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 →***program ID***Block*

*Block***→ begin***VarSection**StatementList***end**

*VarSection***→***VarSection VarDecl*|*VarDecl*

*VarDecl***→***Type NameList***;**

*NameList***→***NameList***, ID**|**ID**

*Type***→ int**|**real | boolean**

*StatementList***→***StatementList***;***Statement*|*Statement*

*Statement***→ for (***Expression***,***Expression***,***Expression***)***Statement*

**|***Assignment*

**|****read(***NameList***)**

|**write (***ExpressionList***)**

**|***Block*

*Assignement***→ ID***= Expression*

*Expression***→***Expression***OR***AndExpression | AndExpression*

*AndExpression***→***AndExpression***AND***RelExpression*| *RelExpression*

*RelExpression***→***ArExpression**RelOp ArExpression***|***ArExpression*

*RelOp*→**<**|**<=**| **==**| **<>**| **>**| **>=**

*ArExpression***→***ArExpression***+***Term*|*ArExpression***-***Term*|*Term*

*Term***→***Term***\****Factor*|*Term***/***Factor*|*Factor*

*Factor***→ ID | CONST | (***Expression***)**

*ExpressionList***→***ExpressionList***,***Expression*|*Expression*

program a

begin

var int i = 5;

for(int j = 0 , j<=i, j+1) write(i)

int k=i+1;

write(k)

real r = k+i+0.5;

write(r)

end

Terminalni simbol ID u ovom programskom jeziku označava identifikator (niz slova I cifara 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**:

[<oznaka\_osnove>]<niz\_cifara\_zadate\_osnove>

Pri čemu oznaka osnove može biti

0 - za brojni sistem sa osnovom 8,

0x - za brojni sistem sa osnovom 16,

Ukoliko je oznaka osnove izostavljena, podrazumeva se osnova 10.

2. Konstante tipa **real**:

0. <niz\_cifara>[E[±]<niz\_cifara>]

3. Konstante tipa **boolean**:

TRUE ili FALSE

Komentari u ovom programskom jeziku počinju simbolom // i završavaju se prelazom na novi red.

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 ugnježdenja.

[1 poen] Ne može se koristi promenljiva koja nije deklarisana.

[1 poen] Ne može se koristiti vrednost promenljive koja nije inicijalizovana.

[1 poen] Uslov ponavljanja petlje (drugi izraz u zaglavlju petlje) je izraz tipa **boolean**.

[1 poen] Promenljivoj tipa **boolean**se može dodeliti samo vrednost izraza tipa **boolean** dok se kod dodele vrednosti promenljivoj nekog numeričkog tipa može vršiti implicitna konverzija iz nižeg u viši tip.

[1 poen] Aritmetički operatori se mogu primeniti nad operandima numeričkog tipa (**int** ili **real**) pri čemu je dozvoljena „implicitna“ konverzija nižeg u viši tip tako da je rezultat uvek višeg tipa.

[1 poen] Relacioni operatori se mogu primeniti nad operandima numeričkog tipa, a rezultat je tipa **boolean**.

[1 poen] Logički operatori se primenjuju nad operandima tipa **boolean** 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***→ program ID***Block*

*Block***→ begin***VarSection**StatementList***end**

*VarSection* **→***VarSection VarDecl*|*VarDecl*

*VarDecl***→***Type NameList***;**

*NameList***→***NameList***, ID**|**ID**

*Type***→ int**|**real | boolean**

*StatementList***→***StatementList***;***Statement*|*Statement*

*Statement***→ for (***Expression***,***Expression***,***Expression***)***Statement*

| *Assignment*

| *Block*

*Assignement***→ ID***= Expression*

*Expression***→***Expression***OR***AndExpression | AndExpression*

*AndExpression***→***AndExpression***AND***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.