Adaptation de l’analyseur lexical

Tout d’abord, j’ai récupéré le fichier **lexer** réalisé par *Melliti*.
À partir de ce fichier, j’ai créé un nouveau fichier nommé lexer_bison.l.

Dans ce nouveau lexer, j’ai :

- supprimé tout ce qui n’était pas nécessaire pour l’analyse syntaxique,
- conservé uniquement :
    - la déclaration des **tokens**,
    - les **expressions régulières**,
    - la reconnaissance des **identificateurs**, **mots-clés**, **littéraux**, etc.

L’objectif de cette étape était d’obtenir un **analyseur lexical compatible avec Bison**, qui fournit uniquement les tokens nécessaires à la grammaire.

Définition de la grammaire avec Bison

Ensuite, en me basant sur **la grammaire définie dans le rapport de Chaima**, j’ai implémenté cette grammaire en **Bison** dans le fichier parser.y.

Dans ce fichier, j’ai :

- défini les **symboles terminaux** (tokens venant du lexer),
- défini les **symboles non-terminaux**,
- écrit les règles de production de la grammaire,
- associé certaines règles à des actions sémantiques.

Cela permet à Bison de reconnaître la structure syntaxique correcte des programmes écrits dans le langage.

Construction de l’AST (Arbre Syntaxique Abstrait)

Après la définition de la grammaire, j’ai implémenté la construction de **l’arbre syntaxique abstrait (AST)**.

Pour cela, j’ai créé :

- un fichier ast.h pour déclarer :
    - les structures de l’AST,
    - les fonctions de création des nœuds,

- un fichier ast.c pour implémenter :
    - les fonctions de création des nœuds binaires,
    - les nœuds d’instructions (print, affectation, etc.),
    - les fonctions permettant de relier les sous-arbres.

Ainsi, lors de l’analyse syntaxique, chaque règle de la grammaire construit automatiquement une partie de l’AST, ce qui permet d’obtenir une représentation arborescente du programme.

Programme principal

Ensuite, j’ai écrit le fichier main.c, qui :

- lance l’analyse syntaxique,
- appelle le parseur généré par Bison,
- affiche ou traite l’arbre syntaxique obtenu.

À la fin de l’analyse, un exécutable parser est généré.

Compilation

La compilation du projet se fait à l’aide des commandes suivantes :

```
flex lexer_bison.l
bison -d -v parser.y
gcc lex.yy.c parser.tab.c main.c ast.c -o parser
```

Ces commandes permettent de :

- générer l’analyseur lexical (lex.yy.c),
- générer l’analyseur syntaxique (parser.tab.c et parser.tab.h),
- compiler l’ensemble du projet.

Tests

Enfin, j’ai créé deux fichiers de test :

- test_correct.ql : contenant un programme syntaxiquement correct,
- test_error.ql : contenant des erreurs syntaxiques.

Les tests sont effectués à l’aide des commandes suivantes :

```
./parser test_correct.ql
./parser test_error.ql
```

Ces tests permettent de vérifier :

- la validité de la grammaire,
- la détection correcte des erreurs syntaxiques,
- la construction correcte de l’arbre syntaxique abstrait.