TP N° 3: Flex y Bison

Grupo 4

Comisión: K2055

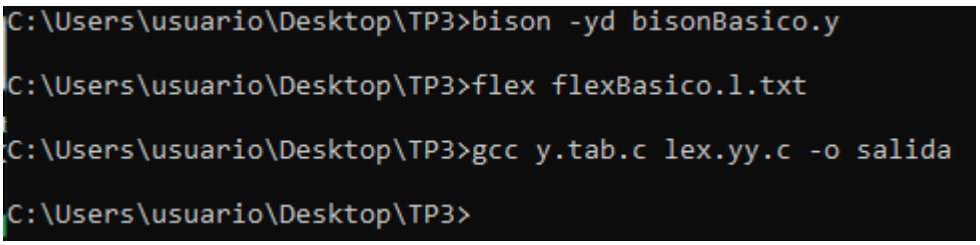
Docente: Roxana Leituz

Integrantes:

- Tomás Iván Barbieri (209.945-7)
- Joaquín López (208.924-5)
- Mercedes Cantarero (203.788-9)

Implementamos el código de Flex y Bison que la docente puso a nuestra disposición, y sobre eso trabajamos en elaborar dos **rutinas semánticas**.

Para generar ejecutable en la consola:



La primera rutina semántica implementada fue la de **variable no inicializada**. Es decir, que en caso de querer usar una variable que no haya sido inicializada previamente, habrá un **error** desplegado en pantalla.

El error de prueba lo tenemos en un archivo llamado “noInicializado.txt” que contiene lo siguiente:

```
b := a + 4;
```

En este caso, el identificador “**a”** no fue declarado ni inicializado previamente.

El ejecutable **salida** recibe el archivo de texto.

```
C:\Users\usuario\Desktop\TP3>salida < noInicializado.txt
  Variable utilizada: a
Mi error personalizado: Error semantico: variable no inicializada.
```

La segunda rutina semántica implementada fue la de **constante ya inicializada**, donde al momento de darle valor a una constante, esta no puede cambiar de valor.

Para lograr esto tuvo que modificarse el archivo **lex**, donde se añadió el token **CONST**

```
"const" {return CONST;}
":=" {return ASIGNACION;}
{constEntera} {yylval.num = atoi(yytext); return CONSTANTE;}
{IDENTIFICADOR} {return ID;}
";" {return PYCOMA;}
"(" {return PARENIZQUIERDO;}
")" {return PARENDERECHO;}
"+" {return SUMA;}
"-" {return RESTA;}
```

El archivo de prueba es “constanteInicializada.txt”, y tiene el siguiente código:

```
const A := 5;
A := 10;
```

```
C:\Users\usuario\Desktop\TP3>salida < constanteInicializada.txt
   Valor constante: 5
Constante declarada: ;

  Valor constante: 10
Asignacion a variable: ;
Mi error personalizado: Error semantico: la constante ya fue inicializada.
```

Funciones o estructuras usadas:

una tabla de símbolos, con el nombre del identificador y campos booleanos para determinar si fue inicializada y si es una constante.

```
// Tabla de símbolos
struct SymbolTable {
    char nombre[32];
    int inicializada;
    int esConstante;
};
```

Función para encontrar la posición de una variable

```
int buscarVariable(char* nombre) {
    for (int i = 0; i < numSimbolos; i++) {
        if (strcmp(tablaSimbolos[i].nombre, nombre) == 0) {
            return i; // Variable encontrada
        }
    }
    return -1; // Variable no encontrada
}
```

Función para agregar variable

```
void agregarVariable(char* nombre, int esConstante) {
    if (numSimbolos < MAX_SYMBOLS) {
        strcpy(tablaSimbolos[numSimbolos].nombre, nombre);
        tablaSimbolos[numSimbolos].inicializada = 0; // Inicialmente no esta asignada
        tablaSimbolos[numSimbolos].esConstante = esConstante; // Marcar como constante si corresponde
        numSimbolos++;
    }
}
```

Función para inicializar variable

```
void inicializarVariable(char* nombre) {
    int index = buscarVariable(nombre);
    if (index != -1) {
        tablaSimbolos[index].inicializada = 1; // Marcamos como inicializada
    }
}
```

Función para determinar si una variable ya esta inicializada

```
int estaInicializada(char* nombre) {
    int index = buscarVariable(nombre);
    if (index != -1) {
        return tablaSimbolos[index].inicializada; // Devuelve si está inicializada
    }
    return 0; // Si no está en la tabla, consideramos que no está inicializada
}
```

Función para determinar si una variable es constante o no

```
int esConstante(char* nombre) {
    int index = buscarVariable(nombre);
    if (index != -1) {
        return tablaSimbolos[index].esConstante;
    }
    return 0; // Si no está en la tabla, consideramos que no es constante
}
```

```
sentencias: sentencias sentencia
| sentencia
;

sentencia: CONST ID ASIGNACION expresion PYCOMA {
    // Declaración de constante
    printf("Constante declarada: %s\n", yytext);
    int idx = buscarVariable(yytext);
    if (idx == -1) {
        agregarVariable(yytext, 1); // Marca como constante
    }
}
| ID ASIGNACION expresion PYCOMA {
    // Asignación de variable
    printf("Asignacion a variable: %s\n", yytext);
    if (esConstante(yytext)) {
        yyerror("Error semantico: la constante ya fue inicializada.");
    }
    int idx = buscarVariable(yytext);
    if (idx == -1) {
        agregarVariable(yytext, 0); // Marca como variable normal
    }
    else {
        inicializarVariable(yytext); // Si la variable existe, inicialízala
    }
}
;
```

```
expresion: primaria
| expresion operadorAditivo primaria
;

primaria: ID {
    printf("Variable utilizada: %s\n", yytext);
    if (!estaInicializada(yytext)) {
        yyerror("Error semantico: variable no inicializada.");
    }
}
| CONSTANTE { printf("Valor constante: %d\n", atoi(yytext)); }
| PARENIZQUIERDO expresion PARENDERECHO
;
```