Objectif :

- Prendre en main de l'outil LEX sous l'outil PARSER GENERATEUR
- Ecrire et générer un analyseur en LEX pour le C et pour le JAVA
- Exécuter un analyseur généré par LEX sous MS VS et sous un IDE Java (Eclipse)

1. Activités 1 : Prise en main de LEX à l'aide de l'outil PARSER GENERATEUR

1. Créer un projet **ATLEX** sous l'outil PARSER GENENRATEUR dans un emplacement de votre choix
2. Ecrire le programme LEX suivant et générer l'analyseur correspondant pour C et MS VCPP
3. Expliquer ce que fait cet analyseur

```
%{
      #include <stdlib.h>
      int k;
%}
%%
\+?[0-9]+  {
                k=atoi (yytext) ;
                printf ("%d", k%2 == 0 ? k+1 : k);
              }
-?[0-9]+       |
[A-Za-z][A-Za-z0-9]+     ECHO;
.            ;
%%
```

2. Activité 2 : Ecrire le même analyseur dans le langage JAVA pour un ide Eclipse

- Créer un projet sous PARSER GENERATOR
- Ecrie le programme LEX en JAVA
- Compiler et exécuter le fichier JAR

I.Quelques fonctions et variables de LEX

1. L'analyse lexicale est effectuée par une fonction **yylex()**
2. La fonction **yylex()** qui doit être appelée pour utiliser l 'analyseur lexical :
   - Analyse séquentiellement un fichier d 'entrée
   - Retourne 0 lorsqu'elle rencontre une fin de fichier
   - Effectue des opérations spécifiées par le programme Lex, lorsqu'une unité lexicale est reconnue
3. LEX fournit un ensemble de variables et de routines :
   - **yytext** = variable contenant la chaîne de caractères courante qui a été reconnue
   - **yyleng** = longueur de la chaîne yytext
   - **yyless(k**) = fonction admettant un entier comme argument
     - Supprime les (yyleng-k) derniers caractères de yytext, dont la longueur devient alors k
     - Recule le pointeur de lecture sur le fichier d 'entrée de yyleng-k positions, les caractères supprimés de yytext seront donc considérés pour la reconnaissance des prochaines unités lexicales
   - **yymore()** = fonction qui concatène la chaîne actuelle yytext avec celle qui a été reconnue avant
   - **yywrap()** = fonction appelée lorsque yylex() rencontre une fin de fichier
     - si yywrap() retourne `true`, alors yylex() retourne 0 pour indiquer une fin de fichier
     - si yywrap() retourne `false`, alors yylex() ignore la fin de fichier et continue son analyse
     - Par défaut, yywrap() retourne 1, mais on peut la redéfinir.
4. L'entrée est lue via **yyin** et la sortie est écrite dans **yyout**

II. Rappel des expressions régulière sous LEX

Expression	Désigne	Exemple
a	tout caractère **a** qui n'est pas un opérateur	a
\c	caractère littéral c	*
"s"	chaîne littérale s	"**"
.	Tout caractère sauf fin de ligne	a.*b
^	début de ligne	^abc
$	fin de ligne	abc$
[s]	tout caractère appartenant à s	[abc]
[^s]	tout caractère n'appartenant pas à s	[^abc]
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_1r_2	r_1 puis r_2	ab
$r_1\|r_2$	r_1 ou r_2	a|b
(r)	r	(a|b)
r_1/r_2	r_1 quand suivi de r_2	abc/123

III. Etapes de configuration de MS Visual Studio pour compiler un analyseur généré par LEX pour le C

A. Configuration des chemins d'accès aux bibliothèque LEX de l'outil PARSER GENERATOR en ajoutant le chemin vers les fichiers entête (pour l'inclusion des .h) et des bibliothèques (.lib) :

- Ouvrir le menu des « ***propriétés de configuration »***
- Sélectionner ***Répertoires VC++*** et ajouter les répertoires suivants (en supposant que Parser Generator a été installé dans C:\Program Files\Parser Generator 2) :
  - Dans ***répertoires Include***, ajouter le répertoire ***`C:\Program Files\Parser Generator 2\Cpp\include`***
  - Dans ***répertoires de bibliothèques***, ajouter le répertoire ***`C:\Program Files\Parser Generator 2\Cpp\Lib\msvc32`***

1. Configurer le projet :

- Créer un nouveau projet Menu ***Fichier →Nouveau Projet***
- Choisir ***C++ Win32***, ***Projet Win32*** et nommer votre projet
- Choisir le type du projet ***Application console*** et dans ***options supplémentaires*** choisir ***Projet vide*** (car nous avons déjà du code source)
- Ajouter au projet les fichiers sources générés (***Fichiers d'en-tête (.c)*** et ***Fichiers source (.h)***) via l'action du Menu ***Projet → Ajouter un élément existant***.
- Afficher le Gestionnaire de propriétés du projet ***Affichage → Gestionnaire de propriétés*** du projet :
  - Dans ***propriétés de configuration***, sélectionner ***C/C++ → Génération de code***. Dans la rubrique ***Bibliothèque Runtime*** choisir ***Débogage multithread (/MTd).***
  - Dans ***propriétés de configuration***, sélectionner ***Editeur de liens → Entrée***. Dans la rubrique ***Bibliothèques par défaut spécifiques ignorées*** mettre ***libcmt.lib*** et ***msvcrtd.lib***
  - Dans ***propriétés de configuration***, sélectionner ***Editeur de liens → Ligne de commande***. Dans la rubrique ***Options supplémentaires*** mettre ***ylmt.lib*** et ***ylmtd.lib***

IV. Etape de configuration d'un environnement de compilation des analyseurs générés en JAVA

1. Créer un projet JAVA
2. Ajouter le fichier source JAVA
3. Ajouter le JAR externe **« yl.jar »** fourni par PARSER GENERATOR dans les propriétés de Build de l'IDE utilisé
4. Compiler et exécuter l'analyseur et généré éventuellement le fichier JAR exécutable