CASTRO CRUCES JORGE E.	INSTITUTO POLITÉCNICO NACIONAL	ESCUELA SUPERIOR DE CÓMPUTO
TERCER PARCIAL	PRÁCTICA 5: Introducción a BISON	13/06/2021
3CV18	COMPILADORES	Hernández Olvera Luis E

SECCIÓN TEÓRICA O INTRODUCCIÓN

BISON

GNU Bison , comúnmente conocido como Bison, es un generador de analizador que forma parte del Proyecto GNU . Bison lee una especificación de un lenguaje libre de contexto , advierte sobre cualquier ambigüedad de análisis y genera un analizador (ya sea en C , C ++ o Java) que lee secuencias de tokens y decide si la secuencia se ajusta a la sintaxis especificada por la gramática. Los analizadores generados son portátiles: no requieren ningún compilador específico. Bison genera por defecto analizadores LALR (1) pero también puede generar analizadores LR , IELR (1) y GLR canónicos.

En el modo POSIX , Bison es compatible con Yacc , pero también tiene varias extensiones sobre este programa anterior, que incluyen:

- Generación de contraejemplos de conflictos
- Seguimiento de ubicación (p. Ej., Archivo, línea, columna)
- Mensajes de error de sintaxis ricos e internacionalizables en los analizadores generados
- Generación de errores de sintaxis personalizables,
- Analizadores reentrantes
- Analizadores push, con autocompletado
- Soporte para referencias nombradas
- Varios tipos de informes (gráficos, XML) en el analizador generado
- Soporte para varios lenguajes de programación
- Flex , un analizador léxico automático , se usa a menudo con Bison, para tokenizar los datos de entrada y proporcionar tokens a Bison.

Bison fue escrito originalmente por Robert Corbett en 1985. Más tarde, en 1989, Robert Corbett lanzó otro generador de analizador sintáctico llamado Berkeley Yacc . Bison se hizo compatible con Yacc por Richard Stallman .

Bison es software libre y está disponible bajo la Licencia Pública General GNU , con una excepción (discutida a continuación) que permite que su código generado se use sin activar los requisitos de copyleft de la licencia. [1]

Bison es un generador de analizadores sintácticos, convierte una descripción para una gramática independiente del contexto en un programa en C a fin de analizar esa gramática. Es compatible con Yacc, otra herramienta de GNU que genera también analizadores léxicos. Junto a Flex Bison permite construir compiladores de lenguajes. Bison fue escrito originalmente por Robert Corbett; Richard Stallman lo hizo compatible con Yacc. El lenguaje debe estar escrito en una gramática independiente del contexto para poder ser analizada por el generador léxico Bison, lo que significa que se deben definir y detallar grupos sintácticos y proporcionar las reglas para construirlos. El sistema formal más utilizado para presentar las reglas en una gramática independiente del contexto es la Forma de Backus-Naur (BNF, por sus siglas en inglés).[2]

Bison is a general-purpose parser generator that converts an annotated context-free grammar into a deterministic LR or generalized LR (GLR) parser employing LALR(1), IELR(1) or canonical LR(1) parser tables. Once you are proficient with Bison, you can use it to develop a wide range of language parsers, from those used in simple desk calculators to complex programming languages.

Bison is upward compatible with Yacc: all properly-written Yacc grammars ought to work with Bison with no change. Anyone familiar with Yacc should be able to use Bison with little trouble. You need to be fluent in C, C++ or Java programming in order to use Bison or to understand this manual.

We begin with tutorial chapters that explain the basic concepts of using Bison and show three explained examples, each building on the last. If you don’t know Bison or Yacc, start by reading these chapters. Reference chapters follow, which describe specific aspects of Bison in detail.

Bison was written originally by Robert Corbett. Richard Stallman made it Yacc-compatible. Wilfred Hansen of Carnegie Mellon University added multi-character string literals and other features. Since then, Bison has grown more robust and evolved many other new features thanks to the hard work of a long list of volunteers. For details, see the THANKS and ChangeLog files included in the Bison distribution.[3]

PROBLEMAS RELACIONADOS AL TEMA A LOS CUALES TE ENFRENTASTE AL PROGRAMAR LA PRÁCTICA

Para esta práctica, el principal problema al que me enfrenté fue aprender y comprende un nuevo lenguaje; Porque, a pesar de que Bison es bastante conocido en el ámbito computacional, yo no había tenido la oportunidad de aprenderlo ni ocuparlo para ninguna materia.

Otra gran problemática a la que me enfrenté fue: La programación de la práctica en sí, porque no fue nada fácil concebir ni lograr programar un programa que reconozca los lexemas que se pidieron, demás, tuve dificultad para programar el programa que me permitió reconocer las gramáticas que solicitó el profesor.

Para terminar, un gran problema que tuve fue a la hora de restar, ya que el programa arrojaba un error a la hora de hacer la operación, porque no sabía si era un número con signo o era el signo de resta. Por lo tanto, tuve que hacer la operación de resta, primero poniendo el número con signo y después, el número sin signo; Solo así pude resolver el problema.

Dicho lo anterior, es acertado pensar que me costó trabajo llevar a cabo esta práctica y que a pesar de las dificultades, logré sacar avante la tarea y eso me demuestra a mí mismo la calidad y la determinación que tengo como persona y como estudiante.

CAPTURAS DE PANTALLA

Primero compilamos nuestro programa que permite reconocer los lexemas de:

- Números enteros con y sin signo. **Ejemplo (5, 34, -100).**
- Números decimales con y sin signo. **Ejemplo (.05, 0.51, - 13.1, -3.1416).**

Después, compilamos nuestro programa que permite reconocer las gramáticas para las siguientes formas:

- Operaciones matemáticas **(+,-,*,/).**
- Modulo como función **Ejemplo(Mod(5,3) ó mOd(8.5,3)).**

Por último unimos ambos programas para que el primero alimente de tokens al segundo.

```
eduardocc@eduardocc-VirtualBox:~/Escritorio/7moSemestre/Compis/Practica4$ flex Practica5.l
eduardocc@eduardocc-VirtualBox:~/Escritorio/7moSemestre/Compis/Practica4$ bison Practica5.y -d
eduardocc@eduardocc-VirtualBox:~/Escritorio/7moSemestre/Compis/Practica4$ gcc lex.yy.c Practica5.tab.c
Practica5.tab.c: In function ‘yyparse’:
Practica5.tab.c:1560:7: warning: implicit declaration of function ‘yyerror’; did you mean ‘yyerrok’? [-Wimplicit-function-declaration]
 1560 |       yyerror (YY_("syntax error"));
      |       ^~~~~~~
      |       yyerrok
eduardocc@eduardocc-VirtualBox:~/Escritorio/7moSemestre/Compis/Practica4$ ./a.out
```

Pasamos a la etapa de pruebas de escritorio:

- Número entero sin signo
- Número entero con signo
- Número flotante sin signo
- Número flotante con signo

```
5
Numero entero sin signo 5
Salto de linea
        resultado: 5
-5
Numero entero con signo -5
Salto de linea
        resultado: -5
5.5
Numero flotante sin signo 5.5
Salto de linea
        resultado: 5.5
-5.5
Numero flotante con signo -5.5
Salto de linea
        resultado: -5.5
```

- Suma de dos números enteros sin signo
- Multiplicación de dos números enteros sin signo
- División de dos números enteros sin signo
- Resta de dos números enteros sin signo

```
5+5
Numero entero sin signo 5
Signo op
Numero entero sin signo 5
Salto de linea
        resultado: 10
5*5
Numero entero sin signo 5
Signo op
Numero entero sin signo 5
Salto de linea
        resultado: 25
5/5
Numero entero sin signo 5
Signo op
Numero entero sin signo 5
Salto de linea
        resultado: 1
-5+5
Numero entero con signo -5
Signo op
Numero entero sin signo 5
Salto de linea
        resultado: 0
```

- Suma de dos números flotantes sin signo
- Multiplicación de un número entero sin signo por un número flotante sin signo
- Multiplicación de dos números flotantes sin signo
- División de dos números flotantes sin signo
- División de un número flotante con signo entre un número flotante sin signo

```
5.5+5.5
Numero flotante sin signo 5.5
Signo op
Numero flotante sin signo 5.5
Salto de linea
        resultado: 11
5*5.5
Numero entero sin signo 5
Signo op
Numero flotante sin signo 5.5
Salto de linea
        resultado: 27.5
5.5*5.5
Numero flotante sin signo 5.5
Signo op
Numero flotante sin signo 5.5
Salto de linea
        resultado: 30.25
5.5/5.5
Numero flotante sin signo 5.5
Signo op
Numero flotante sin signo 5.5
Salto de linea
        resultado: 1
-5.5/5.5
Numero flotante con signo -5.5
Signo op
Numero flotante sin signo 5.5
Salto de linea
        resultado: -1
```

- Multiplicación de un número flotante con signo por un número flotante sin signo
- Módulo de dos números enteros sin signo
- Módulo de un número entero con signo entre un número entero sin signo

```
-5.5*5.5
Numero flotante con signo -5.5
Signo op
Numero flotante sin signo 5.5
Salto de linea
        resultado: -30.25
mod(5,2)
Modulo
Signo parentesis
Numero entero sin signo 5
Coma
Numero entero sin signo 2
Signo parentesis
Salto de linea
        resultado: 1
mod(-5,2)
Modulo
Signo parentesis
Numero entero con signo -5
Coma
Numero entero sin signo 2
Signo parentesis
Salto de linea
        resultado: -1
```

CONCLUSIÓN

¿Qué aprendí de esta práctica?

Claramente, aprendí muchas cosas, entre ellas está:

- Aprender a programar de forma orientada a objetos.
- Aprender a programar en Bison
- Aprender a programar de forma recursiva.
- Comprender de mejor forma las gramáticas.
- Aprender la diferencia entre un terminal y un no terminal

¿En qué me ayudó esta práctica?

Claramente, me ayudo a comprender de mejor forma la utilización y las ventajas de programar en Bison y el gran poder que posee y nos otorga la programación orientada a objetos, así como, la programación recursiva

También, expandió mi conocimiento sobre la materia de Compiladores y el abanico de posibilidades que esta herramienta nos otorga.

REFERENCIAS

[1] En.wikipedia.org. 2021. *GNU Bison - Wikipedia*. [online] Available at: <*https://en.wikipedia.org/wiki/GNU_Bison*> [Accessed 8 June 2021].

[2] Ecured.cu. 2021. *GNU Bison - EcuRed*. [online] Available at: <*https://www.ecured.cu/GNU_Bison*> [Accessed 8 June 2021].

[3] Gnu.org. 2021. *Introduction (Bison 3.7.6)*. [online] Available at: <*https://www.gnu.org/software/bison/manual/html_node/Introduction.html*> [Accessed 8 June 2021].