Lenguajes de programación

Documentación de la

Entrega 3C

Miembros del grupo L07:

Fernández Ángulo, Óscar

Martinez Arias, Miguel

Pastor Diaz, Guillermo

Rodríguez Ares, Silvia

1. Introducción
2. Entrada y salida del programa
3. Instrucciones de compilación y ejecución
4. Desarrollo del programa y aclaraciones
5. Posibles mejoras
6. Bibliografía

1. Introducción

El presente informe documenta el funcionamiento y desarrollo del programa de la práctica 3c. Se ha utilizado ANTLR, un potente analizador que acepta código de análisis léxico, sintáctico y semántico. Es decir, admite gramática, reglas léxicas y acciones semánticas para generar un analizador que puede leer, gestionar, ejecutar o traducir texto estructurado o archivos binarios.

El programa desarrollado procesa un fichero con código fuente del lenguaje de programación C, para generar una salida de tipo texto que muestre: las funciones declaradas y la relación de llamadas que hay entre ellas, al igual que las variables declaradas, y su ámbito.

2. Entrada y salida del programa

El programa tiene como entrada un código fuente correcto de un programa C. Es importante que el código sea correcto, ya que a la hora de desarrollar el analizador se ha tenido muy en cuenta. Aspectos como que no se puede declarar dos veces una variable con el mismo nombre o que en C puro no hay sobrecarga de funciones (mismo nombre de función con distintos parámetros), han determinado el desarrollo del programa.

La salida es un fichero de texto sin extensión llamado "*salida*". En él se muestran, en orden:

- Las variables globales encontradas.
- El nombre de cada función con:
    - el número y nombre de sus parámetros,
    - el número y nombre de las llamadas a funciones que realiza
    - y, el número y nombre de las variables locales que tiene definidas

A continuación se muestra un ejemplo de la entrada y salida del programa:

```
double uno;
int dos, tres=3;

void funcion1(){

      int a=2;
      if(1>2)
      a=3;
}

char c;

funcion2(double d){
      funcion1();
      int x=0;
      double d2;
      for(x=2;x<10;x=2){
            d2=d/x;
      }
}
main(){}
```

```
Variables globales: 4
        uno (double)
        dos (int)
        tres (int)
        c (char)
Funcion: funcion1
        Parametros: 0
        Llamadas a funciones: 0
        Variables locales: 1
                a (int)
Funcion: funcion2
        Parametros: 1
                d (double)
        Llamadas a funciones: 1
                funcion1
        Variables locales: 2
                x (int)
                d2 (double)
Funcion: main
        Parametros: 0
        Llamadas a funciones: 0
        Variables locales: 0
```

3. Instrucciones de ejecución y compilación

Para la compilación y ejecución del programa es crucial realizar una configuración previa. Dicha configuración varía dependiendo del sistema operativo, pero a grandes rasgos se trata de configurar la variable de entorno *classpath* y establecer alias para el comando *antlr4* y *grun*.

Configuración en Linux:

Se necesitar tener instalado Java (versión 1.6 o mayor).
Descargar la librería *antlr-4.5-complete.jar* y situarla en *C:\JavaLib*.
Ejecutar el siguiente script:
`$ export CLASSPATH=".:/usr/local/lib/antlr-4.5-complete.jar:$CLASSPATH"`
`$ alias antlr4='java -Xmx500M -cp "/usr/local/lib/antlr-4.5-complete.jar:$CLASSPATH" org.antlr.v4.Tool'`
`$ alias grun='java org.antlr.v4.runtime.misc.TestRig'`

Configuración en Windows:
Se necesitar tener instalado Java (versión 1.6 o mayor). Descargar la librería *antlr-4.5-complete.jar* y situarla en */usr/local/lib*. Agregar la ruta *C:\Javalib\antlr-4.5-complete.jar;* a la variable de entorno *classpath*. Crear dos archivos *.bat* y situarlos en la carpeta del sistema: *Antlr4.bat:* java org.antlr.v4.Tool %* *Grun.bat:* java org.antlr.v4.runtime.misc.TestRig %*

Para realizar una ejecución simple del programa hay que ejecutar los siguientes comandos:

`antlr4 p3b.g4`
`javac *.java`
`grun p3b prog prueba.c`
	p3b.g4 es el nombre del programa
	prog es el símbolo inicial de la gramática
	prueba.c es el fichero de entrada

En fase de desarrollo se han utilizado otros modos de ejecución como: *-tokens* devuelve la información de los tokens detectados, *-gui* saca una ventana emergente java con el árbol sintáctico que ha creado.

Además, se han utilizado scripts para la ejecución rápida del programa, de tal manera que no es necesaria la ejecución de los comandos uno a uno.

4. Desarrollo del programa y aclaraciones

El programa está debidamente documentado, pero se procede a explicar algunas aclaraciones que nos parecen importantes.

El programa consta de un único fichero *.g4* en el que se encuentra la gramática, el léxico y las acciones. Se ha implementado toda la funcionalidad introduciendo las acciones en las reglas, teniendo en cuenta el orden en el que se va generando el árbol de análisis.

Por lo tanto, no se ha utilizado el patrón *listener* ni el *visitor*, si no que el programa es un único archivo "*p3b.g4*" que tiene dos partes diferenciadas: una primera parte con la gramática y las acciones intercaladas en las reglas; otra parte al final del fichero con el léxico.

Para la implementación se han utilizado listas para guardar toda la información sobre las funciones y variables; y banderas para indicar en que función se está y el ámbito. Se detallan los algoritmos seguidos:

Algoritmo seguido para las funciones:

- Se recorre el árbol
- Si se encuentra una declaración de función:
    - Cuando se encuentre su nombre:
        - se añade su nombre a la lista de nombres de funciones (*funciones*)
        - se crea una nueva lista donde se guardaran las llamadas a funciones (*llamadas*)
        - se actualiza la bandera de la función en la que nos encontramos (*funcionActual*)
        - se actualiza la bandera del ámbito en el que nos encontramos (*esGlobal*)
    - Cuando se acabe de leer el nombre de los parámetros:
        - se guarda el nombre y tipo del parámetro en la lista de parámetros de las funciones (*param*)
    - Cuando se recorra su bloque de sentencias, si tiene:
        - recorre las sentencias
        - si encuentra una llamada a una función, se añade el nombre de la función invocada a la lista de llamadas a funciones (*llamadas*)
    - Cuando acabe su bloque de sentencias, si le tiene:
        - Se actualiza la bandera del ámbito (*esGlobal*)

Algoritmo seguido para las variables:

- Se recorre el árbol
- Si se encuentra una declaración de variable:
    - Cuando se encuentre su nombre:
        - se añade el nombre de la variable a la lista de variables globales (*globVar*) o a la de variables locales(*variables*) dependiendo de la bandera del ámbito (*esGlobal*)

Se ha utilizado *@header{…}* para importar diferentes bibliotecas utilizadas en el código, y *@parser::members{…}* para introducir todo el código necesario que utilizará el parser.

En ésta última sección se definen todas las variables y listas que se utilizan, y el método "*salidaFichero()*", que imprime en un fichero toda la información recopilada sobre funciones y variables en las listas. La invocación a dicho método se debe realizar al final del análisis, es decir, al acabar de procesar el documento. Para ello utilizamos *@after{…}* en el símbolo inicial.

5. Posibles mejoras

El analizador construido funciona correctamente, pero es simple y pensamos que se podría mejorar para incluirle diversas funcionalidades que consideramos interesantes:

- Mejorar la gramática para que sea capaz de analizar programas complejos en C:
    - perfeccionar el reconocimiento de macros,
    - perfeccionar las sentencias iteradoras y de selección,
    - mejorar la asignación de variables
    - y, manejo de punteros.
- Aportar más información sobre las funciones:
    - argumentos pasados a las funciones en su llamada
    - y, tipo de dato devuelto por la función.
- Aportar información sobre el uso de las variables en las funciones:
    - nombre de la variable y su ámbito
    - Y, si se realizan redefiniciones.

La mayoría de éstas funcionalidades las implementaríamos de la misma forma que las ya realizadas: utilizando listas y banderas, y siguiendo un algoritmos parecidos a los ya comentados anteriormente.

6. Bibliografía

· *Configuración y ejecución: apartado "Getting Started with ANTLR v4" de la documentación oficial antlr4 -* https://github.com/antlr/antlr4/blob/master/doc/getting-started.md

· *Desarrollo del programa: apartados "Grammar Structure", "Parser Rules" y "Actions and Attributes" de la documentación oficial antlr4 -* https://github.com/antlr/antlr4/blob/master/doc/index.md

· *Gramática del programa: gramática completa de C -* https://github.com/antlr/grammars-v4/blob/master/c/C.g4