



TP N°3


FLEX Y BISON

Sintaxis y Semántica de
los Lenguajes.
UTN FRBA.
Docente: Roxana Leituz.



Grupo N°5

Agustina Guitman Dalla Riva | Legajo: 1772879
Bryan Aldo Battagliese | Legajo: 2034013
Christian Nahuel Jesús Vázquez García | Legajo: 2041467
Macarena Ugolini | Legajo: 2041224
Tiago Pereyra | Legajo: 2139881



• INTRODUCCIÓN:

En este trabajo práctico, se nos pidió realizar un programa utilizando Flex y Bison que realice análisis léxico, sintáctico y semántico del lenguaje micro.

En este documento describimos el proceso de resolución e implementación del mismo, incluyendo el uso de rutinas semánticas y errores personalizados, y el manual de usuario para usar el compilador correctamente.

• RESOLUCIÓN E IMPLEMENTACIÓN:

❖ *Creación del compilador:*

Para empezar, creamos el archivo `tp_flex.l`, encargado de contener las reglas léxicas del lenguaje micro en C. La estructura del mismo es:

- Declaraciones en C: aquí incluimos las bibliotecas necesarias (incluida `y.tab.h`, para lograr la vinculación con bison) para definir las variables y tipos de datos, y dar nombre a ciertas expresiones regulares.
- Reglas: indicamos mediante expresiones regulares los tokens a reconocer y mediante código C las acciones a realizar una vez reconocidos los mismos.
- Código de usuario: en nuestro caso, no aplicamos código adicional para copiar al final del archivo fuente.

Luego, creamos el archivo `tp_bison.y`, encargado de contener las reglas sintácticas del lenguaje micro en C usando los tokens generados por flex para construir el árbol de análisis sintáctico correspondiente. Su estructura es:

- Declaraciones en C: devuelta, aquí incluimos las bibliotecas necesarias y la declaración de los tipos de tokens, además de las funciones de análisis léxico necesarias como `yytext`, `yylen`, etc.
- Declaraciones de bison: usando `%union`, definimos los tipos de datos asociados a los tokens y las reglas; y usando `%token` definimos los tokens.
- Reglas gramaticales: en esta parte escribimos las reglas de la gramática, es decir, la composición de las sentencias y expresiones a reconocer. Además, entran en juego las rutinas semánticas como la evaluación de expresiones o el manejo correcto de longitud de identificadores.

Finalmente, creamos el archivo de funciones auxiliares `funciones.h`, el cual se encuentra incluido en las declaraciones en C de bison, y contiene la gestión de la tabla de símbolos y el procesamiento de identificadores y constantes.

❖ *Vinculación de archivos:*

Al estar listos los diferentes archivos, los vinculamos y creamos nuestro archivo ejecutable abriendo la consola cmd y usando los comandos:

- `bison -yd tp_bison.y`

- flex tp_flex.l
- gcc y.tab.c lex.yy.c -o ejecutable

•MANUAL DE USUARIO:

❖ *Requisitos:*

Sistema operativo Linux/Windows, con MinGW instalado. Herramientas Flex, Bison, GCC. Archivos tp_flex.l , tp_bison.y, funciones.h.

❖ *Proceso de compilación y ejecución:*

Primero, hay que generar los archivos intermedios (y.tab.c y lex.yy.c), esto lo logramos usando en el cmd con los comandos previamente especificados. Luego, abrimos el ejecutable con el comando ejecutable.exe, y ya podemos empezar a utilizar el compilador inicializando con la palabra reservada “inicio”, y al terminar finalizando con la palabra reservada “fin”.

•PANTALLAS DE PRUEBA DEL FUNCIONAMIENTO:

❖ *Compilación:*

```

Microsoft Windows [Versión 10.0.19045.5131]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\macau>cd C:\Users\macau\OneDrive - UTN.BA\UTN\2-anio\sintaxis-semantica\TP-Flex-Bison

C:\Users\macau\OneDrive - UTN.BA\UTN\2-anio\sintaxis-semantica\TP-Flex-Bison>bison -yd tp_bison.y

C:\Users\macau\OneDrive - UTN.BA\UTN\2-anio\sintaxis-semantica\TP-Flex-Bison>flex tp_flex.l

C:\Users\macau\OneDrive - UTN.BA\UTN\2-anio\sintaxis-semantica\TP-Flex-Bison>gcc y.tab.c lex.yy.c -o ejecutable

C:\Users\macau\OneDrive - UTN.BA\UTN\2-anio\sintaxis-semantica\TP-Flex-Bison>ejecutable.exe
  
```

❖ *Ejecución:*

En nuestro caso, le pasamos al ejecutable un archivo de prueba llamado prueba.micro.txt.

```

prueba.micro: Bloc de notas

Archivo Edición Formato Ver Ayuda

inicio
id
20
leer(x,y);
escribir(2+1,4-2);
2+2
5-3
fin
  
```

```

C:\Users\macau\OneDrive - UTN.BA\UTN\2-anio\sintaxis-semantica\TP-Flex-Bison>ejecutable.exe < prueba.micro.txt

Identificador reconocido! La longitud es: 2
Identificador reconocido!

Digito reconocido!
Identificador reconocido: x
Otro identificador reconocido: y
Lista de ID leida!

Expresion reconocida: 1
Otra expresion reconocida: 2
Lista de Expresiones escrita!

Expresion reconocida!
Resultado de la expresion: 4

Expresion reconocida!
Resultado de la expresion: 2
Programa reconocido!
  
```

•CONCLUSIÓN:

Este compilador del lenguaje micro en C, basado en Flex y Bison, nos permitió analizar código fuente, verificar su funcionamiento léxico y sintáctico, y realizar operaciones simples. Además, logramos implementar rutinas semánticas para verificar identificadores y constantes, e imprimir mensajes que le den un uso más didáctico y fácil de comprender a la interfaz