Fakulteta za elektrotehniko, računalništvo in informatiko

# Prevajanje programskih jezikov
# Projektna naloga: Poročilo

Študent: David Leskovar

Študijski program: ITK UN

Datum: 06.06.2022

# 1. Konstrukti

## *(I) Števila*

Primer: **`1, 1.1 ...`**

## *(II) Stringi*

Primer: **`“Maribor”, “Blejsko jezero”, “Krožišče štuk”`** **…**

## *(III) Točke*

Točke na zemljevidu predstavljajo lokacijo, kjer prvo število predstavlja koordinato x, drugo pa koordinato y.

Primer: **`(1,2), (9.1,3)`** **…**

## *(IV) Bloki*

Blok **`city`** predstavlja mesto, ki ga opisujemo. Blok **`city`** lahko vsebuje vse ostale bloke, razen samega sebe. Vsebuje lahko tudi vse razpoložljive ukaze. Število blokov in ukazov je poljubno veliko, prav tako pa je možno, da blok ne vsebuje niti enega bloka oz. ukaza.

Primer:

```
city “Slovenj Gradec” {
        COMMANDS...
        BLOCKS...
}
```

Blok **road** predstavlja cesto.

Blok **river** predstavljo reko.

Blok **building** predstavlja zgradbo.

Blok **field** predstavlja polje.

Blok **park** predstavlja park.

Blok **lake** predstavlja jezero.

Blok **roundabout** predstavlja krožišče.

Primer:

```
city “Slovenj Gradec” {
    building “Špar” {
        COMMANDS...
    };
    river “Mislinja” {
        COMMANDS...
    };
}
```

# *(V) Ukazi*

Vsi bloki imajo skupno kolekcijo ukazov, ki jih lahko uporabljajo. Vsak blok lahko uporabi 0 ali več ukazov.

Ukaz **`line(POINT,POINT)`** na zemljevidu izriše črto med podanima točkama.

Primer:

```
river “Mislinja” {
     line((5,1),(3,10));
};
```

Ukaz **`bend(POINT,POINT,number)`** na zemljevidu izriše črto med podanima točkama, ob tem pa upošteva tudi stopnjo ukrivljenosti, ki jo podamo s tretjim parametrom.

Primer:

```
road “Glavna ulica” {
     bend((5,1),(3,10),30);
};
```

Ukaz **`box(POINT,POINT)`** na zemljevidu izriše pravokotnik med podanima točkama.

Primer:

```
building “Lidl” {
     box((1,1),(2,2));
};
```

Ukaz **circle(POINT,number)** na zemljevidu izriše krog, ki ima središče v podani točki in ima polmer, katerega podamo z drugim parametrom.

Primer:

```
roundabout "Krozno" {
     circle((5,5), 2);
};
```

Ukaz **elipse(POINT,number,number)** na zemljevidu izriše elipso katera ima središče v podani točki, ostala parametra pa predstavljata velikost male in velike polosi.

Primer:

```
Jezero "Ivarcko jezero" {
     elipse((5,5), 2,5);
};
```

Posebnost so ukazi tree(POINT), sign(POINT) in bin(POINT), saj za delovanje ne potrebujejo biti znotraj bloka. Na zemljevidu predstavljajo eno samo točko.

Ukaz **tree(POINT)** na zemljevidu označi drevo, ukaz **sign(POINT)** na zemljevidu označi znak, ukaz **bin(POINT)** na zemljevidu označi koš.

Primer:

```
city "Slovenj Gradec" {
     tree(1,5);
     tree(1,6);
     bin(6,3);
}
```

# 2. Spremenljivke

Z **var** in **point** lahko neke vrednosti in točke shranimo v spremenljivke in jih kasneje uporabimo. **var** lahko spremenljivki dodeli vrednost tipa **Float**, **point** pa spremenljivki dodeli vrednost tipa **POINT**.

Primer:

```
building “Spar” {
     var a := 15;
     var b := a;
     point tocka := (a,5);

     box((15,15),tocka);
     box((10,a),tocka);
};

river “Mislinja” {
     var c := 3;
     var d := 1;
     point tocka1 := (c,d);

     bend((1,c),tocka1, d);
};
```

# 3. Izjave

Vrednosti lahko izračunamo na podlagi parametrov in že obstoječih spremenljivk. S pomočjo osnovnih računskih operacij pa lahko vrednosti parametrov izračunamo na mestu parametra.

Primer:

```
building “Spar” {
     var a := 15;
     var b := a-2;
     var c := ((a*b)/2);

     point tocka := (a+3,5);

     box((15+3/4,15),tocka);
     box((10,-a),tocka);
     box(tocka,(a,(b/2)*10));
};
```

# 4. Kontrola toka

Za zanke imamo na voljo **`while`** zanko.

```
while(condition){
     COMMANDS...
}
```

Primer:

```
 building “Spar” {
      var a := 15;
      var b := 5;
      while(a>b){
            box((5,b),(a,1));
            var b := b + 1;
            while(b>10){
                  box((b,1),(3,1));
             }
      }
};
```

Za preverjanje pogoja imamo na voljo **`if/else`**.

```
if(condition){
      COMMANDS...
}else{
      COMMANDS...
}
```

Primer:

```
building “Spar” {

     var a := 5;
     var b := 1;

          If(a>b){
               box((1,1),(a,b));
          }
          else{
               Box((1,1),(b,a));
               tree(a,b);
          }

};
```

# 5. Seznam

Vrednosti in točke lahko shranimo v sezname. Obstajata 2 vrsti seznamov in sicer seznam, katerega lahko napolnimo z vrednostmi tipa **Float** in seznam, katerega lahko napolnimo z vrednostmi tipa **POINT**.

Primer:

```
building “Spar” {

     list<var> list1 := (1,2,3,4,5);
     list<var> list2 := (1+1,a+2,((a+3)/(2*b)),c*c,-13);

     point tocka := (10,10);

     list<point> list3 := ((1,2),(a+b,3),tocka,
     (4,4*ab));
};
```

Skozi sezname lahko iteriramo s pomočjo **for each** zanke.

Primer:

```
building “Spar” {

     list<var> list1 := (1,2,3,4,5);

     foreach x in list1{
          box((x,5),(1,x));
     }
};

river “Mislinja” {

     list<point> list2 := ((1,1),(2,2),(3,3));

     foreach y in list2{
          bend(y,(3,1),40);
     }
};
```

# BNF Gramatika:

**E:=  city STRING {BP}**
**BP:= B BP| 3**
**B:= BLOCKS | COMMANDS**

**BLOCKS:= road STRING {MULTILINECC}; | river STRING {MULTILINECC}; | building STRING{MULTILINECC}; | field STRING{MULTILINECC};**
**|park STRING{MULTILINECC}; |lake STRING {MULTILINECC}; | roundabout STRING{MULTILINECC};**

**MULTILINECC:= COMMANDS MULTILINECC | 3**
**COMMANDS:= BEND|LINE | BOX | CIRCLE | ELIPSE | IFELSE | WHILE | FOREACH | TREEBINSIGN | VAR**

**IFELSE:= if(statement) {MULTILINECC} else{MULTILINECC}**

**WHILE:=while(statement){MULTILINECC}**

**FOREACH:=foreach VARIABLE in VARIABLE{MULTILINECC}**

**STATEMENT:= O OPERATOR O**
**OPERATOR:= < | > | =**

**TREEBINSIGN:=  tree(POINT); | bin(POINT); | sign(POINT);**

```
VAR:= var STRING :=O; |  point STRING :=POINT; | list<var>
VARIABLE := (VCC); | list<point> VARIABLE := (PCC);

VCC:= V VCC | 3
V:= O | O,

PCC:= P PCC | 3
P:= VARIABLE | POINT


BEND:=bend(POINT,POINT,O);
LINE:=line(POINT,POINT);
BOX:=box(POINT,POINT);
CIRCLE:=circle(POINT,O);
ELIPSE:=elipse(POINT,O,O);
POINT:=(O,O)


O ::= T EE
EE ::= + T EE | - T EE | 3;
T ::= F TT
TT ::= * F TT | / F TT | ^ F TT | 3
F ::= ( O ) | number | - number | variable | -variable
```

# Implementacija:

## (1) Scanner

Na podlagi BNF gramatike najprej razvijemo pregledovalnik, ki odstrani vse komentarje in prazna mesta, prav tako pa vrne seznam vseh leksikalnih simbolov, ki jih naša BNF gramatika podpira. Za opis podprtih terminalnih simbolov uporabimo deterministični končni avtomat, ki ga implementiramo s pomočjo tabele prehoda stanj. Če je celotna izvorna koda ustrezna, pregledovalnik vrne seznam vseh leksikalnih simbolov, v nasprotnem primeru pa javi napako na vrstici in stolpcu, kjer je do te prišlo.

## (2) Parser

Na podlagi BNF gramatike razvijemo še LL1 razpoznavalnik s tehniko rekurzivnega navzdolnjega razpoznavanja. Razpoznavalnik za razpoznavanje uporablja pregledovalnik preko katerega se pomikamo po leksikalnih simbolih. V primeru da je izvorna koda ustrezna, razpoznavalnik vrne “accept”, v nasprotnem primeru pa “reject”.