НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. Ігоря СІКОРСЬКОГО»
ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ
Кафедра системного програмування і спеціалізованих комп'ютерних систем

Лабораторна робота №1

з дисципліни

«Основи проектування трансляторів»

Тема: «Розробка лексичного аналізатора»

Виконав: студент III курсу
ФПМ групи КВ-01
Шкільнюк В. О.
Перевірив(ла):

Київ 2023

Постановка задачі:

1. Розробити програму лексичного аналізатора (ЛА) для підмно-жини мови програмування SIGNAL.

2. Лексичний аналізатор має забезпечувати наступні дії:

• видалення (пропускання) пробільних символів: пробіл (код ASCII 32), повернення каретки (код ASCII 13); перехід на новий ря-док (код ASCII 10), горизонтальна та вертикальна табуляція (коди ASCII 9 та 11), перехід на нову сторінку (код ASCII 12);

• згортання ключових слів;

• згортання багато-символьних роздільників (якщо передбача-ються граматикою варіанту);

• згортання констант із занесенням до таблиці значення та типу константи (якщо передбачаються граматикою варіанту);

• згортання ідентифікаторів;
• видалення коментарів, заданих у вигляді **(*** <текст комен-таря> ***)**;

• формування рядка лексем зінформацією про позиції лексем;

• заповнення таблиць ідентифікаторів та констант інформацією, отриманою підчас згортки лексем;

• виведення повідомлень про помилки.

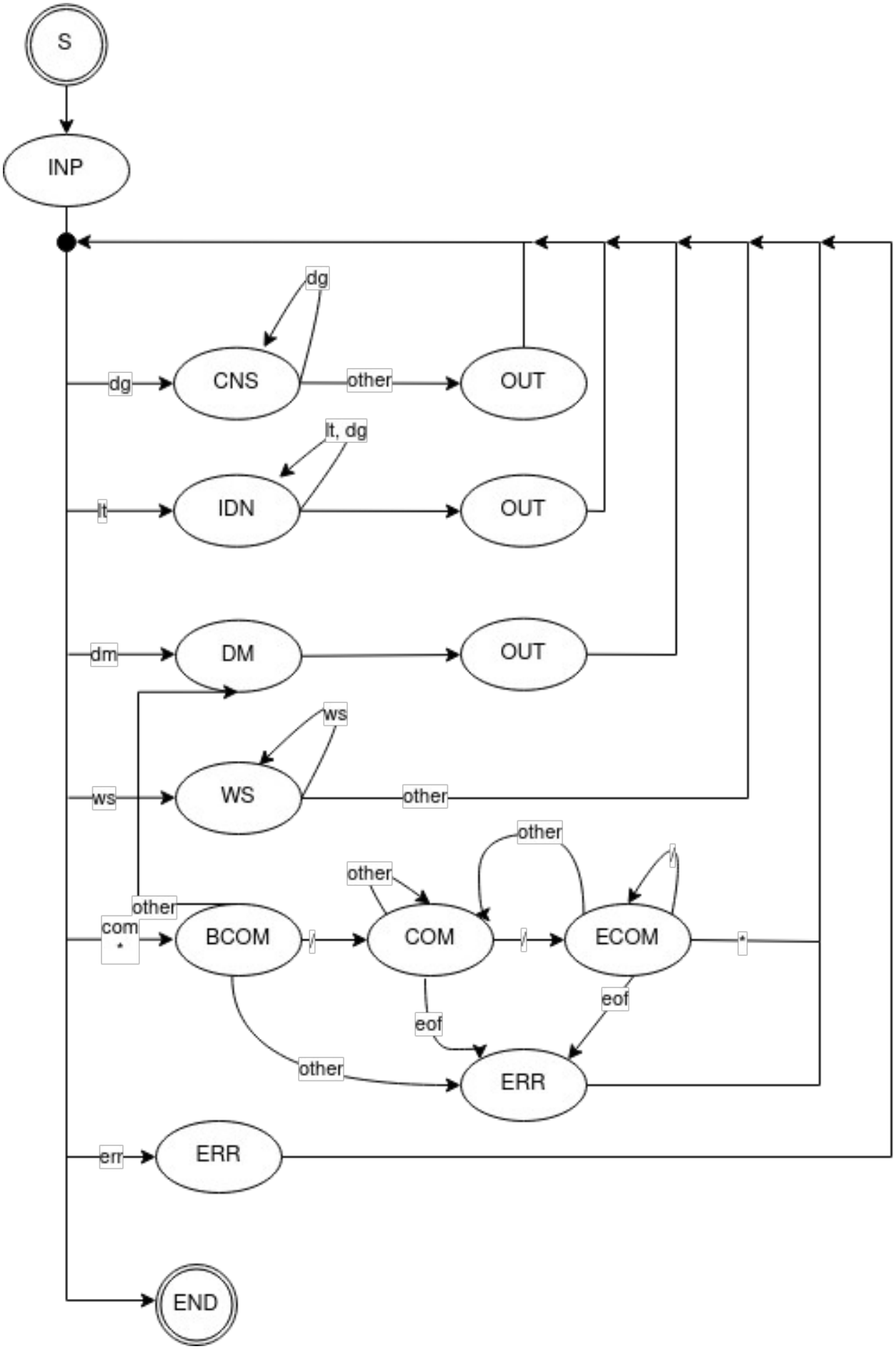
Завдання за варіантом 18:

Граматика підмножини мови програмування SIGNAL:

Варіант 18

```
1.    <signal-program> --> <program>
2.    <program> --> PROGRAM <procedure-identifier> ;
                    <block>.
3.    <block> --> BEGIN <statements-list> END
4.    <statements-list> --> <statement> <statements-list>
                    |
                    <empty>
5.    <statement> --> LOOP <statements-list> ENDLOOP ; |
                    CASE <expression> OF <alternatives-list>
                    ENDCASE ;
6.    <alternatives-list> --> <alternative> <alternatives-
                    list> |
                    <empty>
7.    <alternative> --> <expression> : / <statements-list>
                    \
8.    <expression> --> <multiplier><multipliers-list>
9.    <multipliers-list> --> <multiplication-instruction>
                    <multiplier><multipliers-list>  |
                    <empty>
10.   <multiplication-instruction> -->     *   |
                    /   |
                    MOD
11.   <multiplier> --> <variable-identifier> |
                    <unsigned-integer>
12.   <variable-identifier> --> <identifier>
13.   <procedure-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
```

Діаграма переходів ЛА:



Лістинг програми мовою C++

main.cpp

```
#include <iostream>
#include "tables.h"

int main() {
	TokenAnalyser t("program.txt");
	t.init();
	t.analyze();
	return 0;
}
```

analyser.h

```
#ifndef __ANALYSER_H__
#define __ANALYSER_H__

#include <iostream>
#include <string>
#include <vector>
#include <map>
#include <fstream>

class TokenAnalyser {
public:
	TokenAnalyser() = default;
	TokenAnalyser(std::string fileName);
	void init();
	std::pair<char, int> getChar();
	void writeToken(std::string token, int tokenCode, bool delimiter = false);
	void writeError(std::string);
	int idnTabSearch(std::string);
	int kTabSearch(std::string);
	int constTabSearch(std::string);
	void analyze();
private:
	int findMax(std::string);
	enum tokenValues {
		WS,
		CNS,
		IDN,
		DM1,
		DM2,
		ERR
	};
	enum keywordTokenValues {
		PROGRAM = 401,
		END,
		LOOP,
		ENDLOOP,
```

```
            CASE,
            OF,
            ENDCASE,
            MOD,
            BEGIN
      };
      std::vector<tokenValues> tokens{128, tokenValues::ERR};
      std::map<std::string, keywordTokenValues> kTokens;
      std::map<std::string, int> idnTokens;
      std::map<std::string, int> constTokens;
      std::string initialProgram;
      std::ifstream program;
      std::ofstream outProgram;
      int row;
      int column;
};

#endif
```

analyser.cpp

```
#include "analyser.h"

#include <memory>


void TokenAnalyser::init() {
for(int i = 0; i < 128; i++) {
tokens.at(i) = tokenValues::ERR;
}
for(int i = 8; i < 16; i++) {
tokens.at(i) = tokenValues::WS;
}
tokens.at(32) = tokenValues::WS;
for(int i = 48; i < 58; i++) {
tokens.at(i) = tokenValues::CNS; // digits
}

for(int i = 65; i < 91; i++) {
tokens.at(i) = tokenValues::IDN; // letters
}

tokens.at(42) = tokenValues::DM1; // *
tokens.at(58) = tokenValues::DM2; // :
tokens.at(59) = tokenValues::DM2; // ;
tokens.at(46) = tokenValues::DM2; // .
kTokens["PROGRAM"] = keywordTokenValues::PROGRAM;
kTokens["BEGIN"] = keywordTokenValues::BEGIN;
kTokens["END"] = keywordTokenValues::END;
kTokens["LOOP"] = keywordTokenValues::LOOP;
kTokens["ENDLOOP"] = keywordTokenValues::ENDLOOP;
kTokens["CASE"] = keywordTokenValues::CASE;
kTokens["OF"] = keywordTokenValues::OF;
```

```
kTokens["ENDCASE"] = keywordTokenValues::ENDCASE;
kTokens["MOD"] = keywordTokenValues::MOD;
}

TokenAnalyser::TokenAnalyser(std::string fileName) {
std::ifstream inputFile;
program.open(std::string("../") + fileName);
outProgram.open(std::string("../") + "out.txt");
if(!program.is_open()) {
writeError("Could not load a program file");
}
if(!outProgram.is_open()) {
writeError("Could not load an output file");
}
char c;
row = 1;
column = 0;
}

std::pair<char, int> TokenAnalyser::getChar() {
if(program.is_open()) {
char c;
if(program.get(c)) {
column++;
return std::pair<char, int>(c, tokens.at(c));
}
else {
return {-1, -1};
}
}
}

void TokenAnalyser::writeToken(std::string token, int tokenCode, bool delimiter)
{
outProgram << row << "\t\t" << column - token.length() << "\t\t" << tokenCode <<
"\t" << token << std::endl;
}

void TokenAnalyser::writeError(std::string error) {
outProgram << row << "\t\t" << column << "\t\t" << error << std::endl;
}

int TokenAnalyser::findMax(std::string tokenMap) {
int value = -1;
std::shared_ptr<std::map<std::string, int>> tMap = nullptr;
if(tokenMap == "idnTokens") {
tMap = std::make_shared<std::map<std::string, int>>(idnTokens);
value = 1000;
}
else {
tMap = std::make_shared<std::map<std::string, int>>(constTokens);
```

```
value = 500;
}
if(tMap->empty()) {
return value;
}
int max = value;
for(const auto& kv : *tMap) {
if(kv.second > max) {
max = kv.second;
}
}

return max;
}

int TokenAnalyser::idnTabSearch(std::string keyword) {
if(idnTokens.find(keyword) == idnTokens.end()) {
idnTokens[keyword] = findMax("idnTokens") + 1;
}
return idnTokens[keyword];
}

int TokenAnalyser::kTabSearch(std::string keyword) {
if(kTokens.find(keyword) == kTokens.end()) {
return -1;
}
return kTokens[keyword];
}

int TokenAnalyser::constTabSearch(std::string keyword) {
if(constTokens.find(keyword) == constTokens.end()) {
constTokens[keyword] = findMax("constTokens") + 1;
}
return constTokens[keyword];
}

void TokenAnalyser::analyze() {
auto symbol = getChar();
while(!program.eof()) {
std::string buffer = std::string("");
bool supressOutput = false;

switch(symbol.second) {
case 0:
while(!program.eof()) {
if(symbol.first == '\n') {
row++;
column = 0;
}
symbol = getChar();
if(symbol.second > 0 || symbol.second < 0) {
```

```
break;
}
}
break;
case 1:
while(!program.eof() && symbol.second == 1) {
buffer += symbol.first;
symbol = getChar();
}
writeToken(buffer, constTabSearch(buffer));
break;
case 2: {
while(!program.eof() && symbol.second == 2 || symbol.second == 1) {
buffer += symbol.first;
symbol = getChar();
}
int tokenCode = kTabSearch(buffer);
if(tokenCode == -1) {
tokenCode = idnTabSearch(buffer);
}
writeToken(buffer, tokenCode);
break;
}
case 3: {
if(program.eof()) {
std::string token = "";
token += symbol.first;
writeToken(token, symbol.first);
}
else {
symbol = getChar();
if(symbol.first == '/') {
if(program.eof()) {
writeError("Expected / but end of file found");
}
else {
symbol = getChar();
do {
while(!program.eof() && symbol.first != '/') {
symbol = getChar();
}
if(program.eof()) {
writeError("Expected / but end of file found");
symbol.first = '+';
break;
}
else {
symbol = getChar();
}
}while(symbol.first != '*');
if(!program.eof()) {
```

```
symbol = getChar();
}
}
}
else {
writeToken("*", '*');
}
}
break;
}
case 4: {
std::string delimiterToString = "";
char delimiterToken = symbol.first;
delimiterToString += delimiterToken;
symbol = getChar();
writeToken(delimiterToString, delimiterToken);
break;
}
case 5:
writeError("Illegal symbol");
symbol = getChar();
}
}
program.close();
outProgram.close();
}
```

Тестування програми:

main.cpp | program.txt м | analyser.h

program.txt

```
PROGRAM MAIN ;
BEGIN
    LOOP
        CASE VARIABLE OF
            VARIABLE1 MOD 411 : VARIABLE * 22;
            VARIABLE2 * 2 : VARIABLE2 MOD 1;
            VARIABLE3 MOD 51 : VARIABLE2 * 2;
        ENDCASE;
    ENDLOOP;
END.
```

analyser.cpp | analyser.h | CMakeLists.txt | out.txt u

out.txt

```
1   1   401 PROGRAM
1   9   1001    MAIN
1   14  59  ;
2   1   409 BEGIN
3   5   403 LOOP
4   9   405 CASE
4   14  Illegal symbol
4   15  1002    ARIABLE
4   23  406 OF
5   13  1003    VARIABLE1
5   23  Illegal symbol
5   24  1004    OD
5   27  501 411
5   31  58  :
5   33  1005    VARIABLE
5   42  42  *
5   44  502 22
5   46  59  ;
6   13  1006    VARIABLE2
6   23  42  *
6   25  503 2
6   27  58  :
6   29  1006    VARIABLE2
6   39  408 MOD
6   43  504 1
6   44  59  ;
7   13  1007    VARIABLE3
7   23  408 MOD
7   27  505 51
7   30  58  :
7   32  1006    VARIABLE2
7   42  42  *
7   44  503 2
7   45  59  ;
8   9   407 ENDCASE
8   16  59  ;
9   5   404 ENDLOOP
9   12  59  ;
10  1   402 END
10  3   46  .
```

main.cpp | program.txt м | analyser.h

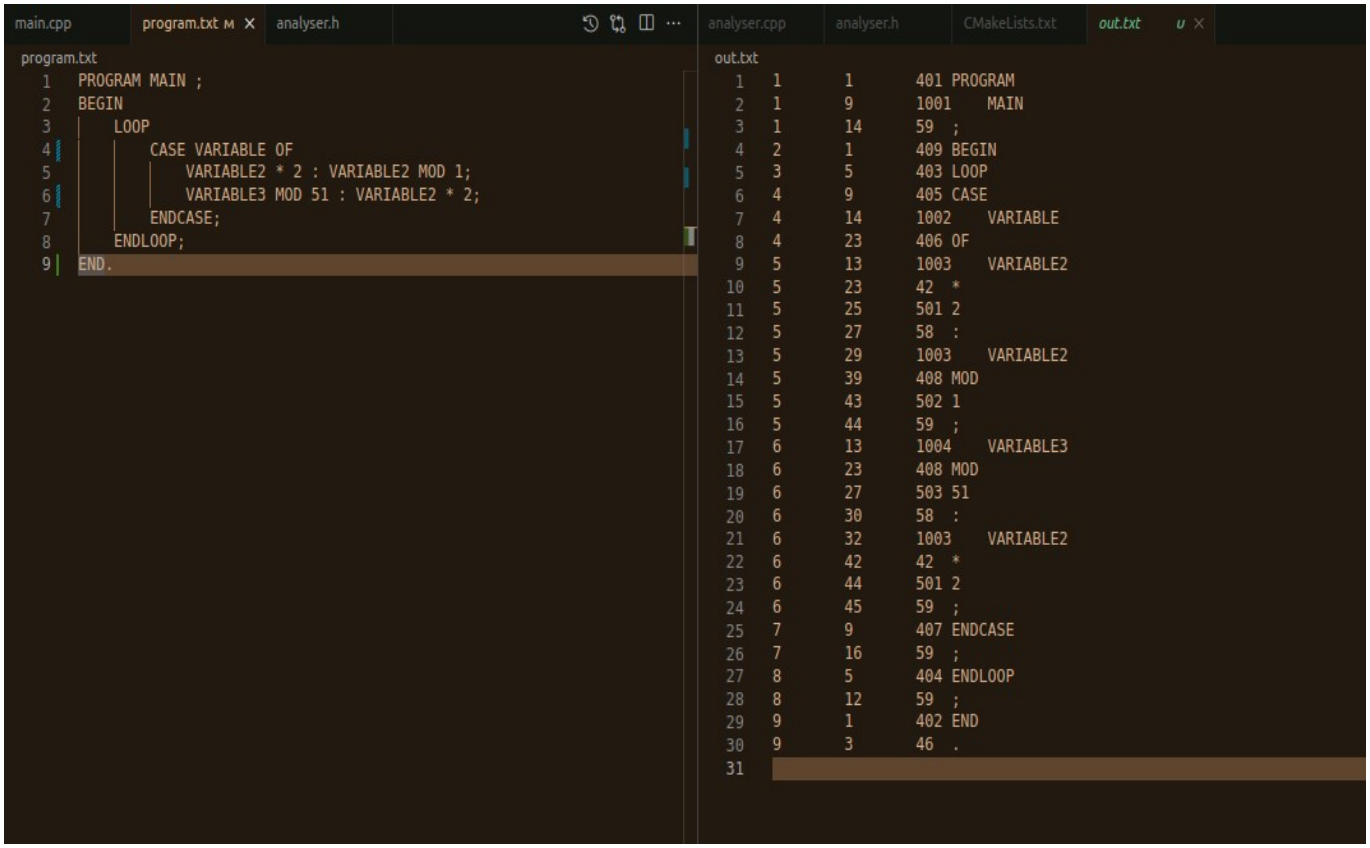
program.txt

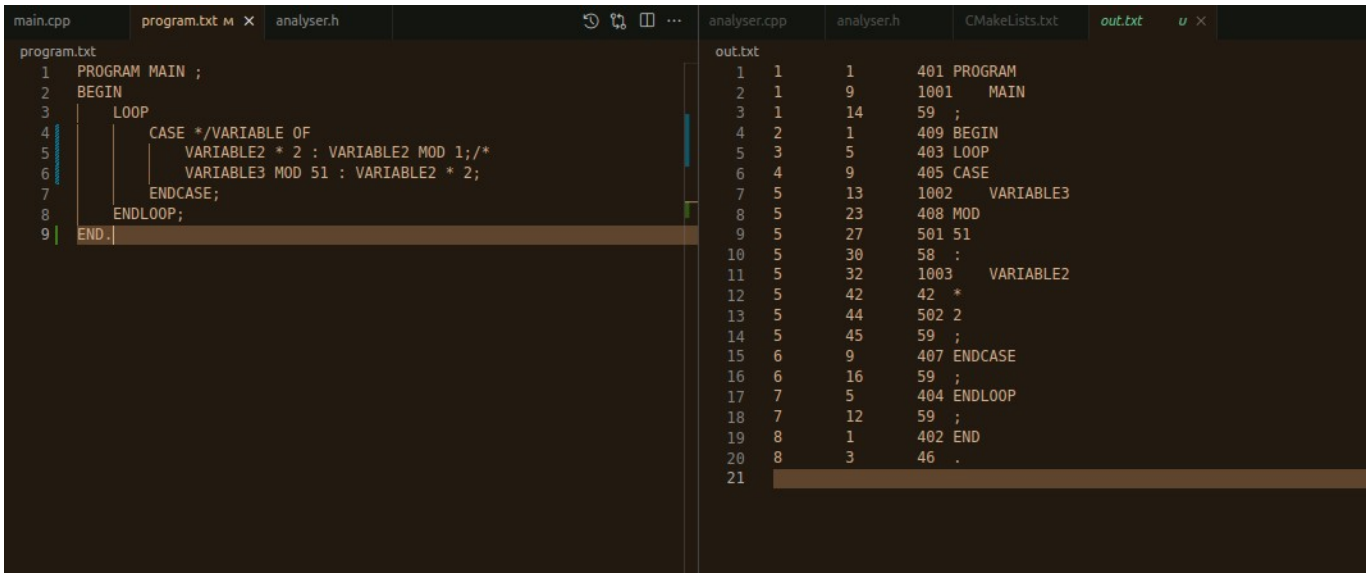
```
PROGRAM MAIN ;
BEGIN
    LOOP
        CASE */comment/*VARIABLE OF
            VARIABLE1 MOD 411 : VARIABLE * 22;
            VARIABLE2 * 2 : VARIABLE2 MOD 1;
            VARIABLE3 MOD 51 : VARIABLE2 * 2;
        ENDCASE;
    ENDLOOP;
END.
```

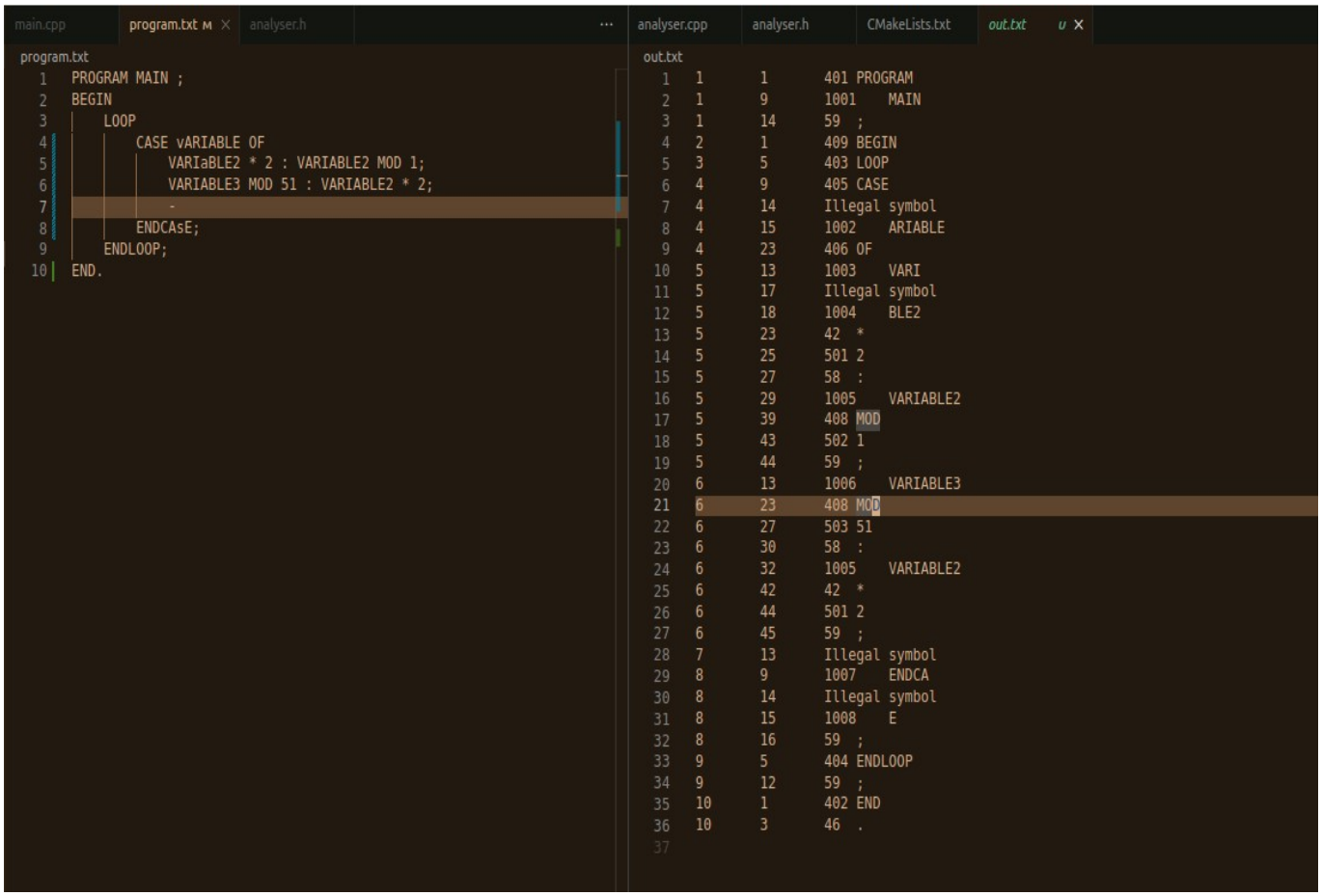
analyser.cpp | analyser.h | CMakeLists.txt | out.txt u

out.txt

```
1   1   401 PROGRAM
1   9   1001    MAIN
1   14  59  ;
2   1   409 BEGIN
3   5   403 LOOP
4   9   405 CASE
4   25  1002    VARIABLE
4   34  406 OF
5   13  1003    VARIABLE1
5   23  408 MOD
5   27  501 411
5   31  58  :
5   33  1002    VARIABLE
5   42  42  *
5   44  502 22
5   46  59  ;
6   13  1004    VARIABLE2
6   23  42  *
6   25  503 2
6   27  58  :
6   29  1004    VARIABLE2
6   39  408 MOD
6   43  504 1
6   44  59  ;
7   13  1005    VARIABLE3
7   23  408 MOD
7   27  505 51
7   30  58  :
7   32  1004    VARIABLE2
7   42  42  *
7   44  503 2
7   45  59  ;
8   9   407 ENDCASE
8   16  59  ;
9   5   404 ENDLOOP
9   12  59  ;
10  1   402 END
10  3   46  .
```


main.cpp
program.txt M
analyser.h
program.txt
PROGRAM MAIN ;
BEGIN
LOOP
CASE VARIABLE OF
VARIABLE2 * 2 : VARIABLE2 MOD 1;
VARIABLE3 MOD 51 : VARIABLE2 * 2;
ENDCASE;
ENDLOOP;
END.
analyser.cpp
analyser.h
CMakeLists.txt
out.txt U
out.txt
1 1 401 PROGRAM
1 9 1001 MAIN
1 14 59 ;
2 1 409 BEGIN
3 5 403 LOOP
4 9 405 CASE
4 14 1002 VARIABLE
4 23 406 OF
5 13 1003 VARIABLE2
5 23 42 *
5 25 501 2
5 27 58 :
5 29 1003 VARIABLE2
5 39 408 MOD
5 43 502 1
5 44 59 ;
6 13 1004 VARIABLE3
6 23 408 MOD
6 27 503 51
6 30 58 :
6 32 1003 VARIABLE2
6 42 42 *
6 44 501 2
6 45 59 ;
7 9 407 ENDCASE
7 16 59 ;
8 5 404 ENDLOOP
8 12 59 ;
9 1 402 END
9 3 46 .


main.cpp
program.txt M
analyser.h
program.txt
PROGRAM MAIN ;
BEGIN
LOOP
CASE */VARIABLE OF
VARIABLE2 * 2 : VARIABLE2 MOD 1;/*
VARIABLE3 MOD 51 : VARIABLE2 * 2;
ENDCASE;
ENDLOOP;
END.
analyser.cpp
analyser.h
CMakeLists.txt
out.txt U
out.txt
1 1 401 PROGRAM
1 9 1001 MAIN
1 14 59 ;
2 1 409 BEGIN
3 5 403 LOOP
4 9 405 CASE
5 13 1002 VARIABLE3
5 23 408 MOD
5 27 501 51
5 30 58 :
5 32 1003 VARIABLE2
5 42 42 *
5 44 502 2
5 45 59 ;
6 9 407 ENDCASE
6 16 59 ;
7 5 404 ENDLOOP
7 12 59 ;
8 1 402 END
8 3 46 .

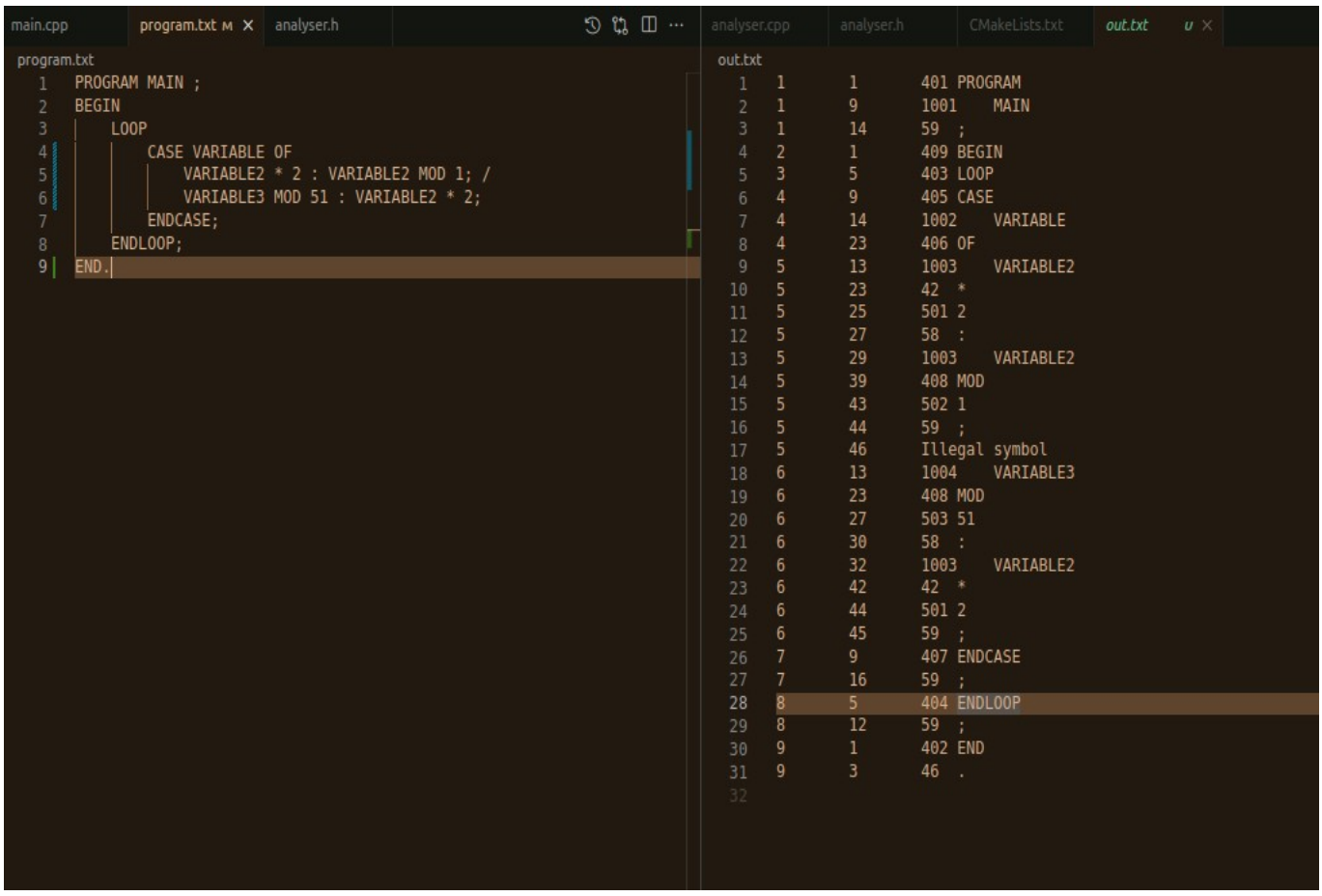


program.txt

```
PROGRAM MAIN ;
BEGIN
    LOOP
        CASE vARIABLE OF
            VARIaBLE2 * 2 : VARIABLE2 MOD 1;
            VARIABLE3 MOD 51 : VARIABLE2 * 2;
            -
        ENDCAsE;
    ENDLOOP;
END.
```

out.txt

```
1   1   401 PROGRAM
1   9   1001    MAIN
1   14  59  ;
2   1   409 BEGIN
3   5   403 LOOP
4   9   405 CASE
4   14  Illegal symbol
4   15  1002    ARIABLE
4   23  406 OF
5   13  1003    VARI
5   17  Illegal symbol
5   18  1004    BLE2
5   23  42  *
5   25  501 2
5   27  58  :
5   29  1005    VARIABLE2
5   39  408 MOD
5   43  502 1
5   44  59  ;
6   13  1006    VARIABLE3
6   23  408 MOD
6   27  503 51
6   30  58  :
6   32  1005    VARIABLE2
6   42  42  *
6   44  501 2
6   45  59  ;
7   13  Illegal symbol
8   9   1007    ENDCA
8   14  Illegal symbol
8   15  1008    E
8   16  59  ;
9   5   404 ENDLOOP
9   12  59  ;
10  1   402 END
10  3   46  .
```

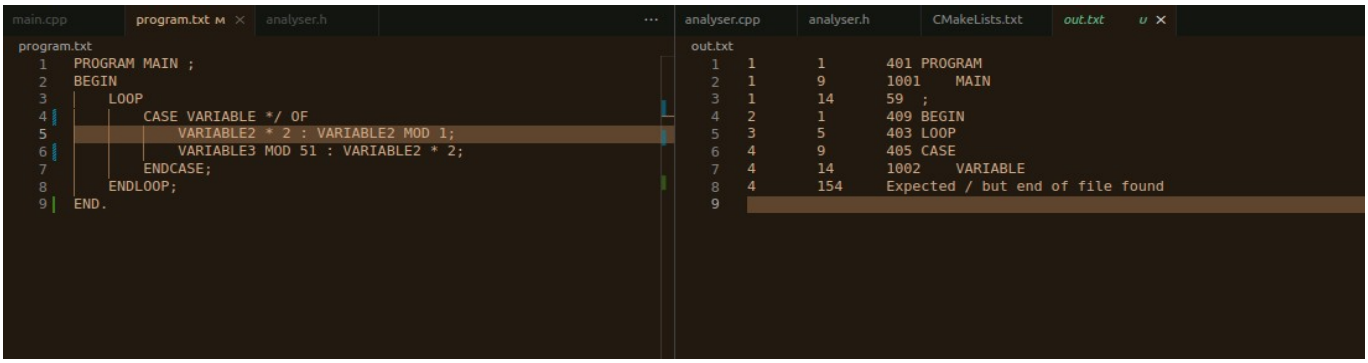


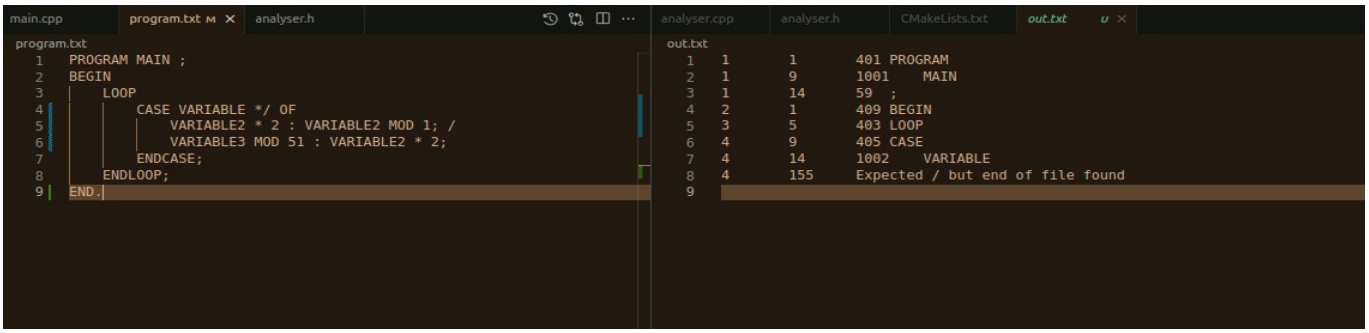
program.txt

```
PROGRAM MAIN ;
BEGIN
    LOOP
        CASE VARIABLE OF
            VARIABLE2 * 2 : VARIABLE2 MOD 1; /
            VARIABLE3 MOD 51 : VARIABLE2 * 2;
        ENDCASE;
    ENDLOOP;
END.
```

out.txt

```
1   1   401 PROGRAM
1   9   1001    MAIN
1   14  59  ;
2   1   409 BEGIN
3   5   403 LOOP
4   9   405 CASE
4   14  1002    VARIABLE
4   23  406 OF
5   13  1003    VARIABLE2
5   23  42  *
5   25  501 2
5   27  58  :
5   29  1003    VARIABLE2
5   39  408 MOD
5   43  502 1
5   44  59  ;
5   46  Illegal symbol
6   13  1004    VARIABLE3
6   23  408 MOD
6   27  503 51
6   30  58  :
6   32  1003    VARIABLE2
6   42  42  *
6   44  501 2
6   45  59  ;
7   9   407 ENDCASE
7   16  59  ;
8   5   404 ENDLOOP
8   12  59  ;
9   1   402 END
9   3   46  .
```


main.cpp
program.txt M
analyser.h
program.txt
PROGRAM MAIN ;
BEGIN
LOOP
CASE VARIABLE */ OF
VARIABLE2 * 2 : VARIABLE2 MOD 1;
VARIABLE3 MOD 51 : VARIABLE2 * 2;
ENDCASE;
ENDLOOP;
END.
analyser.cpp
analyser.h
CMakeLists.txt
out.txt U
out.txt
1 1 401 PROGRAM
1 9 1001 MAIN
1 14 59 ;
2 1 409 BEGIN
3 5 403 LOOP
4 9 405 CASE
4 14 1002 VARIABLE
4 154 Expected / but end of file found


main.cpp
program.txt M
analyser.h
program.txt
PROGRAM MAIN ;
BEGIN
LOOP
CASE VARIABLE */ OF
VARIABLE2 * 2 : VARIABLE2 MOD 1; /
VARIABLE3 MOD 51 : VARIABLE2 * 2;
ENDCASE;
ENDLOOP;
END.
analyser.cpp
analyser.h
CMakeLists.txt
out.txt U
out.txt
1 1 401 PROGRAM
1 9 1001 MAIN
1 14 59 ;
2 1 409 BEGIN
3 5 403 LOOP
4 9 405 CASE
4 14 1002 VARIABLE
4 155 Expected / but end of file found