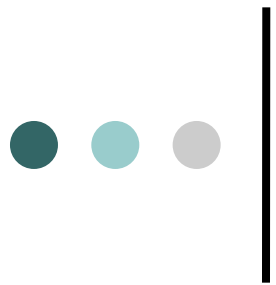




# BISON. Tercera parte

Gestión de acciones  
semánticas



# BISON

- Acciones semánticas
- Distinción de tipos de expresiones
- Errores semánticos



# Acciones semánticas

- Cada producción de una gramática puede llevar asociada una serie de **acciones semánticas**
- Estas acciones son instrucciones de C/C++
- Van construyendo el resultado del análisis sintáctico



# Acciones semánticas

```
linea: expr '\n'          {cout << "El resultado es " << $1 << endl;
                             prompt();}
| IDENTIFICADOR '=' expr '\n' {cout << "A la variable "
                                << $1 << " se le asigna el valor "
                                << $3 << endl; prompt();}
| SALIR '\n'               {return(0); }
| error '\n'               {prompt();}
;
```

# Distinción de tipos de expresiones

- ¿Cómo distinguir el tipo del resultado de la evaluación de una expresión?

```
%type <c_real> expr
```

```
...
```

```
expr: ENTERO                { $$=$1; }  
    | REAL                  { $$=$1; }  
    | expr '+' expr         { $$=$1+$3; }  
    | expr '-' expr         { $$=$1-$3; }  
    | expr '*' expr         { $$=$1*$3; }
```



# Distinción de tipos de expresiones

- ¿Cómo distinguir el tipo del resultado de la evaluación de una expresión?
- Definiendo dos símbolos no terminales distintos: `expr_real`, `expr_entera`
- Usando una bandera
- Utilizando un registro como atributo de la expresión



# Tipos de expresiones: dos símbolos no terminales

```
%type <c_entero> expr_ent
```

```
%type <c_real> expr_real
```

```
. . .
```

```
expr_ent: ENTERO                { $$=$1; }  
        | expr_ent '+' expr_ent { $$=$1+$3; }
```

```
. . .
```

```
expr_real: REAL                { $$=$1; }  
        | expr_real '+' expr_real { $$=$1+$3; }  
        | expr_real '+' expr_ent  { $$=$1+$3; }  
        | expr_ent  '+' expr_real { $$=$1+$3; }
```



# Tipos de expresiones: dos símbolos no terminales

```
linea: expr_ent '\n' {cout << "Resultado: " << $1  
                        << "de tipo entero"  
                        << endl;  
                        prompt();}  
| expr_real '\n'      {cout << "Resultado: " << $1  
                        << " de tipo real"  
                        << endl;  
                        prompt();}
```





# Tipos de expresiones: con bandera

```
//Zona de definiciones
bool esReal = false; // Inicialmente, entera
. . .
%type <c_real> expr
. . .
expr: ENTERO                {$$=$1;}
    | REAL                  {$$=$1; esReal=true;}
    . . .
```



# Tipos de expresiones: con bandera

```
expr: ENTERO                { $$=$1; }  
    . . .  
    | expr '/' expr         { if (esReal)  
                             $$=$1/$3;  
                             else  
                             $$=int($1)/int($3);  
                             }  
    ;
```



# Tipos de expresiones: con bandera

```
linea: expr '\n'    {cout << "Resultado: " << $1;
                      if (esReal)
                        cout << "de tipo real" << endl;
                      else
                        cout << "de tipo entero" << endl;
                      prompt();
                      esReal = false;
                      }
```



# Tipos de expresiones: con bandera

línea:

```
. . .  
|error '\n'      { esReal = false;  
                  yyerrok;  
                  prompt();  
                  }
```

Ejemplo:

```
> 3 * 2.5 + 8 * aaa + 10
```

Error sintáctico

```
> 2 + 1 / 2
```



# Tipos de expresiones: con bandera

¿Cuál es el resultado de estas dos expresiones?

> 1 / 2 + 2.5

> 2.5 + 1 / 2



# Tipos de expresiones: con bandera

¿Cuál es el resultado de estas dos expresiones?

```
> 1 / 2 + 2.5
```

```
Resultado: 2.5 de tipo real
```

```
> 2.5 + 1 / 2
```

```
Resultado: 3 de tipo real
```

¿Qué pasa?



# Tipos de expresiones: atributo compuesto

- El tipo de una expresión solo depende del tipo de los operandos y del operador.
- No depende de operaciones anteriores o posteriores.
- Además del valor de una expresión, necesitamos almacenar también su tipo



# Tipos de expresiones: atributo compuesto

```
%union{  
    int c_entero;  
    float c_real;  
    char c_cadena[20];  
    struct {  
        float valor;  
        bool esReal;  
    } c_expresion;  
}
```





# Tipos de expresiones: atributo compuesto

```
%type <c_expresion> expr
```

```
. . .
```

```
expr: ENTERO      { $$ . esReal = false;  
                   $$ . valor=$1; }  
    | REAL        { $$ . esReal = true;  
                   $$ . valor=$1; }
```



# Tipos de expresiones: atributo compuesto

expr:

. . .

```
| expr '/' expr { $$ . esReal = $1 . esReal ||  
                                                         $3 . esReal;  
                if ( $$ . esReal )  
                    $$ . valor = $1 . valor / $3 . valor;  
                else  
                    $$ . valor = int ( $1 . valor ) / int ( $3 . valor );  
            }
```



# Tipos de expresiones: atributo compuesto

```
linea: expr '\n' {cout << "El resultado es "  
                  << $1.valor;  
                  if ($1.esReal)  
                      cout<<" de tipo real"<<endl;  
                  else  
                      cout<<" de tipo entero"<<endl;  
                  prompt( ); }
```



# Distinción de tipos de expresiones

- ¿Cómo distinguir el tipo del resultado de la evaluación de una expresión?
- Definiendo dos símbolos no terminales distintos: `expr_real`, `expr_entera`
- Usando una bandera
- Utilizando un registro como atributo de la expresión



# Errores semánticos

- BISON detecta los **errores sintácticos** (secuencias de *tokens* que no pertenecen a la gramática)
- ¿Cómo detectamos los **errores semánticos** (secuencias de *tokens* que pertenecen a la gramática pero cuyo significado no es correcto)?



# Errores semánticos

`expr :`

`. . .`

`| expr ' / ' expr { $$=$1/$3; }`

`;`

Ejemplo:

`26 / 0`

Esta expresión es sintácticamente correcta pero no podemos calcular su resultado



# Errores semánticos

```
//Zona de definiciones
```

```
bool errorSemantico = false;
```

```
expr:
```

```
| expr '/' expr
```

```
    {if ($3==0){
```

```
        errorSemantico = true;
```

```
        cout << "División por 0";
```

```
    } else $$=$1/$3
```

```
    }
```

```
Laboratorio TL. Sesión 07. Bison
```

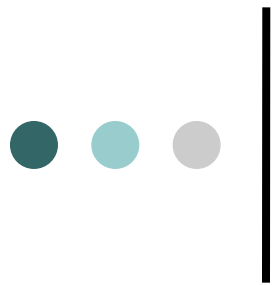
```
;
```



# Errores semánticos

```
linea: expr '\n' {  
    if (errorSemantico) {  
        cout << "Error semántico"<<endl;  
    } else {cout << "Resultado:" << $1;  
        . . .  
    }  
    errorSemantico = false;  
}
```





# BISON

- Acciones semánticas
- Distinción de tipos de expresiones
- Errores semánticos