Konwerter Markdown na LaTeX

Projekt zaliczeniowy z przedmiotu Teoria Kompilacji i Kompilatory autorstwa Artura Sojki i Piotra Waluszka.

Narzędzia i technologie

- **ANTLRv4**: Narzędzie do generowania parserów, które umożliwia nam analizę składniową. Wykorzystujemy ANTLR do definiowania gramatyki, z której generujemy klasy parserów w C#.
- **C#**: Język programowania, w którym tworzymy logikę aplikacji. C# posiada wiele wbudowanych w język funkcji, pomocnych przy parsowaniu i konwersji kodu.
- **CommonMark Specification**: Podstawowa specyfikacja składni Markdown, na której wzorujemy nasz konwerter. CommonMark jest powszechnie używaną próbą standaryzacji Markdowna, co pozwala na zachowanie spójności w interpretacji tego formatu przez różne programy.
- **Pandoc**: Pandoc to wszechstronne narzędzie do renderowania plików Markdown, a także ich konwersji na różne formaty, w tym LaTeX. W naszym projekcie korzystamy z części rozszerzeń Pandoc, co pozwala na obsługę niektórych specyficznych funkcji używanych w Markdown.

Format wejściowy

Wejściem do programu jest plik Markdown o zredukowanej składni. Nasz konwerter bazuje głównie na specyfikacji CommonMark oraz, w mniejszym stopniu, na funkcjonalnościach oferowanych przez Pandoc.

Ograniczyliśmy składnię do następujących struktur, które nie mogą być w sobie zagnieżdżane:

- Nagłówki (tylko te rozpoczynające się od `#`)
- Linie poziome
- Paragrafy tekstu
- Listy numerowane i nienumerowane (z najwyżej trzypoziomowym zagnieżdżaniem, elementy nie są kontynuowane w następnej linii)
- Wcięte bloki kodu
- Odgrodzone bloki kodu (oznaczone przez dokładnie 3 `` ` `` lub `~` i zawsze poprawnie zamknięte)
- Cytaty blokowe (nie są kontynuowane w następnej linii, jeśli nie rozpoczyna się znakiem `>`)
- Obrazy
- Tabele

W samym tekście stylizowane są na razie tylko linki do stron internetowych.

Instalacja (Windows)

1. Pobierz plik .jar - ANTLR 4.13.1
2. Przenieś go do osobnego folderu.

3. Dodaj ścieżkę do pobranego pliku do zmiennej środowiskowej `CLASSPATH` (stwórz ją, jeśli nie istnieje).
4. W jednym z folderów znajdujących się w zmiennej środowiskowej `PATH` utwórz dwa pliki .bat:

   antlr4.bat

   ```
   java org.antlr.v4.Tool %*
   ```

   Komenda `antlr4` służy do generowania kodu.

   grun.bat

   ```
   @ECHO OFF
   SET TEST_CURRENT_DIR=%CLASSPATH:.=;%
   if "%TEST_CURRENT_DIR%" == "%CLASSPATH%" ( SET CLASSPATH=.;%CLASSPATH% )
   @ECHO ON
   java org.antlr.v4.gui.TestRig %*
   ```

   Komenda `grun` służy do wizualizacji drzewa składniowego.
5. Przetestuj działanie komend przez wywołanie ich bez żadnych argumentów (powinna wypisać się lista możliwych argumentów).

(Instalacja na podstawie instrukcji - tam również znajdziesz instrukcje dla innych systemów operacyjnych).

Plugin do JetBrains

Konfiguracja

1. W `Settings > Plugins > Marketplace` dodaj plugin ANTLR v4.
2. Mając wybrany plik .g4 z gramatyką, przejdź do `Tools > Configure ANTLR...`.
3. Ustaw `Language` na CSharp, Python3, Java itp.
4. Polecam ustawić także `Output directory` oraz `Package/namespace` na coś sensownego.

Używanie

1. Kliknij prawym przyciskiem myszy na główną regułę gramatyki.
2. Wybierz `Test Rule ...` (na samym dole).
3. W oknie można teraz podać tekst lub plik do parsowania, drzewo składniowe będzie się generowało automatycznie przy każdej zmianie tekstu lub zapisaniu pliku z gramatyką.
4. Gdy gramatyka jest gotowa, użyj opcji `Tools > Generate ANTLR Recognizer` (skrót CTRL + SHIFT + G).
5. Zostanie wygenerowany kod do parsowania tej gramatyki.

Używanie w C#

1. Używając NuGet'a (na dolnym pasku Rider'a), zainstaluj pakiet `Antlr4.Runtime.Standard`.
2. Może pojawić się potrzeba kopiowania niektórych plików wejściowych do folderu ze skompilowanym programem - kliknij prawym przyciskiem na dany plik > `Properties` i ustaw `Copy to output directory` na `Copy always`.

Używanie w Pythonie

1. Używając PIP'a (lub poprzez PyCharma), zainstaluj pakiet `antlr4-python3-runtime`.

Dokumentacja do używanaia i instalacji

- Ogólna: ANTLR4 Documentation
- Podstawy dla Pythona: ANTLR4 Python Target
- Podstawy dla C#: ANTLR4 C#

Część właściwa

Kod Preprocesora

```
namespace Markdown_to_LaTeX;

public class MarkdownPreprocessor
{
    private string FilePath { get; }

    public MarkdownPreprocessor(string filePath)
    {
        FilePath = filePath;
    }

    public void ProcessFile()
    {
        string[] lines = File.ReadAllLines(FilePath);
        using (StreamWriter writer = new StreamWriter(FilePath, false))
        {
            bool firstLineBlank = string.IsNullOrWhiteSpace(lines[0]);
            bool lastLineBlank = string.IsNullOrWhiteSpace(lines[^1]);

            if (!firstLineBlank)
            {
                writer.WriteLine();
            }

            foreach (string line in lines)
            {
                string processedLine = line.Replace("    ", "    ");
                processedLine = AdjustLeadingSpaces(processedLine);
                processedLine = processedLine.TrimEnd();
                processedLine = processedLine.Replace('', '');

                writer.WriteLine(processedLine);
            }

            if (!lastLineBlank)
            {
                writer.WriteLine();
```

```
            }
        }
    }

    private string AdjustLeadingSpaces(string line)
    {
        int leadingSpaces = 0;
        foreach (char c in line)
        {
            if (c == ' ')
            {
                leadingSpaces++;
            }
            else
            {
                break;
            }
        }
        leadingSpaces -= leadingSpaces % 4;
        return new string(' ', leadingSpaces) + line.TrimStart();
    }
}
```

Lexer MarkdownLexer

```
lexer grammar MarkdownLexer;

// Whitespace
Newline
    : '\r'? '\n'
    | '\r'
    ;
Space : ' ';
Tab : '\t';

// Escaped symbols
EscExclamation : '\\!';
EscDoubleQuote : '\\"';
EscSharp : '\\#';
EscDolar : '\\$';
EscPercent : '\\%';
EscAmp : '\\&';
EscQuote : '\\'';
EscLPAREN : '\\(';
EscRPAREN : '\\)';
EscStar : '\\*';
EscPlus : '\\+';
EscComma : '\\,';
EscDash : '\\-';
EscDot : '\\.';
EscSlash : '\\/';
```

```
EscColon : '\\:';
EscSemicilon : '\\;';
EscLT : '\\<';
EscEqual : '\\=';
EscGT : '\\>';
EscQuestion : '\\?';
EscAt : '\\@';
EscLBRACKET : '\\[';
EscBackslash : '\\\\';
EscRBRACKET : '\\]';
EscCarrot : '\\^';
EscUnderscore : '\\_';
EscCode : '\\`';
EscLBRACE : '\\{';
EscPipe : '\\|';
EscRBRACE : '\\}';
EscTilde : '\\~';

// Special symbols
LBRACKET : '[';
RBRACKET : ']';
LPAREN : '(';
RPAREN : ')';
LBRACE : '{';
RBRACE : '}';
LT : '<';
GT : '>';
Dot : '.';
Slash : '/';
Backslash : '\\';
Colon : ':';
Semicolon : ';';
Exclamation : '!';
Quote : '\'';
DoubleQuote : '"';
Sharp: '#';
Dolar: '$';
Percent : '%';
Amp : '&';
Comma : ',';
Equal : '=';
Question : '?';
At : '@';
Carrot : '^';
Underscore :  '_';
Pipe : '|';
Tilde : '~';
Dash : '-';
Star : '*';
Plus : '+';
Code : '`';
```

```
// Other characters
Digit : [0-9];
Other : .;
```

Parser Markdown

```
parser grammar Markdown;
options{
    tokenVocab=MarkdownLexer;
}

// Main rule
document : (heading | horizontalLine | fencedCode | indentedCodeBlock | list |
blockQuote | table | imageLine | textLine)* EOF;

// Headings
heading
    : headingStart Space headingText (Space Sharp+)?  #atxHeading
    | Newline requiredText setextEnd                  #setextHeading
    | headingStart                                    #emptyHeading
    ;
headingStart : Newline Sharp (Sharp (Sharp (Sharp (Sharp Sharp?)?)?)?)?;
setextEnd
    : Newline Dash Dash+ Newline
    | Newline Equal Equal+ Newline
    ;

// Horizontal lines - require a trailing empty line
horizontalLine
    : hLineDBegin (Space|Dash)* Newline
    | hLineUBegin (Space|Underscore)* Newline
    | hLineSBegin (Space|Star)* Newline
    ;
hLineDBegin : Newline Dash Space* Dash Space* Dash;
hLineUBegin : Newline Underscore Space* Underscore Space* Underscore;
hLineSBegin : Newline Star Space* Star Space* Star;

// Indents - for list items and indented code
indent3 : Newline Space Space Space Space Space Space Space Space Space Space Space
Space;
indent2 : Newline Space Space Space Space Space Space Space Space;
indent1 : Newline Space Space Space Space;

// Fenced code blocks
fencedCode
    : Newline Code Code Code fencedText Code Code Code optionalText
    | Newline Tilde Tilde Tilde fencedText Tilde Tilde Tilde optionalText
    ;

// Indented code blocks - require a preceding empty line
indentedCodeBlock : Newline indentedCode1 (Newline? indentedCode1)*;
```

```
indentedCode1 : indent1 requiredText;

// Lists - itemized and enumerated with max 3 levels of indentation
list  : iList0 | eList0;
list1 : iList1 | eList1;
list2 : iList2 | eList2;
list3 : iList3 | eList3;

iList3 : iListItem3+;
iList2 : iListItem2+;
iList1 : iListItem1+;
iList0 : iListItem0+;

iListItem3 : indent3 iListBegin (Space requiredText)?;
iListItem2 : indent2 iListBegin (Space requiredText)? list3?;
iListItem1 : indent1 iListBegin (Space requiredText)? list2?;
iListItem0 : Newline iListBegin (Space requiredText)? list1?;

iListBegin : Dash | Plus | Star;

eList3 : eListItem3+;
eList2 : eListItem2+;
eList1 : eListItem1+;
eList0 : eListItem0+;

eListItem3 : indent3 eListBegin (Space requiredText)?;
eListItem2 : indent2 eListBegin (Space requiredText)? list3?;
eListItem1 : indent1 eListBegin (Space requiredText)? list2?;
eListItem0 : Newline eListBegin (Space requiredText)? list1?;

eListBegin : Digit+ (Dot | RPAREN);

// Block Quotes - require a space before text
blockQuote : (blockQuoteLine | emptyBlockQuoteLine)+;
blockQuoteLine : blockQuoteStart Space requiredText;
emptyBlockQuoteLine : blockQuoteStart;
blockQuoteStart : Newline GT (GT (GT (GT (GT GT?)?)?)?)?;

// Tables - require a trailing empty line
table : Newline headerRow Newline separatorRow Newline contentRow (Newline
contentRow)* Newline;
headerRow : Pipe cellContent (Pipe cellContent)* Pipe;
contentRow : Pipe cellContent (Pipe cellContent)* Pipe;
cellContent : ~(Newline | Pipe)*;
separatorRow : Pipe separatorContent (Pipe separatorContent)* Pipe;
separatorContent : Space? Colon? Dash+ Colon? Space?;

// Images
imageLine : Newline Exclamation LBRACKET displayText RBRACKET LPAREN linkText RPAREN
optionalText;

// Text
```

```
textLine : Newline optionalText;
requiredText : (link | ~Newline)+;
optionalText : (link | ~Newline)*;
displayText : ~(Newline|RBRACKET)+;
linkText : ~(Newline|RPAREN)+;
urlText : ~(Newline|GT)+;
headingText : (link | ~Newline)+?;
fencedText : .*?;

// Links
link : urlLink | textLink;
urlLink : LT urlText GT;
textLink : LBRACKET displayText RBRACKET LPAREN linkText RPAREN;
```

Inne linki

- Gramatyka C11: C.g4
- Tutoriale (głównie do C#):
  - Getting Started with ANTLR in C#
  - Listeners and Visitors