labeled_point = ident , ":";	labeled_point(ident_terminal) = ident , ":";
goto_label = "GOTO" , ident;	goto_label("GOTO") = "GOTO" , ident;
program_name = ident;	program_name(ident_terminal) = ident;
value_type = "INTEGER16";	value_type("INTEGER16") = "INTEGER16";
	array_specify("[") = "[", unsigned_value_terminal , "]";
declaration_element = ident , ["[", unsigned_value , "]"];	declaration_element(ident_terminal) = ident , array_specify_optional;
	array_specify_optional("[") = array_specify;
	array_specify_optional(!"[") = ε;
other_declaration_ident = "," , declaration_element;	other_declaration_ident(",") = "," , declaration_element;
declaration = value_type , declaration_element , {other_declaration_ident};	declaration("INTEGER16") = value_type , declaration_element , other_declaration_ident__iteration;
	other_declaration_ident__iteration(",") = other_declaration_ident, other_declaration_ident__iteration;
	other_declaration_ident__iteration(!",") = ε;
index_action = "[" , expression , "]";	index_action("[") = "[" , expression , "]";
unary_operator = "NOT" | "-" | "+";	unary_operator("NOT") = "NOT";
	unary_operator("-") = "-";
	unary_operator("+") = "+";
unary_operation = unary_operator , expression;	unary_operation("NOT", "+", "-") = unary_operator , expression;
binary_operator = "AND" | "OR" | "==" | "!=" | "<=" | ">=" | "+" | "-" | "*" | "DIV" | "MOD";	binary_operator("AND") = "AND";
	binary_operator("OR") = "OR";
	binary_operator("==") = "==";
	binary_operator("!=") = "!=";
	binary_operator("<=") = "<=";
	binary_operator(">=") = ">=";
	binary_operator("+") = "+";
	binary_operator("-") = "-";
	binary_operator("*") = "*";
	binary_operator("DIV") = "DIV";

	binary_operator("MOD") = "MOD";
binary_action = binary_operator , expression;	binary_action("AND", "OR", "==", "!=", "<=", ">=", "+", "-", "*", "DIV", "MOD") = binary_operator , expression;
left_expression = group_expression | unary_operation | ident , [index_action] | value;	left_expression("(") = group_expression;
	left_expression("NOT") = unary_operation;
	left_expression(~~"NOT",~~"+", "-") = ([+1](unsigned_value_terminal) : value; ([+1]!(unsigned_value_terminal)) : unary_operation;
	left_expression(ident_terminal) = ident , index_action_optional;
	left_expression(unsigned_value_terminal~~, "+", "-"~~) = value;
	index_action_optional("[") = index_action;
	index_action_optional(!"[") = ε;
expression = left_expression , {binary_action};	expression("(","NOT", "+", "-", ident_terminal, unsigned_value_terminal) = left_expression , binary_action__iteration;
	binary_action__iteration("AND", "OR", "==", "!=", "<=", ">=", "+", "-", "*", "DIV", "MOD") = binary_action, binary_action__iteration;
	binary_action__iteration(!("AND", "OR", "==", "!=", "<=", ">=", "+", "-", "*", "DIV", "MOD")) = ε;
group_expression = "(" , expression , ")";	group_expression("(") = "(" , expression , ")";
bind_right_to_left = ident , [index_action] , ":=" , expression;	bind_right_to_left(ident_terminal) = ident , index_action_optional , ":=" , expression;
bind_left_to_right = expression , "=:" , ident , [index_action];	bind_left_to_right("(","NOT", "+", "-", ident_terminal, unsigned_value_terminal) = expression , "=:" , ident , index_action_optional;
if_expression = expression;	if_expression("(","NOT", "+", "-", ident_terminal, unsigned_value_terminal) = expression;
body_for_true = block_statements_in_while_and_if_body;	body_for_true("{") = block_statements_in_while_and_if_body;
false_cond_block = "ELSE" , cond_block;	***false_cond_block_without_else***("ELSE") = "ELSE" , "IF" , ***if_expression , body_for_true***;
body_for_false = "ELSE" , block_statements_in_while_and_if_body;	body_for_false("ELSE") = "ELSE" , block_statements_in_while_and_if_body;
cond_block = "IF" , if_expression , body_for_true , {false_cond_block} , [body_for_false];	cond_block("IF") = "IF" , if_expression , body_for_true , false_cond_block_without_else__iteration , body_for_false_optional;
	false_cond_block_without_else__iteration("ELSE") =

	([+1]"IF") : ***false_cond_block_without_else***, false_cond_block_without_else__iteration; ([+1]!("IF")) : ε;
	false_cond_block_without_else__iteration(!"ELSE") = ε;
	body_for_false_optional("ELSE") = body_for_false;
	body_for_false_optional(!"ELSE") = ε;
cycle_begin_expression = expression;	cycle_begin_expression("(","NOT", "+", "-", ident_terminal, unsigned_value_terminal) = expression;
cycle_end_expression = expression;	cycle_end_expression("(","NOT", "+", "-", ident_terminal, unsigned_value_terminal) = expression;
cycle_counter = ident;	cycle_counter(ident_terminal) = ident;
cycle_counter_rl_init = cycle_counter , ":=" , cycle_begin_expression;	cycle_counter_rl_init(ident_terminal) = cycle_counter , ":=" , cycle_begin_expression;
cycle_counter_lr_init = cycle_begin_expression , "=:" , cycle_counter;	cycle_counter_lr_init("(","NOT", "+", "-", ident_terminal, unsigned_value_terminal) = cycle_begin_expression , "=:" , cycle_counter;
cycle_counter_init = cycle_counter_rl_init | cycle_counter_lr_init;	cycle_counter_init(ident_terminal) = ([+1](":=")) : cycle_counter_rl_init; ([+1]!(":=")) : cycle_counter_lr_init;
	cycle_counter_init("(","NOT", "+", "-", ~~ident_terminal~~, unsigned_value_terminal) = cycle_counter_lr_init;
cycle_counter_last_value = cycle_end_expression;	cycle_counter_last_value("(","NOT", "+", "-", ident_terminal, unsigned_value_terminal) = cycle_end_expression;
cycle_body = "DO" , (statement | block_statements);	cycle_body("DO") = "DO" , statement__or__block_statements;
forto_cycle = "FOR" , cycle_counter_init , "TO" , cycle_counter_last_value , cycle_body;	forto_cycle("FOR") = "FOR" , cycle_counter_init , "TO" , cycle_counter_last_value , cycle_body;
statement_in_while_and_if_body = statement | "CONTINUE" | "BREAK";	statement_in_while_and_if_body(ident_terminal, "(", "NOT", unsigned_value_terminal, "+", "-", "IF", "FOR", "WHILE", "REPEAT", "GOTO", "GET", "PUT", ";") = statement;
	statement_in_while_and_if_body("CONTINUE") = "CONTINUE";
	statement_in_while_and_if_body("BREAK") = "BREAK";
block_statements_in_while_and_if_body = "{" , {statement_in_while_and_if_body} , "}";	block_statements_in_while_and_if_body("{") = "{" , statement_in_while_and_if_body__iteration , "}";
	statement_in_while_and_if_body__iteration(ident_terminal, "(", "NOT",

	unsigned_value_terminal, "+", "-", "IF", "FOR", "WHILE", "REPEAT", "GOTO", "GET", "PUT", ";", "CONTINUE", "BREAK") = statement_in_while_and_if_body, statement_in_while_and_if_body __iteration;
	statement_in_while_and_if_body __iteration(!(ident_terminal, "(", "NOT", unsigned_value_terminal, "+", "-", "IF", "FOR", "WHILE", "REPEAT", "GOTO", "GET", "PUT", ";", "CONTINUE", "BREAK")) = ε;
while_cycle_head_expression = expression;	while_cycle_head_expression("(","NOT", "+", "-", ident_terminal, unsigned_value_terminal) = expression;
while_cycle = "WHILE" , while_cycle_head_expression , block_statements_in_while_and_if_body;	while_cycle("WHILE") = "WHILE" , while_cycle_head_expression , block_statements_in_while_and_if_body;
repeat_until_cycle_cond = expression;	repeat_until_cycle_cond("(","NOT", "+", "-", ident_terminal, unsigned_value_terminal) = expression;
repeat_until_cycle = "REPEAT" , (statement | block_statements) , "UNTIL" , repeat_until_cycle_cond;	repeat_until_cycle("REPEAT") = "REPEAT" , statement__or__block_statements , "UNTIL" , repeat_until_cycle_cond;
	statement__or__block_statements(ident_terminal, "(", "NOT", unsigned_value_terminal, "+", "-", "IF", "FOR", "WHILE", "REPEAT", "GOTO", "GET", "PUT", ";") = statement;
	statement__or__block_statements("{") = block_statements;
input = "GET" , (ident , [index_action] | "(" , ident , [index_action] , ")");	input("GET") = "GET" , argument_for_input;
	argument_for_input(ident_terminal) = ident , index_action_optional;
	argument_for_input ("(") = "(" , ident , index_action_optional , ")";
output = "PUT" , expression;	output("PUT") = "PUT" , expression;
statement = bind_right_to_left | bind_left_to_right | cond_block | forto_cycle | while_cycle | repeat_until_cycle | labeled_point | goto_label | input | output | ";";	statement(ident_terminal) = _____ ([+1]":=") : bind_right_to_left; ([+1]":") : labeled_point; ([+1]!(":=", ":")) : bind_left_to_right;
	statement("(", "NOT", ~~ident_terminal~~, unsigned_value_terminal, "+", "-") = bind_left_to_right;
	statement("IF") = cond_block;
	statement("FOR") = forto_cycle;
	statement("WHILE") = while_cycle;

	statement("REPEAT") = repeat_until_cycle;
	~~statement(ident_terminal) = labeled_point;~~
	statement("GOTO") = goto_label;
	statement("GET") = input;
	statement("PUT") = output;
	statement(";") = ";";
block_statements = "{" , {statement} , "}";	block_statements("{") = "{" , statement__iteration, "}";
	statement__iteration(ident_terminal, "(", "NOT", unsigned_value_terminal, "+", "-", "IF", "FOR", "WHILE", "REPEAT", "GOTO", "GET", "PUT", ";") = statement, statement__iteration;
	statement__iteration(!(ident_terminal, "(", "NOT", unsigned_value_terminal, "+", "-", "IF", "FOR", "WHILE", "REPEAT", "GOTO", "GET", "PUT", ";")) = ε;
program = "NAME" , program_name , ";" , "BODY" , "DATA", [declaration] , ";" , {statement} , "END";	program("NAME") = "NAME" , program_name , ";" , "BODY" , "DATA", declaration_optional , ";" , statement__iteration , "END";
	declaration_optional("INTEGER16") = declaration;
	declaration_optional(!"INTEGER16") = ε;
digit = "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9";	
non_zero_digit = "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9";	
unsigned_value = (non_zero_digit , {digit}) | "0";	unsigned_value(unsigned_value_terminal) = unsigned_value_terminal;
value = [sign] , unsigned_value;	value(unsigned_value_terminal, "+", "-") = sign_optional , unsigned_value;
	sign_optional("+","-") = sign;

	sign_optional(!("+","-")) = ε;
letter_in_lower_case = "a" | "b" | "c" | "d" | "e" | "f" | "g" | "h" | "i" | "j" | "k" | "l" | "m" | "n" | "o" | "p" | "q" | "r" | "s" | "t" | "u" | "v" | "w" | "x" | "y" | "z";	
letter_in_upper_case = "A" | "B" | "C" | "D" | "E" | "F" | "G" | "H" | "I" | "J" | "K" | "L" | "M" | "N" | "O" | "P" | "Q" | "R" | "S" | "T" | "U" | "V" | "W" | "X" | "Y" | "Z";	
ident = "_" , letter_in_upper_case , letter_in_upper_case , letter_in_upper_case , letter_in_upper_case , letter_in_upper_case , letter_in_upper_case;	ident(ident_terminal) = ident_terminal;
sign = "+" | "-";	sign("+") = "+";
	sign("-") = "-";