Tecnológico de Monterrey

Desarrollo de aplicaciones avanzadas de ciencias computacionales

Sergio Lopez Urzaiz A00827462

Mini-Proyecto parte 3

Profesora:

Elda G. Quiroga González, M.Sc.

Descripción de pilas y filas

Para poder resolver esta problemática, tuve que agregar varias pilas y filas que estuvieron trabajando en conjunto para diversas operaciones. La fila principal era la fila de cuadruplos, la cual básicamente servía como una centralización de toda la información para tanto funcionalidad como despliegue final. En esta se guardan los pasos de las expresiones y los saltos tanto crudos como ya procesados.

En cuanto a expresiones, simplemente bastó con tener una pila de operadores y una pila de operandos. Las reglas de gramática formal que habíamos previamente declarado hacían la lógica de jerarquía por sí solas. Esto facilita el proceso de resolución de expresiones ya que el parser literalmente estará encontrando expresiones con la prioridad más alta y descendiendo a la más baja. Entonces, simplemente tenemos que decirle a nuestro código vaya resolviendo operaciones a medida que tenga dos operadores y un operador para luego guardar el proceso en la fila de cuádruplos. El resultado lo terminamos guardando en una variable temporal que tiene la estructura $T + 'count'$ para luego poder ser utilizada por otro tipo de operaciones en el código.

Lo complicado de esta problemática fue los saltos. Tuve que crear 3 pilas que estaban trabajando de la mano en todo momento para el funcionamiento, pero no siempre son usadas las 3. En el caso de condiciones como *if* y *else*, se utilizaban dos pilas: pila de saltos pendientes y pila de saltos. Los nombres son muy similares, pero tenemos que tener estas dos pilas para poder tener un funcionamiento completo. La razón de esto es que el parser lee una *condición* como un bloque completo, permitiéndonos entonces hacer “operaciones” únicamente tanto al principio como al final. Pensemos entonces en la manera que se procesa un salto. Primero es introducido a la fila de cuadruplos con el tipo de salto y la condición de la cual depende (si es que depende). Eventualmente se agrega hacía que cuádruplo debe hacer el salto. El punto clave es la condición. Primero tenemos que introducir a nuestra fila de cuadruplos todas las operaciones para evaluar la condición y luego podemos introducir el salto. Es esta la razón por la que tenemos la pila de saltos pendientes. Al acabar una expresión y entrar al cuerpo de un *statement*, tenemos que checar que no haya saltos pendientes por procesar. Si es que existe uno, lo introducimos a la lista de cuádruplos. Al ya estar dentro de la lista de cuadruplos, lo podemos introducir ahora si a la pila de saltos formal. La única diferencia es que dentro de esta pila únicamente tendremos el tipo de salto y su posición en la fila de cuádruplos. Al terminar una *condición*, podemos entonces procesar ese salto pendiente y usar la posición guardada para saber dónde debe ir el código en caso de cumplirse la condición (si es que depende). Ahora, para el caso de ciclos es bastante parecido, pero es al revés. En vez de ir hacia adelante en el código, nuestro ciclo regresará hacia un

punto anterior. Es aquí donde entra la tercera pila: la pila de migajas de pan. Cada vez que entremos a un *ciclo*, tenemos que introducir estas migajas de pan a la pila con la posición en la que actualmente nos encontramos. Entonces, imaginemos que el proceso es el mismo - se introduce el salto pendiente, se pasa a la lista de saltos y luego se llega al final de un *ciclo* (a diferencia de una *condición* como en el *if* y *else*). Entonces, a la hora de terminar de pulir nuestro salto, en vez de usar el tamaño de la fila de cuadruplos, tenemos que consultar nuestra pila de migajas y usar el valor hasta arriba de la pila como nuestro siguiente paso en el código.

Pruebas

```
program_string = '''program pt2; var a, b, c, d, e: int; {
                    do {
                        a = b / c;
                    }
                    while (a + b);
                } end'''
```

```
----- QUADRUPLES -----
1 ['/', 'b', 'c', 'T1']
2 ['=', 'T1', 'a', 'T2']
3 ['+', 'a', 'b', 'T3']
4 ['goToT', 'T3', 1]
----------------------
```

```
program_string = '''program pt2; var a, b, c, d, e: int; {
                    if (a > b) {
                    }
                    else {
                        b = a;
                    };
                } end'''
```

```
----- QUADRUPLES -----
1 ['>', 'a', 'b', 'T1']
2 ['goToF', 'T1', 4]
3 ['goTo', '', 5]
4 ['=', 'a', 'b', 'T2']
----------------------
```

```
program_string = '''program pt2; var a, b, c, d, e: int; {
                    if (a > b) {
                        a = b * c;
                        if (a > b) {
                            a = b * c;
                        } ;
                    } ;
                } end'''
```

```
----- QUADRUPLES -----
1 ['>', 'a', 'b', 'T1']
2 ['goToF', 'T1', 9]
3 ['*', 'b', 'c', 'T2']
4 ['=', 'T2', 'a', 'T3']
5 ['>', 'a', 'b', 'T4']
6 ['goToF', 'T4', 9]
7 ['*', 'b', 'c', 'T5']
8 ['=', 'T5', 'a', 'T6']
----------------------
```

```
program_string = '''program pt2; var a, b, c, d, e: int; {
                        if (a > b) {
                            a = b * c;
                            if ( a > b) {
                                a = b * c;
                            };
                            do {
                                a = b * c;
                            } while (a > b);
                        } ;
                    } end'''
```

i

```
program_string = '''program pt2; var a, b, c, d, e: int; {
                            if (a > b) {
                                a = b * c;
                                do {
                                    a = b * c;
                                    if ( a > b) {
                                        a = b * c;
                                    }
                                    else {
                                        a = b * c;
                                    };
                                } while (a > b);
                            } ;
                        } end'''
```

d

```
----- QUADRUPLES -----
1 ['>', 'a', 'b', 'T1']
2 ['goToF', 'T1', 16]
3 ['*', 'b', 'c', 'T2']
4 ['=', 'T2', 'a', 'T3']
5 ['*', 'b', 'c', 'T4']
6 ['=', 'T4', 'a', 'T5']
7 ['>', 'a', 'b', 'T6']
8 ['goToF', 'T6', 12]
9 ['*', 'b', 'c', 'T7']
10 ['=', 'T7', 'a', 'T8']
11 ['goTo', '', 14]
12 ['*', 'b', 'c', 'T9']
13 ['=', 'T9', 'a', 'T10']
14 ['>', 'a', 'b', 'T11']
15 ['goToT', 'T11', 5]
----------------------
```