Compilador

Generado por Doxygen 1.14.0

Chapter 1

Índice de clases

1.1 Lista de clases

Lista de clases, estructuras, uniones e interfaces con breves descripciones:

Chapter 2

Índice de archivos

2.1 Lista de archivos

Lista de todos los archivos con breves descripciones:

Chapter 3

Documentación de clases

3.1 Referencia de la estructura token_t

Definición de la estructura Token.

`#include <lexer.h>`

Atributos públicos

- TokenType type
- char * lexeme
- size_t line
- size_t column
- struct token_t * next

3.1.1 Descripción detallada

Definición de la estructura Token.

3.1.2 Documentación de datos miembro

3.1.2.1 column

`size_t token_t::column`

Columna donde se encontró el token

3.1.2.2 lexeme

`char* token_t::lexeme`

Lexema del token

3.1.2.3 line

```
size_t token_t::line
```

Línea donde se encontró el token

3.1.2.4 next

```
struct token_t* token_t::next
```

Puntero al siguiente token (para lista enlazada)

3.1.2.5 type

```
TokenType token_t::type
```

Tipo de token

La documentación de esta estructura está generada del siguiente archivo:

- include/lexer.h

Chapter 4

Documentación de archivos

4.1 Referencia del archivo include/keywords.h

Declaración de la función is_keyword.

```
#include <stdbool.h>
```

Funciones

- bool is_keyword (const char *lexeme)

4.1.1 Descripción detallada

Declaración de la función is_keyword.

Contiene la declaración de la función is_keyword que verifica si un lexema es una palabra clave.

4.1.2 Documentación de funciones

4.1.2.1 is_keyword()

```
bool is_keyword (
            const char * lexeme)
```

4.2 keywords.h

Ir a la documentación de este archivo.

```
00001
00007
00008 #ifndef KEYWORDS_H
00009 #define KEYWORDS_H
00010
00011 #include <stdbool.h>
00012
00013 bool is_keyword(const char *lexeme);
00014
00015 #endif
```

4.3 Referencia del archivo include/lexer.h

Definición del enum TokenType.

```
#include <stddef.h>
#include <stdint.h>
```

Clases

- struct token_t

  Definición de la estructura Token.

typedefs

- typedef enum TokenType TokenType

  Definición del enum TokenType.
- typedef struct token_t token_t

  Definición de la estructura Token.

Enumeraciones

- enum TokenType {
  TOKEN_IDENTIFIER , TOKEN_NUMBER , TOKEN_STRING , TOKEN_OPERATOR ,
  TOKEN_DELIMITER , TOKEN_KEYWORD , TOKEN_UNKNOWN , TOKEN_EOF }

  Definición del enum TokenType.

Funciones

- token_t ∗ create_token (TokenType type, const char ∗lexeme, size_t line, size_t column)

  Crea un nuevo token.
- void free_token (token_t ∗token)

  Libera un token.
- token_t ∗ get_next_token (const char ∗source)

  Función principal del lexer que procesa la fuente y devuelve el siguiente token.

4.3.1 Descripción detallada

Definición del enum TokenType.

Contiene la definición del enum TokenType que representa los diferentes tipos de tokens que el lexer puede identificar en el código fuente.

4.3.2 Documentación de «typedef»

4.3.2.1 token_t

```
typedef struct token_t token_t
```

Definición de la estructura Token.

4.3.2.2 TokenType

```
typedef enum TokenType TokenType
```

Definición del enum TokenType.

4.3.3 Documentación de enumeraciones

4.3.3.1 TokenType

```
enum TokenType
```

Definición del enum TokenType.

Valores de enumeraciones

TOKEN_IDENTIFIER	Identificador
TOKEN_NUMBER	Número
TOKEN_STRING	Cadena de texto
TOKEN_OPERATOR	Operador
TOKEN_DELIMITER	Delimitador
TOKEN_KEYWORD	Palabra clave
TOKEN_UNKNOWN	Token desconocido
TOKEN_EOF	Fin de archivo

4.3.4 Documentación de funciones

4.3.4.1 create_token()

```
token_t * create_token (
            TokenType type,
            const char * lexeme,
            size_t line,
            size_t column)
```

Crea un nuevo token.

Parámetros

type	Tipo de token
lexeme	Lexema del token
line	Línea donde se encontró el token
column	Columna donde se encontró el token

Devuelve

Puntero al token creado

4.3.4.2 free_token()

```
void free_token (
            token_t * token)
```

Libera un token.

Parámetros

token	Puntero al token a liberar

Devuelve

void

4.3.4.3 get_next_token()

```
token_t * get_next_token (
            const char * source)
```

Función principal del lexer que procesa la fuente y devuelve el siguiente token.

Parámetros

source	Fuente de código a analizar

Devuelve

Puntero al siguiente token encontrado

4.4 lexer.h

Ir a la documentación de este archivo.

```
00001
00008
00009 #ifndef LEXER_H
00010 #define LEXER_H
00011 #include <stddef.h>
00012 #include <stdint.h>
00013
00017 typedef enum TokenType {
00018     TOKEN_IDENTIFIER,
00019     TOKEN_NUMBER,
00020     TOKEN_STRING,
00021     TOKEN_OPERATOR,
00022     TOKEN_DELIMITER,
00023     TOKEN_KEYWORD,
00024     TOKEN_UNKNOWN,
00025     TOKEN_EOF
00026 } TokenType;
00027
00031 typedef struct token_t{
00032     TokenType type;
00033     char *lexeme;
00034     size_t line;
00035     size_t column;
00036     struct token_t *next;
00037 } token_t;
00038
00039 token_t *create_token(TokenType type, const char *lexeme,size_t line, size_t column);
00040 void free_token(token_t *token);
00041 token_t *get_next_token(const char *source);
00042
00043 #endif // LEXER_H
```

4.5 Referencia del archivo src/lexer/keywords.c

Implementación de la verificación de palabras clave.

```
#include "keywords.h"
#include <string.h>
```

Funciones

- bool is_keyword (const char *lexeme)

Variables

- static const char * keywords [ ]

  Lista de palabras clave del lenguaje.

4.5.1 Descripción detallada

Implementación de la verificación de palabras clave.

Contiene la implementación de la función para verificar si un lexema es una palabra clave.

4.5.2 Documentación de funciones

4.5.2.1 is_keyword()

```
bool is_keyword (
            const char * lexeme)
```

4.5.3 Documentación de variables

4.5.3.1 keywords

```
const char* keywords[]  [static]
```

Valor inicial:

```
= {
    "fn",
    "let",
    "mut",
    "if",
    "else",
    "match",
    "while",
    "loop",
    "for",
    "in",
    "break",
    "continue",
    "return",
    "true",
    "false",
    NULL
}
```

Lista de palabras clave del lenguaje.

La lista debe coincidir con las definidas en el lexer.

4.6 Referencia del archivo src/lexer/lexer.c

Implementación de estructuras y funciones del lexer.

```
#include "lexer.h"
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
```

defines

- #define NUM_STATES 31
- #define NUM_CHAR_TYPES 24

typedefs

- typedef enum CharType CharType

Enumeraciones

- enum State {
  STATE_START , STATE_IDENTIFIER , STATE_INT , STATE_SIGN ,
  STATE_BIN_PREFIX , STATE_BIN , STATE_HEX_PREFIX , STATE_HEX ,
  STATE_REAL , STATE_REAL_FRACTION , STATE_EXPONENT_MARK , STATE_EXPONENT_SIGN ,
  STATE_EXPONENT , STATE_STRING , STATE_STRING_ESCAPE , STATE_CHAR ,
  STATE_CHAR_ESCAPE , STATE_CHAR_END , STATE_SLASH , STATE_COMMENT_LINE ,
  STATE_COMMENT_BLOCK , STATE_COMMENT_BLOCK_END , STATE_OPERATOR , STATE_OPERATOR_EQ
  ,
  STATE_OPERATOR_AND , STATE_OPERATOR_OR , STATE_DELIMITER , STATE_WHITESPACE ,
  STATE_FINAL , STATE_ERROR , STATE_EOF }
- enum CharType {
  CHAR_LETTER , CHAR_DIGIT , CHAR_UNDERSCORE , CHAR_QUOTE ,
  CHAR_APOSTROPHE , CHAR_BACKSLASH , CHAR_PLUS , CHAR_MINUS ,
  CHAR_STAR , CHAR_SLASH , CHAR_PERCENT , CHAR_EQUAL ,
  CHAR_EXCLAMATION , CHAR_AMPERSAND , CHAR_PIPE , CHAR_LT ,
  CHAR_GT , CHAR_HEXLETTER , CHAR_DOT , CHAR_DELIMITER ,
  CHAR_WHITESPACE , CHAR_NEWLINE , CHAR_EOF , CHAR_UNKNOWN }

Funciones

- const char * char_type_to_string (CharType type)

  Convierte un CharType a su representación en cadena.
- CharType get_char_type (int c)

  Obtiene el tipo de carácter de un carácter dado.
- token_t * create_token (TokenType type, const char *lexeme, size_t line, size_t column)

  Crea un nuevo token.
- void free_token (token_t *token)

  Libera un token.
- token_t * get_next_token (const char *source)

  Función principal del lexer que procesa la fuente y devuelve el siguiente token.
- char * read_file (const char *filename)

  Lee el contenido de un archivo y lo devuelve como una cadena.
- int main ()

4.6.1 Descripción detallada

Implementación de estructuras y funciones del lexer.

Contiene la implementación de las funciones para crear y liberar tokens, así como la definición de la estructura token_t.

4.6.2 Documentación de «define»

4.6.2.1 NUM_CHAR_TYPES

```
#define NUM_CHAR_TYPES 24
```

4.6.2.2 NUM_STATES

```
#define NUM_STATES 31
```

4.6.3 Documentación de «typedef»

4.6.3.1 CharType

```
typedef enum CharType CharType
```

4.6.4 Documentación de enumeraciones

4.6.4.1 CharType

```
enum CharType
```

Valores de enumeraciones

CHAR_LETTER	
CHAR_DIGIT	
CHAR_UNDERSCORE	
CHAR_QUOTE	
CHAR_APOSTROPHE	
CHAR_BACKSLASH	
CHAR_PLUS	
CHAR_MINUS	
CHAR_STAR	
CHAR_SLASH	
CHAR_PERCENT	
CHAR_EQUAL	
CHAR_EXCLAMATION	
CHAR_AMPERSAND	
CHAR_PIPE	
CHAR_LT	

Valores de enumeraciones

CHAR_GT	
CHAR_HEXLETTER	
CHAR_DOT	
CHAR_DELIMITER	
CHAR_WHITESPACE	
CHAR_NEWLINE	
CHAR_EOF	
CHAR_UNKNOWN	

4.6.4.2 State

```
enum State
```

Valores de enumeraciones

STATE_START	
STATE_IDENTIFIER	
STATE_INT	
STATE_SIGN	
STATE_BIN_PREFIX	
STATE_BIN	
STATE_HEX_PREFIX	
STATE_HEX	
STATE_REAL	
STATE_REAL_FRACTION	
STATE_EXPONENT_MARK	
STATE_EXPONENT_SIGN	
STATE_EXPONENT	
STATE_STRING	
STATE_STRING_ESCAPE	
STATE_CHAR	
STATE_CHAR_ESCAPE	
STATE_CHAR_END	
STATE_SLASH	
STATE_COMMENT_LINE	
STATE_COMMENT_BLOCK	
STATE_COMMENT_BLOCK_END	
STATE_OPERATOR	
STATE_OPERATOR_EQ	
STATE_OPERATOR_AND	
STATE_OPERATOR_OR	
STATE_DELIMITER	
STATE_WHITESPACE	
STATE_FINAL	
STATE_ERROR	
STATE_EOF	

4.6.5 Documentación de funciones

4.6.5.1 char_type_to_string()

```
const char * char_type_to_string (
            CharType type)
```

Convierte un CharType a su representación en cadena.

Parámetros

type	Tipo de carácter

Devuelve

Cadena que representa el tipo de carácter

4.6.5.2 create_token()

```
token_t * create_token (
            TokenType type,
            const char * lexeme,
            size_t line,
            size_t column)
```

Crea un nuevo token.

Parámetros

type	Tipo de token
lexeme	Lexema del token
line	Línea donde se encontró el token
column	Columna donde se encontró el token

Devuelve

Puntero al token creado

4.6.5.3 free_token()

```
void free_token (
            token_t * token)
```

Libera un token.

Parámetros

token	Puntero al token a liberar

Devuelve

void

4.6.5.4 get_char_type()

```
CharType get_char_type (
            int c)
```

Obtiene el tipo de carácter de un carácter dado.

Parámetros

c	Carácter a evaluar

Devuelve

Tipo de carácter correspondiente

4.6.5.5 get_next_token()

```
token_t * get_next_token (
            const char * source)
```

Función principal del lexer que procesa la fuente y devuelve el siguiente token.

Parámetros

source	Fuente de código a analizar

Devuelve

Puntero al siguiente token encontrado

4.6.5.6 main()

```
int main ()
```

4.6.5.7 read_file()

```
char * read_file (
            const char * filename)
```

Lee el contenido de un archivo y lo devuelve como una cadena.

Parámetros

filename	Nombre del archivo a leer

Devuelve

Cadena con el contenido del archivo, o NULL si hubo un error

4.7 Referencia del archivo src/main.c

Punto de entrada del compilador.

```
#include <stdio.h>
#include "lexer.h"
```

Funciones

- int main (int argc, char ∗argv[ ])

4.7.1 Descripción detallada

Punto de entrada del compilador.

Este archivo contiene la función principal del compilador que inicializa el lexer, procesa el archivo fuente y maneja los tokens generados.

4.7.2 Documentación de funciones

4.7.2.1 main()

```
int main (
            int argc,
            char * argv[ ])
```