Trabajo Práctico Nº 5 – Parser

Usando bison

1. **Objetivos**
   a. Armar un parser para lenguaje mini.
   b. Afianzar el conocimiento de bison y su integración con flex.
2. **Temas**
   a. Parser.
   b. Coordinación entre parser y escáner.
3. **Tareas**
   a. Se pide armar un parser usando las herramientas flex y bison para el lenguaje mini que fue descripto en un TP previo. Como escáner le acoplaremos, con las modificaciones necesarias, el realizado en el TP3 con flex.
   Seguiremos trabajando contra stdin y stdout, y redireccionando el archivo a probar al flujo stdin. Se proveen 2 archivos de entrada y sus correspondientes salidas.
   Comentarios:
      1. El archivo `entradaok.txt` es uno totalmente correcto, en tanto `entradaerr.txt` tiene errores léxicos y sintácticos. El archivo `salidaerrDet.txt` es una variante donde se detectan posibles asignaciones mal formadas (anecdótico, es por si alguno toma ese camino).
      2. La idea es mostrar un mínimo de por donde pasa el parser. Marcaremos el principio, el fin y para cada sentencia de que tipo es, en modo similar al TP4. Esto se puede ver claramente en `salidaok.txt`.
      3. Haremos las siguientes modificaciones.
         i. En sentencia de declaración mostraremos el símbolo que estamos declarando.
         ii. Cada vez que hacemos alguna operación aritmética la informamos. (multiplicación, división, suma, resta, módulo e inversión)
         iii. Informo luego de abrir o cerrar un paréntesis.
      4. Armar la gramática en bison achatada y usando una “tabla de precedencias y asociaciones”.
      5. Utilice directivas para generar los fuentes y encabezados con el nombre que especificamos (`scanner.c` , `scanner.h` y `parser.c` , `parser.h`)
      6. Usar la directiva que permita mensajes de error descriptivos por parte de bison, y usar el no terminal `error` para agregar una opción más en el no terminal `sentencia`, de modo que se recupere de una sentencia errónea sincronizando con el `;` al final de la misma.
   b. Programar usando los siguientes fuentes:
      1. `main.c` llama al parser y hace el informe final de la ejecución.
      2. `scanner.l` con la especificación para que flex arme los fuentes del scanner.
      3. `parser.y` con la especifiación para que bison arme los fuentes del parser.
      4. `makefile` para poder armar todo el proyecto, es decir, correr flex, bison y compilar.
   c. Los errores léxicos a reconocer son los mismos de TP3 y volvemos a tomar en consideración los comentarios.

4. Productos

```
`24-002-xx                      //Repositorio del grupo
    |-- readme.md               // Carátula del grupo, ya hecha en TP1
    `-- TP5                     //Directorio para el TP2
        |-- readme.md           // Carátula del TP
        |-- main.c              // Inicio del programa
        |-- scanner.l           // Definición para flex
        |-- parser.y            // Definición para bison
        |-- makefile            // para armar el proyecto
```

5. Fechas de entrega

a. Última fecha para primera entrega: 12/11/2024

b. Última fecha para segunda entrega: 26/11/2024