

TP2_a

Generated by Doxygen 1.8.11

Contents

1	Data Structure Index	1
1.1	Data Structures	1
2	File Index	3
2.1	File List	3
3	Data Structure Documentation	5
3.1	TLex Struct Reference	5
3.1.1	Detailed Description	5
3.1.2	Field Documentation	6
3.1.2.1	data	6
3.1.2.2	nbLignes	6
3.1.2.3	nbSymboles	6
3.1.2.4	startPos	6
3.1.2.5	tableSymboles	6
3.2	TSymbole Union Reference	6
3.2.1	Detailed Description	6
3.2.2	Field Documentation	7
3.2.2.1	type	7
3.2.2.2	val	7

4 File Documentation	9
4.1 tp2_a.c File Reference	9
4.1.1 Detailed Description	10
4.1.2 Macro Definition Documentation	11
4.1.2.1 JSON_COLON	11
4.1.2.2 JSON_COMMA	11
4.1.2.3 JSON_FALSE	11
4.1.2.4 JSON_INT_NUMBER	11
4.1.2.5 JSON_LB	11
4.1.2.6 JSON_LCB	11
4.1.2.7 JSON_LEX_ERROR	11
4.1.2.8 JSON_NULL	11
4.1.2.9 JSON_RB	11
4.1.2.10 JSON_RCB	11
4.1.2.11 JSON_REAL_NUMBER	12
4.1.2.12 JSON_STRING	12
4.1.2.13 JSON_TRUE	12
4.1.3 Function Documentation	12
4.1.3.1 addIntSymbolToLexData(TLex *_lexData, const int _val)	12
4.1.3.2 addRealSymbolToLexData(TLex *_lexData, const float _val)	12
4.1.3.3 addStringSymbolToLexData(TLex *_lexData, char *_val)	12
4.1.3.4 deleteLexData(TLex **_lexData)	13
4.1.3.5 initLexData(char *_data)	13
4.1.3.6 isSep(const char _symb)	13
4.1.3.7 lex(TLex *_lexData)	13
4.1.3.8 printLexData(TLex *_lexData)	14
4.1.3.9 subString(TLex *_lex_data, int nbCaracteres)	14
Index	15

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

TLex	Structure contenant tous les parametres/donnees pour l'analyse lexicale	5
TSymbole	Union permettant de manipuler un entier/reel/chaine pour la table des symboles	6

Chapter 2

File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

tp2_a.c	Analyseur lexical pour le langage JSON	9
-------------------------	--	---

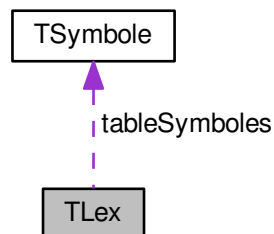
Chapter 3

Data Structure Documentation

3.1 TLex Struct Reference

structure contenant tous les parametres/donnees pour l'analyse lexicale

Collaboration diagram for TLex:



Data Fields

- char * [data](#)
- char * [startPos](#)
- int [nbLignes](#)
- [TSymbole](#) * [tableSymboles](#)
- int [nbSymboles](#)

3.1.1 Detailed Description

structure contenant tous les parametres/donnees pour l'analyse lexicale

3.1.2 Field Documentation

3.1.2.1 `char* TLex::data`

chaîne a parcourir

3.1.2.2 `int TLex::nbLignes`

nb de lignes analysées

3.1.2.3 `int TLex::nbSymboles`

taille du tableau `tableSymboles`

3.1.2.4 `char* TLex::startPos`

position de départ pour la prochaine analyse

3.1.2.5 `TSymbole* TLex::tableSymboles`

tableau des symboles : chaînes/entier/reel

The documentation for this struct was generated from the following file:

- [tp2_a.c](#)

3.2 TSymbole Union Reference

union permettant de manipuler un entier/reel/chaîne pour la table des symboles

Data Fields

- int `type`
- union {
 - int `entier`
 - float `reel`
 - char * `chaîne`
- } `val`

3.2.1 Detailed Description

union permettant de manipuler un entier/reel/chaîne pour la table des symboles

3.2.2 Field Documentation

3.2.2.1 int TSymbole::type

l'un des 3 types suivants : JSON_STRING/JSON_INT_NUMBER/JSON_REAL_NUMBER

3.2.2.2 union { ... } TSymbole::val

valeur associer a un element de la table des symboles

The documentation for this union was generated from the following file:

- [tp2_a.c](#)

Chapter 4

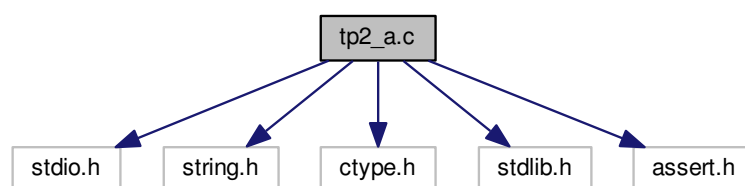
File Documentation

4.1 tp2_a.c File Reference

analyseur lexical pour le langage JSON

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include <stdlib.h>
#include <assert.h>
```

Include dependency graph for tp2_a.c:



Data Structures

- union [TSymbole](#)

union permettant de manipuler un entier/reel/chaine pour la table des symboles

- struct [TLex](#)

structure contenant tous les parametres/donnees pour l'analyse lexicale

Macros

- `#define JSON_LEX_ERROR -1`
- `#define JSON_TRUE 1`
- `#define JSON_FALSE 2`
- `#define JSON_NULL 3`
- `#define JSON_LCB 4`
- `#define JSON_RCB 5`
- `#define JSON_LB 6`
- `#define JSON_RB 7`
- `#define JSON_COMMA 8`
- `#define JSON_COLON 9`
- `#define JSON_STRING 10`
- `#define JSON_INT_NUMBER 11`
- `#define JSON_REAL_NUMBER 12`

Functions

- `char * subString (TLex *lex_data, int nbCaracteres)`
fonction qui rogne une chaine de caracteres
- `int isSep (const char _symb)`
fonction qui teste si un symbole fait partie des separateurs
- `TLex * initLexData (char *_data)`
fonction qui reserve la memoire et initialise les donnees pour l'analyseur lexical
- `void deleteLexData (TLex **_lexData)`
fonction qui supprime de la memoire les donnees pour l'analyseur lexical
- `void printLexData (TLex *_lexData)`
fonction qui affiche les donnees pour l'analyseur lexical
- `void addIntSymbolToLexData (TLex *_lexData, const int _val)`
fonction qui ajoute un symbole entier a la table des symboles
- `void addRealSymbolToLexData (TLex *_lexData, const float _val)`
fonction qui ajoute un symbole reel a la table des symboles
- `void addStringSymbolToLexData (TLex *_lexData, char *_val)`
fonction qui ajoute une chaine de caracteres a la table des symboles
- `int lex (TLex *_lexData)`
fonction qui effectue l'analyse lexicale (contient le code l'automate fini)
- `int main ()`
fonction principale

4.1.1 Detailed Description

analyseur lexical pour le langage JSON

Author

NM
Rémy BOUTELOUP
Pierrick BOBET

Version

0.1

Date

02/01/2017

4.1.2 Macro Definition Documentation

4.1.2.1 #define JSON_COLON 9

entite lexicale :

4.1.2.2 #define JSON_COMMA 8

entite lexicale ,

4.1.2.3 #define JSON_FALSE 2

entite lexicale false

4.1.2.4 #define JSON_INT_NUMBER 11

entite lexicale nombre entier

4.1.2.5 #define JSON_LB 6

entite lexicale [

4.1.2.6 #define JSON_LCB 4

entite lexicale {

4.1.2.7 #define JSON_LEX_ERROR -1

code d'erreur lexicale

4.1.2.8 #define JSON_NULL 3

entite lexicale null

4.1.2.9 #define JSON_RB 7

entite lexicale]

4.1.2.10 #define JSON_RCB 5

entite lexicale }

4.1.2.11 #define JSON_REAL_NUMBER 12

entite lexicale nombre reel

4.1.2.12 #define JSON_STRING 10

entite lexicale chaine de caracteres

4.1.2.13 #define JSON_TRUE 1

entite lexicale true

4.1.3 Function Documentation

4.1.3.1 void addIntSymbolToLexData (TLex * *_lexData*, const int *_val*)

fonction qui ajoute un symbole entier a la table des symboles

Parameters

	<i>_lexData</i>	donnees de l'analyseur lexical
in	<i>_val</i>	valeur entiere a ajouter

Returns

neant

4.1.3.2 void addRealSymbolToLexData (TLex * *_lexData*, const float *_val*)

fonction qui ajoute un symbole reel a la table des symboles

Parameters

	<i>_lexData</i>	donnees de l'analyseur lexical
in	<i>_val</i>	valeur reelle a ajouter

4.1.3.3 void addStringSymbolToLexData (TLex * *_lexData*, char * *_val*)

fonction qui ajoute une chaine de caracteres a la table des symboles

Parameters

	<i>_lexData</i>	donnees de l'analyseur lexical
in	<i>_val</i>	chaine a ajouter

4.1.3.4 void deleteLexData (TLex ** _lexData)

fonction qui supprime de la memoire les donnees pour l'analyseur lexical

Parameters

<code>_lexData</code>	donnees de l'analyseur lexical
-----------------------	--------------------------------

Returns

neant

4.1.3.5 TLex * initLexData (char * _data)

fonction qui reserve la memoire et initialise les donnees pour l'analyseur lexical

Parameters

in	<code>_data</code>	chaîne a analyser
----	--------------------	-------------------

Returns

pointeur sur la structure de donnees creee

4.1.3.6 int isSep (const char _symb)

fonction qui teste si un symbole fait partie des separateurs

Parameters

in	<code>_symb</code>	symbole a analyser
----	--------------------	--------------------

Returns

1 (vrai) si `_symb` est un separateur, 0 (faux) sinon

4.1.3.7 int lex (TLex * _lexData)

fonction qui effectue l'analyse lexicale (contient le code l'automate fini)

Parameters

<code>_lexData</code>	donnees de suivi de l'analyse lexicale
-----------------------	--

Returns

code d'identification de l'entite lexicale trouvee

4.1.3.8 void printLexData (TLex * *_lexData*)

fonction qui affiche les donnees pour l'analyseur lexical

Parameters

<i>_lexData</i>	donnees de l'analyseur lexical
-----------------	--------------------------------

Returns

neant

4.1.3.9 char * subString (TLex * *lex_data*, int *nbCaracteres*)

fonction qui rogne une chaine de caracteres

Parameters

	<i>lex_data</i>	donnees de suivi de l'analyse lexicale
in	<i>nbCaracteres</i>	le nombre de caracteres a supprimer

Returns

la nouvelle chaine de caracteres

Index

addIntSymbolToLexData
 tp2_a.c, [12](#)
addRealSymbolToLexData
 tp2_a.c, [12](#)
addStringSymbolToLexData
 tp2_a.c, [12](#)

data
 TLex, [6](#)
deleteLexData
 tp2_a.c, [13](#)

initLexData
 tp2_a.c, [13](#)
isSep
 tp2_a.c, [13](#)

JSON_COLON
 tp2_a.c, [11](#)
JSON_COMMA
 tp2_a.c, [11](#)
JSON_FALSE
 tp2_a.c, [11](#)
JSON_INT_NUMBER
 tp2_a.c, [11](#)
JSON_LCB
 tp2_a.c, [11](#)
JSON_LEX_ERROR
 tp2_a.c, [11](#)
JSON_LB
 tp2_a.c, [11](#)
JSON_NULL
 tp2_a.c, [11](#)
JSON_RCB
 tp2_a.c, [11](#)
JSON_REAL_NUMBER
 tp2_a.c, [11](#)
JSON_RB
 tp2_a.c, [11](#)
JSON_STRING
 tp2_a.c, [12](#)
JSON_TRUE
 tp2_a.c, [12](#)

lex
 tp2_a.c, [13](#)

nbLignes
 TLex, [6](#)
nbSymboles
 TLex, [6](#)

printLexData
 tp2_a.c, [14](#)

startPos
 TLex, [6](#)
subString
 tp2_a.c, [14](#)

TLex, [5](#)
 data, [6](#)
 nbLignes, [6](#)
 nbSymboles, [6](#)
 startPos, [6](#)
 tableSymboles, [6](#)

TSymbole, [6](#)
 type, [7](#)
 val, [7](#)

tableSymboles
 TLex, [6](#)

tp2_a.c, [9](#)
 addIntSymbolToLexData, [12](#)
 addRealSymbolToLexData, [12](#)
 addStringSymbolToLexData, [12](#)
 deleteLexData, [13](#)
 initLexData, [13](#)
 isSep, [13](#)
 JSON_COLON, [11](#)
 JSON_COMMA, [11](#)
 JSON_FALSE, [11](#)
 JSON_INT_NUMBER, [11](#)
 JSON_LCB, [11](#)
 JSON_LEX_ERROR, [11](#)
 JSON_LB, [11](#)
 JSON_NULL, [11](#)
 JSON_RCB, [11](#)
 JSON_REAL_NUMBER, [11](#)
 JSON_RB, [11](#)
 JSON_STRING, [12](#)
 JSON_TRUE, [12](#)
 lex, [13](#)
 printLexData, [14](#)
 subString, [14](#)

type
 TSymbole, [7](#)

val
 TSymbole, [7](#)