Diseño y Construcción de Compiladores - 2023

Guía para abordar el Práctico de Máquina

A continuación, se presentan una serie de ejercicios que le servirán como guía a lo largo de las entregas del práctico de máquina. Tenga en cuenta que al comienzo tal vez no pueda responder claramente algunos de los ítems. Sin embargo, los debería poder responder conforme vaya profundizando su conocimiento en la materia. Por lo que se le recomienda repasar los ejercicios de esta guía y sus respuestas en cada entrega.

1. Analice la gramática dada y el sublenguaje de C++ que genera. Hágase preguntas como:

   ¿Se permiten prototipos de funciones?

   ¿Se puede declarar variables en cualquier parte?

   ¿Cómo está formado un identificador válido?

   ¿Se permite el anidamiento de funciones?

   ¿Cuál es la sintaxis para declarar una variable? Entre otras.

2. La gramática ¿es apta para construir un PDR? Justifique su respuesta. En caso de no serlo ¿cómo se ha abordado esta situación?

3. Proponga programas generados a partir de la gramática, es decir, lotes de prueba sin errores sintácticos.

4. Compile los archivos fuentes entregados con el GNU C Compiler y a continuación (si todo salió bien), compile los lotes de prueba creados en el ejercicio anterior con el ejecutable generado (archivo **ucc**, el compilador que Ud. está diseñando y construyendo). ¿Qué obtuvo como salida?

5. Construya los grafos sintácticos y asocie cada función del PDR en el archivo parser.c con su correspondiente símbolo no terminal de la gramática.

6. ¾Cuál es el motivo por el que la producción <relación> no se encuentra especificada en el

parser? 7. ¿Por qué la primera sentencia en proposicion_retorno() es una llamada al scanner?

8. En el procedimiento especificador_declaracion(), ¿por qué CPYCOMA es una alternativa válida antes de llamar a declaracion_variable()?

9. Explique por qué declaracion_parametro() no está definida de la siguiente forma:

```
void declaracion_parametro(){
      especificador_tipo();
      if(lookahead_in(CAMPER))
            scanner();
      match(CIDENT,e);
      match(CCOR_ABR,e);
      match(CCOR_CIE,e);
}
```

10. Analice cada ítem de la semántica del sublenguaje C++ y establezca qué analizador es el encargado de controlarlo/implementarlo.