Student:Relja Ivanovic

Email:rivanovic10120rn@raf.rs
Sacuvan kao: rivanovic10120rn@raf.rs (Relja Ivanovic)

Projekat iz predmeta Programski prevodioci, 2021/2022

[7 poena] Kreirati jflex specifikaciju za generisanje leksičkog analizatora jezika koji je definisan gramatikom G.

[10 poena] Napisati cup specifikaciju za zadatu gramatiku G.

G:

Program → **main ( )** *Block*
Block → **{** *VarList StatementList* **}**
VarList → *VarList VarDecl* | *VarDecl*
VarDecl → *NameList* **:** *Type***;**
NameList → *NameList* **, ID** | **ID**
Type → **int** | **char** | **real** | **bool**
StatementList → *StatementList Statement* | *Statement*
Statement → **do** *Statement* **while** *Expression*
| **ID** = *Expression* **;**
| **read(ID) ;**
| **write (** *Expression* **) ;**
| *Block*
Expression → *Expression* || *AndExpression* | *AndExpression*
AndExpression → *AndExpression* **&&** *RelExpression* | *RelExpression*
RelExpression → *ArExpression RelOp ArExpression* | *ArExpression*
RelOp → **<** | **<=** | **==** | **!=** | **>** | **>=**
ArExpression → *ArExpression* + *Term* | *ArExpression* - *Term* | *Term*
Term → *Term* * *Factor* | *Term* / *Factor* | *Factor*
Factor → **ID** | **CONST** | **(** *Expression* **)**

Terminalni simbol ID u ovom programskom jeziku označava identifikator (niz slova, cifara, i '_' u kojem prvi znak ne može da bude cifra), a simbol CONST konstantu koja moze da bude zadata u jednom od sledećih formata:

1. Konstante tipa **int**:
[$]<niz_cifara_zadate_osnove>

Pri čemu ako je znak $ naveden radi se o osnovi 16, a ukoliko nije naveden podrazumeva se osnova 10.

2. Konstante tipa **real**:
<niz_cifara>.[<niz_cifara>][E[±]<niz_cifara>]
ili
. <niz_cifara>[E[±]<niz_cifara>]

3. Konstante tipa **char**:
'<znak>'

4. Konstante tipa **bool:**
true i **false**

Komentari u ovom programskom jeziku počinju simbolom -- i završavaju se simbolom --.

Dopuniti napisanu cup specifikaciju tako da generisani analizator prijavi semantičke greške u kodu. Semantička pravila jezika data su u nastavku.

[1 poen] Jedno ime u jednoj oblasti važenja može biti definisano najviše jednom.

[1 poen] Ukoliko su u nekoj tački postoji veći broj definicija istog imena, validna je ona koja se nalazi na najdubljem nivou ugnježđenja.
[1 poen] Ne može se koristi promenljiva koja nije deklarisana.
[1 poen] Ne može se koristiti vrednost promenljive koja nije inicijalizovana.
[1 poen] *Expression* koji se koristi kao uslov **do while** petlje mora biti tipa **bool**.
[1 poen] Promenljivoj se može dodeliti samo vrednost izraza istog tipa.
[1 poen] Aritmetički operatori se mogu primeniti nad operandima numeričkog tipa (**char**, **int** ili **real**) pri čemu je bilo koja operacija moguća samo nad operandima istog tipa i tip rezultata je jednak tipu operanada.
[1 poen] Relacioni operatori se mogu primeniti nad operandima numeričkog tipa, a rezultat je tipa **bool**.
[1 poen] Logički operatori se primenjuju nad operandima tipa **bool** i rezultat je istog tipa.

[2 poena] Napisati ili generisati 20 programskih kodova pisanih u gramatici G i izvršiti njihovo prevodjenje korišćenjem generisanog analizatora.

[2 poena] Dopuniti cup specifikaciju da za dati ulazni kod kreira apstraktno sintaksno stablo (AST) na osnovu koga se generiše međukod niskog nivoa. Za implementaciju AST-a koristiti paket *AST* koji je dat na Google Drive-u. Koristiti skraćenu verziju gramatike i skup instrukcija hipotetičkog međukoda koji su dati u nastavku dokumenta.

Skraćena verzija G (koristi se za AST generisanje međukoda):

Program → **main ( )** *Block*
Block → **{** *VarList StatementList* **}**
VarList → *VarList VarDecl* | *VarDecl*
VarDecl → *NameList* **:** *Type* **;**
NameList → *NameList* **, ID** | **ID**
Type → **int** | **char** | **real** | **bool**
StatementList → *StatementList Statement* | *Statement*
Statement → **do** *Statement* **while** *Expression*
| **ID** = *Expression* **;**
| *Block*
Expression → *Expression* **||** *AndExpression* | *AndExpression*
AndExpression → *AndExpression* **&&** *RelExpression* | *RelExpression*
RelExpression → *ArExpression RelOp ArExpression* | *ArExpression*
RelOp → **<** | **==** | **>**
ArExpression → *ArExpression* **+** *Term* | *ArExpression* **-** *Term* | *Term*
Term → *Term* ***** *Factor* | *Term* **/** *Factor* | *Factor*
Factor → **ID** | **CONST** | **(** *Expression* **)**

Instrukcije međukoda niskog nivoa

Instrukcija	**Značenje**
Load_Const Rn, c	(Rn) = c
Load_Mem Rn, x	(Rn) = x
Load_Arr Rn, x[Rm]	(Rn) = x[(Rm)]
Store Rn, x	(x) = (Rn)
Add Rn,Rm	(Rn) = (Rn) + (Rm)
Sub Rn,Rm	(Rn) = (Rn) - (Rm)

Mul Rn,Rm	(Rn) = (Rn) * (Rm)
Div Rn,Rm	(Rn) = (Rn) / (Rm)
Compare_Equal Rn,Rm	(Rn) = (Rn) == (Rm)
Compare_Greater Rn,Rm	(Rn) = (Rn) > (Rm)
Compare_Less Rn,Rm	(Rn) = (Rn) < (Rm)
And Rn,Rm	(Rn) = (Rn) ∧ (Rm)
Or Rn,Rm	(Rn) = (Rn) ∨ (Rm)
Jump lab	skok na naredbu sa oznakom *lab*
JumpIfZero Rn, lab	skok na naredbu sa oznakom *lab* ukolikoko je (Rn)=0
JumpIfNotZero Rn, lab	skok na naredbu sa oznakom *lab* ukolikoko je Rn)≠0

Pretpostaviti da se logičke vrednosti true i false predstavljaju celobrojnim vrednostima 1 i 0, respektivno.