Lenguajes y Compiladores

Implementación de YACC/BISON

Yet
Another
Compilers
Compiler

YACC

Introducción

Está herramienta está escrita en lenguaje C y genera un compilador para un lenguaje determinado. Utiliza el método de parsing ascendente LALR.

Incluye:

a) Reglas sintácticas que describen la estructura del lenguaje a analizar
b) Código a ser invocado cuando esas reglas son reconocidas
c) Código auxiliar para el soporte del código de entrada

YACC

Introducción

Yacc lleva adelante el proceso de análisis sintáctico a través de la función especial propia:

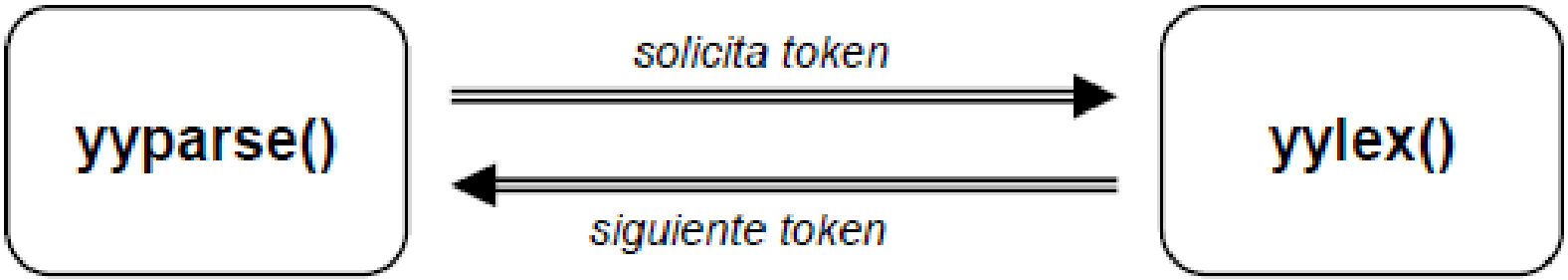
yyparse()

que invocará una función especial para controlar el proceso de entrada:

yylex()

Así deberá llamarse su analizador léxico para poder interactuar con **yyparse()** cuando esta última necesite analizar los tokens del programa fuente original de acuerdo al flujo de entrada.

YACC
Introducción



Cuando una regla se reconoce se dispara una acción, que podrá ser implementada a través de código.

YACC

Estructura–Secciones–Especificaciones Básicas

Un programa **YACC** se organiza en 4 grandes secciones:

1. Declaraciones
2. Tokens, start
3. Definición de reglas
4. Código

YACC

Estructura–Secciones–Especificaciones Básicas

Sección Declaraciones:

En esta sección deberá incluir todas las declaraciones que se utilizarán en el código y deberán enmarcarse por los signos

%{ %}

Las declaraciones pueden ser:

- Includes
- Defines
- Variables Globales

YACC

Estructura–Secciones–Especificaciones Básicas

Sección Declaraciones - Ejemplo:

```
%{
#include <stdio.h>
#define …
int matrizEstados [34] [22] ;
int ( *proceso[34] [22] ) (void) ;
int q;
typedef pila_ass* pilaAss;
…
%}
```

YACC

Estructura–Secciones–Especificaciones Básicas

Sección Tokens:

En esta sección deberá incluir todas las declaraciones de:

- **tokens** (%token)
- **asociatividades** (%left, %right,…)
- **start symbol** (%start)

No es obligatorio declarar el start symbol explícitamente dentro de la sección de declaraciones. Si no se declara se tomará el no terminal que se encuentre en la primera regla escrita en la sección de reglas

YACC

Estructura–Secciones–Especificaciones Básicas

Sección Tokens: Ejemplo

```
%token ID
%token CTE
%left MAS MENOS
%right IGUAL
%token PARENTA, PARENTC, MAS , MENOS
%start programa
```

YACC

Estructura–Secciones–Especificaciones Básicas

Sección Tokens: Ejemplo

```
%token ID
%token CTE
%left MAS MENOS
%right IGUAL
%token PARENTA, PARENTC, MAS , MENOS
%start programa
```

YACC

Estructura–Secciones–Especificaciones Básicas

Sección Definición de Reglas:

Esta sección deberá incluir todas las definiciones de las reglas gramaticales necesarias para llevar adelante el análisis sintáctico.

Deberá enmarcarse por los signos:

%% %%

YACC

Estructura – Secciones – Especificaciones Básicas

Sección Definición de Reglas: (cont.)

Los nombres de los elementos

no terminales pueden:

- tener una longitud arbitraria
- estar conformados por letras, puntos, “underscore” y dígitos (no pueden empezar con éstos).
- Mayúsculas y minúsculas son indistintas.

YACC

Estructura–Secciones–Especificaciones Básicas

Sección Definición de Reglas: (cont.)

Los elementos **terminales** deberán corresponderse con los objetos definidos en la sección declaración de tokens *(Ver Sección Definición Tokens)*

YACC

Estructura–Secciones–Especificaciones Básicas

Sección Definición de Reglas: (cont.)

Reglas

- Si hay varias reglas gramaticales con el mismo no terminal en su lado izquierdo, la barra vertical “**|**” puede ser utilizada para representar varias opciones sobre el mismo no terminal.
- Todas las reglas deben finalizar con un **;**

YACC

Estructura–Secciones–Especificaciones Básicas

Sección Definición de Reglas : Ejemplo

Las siguientes reglas

expresion: expresion MAS termino;
expresion: expresion MENOS termino
expresion: PARENTA expresión PARENTC;

Pueden ser escritas como sigue:

expresion: expresion MAS termino
| expresion MENOS termino
| PARENTA expresión PARENTC;

YACC

Estructura–Secciones–Especificaciones Básicas

Sección Código:

Esta última sección deberá incluir todo el código necesario para la ejecución del programa.

YACC

Estructura–Secciones–Especificaciones Básicas


```
int yylex(void)
{
...
yyval = posicion en tabla de simbolos(token)
return = token
{
int yyerror(char * s) {
fprintf(stderr, "%s\n", s);
}

int main (int argc, char *argv[])
{
 while (feof(archivo)== 0)
{
yyparse();                    }  }
```

YACC
Acciones

Las **acciones** son ejecutadas cada vez que una regla es reconocida con el proceso de entrada.

Esas acciones pueden retornar valores y pueden obtener valores retornados por acciones previas.

Una acción es una sentencia arbitraria escrita en el lenguaje que compila **yacc**. Deben aparecer encerradas entre llaves "**{""}**" y finalizada con un "**;**"

YACC
Acciones

Ejemplo :

expresion: expresion OPMAS termino **{ generarPolaca();}**
expresion OPMENOS termino **{generarPolaca();}|**
termino **{printf("termino \n");}**;

YACC

Acciones

En las acciones es posible **asignar o retornar** un valor.

Existen **variables** propias o **pseudo variables** de yacc a las que se les puede asignar o retornar un valor.

Para obtener los valores retornados por las acciones previas y el analizador léxico, la acción puede usar las **pseudo variables $1, $2**…. que se refieren a los valores retornados por los componentes del lado derecho de una regla, leídos de izquierda a derecha.

La **variable $$** que se refiere al valor retornado por el componente del lado izquierdo.

YACC
Acciones

Ejemplo:

Una acción que no hace nada pero retorna el valor 1 se escribe como:

{ **$$** = 1; }

Sea la siguiente regla:

A : B C D ;

$1 tiene el valor retornado por **B**

$2 tiene el valor retornado por **C**

$3 tiene el valor retornado por **D**

YACC
Acciones

Ejemplo (más concreto) :

expr : PARENTA expr PARENTC;

El valor retornado por esta regla es, de manera usual, el valor de la expr entre paréntesis. Esto se indica como:

expr : PARENTA expr PARENTC **{ $$ = $2; }**

YACC
Acciones

Nota:

Por defecto, el valor de una regla es el valor del primer elemento y no necesita una acción explícita.

Por ejemplo:

A : B ;

Frecuentemente no necesita tener una acción explícita.

YACC

Ambigüedades y conflictos

Un conjunto de reglas gramaticales (sintácticas) son **ambiguas** si hay alguna cadena de entrada que puede ser estructurada de 2 o más maneras distintas.

Por ejemplo en la **gramática**:

expr : expr MENOS expr

Si la **entrada** es la siguiente:

expr - expr - expr

YACC

Ambigüedades y conflictos

La regla permite estructurar la entrada de este modo:

(expr - expr) - expr (**asociación a izquierda**)

o de este otro modo:

expr - (expr - expr) (**asociación a derecha**)

Yacc detecta esta **ambigüedad** cuando intenta efectuar el parsing. Esto es un **problema** con el cual debe enfrentarse al encontrar entradas de este tipo y **deberá ser solucionado**.

YACC
Precedencia

La precedencia y las asociaciones son asignadas a los *tokens* en la sección de declaraciones.

La manera en la que esto se realiza es utilizando las palabras:

- **%left** (asociación a izq.)
- **%right** (asociación a der.)
- **%nonassoc** (operadores que no puedan ser asociativos por si solos)

al comienzo de la línea, seguida por una lista de *tokens*.

Todos los *tokens* en la misma línea tendrán el mismo nivel de precedencia y asociatividad.

YACC

Precedencia

Ejemplo:

```
%right IGUAL
%left MAS MENOS
%left MUL DIVISION
%%
expr : expr IGUAL expr
| expr MAS expr
| expr MENOS expr
| expr MUL expr
| expr DIVISION expr
| NAME;
```

YACC
Precedencia

La regla anterior permite manejar la siguiente sentencia:

a = b = c*d - e - f*g

de la siguiente manera:

a = (b = (((c*d) - e) - (f*g)))

YACC
Menos Unario

¿El "-" es resta o es un número negativo?

- Si es resta (menos binario), la prioridad es inferior al producto o división
- Si es un menos unario, la prioridad es superior.

YACC proporciona un modificador %prec para modificar la prioridad y asociatividad de una regla.

YACC

Menos Unario - Ejemplo

```
%left '+' '-'
%left '*' '/'
%%
e       : e '+' e
        | e '-' e
        | e '*' e
        | e '/' e
        | '-' e %prec '*'
        |'(' e ')'
        | ID
        | NUM
        ;
```

De esta indicamos que la regla de "menos unario" tiene tanta prioridad como la del producto y, en caso de conflicto, es asociativa por izquierda.

YACC
Menos Unario

Se presentan 2 problemas:

1) La prioridad del menos unario no es igual a la del '*', es MAYOR!

2) El "menos unario" no es asociativo por izquierda, es por DERECHA!

YACC

Menos Unario - Solución

Declarar un terminal que represente a un "**operador ficticio**" para la prioridad y asociatividad para ser usado sólo en la cláusula **%prec**.

Este token **NO SERÁ RETORNADO** por el **Analizador Léxico**.

Esto nos permitirá resolver el problema de la prioridad y la asociatividad

YACC

Menos Unario – Solución - Ejemplo


```
%left '+' '-'
%left '*' '/'
%right MENOS_UNARIO
%%

e       : e '+' e
        | e '-' e
        | e '*' e
        | e '/' e
        | '-' e %prec MENOS_UNARIO
        |'(' e ')'
        | ID
        | NUM ;
```

YACC
Compilación

Todas las especificaciones vistas en los puntos anteriores deberán incluirse en un archivo de extensión “.y” , por ejemplo:

miCompilador.y

Este archivo se compilará con el comando:

bison miCompilador.y

YACC
Compilación

Si el **resultado** de la compilación es **satisfactorio** debiera generarse el archivo:

- **y.tab.c** (archivo que contiene el parsing del programa original del miCompilador)

Si la compilación se hiciese con los parámetros –dyv :

bison –dyv miCompilador.y

se generarán los siguientes archivos

- **y.tab.h** que contiene las redefiniciones (#define) de los tokens del compilador
- **y.output** que los estados del parsing LALR y posibles conflictos si los hubiese

YACC
Compilación

Para obtener el compilador final se deberá compilar el resultado del analizador sintáctico con un compilador del lenguaje elegido para desarrollar las acciones semánticas (por ejemplo C) y a la vez integrarlo con el analizador léxico

gcc.exe lex.yy.c y.tab.c -o TPFinal.exe

YACC

- http://www.mingw.org/
- http://www.gnu.org/software/bison/

YACC
ANSI C

¿Preguntas?