TP COMPILATION

LE LANGAGE FLEX

EMSI - 4ÈME IIR

2017/2018

Prof. M. D. RAHMANI

Expressions régulières du langage FLEX

Expression	Signification	Exemple
c	tout caractère 'c' qui n'est pas un opérateur	a
\c	caractère littéral 'c'	*
"s"	chaîne littérale s	"**"
.	tout caractère sauf fin de ligne	a.b
^r	r en début de ligne	^abc
r$	r en fin de ligne	abc$
[s]	tout caractère appartenant à s	[abc]
[^s]	tout caractère n'appartenant pas à s	[^abc]
[a-z]	tout caractère (lettre minuscule) entre 'a' et 'z'	d
[^a-z]	tout caractère qui n'est pas une lettre minuscule	5
r*	zéro ou plusieurs r	a*
r+	un ou plusieurs r	a+
r?	zéro ou un r	a?
r{m,n}	entre m et n occurrences de r	a{1,5}
r{3,}	trois r ou plus	b{3,}
r{2}	exactement deux r	c{2}
rs	r puis s	ab
r|s	r ou s	a|b
(r)	r	(a|b)
r/s	r quand suivi de s	abc / 123

Fonctions prédéfinies en FLEX:

- **char yytext[]**: tableau de caractères qui contient la chaîne reconnue.
- **int yyleng**: longueur de la chaîne reconnue.
- **int yylex()**: fonction qui lance l'analyseur (et appelle yywrap()).
- **int yywrap()**: fonction toujours appelée en fin du flot d'entrée. Elle ne fait rien par défaut, mais l'utilisateur peut la redéfinir dans la section des fonctions supplémentaires. yywrap() retourne 0 si l'analyse doit se poursuivre (sur un autre fichier d'entrée) et 1 sinon.
- **int yylineno**: numéro de la ligne courante.
- **int main()**: la fonction principale du langage C, elle doit appeler la fonction yylex().

Exercice 1

Ecrire un programme en langage ***FLEX*** qui vérifie si une expression est correctement parenthésée.

Exercice 2

Ecrire un programme en langage ***FLEX*** qui traduit les abréviations contenues dans un texte donnée.

On considérera les abréviations suivantes :

- cad : abréviation de c'est à dire,
- ssi : abréviation de si et seulement si,
- afd : automate à états finis déterministe.

Exercice 3

1- Ecrire un programme en langage ***FLEX*** qui compte le nombre de mots d'un texte saisi au clavier.

2- Même exercice en analysant un fichier texte.

Exercice 4

1/ Ecrire un programme en langage ***FLEX*** qui compte le nombre de caractères et de lignes d'un texte source.

2/ Ecrire un programme en langage ***FLEX*** qui compte le nombre de caractères, le nombre de mots et de lignes d'un texte source.

Exercice 5

Ecrire un programme en langage ***FLEX*** qui remplace toute suite de blancs ou de tabulations par un seul blanc, supprime les espaces de fin de ligne, ainsi que les lignes vides.

Exercice 6

Ecrire un programme en langage ***FLEX*** qui accepte les commentaires à la C++.

Exercice 7

Ecrire un programme en langage ***FLEX*** qui accepte les commentaires à la C.

Exercice 8

Ecrire un programme en langage ***FLEX*** qui les mots clés, les identificateurs, les réels, les opérateurs de relation, les parenthèses et insère les lexèmes et les unités lexicales correspondantes dans une table de symboles qu'il affiche à la fin de l'analyse.

Rappel : La table des symboles est une structure de données constituée des champs suivants:

- un pointeur (`ptrlex`) pointant sur l'adresse du lexème figurant dans le tampon (`lexemes`);
- une chaîne de caractères (`unilex`) qui contient l'unité lexicale du lexème détecté;
- un attribut qui est un indice de la position du lexème dans la table des symboles.

La table de symboles (1)

La table des symboles est une structure de données constituée des champs suivants:

- un pointeur (`ptrlex`) pointant sur l'adresse du lexème figurant dans le tampon (`lexèmes`);

- une chaîne de caractères (`unilex`) qui contient l'unité lexicale du lexème détecté;

- un attribut qui est un indice de la position du lexème dans la table des symboles.

La table de symboles (2)

Supposons que le texte d'entrée est: **`"si gamma=10 alors aire >= 78 sinon g>1.3"`**

La table des symboles aura la forme suivante:

ptrlex	unilex	indice
0	si	1
3	id	2
9	operel	3
11	nb	4
14	alors	5
20	id	6
25	operel	7
28	nb	8
31	sinon	9
37	id	10
39	operel	11
41	nb	12

La table de symboles (3)

La chaîne engendrée est:

```
"si$gamma$=$10$alors$aire$>=78$sinon$g$>$1.3"
```

Pour implanter la table des symboles, nous avons besoin des 2 fonctions suivantes:

- une fonction d'insertion;

- une fonction de recherche.

La table de symboles (4)

L'algorithme de la fonction d'insertion:

```
Fonction inserer
début
   indice        ←——     indice+1;
   TS[indice].ptrlex ←——
                  tampon lexemes
   TS[indice].unilex ←——
   retourner(indice)
fin inserer
```

création d'une nouvelle entrée de la TS

l'adresse du début du lexème dans le

l'unité lexicale associée

La table de symboles (5)

L'algorithme de la fonction de recherche:

```
Fonction chercher
début
   j←————————0;  pour parcourir la TS
   trouve=faux;  boolean pour arrêter la recherche
   tant que (j<taille de la TS et non trouvé) faire
      si TS[j].unilex = l'UL du lexème à chercher alors
            si TS[j].ptrlex pointe sur le lexème
            alors
                  trouve    ←————    vrai;
                  retourne(j);
            sinon j     ←————    j+1;
      sinon trouve retourne(-1)
Fin Fonction chercher
```