Zadaća 3 iz predmeta Dizajn Kompajlera

17. Nov 2019.

Problem 1

Napisati lekser koji će odraditi leksičku analizu ulaza u kalkulator i izdvojiti tokene koji su definisani u fajlu `token.hpp`. Kao što je navedeno u `token.hpp` fajlu, lekser treba da prepoznaje identifikatore, cijele brojeve, operator dodjeljivanja, zagrade i osnovne matematičke operacije. Također je potrebno implementirati ispis svih pronađenih tokena pozivom `print()` metode iz `token.hpp` fajla.

Možete testirati ispravnost leksera koristeći priloženi `ulaz.txt` fajl, a izlaz iz programa porediti sa priloženim `izlaz.txt` fajlom.

Problem 2

Leksički analizator iz zadatka 1 napisati koristeći flex alat. Kao i u zadatku 1, potrebno je vršiti ispis svih tokena u istom formatu.

Možete testirati ispravnost vašeg programa poredeći izlaz sa priloženim fajlom.

Template main funkcije za zadnja dva zadatka se nalazi ispod:

```
int main(void) {
  Token t;
  while((t = getToken()).tag != 0){
    t.print();
  }
  return 0;
}
```

Problem 3.

Implementirati leksički analizator (koristeći source code sa primjera na predavanju i vježbama) koji analizira lekseme XML fajlova na način da prepoznaje sljedeće klase leksema:

- TAGBEGIN - početak taga u formatu <NAME
- TAGEND - kraj taga u formatu >
- TAGCLOSE - zatvaranje taga u formatu
- TAGENDANDCLOSE - kraj i zatvaranje taga u formatu />
- ATTRIBUTENAME - u formatu NAME
- EQUAL - znak jednakosti =
- ATTRIBUTEVALUE - vrijednost atributa koja se nalazi pod navodnicima. Vrijednost moze biti bilo šta.
- CONTENT - sadržaj izmeu početka i kraja tagova

Lekser treba da na kraju ispiše:

- Sve pronađene tokene sa pripadajućim klasama.
- Uz ispis svakog tokena napisati broj linije i broj kolone gdje je početak pronađenog leksema,
- Broj tokena u pojedinim klasama.

Npr. za jednostavan xml primjer:

```
<item attr1="value1">
  <subitem>
    content for tag subitem
  </subitem>
</item>
```

ispis treba da bude u formatu:

```
$ ./problem1 < input_fajl.xml
<TAGBEGIN, <item>: line 1, column 1
<ATTRIBUTENAME, attr1>: line 1, column 7
<EQUAL, =>: line 1, column 12
<ATTRIBUTEVALUE, value1>: line 1, column 14
<TAGEND, >>: line 1, column 21
<TAGBEGIN, <subitem>: line 2, column 1
<TAGEND, >>: line 2, column 9
<CONTENT, content for tag subitem>: line 3, column 1
<TAGCLOSE, </subitem>>: line 4, column 1
<TAGCLOSE, </item>>: line 5, column 1
TAGBEGIN: 2
TAGEND: 2
TAGCLOSE: 2
TAGENDANDCLOSE: 0
ATTRIBUTENAME: 1
EQUAL: 1
ATTRIBUTEVALUE: 1
CONTENT: 1
```

Problem 4

Zadatak iz problema 3 implementirati koristeći alat flex. Ispis pri pozivu:

```
$ ./problem2 < input_fajl.xml
```

treba biti potpuno isti kao u problemu 3.

Predaja zadaće

Za svaki problem potrebno je napraviti zaseban direktorij sa imenima *problem1*, *problem2*, i td. Unutar direktorija se nalaze sve potrebne datoteke za kompajliranje programa. Svaki problem mora imati svoj Makefile koji će komandom `make` kompajlirati program i napraviti jedan izvršni file koji će nositi ime kao i problem, dakle *problem1*, *problem2* i td.

Na primjer, naredna sekvenca shell komandi treba biti validna

```
cd problem1
make
```

```
./problem1 < input_fajl > output_fajl
```

Program će koristit standardni ulaz i izlaz za leksičku analizu.

Primjer Makefilea koji prepoznaje flex možete pronaći u rješenjima laba 3 za primjere `json` i `scanner`.

U zadacima 1 i 2 po potrebi možete modifikovati main funkciju i token.hpp, ali ispis treba ostati identičan kao u priloženim fajlovima.