Linguaggi Formali e Compilatori
Proff. Breveglieri, Crespi Reghizzi, Morzenti
Prova scritta [1]: Domanda relativa alle esercitazioni
28/02/2013

COGNOME: .....
NOME: ..... Matricola: .....
Corso: ○ Laurea Specialistica ○ V. O. ○ Laurea Triennale ○ Altro: ...
Sezione: ○ Prof. Breveglieri ○ Prof. Crespi ○ Prof.Morzenti

Per la risoluzione della domanda relativa alle esercitazioni si deve utilizzare l'implementazione del compilatore `Acse` che viene fornita insieme al compito.

Si richiede di modificare la specifica dell'analizzatore lessicale da fornire a `flex`, quella dell'analizzatore sintattico da fornire a `bison` ed i file sorgenti per cui si ritengono necessarie delle modifiche in modo da estendere il compilatore `Acse` con la possibilità di gestire il tipo di dato *matrice*. Si consideri il seguente snippet di codice:

```
int matrix[3, 2]; // 3 rows, 2 columns
int i = 1;
matrix[i - 1, 0] = 1;
matrix[i - 1, 1] = 2;
matrix[i, 0] = matrix[i - 1, 0] * 2;
matrix[i, 1] = matrix[i - 1, 1] * 3;
```

Analogamente agli array, una matrice è dichiarata con dimensioni costanti. È possibile assegnare ed utilizzare il valore di una singola cella specificandone l'*indice di riga e di colonna*. Per implementare tale supporto è necessario *linearizzare* la matrice, ovvero mappare gli elementi della stessa in elementi di un array monodimensionale equivalente la cui lunghezza è pari al numero di elementi della matrice. **Si richiede che le matrici siano linearizzate per colonne**.

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \quad \Rightarrow \quad \begin{bmatrix} 1 & 4 & 2 & 5 & 3 & 6 \end{bmatrix}$$

A tal fine si assuma che:

- nelle strutture `t_axe_variable` e `t_axe_declaration` siano presenti anche i *campi di tipo intero* `isMatrix`, `rows`, `columns`, utilizzabili per tracciare le informazioni del tipo matrice,
- internamente il compilatore supporti tali informazioni e le propaghi correttamente.

[1] Tempo 60'. Libri e appunti personali possono essere consultati.
E' consentito scrivere a matita. Scrivere il proprio nome sugli eventuali fogli aggiuntivi.

1. Definire i token (e le relative dichiarazioni in `Acse.lex` e `Acse.y`) necessari per ottenere la funzionalità richiesta. (1 punto)
2. Definire le regole sintattiche (o le modifiche a quelle esistenti) necessarie per ottenere la funzionalità richiesta. (3 punti)
3. Definire le azioni semantiche (o le modifiche a quelle esistenti) necessarie per ottenere la funzionalità richiesta. (16 punti)

   La soluzione è riportata nella patch allegata.

4. Si consideri la seguente funzione:

```
void createVariable(t_program_infos *program, char *ID, int type,
                    int isArray, int arraySize, int init_val) {
  t_axe_variable *var;

  if (program == NULL)
     notifyError(AXE_PROGRAM_NOT_INITIALIZED);

  var = alloc_variable(ID, type, isArray, arraySize, init_val);
  if (var == NULL)
     notifyError(AXE_OUT_OF_MEMORY);

  var->labelID = newLabel(program);

  addVariable(program, var);
}
```

Figura 1: Funzione `createVariable`.

e il seguente snippet di codice estratto dalla funzione `set_new_variables`:

```
while (current_element != NULL) {
  current_decl = (t_axe_declaration *) LDATA(current_element);
  if (current_decl == NULL) {
     free_new_variables(variables);
     notifyError(AXE_NULL_DECLARATION);
  }

  createVariable(program, current_decl->ID, varType,
                 current_decl->isArray, current_decl->arraySize,
                 current_decl->init_val);

  current_element = LNEXT(current_element);
}
```

Figura 2: Porzione di codice di `set_new_variables`.

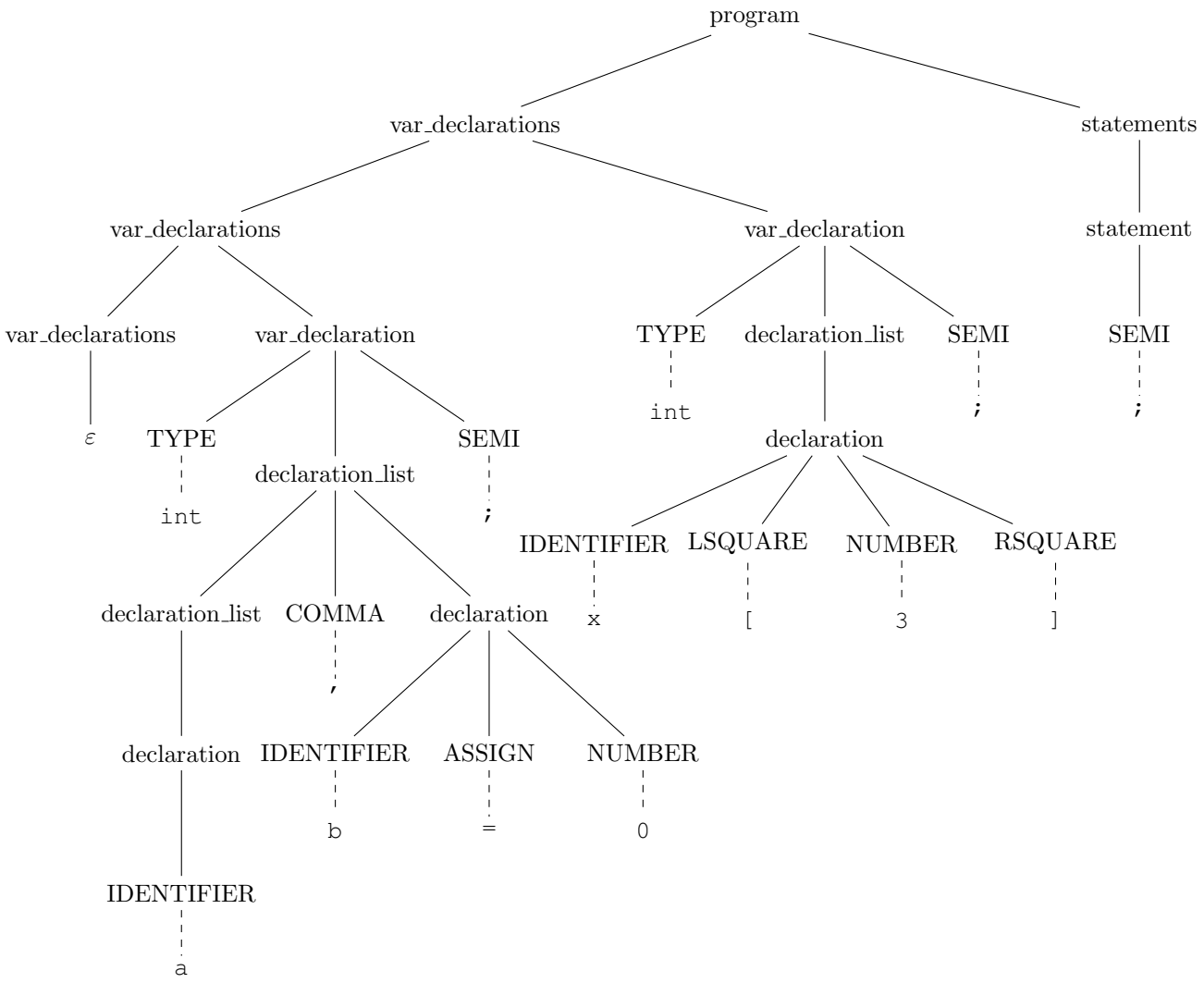
Si proponga una modifica dello snippet estratto dalla funzione `set_new_variables` al fine di gestire correttamente il tipo matrice. (5 punti)

La soluzione è riportata nella patch allegata.

5. Data il seguente snippet di codice `Lance`:

```
int a, b = 0; int x[3]; ;
```

Scrivere l'albero sintattico relativo partendo dalla grammatica Bison definita in `Acse.y`. (5 punti)



6. (**Bonus**) Descrivere come modificare la soluzione proposta in modo da estendere il supporto al tipo matrice rendendo possibile la specifica del tipo di linearizzazione desiderata (per righe o colonne) al momento della dichiarazione:

```
int matrix_A[3, 5] by rows;
int matrix_B[3, 5] by cols;
```