FrontC en OCaml

Kelun Chai, Djaber Solimani

Installation

make

Test

`./frontc [testfile]`

Fonctions implémentées

- Analyse lexicale (lexer.mll) √
- Analyse syntaxique (parser.mly) √
- Localisation des erreurs (main.ml) √
- Typage(ast.ml) ×

Fonction	Commentaire
Lexer	Faire ensemble
Parser	Faire ensemble (Résoudre les conflits `%nonassoc` et `%prec`)
Localisation des erreurs	Kelun implémente la fonction de localisation des erreurs
Arbre de typage	Kelun a essayé mais échoué

Résultat

Test_bons	Résultat
testfile-binop14	√
testfile-binop17	√
testfile-binop18	√
testfile-commentaire1	√
testfile-constantes2	√
testfile-constantes6	√
testfile-constantes11	√
testfile-identificateurs3	√
testfile-identificateurs8	√
`int main() { return 0; }` (test.c)	√

Test_mauvais	Résultat	Message
testfile-char7	√	`File "tests-lex-yacc/mauvais/testfile-char7.c", line 2, characters 0-1: lexical error: Caractère illégal:[']` `'` '(0-1)
testfile-char8	√	`File "tests-lex-yacc/mauvais/testfile-char8.c", line 2, characters 0-1: lexical error: Caractère illégal:["]` `"` "(0-1)
testfile-commentaires3	√	`File "tests-lex-yacc/mauvais/testfile-commentaires3.c", line 4, characters 0-1: syntax error` `// /*` `toto;` `*` /(0-1)
testfile-commentaires4	√	`File "tests-lex-yacc/mauvais/testfile-commentaires4.c", line 3, characters 17-18: lexical error: Commentaire non terminé.` /* non terminé (17-18)
testfile-decl_vars5	√	`File "tests-lex-yacc/mauvais/testfile-decl_vars5.c", line 2, characters 4-5: syntax error` int ; (4-5)
testfile-decl_vars6	√	`File "tests-lex-yacc/mauvais/testfile-decl_vars6.c", line 2, characters 8-9: syntax error` int a ; `b` ; (8-9)
testfile-decl_vars8	√	`File "tests-lex-yacc/mauvais/testfile-decl_vars8.c", line 2, characters 12-13: syntax error` int a = 1 , `1` ; (12-13)
testfile-decl_vars9	√	`File "tests-lex-yacc/mauvais/testfile-decl_vars9.c", line 2, characters 12-13: syntax error` int a = 1 , `i`nt b = 2 ; (12-13)
testfile-expressions1	√	`File "tests-lex-yacc/mauvais/testfile-expressions1.c", line 1, characters 9-10: syntax error` int a=1+`+`1; (9-10)
testfile-expressions4	√	`File "tests-lex-yacc/mauvais/testfile-expressions4.c", line 2, characters 6-7: syntax error` int a=`.`e1 (6-7)

Test_mauvais	Résultat	Message
testfile-expressions7	√	`File "tests-lex-yacc/mauvais/testfile-expressions7.c", line 2, characters 6-7: syntax error` int a=`i`nt; (6-7)
testfile-functions1	√	`File "tests-lex-yacc/mauvais/testfile-functions1.c", line 1, characters 10-11: syntax error` void f(){}`;` (10-11)
testfile-functions3	√	`File "tests-lex-yacc/mauvais/testfile-functions3.c", line 2, characters 7-8: syntax error` void f `{}` (7-8)
testfile-instructions2	√	`File "tests-lex-yacc/mauvais/testfile-instructions2.c", line 3, characters 5-6: syntax error` __if 1 ; (5-6)
testfile-instructions8	√	`File "tests-lex-yacc/mauvais/testfile-instructions8.c", line 4, characters 0-1: syntax error` `}` (0-1)
testfile-instructions10	√	`File "tests-lex-yacc/mauvais/testfile-instructions10.c", line 3, characters 9-10: syntax error` __while `()` ; (9-10)
testfile-structures1	√	`File "tests-lex-yacc/mauvais/testfile-structures1.c", line 2, characters 0-1: syntax error` `(`0-1, expected a `;`)
testfile-structures2	√	`File "tests-lex-yacc/mauvais/testfile-structures2.c", line 2, characters 36-37: syntax error`

Problème et résolution

1. Parser √

   Au début, nous avons rencontré 20 conflits shift/reduce.

   Nous avons constaté que le problème est sur la règle `expr operateur expr`. En utilisant `%prec`, nous avons résolu 19 conflits.

   Il existe un conflit sur la règle `if (expr) inst else inst`.
   On définit la priorité des `IF ELSE` comme `%nonassoc` et nous avons résolu le problème.

2. Lexer √

J'ai écrit une fonction `newline` qui localise le numéro de lignes, mais ce n'est pas très correct.

```
let newline lexbuf =
    let pos = lexbuf.lex_curr_p in
    lexbuf.lex_curr_p <-
        { pos with pos_lnum = pos.pos_lnum + 1;
          pos_bol = pos.pos_cnum;
          pos_cnum=0 }
```

Enfin, j'ai utilisé la fonction intégrée `Lexing.new_line lexbuf;`

3. Localisation des erreurs √

   Afin de pouvoir localiser les erreurs, je définis la structure de données ident comme: `{id: string; id_loc: loc}`,où `type loc = {fr_start:Lexing.position; fr_end:Lexing.position}` et j'obtiens la position du caractère avec les deux fonction:

```
(* Gets the file range corresponding to the current parser "symbol". *)
let symbol_range () = {
    fr_start = Parsing.symbol_start_pos ();
    fr_end   = Parsing.symbol_end_pos ();
}
 (* Gets the file range corresponding to the specified matching symbol on
the right-hand side of the current rule (indexed from 1). *)
let rhs_range n = {
    fr_start = Parsing.rhs_start_pos n;
    fr_end   = Parsing.rhs_end_pos n;
}
```

   Mais après, en utilisant la fonction intégrée `Parsing.Parse_error`, je n'ai plus besoin ces fonctions.

4. Typage ×

   J'ai essayé de définir une structure de données pour parser, `decl_struct` et `décl_fun` se sont bien déroulés, mais quand j'ai essayé de définir `decl_vars_init`, il y avait plusieurs niveaux de définition de type imbriqué.

   Pour définir `décl_vars_init`, je dois redéfinir `vars_init`; lorsque je définis `vars_init`, je rencontre le problèmes de définition de type pour `(= init)?`.

   Ces fichiers se trouvent dans le répertoire **Error**.