FORMÁLNÍ JAZYKY A PŘEKLADAČE
SEMESTRÁLNÍ PRÁCE
PŘEKLADAČ JAZYKA „IPV4“ DO PL/0

VYPRACOVAL:

JIŘÍ NOHÁČ

PŘEMYSL KOUBA

ZÁKLADNÍ FUNKCE JAZYKA

- Povinné konstrukce
- Nepovinné:
  - pole
  - for, while, do-while
  - else větev
  - datové typy :
    - boolean + operace
    - string + porovnání

•paralelní přiřazení
•násobné přiřazení
•podmíněné přiřazení
•vlastní interpret

ZÁPIS KÓDU A JEHO VÝZNAM

```
#proměnné

string var;

num number;

boolean isTrue;

#funkce

void hello_world() {

      var = "Ahoj svete";

      boolean isTrue = true;

      if(true) {

            number = 40;

      } else {

            number = 20;

      }

}
```

- 10.0.0.1 127.2.0.0
  4.0.0.128 127.2.0.0
  8.0.0.255 127.2.0.0
  12.0.0.1 127.2.0.1
  10.0.0.1 127.1.0.0
  127.2.0.7 3.65.104.111
  3.106.32.115
  3.118.101.116 3.101.10.0
  127.2.0.7 127.2.0.0
  8.0.0.255 127.1.0.0
  0.0.0.1 127.2.0.0
  127.4.0.0 127.2.0.3
  4.0.0.255 127.1.0.1
  0.0.0.1 127.2.0.4
  127.2.0.1 4.0.0.128
  127.1.0.0 0.0.0.40
  127.2.0.0 127.2.0.2
  127.4.0.1 127.2.0.1
  4.0.0.128 127.1.0.0
  0.0.0.20 127.2.0.0
  127.2.0.2 127.2.0.2

```
i_str:1 sz_com
i_int:128 sz_com
i_bl:255 sz_com
void:1 sz_{ i_str:1 o_=
sz_quo Ahoj svete
sz_quo sz_com
i_bl:255 o_= v_int:1
sz_com
c_if sz_( i_int:255 o_==
v_int:1 sz_) sz_{
i_int:128 o_= v_int:40
sz_com
sz_} c_el sz_{ i_int:128
o_= v_int:20 sz_com
sz_} sz_}
```

TECHNOLOGIE IMPLEMENTACE

- Zvolený jazyk
  - Java
- Zpracování gramatiky
  - ANTLR (rekurzivní sestup)

CO JE VLASTNĚ HOTOVO?

V momentálním stavu máme:

- Lexikální analýza kódu
- Interpret PL0

Nutno vypracovat:

- Kompletně vyřešená gramatika
- Syntaktický analyzátor v ANTLR
- Sémantická analýza a generování kódu do PL0

LEXIKÁLNÍ ANALÝZA

Aplikace provede:

- Načtení seznamu všech klíčových IP adres a jejich přepisovacích tokenů
- Postupné zpracování každé IP adresy ze vstupního kódu
- Zapsání příslušného tokenu a popřípadě jeho hodnoty do souboru

```
.
.
.
10.x.x.x i_str;# identifikator pro string
11.x.x.x i_strc;# identifikator pro string const
#funkce
12.x.x.x f_int;# function s navratovou hodnotou int
13.x.x.x f_char;# function s navratovou hodnotou char
14.x.x.x f_bl;# function s navratovou hodnotou boolean
15.x.x.x f_str;# function s navratovou hodnotou string
#operandy 127.1.0.x
127.1.0.0 o_=;# prirazeni =
127.1.0.1 o_==;# porovnani ==
127.1.0.2 o_+;# soucet +
127.1.0.3 o_-;# odpocet -
127.1.0.4 o_*;# nasobeni *
127.1.0.5 o_/;# deleni /
127.1.0.6 o_>;# vetsi >
.
.
.
```

LEXIKÁLNÍ ANALYZÁTOR PŘÍKLAD

#int cislo = 567;

#int cislo2 = 25;

#int vys = cislo + cislo2;

#str k = "ahoj";

i_int:0 o_= v_int:567 sz_com

i_int:1 o_= v_int:25 sz_com

i_int:2 o_= i_int:0 o_+ i_int:1 sz_com

i_str:0 o_= sz_quo ahoj sz_quo sz_com

4.0.0.0 127.1.0.0 0.0.2.55 127.2.0.0

4.0.0.1 127.1.0.0 0.0.0.25 127.2.0.0

4.0.0.2 127.1.0.0 4.0.0.0 127.1.0.2 4.0.0.1 127.2.0.0

10.0.0.0 127.1.0.0 127.2.0.7 3.97.104.111 3.106.0.0 127.2.0.7 127.2.0.0

INTERPRET PL0

Aplikace provede:

- Načtení vstupního souboru s PL0 kódem programu
- Vytvoření zásobníkové struktury, hlavičky, báze a inst.
- Postupné provádění operací a jejich důsledek na zásobníkovou strukturu
- Každá změna stavu se projeví na obrazovce uživateli viz. obr

```
0   JMP 0   13
1   JMP 0   2
2   INT 0   3
3   LOD 1   3
4   LIT 0   1
5   OPR 0   2
6   STO 1   3
7   LOD 1   3
8   LIT 0   2
9   OPR 0   13
10  JMC 0   12
11  CAL 1   2
12  RET 0   0
13  INT 0   4
14  LIT 0   0
15  STO 0   3
16  CAL 0   2
17  RET 0   0
```

```
----------------------------
CURR INSTRUCTION: 6 STO 1 3
NEXT INSTRUCTION: 7 LOD 1 3
INST: 6
BASE: 4
HEAD: 6
[0, 0, -1, 1, 0, 0, 17]
----------------------------

----------------------------
CURR INSTRUCTION: 7 LOD 1 3
NEXT INSTRUCTION: 8 LIT 0 2
INST: 7
BASE: 4
HEAD: 7
[0, 0, -1, 1, 0, 0, 17, 1]
----------------------------
```

ZÁVĚR

- Lex. analýza a interpret vyřešen
- Zpracovat finální verzi gramatiky
- Naprogramovat zbytek klíčových komponent

DĚKUJEME VÁM ZA POZORNOST

GitHub Repozitář: https://github.com/rizir01/FJP_IP.git