PROLE-Practica-3-Bison.pdf

Tori_

Procesadores del Lenguaje

3º Grado en Ingeniería Informática

Escuela Politécnica Superior
Universidad de Burgos

Práctica 3: Bison

Procesadores de lenguajes, Universidad de Burgos

Enunciado

Enunciado de la tercera práctica para la segunda convocatoria de PL.

Se trata de utilizar bison para obtener un compilador que traduzca un lenguaje de alto nivel a código de la máquina de pila abstracta.

El lenguaje de alto nivel es muy sencillo. No tiene declaración de tipos. Y el único tipo que permite es el tipo entero. En las condiciones de las instrucciones if y while el valor 0 se interpreta como falso y cualquier otro valor como cierto.

Además de la asignación normal (<-), tiene incremento unario '++' y decremento unario '--'

Los comentarios pueden ser de dos tipos:
- en línea comenzando con el carácter '#'.
- multilínea comenzando con la secuencia '<#' y finalizando con '#>'.

El terminal 'NUM' representa un número entero e 'ID' un identificador/variable del lenguaje (pueden contener números, letras y guiones bajos, pero no pueden comenzar por número).

Debe ser capaz de leer por entrada estándar (teclado) y por un fichero que se le pase por argumento.

La gramática del lenguaje es la siguiente:

```
  stmts -> (stmt ';')+
   stmt -> 'while' '(' expr ')' stmts 'endwhile'
      | 'if' '(' expr ')' stmts elseopt
      | 'print' '(' expr ')'
      | assig
elseopt -> 'elseif' '(' expr ')' stmts elseopt
      | 'else' stmts 'endif'
      | 'endif'
  assig -> ID '<-' expr
      | ID '++'
      | ID '--'
   expr -> expr '+' mult
      | expr '-' mult
      | mult
   mult -> mult '*' val
      | mult '/' val
      | val
    val -> NUM
      | ID
      | '(' expr ')'
```

NOTA: Podría ser necesaria alguna transformación en la la gramática (eliminación de recursividad, factorización...) antes de empezar a programar en JavaCC. De hecho, la transformación también podría ser la utilización de los operadores avanzados de JavaCC, como: '?', '*'...

Ante una entrada como:

```
Var <- 5 * 25;    # la variable var toma el valor de 5 *
25
while (Var)
  Var <- 2*v;
endwhile;
if (5 - Var)     # Condicional
            # Cuerpo con varias instrucciones
separadas por ;
 v --;
 print (v * 3) ;   # Imprimir valor expresión
else
 print(Var + 10); # Imprimir Var + 10
endif;
```

Debería mostrar (excepto quizá el número de las etiquetas):

```
        valori Var
        mete 5
        mete 25
        mul
        asigna
LBL0
        valord Var
        sifalsovea LBL1
        valori Var
        mete 2
        valord v
        mul
        asigna
        vea LBL0
LBL1
        mete 5
        valord Var
        sub
        sifalsovea LBL2
        valori v
        valord v
        mete 1
        sub
        asigna
        valord v
        mete 3
        mul
        print
        vea LBL3
LBL2
        valord Var
        mete 10
        sum
        print
LBL3
```

Ante una entrada como:

```
# Ejemplo de while
while (7 - 5 + a)
 if (c - d) print (c);
 else print (d);
 endif;
endwhile;
```

Debería mostrar (excepto quizá el número de las etiquetas):

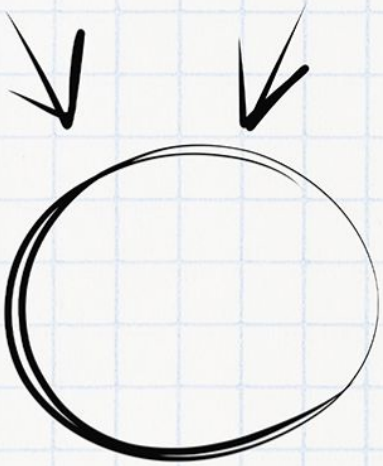
```
LBL0
        mete 7
        mete 5
        sub
        valord a
        sum
        sifalsovea LBL1
        valord c
        valord d
        sub
        sifalsovea LBL2
        valord c
        print
        vea LBL3
LBL2
        valord d
        print
LBL3
        vea LBL0
LBL1
```

Imagínate aprobando el examen

Necesitas tiempo y concentración

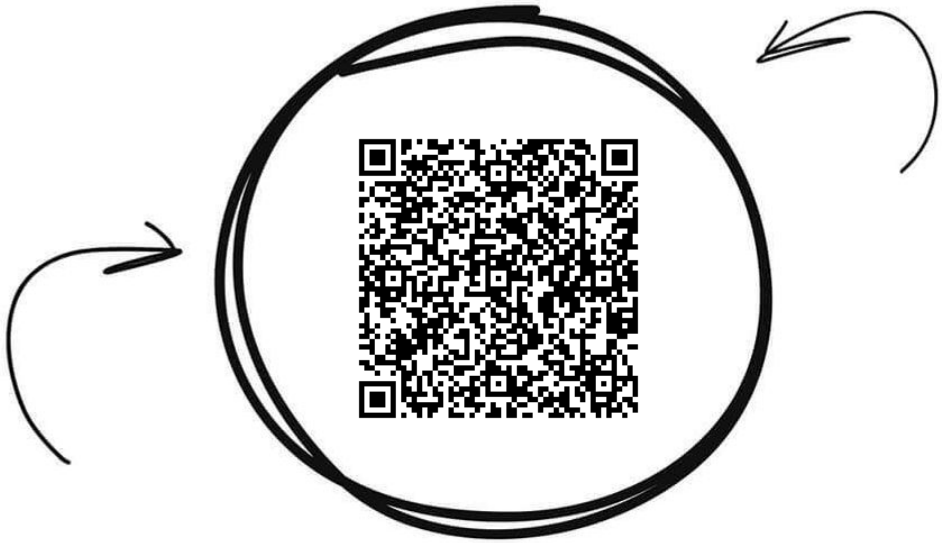
Planes	PLAN TURBO	PLAN PRO	PLAN PRO+
Descargas sin publi al mes	10	40	80
Elimina el video entre descargas	✓	✓	✓
Descarga carpetas	✗	✓	✓
Descarga archivos grandes	✗	✓	✓
Visualiza apuntes online sin publi	✗	✓	✓
Elimina toda la publi web	✗	✗	✓
Precios Anual	0,99 € / mes	3,99 € / mes	7,99 € / mes

Ahora que puedes conseguirlo, ¿Qué nota vas a sacar?



WUOLAH

Procesadores del Lenguaje



Banco de apuntes de la

WUOLAH

Comparte estos flyers en tu clase y consigue más dinero y recompensas

1. Imprime esta hoja
2. Recorta por la mitad
3. Coloca en un lugar visible para que tus compis puedan escanar y acceder a apuntes
4. Llévate dinero por cada descarga de los documentos descargados a través de tu QR

Ante una entrada como:

```
<# if ... elseif ... fin
#>
if (a)
  print (1);
elseif (b)
  print (2);
else
  print (3);
endif ;
```

Debería mostrar (excepto quizá el número de las etiquetas):

```
        valord a
        sifalsovea LBL0
        mete 1
        print
        vea LBL1
LBL0
        valord b
        sifalsovea LBL2
        mete 2
        print
        vea LBL1
LBL2
        mete 3
        print
LBL1
```

* Podría reutilizar una única etiqueta para salir (como en
el ejemplo anterior) o acumular una etiqueta de salida por
cada elseif que apareciera.

```
  valord a
  sifalsovea LBL0
  mete 1
  print
  vea LBL1
LBL0:
  valord b
  sifalsovea LBL2
  mete 2
  print
  vea LBL3
LBL2:
  mete 3
  print
LBL3:
LBL1:
```

Resolución

analex

```
/*
 * Practica3.l: Analizador léxico que pasa los tókenes al analizador sintáctico generado por bison.
 * @author REV
 * @date 30-01-2024
 * @version 1.0
 */

%{
 #include "y.tab.h"
%}
%option noyywrap
%x COM
%%

"<#"        {BEGIN(COM);}   //Comentarios multilínea
<COM>"#>"   {BEGIN(INITIAL);}
<COM>.|\n   ;

"print"     return PRINT;
"while"     return WHILE;
"endwhile"  return ENDWHILE;
"if"        return IF;
"else"      return ELSE;
"elseif"    return ELSEIF;
"endif"     return ENDIF;
"<-"        return ASIGNAR;
"++"        return INCRASIG;
"--"        return DECRASIG;

[0-9]+                  {yylval.num = atoi(yytext); return NUM;}
[A-Za-z_][0-9A-Za-z_]*  {yylval.id = strdup(yytext); return ID;}

" "|\t|\n|\r    ;   // Ignorar espacios, tabuladores, saltos de línea
"#".*\n         ;   // Ignorar comentarios de linea
.               return yytext[0];   // Devolver cualquier otro caracter
```

Anasin

```
/* Practica3.y: Compilador que traduce un lenguaje de alto nivel a código de la máquina de pila abstracta mediante bison.
 * @author REV
 * @date 30-01-2024
 * @version 1.0
 */
%{
    #include <stdio.h>
    #include <stdlib.h>

    #ifdef YYDEBUG
    int yydebug = 1;
    #endif

    extern int yyparse();
    extern FILE* yyin;

    void yyerror(const char* s); // Declaración de la función de manejo de errores

    // Función para generar números únicos para las etiquetas
    int getNextNumber() {
        static int nextNumber = -1;
        return ++nextNumber;
    }

%}

%union {
    char *id; // Para almacenar identificadores
    int num; // Para almacenar números
}

// Definición de tokens y tipos
%token <id>ID <num>NUM PRINT WHILE ENDWHILE IF ELSE ELSEIF ENDIF ASIGNAR INCRASIG DECRASIG
%type <num>LBL

// Definición de precedencia de operadores
%left '+' '-'
%left '*' '/'

%%
// Reglas de producción

stmts: stmt ';'
    | stmt ';' stmts
    ;

stmt: WHILE
        LBL // etiqueta de inicio de bucle
        LBL // etiqueta fin del bucle
        {printf("LBL%d \n", $<num>2);}
        '(' expr ')'
```

```
        {printf("\tsifalsovea LBL%d\n", $<num>3);}
        stmts ENDWHILE
        {printf("\tvea LBL%d \n", $<num>2);}
        {printf("LBL%d \n", $<num>3);}
    | IF
        LBL // etiqueta else nueva
        LBL // etiqueta fin nueva
        '(' expr ')'
        {printf("\tsifalsovea LBL%d\n", $<num>2);}
        stmts
        {printf("\tvea LBL%d \n", $<num>3);}
        {printf("LBL%d \n", $<num>2);}
        {$<num>$=$<num>3;}  // referencia la etiqueta de fin
        elseopt
    | PRINT expr {printf("\tprint\n");}
    | assig
    ;

elseopt: ELSEIF
        LBL // etiqueta else nueva
        '(' expr ')'
        {printf("\tsifalsovea LBL%d \n", $<num>2);}
        stmts
        {printf("\tvea LBL%d \n", $<num>0);}
        {printf("LBL%d \n", $<num>2);}
        {$<num>$=$<num>0;} // referencia la etiqueta de fin
        elseopt
    | ELSE stmts ENDIF {printf("LBL%d \n", $<num>0);}
    | ENDIF {printf("LBL%d \n", $<num>0);}
    ;

assig: ID {printf("\tvalori %s\n", $1);} assig_p {printf("\tasigna\n");}
    ;

assig_p: ASIGNAR expr
    | INCRASIG {printf("\tvalord %s\n\tmete 1\n\tsum\n",$<id>-1);}
    | DECRASIG {printf("\tvalord %s\n\tmete 1\n\tsub\n",$<id>-1);}
    ;

expr: expr '+' mult {printf("\tsum\n");}
    | expr '-' mult {printf("\tsub\n");}
    | mult
    ;

mult: mult '*' val  {printf("\tmul\n");}
    | mult '/' val  {printf("\tdiv\n");}
    | val
    ;

val:  NUM   {printf("\tmete %d\n", $<num>1);}
    | ID    {printf("\tvalord %s\n", $<id>1);}
    | '(' expr ')'
    ;
```

```
// Produccion auxiliar para asignar los numeros de las etiquetas.
LBL : { $<num>$ = getNextNumber();} ;

%%

// Función para manejar errores de análisis
void yyerror(const char* s) {
    fprintf(stderr, "Error: %s\n", s);
    exit(1);
}

// Función principal que inicia el análisis
int main(int argc, char **argv) {
    if (argc > 1) {
        FILE* file = fopen(argv[1], "r");
        if (!file){
            fprintf(stderr,"Error con el fichero: %s \n", argv[1]);
            exit(1);
        }
        yyin=file;
    }else{
        printf("\nIntruduzca una expresión:\n");
        yyin = stdin;
    }
    yyparse(); // Calls yylex() for tokens.
    return 0;
}
```