НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені Ігоря Сікорського»
ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування та спеціалізованих комп'ютерних систем

РОЗРАХУНКОВО-ГРАФІЧНА РОБОТА

з дисципліни
«Основи проектування трансляторів»
Тема: «РОЗРОБКА СИНТАКСИЧНОГО АНАЛІЗАТОРА»

Виконав: студент IV курсу
групи КВ-84 ФПМ
Іванюк В.І.
Перевірив:

Київ

2021

Мета лабораторної роботи

Метою розрахунково-графічної роботи «Розробка синтаксичного аналізатора» є засвоєння теоретичного матеріалу та набуття практичного досвіду і практичних навичок розробки синтаксичних аналізаторів (парсерів).

Варіант 12

1. < signal - program > -- > < program>
2. < program > -- > PROCEDURE <procedure - identifier> <parameters - list>; <block>;
3. < block > -- > <declarations> BEGIN <statements-list> END
4. < declarations > -- > < label - declarations>
5. < label - declarations > -- > LABEL <unsigned-integer> <labels - list>; | < empty>
6. < labels - list > -- > , <unsigned - integer> <labels - list> | < empty>
7. < parameters - list > -- > (<variable - identifier> <identifiers - list>) | < empty>
8. < identifiers - list > -- > , <variable - identifier> <identifiers - list> | < empty>
9. < statements - list > -- > <statement> <statements-list> | < empty>
10. < statement > -- > <unsigned - integer> : <statement> | GOTO <unsigned - integer>; | RETURN; | ; | ($ <assembly - insert - file - identifier> $)
11. < variable - identifier > -- > < identifier>
12. < procedure - identifier > -- > < identifier>
13. < assembly - insert - file - identifier > -- > < identifier>
14. < identifier > -- > <letter> < string>
15. < string > -- > <letter><string> | <digit><string> | < empty>
16. < unsigned - integer > -- > <digit> < digits - string>
17. < digits - string > -- > <digit><digits - string> | <empty>
18. < digit > -- > 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
19. < letter > -- > A | B | C | D | ... | Z

Лістинг програми

OPT_lab1.cpp

```
#include "LexerGeneration.h"
#include "BinTree.h"

int main(int argc, char* argv[]) {
    if (argc != 2) {
        printf("Lexer: Invalid number of parameters.");
        return 1;
    }
    else {
        for (int i = 1; i < argc; i++) {
            printf("%s \n", argv[i]);
        }

    }

    FILE* test, * gen;
    char input[30];
    char output[30];
    char inputfile[] = "\\input.sig";
    char outputfile[] = "\\generated.txt";
    unsigned int x = -1, y = -222;

    cout << x << " " << y << endl;
    // For Visual Studio 2019
    strcpy_s(input, _countof(input), argv[1]);
    strcat_s(input, _countof(input), inputfile);
    strcpy_s(output, _countof(output), argv[1]);
    strcat_s(output, _countof(output), outputfile);
    errno_t err_test, err_gen;
    if ((err_test = fopen_s(&test, input, "r") != 0) || (err_gen = fopen_s(&gen,
output, "w") != 0)) {
        return 1;
    }

    // For g++
    /*strcpy(input, argv[1]);
    strcat(input, inputfile);
    strcpy(output, argv[1]);
    strcat(output, outputfile);
    if (((test = fopen(input, "r")) == NULL) || ((gen = fopen(output, "w")) ==
NULL)) {
        return 1;
    }*/



    else {
        if (!lexer(test, gen))
            parsing(gen);
        else
            fprintf(gen, "Pasrer cannt work : Lexer found errors\n");
        fclose(test);
        fclose(gen);
    }

    return 0;

}
```

LexerGeneration.h

```
#pragma once
#ifndef LEXERGENERATION_H
#define LEXERGENERATION_H

#include <iostream>
#include <string>

#include <cctype>
#include <algorithm>
#include <vector>
#include <typeinfo>
#include <cstring>
#include <stdio.h>
#include <map>
#include <vector>

using namespace std;

enum symbolCategories {
    whitespaces,
    digits,
    letters,
    unifier,
    separators,
    errors,
    tests
};

struct Token {
    Token() {};
    Token(int _row, int _column, int _id, string _value) {
        row = _row;
        column = _column;
        id = _id;
        value = _value;
    }
    int row, column, id;
    string value;
};



vector<Token> getVectorToken();
void printTables(FILE *gen);

/* File operations */
Token* dumpToken(FILE* generated, int row, int column, string token, Token*
tokenStruct, const int count);
void dumpLexError(FILE* generated, int row, int column, string undefinedToken);
void dumpTokError(FILE* generated, int row, int* column, char* err_symb, string
token, int count);

/* Struct operations */
Token* AddToken(int row, int column, int id, string token, Token* tokenStruct,
const  int count);
void showTokens();

/* Lexer operations */
bool lexer(FILE* test, FILE* gen);
int findID(string _token);
int symbolClassifier(char symbol);

#endif
```

LexerGeneration.cpp

```
#include "LexerGeneration.h"
vector<Token> token_vector;

vector<Token> getVectorToken() {
    return token_vector;
}

Token* dumpToken(FILE* generated, int row, int column, string token, Token*
tokenStruct, const int count) {
    tokenStruct = AddToken(row, column, findID(token), token, tokenStruct,
count);
    fprintf(generated, " %4d | %6d | %11d | %s\n", row, column, findID(token),
token.c_str());
    return tokenStruct;
}

void dumpTokError(FILE* generated, int row, int *column, char *err_symb, string
token, int count) {
    fprintf(generated, " Lexer : Error. Illegam symbol : ");
    for (int i = 0; i < count; i++) {
        fprintf(generated, "'%c'[%d, %d] ", err_symb[i], row, column[i]);
    }
    fprintf(generated, "in %s\n", token.c_str());
}

void dumpLexError(FILE* generated, int row, int column, string token) {
    fprintf(generated, " Lexer : Error. Illegam symbol : '%s'[%d, %d]\n",
token.c_str(), row, column);
}

Token* AddToken(int row, int column, int id, string token, Token* tokenStruct,
const  int count) {
    /*if (count == 0) {
        tokenStruct = new Token[count + 1];
    }
    else {
        Token* tmpToken = new Token[count + 1];
        for (int i = 0; i < count; i++) {
            tmpToken[i] = tokenStruct[i];
        }
        delete[] tokenStruct;

        tokenStruct = tmpToken;
    }
    tokenStruct[count].row = row;
    tokenStruct[count].column = column;
    tokenStruct[count].id = id;
    tokenStruct[count].value = token;*/

    Token tmp(row, column, id, token);
    token_vector.push_back(tmp);

    return tokenStruct;
}

void showTokens() {
    for (vector<Token>::iterator it = token_vector.begin(); it !=
token_vector.end(); it++ ) {
        cout << it->row << " | " << it->column << " | " << it->id << " | " << it-
>value << endl;
    }
}

int symbolClassifier(char symbol) {
    if (symbol == 32 || symbol == 13 || symbol == 10 || symbol == 9 || symbol ==
11 || symbol == 12) {
        return whitespaces;
    }
    else if (48 <= symbol && symbol <= 57) { //from '0' to '9'
        return digits;
    }
    else if ((65 <= symbol && symbol <= 90) || (97 <= symbol && symbol <= 122)) {
//from 'A' to 'Z' or from 'a' to 'z'
        return letters;
    }
    else if (symbol == 59 || symbol == 58 || symbol == 44 || symbol == 36 ||
symbol == 40 || symbol == 41) {
        return separators;
    }
    /*else if (symbol == 35) {
        return tests;
    }*/
    else if (symbol != -1) {
        return errors;
    }

}

bool lexer(FILE* test, FILE* gen) {
    fprintf(gen, " Line | Column | Ident token | Token\n-------------------------
------------------------\n");
    char symbol = fgetc(test);
    char buff[255], err_symbols[255];
    string lexem;
    int row = 1, column = 1, token_count = 0, buffLen, unifier_col, unifier_row,
err_count, err_column[255];
    bool err_flag = false;
    Token* token_struct = 0;
    bool error_check = false;

    while (symbol != -1) {
        switch (symbolClassifier(symbol)) {
        case whitespaces :
            while (symbolClassifier(symbol) == whitespaces) {
                column++;
                if (symbol == 10) {
                    row++;
                    column = 1;
                }
                symbol = fgetc(test);
            }
            break;
        case digits:
            buffLen = 0;
            err_count = 0;
            while (symbolClassifier(symbol) == digits || symbolClassifier(symbol)
== errors
                || symbolClassifier(symbol) == letters)
            {
                if (symbolClassifier(symbol) == errors ||
symbolClassifier(symbol) == letters) {
```

```
                    err_flag = true;
                    err_symbols[err_count] = symbol;
                    err_column[err_count] = column + buffLen;
                    err_count++;
                }
                buff[buffLen] = symbol;
                buffLen++;
                symbol = fgetc(test);
            }
            buff[buffLen] = '\0';
            lexem = string(buff);
            if (err_flag == false) {
                token_struct = dumpToken(gen, row, column, lexem, token_struct,
token_count);
                token_count++;
            }
            else {
                error_check = true;
                dumpTokError(gen, row, err_column, err_symbols, lexem,
err_count);
            }
            column += buffLen;
            err_flag = false;
            break;
        case letters:
            buffLen = 0;
            err_count = 0;
            while (symbolClassifier(symbol) == digits || symbolClassifier(symbol)
== errors
                || symbolClassifier(symbol) == letters)
            {
                if (symbolClassifier(symbol) == errors) {
                    err_flag = true;
                    err_symbols[err_count] = symbol;
                    err_column[err_count] = column + buffLen;
                    err_count++;
                }
                buff[buffLen] = symbol;
                buffLen++;
                symbol = fgetc(test);
            }
            buff[buffLen] = '\0';
            lexem = string(buff);
            if (err_flag == false) {
                token_struct = dumpToken(gen, row, column, lexem, token_struct,
token_count);
                token_count++;
            }
            else {
                error_check = true;
                dumpTokError(gen, row, err_column, err_symbols, lexem,
err_count);
            }
            column += buffLen;
            err_flag = false;
            break;
        case separators:
            if (symbol == 59) { // ;
                token_struct = dumpToken(gen, row, column, ";", token_struct,
token_count);
                token_count++;
                column++;
                symbol = fgetc(test);
                break;
            }
            else if (symbol == 58) { //:
                token_struct = dumpToken(gen, row, column, ":", token_struct,
token_count);
                token_count++;
                column++;
                symbol = fgetc(test);
                break;
            }
            else if (symbol == 44) { // ,
                token_struct = dumpToken(gen, row, column, ",", token_struct,
token_count);
                token_count++;
                column++;
                symbol = fgetc(test);
                break;
            }
            if (symbol == 40) { // (
                unifier_row = row;
                unifier_col = column;
                symbol = fgetc(test);
                column++;
                if (symbol == 42) { // *
                    while (true) {
                        if (symbol == 10) {
                            row++;
                            column = 0;
                        }
                        if (symbol == -1) {
                            fprintf(gen, " Lexer : Error. Unclosed commet [%d,
%d]\n", unifier_row, unifier_col);
                            error_check = true;
                            break;
                        }
                        if (symbol == 42) {
                            column++;
                            symbol = fgetc(test);
                            if (symbol == 41) {
                                column++;
                                break;
                            }
                        }
                        else {
                            symbol = fgetc(test);
                            column++;
                        }
                    }
                    symbol = fgetc(test);
                }
                else {
                    token_struct = dumpToken(gen, unifier_row, unifier_col, "(",
token_struct, token_count);
                    token_count++;
                    break;
                }
            }
            else if (symbol == 41) { // )
                token_struct = dumpToken(gen, row, column, ")", token_struct,
token_count);
                token_count++;
                column++;
                symbol = fgetc(test);
                break;
            }
            else if (symbol == 36) { // $
                token_struct = dumpToken(gen, row, column, "$", token_struct,
token_count);
                token_count++;
                column++;
                symbol = fgetc(test);
                break;
            }
            break;
        case errors:
            error_check = true;
            buffLen = 0;
            err_count = 0;
            while (symbolClassifier(symbol) == digits || symbolClassifier(symbol)
== errors
                || symbolClassifier(symbol) == letters)
            {
                if (symbolClassifier(symbol) == errors) {
                    err_flag = true;
                    err_symbols[err_count] = symbol;
                    err_column[err_count] = column + buffLen;
                    err_count++;
                }
                buff[buffLen] = symbol;
                buffLen++;
                symbol = fgetc(test);
            }
            buff[buffLen] = '\0';
            lexem = string(buff);
            if (buffLen > 1)
                dumpTokError(gen, row, err_column, err_symbols, lexem,
err_count);
            else
                dumpLexError(gen, row, column, lexem);
            column += buffLen;
            err_flag = false;
            break;
        }
    }
    showTokens();
    printTables(gen);
    return error_check;
}
```

LexerTables.cpp

```
#include "LexerGeneration.h"

int ident_count = 1001;
int const_count = 501;
int test_count = 5001;

map <string, int> kwrd = {
  {"PROCEDURE", 401},
  {"BEGIN", 402},
  {"END", 403},
  {"LABEL", 404},
  {"GOTO", 405},
  {"RETURN", 406}
};

map <string, int> sep = {
    {";", 59},
    {",", 44},
    {":", 58},
    {"(", 40},
    {")", 41},
    {"$", 36}
};

map <string, int> ident;
map <string, int> _const;
map <string, int> test;
```

```cpp
int findID(string _token) {
    Token token;
    token.value = _token;
    map<string, int>::iterator iter;

    if (symbolClassifier(token.value[0]) == letters) {
        if (kwrd.count(token.value) == 1) {
            iter = kwrd.find(token.value);
            token.id = iter->second;
        }
        else if (ident.count(token.value) == 0) {
            ident.insert(make_pair(token.value, ident_count));
            token.id = ident_count;
            ident_count++;
        }
        else {
            iter = ident.find(token.value);
            token.id = iter->second;
        }
    }
    else if (symbolClassifier(token.value[0]) == digits) {
        if (_const.count(token.value) == 0) {
            _const.insert(pair<string, int>(token.value, const_count));
            token.id = const_count;
            const_count++;
        }
        else {
            iter = _const.find(token.value);
            token.id = iter->second;
        }
    }
    else if (symbolClassifier(token.value[0]) == tests) {
        if (test.count(token.value) == 0) {
            test.insert(pair<string, int>(token.value, test_count));
            token.id = test_count;
            test_count++;
        }
        else {
            iter = test.find(token.value);
            token.id = iter->second;
        }
    }
    else if (sep.count(token.value) == 1) {
        iter = sep.find(token.value);
        token.id = iter->second;
    }

    return token.id;
}
```

```
void printTables(FILE* gen) {

    fprintf(gen, "\nIdentifier table\n");
    for (const auto& it : ident) {
        cout << it.first << " " << it.second << endl;
        fprintf(gen, "%s %d\n", it.first.c_str(), it.second);
    }

    fprintf(gen, "\nConstant table\n");
    for(const auto& it : _const){
        cout << it.first << " " << it.second << endl;
        fprintf(gen, "%s %d\n", it.first.c_str(), it.second);
    }
}
```

BinTree.h

```
#pragma once
#ifndef BIN_TREE_H
#define BIN_TREE_H

#include "LexerGeneration.h"

struct Nodes {
    Nodes() {};
    Nodes(int _lexem_code, string _lexem_name, Nodes* _parent) {
        lexem_code = _lexem_code;
        lexem_name = _lexem_name;
        parent = _parent;
    }
    int lexem_code;
    string lexem_name;
    Nodes* parent;
    vector<Nodes> childNodes;
};


enum errorCode {
    key_word_not_found,
    delimiter_not_found,
    ident_not_found,
    const_not_found,
    wrong_delimiter,
    wrong_key_word,
    no_equal_rows,
    no_statement
};

void parsing(FILE* generated);
void createRoot(int _lexem_code, string _lexem_name);
void addChild(int _lexem_code, string _lexem_name);
void gotoChild(string _lexem_name);
void setCurrentNode(Nodes* child);
void gotoLastChild();
Nodes* getCurrentNode();
bool gotoParent();

Token getToken();
Token checkKeyToken(Token checkToken, string keyToken);
Token delimiters(Token prev_token, Token current_token, int token_id);

void program(Token token);
Token identifier(Token token);
Token procedureIdentifier(Token prev_token, Token current_token);
Token parametersList(Token prev_token, Token current_token);
Token variableIdentifier(Token prev_token, Token current_token);
Token identifierList(Token prev_token, Token current_token);
Token blok(Token current_token);
Token declaration(Token current_token);
Token labelDeclaration(Token current_token);
Token unsignedInteger(Token prev_token, Token current_token);
Token labelList(Token prev_token, Token current_token);
Token statementList(Token prev_token, Token current_token);
Token statement(Token prev_token, Token current_token);
Token assemblyInsertFileIdentifier(Token prev_token, Token current_token);

void errorOutput(int error_code, Token error_token = Token(), string token = "");
void printTree(FILE* gen, Nodes _tree, int _depth);
void printTree(FILE* gen);
#endif // !BIN_TREE_H
```

BinTree.cpp

```
#include "BinTree.h"
#include "LexerGeneration.h"

Nodes root;
Nodes* currentNode = &root;

void createRoot(int _lexem_code, string _lexem_name) {
    root.lexem_code = _lexem_code;
    root.lexem_name = _lexem_name;
    root.parent = NULL;
}

void addChild(int lexem_code, string lexem_name) {
    Nodes tmp(lexem_code, lexem_name, currentNode);
    currentNode->childNodes.push_back(tmp);
}
```

```
Nodes* getCurrentNode() {
    return currentNode;
}

void setCurrentNode(Nodes* newCurrentNode) {
    currentNode = newCurrentNode;
}

void gotoChild(int index) {
    setCurrentNode(&currentNode->childNodes[index]);
}

void gotoChild(string _lexem_name) {
    for (int i = 0; i < (int)currentNode->childNodes.size(); i++) {
        if (currentNode->childNodes[i].lexem_name == _lexem_name) {
            gotoChild(i);
            return;
        }
    }
}

void gotoLastChild() {
    setCurrentNode(&currentNode->childNodes.back());
}

bool gotoParent() {
    if (currentNode == &root) return false;
    currentNode = currentNode->parent;
    return true;
}

void printTree(FILE* gen, Nodes tree, int _depth) {
    if (tree.lexem_code == -1) {
        cout << tree.lexem_name << endl;
        fprintf(gen, "%s\n", tree.lexem_name.c_str());
    }
    else {
        cout << tree.lexem_code << "  " << tree.lexem_name << endl;
        fprintf(gen, "%d %s\n", tree.lexem_code, tree.lexem_name.c_str());
    }

    if (!tree.childNodes.empty()) {
        for (int i = 0; i < (int)tree.childNodes.size(); i++) {
            for (int i = 0; i <= _depth; i++) {
                cout << "..";
                fprintf(gen, "..");
            }
            printTree(gen, tree.childNodes[i], _depth + 1);
        }
    }
}

void printTree(FILE* gen) {
    cout << endl << "Parse tree" << endl;
    fprintf(gen, "\nParse tree\n");
    printTree(gen, root, 0);
}
```

SyntaxAnalyzer.cpp

```
#include "LexerGeneration.h"
#include "BinTree.h"

vector<Token> vector_lexem;
FILE* gen;

Token getToken() {
    Token tmp = *vector_lexem.begin();
    vector_lexem.erase(vector_lexem.begin());
    return tmp;
}

Token checkKeyToken(Token checkToken, string keyToken) {
    if (checkToken.id == findID(keyToken)) {
        addChild(checkToken.id, checkToken.value);
    }
    else {
        errorOutput(key_word_not_found, checkToken, keyToken);
    }
    return getToken();
}

void errorOutput(int error_code, Token error_token, string token) {
    printTree(gen);
    switch (error_code) {
    case key_word_not_found:
        printf("Parser : Error. Key word \'%s\'[%d, %d] not found.\n",
token.c_str(), error_token.row, error_token.column);
        fprintf(gen, "Parser : Error. Key word \'%s\'[%d, %d] not found.\n",
token.c_str(), error_token.row, error_token.column);
        break;
    case delimiter_not_found:
        printf("Parser : Error. Delimiter \'%s\'[%d, %d] not found.\n",
token.c_str(), error_token.row, error_token.column);
        fprintf(gen, "Parser : Error. Delimiter \'%s\'[%d, %d] not found.\n",
token.c_str(), error_token.row, error_token.column);
        break;
    case ident_not_found:
        printf("Parser : Error [%d, %d]. Identifier not found.\n",
error_token.row, error_token.column);
```

```
        fprintf(gen, "Parser : Error [%d, %d]. Identifier not found.\n", 
error_token.row, error_token.column);
        break;
    case const_not_found:
        printf("Parser : Error [%d, %d]. Unsigned integer not found.\n", 
error_token.row, error_token.column);
        fprintf(gen, "Parser : Error [%d, %d]. Unsigned integer not found.\n", 
error_token.row, error_token.column);
        break;
    case wrong_delimiter:
        printf("Parser : Error [%d, %d]. Wrong delimiter.\n", error_token.row, 
error_token.column);
        fprintf(gen, "Parser : Error [%d, %d]. Wrong delimiter.\n", 
error_token.row, error_token.column);
        break;
    case wrong_key_word:
        printf("Parser : Error [%d, %d]. Wrong key word.\n", error_token.row, 
error_token.column);
        fprintf(gen, "Parser : Error [%d, %d]. Wrong key word.\n", 
error_token.row, error_token.column);
        break;
    case no_equal_rows:
        printf("Parser : Error [%d, %d]. Tokens must be on the same line.\n", 
error_token.row, error_token.column);
        fprintf(gen, "Parser : Error [%d, %d]. Tokens must be on the same 
line.\n", error_token.row, error_token.column);
        break;
    case no_statement:
        printf("Parser : Error [%d, %d]. After the mark should be statement.\n", 
error_token.row, error_token.column);
        fprintf(gen, "Parser : Error [%d, %d]. After the mark should be 
statement.\n", error_token.row, error_token.column);
        break;
    }
    exit(error_code);
}

void parsing(FILE* generated) {
    gen = generated;
    vector_lexem = getVectorToken();
    if (vector_lexem.size() == 0) {
        fprintf(generated, " File is empty");
    }
    createRoot(-1, "<signal-program>");
    program(getToken());
    printTree(gen);
}

void program(Token token) {
    addChild(-1, "<program>");
    gotoLastChild();

    Token checkKeyWord = checkKeyToken(token, "PROCEDURE");
    Nodes* currentNode = getCurrentNode();
    Token next_token = procedureIdentifier(token, checkKeyWord);
    setCurrentNode(currentNode);

    next_token = parametersList(checkKeyWord, next_token);
    setCurrentNode(currentNode);
    next_token = delimiters(checkKeyWord, next_token, 59);
    next_token = blok(next_token);

}

Token procedureIdentifier(Token prev_token, Token current_token) {
    if (prev_token.row == current_token.row) {
        addChild(-1, "<procedure-identifier>");
        gotoLastChild();

        return identifier(current_token);
    }
    else {
        errorOutput(no_equal_rows, current_token);
    }
}

Token identifier(Token token) {
    if (token.id > 1000) {
        addChild(-1, "<identifier>");
        gotoChild("<identifier>");
        addChild(token.id, token.value);
    }
    else {
        errorOutput(ident_not_found, token);
    }
    return getToken();
}

Token delimiters(Token prev_token, Token current_token, int token_id) {
    if (current_token.id > 0 && current_token.id < 255) {
        if (current_token.id == token_id) {
            if (prev_token.row == current_token.row) {
                addChild(current_token.id, current_token.value);
            }
            else {
                errorOutput(no_equal_rows, current_token);
            }
        }
        else {
            errorOutput(wrong_delimiter, current_token);
        }
    }
    else {
        char buff[2];
        buff[0] = (char)token_id;
        buff[1] = '\0';
        string token = string(buff);
        errorOutput(delimiter_not_found, current_token, token);
    }
    if (!vector_lexem.empty()) {
        current_token = getToken();
    }
    return current_token;
}

Token variableIdentifier(Token prev_token, Token current_token) {
    if (prev_token.row == current_token.row) {
        addChild(-1, "<variable-identifier>");
        gotoLastChild();

        return identifier(current_token);
    }
    else {
        errorOutput(no_equal_rows, current_token);
    }
}

Token identifierList(Token prev_token, Token current_token) {
    bool isIdentifierList = false;
    addChild(-1, "<identifier-list>");
    gotoLastChild();
    Token next_token;
    Nodes* currentNode = getCurrentNode();
    if (current_token.id != 41) {
        isIdentifierList = true;
        next_token = delimiters(prev_token, current_token, 44);
        if (next_token.id > 1000) {
            next_token = variableIdentifier(current_token, next_token);
            setCurrentNode(currentNode);
            next_token = identifierList(current_token, next_token);
            return next_token;
        }
        else {
            errorOutput(ident_not_found, next_token);
        }
    }


    if (!isIdentifierList) {
        addChild(-1, "<empty>");
    }

    return current_token;
}

Token parametersList(Token prev_token, Token current_token) {
    bool isParameterList = false;
    addChild(-1, "<parameters-list>");
    gotoLastChild();
    Token next_token;
    Nodes* currentNode = getCurrentNode();
    if (current_token.id != 59) {
        isParameterList = true;
        next_token = delimiters(prev_token, current_token, 40);
        current_token = next_token;
        next_token = variableIdentifier(current_token, next_token);
        setCurrentNode(currentNode);
        next_token = identifierList(current_token, next_token);
        setCurrentNode(currentNode);
        next_token = delimiters(prev_token, next_token, 41);

        current_token = next_token;
    }

    if (!isParameterList) {
        addChild(-1, "<empty>");
    }

    return current_token;
}


Token blok(Token current_token) {
    addChild(-1, "<block>");
    gotoLastChild();
    Token next_token;
    Nodes* currentNode = getCurrentNode();
    current_token = declaration(current_token);
    setCurrentNode(currentNode);
    next_token = checkKeyToken(current_token, "BEGIN");
    setCurrentNode(currentNode);
    current_token = statementList(current_token, next_token);
    setCurrentNode(currentNode);
    next_token = checkKeyToken(current_token, "END");
    gotoParent();
    next_token = delimiters(current_token, next_token, 59);
    return next_token;
}
```

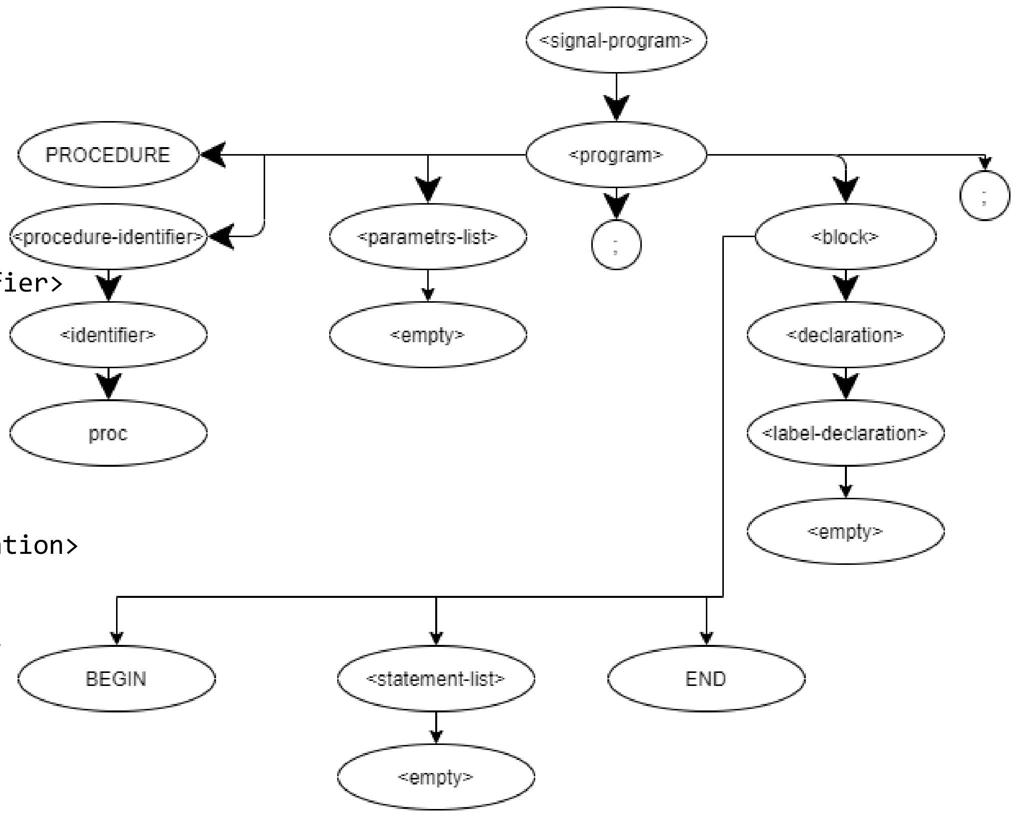
```
Token declaration(Token current_token) {
    addChild(-1, "<declaration>");
    gotoLastChild();
    current_token = labelDeclaration(current_token);
    return current_token;
}

Token labelDeclaration(Token current_token) {
    bool isLabelDeclaration = false;
    addChild(-1, "<label-declaration>");
    gotoLastChild();

    Nodes* currentNode = getCurrentNode();

    if (current_token.id != 402) {
        isLabelDeclaration = true;
        Token next_token = checkKeyToken(current_token, "LABEL");
        Nodes* currentNode = getCurrentNode();
        next_token = unsignedInteger(current_token, next_token);
        setCurrentNode(currentNode);
        next_token = labelList(current_token, next_token);
        setCurrentNode(currentNode);
        next_token = delimiters(current_token, next_token, 59);
        current_token = next_token;
    }


    if (!isLabelDeclaration) {
        addChild(-1, "<empty>");
    }

    return current_token;
}

Token unsignedInteger(Token prev_token, Token current_token) {
    if (prev_token.row == current_token.row) {
        if (current_token.id > 500 && current_token.id <= 1000) {
            addChild(-1, "<unsigned-integer>");
            gotoChild("<unsigned-integer>");
            addChild(current_token.id, current_token.value);
        }
        else {
            errorOutput(const_not_found, current_token);
        }
    }
    else {
        errorOutput(no_equal_rows, current_token);
    }
    return getToken();
}
```

```
Token labelList(Token prev_token, Token current_token) {
    bool isLabelList = false;
    addChild(-1, "<label-list>");
    gotoLastChild();
    Token next_token;
    Nodes* currentNode = getCurrentNode();
    if (current_token.id != 59) {
        isLabelList = true;
        next_token = delimiters(prev_token, current_token, 44);
        next_token = unsignedInteger(current_token, next_token);
        setCurrentNode(currentNode);
        next_token = labelList(current_token, next_token);
        return next_token;
    }


    if (!isLabelList) {
        addChild(-1, "<empty>");
    }

    return current_token;
}

Token statement(Token prev_token, Token current_token) {
    addChild(-1, "<statement>");
    gotoLastChild();
    Nodes* currentNode = getCurrentNode();
    Token next_token;
    if (current_token.id > 500 && current_token.id <= 1000) {
        next_token = unsignedInteger(prev_token, current_token);
        current_token = delimiters(prev_token, next_token, 58);
        current_token = statement(next_token, current_token);
    }
    else if (current_token.id > 400 && current_token.id <= 500) {
        if (current_token.id == 405) {
            prev_token = current_token;
            current_token = checkKeyToken(current_token, "GOTO");
            current_token = unsignedInteger(prev_token, current_token);
            setCurrentNode(currentNode);
            current_token = delimiters(prev_token, current_token, 59);
        }
        else if (current_token.id == 406) {
            prev_token = current_token;
            current_token = checkKeyToken(current_token, "RETURN");
            current_token = delimiters(prev_token, current_token, 59);
        }
        else {
            errorOutput(wrong_key_word, current_token);
        }
    }
    else if (current_token.id > 0 && current_token.id < 255) {
        if (current_token.id == 59) {
            current_token = delimiters(prev_token, current_token, 59);
        }
        else if (current_token.id == 40) {
            next_token = delimiters(prev_token, current_token, 40);
            current_token = delimiters(prev_token, next_token, 36);
            next_token = assemblyInsertFileIdentifier(next_token, current_token);
            setCurrentNode(currentNode);
            current_token = delimiters(current_token, next_token, 36);
            next_token = delimiters(next_token, current_token, 41);
            return next_token;
        }
        else {
            errorOutput(wrong_delimiter, current_token);
        }
    }
    else {
        errorOutput(no_statement, current_token);
    }

    return current_token;
}

Token statementList(Token prev_token, Token current_token) {
    bool isStatementList = false;
    addChild(-1, "<statement-list>");
    gotoLastChild();
    Nodes* currentNode = getCurrentNode();
    Token next_token;
    if (current_token.id != 403) {
        isStatementList = true;
        current_token = statement(prev_token, current_token);
        setCurrentNode(currentNode);
        current_token = statementList(prev_token, current_token);
    }


    if (!isStatementList) {
        addChild(-1, "<empty>");
    }

    return current_token;
}

Token assemblyInsertFileIdentifier(Token prev_token, Token current_token) {
    addChild(-1, "<assembly-insert-file-identifier>");
    gotoLastChild();
    if (prev_token.row == current_token.row) {
        current_token = identifier(current_token);
    }
    else {
        errorOutput(no_equal_rows, current_token);
    }


    return current_token;
}
```

Результати тестування

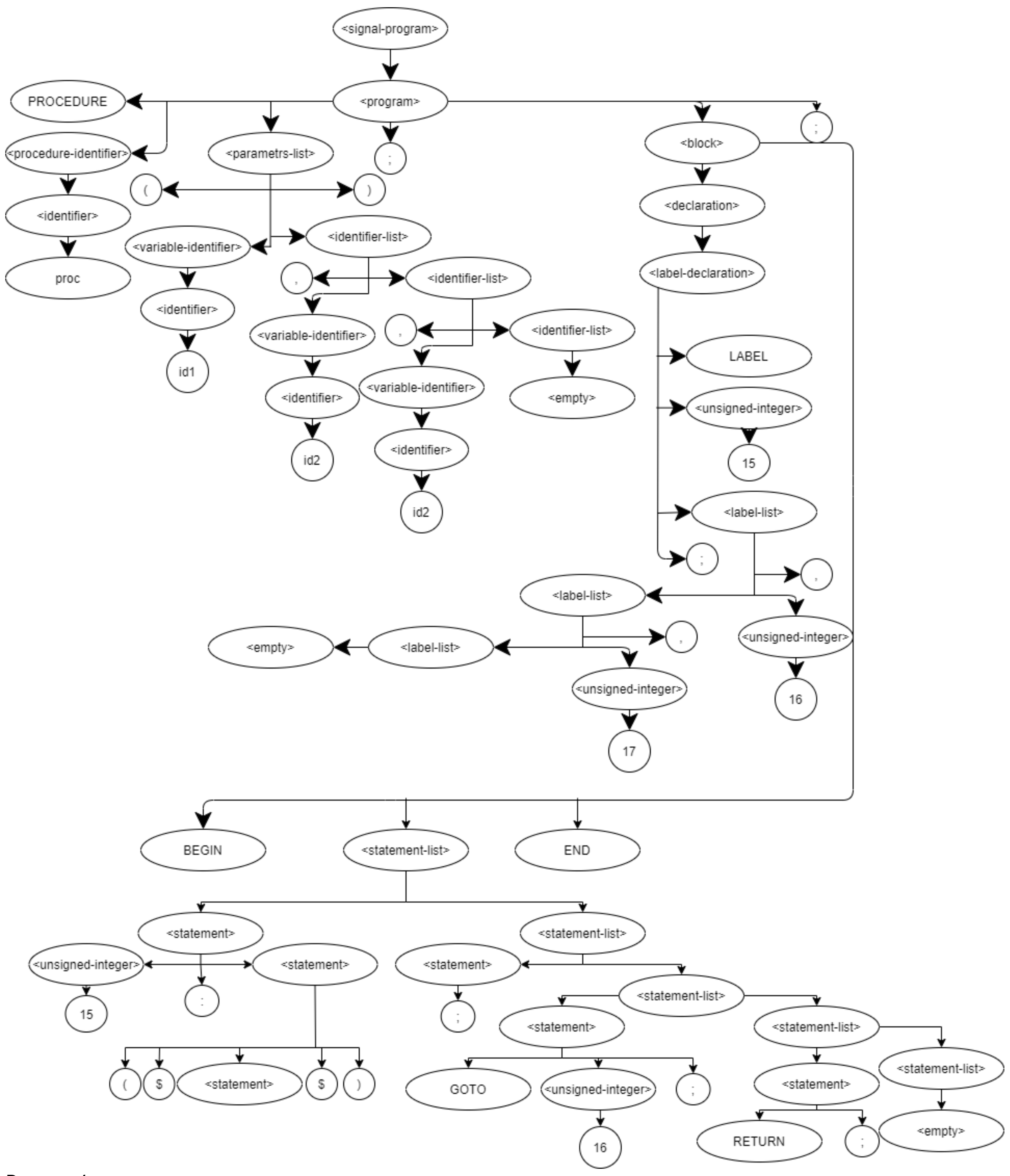
Test01:

```
Line | Column | Ident token | Token
-------------------------------------------------
    1 |      1 |         401 | PROCEDURE
    1 |     11 |        1001 | proc
    1 |     15 |          59 | ;
    2 |      1 |         402 | BEGIN
    3 |      1 |         403 | END
    3 |      4 |          59 | ;

Identifier table
proc 1001


Parse tree
<signal-program>
..<program>
....401 PROCEDURE
....<procedure-identifier>
......<identifier>
........1001 proc
....<parameters-list>
......<empty>
....59 ;
....<block>
......<declaration>
........<label-declaration>
..........<empty>
......402 BEGIN
......<statement-list>
........<empty>
......403 END
....59 ;
```



Test02:

```
Line | Column | Ident token | Token
-------------------------------------------------
    1 |      1 |         401 | PROCEDURE
    1 |     11 |        1001 | proc
    1 |     15 |          40 | (
    1 |     16 |        1002 | id1
    1 |     19 |          44 | ,
    1 |     21 |        1003 | id2
    1 |      24 |          44 | ,
    1 |      26 |        1004 | id3
    1 |      29 |          41 | )
    1 |      30 |          59 | ;
    2 |       1 |         404 | LABEL
    2 |       7 |         501 | 15
    2 |       9 |          44 | ,
    2 |      11 |         502 | 16
    2 |      13 |          44 | ,
    2 |      15 |         503 | 17
    2 |      17 |          59 | ;
    3 |       1 |         402 | BEGIN
    3 |       7 |         501 | 15
    3 |      10 |          58 | :
    3 |      12 |          40 | (
    3 |      13 |          36 | $
    3 |      15 |        1005 | asmFile
    3 |      23 |          36 | $
    3 |      24 |          41 | )
    3 |      25 |          59 | ;
    4 |       1 |         405 | GOTO
    4 |       6 |         502 | 16
    4 |       8 |          59 | ;
    5 |       1 |         406 | RETURN
    5 |       7 |          59 | ;
    6 |       1 |         403 | END
    6 |       4 |          59 | ;

Identifier table
asmFile 1005
id1 1002
id2 1003
id3 1004
proc 1001

Constant table
15 501
16 502
17 503
```

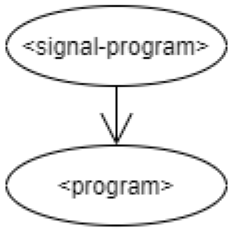


```
Parse tree
<signal-program>
..<program>
....401 PROCEDURE
....<procedure-identifier>
......<identifier>
........1001 proc
....<parameters-list>
......40 (
......<variable-identifier>
........<identifier>
..........1002 id1
......<identifier-list>
........44 ,
........<variable-identifier>
..........<identifier>
............1003 id2
........<identifier-list>
..........44 ,
..........<variable-identifier>
............<identifier>
..............1004 id3
..........<identifier-list>
............<empty>
......41 )
....59 ;
....<block>
......<declaration>
........<label-declaration>
..........404 LABEL
..........<unsigned-integer>
............501 15
..........<label-list>
............44 ,
............<unsigned-integer>
..............502 16
............<label-list>
..............44 ,
..............<unsigned-integer>
................503 17
..............<label-list>
................<empty>
..........59 ;
......402 BEGIN
......<statement-list>
........<statement>
..........<unsigned-integer>
............501 15
..........58 :
..........<statement>
............40 (
............36 $
............<assembly-insert-file-identifier>
..............<identifier>
................1005 asmFile
............36 $
............41 )
........<statement-list>
..........<statement>
............59 ;
..........<statement-list>
............<statement>
..............405 GOTO
..............<unsigned-integer>
................502 16
..............59 ;
............<statement-list>
..............<statement>
................406 RETURN
................59 ;
..............<statement-list>
................<empty>
```

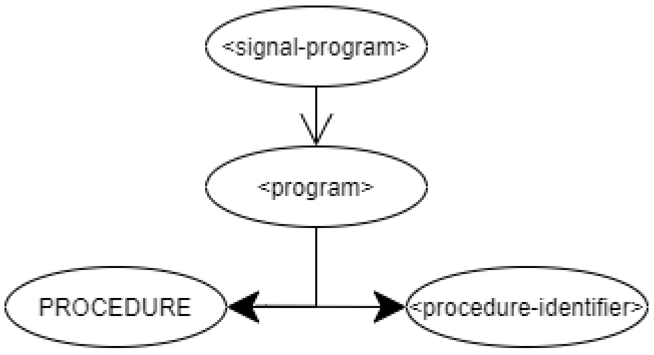
```
......403 END
....59 ;
```

Test03:

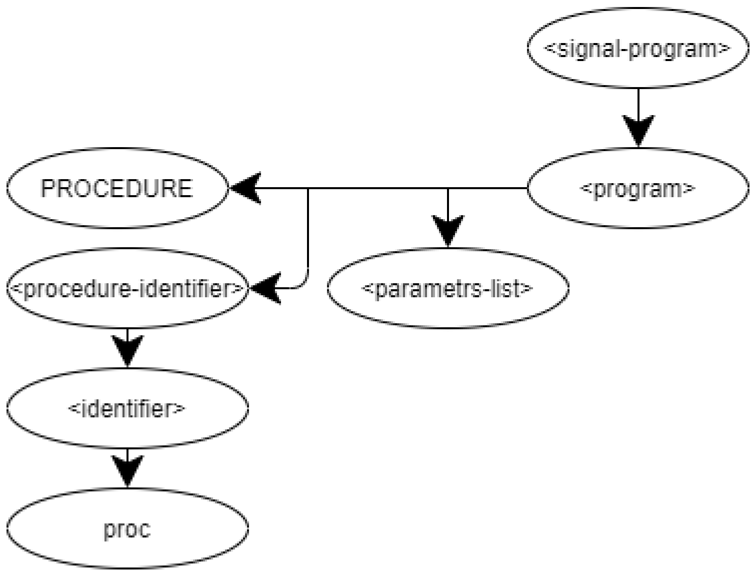
```
Line | Column | Ident token | Token
-------------------------------------------------
   1 |      1 |        1001 | PROCEDU1RE
   1 |     12 |        1002 | proc
   1 |     16 |          59 | ;
   2 |      1 |        1002 | proc
   2 |      6 |         402 | BEGIN
   2 |     11 |          58 | :
   3 |      1 |         404 | LABEL
   3 |      7 |        1003 | label1
   3 |     13 |          58 | :
   4 |      1 |          40 | (
   4 |      2 |        1004 | var1
   4 |      7 |        1005 | var2
   4 |     12 |        1006 | var3
   4 |     16 |          41 | )
   5 |      1 |          40 | (
   5 |      2 |          36 | $
   5 |      4 |        1007 | asmFile
   5 |     12 |          36 | $
   5 |     13 |          41 | )
   6 |      1 |         501 | 10
   7 |      1 |         403 | END

Identifier table
PROCEDU1RE 1001
asmFile 1007
label1 1003
proc 1002
var1 1004
var2 1005
var3 1006

Constant table
10 501

Parse tree
<signal-program>
..<program>
Parser : Error. Key word 'PROCEDURE'[1, 1] not found.
```



```
Test04:
Line | Column | Ident token | Token
-------------------------------------------------
   1 |      1 |         401 | PROCEDURE
   1 |     11 |          59 | ;
   2 |      1 |        1001 | proc
   2 |      6 |         402 | BEGIN
   2 |     11 |          58 | :
   3 |      1 |         404 | LABEL
   3 |      7 |         501 | 1
   3 |      8 |          58 | :
   3 |     10 |          40 | (
```

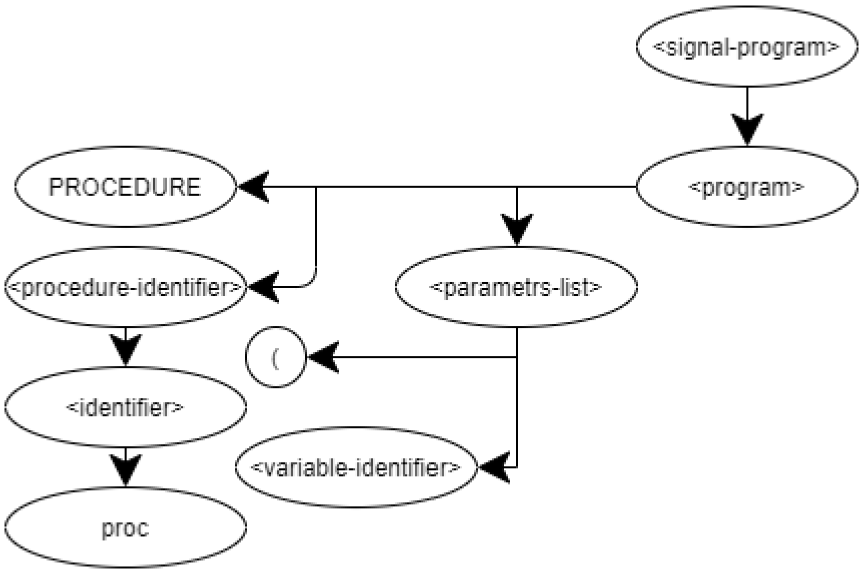


```
    3 |      11 |          36 | $
    3 |      13 |        1002 | asmFile
    3 |      21 |          36 | $
    3 |      22 |          41 | )
    4 |       1 |         403 | END
    4 |       4 |          59 | ;

Identifier table
asmFile 1002
proc 1001

Constant table
1 501

Parse tree
<signal-program>
..<program>
....401 PROCEDURE
....<procedure-identifier>
Parser : Error [1, 11]. Identifier not found.

Test05:
Line | Column | Ident token | Token
-------------------------------------------------
    1 |       1 |         401 | PROCEDURE
    1 |      11 |        1001 | proc
    1 |      16 |        1002 | id1
    1 |      19 |          44 | ,
    1 |      21 |        1003 | id2
    1 |      24 |          44 | ,
    1 |      26 |        1004 | id3
    1 |      29 |          41 | )
    1 |      30 |          59 | ;
    2 |       1 |         404 | LABEL
    2 |       7 |         501 | 15
    2 |       9 |          44 | ,
    2 |      11 |         502 | 16
    2 |      13 |          44 | ,
    2 |      15 |         503 | 17
    2 |      17 |          59 | ;
    3 |       1 |         402 | BEGIN
    3 |       7 |         501 | 15
    3 |      10 |          58 | :
    3 |      12 |          40 | (
    3 |      13 |          36 | $
    3 |      15 |        1005 | asmFile
    3 |      23 |          36 | $
    3 |      24 |          41 | )
    3 |      25 |          59 | ;
    4 |       1 |         405 | GOTO
    4 |       6 |         502 | 16
    4 |       8 |          59 | ;
    5 |       1 |         406 | RETURN
    5 |       7 |          59 | ;
    6 |       1 |         403 | END
    6 |       4 |          59 | ;

Identifier table
```



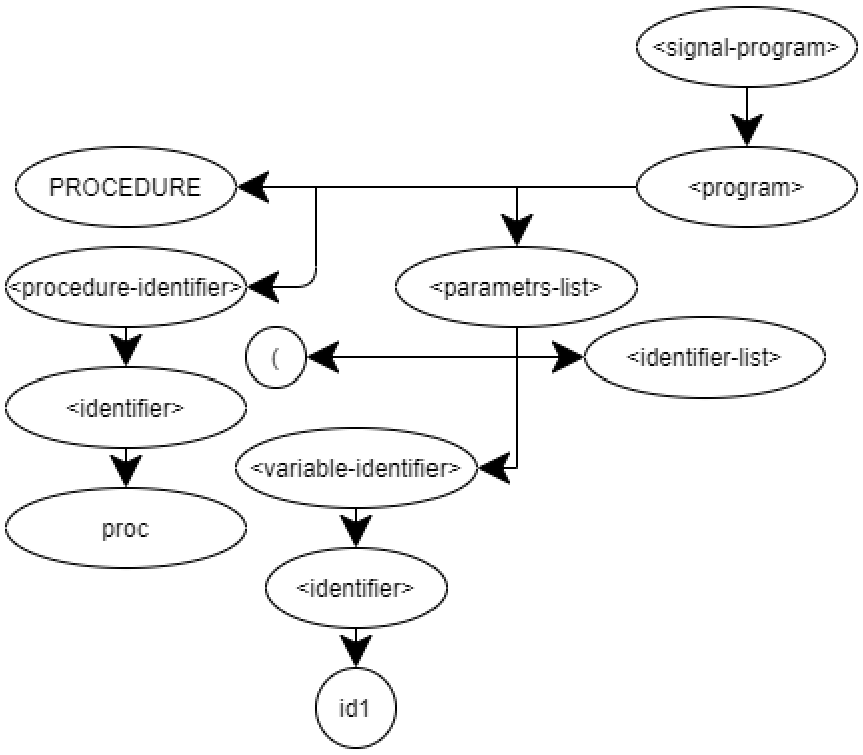
```
asmFile 1005
id1 1002
id2 1003
id3 1004
proc 1001

Constant table
15 501
16 502
17 503

Parse tree
<signal-program>
..<program>
....401 PROCEDURE
....<procedure-identifier>
......<identifier>
........1001 proc
....<parameters-list>
Parser : Error. Delimiter '('[1, 16] not found.

Test06:
Line | Column | Ident token | Token
-------------------------------------------------
    1 |      1 |         401 | PROCEDURE
    1 |     11 |        1001 | proc
    1 |     15 |          40 | (
    1 |     17 |          44 | ,
    1 |     19 |        1002 | id2
    1 |     22 |          44 | ,
    1 |     24 |        1003 | id3
    1 |     27 |          41 | )
    1 |     28 |          59 | ;
    2 |      1 |         404 | LABEL
    2 |      7 |         501 | 15
    2 |      9 |          44 | ,
    2 |     11 |         502 | 16
    2 |     13 |          44 | ,
    2 |     15 |         503 | 17
    2 |     17 |          59 | ;
    3 |      1 |         402 | BEGIN
    3 |      7 |         501 | 15
    3 |     10 |          58 | :
    3 |     12 |          40 | (
    3 |     13 |          36 | $
    3 |     15 |        1004 | asmFile
    3 |     23 |          36 | $
    3 |     24 |          41 | )
    3 |     25 |          59 | ;
    4 |      1 |         405 | GOTO
    4 |      6 |         502 | 16
    4 |      8 |          59 | ;
    5 |      1 |         406 | RETURN
    5 |      7 |          59 | ;
    6 |      1 |         403 | END
    6 |      4 |          59 | ;

Identifier table
```

```
asmFile 1004
id2 1002
id3 1003
proc 1001

Constant table
15 501
16 502
17 503

Parse tree
<signal-program>
..<program>
....401 PROCEDURE
....<procedure-identifier>
......<identifier>
........1001 proc
....<parameters-list>
......40 (
......<variable-identifier>
Parser : Error [1, 17]. Identifier not found.
```



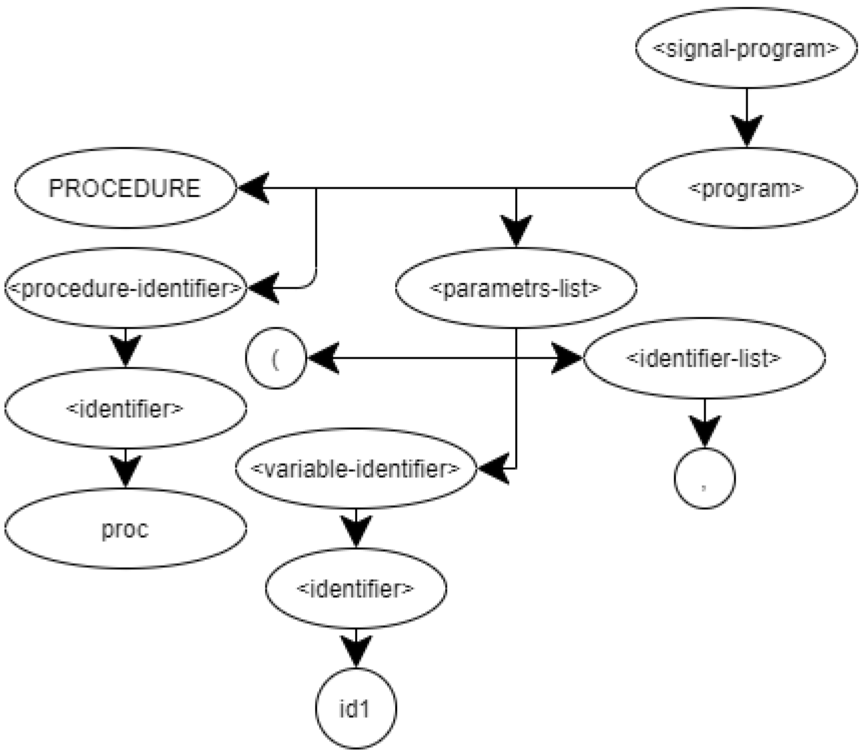
```
Test07:
Line | Column | Ident token | Token
------------------------------------------------
    1 |      1 |         401 | PROCEDURE
    1 |     11 |        1001 | proc
    1 |     15 |          40 | (
    1 |     16 |        1002 | id1
    1 |     20 |        1003 | id2
    1 |     23 |          44 | ,
    1 |     25 |        1004 | id3
    1 |     28 |          41 | )
    1 |     29 |          59 | ;
    2 |      1 |         404 | LABEL
    2 |      7 |         501 | 15
    2 |      9 |          44 | ,
    2 |     11 |         502 | 16
    2 |     13 |          44 | ,
    2 |     15 |         503 | 17
    2 |     17 |          59 | ;
    3 |      1 |         402 | BEGIN
    3 |      7 |         501 | 15
    3 |     10 |          58 | :
    3 |     12 |          40 | (
    3 |     13 |          36 | $
    3 |     15 |        1005 | asmFile
    3 |     23 |          36 | $
    3 |     24 |          41 | )
    3 |     25 |          59 | ;
    4 |      1 |         405 | GOTO
    4 |      6 |         502 | 16
    4 |      8 |          59 | ;
    5 |      1 |         406 | RETURN
    5 |      7 |          59 | ;
    6 |      1 |         403 | END
    6 |      4 |          59 | ;
```

```
Identifier table
asmFile 1005
id1 1002
id2 1003
id3 1004
proc 1001

Constant table
15 501
16 502
17 503

Parse tree
<signal-program>
..<program>
....401 PROCEDURE
....<procedure-identifier>
......<identifier>
........1001 proc
....<parameters-list>
......40 (
......<variable-identifier>
........<identifier>
..........1002 id1
......<identifier-list>
Parser : Error. Delimiter ','[1, 20] not found.
```

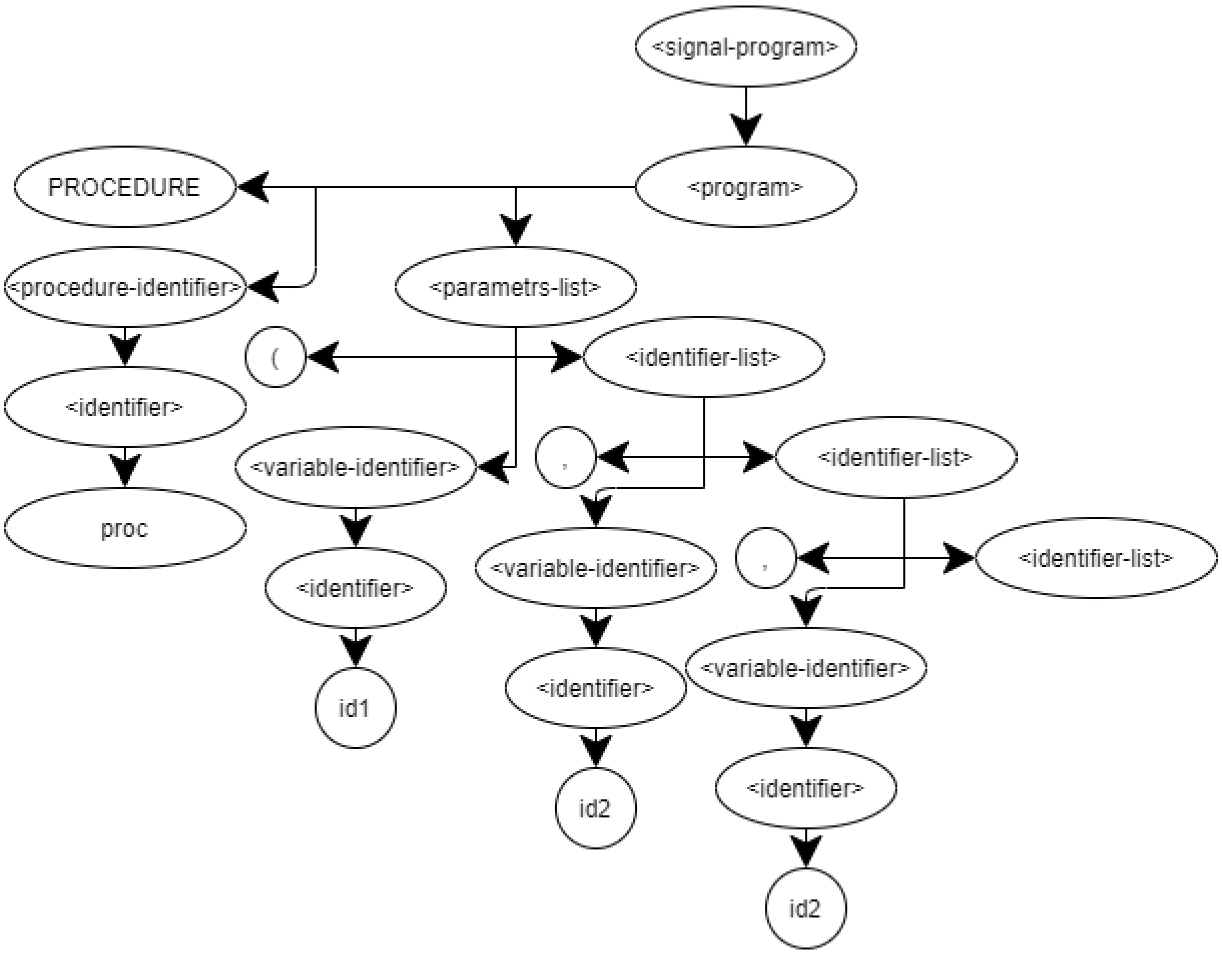


```
Test08:
Line | Column | Ident token | Token
-------------------------------------------------
   1 |      1 |         401 | PROCEDURE
   1 |     11 |        1001 | proc
   1 |     15 |          40 | (
   1 |     16 |        1002 | id1
   1 |     19 |          44 | ,
   1 |     21 |          44 | ,
   1 |     23 |        1003 | id3
   1 |     26 |          41 | )
   1 |     27 |          59 | ;
   2 |      1 |         404 | LABEL
   2 |      7 |         501 | 15
   2 |      9 |          44 | ,
   2 |     11 |         502 | 16
   2 |     13 |          44 | ,
   2 |     15 |         503 | 17
   2 |     17 |          59 | ;
   3 |      1 |         402 | BEGIN
   3 |      7 |         501 | 15
   3 |     10 |          58 | :
   3 |     12 |          40 | (
   3 |     13 |          36 | $
   3 |     15 |        1004 | asmFile
   3 |     23 |          36 | $
   3 |     24 |          41 | )
   3 |     25 |          59 | ;
   4 |      1 |         405 | GOTO
   4 |      6 |         502 | 16
   4 |      8 |          59 | ;
```

```
    5 |       1 |         406 | RETURN
    5 |       7 |          59 | ;
    6 |       1 |         403 | END
    6 |       4 |          59 | ;

Identifier table
asmFile 1004
id1 1002
id3 1003
proc 1001

Constant table
15 501
16 502
17 503

Parse tree
<signal-program>
..<program>
....401 PROCEDURE
....<procedure-identifier>
......<identifier>
........1001 proc
....<parameters-list>
......40 (
......<variable-identifier>
........<identifier>
..........1002 id1
......<identifier-list>
........44 ,
Parser : Error [1, 21]. Identifier not found.
```



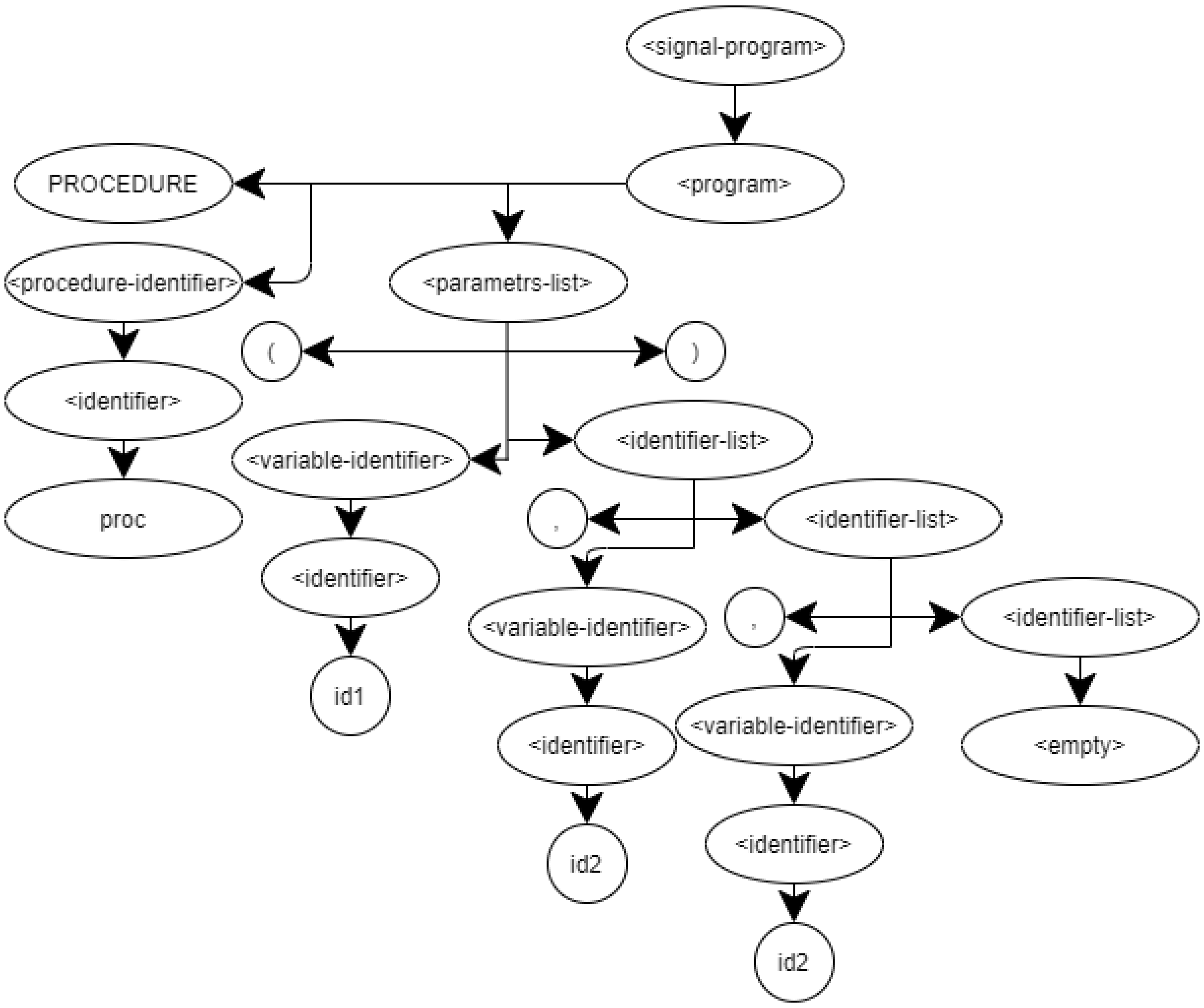
```
Test09:
Line | Column | Ident token | Token
---------------------------------------------
    1 |       1 |         401 | PROCEDURE
    1 |      11 |        1001 | proc
    1 |      15 |          40 | (
    1 |      16 |        1002 | id1
    1 |      19 |          44 | ,
    1 |      21 |        1003 | id2
    1 |      24 |          44 | ,
    1 |      26 |        1004 | id3
    1 |      29 |          59 | ;
    2 |       1 |         404 | LABEL
    2 |       7 |         501 | 15
    2 |       9 |          44 | ,
    2 |      11 |         502 | 16
    2 |      13 |          44 | ,
    2 |      15 |         503 | 17
    2 |      17 |          59 | ;
    3 |       1 |         402 | BEGIN
    3 |       7 |         501 | 15
    3 |      10 |          58 | :
    3 |      12 |          40 | (
    3 |      13 |          36 | $
    3 |      15 |        1005 | asmFile
    3 |      23 |          36 | $
    3 |      24 |          41 | )
    3 |      25 |          59 | ;
    4 |       1 |         405 | GOTO
    4 |       6 |         502 | 16
    4 |       8 |          59 | ;
    5 |       1 |         406 | RETURN
    5 |       7 |          59 | ;
    6 |       1 |         403 | END
    6 |       4 |          59 | ;

Identifier table
asmFile 1005
id1 1002
id2 1003
id3 1004
proc 1001

Constant table
15 501
16 502
17 503
```



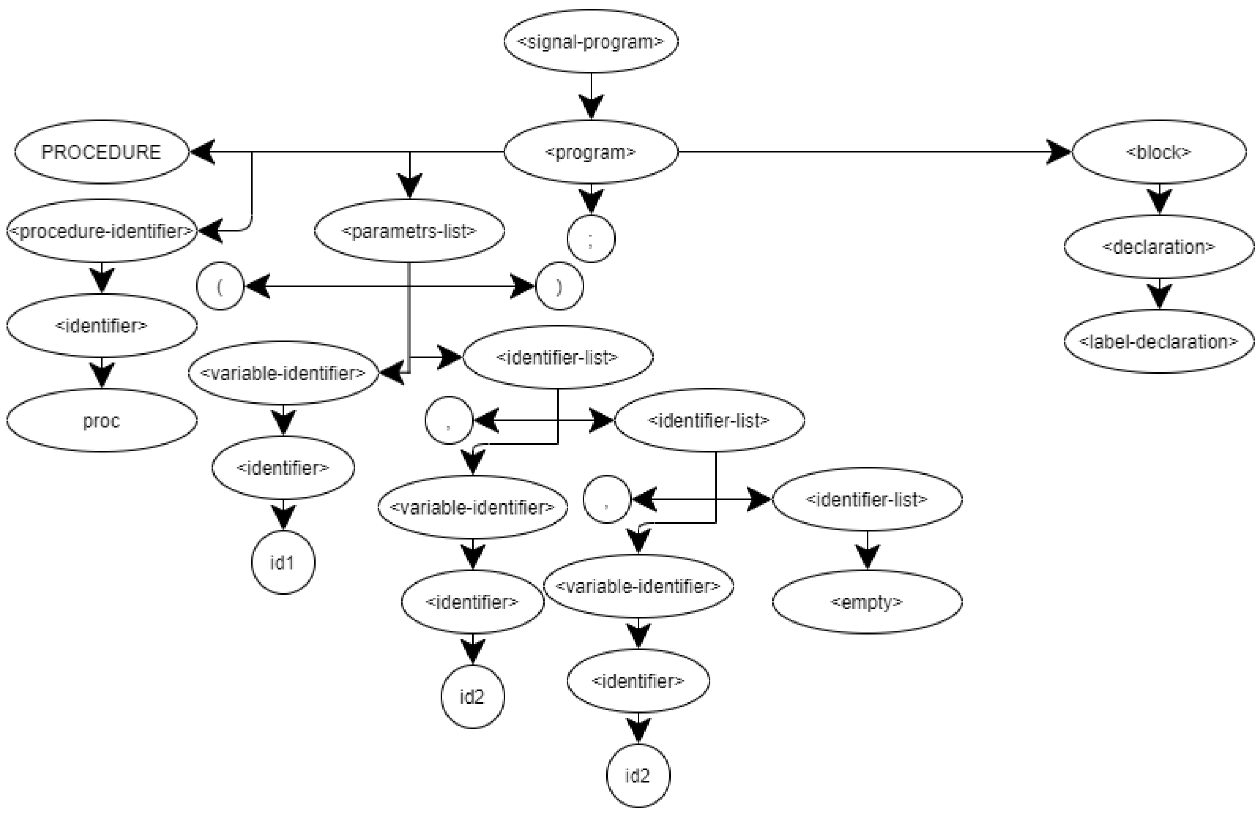
```
Parse tree
<signal-program>
..<program>
....401 PROCEDURE
....<procedure-identifier>
......<identifier>
........1001 proc
```

```
....<parameters-list>
......40 (
......<variable-identifier>
........<identifier>
..........1002 id1
......<identifier-list>
........44 ,
........<variable-identifier>
..........<identifier>
............1003 id2
........<identifier-list>
..........44 ,
..........<variable-identifier>
............<identifier>
..............1004 id3
..........<identifier-list>
Parser : Error [1, 29]. Wrong delimiter.

Test10:
Line | Column | Ident token | Token
-------------------------------------------------
    1 |      1 |         401 | PROCEDURE
    1 |     11 |        1001 | proc
    1 |     15 |          40 | (
    1 |     16 |        1002 | id1
    1 |     19 |          44 | ,
    1 |     21 |        1003 | id2
    1 |     24 |          44 | ,
    1 |     26 |        1004 | id3
    1 |     29 |          41 | )
    2 |      1 |         404 | LABEL
    2 |      7 |         501 | 15
    2 |      9 |          44 | ,
    2 |     11 |         502 | 16
    2 |     13 |          44 | ,
    2 |     15 |         503 | 17
    2 |     17 |          59 | ;
    3 |      1 |         402 | BEGIN
    3 |      7 |         501 | 15
    3 |     10 |          58 | :
    3 |     12 |          40 | (
    3 |     13 |          36 | $
    3 |     15 |        1005 | asmFile
    3 |     23 |          36 | $
    3 |     24 |          41 | )
    3 |     25 |          59 | ;
    4 |      1 |         405 | GOTO
    4 |      6 |         502 | 16
    4 |      8 |          59 | ;
    5 |      1 |         406 | RETURN
    5 |      7 |          59 | ;
    6 |      1 |         403 | END
    6 |      4 |          59 | ;

Identifier table
asmFile 1005
id1 1002
id2 1003
```

```
id3 1004
proc 1001

Constant table
15 501
16 502
17 503
```

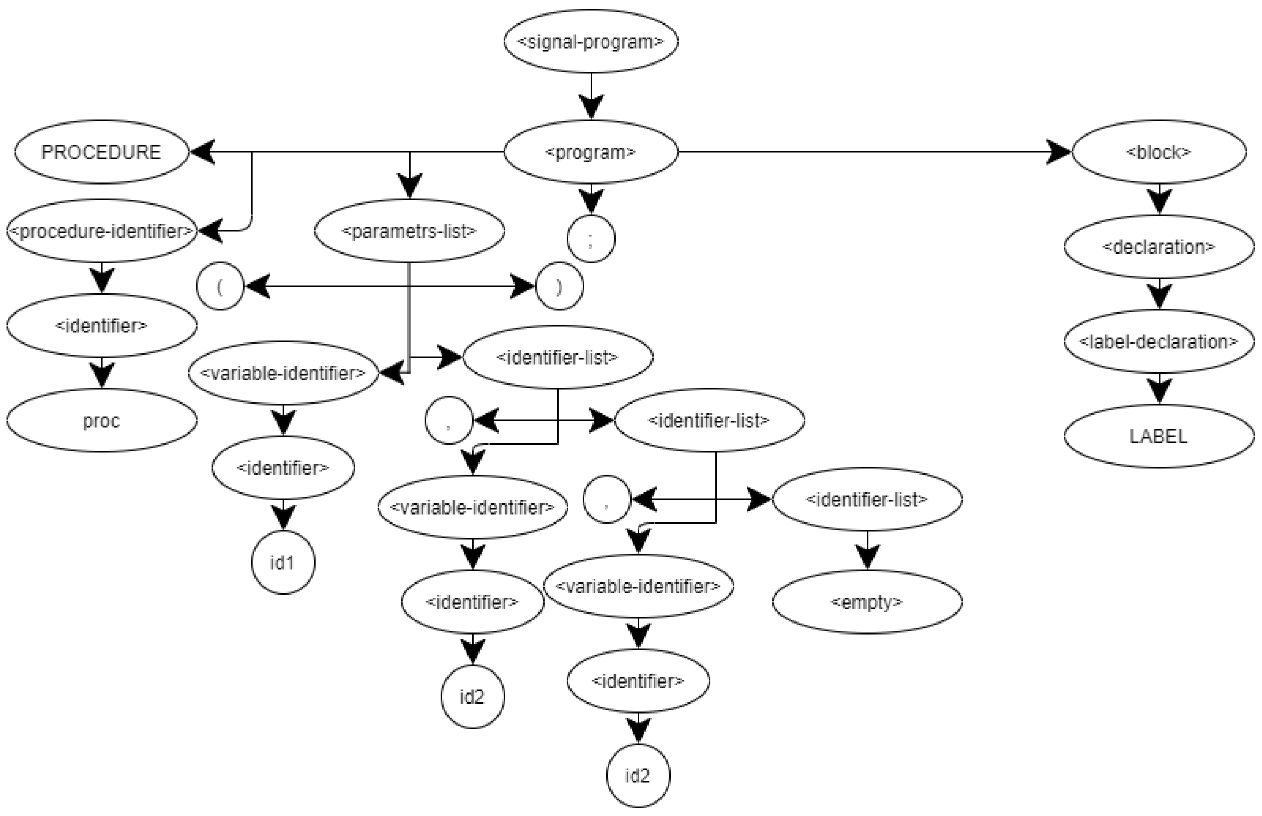


```
Parse tree
<signal-program>
..<program>
....401 PROCEDURE
....<procedure-identifier>
......<identifier>
........1001 proc
....<parameters-list>
......40 (
......<variable-identifier>
........<identifier>
..........1002 id1
......<identifier-list>
........44 ,
........<variable-identifier>
..........<identifier>
............1003 id2
........<identifier-list>
..........44 ,
..........<variable-identifier>
............<identifier>
..............1004 id3
..........<identifier-list>
............<empty>
......41 )
Parser : Error. Delimiter ';'[2, 1] not found.

Test11:
Line | Column | Ident token | Token
-------------------------------------------------
    1 |      1 |         401 | PROCEDURE
    1 |     11 |        1001 | proc
    1 |     15 |          40 | (
    1 |     16 |        1002 | id1
    1 |     19 |          44 | ,
    1 |     21 |        1003 | id2
    1 |     24 |          44 | ,
    1 |     26 |        1004 | id3
    1 |     29 |          41 | )
    1 |     30 |          59 | ;
    2 |      1 |        1005 | LABUL
    2 |      7 |         501 | 15
    2 |      9 |          44 | ,
    2 |     11 |         502 | 16
    2 |     13 |          44 | ,
    2 |     15 |         503 | 17
    2 |     17 |          59 | ;
    3 |      1 |         402 | BEGIN
    3 |      7 |         501 | 15
    3 |     10 |          58 | :
    3 |     12 |          40 | (
    3 |     13 |          36 | $
    3 |     15 |        1006 | asmFile
    3 |     23 |          36 | $
    3 |     24 |          41 | )
    3 |     25 |          59 | ;
    4 |      1 |         405 | GOTO
    4 |      6 |         502 | 16
    4 |      8 |          59 | ;
    5 |      1 |         406 | RETURN
    5 |      7 |          59 | ;
    6 |      1 |         403 | END
    6 |      4 |          59 | ;

Identifier table
LABUL 1005
asmFile 1006
id1 1002
id2 1003
id3 1004
proc 1001

Constant table
15 501
16 502
17 503
```



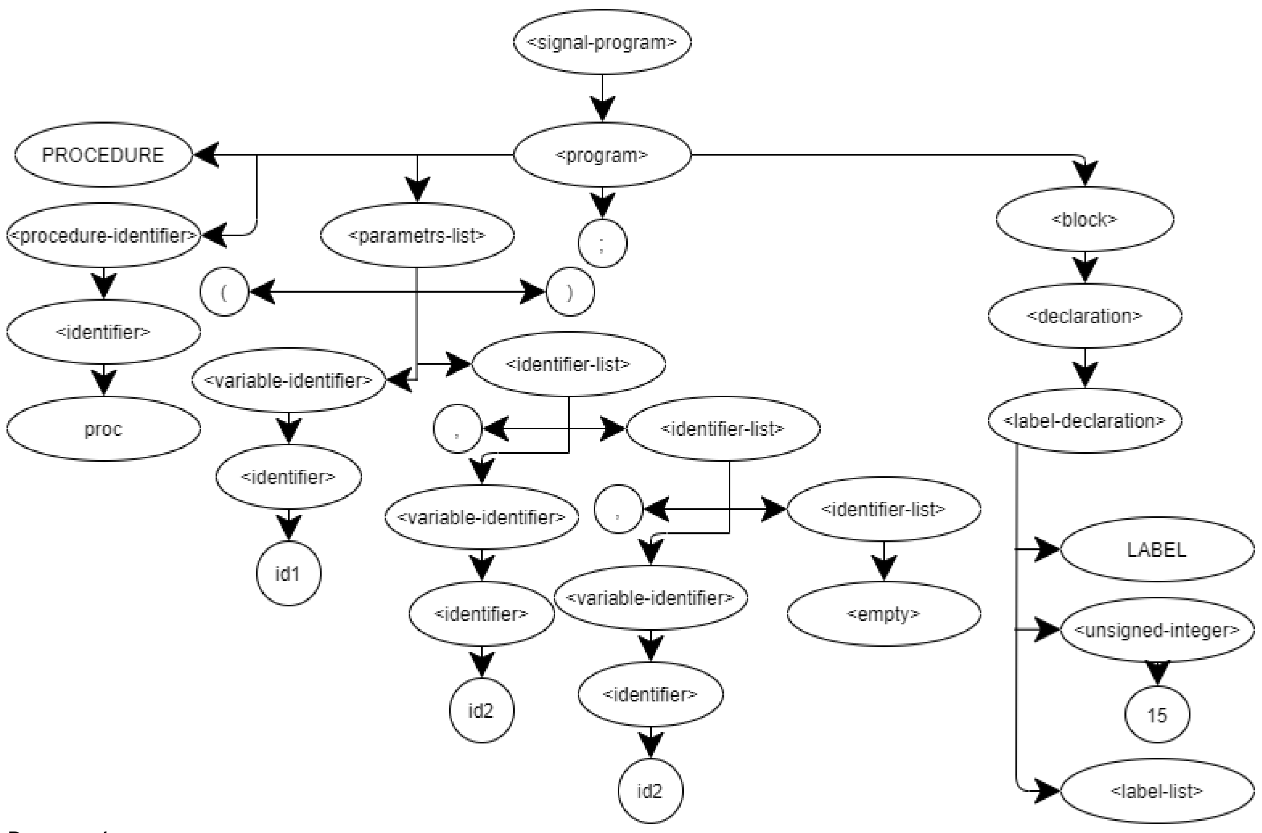
```
Parse tree
<signal-program>
..<program>
....401 PROCEDURE
....<procedure-identifier>
......<identifier>
........1001 proc
....<parameters-list>
......40 (
......<variable-identifier>
........<identifier>
..........1002 id1
......<identifier-list>
........44 ,
........<variable-identifier>
..........<identifier>
............1003 id2
........<identifier-list>
..........44 ,
..........<variable-identifier>
............<identifier>
..............1004 id3
..........<identifier-list>
............<empty>
......41 )
....59 ;
....<block>
......<declaration>
........<label-declaration>
Parser : Error. Key word 'LABEL'[2, 1] not found.

Test12:
Line | Column | Ident token | Token
-------------------------------------------------
```

```
    1 |      1 |         401 | PROCEDURE
    1 |     11 |        1001 | proc
    1 |     15 |          40 | (
    1 |     16 |        1002 | id1
    1 |     19 |          44 | ,
    1 |     21 |        1003 | id2
    1 |     24 |          44 | ,
    1 |     26 |        1004 | id3
    1 |     29 |          41 | )
    1 |     30 |          59 | ;
    2 |      1 |         404 | LABEL
    2 |      7 |          44 | ,
    2 |      9 |         501 | 16
    2 |     11 |          44 | ,
    2 |     13 |         502 | 17
    2 |     15 |          59 | ;
    3 |      1 |         402 | BEGIN
    3 |      7 |         503 | 15
    3 |     10 |          58 | :
    3 |     12 |          40 | (
    3 |     13 |          36 | $
    3 |     15 |        1005 | asmFile
    3 |     23 |          36 | $
    3 |     24 |          41 | )
    3 |     25 |          59 | ;
    4 |      1 |         405 | GOTO
    4 |      6 |         501 | 16
    4 |      8 |          59 | ;
    5 |      1 |         406 | RETURN
    5 |      7 |          59 | ;
    6 |      1 |         403 | END
    6 |      4 |          59 | ;

Identifier table
asmFile 1005
id1 1002
id2 1003
id3 1004
proc 1001

Constant table
15 503
16 501
17 502
```



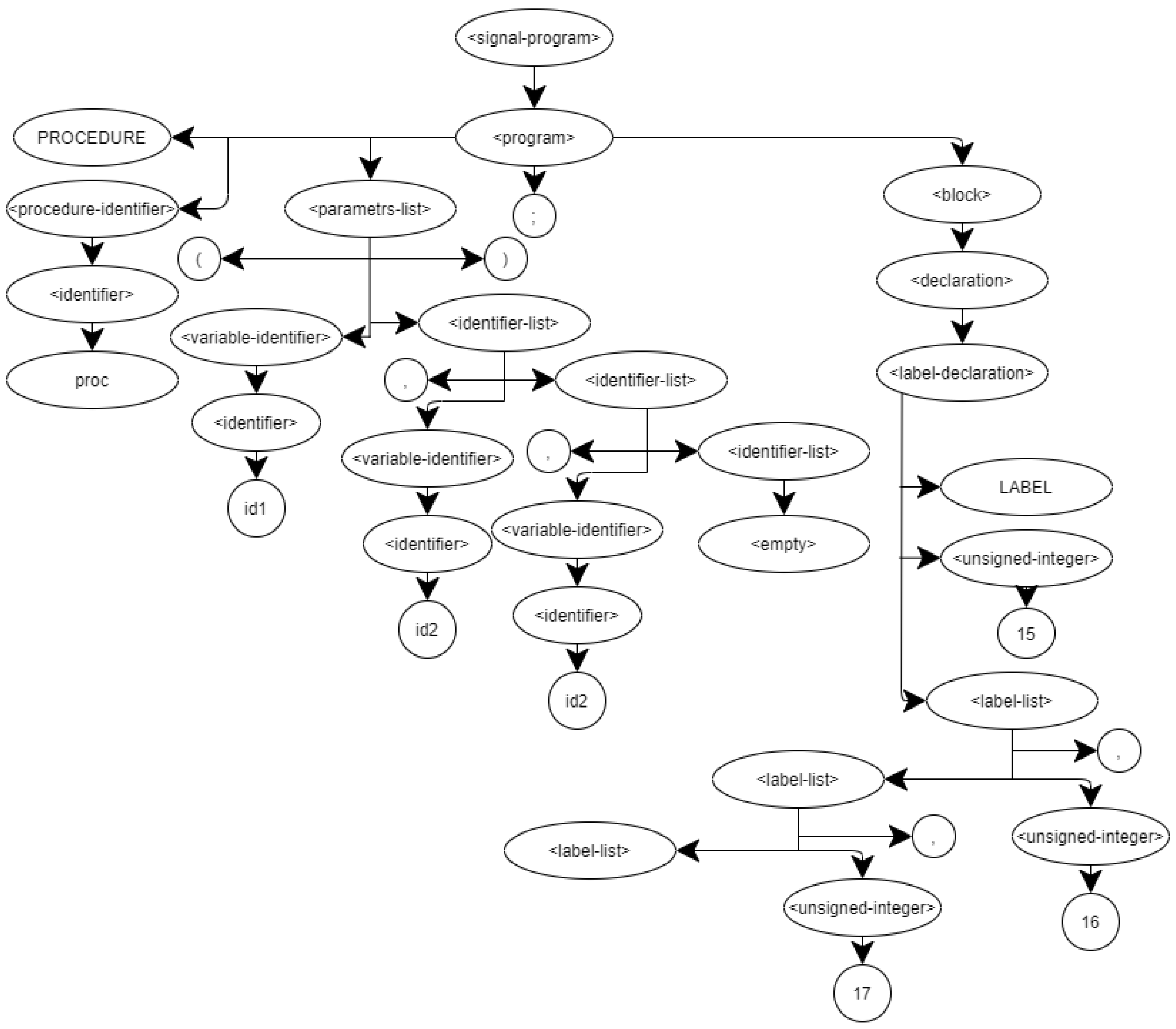
```
Parse tree
<signal-program>
..<program>
....401 PROCEDURE
....<procedure-identifier>
......<identifier>
........1001 proc
....<parameters-list>
......40 (
......<variable-identifier>
........<identifier>
..........1002 id1
......<identifier-list>
........44 ,
........<variable-identifier>
..........<identifier>
............1003 id2
........<identifier-list>
..........44 ,
..........<variable-identifier>
............<identifier>
..............1004 id3
..........<identifier-list>
............<empty>
......41 )
....59 ;
....<block>
......<declaration>
........<label-declaration>
..........404 LABEL
Parser : Error [2, 7]. Unsigned integer not found.

Test13:
Line | Column | Ident token | Token
```

```
--------------------------------------------------
    1 |      1 |         401 | PROCEDURE
    1 |     11 |        1001 | proc
    1 |     15 |          40 | (
    1 |     16 |        1002 | id1
    1 |     19 |          44 | ,
    1 |     21 |        1003 | id2
    1 |     24 |          44 | ,
    1 |     26 |        1004 | id3
    1 |     29 |          41 | )
    1 |     30 |          59 | ;
    2 |      1 |         404 | LABEL
    2 |      7 |         501 | 15
    2 |     10 |         502 | 16
    2 |     12 |          44 | ,
    2 |     14 |         503 | 17
    2 |     16 |          59 | ;
    3 |      1 |         402 | BEGIN
    3 |      7 |         501 | 15
    3 |     10 |          58 | :
    3 |     12 |          40 | (
    3 |     13 |          36 | $
    3 |     15 |        1005 | asmFile
    3 |     23 |          36 | $
    3 |     24 |          41 | )
    3 |     25 |          59 | ;
    4 |      1 |         405 | GOTO
    4 |      6 |         502 | 16
    4 |      8 |          59 | ;
    5 |      1 |         406 | RETURN
    5 |      7 |          59 | ;
    6 |      1 |         403 | END
    6 |      4 |          59 | ;

Identifier table
asmFile 1005
id1 1002
id2 1003
id3 1004
proc 1001

Constant table
15 501
16 502
17 503
```



```
Parse tree
<signal-program>
..<program>
....401 PROCEDURE
....<procedure-identifier>
......<identifier>
........1001 proc
....<parameters-list>
......40 (
......<variable-identifier>
........<identifier>
..........1002 id1
......<identifier-list>
........44 ,
........<variable-identifier>
..........<identifier>
............1003 id2
........<identifier-list>
..........44 ,
..........<variable-identifier>
............<identifier>
..............1004 id3
..........<identifier-list>
............<empty>
......41 )
....59 ;
....<block>
......<declaration>
........<label-declaration>
..........404 LABEL
..........<unsigned-integer>
............501 15
..........<label-list>
```

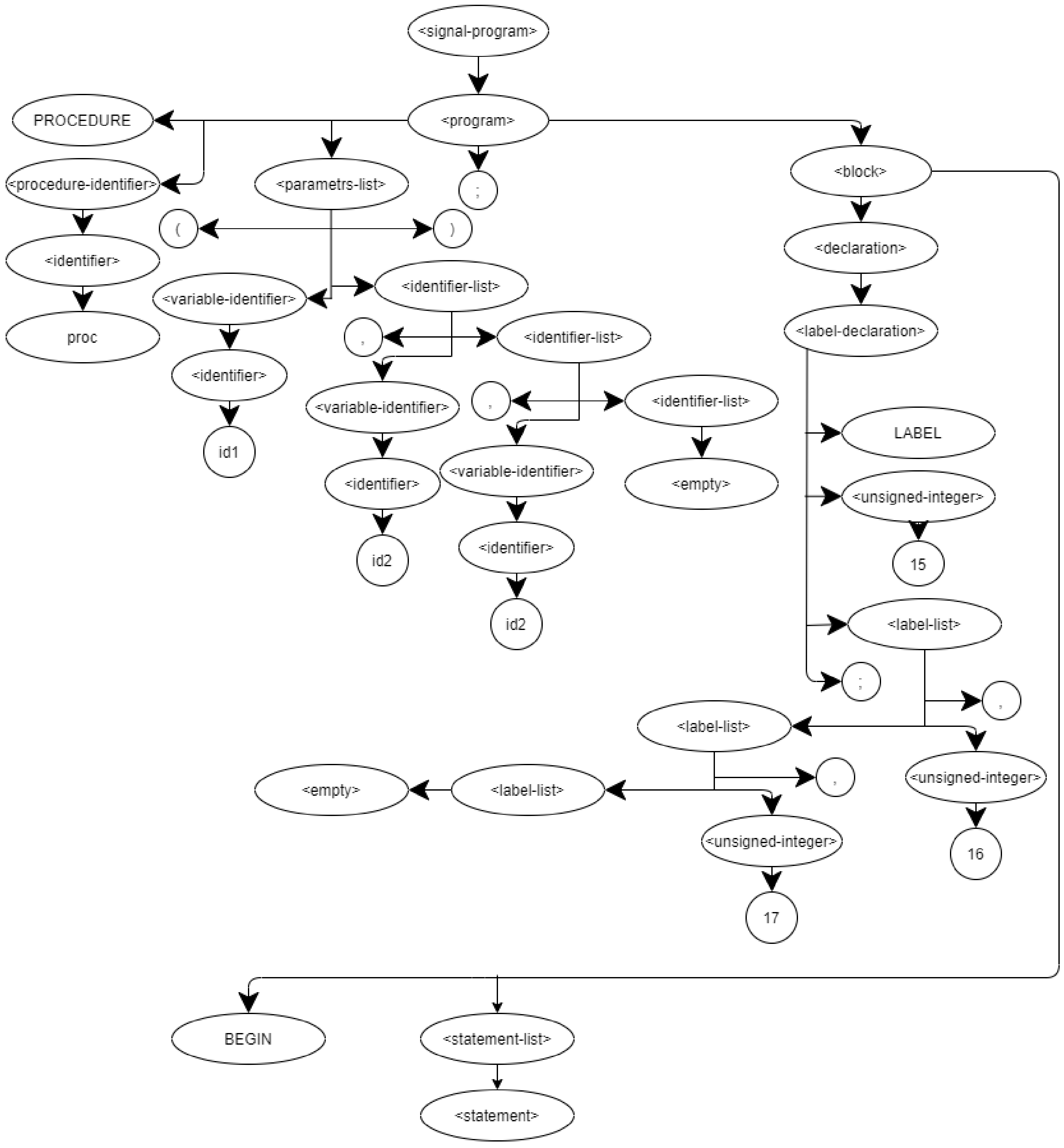
```
Parser : Error. Delimiter ','[2, 10] not found.

Test14:
Line | Column | Ident token | Token
-------------------------------------------------
    1 |      1 |         401 | PROCEDURE
    1 |     11 |        1001 | proc
    1 |     15 |          40 | (
    1 |     16 |        1002 | id1
    1 |     19 |          44 | ,
    1 |     21 |        1003 | id2
    1 |     24 |          44 | ,
    1 |     26 |        1004 | id3
    1 |     29 |          41 | )
    1 |     30 |          59 | ;
    2 |      1 |         404 | LABEL
    2 |      7 |         501 | 15
    2 |      9 |          44 | ,
    2 |     11 |         502 | 16
    2 |     13 |          44 | ,
    2 |     15 |         503 | 17
    3 |      1 |         402 | BEGIN
    3 |      7 |         501 | 15
    3 |     10 |          58 | :
    3 |     12 |          40 | (
    3 |     13 |          36 | $
    3 |     15 |        1005 | asmFile
    3 |     23 |          36 | $
    3 |     24 |          41 | )
    3 |     25 |          59 | ;
    4 |      1 |         405 | GOTO
    4 |      6 |         502 | 16
    4 |      8 |          59 | ;
    5 |      1 |         406 | RETURN
    5 |      7 |          59 | ;
    6 |      1 |         403 | END
    6 |      4 |          59 | ;

Identifier table
asmFile 1005
id1 1002
id2 1003
id3 1004
proc 1001

Constant table
15 501
16 502
17 503
```



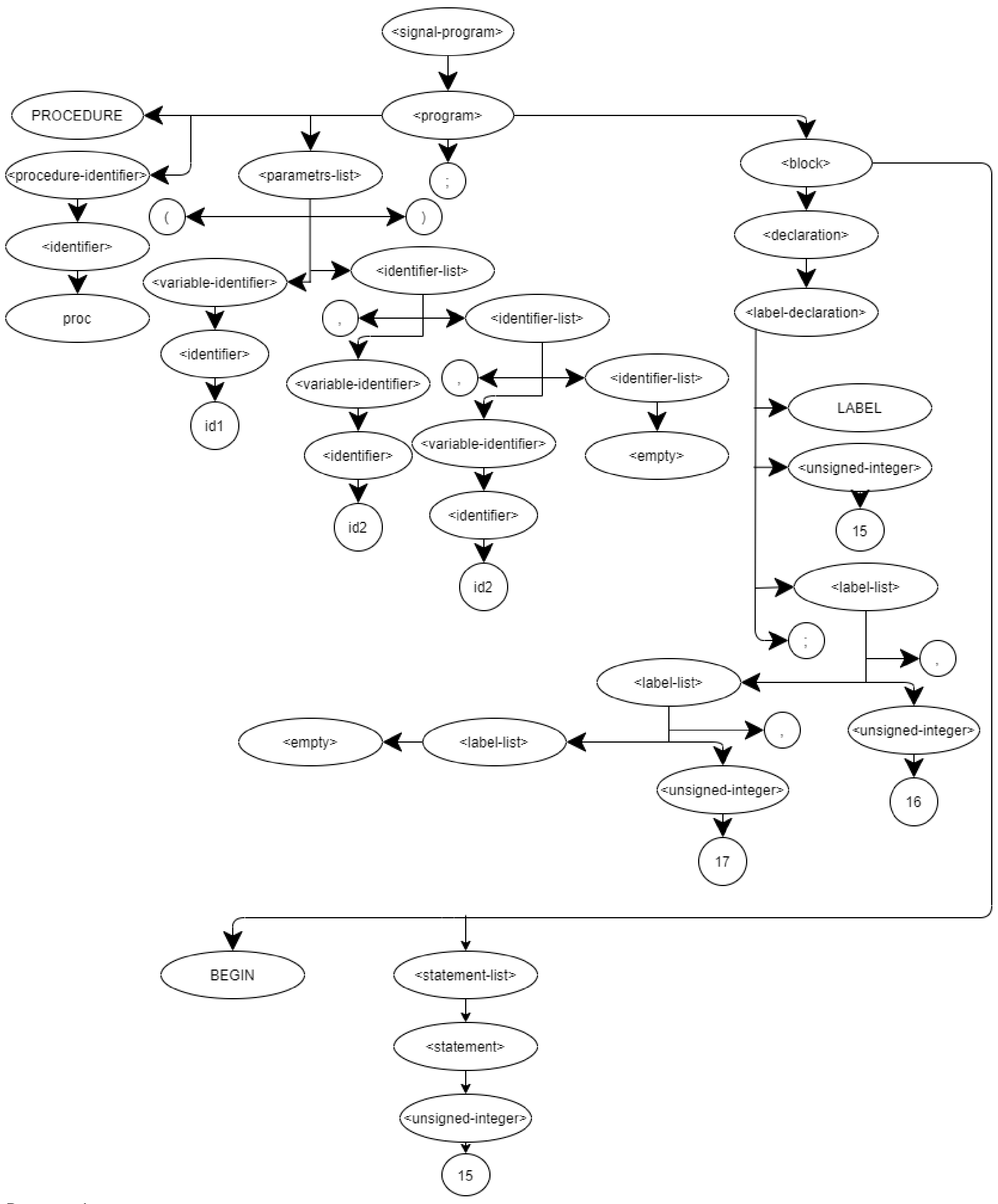
```
Parse tree
<signal-program>
..<program>
....401 PROCEDURE
....<procedure-identifier>
......<identifier>
........1001 proc
....<parameters-list>
......40 (
......<variable-identifier>
........<identifier>
..........1002 id1
......<identifier-list>
........44 ,
........<variable-identifier>
..........<identifier>
............1003 id2
........<identifier-list>
..........44 ,
..........<variable-identifier>
............<identifier>
..............1004 id3
..........<identifier-list>
............<empty>
......41 )
```

```
....59 ;
....<block>
......<declaration>
........<label-declaration>
..........404 LABEL
..........<unsigned-integer>
............501 15
..........<label-list>
............44 ,
............<unsigned-integer>
..............502 16
............<label-list>
..............44 ,
..............<unsigned-integer>
................503 17
..............<label-list>
Parser : Error. Delimiter ','[3, 1] not found.

Test15:
Line | Column | Ident token | Token
-------------------------------------------------
    1 |      1 |         401 | PROCEDURE
    1 |     11 |        1001 | proc
    1 |     15 |          40 | (
    1 |     16 |        1002 | id1
    1 |     19 |          44 | ,
    1 |     21 |        1003 | id2
    1 |     24 |          44 | ,
    1 |     26 |        1004 | id3
    1 |     29 |          41 | )
    1 |     30 |          59 | ;
    2 |      1 |         404 | LABEL
    2 |      7 |         501 | 15
    2 |      9 |          44 | ,
    2 |     11 |         502 | 16
    2 |     13 |          44 | ,
    2 |     15 |         503 | 17
    2 |     17 |          59 | ;
    3 |      1 |         402 | BEGIN
    3 |      8 |          58 | :
    3 |     10 |          40 | (
    3 |     11 |          36 | $
    3 |     13 |        1005 | asmFile
    3 |     21 |          36 | $
    3 |     22 |          41 | )
    3 |     23 |          59 | ;
    4 |      1 |         405 | GOTO
    4 |      6 |         502 | 16
    4 |      8 |          59 | ;
    5 |      1 |         406 | RETURN
    5 |      7 |          59 | ;
    6 |      1 |         403 | END
    6 |      4 |          59 | ;

Identifier table
asmFile 1005
id1 1002
id2 1003
```

```
id3 1004
proc 1001

Constant table
15 501
16 502
17 503
```

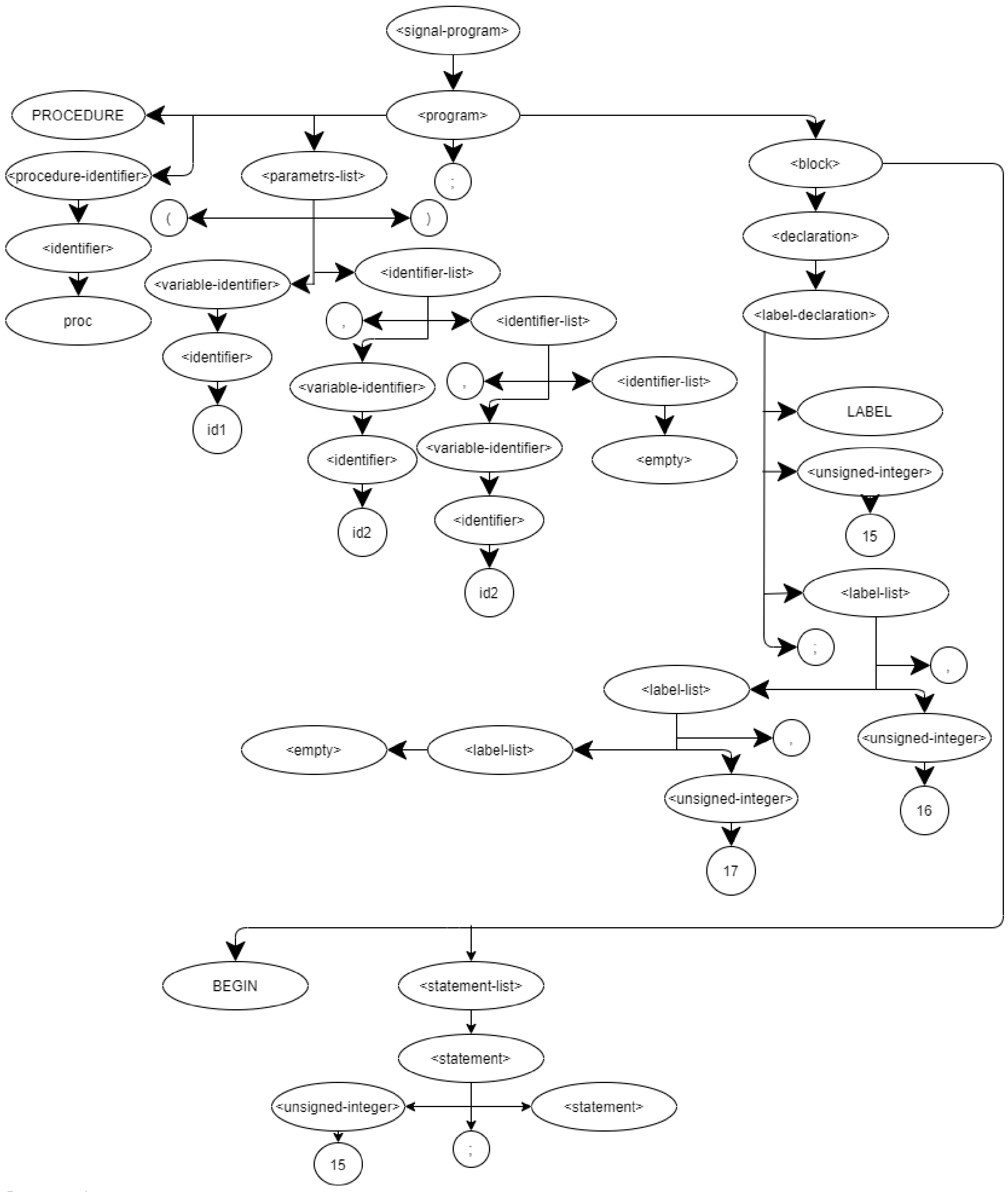


```
Parse tree
<signal-program>
..<program>
....401 PROCEDURE
....<procedure-identifier>
......<identifier>
........1001 proc
....<parameters-list>
......40 (
......<variable-identifier>
........<identifier>
..........1002 id1
......<identifier-list>
........44 ,
........<variable-identifier>
..........<identifier>
............1003 id2
........<identifier-list>
..........44 ,
..........<variable-identifier>
............<identifier>
..............1004 id3
..........<identifier-list>
............<empty>
......41 )
....59 ;
....<block>
......<declaration>
........<label-declaration>
..........404 LABEL
..........<unsigned-integer>
............501 15
..........<label-list>
............44 ,
............<unsigned-integer>
..............502 16
............<label-list>
..............44 ,
..............<unsigned-integer>
................503 17
..............<label-list>
................<empty>
..........59 ;
......402 BEGIN
......<statement-list>
........<statement>
Parser : Error [3, 8]. Wrong delimiter.

Test16:
Line | Column | Ident token | Token
-------------------------------------------------
    1 |      1 |         401 | PROCEDURE
    1 |     11 |        1001 | proc
    1 |     15 |          40 | (
    1 |     16 |        1002 | id1
    1 |     19 |          44 | ,
    1 |     21 |        1003 | id2
    1 |     24 |          44 | ,
    1 |     26 |        1004 | id3
    1 |     29 |          41 | )
    1 |     30 |          59 | ;
    2 |      1 |         404 | LABEL
    2 |      7 |         501 | 15
    2 |      9 |          44 | ,
    2 |     11 |         502 | 16
    2 |     13 |          44 | ,
    2 |     15 |         503 | 17
    2 |     17 |          59 | ;
    3 |      1 |         402 | BEGIN
    3 |       7 |         501 | 15
    3 |      11 |          40 | (
    3 |      12 |          36 | $
    3 |      14 |        1005 | asmFile
    3 |      22 |          36 | $
    3 |      23 |          41 | )
    3 |      24 |          59 | ;
    4 |       1 |         405 | GOTO
    4 |       6 |         502 | 16
    4 |       8 |          59 | ;
    5 |       1 |         406 | RETURN
    5 |       7 |          59 | ;
    6 |       1 |         403 | END
    6 |       4 |          59 | ;

Identifier table
asmFile 1005
id1 1002
id2 1003
id3 1004
proc 1001

Constant table
15 501
16 502
17 503
```



```
Parse tree
<signal-program>
..<program>
....401 PROCEDURE
....<procedure-identifier>
......<identifier>
........1001 proc
....<parameters-list>
......40 (
......<variable-identifier>
........<identifier>
..........1002 id1
......<identifier-list>
........44 ,
........<variable-identifier>
..........<identifier>
............1003 id2
........<identifier-list>
..........44 ,
..........<variable-identifier>
............<identifier>
..............1004 id3
..........<identifier-list>
............<empty>
......41 )
....59 ;
....<block>
......<declaration>
........<label-declaration>
..........404 LABEL
..........<unsigned-integer>
............501 15
..........<label-list>
............44 ,
............<unsigned-integer>
..............502 16
............<label-list>
..............44 ,
..............<unsigned-integer>
................503 17
..............<label-list>
................<empty>
..........59 ;
......402 BEGIN
......<statement-list>
........<statement>
..........<unsigned-integer>
............501 15
Parser : Error [3, 11]. Wrong delimiter.
```

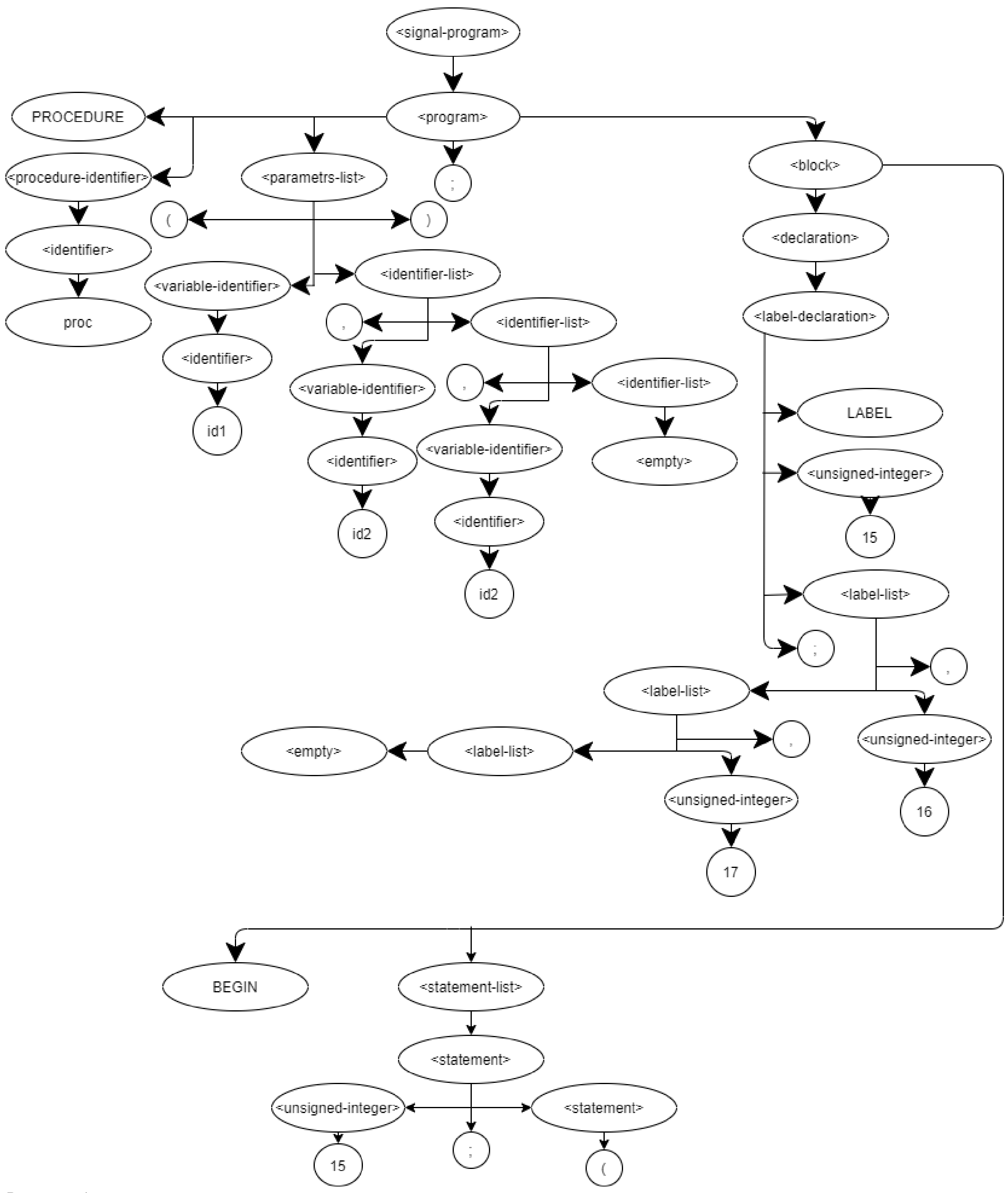
```
Test17:
Line | Column | Ident token | Token
-------------------------------------------------
    1 |      1 |         401 | PROCEDURE
    1 |     11 |        1001 | proc
    1 |     15 |          40 | (
    1 |     16 |        1002 | id1
    1 |     19 |          44 | ,
    1 |     21 |        1003 | id2
    1 |     24 |          44 | ,
    1 |     26 |        1004 | id3
    1 |     29 |          41 | )
    1 |     30 |          59 | ;
    2 |      1 |         404 | LABEL
    2 |      7 |         501 | 15
    2 |      9 |          44 | ,
    2 |     11 |         502 | 16
    2 |     13 |          44 | ,
    2 |     15 |         503 | 17
    2 |     17 |          59 | ;
    3 |      1 |         402 | BEGIN
    3 |      7 |         501 | 15
    3 |     10 |          58 | :
    3 |     12 |          36 | $
    3 |     14 |        1005 | asmFile
    3 |     22 |          36 | $
    3 |     23 |          41 | )
    3 |     24 |          59 | ;
    4 |      1 |         405 | GOTO
    4 |      6 |         502 | 16
    4 |      8 |          59 | ;
    5 |      1 |         406 | RETURN
    5 |      7 |          59 | ;
    6 |      1 |         403 | END
    6 |      4 |          59 | ;

Identifier table
asmFile 1005
id1 1002
id2 1003
id3 1004
proc 1001

Constant table
15 501
16 502
17 503
```



```
Parse tree
<signal-program>
..<program>
....401 PROCEDURE
....<procedure-identifier>
......<identifier>
........1001 proc
....<parameters-list>
......40 (
......<variable-identifier>
........<identifier>
..........1002 id1
......<identifier-list>
........44 ,
........<variable-identifier>
..........<identifier>
............1003 id2
........<identifier-list>
..........44 ,
..........<variable-identifier>
............<identifier>
..............1004 id3
..........<identifier-list>
............<empty>
......41 )
....59 ;
....<block>
......<declaration>
........<label-declaration>
..........404 LABEL
..........<unsigned-integer>
............501 15
..........<label-list>
............44 ,
............<unsigned-integer>
..............502 16
............<label-list>
..............44 ,
..............<unsigned-integer>
................503 17
..............<label-list>
................<empty>
..........59 ;
......402 BEGIN
......<statement-list>
........<statement>
..........<unsigned-integer>
............501 15
..........58 :
..........<statement>
Parser : Error [3, 12]. Wrong delimiter.
```

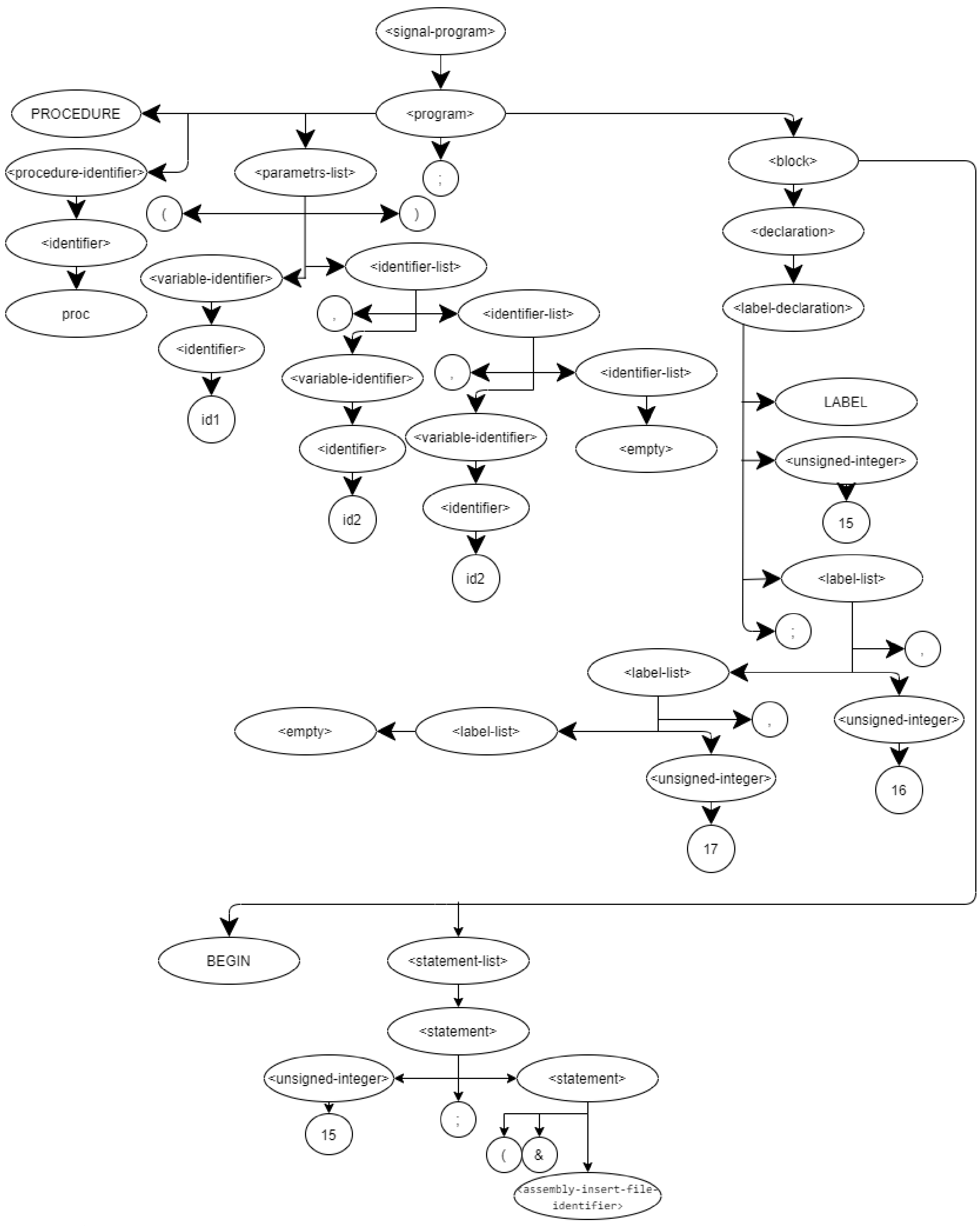
```
Test18:
Line | Column | Ident token | Token
-------------------------------------------------
   1 |      1 |         401 | PROCEDURE
   1 |     11 |        1001 | proc
   1 |     15 |          40 | (
   1 |     16 |        1002 | id1
   1 |     19 |          44 | ,
   1 |     21 |        1003 | id2
   1 |     24 |          44 | ,
   1 |     26 |        1004 | id3
   1 |     29 |          41 | )
   1 |     30 |          59 | ;
   2 |      1 |         404 | LABEL
   2 |      7 |         501 | 15
   2 |      9 |          44 | ,
   2 |     11 |         502 | 16
   2 |     13 |          44 | ,
   2 |     15 |         503 | 17
```

```
    2 |      17 |           59 | ;
    3 |       1 |          402 | BEGIN
    3 |       7 |          501 | 15
    3 |      10 |           58 | :
    3 |      12 |           40 | (
    3 |      14 |         1005 | asmFile
    3 |      22 |           36 | $
    3 |      23 |           41 | )
    3 |      24 |           59 | ;
    4 |       1 |          405 | GOTO
    4 |       6 |          502 | 16
    4 |       8 |           59 | ;
    5 |       1 |          406 | RETURN
    5 |       7 |           59 | ;
    6 |       1 |          403 | END
    6 |       4 |           59 | ;

Identifier table
asmFile 1005
id1 1002
id2 1003
id3 1004
proc 1001

Constant table
15 501
16 502
17 503
```



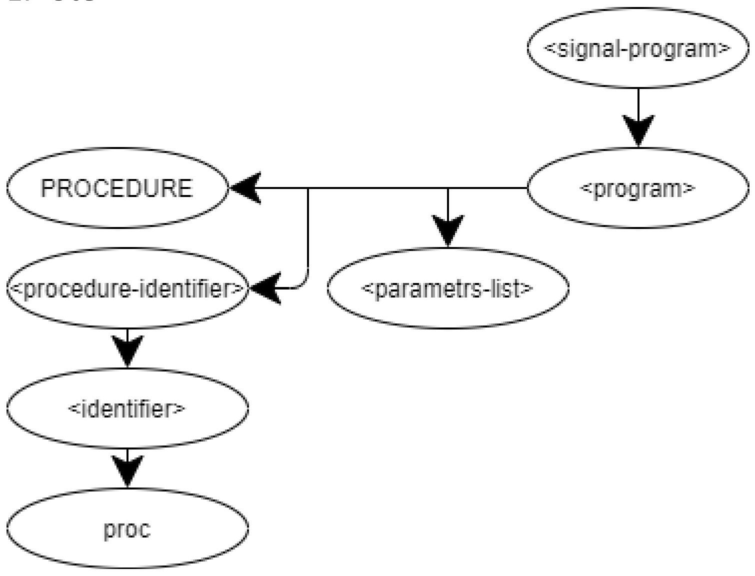
```
Parse tree
<signal-program>
..<program>
....401 PROCEDURE
....<procedure-identifier>
......<identifier>
........1001 proc
....<parameters-list>
......40 (
......<variable-identifier>
........<identifier>
..........1002 id1
......<identifier-list>
........44 ,
........<variable-identifier>
..........<identifier>
............1003 id2
........<identifier-list>
..........44 ,
..........<variable-identifier>
............<identifier>
..............1004 id3
..........<identifier-list>
............<empty>
......41 )
....59 ;
....<block>
......<declaration>
........<label-declaration>
..........404 LABEL
..........<unsigned-integer>
............501 15
..........<label-list>
............44 ,
............<unsigned-integer>
..............502 16
............<label-list>
..............44 ,
..............<unsigned-integer>
................503 17
..............<label-list>
................<empty>
..........59 ;
......402 BEGIN
......<statement-list>
........<statement>
..........<unsigned-integer>
............501 15
..........58 :
..........<statement>
............40 (
Parser : Error. Delimiter '$'[3, 14] not found.

Test19:
Line | Column | Ident token | Token
-------------------------------------------------
   1 |      1 |         401 | PROCEDURE
   1 |     11 |        1001 | proc
   1 |     15 |          40 | (
   1 |     16 |        1002 | id1
   1 |     19 |          44 | ,
   1 |     21 |        1003 | id2
   1 |     24 |          44 | ,
   1 |     26 |        1004 | id3
   1 |     29 |          41 | )
   1 |     30 |          59 | ;
   2 |      1 |         404 | LABEL
   2 |      7 |         501 | 15
   2 |      9 |          44 | ,
   2 |     11 |         502 | 16
   2 |     13 |          44 | ,
   2 |     15 |         503 | 17
```

```
    2 |      17 |           59 | ;
    3 |       1 |          402 | BEGIN
    3 |       7 |          501 | 15
    3 |      10 |           58 | :
    3 |      12 |           40 | (
    3 |      13 |           36 | $
    3 |      16 |           36 | $
    3 |      17 |           41 | )
    3 |      18 |           59 | ;
    4 |       1 |          405 | GOTO
    4 |       6 |          502 | 16
    4 |       8 |           59 | ;
    5 |       1 |          406 | RETURN
    5 |       7 |           59 | ;
    6 |       1 |          403 | END
    6 |       4 |           59 | ;

Identifier table
id1 1002
id2 1003
id3 1004
proc 1001

Constant table
15 501
16 502
17 503
```



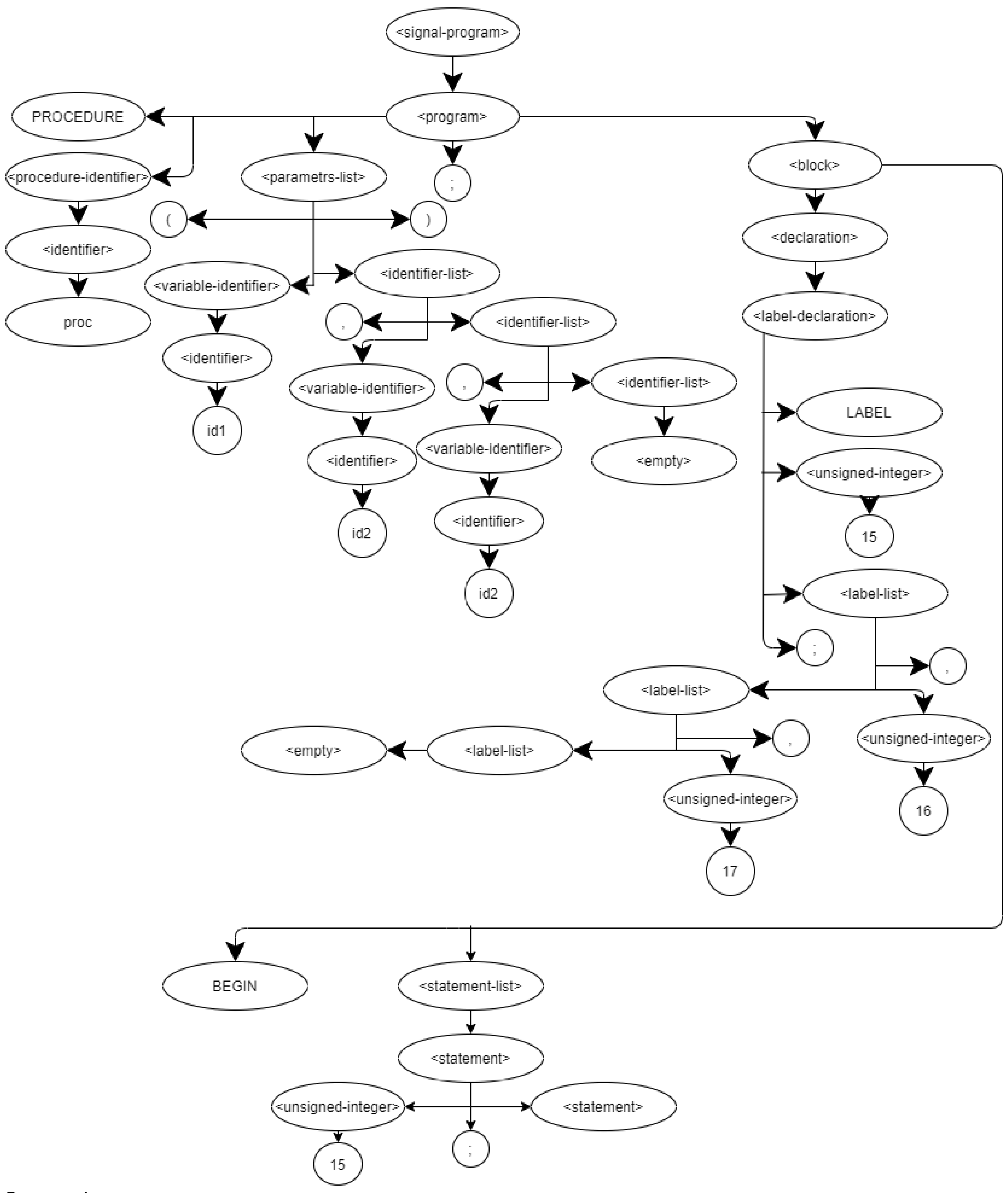
```
Parse tree
<signal-program>
..<program>
....401 PROCEDURE
....<procedure-identifier>
......<identifier>
........1001 proc
....<parameters-list>
......40 (
......<variable-identifier>
........<identifier>
..........1002 id1
```

```
......<identifier-list>
........44 ,
........<variable-identifier>
..........<identifier>
............1003 id2
........<identifier-list>
..........44 ,
..........<variable-identifier>
............<identifier>
..............1004 id3
..........<identifier-list>
............<empty>
......41 )
....59 ;
....<block>
......<declaration>
........<label-declaration>
..........404 LABEL
..........<unsigned-integer>
............501 15
..........<label-list>
............44 ,
............<unsigned-integer>
..............502 16
............<label-list>
..............44 ,
..............<unsigned-integer>
................503 17
..............<label-list>
................<empty>
..........59 ;
......402 BEGIN
......<statement-list>
........<statement>
..........<unsigned-integer>
............501 15
..........58 :
..........<statement>
............40 (
............36 $
............<assembly-insert-file-identifier>
Parser : Error [3, 16]. Identifier not found.

Test20:
Line | Column | Ident token | Token
---------------------------------------------
    1 |      1 |         401 | PROCEDURE
    1 |     11 |        1001 | proc
    2 |      1 |          40 | (
    2 |      2 |        1002 | id1
    2 |      5 |          44 | ,
    2 |      7 |        1003 | id2
    2 |     10 |          44 | ,
    2 |     12 |        1004 | id3
    2 |     15 |          41 | )
    2 |     16 |          59 | ;
    3 |      1 |         404 | LABEL
    3 |      7 |         501 | 15
    3 |      9 |          44 | ,
    3 |     11 |         502 | 16
    3 |     13 |          44 | ,
    3 |     15 |         503 | 17
    3 |     17 |          59 | ;
    4 |      1 |         402 | BEGIN
    4 |      7 |         501 | 15
    4 |     10 |          58 | :
    4 |     12 |          40 | (
    4 |     13 |          36 | $
    4 |     15 |        1005 | asmFile
    4 |     23 |          36 | $
    4 |     24 |          41 | )
    4 |     25 |          59 | ;
    5 |      1 |         405 | GOTO
    5 |      6 |         502 | 16
    5 |      8 |          59 | ;
    6 |      1 |         406 | RETURN
    6 |      7 |          59 | ;
    7 |      1 |         403 | END
    7 |      4 |          59 | ;

Identifier table
asmFile 1005
id1 1002
id2 1003
id3 1004
proc 1001

Constant table
15 501
16 502
17 503
```



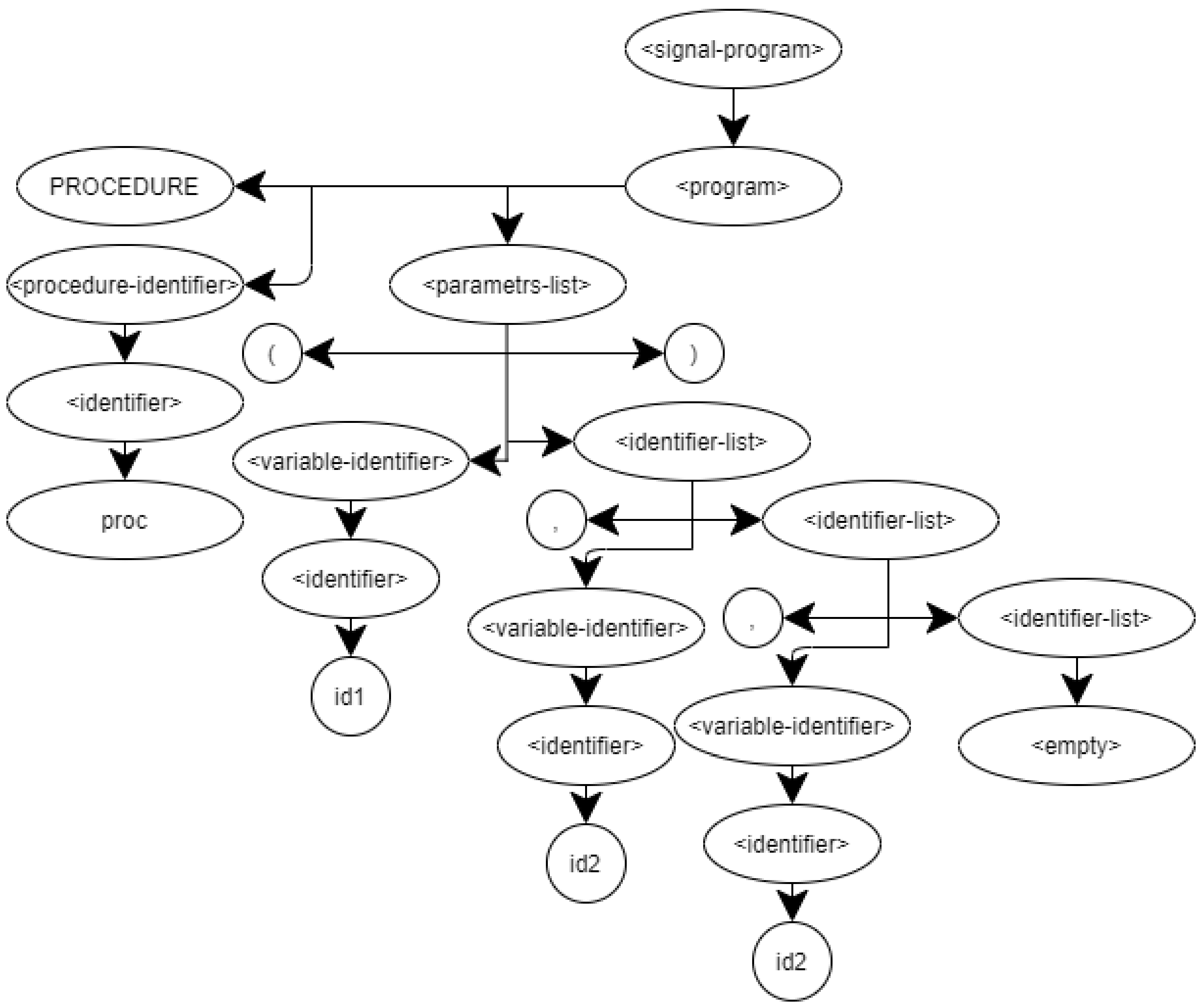
```
Parse tree
<signal-program>
..<program>
....401 PROCEDURE
....<procedure-identifier>
......<identifier>
........1001 proc
....<parameters-list>
```

```
Parser : Error [2, 1]. Tokens must be on the same line.


Test21:
Line | Column | Ident token | Token
-------------------------------------------------
    1 |      1 |         401 | PROCEDURE
    1 |     11 |        1001 | proc
    1 |     15 |          40 | (
    1 |     16 |        1002 | id1
    1 |     19 |          44 | ,
    1 |     21 |        1003 | id2
    1 |     24 |          44 | ,
    1 |     26 |        1004 | id3
    1 |     29 |          41 | )
    1 |     30 |          59 | ;
    2 |      1 |         404 | LABEL
    2 |      7 |         501 | 15
    2 |      9 |          44 | ,
    2 |     11 |         502 | 16
    2 |     13 |          44 | ,
    2 |     15 |         503 | 17
    2 |     17 |          59 | ;
    3 |      1 |         402 | BEGIN
    3 |      7 |         501 | 15
    3 |     10 |          58 | :
    4 |      1 |          40 | (
    4 |      2 |          36 | $
    4 |      4 |        1005 | asmFile
    4 |     12 |          36 | $
    4 |     13 |          41 | )
    4 |     14 |          59 | ;
    5 |      1 |         405 | GOTO
    5 |      6 |         502 | 16
    5 |      8 |          59 | ;
    6 |      1 |         406 | RETURN
    6 |      7 |          59 | ;
    7 |      1 |         403 | END
    7 |      4 |          59 | ;

Identifier table
asmFile 1005
id1 1002
id2 1003
id3 1004
proc 1001

Constant table
15 501
16 502
17 503
```



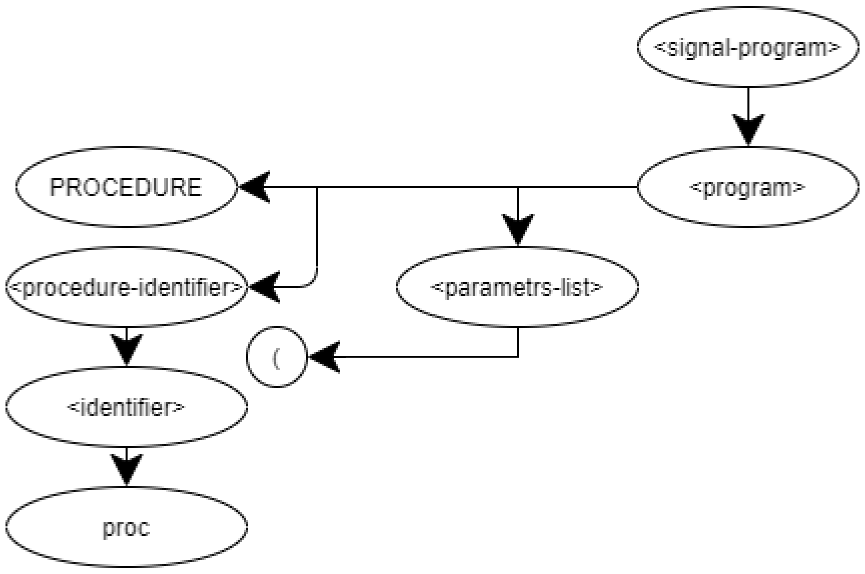
```
Parse tree
<signal-program>
..<program>
....401 PROCEDURE
....<procedure-identifier>
......<identifier>
........1001 proc
....<parameters-list>
......40 (
......<variable-identifier>
........<identifier>
..........1002 id1
......<identifier-list>
........44 ,
........<variable-identifier>
..........<identifier>
............1003 id2
........<identifier-list>
..........44 ,
..........<variable-identifier>
............<identifier>
..............1004 id3
..........<identifier-list>
............<empty>
......41 )
....59 ;
....<block>
......<declaration>
........<label-declaration>
..........404 LABEL
..........<unsigned-integer>
............501 15
..........<label-list>
............44 ,
............<unsigned-integer>
..............502 16
............<label-list>
..............44 ,
..............<unsigned-integer>
................503 17
..............<label-list>
................<empty>
..........59 ;
......402 BEGIN
......<statement-list>
........<statement>
..........<unsigned-integer>
............501 15
..........58 :
..........<statement>
Parser : Error [4, 1]. Tokens must be on the same line.


Test22:
Line | Column | Ident token | Token
-------------------------------------------------
   1 |      1 |         401 | PROCEDURE
   1 |     11 |        1001 | proc
   1 |     15 |          40 | (
   1 |     16 |        1002 | id1
   1 |     19 |          44 | ,
   1 |     21 |        1003 | id2
   1 |     24 |          44 | ,
   1 |     26 |        1004 | id3
   1 |     29 |          41 | )
   2 |      1 |          59 | ;
   3 |      1 |         404 | LABEL
   3 |      7 |         501 | 15
   3 |      9 |          44 | ,
   3 |     11 |         502 | 16
   3 |     13 |          44 | ,
   3 |     15 |         503 | 17
```

```
    3 |      17 |          59 | ;
    4 |       1 |         402 | BEGIN
    4 |       7 |         501 | 15
    4 |      10 |          58 | :
    4 |      12 |          40 | (
    4 |      13 |          36 | $
    4 |      15 |        1005 | asmFile
    4 |      23 |          36 | $
    4 |      24 |          41 | )
    4 |      25 |          59 | ;
    5 |       1 |         405 | GOTO
    5 |       6 |         502 | 16
    5 |       8 |          59 | ;
    6 |       1 |         406 | RETURN
    6 |       7 |          59 | ;
    7 |       1 |         403 | END
    7 |       4 |          59 | ;

Identifier table
asmFile 1005
id1 1002
id2 1003
id3 1004
proc 1001

Constant table
15 501
16 502
17 503
```



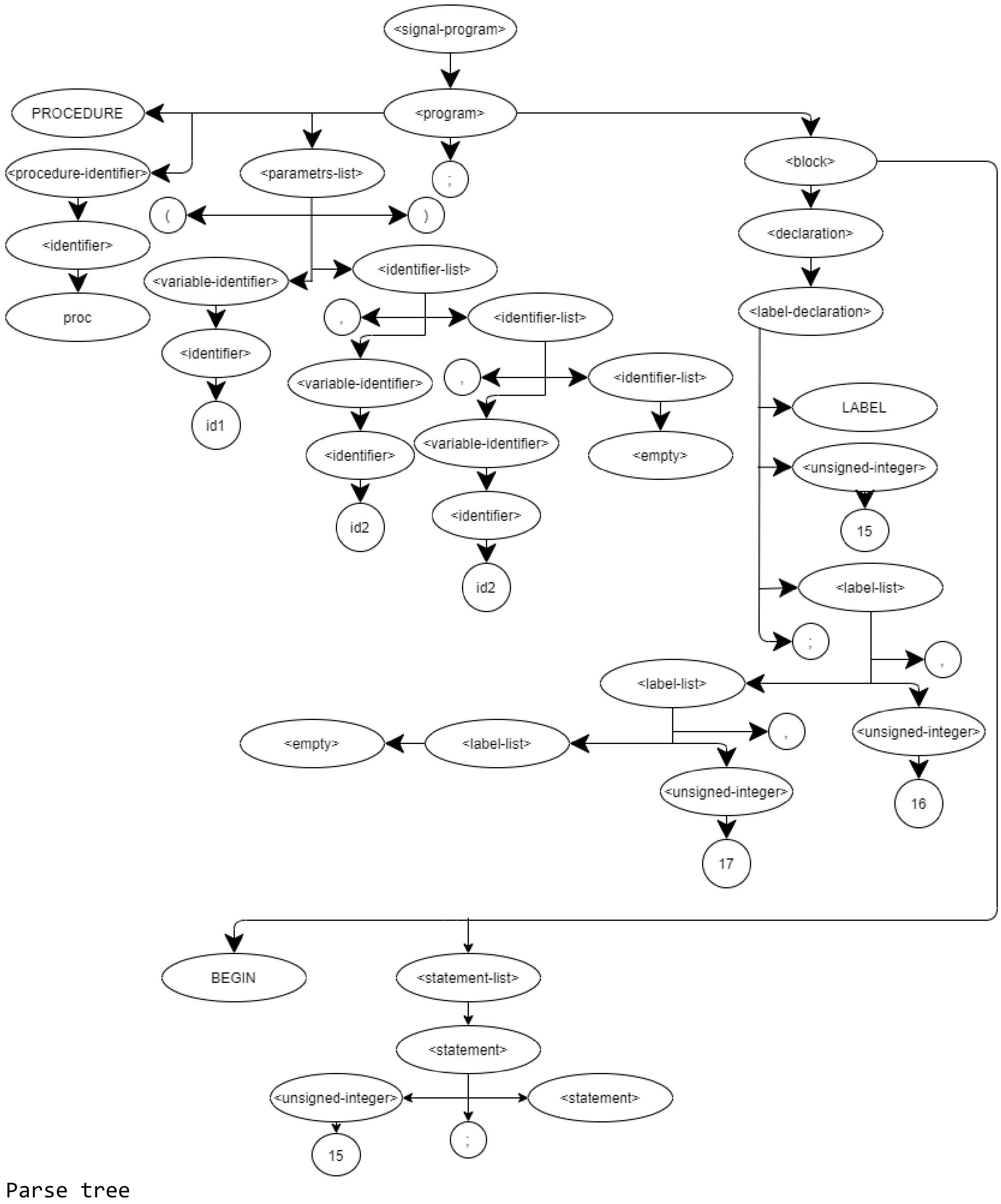
```
Parse tree
<signal-program>
..<program>
....401 PROCEDURE
....<procedure-identifier>
......<identifier>
........1001 proc
....<parameters-list>
......40 (
......<variable-identifier>
........<identifier>
..........1002 id1
......<identifier-list>
........44 ,
........<variable-identifier>
..........<identifier>
............1003 id2
........<identifier-list>
..........44 ,
..........<variable-identifier>
............<identifier>
..............1004 id3
..........<identifier-list>
............<empty>
......41 )
Parser : Error [2, 1]. Tokens must be on the same line.
```

```
Test23:
Line | Column | Ident token | Token
-------------------------------------------------
    1 |      1 |         401 | PROCEDURE
    1 |     11 |        1001 | proc
    1 |     15 |          40 | (
    2 |      1 |        1002 | id1
    2 |      4 |          44 | ,
    2 |      6 |        1003 | id2
    2 |      9 |          44 | ,
    2 |     11 |        1004 | id3
    2 |     14 |          41 | )
    2 |     15 |          59 | ;
    3 |      1 |         404 | LABEL
    3 |      7 |         501 | 15
    3 |      9 |          44 | ,
    3 |     11 |         502 | 16
    3 |     13 |          44 | ,
    3 |     15 |         503 | 17
    3 |     17 |          59 | ;
    4 |      1 |         402 | BEGIN
    4 |      7 |         501 | 15
    4 |     10 |          58 | :
    4 |     12 |          40 | (
    4 |     13 |          36 | $
    4 |     15 |        1005 | asmFile
    4 |     23 |          36 | $
    4 |     24 |          41 | )
    4 |     25 |          59 | ;
    5 |      1 |         405 | GOTO
    5 |      6 |         502 | 16
    5 |      8 |          59 | ;
    6 |      1 |         406 | RETURN
    6 |      7 |          59 | ;
    7 |      1 |         403 | END
    7 |      4 |          59 | ;

Identifier table
asmFile 1005
id1 1002
id2 1003
id3 1004
proc 1001

Constant table
15 501
16 502
17 503
```



```
Parse tree
<signal-program>
..<program>
....401 PROCEDURE
....<procedure-identifier>
......<identifier>
........1001 proc
....<parameters-list>
......40 (
Parser : Error [2, 1]. Tokens must be on the same line.

Test24:
Line | Column | Ident token | Token
--------------------------------------------------
   1 |      1 |         401 | PROCEDURE
   1 |     11 |        1001 | proc
   1 |     15 |          40 | (
   1 |     16 |        1002 | id1
   1 |     19 |          44 | ,
   1 |     21 |        1003 | id2
   1 |     24 |          44 | ,
   1 |     26 |        1004 | id3
   1 |     29 |          41 | )
   1 |     30 |          59 | ;
   2 |      1 |         404 | LABEL
   2 |      7 |         501 | 15
   2 |      9 |          44 | ,
   2 |     11 |         502 | 16
   2 |     13 |          44 | ,
   2 |     15 |         503 | 17
   2 |     17 |          59 | ;
   3 |      1 |         402 | BEGIN
   3 |      7 |         501 | 15
   3 |     10 |          58 | :
   3 |     12 |        1002 | id1
   4 |      1 |         403 | END
   4 |      4 |          59 | ;

Identifier table
id1 1002
id2 1003
```

```
id3 1004
proc 1001

Constant table
15 501
16 502
17 503
```



```
Parse tree
<signal-program>
..<program>
....401 PROCEDURE
....<procedure-identifier>
......<identifier>
........1001 proc
```

```
....<parameters-list>
......40 (
......<variable-identifier>
........<identifier>
..........1002 id1
......<identifier-list>
........44 ,
........<variable-identifier>
..........<identifier>
............1003 id2
........<identifier-list>
..........44 ,
..........<variable-identifier>
............<identifier>
..............1004 id3
..........<identifier-list>
............<empty>
......41 )
....59 ;
....<block>
......<declaration>
........<label-declaration>
..........404 LABEL
..........<unsigned-integer>
............501 15
..........<label-list>
............44 ,
............<unsigned-integer>
..............502 16
............<label-list>
..............44 ,
..............<unsigned-integer>
................503 17
..............<label-list>
................<empty>
..........59 ;
......402 BEGIN
......<statement-list>
........<statement>
..........<unsigned-integer>
............501 15
..........58 :
..........<statement>
Parser : Error [3, 12]. After the mark should be statement.
```