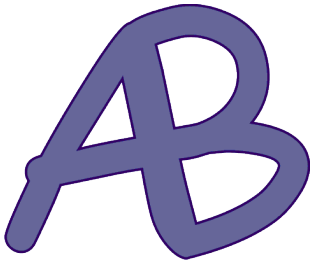
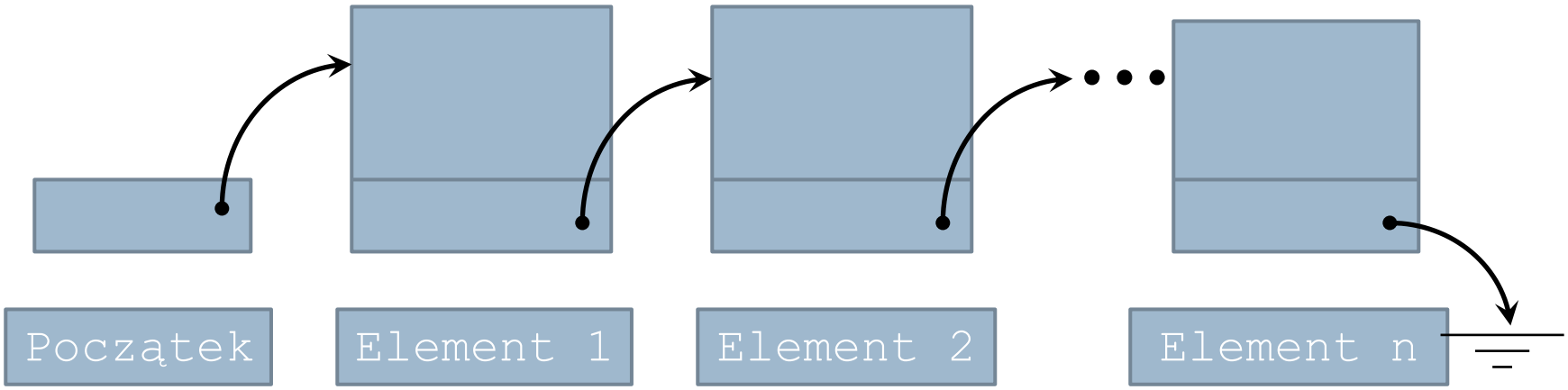


Wykład 7
Pliki tekstowe,

Listy - przypomnienie



Kolekcja „Listy”

Lista - należy do grupy typów ogólnych (ang.generic types).

- ✓ W porównaniu z tablicą (Array) ma tą zaletę, że liczba elementów może być zmieniana już po utworzeniu listy.
- ✓ Można dodawać elementy na koniec, na początek i w środek listy.
- ✓ Można też usuwać dowolny element listy.
- ✓ Dostęp do dowolnego elementu listy możliwy jest, tak samo jak w przypadku tablicy.

Kolekcja „Listy”

Tworzenie listy:

```
List<typ> l = new List<typ>(tab. wart. inicjalizujących);
```

```
List<int> l = new List<int>();
```

```
List<String> s = new List<String>();
```

W parametrze konstruktora listy możemy podać tablicę wartości inicjalizujących.

```
List<int> l = new List<int>(new int[] {1,2,3,4,5});
```

```
List<String> s = new List<String>(new String[] {"aa","bb","cc"});
```

Kolekcja „Listy”

Podstawowe operacje na listach (na przykładzie tablicy String):

```
List<String> nazwa = new List<String>();

nazwa.Add("element");
```
- Dodawanie elementu

```
nazwa.AddRange(new String[] {"aa","bb"});
```
- Daodanie tablcy elementów (na koniec listy)

```
nazwa.Insert(0, "aa");
```
- wstawianie elementu na wskazaną pozycję –UWAGA- nie zastępujemy tylko wstawiamy

```
nazwa.InsertRange(0, new String[]{"aa", "bb"});
```
- wstawianie listy elementu na wskazaną pozycję

```
nazwa.RemoveAt(0);
```
- usunięcie wskazanego elementu

```
nazwa.Remove("bb");
```
- usunięcie elementu o wskazanej wartości,

Kolekcja „Listy”

Podstawowe operacje na listach (na przykładzie tablicy String):

nazwa.Clear();
- wyczyszczenie listy

nazwa.Sort();
- sortowanie listy

nazwa.Reverse();
- odwrócenie listy

nazwa.Count();
- podaje liczbę elementów

nazwa.ToArray(TablicaDocelowa);
- eksportuje listę do tablicy.

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");
    }
}
```

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

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();
}
```

Odczyt pojedynczej linii

Zawartość pliku można zapisać w tablicy – jeden wiersz w każdej komórce.

```
String[] tab = new String[100];
int i = 0;
while (!sr.EndOfStream)
{
    tab[i]= sr.ReadLine();
    i++;
}
```

Częściej jednak zapisujemy plik do listy – puste pola tablicy mogą sprawiać kłopoty.

Strumienie i pliki

Wyświetlenie pliku w kontrolce textBox

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

Jeżeli ustawimy własność kontrolki multiline na true możemy korzystać też ze struktury textBox.Lines, która jest tablicą zmiennych String – jedno pole jedna linijka.

Stąd, jeżeli mamy tablicę łańcuchów możemy ją łatwo wyświetlić w polu textBox.

```
String[] tab = new String[] { "aaa", "bbb", "ccc" };
textBox1.Lines = tab;
```

Strumienie i pliki

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

W parametrze jego konstruktora należy przekazać obiekt klasy FileStream

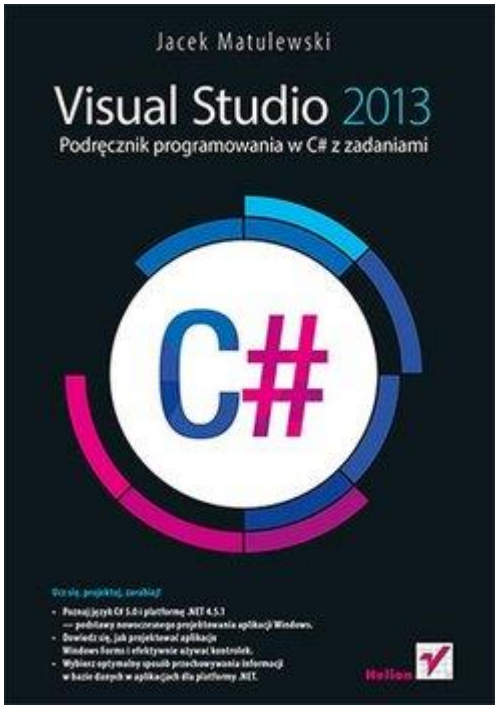
```
if (saveFileDialog1.ShowDialog() == DialogResult.OK)
{
    FileStream fs = new FileStream(saveFileDialog1.FileName,
       FileMode.OpenOrCreate, FileAccess.ReadWrite);
    try
    {
        StreamWriter sw = new StreamWriter(fs);
        sw.WriteLine("Hello World!");
        sw.WriteLine("Bye!");
        sw.Close();
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.ToString());
    }
}
```

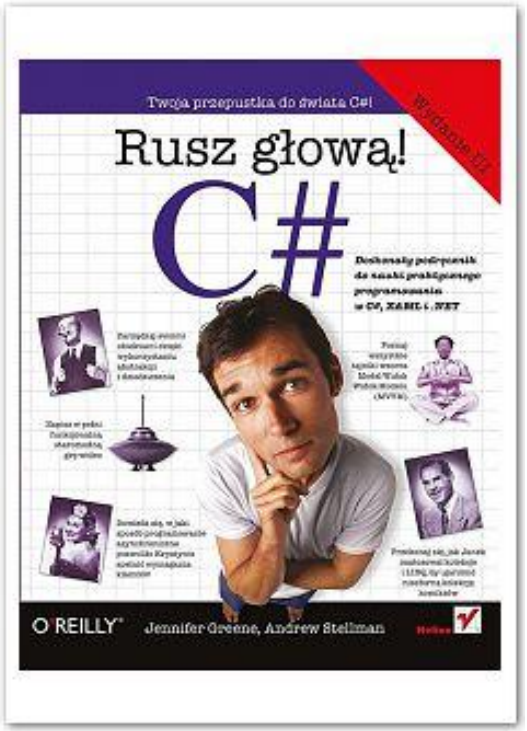
Do zapisu tekstu użyć można metody WriteLine( )

```
if (saveFileDialog1.ShowDialog() == DialogResult.OK)
{
    FileStream fs = new FileStream(saveFileDialog1.FileName,
       FileMode.OpenOrCreate, FileAccess.ReadWrite);
    try
    {
        StreamWriter sw = new StreamWriter(fs);
        int ile=0;
        String[] tab = new String[100];
        ile = textBox1.Lines.Count();
        tab = textBox1.Lines;
        for (int i = 0; i < textBox1.Lines.Count(); i++)
            sw.WriteLine(tab[i]);
        sw.Close();
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.ToString());
    }
}
```

Przykład:
zapis zawartości pola textBox do pliku

Literatura:





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