ANALYSE SÉMANTIQUE

- **char yytext[]:** tableau de caractères qui contient la chaîne d'entrée en cours d'analyse.

```
IDF [a-zA-Z]([a-zA-Z]|[0-9])*
%%
{IDF}  {printf ("idf :  "); printf (" %s ",yytext) ;}
```

```
var
x
```

PgmSource

```
Idf : var
Idf : x
```

PgmCompilé

- **int yyleng**: retourne la longueur de la chaîne d'entrée en cours d'analyse.

```
IDF [a-zA-Z]([a-zA-Z]|[0-9])*
%%
{IDF}  {if (yyleng <=7) printf (" idf ") ;
     else printf ("erreur lexicale : idf trop long") ;
       }
```

```
var
vartoplogue
```

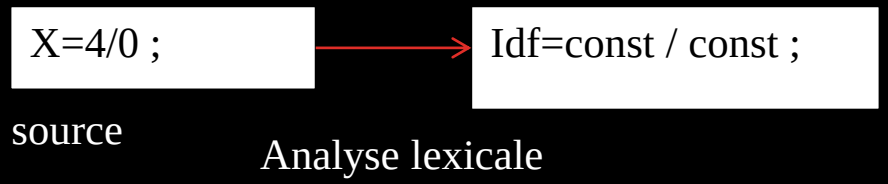
PgmSource

```
Idf
Erreur lexicale :idf trop long
```

PgmCompilé

PARTAGE DE VARIABLE ENTRE FLEX ET BISON

Comment Bison peut traiter les valeurs des entités lexicales, sachant que l'analyseur lexical à coder tout les entités du programme source?



- **Solution:** Flex peut envoyer la valeur d'une entité avant de la coder en utilisant la variable prédéfinie **yylval**

UTILISATION DE YYLVAL

Déclaration de **yylval** dans **FLEX**

```
%{
extern YYSTYPE yylval;
%}
```

Déclaration de type **YYSTYPE** dans **BISON** Comme type composé

```
%union {
int     entier;
char*   str;
}
```

Envoie des valeurs par FLEX

```
{cst} {
yylval.entier=atoi(yytext);
        return cst;
              }
{idf} {yylval.str=strdup(yytext);
          return idf;
        }
```

Traitement des valeurs par BISON

```
%token <str>idf    < entier>cst
```

```
%%
s : idf= idf '/' cst { if ($5==0) printf(" erreur : division par 0")
else  printf("la Division est %s= %s/%d", $1,$3,$5);    YYACCEPT;}
```

-Le symbole $$ référence la valeur associée au non terminal de la partie gauche d'une règle de grammaire.
-Le symbole $i référence la valeur associée au i ème symbole terminal ou non terminal de la partie droite d'une règle de grammaire.

ROUTINE SÉMANTIQUE DE DOUBLE DÉCLARATION D'UNE VARIABLE

Comment savoir si une variable est double déclarée ou non ?

sauvType: variable à déclarer dans la partie déclaration du fichier bison qui servira à màj l'attribut type d'une entité lexicale

Vérifier le champs Type de cette variable dans la table de symbole:

- **Si il est vide alors la variable n'est pas encore déclarée : Insérer le type.**

- Sinon variable est déclarée déjà, signaler double déclaration.

Exemple.y

```
declaration: type listeparam ';' declaration | type listeparam ';' ;
listeparam: listeparam ',' idf { /*vérification de double déclaration*/}
                | idf                { /*vérification de double déclaration*/}
;
type: mc_integer {strcpy(sauvType,$1); }
     | mc_real      {strcpy(sauvType,$1);}
;
```

ROUTINE SÉMANTIQUE DE VARIABLE NON DÉCLARÉE

Comment savoir si une variable est déclarée ou non ?

Vérifier le champs Type de cette variable
- **Si il est vide alors la variable n'est pas encore déclarée :**
  Signaler erreur de non déclaration.
- **Sinon variable est déclarée : utiliser la variable selon sa position**

Exemple.y

AFF: idf '=' idf ';'{ /* vérification de la déclaration des variables, de l'incompatibilité de type et modification d'une constante*/
}

EXEMPLE

Double déclaration:

Fichier.y:

```
%{    int nb_ligne=1, Col=1;  char sauvType[20]; %}
%union { int     entier;   char*   str;   float reel; }
%token <str>idf <str>mc_entier <str>mc_reel sep
%%
DEC: TYPE LISTE_IDF;
LISTE_IDF: idf sep LISTE_IDF { if(doubleDeclaration($1)==0) insererTYPE($1,sauvType);
                                    else
                                    printf("err semantique: double declaration  de %s a la ligne %d\n",$1,nb_ligne);}
          | idf { if(doubleDeclaration($1)==0) insererTYPE($1,sauvType);
                                    else
                                    printf("err semantique: double declaration  de %s a la ligne %d\n",$1,nb_ligne);}
;
TYPE: mc_integer {strcpy(sauvType,$1); }
    | mc_real {strcpy(sauvType,$1);}
;
```

EXEMPLE

Fichier.h:

```
typedef struct
{char NomEntite[20]; char CodeEntite[20];
char TypeEntite[20]; } TypeTS;
TypeTS ts[100];
int CpTabSym=0; //compteur
int recherche(char entite[]){
int i=0;
while(i<CpTabSym)
{ if (strcmp(entite,ts[i].NomEntite)==0) return i;
i++;}
return -1;
}
void insererTYPE(char entite[], char type[])
int pos;
pos=recherche(entite);
if(pos!=-1) strcpy(ts[pos].TypeEntite,type);
}
int doubleDeclaration(char entite[])
{
int pos;
pos=recherche(entite);
if(strcmp(ts[pos].TypeEntite,"")==0) return 0;
   else return -1
}
```