

Université des Sciences et de la Technologie Houari Boumediene
Faculté d'Electronique et d'Informatique
Département Informatique



TP Compilation

07/11/2022

Table des symboles

07/11/2022

Table des symboles

- La table de symboles doit être programmée manuellement dans flex et bison.
- La table de symboles doit contenir des informations sur les entités

```
PROGRAM TpCom  
integer x;  
Const reel y=5;  
BEGIN  
  x=x+1;  
END ;
```

state	name	code/nature	Type	val	Taille
1	TpComp	IDF			
1	X	IDF			
1	Y	IDF			
1	5	const	int	5	
1	y	Const	reel	1	

Table des symboles

- Chaque entité reconnue par le langage doit être insérée dans la table des symboles.

```
typedef struct
{
    int state;
    char name[20];
    char code [20];
    char type[20];
    float val;

} element;
```

```
element tab[1000];
```

**La table des symboles des
constantes, variables**

```
typedef struct
{
    int state;
    char name[20];
    char type[20];
} elt;
```

- **Name** : Nom de l'entité
- **Code** : variable simple, tableau, constante ...etc
- **Type** : type de l'entité (INT, Float ...etc)

```
elt tabs[40], tabm[40];
```

**La table des symboles des
séparateurs et des mots clés**

Fonction d'initialisation de la TS

```
void initialisation()
{
    int i;
    for (i=0;i<1000; i++)    Tab[i]. state=0;
    for (i=0;i<50;i++)
    {
        TabS[i].state=0;
        TabM[i].state=0;
    }
}
```

La fonction de recherche dans la TS

- Permet de vérifier si l'entité existe déjà dans la TS

```
void rechercher (char entite[ ], char type [ ], float val, int y)
```

```
{  
    int j,i;  
    Switch (y)  
    {  
        Case 0: /*verifier si la case dans la tables des IDF et CONST est libre*/  
        FOR (i=0;(   
                ( i<1000)      &&  
                (Tab[i].state==1)) &&  
                (strcmp(entite, Tab[i].name)!=0)  
            ; i++);  
        IF (i<1000)  
            Inserer ( entite, type, val, i, 0);  
        ELSE  
            printf("entité existe déjà\n");  
            break;
```

Position d'insertion

- 0 Recherche dans TS des idf et des constantes
- 1 dans la TS des séparateurs
- 2 dans la TS des mots cles

La fonction de recherche dans la TS

Case 1: /*verifier si la case dans la tables des mots clés est libre*/

```
FOR (i=0;((i<40)&&(TabM[i].state==1))&&(strcmp(entite,Tab[i].name)!=0);i++);
```

```
IF (i<40)
```

```
    inserer(entite,type,val,i,1);
```

```
ELSE
```

```
    Printf ("entité existe déjà\n");
```

```
    break;
```

Case 2: /*verifier si la case dans la tables des séparateurs est libre*/

```
FOR (i=0;((i<40)&&(TabS[i].state==1))&&(strcmp(entite,TabS[i].name)!=0);i++);
```

```
IF(i<40)
```

```
    inserer(entite,type,val,i,2);
```

```
ELSE
```

```
    printf("entité existe déjà\n");
```

```
    break;
```

```
} /* fin switch
```

```
} /* fin recherche
```

La fonction d'insertion dans la TS

```
void inserer (char entite[], char type[], float val, int i, int y)
```

```
{  
    switch (y)  
    {  
        Case 0: /*insertion dans la table des IDF et CONST*/  
            Tab[i].state=1;  
            Strcpy (Tab[i].name, entite);  
            Strcpy (Tab[i].type, type);  
            tab[i].val=val;  
            break;  
  
        Case 1: /*insertion dans la table des mots clés*/  
            TabM[i].state=1;  
            Strcpy (TabM[i].name,entite);  
            Strcpy (TabM[i].type,type);  
            break;  
  
        Case 2: /*insertion dans la table des séparateurs*/  
            TabS[i].state=1;  
            Strcpy (TabS[i].name,entite);  
            Strcpy (TabS[i].type,type);  
            break;                } }  
}
```


TS.h

```
#include <stdio.h>
#include <stdlib.h>
```

```
typedef struct
{
    int state;
    char name[20];
    char type[20];
    float val;
    int sub;
} element;
```

```
typedef struct
{
    int state;
    char name[20];
    char type[20];
} elt;
```

```
element tab[1000];
elt tabs[40],tabm[40];
```

```
void initialisation();
{
```

```
}
```

```
void inserer (char entite[], char type[],float val,int i,int y)
{
}
```

```
void recherche (char entite[], char type[],float val,int y)
{
}
```

```
void afficher()
{
}
```

07/11/2022

Exemple

```
%{  
#include<stdio.h>  
#include<stdlib.h>  
#include<string.h>  
#include "TS.h"  
int yylineo=1;  
int Col=1;  
%}  
lettre [a-zA-Z]  
chiffre [0-9]  
idf {lettre}({lettre}|{chiffre})*  
entier {chiffre}+  
real {chiffre}+"."{chiffre}+  
blanc [ \t]  
SI [ \n]  
%%
```

```
"/" {  
    rechercher (yytext,"sep",0,2);  
    Col = Col + strlen(yytext);  
}  
  
"=" {  
    rechercher (yytext,"sep",0,2);  
    Col = Col + strlen(yytext);  
}  
  
{idf} {  
    Rechercher (yytext," ",0,0);  
    Col = Col + strlen(yytext);  
}  
  
{real} {  
    rechercher (yytext,"real",atof(yytext),0);  
    Col = Col + strlen(yytext);  
}  
  
{entier} {  
    rechercher (yytext,"entier",atoi(yytext),0);  
    Col = Col + strlen(yytext);  
}  
  
{blanc} {Col = Col + strlen(yytext);}  
  
{SI} {yylineo++;Col=1;}
```

Exemple (suite)

```
. { printf ("\n Erreur lexical: Ligne: %d; Collone: %d; Entité << %s >> non reconnu par le langage \n",yylineo, Col, yytext);  
  Col = Col + strlen(yytext);  
}
```

%%

int main()

```
{  
  initialisation();  
  yyin = fopen( "input.txt", "r" );  
  if (yyin==NULL) printf("ERROR \n");  
  else yylex();  
  afficher();  
  return 0;  
}
```