Compiladores
Laboratorio 8

Objetivo

Desarrollar un intérprete para una gramática que incluya sentencias de control, utilizando el patrón de diseño Visitor.

Programa
Se tiene implementado la siguiente gramática,

- Program ::= StmtList
- StmtList := Stmt ('**;**' Stmt)*
- Stmt ::= **id** '**=**' Exp | '**print**' '**(**' Exp '**)**'
- Exp ::= Term ((´**+**´ | ´**-**´) Term)*
- Term ::= Factor ((´*****´|´**/**´) Factor)*
- Factor ::= **id** | **Num** | ´**(**´ Exp ´**)**´

Esta gramática define un lenguaje sencillo que permite asignaciones a variables, impresión de expresiones y operaciones aritméticas básicas, respetando la jerarquía de operadores. Donde podemos utilizar el siguiente comando para compilar:

```
g++ main.cpp exp.cpp parser.cpp scanner.cpp token.cpp visitor.cpp
```

Problema
Modificar la gramática para acetar operaciones de relación y sentencias de control selectivo:

- Program ::= StmtList
- StmtList ::= Stmt **(;** Stmt **)***
- Stmt ::= **id =** CExp |
  print (CExp **)**
  if CExp **then** StmtList [**else** StmtList] **endif**
- CExp ::= Exp [(**<**|**<=**|**==**) Exp]
- Exp ::= Term ((**+** | **-**) Term)*
- Term ::= Factor ((*****|**/**) Factor)*
- Factor ::= **id** | **Num** | **(** Exp **)**

Ayuda:

1. **Agregar nuevos tokens**
   Incorporar los tokens necesarios (MENOR, IF, ELSE, ENDIF, etc.).

2. **Modificar el scanner**
   Asegurarse de que el scanner reconozca correctamente los nuevos tokens definidos.

3. **Verificación de tokens**
   Revisar que la lista de tokens reconocidos esté completa, actualizada y sea consistente con la gramática extendida.

4. **Operaciones de comparación**
   Las operaciones de comparación son binarias y devuelven 0 o 1, por ende, no es necesario definir una nueva clase para ellas; pueden manejarse con las clases existentes de expresiones binarias.

5. **Agregar clase IFStatement**
   Definir una nueva clase IFStatement con los siguientes atributos:
   - CExp* condition;
   - list<Stm*> slist1;
   - list<Stm*> slist2;

6. **Modificar el Visitor**
   Incluir las funciones correspondientes para visit(IFStatement*), inicialmente con implementaciones vacías.

7. **Modificar el parser**
   Modificar la función parseStatement() para que reconozca correctamente la estructura del if, incluyendo sus ramas then y else.

8. **Modificar el visitor**
   Completar los métodos:
   - visit(AssignStatement* stm)
   - visit(PrintStatement* stm)