LEXIC

Alphabet:

a. Upper (A-Z) and lower case letters (a-z) of the English alphabet

b. Underline character '_'

c. Decimal digits (0-9)

Lexic:

a.Special symbols, representing:

- operators:

+, -, *, /, %

==, <, >, <=, >=, !=

and, or, !

&, |, ~, <<, >>

=, /=, +=, -=, *=, %=, &=, |=, ^=, >>=, <<=

++, --

? :, ::, sizeof, static/dynamic/const/reinterpret_cast

- separators: ( ), [ ], { }, :, ;, space, ..., ->, ., ::

- reserved words:

if, else, switch, case, default, while, do, for, break, continue, goto,

return, int, float, double, char, void, bool, wchar_t, long, short, signed,

unsigned, const, volatile, auto, register, static, extern, mutable, private,

protected, public, new, delete, try, catch, throw, typeid, sizeof, this,

operator, dynamic_cast, static_cast, reinterpret_cast, const_cast, namespace,

using, class, struct, union, Enum, virtual, friend, explicit, inline, template,

true, false, nullptr, typedef, typename, decltype, constexpr, noexcept, NULL,

override, final, char16_t, char32_t, alignof, alignas, thread_local, static_assert

then var while write

b.identifiers

-a sequence of letters and digits, such that the first character is a letter; the rule is:

identifier ::= letter/underline | {underline}{letter}{digit}

letter ::= "A" | "B" | ... | "Z" | "a" | "b" | ... | "z"

digit ::= "0" | "1" |...| "9"

underline ::= "_"

c.constants

1.integer - rule:

noconst ::= "+" no | "-" no | no

no:= digit{no}

2.character

character:= 'letter'|'digit'

3.string

constchar:= "string"

string:=char{string}

char:=letter|digit

4. bool

boolean:= "bool"

bool:= false/true

SYNTAX

program ::= "int main()" cmpdstmt decllist

decllist ::= declaration ";" | declaration ";" decllist

declaration ::= type IDENTIFIER

type1 ::= "BOOL" | "CHAR" | "INT" | "FLOAT" | "DOUBLE" | "LONG LONG" | "UNSIGNED ..."

arraydecl ::= type "[" nr "]"

type ::= type1|arraydecl

cmpdstmt ::= "{" stmtlist "}"

stmtlist ::= stmt | stmt ";" stmtlist

stmt ::= simplstmt | structstmt

simplstmt ::= assignstmt | iostmt

assignstmt ::= IDENTIFIER "=" expression

expression ::= expression "+" term | term

term ::= term "*" factor | factor

factor ::= "(" expression ")" | IDENTIFIER | constant

iostmt ::= "READ" | "WRITE" "(" IDENTIFIER ")"

structstmt ::= cmpdstmt | ifstmt | whilestmt

ifstmt ::= "IF" condition "{" stmt "}" ["ELSE" "{" stmt "{"]

whilestmt ::= "WHILE" condition "{" stmt "}"

condition ::= "(" "(" expression RELATION expression ")" "and"/"or" ... ")"

RELATION ::= "<" | "<=" | "=" | "!=" | ">=" | ">"

TOKEN

break	try	catch	char	class	const	continue		
default	delete	auto	else	friend	for	float		
long	new	operator		private	protected		public	return
short	sizeof	static	this	typedef	enum	throw		
mutable		struct	case	register	switch	and	or	
namespace		static_cast		goto	not	xor	bool	do
double	int	unsigned		void	virtual	union	while	