliku w postaci tablicy stringów (jeden wiersz – jedno pole), a następnie utworzenie listy za pomocą wczytanej tablicy.

```
List<string> list = new List<string>(File.ReadAllLines(sciezka));
```

Jak wyżej, tylko bez tablicy pośredniczącej.

Strumienie i pliki

Zapisywanie plików tekstowych za pomocą klasy File

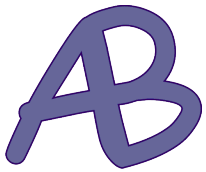
```
File.AppendAllText(sciezka,textBox1.Text);
```

Metoda .AppendALLText( ) otwiera plik, dopisuje do niego tekst i zamyka plik.

```
List<string> lista = new List<string>(File.ReadAllLines(sciezka));
```

```
File.AppendAllLines(sciezka, lista);
```

Metoda .AppendALLLines( ) otwiera plik, dopisuje do niego wszystkie elementy podanej listy i zamyka plik



Strumienie plikowe

Strumienie i pliki

Strumienie

Do odczytywania i zapisywania danych do strumieni używamy odrębnych klas — StreamReader oraz StreamWriter.

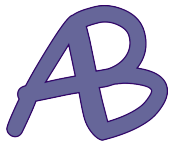
W przypadku danych binarnych są to odpowiednio klasy BinaryWriter i BinaryReader

Zaczynamy od utworzenia egzemplarza klasy FileStream.
Jej konstruktor wymaga podania trzech parametrów:

1. *ścieżki do pliku,*
2. trybu otwarcia pliku,
3. trybu dostępu do pliku.

```
FileStream fs = new FileStream("C:\\test.txt",
                               FileMode.OpenOrCreate,
                               FileAccess.ReadWrite);
```

Strumienie i pliki



Aby odczytać zawartość w pliku tekstowym, należy też utworzyć egzemplarz klasy StreamReader.

W parametrze jego konstruktora należy przekazać obiekt klasy FileStream

```
if (openFileDialog1.ShowDialog()==DialogResult.OK)
{
    FileStream fs = new FileStream(openFileDialog1.FileName,
   FileMode.Open, FileAccess.Read);
    try
    {
        StreamReader sr = new StreamReader(openFileDialog1.FileName);
        textBox1.Text = sr.ReadToEnd();
        sr.Close();
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.ToString());
    }

}
```

Cała zawartość pliku odczytać możemy za pomocą metody ReadToEnd

Jednak cały plik zapisany w pojedynczym łańcuchu jest trudny do przetwarzania

Strumienie i pliki

Częściej odczytujemy plik wiersz po wierszu.

```
while (!sr.EndOfStream)
{
    textBox1.Text += sr.ReadLine();
}
```

.ReadLine( ) - Odczyt pojedynczej linii. Pobierany jest tekst bez znaków końca wiersza.

Zawartość pliku można zapisać w postaci listy – jeden wiersz w jednym elemencie listy.

```
List<String> linie = new List<String>();
while (!sr.EndOfStream)
{
    linie.Add(sr.ReadLine());
}
```

Często zapisujemy plik do listy – w tej postaci można łatwo przetwarzać i edytować tekst.

Strumienie i pliki

Wyświetlenie pliku w kontrolce TextBox

TextBox

Kontrolka TextBox posiada pole TextBox.Text, gdzie zapisać możemy pojedynczy łańcuch – to z niego korzystaliśmy dotychczas.

Aby zapisać zawartość listy w kontrolce TextBox (lub innej przyjmującej tekst w postaci String), należy scalić listę w jeden łańcuch (string) i dopiero w takiej postaci zapisać do kontrolki.

```
List<String> linie = new List<String>();
while (!sr.EndOfStream)
{
    linie.Add(sr.ReadLine());
}
```

```
string calyTekst = "";
foreach (string l in linie)
{
    calyTekst += l;
    calyTekst += "\n";
}
textBox1.Text = calyTekst;
```

Strumienie i pliki

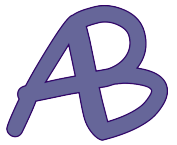
Wyświetlenie pliku w kontrolce TextBlock

Kontrolka TextBlok posiada metodę .Inlines( ) która pozwala dodawać do niej kolejne wiersze.

```
List<String> linie = new List<String>();
while (!sr.EndOfStream)
{
    linie.Add(sr.ReadLine());
}
```

```
foreach (string l in list)
    textBlock1.Inlines.Add(l+"\n");
```

Strumienie i pliki



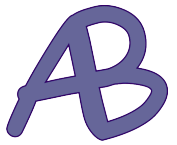
Aby zapisać wartość w pliku tekstowym, należy utworzyć egzemplarz klasy StreamWriter.

W parametrze jego konstruktora należy przekazać obiekt klasy FileStream

```
FileStream fs = new FileStream(sciezka,
                               FileMode.Open, FileAccess.Write);
 StreamWriter sw = new StreamWriter(fs);
 sw.WriteLine("tekst do zapisania w pliku");
 sw.Close();
```

Do zapisu pojedynczej linii tekstu użyć można metody WriteLine( )

Strumienie i pliki



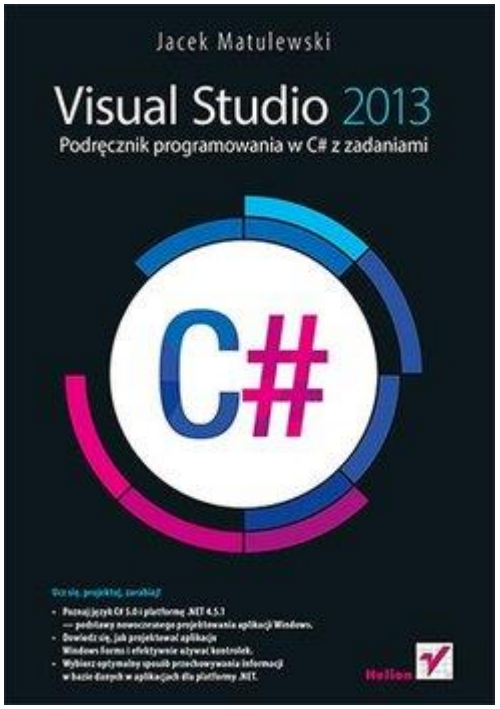
Zapis zawartości pola TextBox do pliku za pośrednictwem listy.
Na teksie zapisanym w liście można wykonać różnego rodzaju operacje – w tym przykładzie- sortowanie.

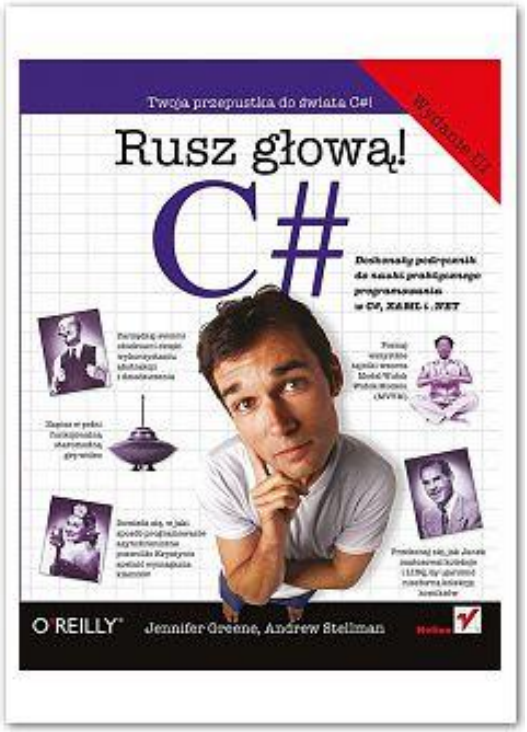
```
FileStream fs = new FileStream(sciezka,
                    FileMode.Open, FileAccess.Write);
 StreamWriter sw = new StreamWriter(fs);
 List<string> list = new List<string>(textBox1.Text.Split('\n'));

 list.Sort(); //przykładowa operacja na liście

 foreach (string l in list)
     sw.WriteLine(l + "\n");
 sw.Close();
```

Literatura:





Użyte w tej prezentacji tabelki pochodzą z książki:Visual Studio 2013. Podręcznik programowania w C# z zadaniami
Autor: Matulewski Jacek, Helion