AKADEMIA GÓRNICZO-HUTNICZA
Wydział Informatyki, Elektrotechniki i Telekomunikacji

Generator spisu publikacji w html na podstawie pliku BibTeX

Prowadzący: mgr Michał Wrzeszcz

Zespół: Alicja Salamon, Michał Torba

1. Wstęp

1.1 Opis zagadnienia

Zadanie polegało na stworzeniu translatora z formatu BiBTeX do formatu HTML. Translator pozwala na otrzymanie nowego spisu publikacji ze spisu otrzymanego na wejściu – zachowując wszystkie zawarte w nim informacje, a nadając im nowy kształt. Dodatkowo przeprowadzane jest szczegółowe sprawdzanie poprawności danych wejściowych.

Obsługiwany jest następujący zestaw danych:

Pozycja	Pola wymagane	Pola opcjonalne
article	author, title, journal, year	volume, number, pages, month, note, key
Book	author, title, publisher, year	volume, series, address, edition, month, note, key
inproceedings	author, title, booktitle, year	editor, volume, series, pages, address, month, organization, publisher, note, key
booklet	title	author, howpublished, address, month, year, note, key
Inbook	author, title, chapter, publisher, year	volume, series, type, address, edition, month, note, key
incollection	author, title, booktitle, publisher, year	editor, volume, series, type, chapter, pages, address, edition, month, note, key
Manual	title	author, organization, address, edition, month, year, note, key
mastersthesis	author, title, school, year	type, address, month, note, key
phdthesis	author, title, school, year	type, address, month, note, key
techreport	author, title, institution, year	editor, volume, series, address, month, organization, publisher, note, key
Misc	-	author, title, howpublished, month, year, note, key
unpublished	author, title, note	month, year, key

1.2 Wykorzystane narzędzia

Do rozwiązania użyliśmy języka programowania Python w wersji 3.0. Wykorzystaliśmy 2 kluczowe moduły ułatwiające nam pracę:

- *Re – pozwalający na użycie wyrażeń regularnych*
- *PLY - generator parserów*

  Generator parserów PLY (Python-Lex-Yacc) jest odpowiednikiem narzędzi bison/flex dla języka Python, generującym **parsery typu LALR** (domyślnie) lub SLR. PLY składa się z dwóch modułów: **lex** (analiza leksykalna) oraz **yacc** (parsing).

2. Opis problemu

2.1 Dane wejściowe

Dane wejściowe to plik zawierający opis publikacji w formacie BibTeX. Przykładowy fragment:

```
@Book{press,
  author    = "Press, W. and Teutolsky, S. and Vetterling",
  title     = "The {A}rt of {S}cientific {C}omputing",
  year      = 2007,
  publisher = "Cambridge University Press"
}
```

2.2 Wyniki

Odpowiadający rezultat na wyjściu (widok w przeglądarce):

testy/poprawne/example.bib

press *book*

- *publisher* - Cambridge University Press
- *title* - The {A}rt of {S}cientific {C}omputing
- *year* - 2007
- *author* - Press, W. and Teutolsky, S. and Vetterling

Oraz w pliku `result.html`

```
<p>
<b>press</b>
<i>book</i>
<ul>
      <li><i>publisher</i> - Cambridge University Press</li>
      <li><i>title</i> - The {A}rt of {S}cientific {C}omputing</li>
      <li><i>year</i> - 2007</li>
      <li><i>author</i> - Press, W. and Teutolsky, S. and Vetterling</li>
</ul>
</p>
```

2.3 Dodatkowe założenia

- Błędem jest wystąpienie pól innych niż obowiązkowe i opcjonalne, powtórzenie pola lub brak pola obowiązkowego
- Pola w pliku wejściowym mogą wystąpić w dowolnej kolejności
- Identyfikator dla każdej publikacji ma być unikalny, powtarzające się ID jest błędem
- Dla nazw pól nie rozróżniamy wielkości liter
- Zbędne białe znaki zostaną usunięte

3. Realizacja rozwiązania

3.1 Parsowanie

Dane wejściowe parsowane zostały wg następującej gramatyki:

```
publications → publication
        | publications publication

publication → type '{' ID ',' fields '}'

fields → field
        | fields ',' field

field → name '=' value

type → @Book
        | @Article
        | @Inproceedings
        | …
```

Jest to język klasy 3 w hierarchii Chomsky'ego, ponieważ można przedstawić ją za pomocą wyrażenia regularnego

```
(@a*{a*, (a*=a*,)* a*=a*})*
```

gdzie a oznacza dowolną literę alfabetu, lub dla uproszczenia:

```
(@type{id, (name=value,)* name=value})*
```

Gdzie `type, id, name, value` to dowolne ciągi znaków alfabetu.

Parsowane wejście przetrzymywane jest w odpowiedniej strukturze. Np. dane w postaci

```
@Book{press,
  author    = "Press, W. and Teutolsky, S. and Vetterling",
  title     = "The {A}rt of {S}cientific {C}omputing",
  year      = 2007,
  publisher = "Cambridge University Press"
}
```

Zostaną sparsowane do struktury:

```
(
('@Book', 1),
('press', 1),
{
      'publisher': ('Cambridge University Press’, 5),
      'title': ('The {A}rt of {S}cientific {C}omputing', 3),
      'year': ('2007', 4),
      'author': ('Press, W. and Teutolsky, S. and Vetterling', 2)
}
)
```

Czyli do trzyelementowej krotki, gdzie pierwszy element opisuje typ publikacji, drugi – słowo kluczowe, a trzeci jest mapą opisującą zestaw pól wraz z odpowiadającymi im wartościami. Ponadto elementy te zawierają informacje o numerze linii, w której pojawiły się w wejściowym pliku.

Efekt ten uzyskaliśmy dzięki modułowi PLY i zdefiniowaniu tokenów:

```
tokens = (
      'RODZAJ_PUBLIKACJI',
      'WARTOSC_W_CUDZYSLOWIE',
      'WARTOSC',
)
literals = "{},="
```

Oraz produkcji:

```
def p_publications1(p):
      """publications : publication"""

def p_publications2(p):
```

```
      """publications : publications publication"""

def p_publication(p):
      """publication : type '{' WARTOSC ',' fields '}'"""

def p_fields1(p):
      """fields : field"""

def p_fields2(p):
      """fields : fields ',' field"""

def p_field1(p):
      """field : WARTOSC '=' WARTOSC_W_CUDZYSLOWIE"""

def p_field2(p):
      """field : WARTOSC '=' WARTOSC """

def p_type(p):
      """type : RODZAJ_PUBLIKACJI"""
```

3.2 Sprawdzanie błędów

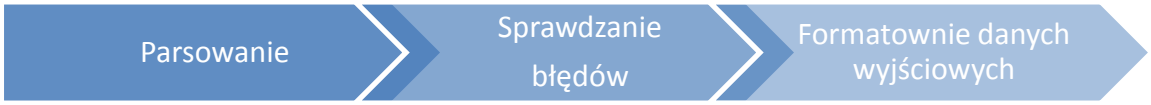
Sprawdzanie błędów odbywa się po sparsowaniu struktur:

1. Czy typ publikacji jest dopuszczalny?
2. Czy podane zostały wszystkie pola obowiązkowe?
3. Czy nie podano pól niedopuszczalnych dla danego typu publikacji?
4. Czy pola nie powtarzają się? (sprawdzane na etapie parsowania)
5. Czy wszystkie ID są unikatowe?

3.3 Formatowanie danych wyjściowych

Jest to prosta funkcja uruchamiana na wszystkich krotkach, które pomyślnie przeszły weryfikację poprawności

3.4 Proces



4. Sposób użytkowania

Sposób użytkowania naszego programu jest bardzo prosty. Wystarczy uruchomić go w konsoli podając jeden argument – plik lub folder, w którym znajduje się opis publikacji w formacie BibTeX. Np. będąc w katalogu zawierającym plik `TKprojekt.py` oraz pliki testowe należy wydać polecenie:

```
python3 TKprojekt.py test
```

Katalogi są przeglądanie rekursywnie.

W paczce zamieszczono folder z plikami testowych, które dobrze prezentują działanie programu.

Wynikiem działania programu są dwa pliki: `result.html`, który należy otworzyć za pomocą przeglądarki oraz plik `bledy.txt`, który zawiera listę błędów w podanych plikach `.bib`.

5. Podsumowanie

Dzięki temu projektowi nauczyliśmy się sprawnie posługiwać narzędziami `ply.lex` i `ply.yacc`. Powtórzyliśmy też przydatne na co dzień informacje o wyrażeniach regularnych i module `re`. Wykorzystaliśmy wiedzę i umiejętności zdobyte na laboratorium z przedmiotu Teoria Kompilacji 1.

6. Bibliografia

1. http://pl.wikipedia.org/wiki/BibTeX
2. http://www.dabeaz.com/ply/ply.html
3. http://docs.python.org/2/tutorial/
4. http://pl.wikipedia.org/wiki/Hierarchia_Chomsky'ego