

Projektna naloga

(RentRide)

Študent:

Jan Alif

Tit Podhraški

Anej Mumel

Datum in mesto:

28.05.2024, Maribor

# **Analiza in načrt jezika**

**Osnovni elementi in sintaksa**

carRide: Osnovni element, celotna pot

road: pot, ki jo avto prepotuje

car: lokacija avta, ki bo prikazana na zemljevidu

start: štartna točka avta

finish: kočna točka avta

crossSection: vmesna križišča na poti če so

roundAbout: vmesna krožišča na poti če so

gasStation: bencinske postaje v bližini poti, opcijsko

electricalStation: električne postaje v bližini poti, opcijsko

parking: parkirišča v bližini končne točke, opcijsko

passenger: vstopna in izstopna točka morebitnega sopotnika

Namen jezika je vnos in prikaz poti na zemlejvidu s pomočjo konstruktov »road«, »car«, »start« in »end« ter s pomočjo poljubnih konstruktov kot so »crossSection«, »roundAbout«, »gasStation«, »electricalStation«, »parking« in »passanger«. S pomočjo vseh teh elementov lahko prikažemo celotno pot avta od začetka do konca, vsa vmestna križišča in krožišča, morebitne bencinske oz. električne postaje, parkirišča v bližini ciljne točke in vstopne in izstopne točke morebitnih sopotnikov. Podprta je tudi možnost deklaracija in uporaba spremenljivk in osnovnih matematičnih izrazov.

**Definicija BNF jezika**

CAR\_RIDE ::= "carRide" string "{" DECLARATIONS ROAD CAR START FINISH CROSS ROUND GAS ELECTRICITY PARKING PASSENGERS "}"

DECLARATIONS ::= DECLARATION DECLARATIONS | ε

DECLARATION ::= VAR\_DECL | POINT\_DECL

VAR\_DECL ::= var "=" EXPR ";"

POINT\_DECL ::= "const" var "=" POINT ";"

ROAD ::= "road" "{" PATH "};"

PATH ::= LINE PATH | BEND PATH | ε

LINE ::= "line (" POINT "," POINT ");"

BEND ::= "bend (" POINT "," POINT "," int ");"

POINT ::= "(" EXPR "," EXPR ")"

CAR ::= "car" string "{" "carPoint" POINT ";" "id" ":" int "};"

START ::= "start" "{" POINT "};"

FINISH ::= "finish" "{" POINT "};"

CROSS ::= "crossSection" string "{" BOX "};" CROSS | ε

BOX ::= "box" "(" POINT ", " POINT ");"

ROUND ::= "roundabout" string "{" CIRC "};" ROUND | ε

CIRC ::= "circ" "(" POINT "," EXPR ");"

GAS ::= "gasStation" string "{" POINTS FILTER "};" | ε

POINTS ::= POINT ";" POINTS | ε

FILTER ::= "let" var "=" "neigh" "(" POINT "," EXPR ");" FOREACH

FOREACH ::= "foreach" var "in" var "{" "highlight" var "}"

ELECTRICITY ::= "electricStation" string "{" POINTS FILTER "};" | ε

PARKING ::= "parking" string "{" POINTS FILTER "};" | ε

PASSENGERS ::= PASSENGER PASSENGERS | ε

PASSENGER ::= "passenger" string "{" START FINISH "};"

EXPR ::= ADDITIVE

ADDITIVE ::= MULTIPLICATIVE ADDITIVE'

ADDITIVE' ::= "+" MULTIPLICATIVE ADDITIVE'

| "-" MULTIPLICATIVE ADDITIVE' | ε

MULTIPLICATIVE ::= UNARY MULTIPLICATIVE'

MULTIPLICATIVE' ::= "\*" UNARY MULTIPLICATIVE'

| "/" UNARY MULTIPLICATIVE' | ε

UNARY ::= "+" PRIMARY | "-" PRIMARY | PRIMARY

PRIMARY ::= int | real | var | "(" ADDITIVE ")"

string = ".\*?"

int = [0-9]+

real = -?\d+\.\d+

var = \_{[A-Za-z]+[0-9]\*}\_

**IZRAČUN FIRST & FOLLOW MNOŽIC**

FIRST( CAR\_RIDE ) = { carRide } FOLLOW( CAR\_RIDE ) = { 0 }

FIRST( DECLARATIONS ) = { const, var, ε } FOLLOW( DECLARATIONS ) = { road }

FIRST( DECLARATION ) = { const, var } FOLLOW( DECLARATION ) = { const, var, road }

FIRST( VAR\_DECL ) = { var } FOLLOW( VAR\_DECL ) = { const, var, road }

FIRST( POINT\_DECL ) = { const } FOLLOW( POINT\_DECL ) = { const, var, road }

FIRST( ROAD ) = { road } FOLLOW( ROAD ) = { car }

FIRST( PATH ) = { line, bend, ε } FOLLOW( PATH ) = { }; }

FIRST( LINE ) = { line } FOLLOW( LINE ) = { line, bend, }; }

FIRST( BEND ) = { bend } FOLLOW( BEND ) = { line, bend, }; }

FIRST( CAR ) = { car } FOLLOW( CAR ) = { start }

FIRST( START ) = { start } FOLLOW( START ) = { finish }

FIRST( FINISH ) = { finish } FOLLOW( FINISH ) = { crossSection , roundabout, gasStation, electricStation, parking, passenger, }, }; }

FIRST( CROSS ) = { crossSection, ε } FOLLOW( CROSS ) = { roundabout, gasStation, electricStation, parking, passenger, } }

FIRST( BOX ) = { box } FOLLOW( BOX ) = { }; }

FIRST( ROUND ) = { roundabout, ε } FOLLOW( ROUND ) = { gasStation, electricStation, parking, passenger, } }

FIRST( CIRC ) = { circ } FOLLOW( CIRC ) = { }; }

FIRST( POINTS ) = { ( } FOLLOW( POINTS ) = { let }

FIRST( POINT ) = { ( } FOLLOW( POINT ) = { ,, ;, };, ); }

FIRST( FILTER ) = { let } FOLLOW( FILTER ) = { }; }

FIRST( FOREACH ) = { foreach } FOLLOW( FOREACH ) = { }; }

FIRST( GAS ) = { gasStation, ε } FOLLOW( GAS ) = { electricStation, parking, passenger, } }

FIRST( ELECTRICITY ) = { electricStation, ε } FOLLOW( ELECTRICITY ) = { parking, passenger,}}

FIRST( PARKING ) = { parking, ε } FOLLOW( PARKING ) = { passenger,} }

FIRST( PASSENGERS ) = { passenger, ε } FOLLOW( PASSENGERS ) = { } }

FIRST( PASSENGER ) = { passenger } FOLLOW( PASSENGER ) = { passenger, } }

FIRST( EXPR ) = { +, -, real, var, ( } FOLLOW( EXPR ) = { ,, ), );, ; }

FIRST( ADDITIVE ) = { +, -, real, var, ( } FOLLOW( ADDITIVE ) = { ,, ), );, ; }

FIRST( ADDITIVE' ) = { +, -, ε } FOLLOW( ADDITIVE' ) = { ,, ), );, ; }

FIRST( MULTIPLICATIVE ) = { +, -, real, var, ( } FOLLOW( MULTIPLICATIVE ) = { +, -, ,, ), );, ; }

FIRST( MULTIPLICATIVE' ) = { \*, /, ε } FOLLOW( MULTIPLICATIVE' ) = { +, -, ,, ), );, ; }

FIRST( UNARY ) = { +, -, real, var, ( } FOLLOW( UNARY ) = { \*, /, +, -, ,, ), );, ; }

FIRST( PRIMARY ) = { int, real, var, ( } FOLLOW( PRIMARY ) = { \*, /, +, -, ,, ), );, ; }