Draw ++ Compilator

Generated by Doxygen 1.9.8

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

Chapter 3

Namespace Documentation

3.1 interpreter Namespace Reference

Functions

- run_file (file_path)
- run_interactive ()
- main ()

Variables

- bool DEBUG = False

3.1.1 Function Documentation

3.1.1.1 main()

```
interpreter.main ( )
```

3.1.1.2 run_file()

```
interpreter.run_file (
              file_path )
```

```
Executes a Draw++ source file.
```

3.1.1.3 run_interactive()

```
interpreter.run_interactive ( )
```

```
Interactive mode for on-the-fly drawing.
```

3.1.2 Variable Documentation

3.1.2.1 DEBUG

```
bool interpreter.DEBUG = False
```

3.2 lexer Namespace Reference

Functions

- t_IDENTIFIER (t)
- t_NUMBER (t)
- t_STRING (t)
- t_BOOLEAN (t)
- t_newline (t)
- t_COMMENT (t)
- t_COMMENT_MULTILINE (t)
- t_error (t)
- find_column (input_text, lexpos)
- init_lexer ()

Variables

- dict keywords
- list tokens
- str t_PLUS = r'\+'
- str t_MINUS = r'-'
- str t_TIMES = r'*'
- str t_DIVIDE = r'/'
- str t_EQUALS = r'='
- str t_LT = r'<'
- str t_GT = r'>'
- str t_LE = r'<='
- str t_GE = r'>='
- str t_EQ = r'=='
- str t_NEQ = r'!='
- str t_AND = r'and'
- str t_OR = r'or'
- str t_NOT = r'not'
- str t_LPAREN = r'\('
- str t_RPAREN = r'\)'
- str t_LBRACE = r'\{'
- str t_RBRACE = r'\}'
- str t_LBRACKET = r'\['
- str t_RBRACKET = r'\]'
- str t_COMMA = r','
- str t_SEMICOLON = r';'
- str t_COLON = r':'
- str t_ignore = ' \t'

3.2.1 Function Documentation

3.2.1.1 find_column()

```
lexer.find_column (
              input_text,
              lexpos )
```

3.2.1.2 init_lexer()

```
lexer.init_lexer ( )
```

3.2.1.3 t_BOOLEAN()

```
lexer.t_BOOLEAN (
              t )
```

3.2.1.4 t_COMMENT()

```
lexer.t_COMMENT (
              t )
```

3.2.1.5 t_COMMENT_MULTILINE()

```
lexer.t_COMMENT_MULTILINE (
              t )
```

3.2.1.6 t_error()

```
lexer.t_error (
              t )
```

3.2.1.7 t_IDENTIFIER()

```
lexer.t_IDENTIFIER (
              t )
```

3.2.1.8 t_newline()

```
lexer.t_newline (
              t )
```

3.2.1.9 t_NUMBER()

```
lexer.t_NUMBER (
              t )
```

3.2.1.10 t_STRING()

```
lexer.t_STRING (
              t )
```

3.2.2 Variable Documentation

3.2.2.1 keywords

```
dict lexer.keywords
```

Initial value:

```
=  {
    'if': 'IF',
    'else': 'ELSE',
    'for': 'FOR',
    'while': 'WHILE',
    'draw': 'DRAW',
    'cursor': 'CURSOR',
    'move': 'MOVE',
    'color': 'COLOR',
    'animate': 'ANIMATE',
    'rotate': 'ROTATE',
    'var': 'VARIABLE',
    'line': 'LINE',
    'circle': 'CIRCLE',
    'square': 'SQUARE',
    'arc': 'ARC',
    'point': 'POINT',
}
```

3.2.2.2 t_AND

```
str lexer.t_AND = r'and'
```

3.2.2.3 t_COLON

```
str lexer.t_COLON = r':'
```

3.2.2.4 t_COMMA

```
str lexer.t_COMMA = r','
```

3.2.2.5 t_DIVIDE

```
str lexer.t_DIVIDE = r'/'
```

3.2.2.6 t_EQ

```
str lexer.t_EQ = r'=='
```

3.2.2.7 t_EQUALS

```
str lexer.t_EQUALS = r'='
```

3.2.2.8 t_GE

```
str lexer.t_GE = r'>='
```

3.2.2.9 t_GT

```
str lexer.t_GT = r'>'
```

3.2.2.10 t_ignore

```
str lexer.t_ignore = ' \t'
```

3.2.2.11 t_LBRACE

```
str lexer.t_LBRACE = r'\{'
```

3.2.2.12 t_LBRACKET

```
str lexer.t_LBRACKET = r'\['
```

3.2.2.13 t_LE

```
str lexer.t_LE = r'<='
```

3.2.2.14 t_LPAREN

```
str lexer.t_LPAREN = r'\('
```

3.2.2.15 t_LT

```
str lexer.t_LT = r'<'
```

3.2.2.16 t_MINUS

```
str lexer.t_MINUS = r'-'
```

3.2.2.17 t_NEQ

```
str lexer.t_NEQ = r'!='
```

3.2.2.18 t_NOT

```
str lexer.t_NOT = r'not'
```

3.2.2.19 t_OR

```
str lexer.t_OR = r'or'
```

3.2.2.20 t_PLUS

```
str lexer.t_PLUS = r'\+'
```

3.2.2.21 t_RBRACE

```
str lexer.t_RBRACE = r'\}'
```

3.2.2.22 t_RBRACKET

```
str lexer.t_RBRACKET = r'\]'
```

3.2.2.23 t_RPAREN

```
str lexer.t_RPAREN = r'\)'
```

3.2.2.24 t_SEMICOLON

```
str lexer.t_SEMICOLON = r';'
```

3.2.2.25 t_TIMES

```
str lexer.t_TIMES = r'\*'
```

3.2.2.26 tokens

```
list lexer.tokens
```

Initial value:

```
=  [
    'IDENTIFIER', 'NUMBER', 'STRING', 'BOOLEAN',
    'PLUS', 'MINUS', 'TIMES', 'DIVIDE', 'EQUALS',
    'LT', 'GT', 'LE', 'GE', 'EQ', 'NEQ', 'AND', 'OR', 'NOT',
    'LPAREN', 'RPAREN', 'LBRACE', 'RBRACE', 'LBRACKET', 'RBRACKET',
    'COMMA', 'SEMICOLON', 'COLON',
] + list(keywords.values())
```

3.3 myast Namespace Reference

Functions

- print_error (error)
- condition_to_c (condition)
- resolve_value_and_find_variable (ast, value, current_position=None)
- translate_node_to_c (ast, node, newline, tabulation, semicolon, current_position=None)
- translate_ast_to_c (ast)
- execute_ast (ast, debug)

3.3.1 Function Documentation

3.3.1.1 condition_to_c()

```
myast.condition_to_c (
              condition )
```

3.3.1.2 execute_ast()

```
myast.execute_ast (
              ast,
              debug )
```

Exécute l'AST généré par le parser.

3.3.1.3 print_error()

```
myast.print_error (
              error )
```

3.3.1.4 resolve_value_and_find_variable()

```
myast.resolve_value_and_find_variable (
              ast,
              value,
              current_position = None )
```

```
Resolve the type and value of a variable or literal.
- Ensures the variable is initialized before the current position.
- Handles literals, variables, operations, and nested AST structures.
```

3.3.1.5 translate_ast_to_c()

```
myast.translate_ast_to_c (
              ast )
```

```
Traduire l'AST en code C.
```

3.3.1.6 translate_node_to_c()

```
myast.translate_node_to_c (
              ast,
              node,
              newline,
              tabulation,
              semicolon,
              current_position = None )
```

```
Traduire une node en code C.
```

3.4 parser Namespace Reference

Functions

- p_programme (p)
- p_instruction (p)
- p_dessin (p)
- p_forme (p)
- p_parametres (p)
- p_arg (p)
- p_expression_arithmetic (p)
- p_expression_arithmetic_atomic (p)
- p_deplacement (p)
- p_rotation (p)
- p_couleur (p)
- p_assignation (p)
- p_modification (p)
- p_valeur (p)
- p_bloc (p)
- p_conditionnelle (p)
- p_expression_logique (p)
- p_operateur_comparaison (p)
- p_boucle (p)
- p_error (p)
- init_parser ()

3.4.1 Function Documentation

3.4.1.1 init_parser()

```
parser.init_parser ( )
```

3.4.1.2 p_arg()

```
parser.p_arg (
              p )
```

```
arg : NUMBER
       | IDENTIFIER
       | STRING
       | BOOLEAN
       | expression_arithmetic
```

3.4.1.3 p_assignation()

```
parser.p_assignation (
              p )
```

```
assignation : VARIABLE IDENTIFIER EQUALS arg
```

3.4.1.4 p_bloc()

```
parser.p_bloc (
              p )
```

```
bloc : LBRACE programme RBRACE
```

3.4.1.5 p_boucle()

```
parser.p_boucle (
              p )
```

```
boucle : WHILE LPAREN expression_logique RPAREN bloc
          | FOR LPAREN assignation SEMICOLON expression_logique SEMICOLON modification RPAREN bloc
```

3.4.1.6 p_conditionnelle()

```
parser.p_conditionnelle (
              p )
```

```
conditionnelle : IF LPAREN expression_logique RPAREN bloc ELSE bloc
                 | IF LPAREN expression_logique RPAREN bloc
```

3.4.1.7 p_couleur()

```
parser.p_couleur (
              p )
```

```
couleur : COLOR LPAREN arg RPAREN
```

3.4.1.8 p_deplacement()

```
parser.p_deplacement (
              p )
```

```
deplacement : MOVE LPAREN arg COMMA arg RPAREN
```

3.4.1.9 p_dessin()

```
parser.p_dessin (
              p )
```

```
dessin : DRAW forme LPAREN parametres RPAREN
```

3.4.1.10 p_error()

```
parser.p_error (
              p )
```

3.4.1.11 p_expression_arithmetic()

```
parser.p_expression_arithmetic (
              p )
```

```
expression_arithmetic : expression_arithmetic PLUS expression_arithmetic
                         | expression_arithmetic MINUS expression_arithmetic
                         | expression_arithmetic TIMES expression_arithmetic
                         | expression_arithmetic DIVIDE expression_arithmetic
```

3.4.1.12 p_expression_arithmetic_atomic()

```
parser.p_expression_arithmetic_atomic (
              p )
```

```
expression_arithmetic : LPAREN expression_arithmetic RPAREN
                         | NUMBER
                         | IDENTIFIER
```

3.4.1.13 p_expression_logique()

```
parser.p_expression_logique (
              p )
```

```
expression_logique : valeur operateur_comparaison valeur
                      | valeur operateur_comparaison BOOLEAN
                      | BOOLEAN operateur_comparaison valeur
                      | BOOLEAN
```

3.4.1.14 p_forme()

```
parser.p_forme (
              p )
```

```
forme : LINE
         | CIRCLE
         | SQUARE
         | ARC
         | POINT
```

3.4.1.15 p_instruction()

```
parser.p_instruction (
              p )
```

```
instruction : dessin
               | deplacement
               | rotation
               | couleur
               | assignation
               | modification
               | conditionnelle
               | boucle
               | bloc
```

3.4.1.16 p_modification()

```
parser.p_modification (
              p )
```

```
modification : IDENTIFIER EQUALS arg
```

3.4.1.17 p_operateur_comparaison()

```
parser.p_operateur_comparaison (
              p )
```

```
operateur_comparaison : EQ
                         | NEQ
                         | LT
                         | GT
                         | LE
                         | GE
```

3.4.1.18 p_parametres()

```
parser.p_parametres (
              p )
```

```
parametres : parametres COMMA arg
              | arg
```

3.4.1.19 p_programme()

```
parser.p_programme (
              p )
```

```
programme : instruction
             | programme instruction
```

3.4.1.20 p_rotation()

```
parser.p_rotation (
              p )
```

```
rotation : ROTATE LPAREN arg RPAREN
```

3.4.1.21 p_valeur()

```
parser.p_valeur (
              p )
```

```
valeur : NUMBER
          | IDENTIFIER
          | STRING
          | BOOLEAN
```

Chapter 4

File Documentation

4.1 interpreter.py File Reference

Namespaces

- namespace interpreter

Functions

- interpreter.run_file (file_path)
- interpreter.run_interactive ()
- interpreter.main ()

Variables

- bool interpreter.DEBUG = False

4.2 lexer.py File Reference

Namespaces

- namespace lexer

Functions

- lexer.t_IDENTIFIER (t)
- lexer.t_NUMBER (t)
- lexer.t_STRING (t)
- lexer.t_BOOLEAN (t)
- lexer.t_newline (t)
- lexer.t_COMMENT (t)
- lexer.t_COMMENT_MULTILINE (t)
- lexer.t_error (t)
- lexer.find_column (input_text, lexpos)
- lexer.init_lexer ()

Variables

- dict lexer.keywords
- list lexer.tokens
- str lexer.t_PLUS = r'\+'
- str lexer.t_MINUS = r'-'
- str lexer.t_TIMES = r'*'
- str lexer.t_DIVIDE = r'/'
- str lexer.t_EQUALS = r'='
- str lexer.t_LT = r'<'
- str lexer.t_GT = r'>'
- str lexer.t_LE = r'<='
- str lexer.t_GE = r'>='
- str lexer.t_EQ = r'=='
- str lexer.t_NEQ = r'!='
- str lexer.t_AND = r'and'
- str lexer.t_OR = r'or'
- str lexer.t_NOT = r'not'
- str lexer.t_LPAREN = r'\('
- str lexer.t_RPAREN = r'\)'
- str lexer.t_LBRACE = r'\{'
- str lexer.t_RBRACE = r'\}'
- str lexer.t_LBRACKET = r'\['
- str lexer.t_RBRACKET = r'\]'
- str lexer.t_COMMA = r','
- str lexer.t_SEMICOLON = r';'
- str lexer.t_COLON = r':'
- str lexer.t_ignore = ' \t'

4.3 myast.py File Reference

Namespaces

- namespace myast

Functions

- myast.print_error (error)
- myast.condition_to_c (condition)
- myast.resolve_value_and_find_variable (ast, value, current_position=None)
- myast.translate_node_to_c (ast, node, newline, tabulation, semicolon, current_position=None)
- myast.translate_ast_to_c (ast)
- myast.execute_ast (ast, debug)

4.4 parser.py File Reference

Namespaces

- namespace parser

Functions

- parser.p_programme (p)
- parser.p_instruction (p)
- parser.p_dessin (p)
- parser.p_forme (p)
- parser.p_parametres (p)
- parser.p_arg (p)
- parser.p_expression_arithmetic (p)
- parser.p_expression_arithmetic_atomic (p)
- parser.p_deplacement (p)
- parser.p_rotation (p)
- parser.p_couleur (p)
- parser.p_assignation (p)
- parser.p_modification (p)
- parser.p_valeur (p)
- parser.p_bloc (p)
- parser.p_conditionnelle (p)
- parser.p_expression_logique (p)
- parser.p_operateur_comparaison (p)
- parser.p_boucle (p)
- parser.p_error (p)
- parser.init_parser ()

Index