Министерство науки и высшего образования Российской Федерации

ФГАОУ ВО «Казанский (Приволжский) федеральный университет»

ИНСТИТУТ ВЫЧИСЛИТЕЛЬНОЙ МАТЕМАТИКИ И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Кафедра информационных систем

**ОТЧЕТ**

По работе в течение семестра по предмету «Математическая лингвистика»

|  |  |
| --- | --- |
|  | Выполнил: студент группы 09-254    Багыманов Р.М. ——————————————  Проверил: старший преподаватель  Прокопьев Н.А. ———————————— |

Задание

Составить синтаксический и лексический анализатор на одном из выбранных языков (Python) с использованием библиотеки ply.

Первое задание представляло собой разработку лексического анализатора с использованием генератора lex. Основные этапы работы включали:

1. Определение языка – задание ключевых слов, таких как:

SORT, FOR, WHILE, ON, AS, TO

ASCENDING, DESCENDING, NOOPTIMIZE

AND, OR, FIELDS

1. Задание токенов для распознавания в исходном коде:

BOOLEAN – булевы значения (.t., .f.)

IDENTIFIER – идентификаторы переменных

NUMBER – целочисленные константы

STRING – строковые литералы в одинарных или двойных кавычках

COMMA – запятая

OPERATOR – операторы сравнения (==, =, !=, >=, <=, <, >)

LPAREN, RPAREN – левая и правая скобки

NEGATION – логическое отрицание (!)

ARITHMETIC\_OP – арифметические операторы (+, -, \*, /)

1. Дополнительная обработка:

Пропуск пробелов и переводов строк

Игнорирование комментариев (начинающихся с #)

Вывод лексической ошибки при обнаружении недопустимого символа с указанием строки и позиции.

Таким образом, лексический анализатор корректно разбивает входной поток символов на токены, обрабатывает служебные конструкции и сообщает об ошибках.

### Второе задание: реализация синтаксического анализатора

Цель задания заключалась в разработке синтаксического анализатора (парсера) с использованием библиотеки PLY (Python Lex-Yacc), который проверяет, соответствует ли входная строка заданной грамматике языка. Парсер обрабатывает команды, начинающиеся с ключевого слова SORT, и поддерживает различные опции, такие как:

* FOR, WHILE – управляющие конструкции;
* ON – указание поля для сортировки;
* NOOPTIMIZE – отключение оптимизации;
* TO – направление сортировки (ASCENDING, DESCENDING);
* MODIFIERS – дополнительные модификаторы;
* FIELDS – перечисление полей.

#### Синтаксический анализ и обработка ошибок

Парсер не только проверяет структуру входных данных, но и выявляет различные типы ошибок:

1. Синтаксические ошибки – возникают при нарушении правил грамматики:

Неожиданный токен (например, если после SORT идет недопустимый символ);

Неправильный порядок или структура выражения (например, пропущено ключевое слово ON перед полем сортировки);

Отсутствие обязательных элементов (например, незакрытая скобка или пропущенный оператор).

1. Семантические ошибки – связаны с логической и типовой корректностью выражений:

Попытка деления на ноль – арифметическая ошибка;

Операции между несовместимыми типами (например, сложение строки и числа);

Логические операции над не-булевыми значениями (например, AND между числами);

Сравнение объектов разных типов (например, "строка" == 123);

Логическое отрицание не-булевого выражения (например, !5 вместо !.t.).

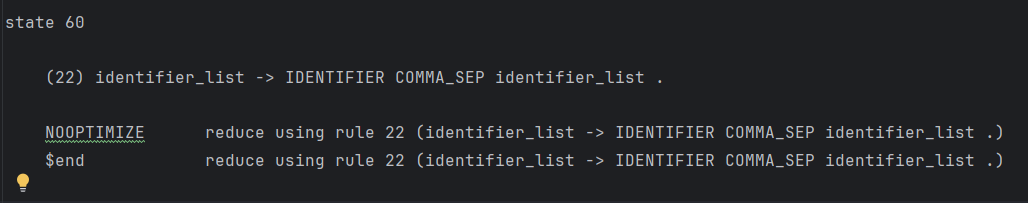
1. Общие ошибки – включают ситуации, которые нарушают целостность разбора:

Неожиданное завершение ввода (EOF) – если выражение оборвано (например, SORT ON name TO без указания направления);

Некорректное использование ключевых слов (например, повторение TO в одной команде).

Каждая ошибка сопровождается точным указанием строки и позиции в исходном коде, что упрощает отладку.

Третье задание заключалось в устранении предупреждений shift/reduce reduce/reduce, возникающих при работе синтаксического анализатора, когда он не понимает каким образом ему следует обработать команду: собрать больше «токенов» перед сверткой их в один или же свернуть уже собранные «токены». При реализации лексического анализатора таких предупреждений не возникло. Ниже приведен снимок экрана, на котором видно, что предупреждений не выявлено:



Далее приведен код реализованной программы:

import ply.lex as lex

import ply.yacc as yacc

# Словарь зарезервированных слов

RESERVED\_WORDS = {

'SORT': 'SORT',

'TO': 'TO',

'ON': 'ON',

'ASCENDING': 'ASCENDING',

'DESCENDING': 'DESCENDING',

'AS': 'AS',

'FOR': 'FOR',

'WHILE': 'WHILE',

'FIELDS': 'FIELDS',

'NOOPTIMIZE': 'NOOPTIMIZE',

'AND': 'AND',

'OR': 'OR',

}

# Список токенов

tokens = [

'BOOLEAN\_VAL',

'IDENTIFIER',

'NUMBER\_VAL',

'STRING\_VAL',

'COMPARISON\_OP',

'LEFT\_PAREN',

'RIGHT\_PAREN',

'NOT\_OP',

'MATH\_OP',

'SORT\_MODIFIER',

'COMMA\_SEP',

] + list(RESERVED\_WORDS.values())

# Определения токенов

t\_LEFT\_PAREN = r'\('

t\_RIGHT\_PAREN = r'\)'

t\_NOT\_OP = r'!'

t\_COMPARISON\_OP = r'==|=|!=|>=|<=|<|>'

t\_MATH\_OP = r'\+|\-|\\*|/'

t\_COMMA\_SEP = r','

t\_SORT\_MODIFIER = r'\/[ACDacd]+'

def t\_BOOLEAN\_VAL(t):

r'\.(T|F|t|f)\.(?![a-zA-Z0-9\_])'

t.value = True if t.value.upper() == '.T.' else False

return t

def t\_STRING\_VAL(t):

r'\'[^\']\*\'|"[^"]\*"'

t.value = t.value[1:-1] # Убираем кавычки

return t

def t\_IDENTIFIER(t):

r'[a-zA-Z\_][a-zA-Z0-9\_]\*'

t.type = RESERVED\_WORDS.get(t.value.upper(), 'IDENTIFIER')

return t

def t\_PATTERN(t):

r'[a-zA-Z\_][a-zA-Z0-9\_]\*[\\*?]|[\\*?][a-zA-Z0-9\_]\*'

t.value = t.value.upper()

return t

def t\_COMMENT(t):

r'\#.\*'

pass # Игнорируем комментарии

def t\_NUMBER\_VAL(t):

r'\d+'

t.value = int(t.value)

return t

# Игнорируемые символы

t\_ignore = ' \t'

def t\_newline(t):

r'\n+'

t.lexer.lineno += len(t.value)

def t\_error(t):

error\_message = f"Лексическая ошибка: Недопустимый символ '{t.value[0]}'"

error\_message += f"\nСтрока {t.lineno}, позиция {t.lexpos}"

raise SyntaxError(error\_message)

# Создание лексера

lexer = lex.lex()

class SemanticAnalysisError(Exception):

"""Исключение для семантических ошибок"""

def \_\_init\_\_(self, message, line\_num, position):

self.message = message

self.line\_num = line\_num

self.position = position

super().\_\_init\_\_(f"Строка {line\_num}, позиция {position}: {message}")

# Приоритет операторов

precedence = (

('left', 'OR'),

('left', 'AND'),

('right', 'NOT\_OP'),

('left', 'COMPARISON\_OP'),

('left', 'MATH\_OP'),

)

def p\_sort\_command(p):

'''sort\_command : SORT TO IDENTIFIER ON field\_list\_for\_sort optional\_modifiers optional\_for optional\_while optional\_fields optional\_nooptimize'''

p[0] = {

'command': 'SORT',

'destination': p[3],

'sort\_on': p[5],

'modifiers': p[6],

'for\_clause': p[7],

'while\_clause': p[8],

'fields\_clause': p[9],

'no\_optimize': p[10],

'line\_number': p.lineno(1),

'position': p.lexpos(1)

}

def p\_field\_list\_for\_sort(p):

'''field\_list\_for\_sort : single\_sort\_field

| single\_sort\_field COMMA\_SEP field\_list\_for\_sort'''

if len(p) == 2:

p[0] = [p[1]]

else:

p[0] = [p[1]] + p[3]

def p\_single\_sort\_field(p):

'''single\_sort\_field : IDENTIFIER

| IDENTIFIER ASCENDING

| IDENTIFIER DESCENDING

| IDENTIFIER AS IDENTIFIER

| IDENTIFIER ASCENDING AS IDENTIFIER

| IDENTIFIER DESCENDING AS IDENTIFIER'''

field\_info = {

'field\_name': p[1],

'sort\_direction': 'ASCENDING', # значение по умолчанию

'field\_alias': None,

'line\_number': p.lineno(1),

'position': p.lexpos(1)

}

if len(p) == 3:

field\_info['sort\_direction'] = p[2].upper()

elif len(p) == 4:

field\_info['field\_alias'] = p[3]

elif len(p) == 5:

field\_info['sort\_direction'] = p[2].upper()

field\_info['field\_alias'] = p[4]

p[0] = field\_info

def p\_optional\_modifiers(p):

'''optional\_modifiers : modifier\_list

| empty\_rule'''

p[0] = p[1] or []

def p\_modifier\_list(p):

'''modifier\_list : single\_modifier

| modifier\_list single\_modifier'''

p[0] = [p[1]] if len(p) == 2 else p[1] + [p[2]]

def p\_single\_modifier(p):

'''single\_modifier : SORT\_MODIFIER'''

p[0] = p[1][1:].upper() # Убираем первый символ '/'

def p\_optional\_for(p):

'''optional\_for : FOR logical\_expression

| empty\_rule'''

p[0] = p[2] if len(p) > 2 else None

def p\_optional\_while(p):

'''optional\_while : WHILE logical\_expression

| empty\_rule'''

p[0] = p[2] if len(p) > 2 else None

def p\_optional\_fields(p):

'''optional\_fields : FIELDS identifier\_list

| empty\_rule'''

p[0] = p[2] if len(p) > 2 else None

def p\_identifier\_list(p):

'''identifier\_list : IDENTIFIER

| IDENTIFIER COMMA\_SEP identifier\_list'''

if len(p) == 2:

p[0] = [p[1]]

else:

p[0] = [p[1]] + p[3]

def p\_optional\_nooptimize(p):

'''optional\_nooptimize : NOOPTIMIZE

| empty\_rule'''

p[0] = True if len(p) == 2 else False

def p\_simple\_value(p):

'''simple\_value : NUMBER\_VAL

| STRING\_VAL

| BOOLEAN\_VAL

| IDENTIFIER'''

current\_token = p.slice[1]

p[0] = {

'data\_type': current\_token.type.lower().replace('\_val', '').replace('\_', ''),

'data\_value': current\_token.value,

'line\_number': current\_token.lineno,

'position': current\_token.lexpos

}

def p\_logical\_expression(p):

'''logical\_expression : logical\_expression AND logical\_expression

| logical\_expression OR logical\_expression

| logical\_expression MATH\_OP logical\_expression

| logical\_expression COMPARISON\_OP logical\_expression

| NOT\_OP logical\_expression

| LEFT\_PAREN logical\_expression RIGHT\_PAREN

| simple\_value'''

if len(p) == 4:

if p[1] == '(':

p[0] = p[2] # Скобки просто группируют

else:

p[0] = {

'operator': p[2],

'left\_operand': p[1],

'right\_operand': p[3],

'line\_number': p.lineno(2),

'position': p.lexpos(2),

'expr\_type': 'binary\_operation'

}

elif len(p) == 3:

p[0] = {

'operator': '!',

'right\_operand': p[2],

'line\_number': p.lineno(1),

'position': p.lexpos(1),

'expr\_type': 'unary\_operation'

}

else:

p[0] = p[1]

def p\_empty\_rule(p):

'empty\_rule :'

p[0] = None

def p\_error(p):

if p:

if p.type == 'IDENTIFIER' and p.value == 'newtable':

error\_message = "Пропущено ключевое слово TO перед именем таблицы"

elif (p.type == 'IDENTIFIER' and

p.lexer.lexdata.rfind('TO', 0, p.lexpos) != -1 and

p.lexer.lexdata.rfind('ON', 0, p.lexpos) == -1):

previous\_tokens = p.lexer.lexdata[:p.lexpos].split()

if 'TO' in [token.upper() for token in previous\_tokens]:

error\_message = "Пропущено ключевое слово ON перед указанием полей сортировки"

error\_message += f"\nСтрока {p.lineno}, позиция {p.lexpos}"

raise SyntaxError(error\_message)

elif p.type in ('COMMA\_SEP', 'FOR', 'WHILE', 'FIELDS', 'ASCENDING', 'DESCENDING'):

error\_message = f"Незавершенное выражение после '{p.value}'"

else:

error\_message = f"Синтаксическая ошибка: Неожиданный токен '{p.value}'"

error\_message += f"\nСтрока {p.lineno}, позиция {p.lexpos}"

raise SyntaxError(error\_message)

else:

raise SyntaxError("Ошибка: неожиданный конец файла (возможно, незавершенная команда)")

def extract\_boolean\_value(expression):

"""Извлекает булево значение из выражения"""

if expression['data\_type'] == 'boolean':

return expression['data\_value']

if expression['data\_type'] == 'number':

return bool(expression['data\_value'])

if expression['data\_type'] in ('identifier', 'string'):

return None

return None

def perform\_semantic\_validation(expression):

"""Выполняет семантическую проверку выражения"""

if expression['expr\_type'] in ('binary\_operation', 'unary\_operation'):

operator = expression['operator']

# Проверяем левый операнд если есть

if 'left\_operand' in expression:

left\_data\_type = expression['left\_operand'].get('data\_type')

perform\_semantic\_validation(expression['left\_operand'])

# Проверяем правый операнд

if 'right\_operand' in expression:

right\_data\_type = expression['right\_operand'].get('data\_type')

perform\_semantic\_validation(expression['right\_operand'])

# Проверка арифметических операций

if operator in ('+', '-', '\*', '/'):

if left\_data\_type != 'number' or right\_data\_type != 'number':

raise SemanticAnalysisError(

f"Арифметическая операция {operator} требует числовых значений. Получены: {left\_data\_type} и {right\_data\_type}",

expression['line\_number'],

expression['position']

)

if operator == '/' and expression['right\_operand']['data\_value'] == 0:

raise SemanticAnalysisError(

"Деление на ноль недопустимо",

expression['right\_operand']['line\_number'],

expression['right\_operand']['position']

)

# Проверка логических операций

elif operator in ('AND', 'OR'):

left\_bool = extract\_boolean\_value(expression['left\_operand'])

right\_bool = extract\_boolean\_value(expression['right\_operand'])

if left\_bool is None or right\_bool is None:

raise SemanticAnalysisError(

f"Логическая операция {operator} требует булевых значений. Получены: {expression['left\_operand']['data\_type']} и {expression['right\_operand']['data\_type']}",

expression['line\_number'],

expression['position']

)

# Проверка операций сравнения

elif operator in ('==', '!=', '<', '>', '<=', '>='):

allowed\_type\_pairs = [

('number', 'number'),

('string', 'string'),

('identifier', 'number'),

('number', 'identifier'),

('identifier', 'identifier')

]

if (left\_data\_type, right\_data\_type) not in allowed\_type\_pairs:

raise SemanticAnalysisError(

f"Недопустимое сравнение {left\_data\_type} с {right\_data\_type}",

expression['line\_number'],

expression['position']

)

# Проверка унарного отрицания

elif operator == '!':

if right\_data\_type != 'boolean':

raise SemanticAnalysisError(

"Отрицание применимо только к булевым значениям",

expression['line\_number'],

expression['position']

)

# Создание парсера

parser = yacc.yacc(start='sort\_command')

def run\_code\_analysis(file\_path):

"""Анализирует код из файла"""

with open(file\_path, "r", encoding='utf-8') as file:

code\_lines = file.readlines()

print("=== Результаты диагностики кода ===")

for line\_index, code\_line in enumerate(code\_lines, 1):

cleaned\_line = code\_line.strip()

if not cleaned\_line:

continue

print(f"\nСтрока {line\_index}: {cleaned\_line}")

try:

lexer.input(cleaned\_line)

parsed\_result = parser.parse(lexer=lexer)

# Семантическая проверка

if parsed\_result.get('for\_clause'):

perform\_semantic\_validation(parsed\_result['for\_clause'])

if parsed\_result.get('while\_clause'):

perform\_semantic\_validation(parsed\_result['while\_clause'])

print("✓ Ошибок не обнаружено")

except (SyntaxError, SemanticAnalysisError) as error:

print(f"✗ {error}")

except Exception as unexpected\_error:

print(f"✗ Неизвестная ошибка: {str(unexpected\_error)}")

if \_\_name\_\_ == "\_\_main\_\_":

run\_code\_analysis("tests\_sort.txt")

Тестовые примеры:

SORT TO newtable ON field1

SORT TO results ON name ASCENDING, age DESCENDING

SORT TO temp ON price FOR price > 1000

SORT TO report ON department AS dept, salary DESCENDING AS sal FOR active = .T. NOOPTIMIZE

SORT TO ON field1

SORT TO table1 ON field1,

SORT TO table1 field1

SORT TO table2 ON field1 ASCENDINGING

SORT TO table3 ON field1 FOR price >

SORT TO table4 ON field1 FIELDS

SORT TO table5 ON field1 FOR AND 10

SORT TO table6 ON field1 FOR OR .T.

SORT TO table7 ON field1 FOR 10 AND "text"

SORT TO table8 ON field1 FOR !100

SORT TO table9 ON field1 FOR price / 0

SORT TO table10 ON FOR name = "X"

SORT TO table11 ON name FOR price + "abc"

SORT TO table12 ON salary ASCEND

SORT TO table13 ON name DESCEND

SORT TO table14 ON name AS

SORT TO table15 ON name ASCENDING AS

SORT TO table16 ON field1, FOR field1 = 1

SORT TO table17 ON field1 FOR = 1

SORT TO table18 field1

SORT TO table19 ON 123name

SORT TO table20 ON name AS 123alias

SORT TO table21 ON field1 WHILE

SORT TO table22 ON field1 FOR

SORT TO table23 ON name FOR name = "x" NOOPTIMIZ

SORT TO table24 ON name FIELDS , field2

SORT TO table25 ON name FIELDS name,

SORT TO table26 ON name FIELDS 123

SORT TO table27 ON name FIELDS name name2

SORT TO table28 ON name WHILE name >

SORT TO table29 ON name FOR (name = "test"

SORT TO table30 ON name FOR name = "a" AND

SORT TO table31 ON name FOR !.T

SORT TO table32 ON name FOR .T

SORT TO table33 ON name FOR 1 / 0

SORT TO table34 ON name FOR name =

SORT TO table35 ON name AS alias FIELDS

SORT TO table36 ON name FOR name = 'test"

SORT TO table37 ON name FOR name = 'test

SORT TO table38 ON name FOR name = "test

SORT TO table39 ON name MODIFIER

SORT TO table40 ON name /A

SORT TO table41 ON name /XZ

SORT TO table42 ON name /a /b /z

SORT TO table44 ON name FOR name + .T.

SORT TO table45 ON name FOR name = "abc" AND 123

SORT TO table46 ON name FOR name = 'abc' AND .F AND

SORT TO table47 ON name ASCENDING DESCENDING

SORT TO table48 ON name FOR

SORT TO table49 name field1

SORT TO table50 ON

SORT TO table51 ON field1 FOR price = .T

SORT TO table53 ON field1 FOR price AND

SORT TO table54 ON field1 FOR AND price

SORT TO table55 ON field1 FIELDS field1 FIELDS field2

SORT TO table56 ON field1 FOR OR

SORT TO table57 ON field1 FOR AND

SORT TO table58 ON field1 FOR field1 = 1 OR

SORT TO table59 ON field1 FOR !.F

SORT TO table60 ON field1 FOR !'text'

SORT TO table61 ON field1 FOR !(.F)

SORT TO table62 ON field1 FOR !(field1)

SORT TO table63 ON field1 FOR price / price - price /

SORT TO table64 ON field1 FOR price / 0 + 1

SORT TO table65 ON field1 FOR "text" + 1

SORT TO table66 ON field1 FOR .T. + 1

SORT TO table67 ON field1 ASCENDING DESCENDING

SORT TO table68 ON field1 AS alias AS alias2

SORT TO table69 ON field1, , field2

SORT TO table70 ON , field1

SORT TO table71 ON field1 AS "alias"

SORT TO table72 ON field1 ASCENDING AS

SORT TO table73 ON field1 DESCENDING AS

SORT TO table74 ON field1 FOR price >=

SORT TO table75 ON field1 FOR (price = 10

SORT TO table76 ON field1 FOR price = )

SORT TO table77 ON field1 FOR )

SORT TO table78 ON field1 FOR ((price > 10)

SORT TO table79 ON field1 FOR price + )

SORT TO table80 ON field1 FOR name AND price

SORT TO table81 ON field1 FOR price OR "abc"

SORT TO table82 ON field1 FOR .T. OR

SORT TO table83 ON field1 FOR field1 / 0

SORT TO table84 ON field1 FOR field1 > 0 AND

SORT TO table85 ON field1 FOR field1 + 5 = "text"

SORT TO table86 ON field1 FOR "abc" = "def"

SORT TO table87 ON field1 FOR "abc" > 100

SORT TO table90 ON field1 FOR field1 = .T OR

SORT TO table91 ON field1 FOR .F AND

SORT TO table92 ON field1 FOR 5 + 5 +

SORT TO table93 ON field1 FOR !(field1 = 1

SORT TO table94 ON field1 FOR field1 = 1)

SORT TO table95 ON field1 AS 1alias

SORT TO table96 ON field1 ASCENDING AS 123

SORT TO table97 ON field1 FOR price + name

SORT TO table98 ON field1 FOR price = 10 + "abc"

SORT TO table99 ON field1 FOR .T AND 123

Результаты теста:

Generating LALR tables

=== Результаты диагностики кода ===

Строка 1: SORT TO newtable ON field1

✓ Ошибок не обнаружено

Строка 2: SORT TO results ON name ASCENDING, age DESCENDING

✓ Ошибок не обнаружено

Строка 3: SORT TO temp ON price FOR price > 1000

✗ Неизвестная ошибка: 'expr\_type'

Строка 4: SORT TO report ON department AS dept, salary DESCENDING AS sal FOR active = .T. NOOPTIMIZE

✗ Неизвестная ошибка: 'expr\_type'

Строка 5: SORT TO ON field1

✗ Синтаксическая ошибка: Неожиданный токен 'ON'

Строка 1, позиция 8

Строка 6: SORT TO table1 ON field1,

✗ Ошибка: неожиданный конец файла (возможно, незавершенная команда)

Строка 7: SORT TO table1 field1

✗ Пропущено ключевое слово ON перед указанием полей сортировки

Строка 1, позиция 15

Строка 8: SORT TO table2 ON field1 ASCENDINGING

✗ Синтаксическая ошибка: Неожиданный токен 'ASCENDINGING'

Строка 1, позиция 25

Строка 9: SORT TO table3 ON field1 FOR price >

✗ Ошибка: неожиданный конец файла (возможно, незавершенная команда)

Строка 10: SORT TO table4 ON field1 FIELDS

✗ Ошибка: неожиданный конец файла (возможно, незавершенная команда)

Строка 11: SORT TO table5 ON field1 FOR AND 10

✗ Синтаксическая ошибка: Неожиданный токен 'AND'

Строка 1, позиция 29

Строка 12: SORT TO table6 ON field1 FOR OR .T.

✗ Синтаксическая ошибка: Неожиданный токен 'OR'

Строка 1, позиция 29

Строка 13: SORT TO table7 ON field1 FOR 10 AND "text"

✗ Неизвестная ошибка: 'expr\_type'

Строка 14: SORT TO table8 ON field1 FOR !100

✗ Неизвестная ошибка: 'expr\_type'

Строка 15: SORT TO table9 ON field1 FOR price / 0

✗ Неизвестная ошибка: 'expr\_type'

Строка 16: SORT TO table10 ON FOR name = "X"

✗ Незавершенное выражение после 'FOR'

Строка 1, позиция 19

Строка 17: SORT TO table11 ON name FOR price + "abc"

✗ Неизвестная ошибка: 'expr\_type'

Строка 18: SORT TO table12 ON salary ASCEND

✗ Синтаксическая ошибка: Неожиданный токен 'ASCEND'

Строка 1, позиция 26

Строка 19: SORT TO table13 ON name DESCEND

✗ Синтаксическая ошибка: Неожиданный токен 'DESCEND'

Строка 1, позиция 24

Строка 20: SORT TO table14 ON name AS

✗ Ошибка: неожиданный конец файла (возможно, незавершенная команда)

Строка 21: SORT TO table15 ON name ASCENDING AS

✗ Ошибка: неожиданный конец файла (возможно, незавершенная команда)

Строка 22: SORT TO table16 ON field1, FOR field1 = 1

✗ Незавершенное выражение после 'FOR'

Строка 1, позиция 27

Строка 23: SORT TO table17 ON field1 FOR = 1

✗ Синтаксическая ошибка: Неожиданный токен '='

Строка 1, позиция 30

Строка 24: SORT TO table18 field1

✗ Пропущено ключевое слово ON перед указанием полей сортировки

Строка 1, позиция 16

Строка 25: SORT TO table19 ON 123name

✗ Синтаксическая ошибка: Неожиданный токен '123'

Строка 1, позиция 19

Строка 26: SORT TO table20 ON name AS 123alias

✗ Синтаксическая ошибка: Неожиданный токен '123'

Строка 1, позиция 27

Строка 27: SORT TO table21 ON field1 WHILE

✗ Ошибка: неожиданный конец файла (возможно, незавершенная команда)

Строка 28: SORT TO table22 ON field1 FOR

✗ Ошибка: неожиданный конец файла (возможно, незавершенная команда)

Строка 29: SORT TO table23 ON name FOR name = "x" NOOPTIMIZ

✗ Синтаксическая ошибка: Неожиданный токен 'NOOPTIMIZ'

Строка 1, позиция 39

Строка 30: SORT TO table24 ON name FIELDS , field2

✗ Незавершенное выражение после ','

Строка 1, позиция 31

Строка 31: SORT TO table25 ON name FIELDS name,

✗ Ошибка: неожиданный конец файла (возможно, незавершенная команда)

Строка 32: SORT TO table26 ON name FIELDS 123

✗ Синтаксическая ошибка: Неожиданный токен '123'

Строка 1, позиция 31

Строка 33: SORT TO table27 ON name FIELDS name name2

✗ Синтаксическая ошибка: Неожиданный токен 'name2'

Строка 1, позиция 36

Строка 34: SORT TO table28 ON name WHILE name >

✗ Ошибка: неожиданный конец файла (возможно, незавершенная команда)

Строка 35: SORT TO table29 ON name FOR (name = "test"

✗ Ошибка: неожиданный конец файла (возможно, незавершенная команда)

Строка 36: SORT TO table30 ON name FOR name = "a" AND

✗ Ошибка: неожиданный конец файла (возможно, незавершенная команда)

Строка 37: SORT TO table31 ON name FOR !.T

✗ Лексическая ошибка: Недопустимый символ '.'

Строка 1, позиция 29

Строка 38: SORT TO table32 ON name FOR .T

✗ Лексическая ошибка: Недопустимый символ '.'

Строка 1, позиция 28

Строка 39: SORT TO table33 ON name FOR 1 / 0

✗ Неизвестная ошибка: 'expr\_type'

Строка 40: SORT TO table34 ON name FOR name =

✗ Ошибка: неожиданный конец файла (возможно, незавершенная команда)

Строка 41: SORT TO table35 ON name AS alias FIELDS

✗ Ошибка: неожиданный конец файла (возможно, незавершенная команда)

Строка 42: SORT TO table36 ON name FOR name = 'test"

✗ Лексическая ошибка: Недопустимый символ '''

Строка 1, позиция 35

Строка 43: SORT TO table37 ON name FOR name = 'test

✗ Лексическая ошибка: Недопустимый символ '''

Строка 1, позиция 35

Строка 44: SORT TO table38 ON name FOR name = "test

✗ Лексическая ошибка: Недопустимый символ '"'

Строка 1, позиция 35

Строка 45: SORT TO table39 ON name MODIFIER

✗ Синтаксическая ошибка: Неожиданный токен 'MODIFIER'

Строка 1, позиция 24

Строка 46: SORT TO table40 ON name /A

✓ Ошибок не обнаружено

Строка 47: SORT TO table41 ON name /XZ

✗ Синтаксическая ошибка: Неожиданный токен '/'

Строка 1, позиция 24

Строка 48: SORT TO table42 ON name /a /b /z

✗ Синтаксическая ошибка: Неожиданный токен '/'

Строка 1, позиция 27

Строка 49: SORT TO table44 ON name FOR name + .T.

✗ Неизвестная ошибка: 'expr\_type'

Строка 50: SORT TO table45 ON name FOR name = "abc" AND 123

✗ Неизвестная ошибка: 'expr\_type'

Строка 51: SORT TO table46 ON name FOR name = 'abc' AND .F AND

✗ Лексическая ошибка: Недопустимый символ '.'

Строка 1, позиция 45

Строка 52: SORT TO table47 ON name ASCENDING DESCENDING

✗ Незавершенное выражение после 'DESCENDING'

Строка 1, позиция 34

Строка 53: SORT TO table48 ON name FOR

✗ Ошибка: неожиданный конец файла (возможно, незавершенная команда)

Строка 54: SORT TO table49 name field1

✗ Пропущено ключевое слово ON перед указанием полей сортировки

Строка 1, позиция 16

Строка 55: SORT TO table50 ON

✗ Ошибка: неожиданный конец файла (возможно, незавершенная команда)

Строка 56: SORT TO table51 ON field1 FOR price = .T

✗ Лексическая ошибка: Недопустимый символ '.'

Строка 1, позиция 38

Строка 57: SORT TO table53 ON field1 FOR price AND

✗ Ошибка: неожиданный конец файла (возможно, незавершенная команда)

Строка 58: SORT TO table54 ON field1 FOR AND price

✗ Синтаксическая ошибка: Неожиданный токен 'AND'

Строка 1, позиция 30

Строка 59: SORT TO table55 ON field1 FIELDS field1 FIELDS field2

✗ Незавершенное выражение после 'FIELDS'

Строка 1, позиция 40

Строка 60: SORT TO table56 ON field1 FOR OR

✗ Синтаксическая ошибка: Неожиданный токен 'OR'

Строка 1, позиция 30

Строка 61: SORT TO table57 ON field1 FOR AND

✗ Синтаксическая ошибка: Неожиданный токен 'AND'

Строка 1, позиция 30

Строка 62: SORT TO table58 ON field1 FOR field1 = 1 OR

✗ Ошибка: неожиданный конец файла (возможно, незавершенная команда)

Строка 63: SORT TO table59 ON field1 FOR !.F

✗ Лексическая ошибка: Недопустимый символ '.'

Строка 1, позиция 31

Строка 64: SORT TO table60 ON field1 FOR !'text'

✗ Неизвестная ошибка: 'expr\_type'

Строка 65: SORT TO table61 ON field1 FOR !(.F)

✗ Лексическая ошибка: Недопустимый символ '.'

Строка 1, позиция 32

Строка 66: SORT TO table62 ON field1 FOR !(field1)

✗ Неизвестная ошибка: 'expr\_type'

Строка 67: SORT TO table63 ON field1 FOR price / price - price /

✗ Ошибка: неожиданный конец файла (возможно, незавершенная команда)

Строка 68: SORT TO table64 ON field1 FOR price / 0 + 1

✗ Неизвестная ошибка: 'expr\_type'

Строка 69: SORT TO table65 ON field1 FOR "text" + 1

✗ Неизвестная ошибка: 'expr\_type'

Строка 70: SORT TO table66 ON field1 FOR .T. + 1

✗ Неизвестная ошибка: 'expr\_type'

Строка 71: SORT TO table67 ON field1 ASCENDING DESCENDING

✗ Незавершенное выражение после 'DESCENDING'

Строка 1, позиция 36

Строка 72: SORT TO table68 ON field1 AS alias AS alias2

✗ Синтаксическая ошибка: Неожиданный токен 'AS'

Строка 1, позиция 35

Строка 73: SORT TO table69 ON field1, , field2

✗ Незавершенное выражение после ','

Строка 1, позиция 27

Строка 74: SORT TO table70 ON , field1

✗ Незавершенное выражение после ','

Строка 1, позиция 19

Строка 75: SORT TO table71 ON field1 AS "alias"

✗ Синтаксическая ошибка: Неожиданный токен 'alias'

Строка 1, позиция 29

Строка 76: SORT TO table72 ON field1 ASCENDING AS

✗ Ошибка: неожиданный конец файла (возможно, незавершенная команда)

Строка 77: SORT TO table73 ON field1 DESCENDING AS

✗ Ошибка: неожиданный конец файла (возможно, незавершенная команда)

Строка 78: SORT TO table74 ON field1 FOR price >=

✗ Ошибка: неожиданный конец файла (возможно, незавершенная команда)

Строка 79: SORT TO table75 ON field1 FOR (price = 10

✗ Ошибка: неожиданный конец файла (возможно, незавершенная команда)

Строка 80: SORT TO table76 ON field1 FOR price = )

✗ Синтаксическая ошибка: Неожиданный токен ')'

Строка 1, позиция 38

Строка 81: SORT TO table77 ON field1 FOR )

✗ Синтаксическая ошибка: Неожиданный токен ')'

Строка 1, позиция 30

Строка 82: SORT TO table78 ON field1 FOR ((price > 10)

✗ Ошибка: неожиданный конец файла (возможно, незавершенная команда)

Строка 83: SORT TO table79 ON field1 FOR price + )

✗ Синтаксическая ошибка: Неожиданный токен ')'

Строка 1, позиция 38

Строка 84: SORT TO table80 ON field1 FOR name AND price

✗ Неизвестная ошибка: 'expr\_type'

Строка 85: SORT TO table81 ON field1 FOR price OR "abc"

✗ Неизвестная ошибка: 'expr\_type'

Строка 86: SORT TO table82 ON field1 FOR .T. OR

✗ Ошибка: неожиданный конец файла (возможно, незавершенная команда)

Строка 87: SORT TO table83 ON field1 FOR field1 / 0

✗ Неизвестная ошибка: 'expr\_type'

Строка 88: SORT TO table84 ON field1 FOR field1 > 0 AND

✗ Ошибка: неожиданный конец файла (возможно, незавершенная команда)

Строка 89: SORT TO table85 ON field1 FOR field1 + 5 = "text"

✗ Неизвестная ошибка: 'expr\_type'

Строка 90: SORT TO table86 ON field1 FOR "abc" = "def"

✗ Неизвестная ошибка: 'expr\_type'

Строка 91: SORT TO table87 ON field1 FOR "abc" > 100

✗ Неизвестная ошибка: 'expr\_type'

Строка 92: SORT TO table90 ON field1 FOR field1 = .T OR

✗ Лексическая ошибка: Недопустимый символ '.'

Строка 1, позиция 39

Строка 93: SORT TO table91 ON field1 FOR .F AND

✗ Лексическая ошибка: Недопустимый символ '.'

Строка 1, позиция 30

Строка 94: SORT TO table92 ON field1 FOR 5 + 5 +

✗ Ошибка: неожиданный конец файла (возможно, незавершенная команда)

Строка 95: SORT TO table93 ON field1 FOR !(field1 = 1

✗ Ошибка: неожиданный конец файла (возможно, незавершенная команда)

Строка 96: SORT TO table94 ON field1 FOR field1 = 1)

✗ Синтаксическая ошибка: Неожиданный токен ')'

Строка 1, позиция 40

Строка 97: SORT TO table95 ON field1 AS 1alias

✗ Синтаксическая ошибка: Неожиданный токен '1'

Строка 1, позиция 29

Строка 98: SORT TO table96 ON field1 ASCENDING AS 123

✗ Синтаксическая ошибка: Неожиданный токен '123'

Строка 1, позиция 39

Строка 99: SORT TO table97 ON field1 FOR price + name

✗ Неизвестная ошибка: 'expr\_type'

Строка 100: SORT TO table98 ON field1 FOR price = 10 + "abc"

✗ Неизвестная ошибка: 'expr\_type'

Строка 101: SORT TO table99 ON field1 FOR .T AND 123

✗ Лексическая ошибка: Недопустимый символ '.'

Строка 1, позиция 30