****

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «САМАРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИМЕНИ АКАДЕМИКА С.П. КОРОЛЕВА  
(САМАРСКИЙ УНИВЕРСИТЕТ)»**

Институт (факультет) информатики и кибернетики

Кафедра информационных систем и технологий

**ОТЧЁТ**

**по лабораторной работе**

Синтаксический анализ автоматных языков  
по дисциплине «Теория автоматов и формальных языков»

Вариант 15

Выполнил Бренева Вероника Андреевна, группа 6201-020302D

Проверила Чигарина Елена Ивановна

Самара  
2024

ЗАДАНИЕ

Написать программу синтаксического анализа автоматного языка с включением семантики, оператора заголовка цикла с предусловием языка QBasic, имеющего вид:

**DO WHILE** <логическое выражение>

<логическое выражение> :: = <операнд>[<логическая операция><операнд>]

<операнд> :: = <терм><операция отношения><терм>

<терм> :: = <идентификатор>|<константа>

<операция отношения> :: = < |> |=

<логическая операция> :: = AND | OR | XOR

<оператор присваивания> :: = <идентификатор>**=**<константа>

<идентификатор> - идентификатор, начинается с буквы, включает последовательность букв, цифр,

<константа > - целое число или число с фиксированной точкой с учетом отсутствия незначащих нулей.

Семантика:

Идентификатор имеет ограничение на длину (не более 8 символов) и не может быть зарезервированным словом (DO, WHILE, LOOP, OR, AND, XOR);

Целое число находится в диапазоне -32768 ÷ +32767

Построить и вывести на печать неповторяющийся список идентификаторов и констант.

Сообщать об ошибках при анализе цепочек языка, указывая курсором место ошибки и ее содержание.

Анализатор работает до первой ошибки, допускает произвольное число пробелов между конструкциями оператора (пробелы могут и отсутствовать), не учитывается регистр символов.

Примеры правильных цепочек:

DO WHILE A > B AND C = D

DO WHILE 12.3 > C

DO WHILE 12.3 = 100 XOR 7 > - 3.098

ГРАФЫ СОСТОЯНИЙ АНАЛИЗАТОРА

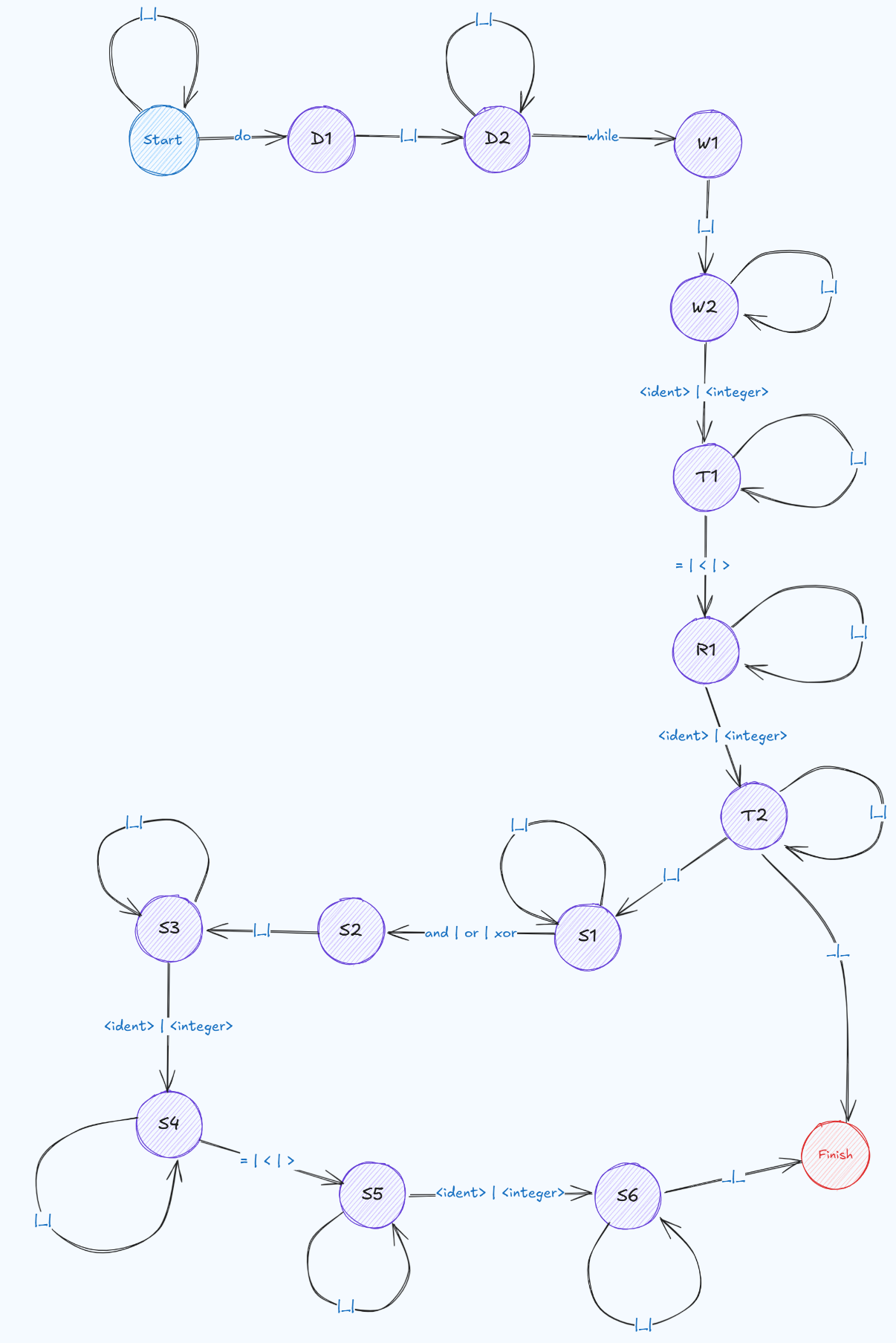


Рисунок 1 – Основной граф состояний анализатора

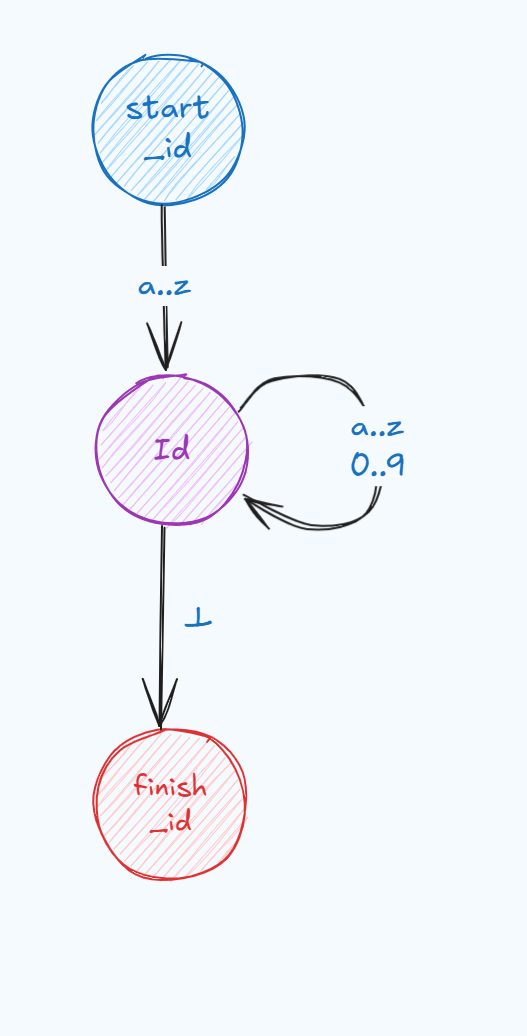


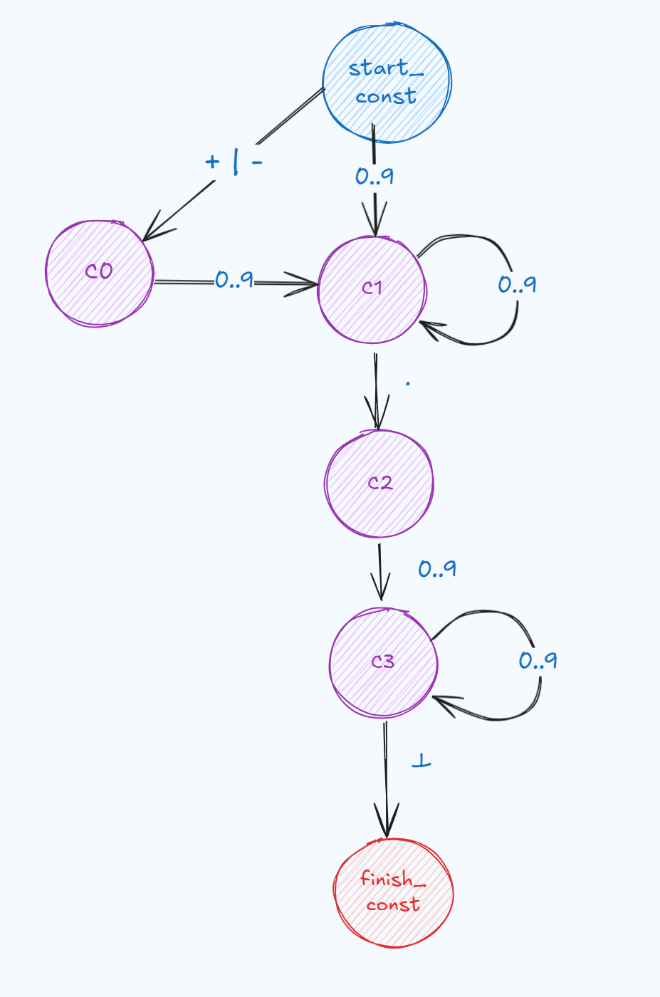
Рисунок 2 – Граф состояний анализатора идентификатора

Рисунок 3 – Граф состояний анализатора константы

ПРИМЕРЫ РАБОТЫ АНАЛИЗАТОРА

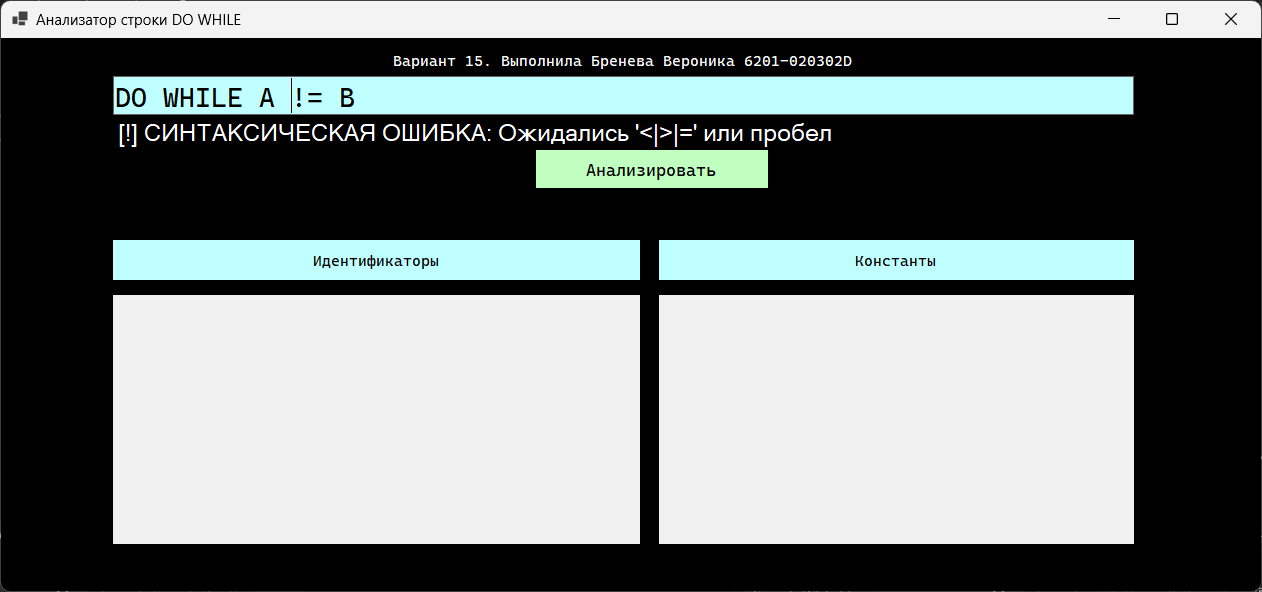


Рисунок 1 – Синтаксическая ошибка №1

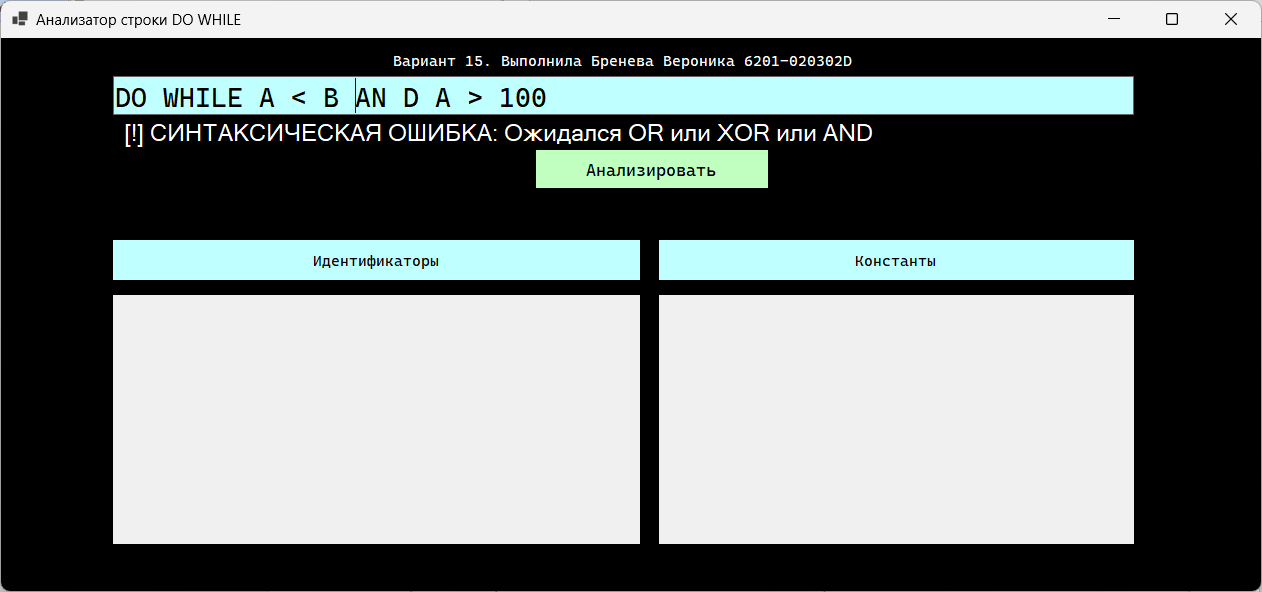


Рисунок 2 – Синтаксическая ошибка №2

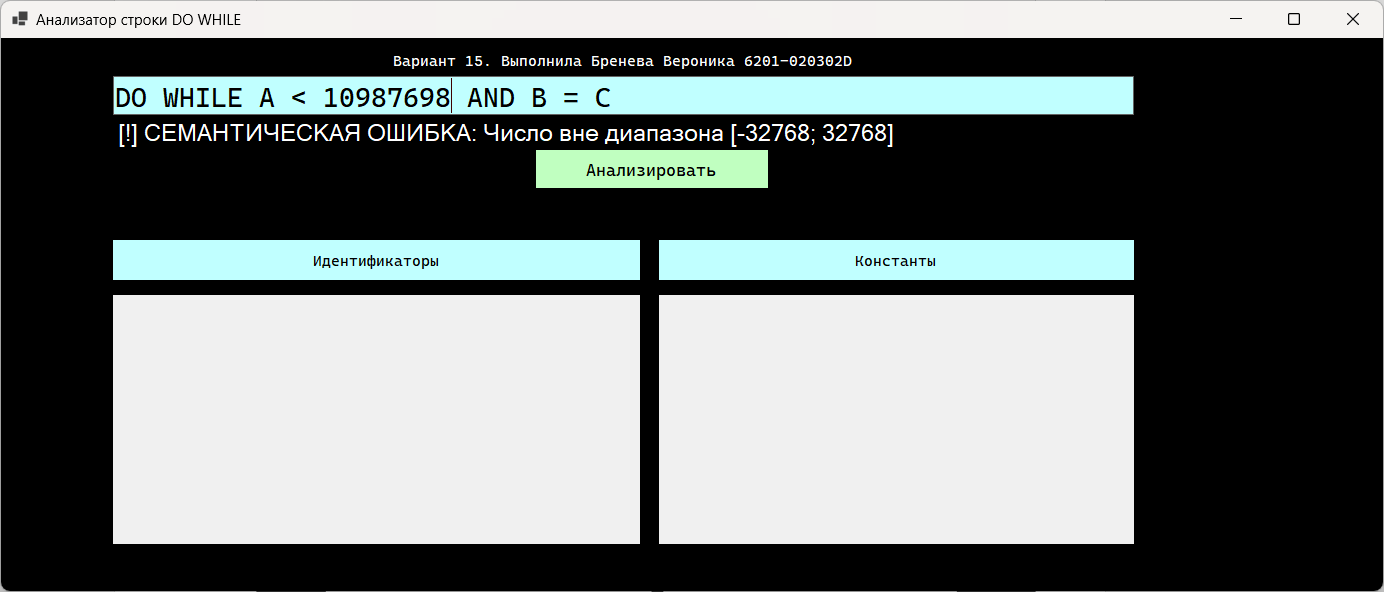


Рисунок 3 – Семантическая ошибка №1

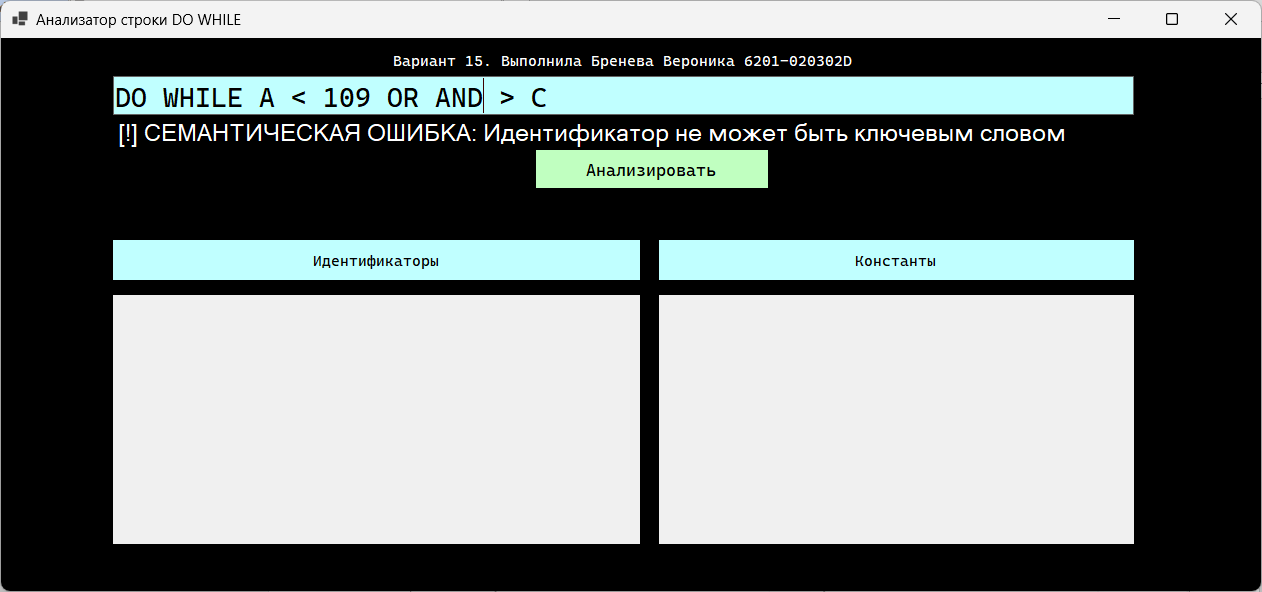


Рисунок 4 – Семантическая ошибка №2

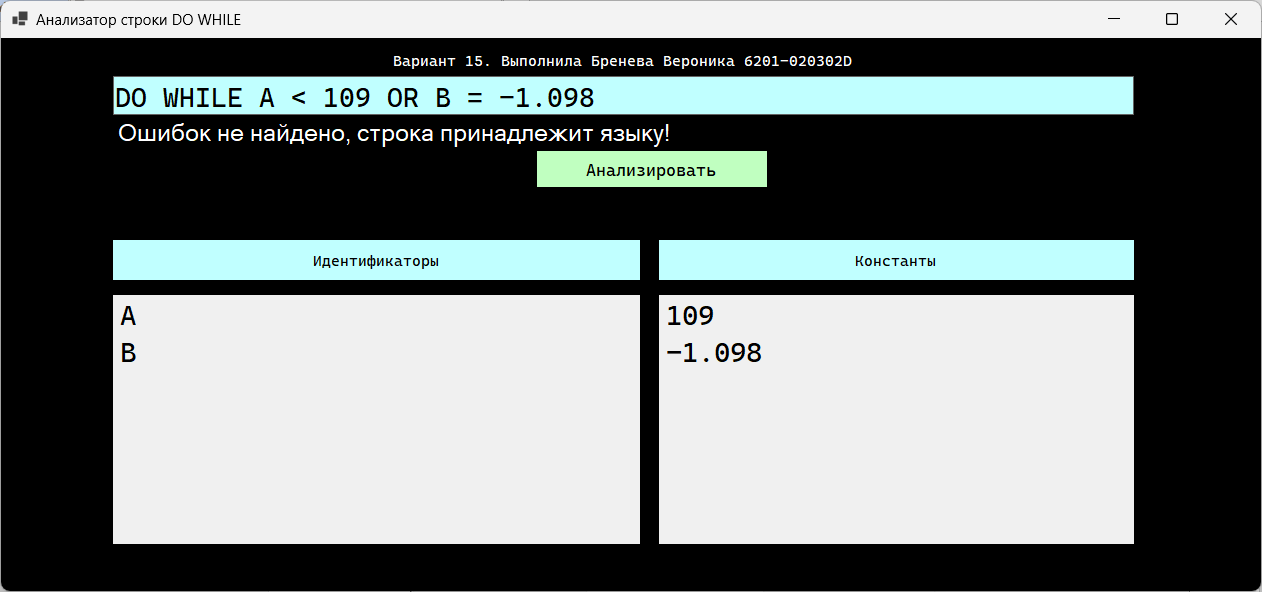


Рисунок 5 – Вывод семантики 1

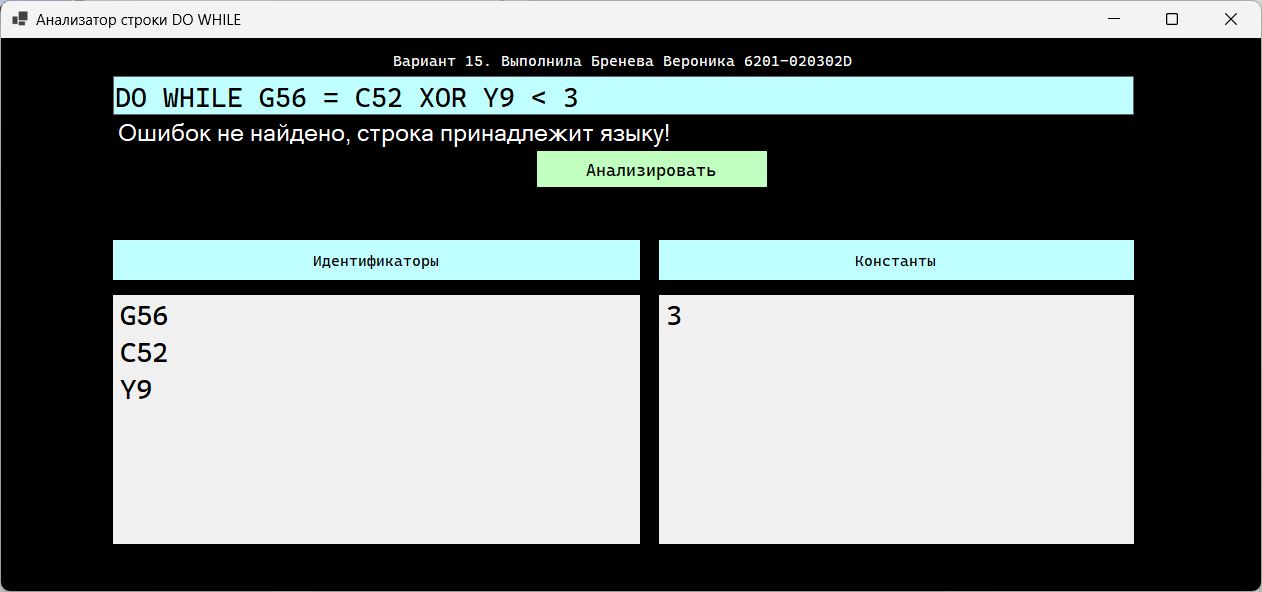


Рисунок 6 – Вывод семантики 2

ЛИСТИНГ ОСНОВНЫХ МОДУЛЕЙ АНАЛИЗАТОРА

Класс «Анализатор состояний»:

Поля класса:

* текущее состояние анализатора;
* исходная строка;
* индекс текущего анализируемого символа;
* список констант;
* список индексов.

Методы:

* синтаксический и семантический анализ входной строки;
* синтаксический и семантический анализ оператора <константа>;
* синтаксический и семантический анализ оператора <идентификатор>;
* вывод результата семантического анализа;
* очистка массивов идентификаторов и констант;
* вывод сообщения об ошибке.

using System;

using System.Collections.Generic;

using System.Diagnostics.Eventing.Reader;

using System.Globalization;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Analyze15

{

internal class Analyze

{

enum State

{

start\_id, finish\_id, Id, start\_const, finish\_const,

C0, C1, C2, C3,

Start, Finish,

D1, D2,

W1, W2,

T1, R1, T2,

S1, S2, S3, S4, S5, S6

};

public static List<string> id\_list = new List<string>();

public static List<string> const\_list = new List<string>();

public static string[] Analyzing(string s)

{

int cur = 0;

State status = State.Start;

s = s + '\n';

s = s.ToUpper();

while (status != State.Finish && cur < s.Length)

{

switch (status)

{

case State.Start:

if (s[cur] == ' ')

{

status = State.Start;

cur += 1;

}

else if (s.Length - cur < 2)

{

cur = s.Length;

throw new ExceptionWithPosition("[!] СИНТАКСИЧЕСКАЯ ОШИБКА: Неожиданный конец строки", cur);

}

else if (s[cur] == 'D' && s[cur + 1] == 'O')

{

status = State.D1;

cur += 2;

}

else

{

throw new ExceptionWithPosition("[!] СИНТАКСИЧЕСКАЯ ОШИБКА: Допустим пробел или DO", cur);

}

break;

case State.D1:

if (s[cur] == ' ') { status = State.D2; cur += 1; }

else throw new ExceptionWithPosition("[!] СИНТАКСИЧЕСКАЯ ОШИБКА: Ожидался пробел", cur);

break;

case State.D2:

if (s[cur] == ' ')

{

status = State.D2;

cur += 1;

}

else if (s.Length - cur < 5)

{

cur = s.Length;

throw new ExceptionWithPosition("[!] СИНТАКСИЧЕСКАЯ ОШИБКА: Неожиданный конец строки", cur);

}

else if (s[cur] == 'W' && s[cur + 1] == 'H' && s[cur + 2] == 'I' && s[cur + 3] == 'L' && s[cur + 4] == 'E')

{

status = State.W1;

cur += 5;

}

else

{

throw new ExceptionWithPosition("[!] СИНТАКСИЧЕСКАЯ ОШИБКА: Ожидался пробел или WHILE", cur);

}

break;

case State.W1:

if (s[cur] == ' ')

{

status = State.W2;

cur++;

}

else throw new ExceptionWithPosition("[!] СИНТАКСИЧЕСКАЯ ОШИБКА: Ожидался пробел", cur);

break;

case State.W2:

if (s[cur] == ' ')

{

status = State.W2;

cur++;

}

else if (char.IsLetter(s[cur]))

{

status = State.T1;

cur = Identificator(s, cur);

}

else if (char.IsDigit(s[cur]) || s[cur] == '-' || s[cur] == '+')

{

status = State.T1;

cur = IsConst(s, cur);

}

else throw new ExceptionWithPosition("[!] СИНТАКСИЧЕСКАЯ ОШИБКА: Ожидался пробел или продолжение идентификатора или константы", cur);

break;

// Состояние после первого сравнения

case State.T1:

if (s[cur] == ' ')

{

status = State.T1;

cur++;

}

else if ((s[cur] == '=') || (s[cur] == '>') || (s[cur] == '<'))

{

status = State.R1;

cur++;

}

else

{

throw new ExceptionWithPosition("[!] СИНТАКСИЧЕСКАЯ ОШИБКА: Ожидались '<|>|=' или пробел", cur);

}

break;

case State.R1:

if (s[cur] == ' ')

{

status = State.R1;

cur++;

}

else if (char.IsLetter(s[cur]))

{

status = State.T2;

cur = Identificator(s, cur);

}

else if (char.IsDigit(s[cur]) || s[cur] == '-' || s[cur] == '+')

{

status = State.T2;

cur = IsConst(s, cur);

}

else throw new ExceptionWithPosition("[!] СИНТАКСИЧЕСКАЯ ОШИБКА: Ожидался пробел или продолжение идентификатора или константы", cur);

break;

case State.T2:

if (s[cur] == '\n') {

status = State.Finish;

}

else if (s[cur] == ' ')

{

status = State.S1;

cur++;

}

else

{

throw new ExceptionWithPosition("[!] СИНТАКСИЧЕСКАЯ ОШИБКА: Ожидался пробел или конец строки", cur);

}

break;

case State.S1:

if (s[cur] == ' ')

{

status = State.S1;

cur++;

}

else if (s[cur] == 'A' && s[cur + 1] == 'N' && s[cur + 2] == 'D')

{

status = State.S2;

cur += 3;

}

else if (s[cur] == 'O' && s[cur + 1] == 'R')

{

status = State.S2;

cur += 2;

}

else if (s[cur] == 'X' && s[cur + 1] == 'O' && s[cur + 2] == 'R')

{

status = State.S2;

cur += 3;

}

else throw new ExceptionWithPosition(" [!] СИНТАКСИЧЕСКАЯ ОШИБКА: Ожидался OR или XOR или AND", cur);

break;

case State.S2:

if (s[cur] == ' ')

{

status = State.S3;

cur += 1;

}

else throw new ExceptionWithPosition("[!] СИНТАКСИЧЕСКАЯ ОШИБКА: Ожидался пробел", cur);

break;

case State.S4:

if (s[cur] == ' ')

{

status = State.S4;

cur++;

}

else if ((s[cur] == '=') || (s[cur] == '>') || (s[cur] == '<'))

{

status = State.S5;

cur++;

}

else

{

throw new ExceptionWithPosition("[!] СИНТАКСИЧЕСКАЯ ОШИБКА: Ожидались '<|>|=' или пробел", cur);

}

break;

case State.S3:

if (s[cur] == ' ')

{

status = State.S3;

cur++;

}

else if (char.IsLetter(s[cur]))

{

status = State.S4;

cur = Identificator(s, cur);

}

else if (char.IsDigit(s[cur]) || s[cur] == '-' || s[cur] == '+')

{

status = State.S4;

cur = IsConst(s, cur);

}

else throw new ExceptionWithPosition("[!] СИНТАКСИЧЕСКАЯ ОШИБКА: Ожидался пробел или продолжение идентификатора или константы", cur);

break;

case State.S5:

if (s[cur] == ' ')

{

status = State.S5;

cur++;

}

else if (char.IsLetter(s[cur]))

{

status = State.S6;

cur = Identificator(s, cur);

}

else if (char.IsDigit(s[cur]) || s[cur] == '-' || s[cur] == '+')

{

status = State.S6;

cur = IsConst(s, cur);

}

else throw new ExceptionWithPosition("[!] СИНТАКСИЧЕСКАЯ ОШИБКА: Ожидался пробел или продолжение идентификатора или константы", cur);

break;

case State.S6:

if (s.Length != cur && s[cur] == ' ')

{

status = State.S6;

cur += 1;

}

else if (s[cur] == '\n')

{

status = State.Finish;

}

else { throw new ExceptionWithPosition("[!] СИНТАКСИЧЕСКАЯ ОШИБКА: Ожилася конец строки", cur); }

break;

}

}

if (status != State.Finish)

{

throw new ExceptionWithPosition("[!] СИНТАКСИЧЕСКАЯ ОШИБКА: Введённая строка не закончена!", cur);

}

return ResultToString(id\_list, const\_list);

}

public static string[] ResultToString(List<string> id\_list, List<string> const\_list)

{

string resultIds = string.Join(Environment.NewLine, id\_list);

string resultConsts = string.Join(Environment.NewLine, const\_list);

Clear\_lists();

return new string[] { resultIds, resultConsts };

}

public static void Clear\_lists()

{

const\_list.Clear();

id\_list.Clear();

}

public static int Identificator(string s, int cur)

{

int len = 0;

int p0 = cur;

State status\_id = State.start\_id;

while (status\_id != State.finish\_id && cur != -1 && cur < s.Length)

{

switch (status\_id)

{

case State.start\_id:

if (char.IsLetter(s[cur]))

{

status\_id = State.Id;

cur++;

len++;

}

else

{

throw new ExceptionWithPosition("[!] СИНТАКСИЧЕСКАЯ ОШИБКА: Ожидался буквенный символ", cur);

}

break;

case State.Id:

if (cur == s.Length)

{

status\_id = State.finish\_id;

}

else if (char.IsLetterOrDigit(s[cur]))

{

status\_id = State.Id;

cur += 1;

len++;

}

else

{

status\_id = State.finish\_id;

}

break;

}

}

if (len > 8)

{

throw new ExceptionWithPosition("[!] СЕМАНТИЧЕСКАЯ ОШИБКА: Слишком длинный идентификатор", cur);

}

string identifier = s.Substring(p0, len);

if (IsKeyword(identifier))

{

throw new ExceptionWithPosition("[!] СЕМАНТИЧЕСКАЯ ОШИБКА: Идентификатор не может быть ключевым словом", cur);

}

UpdateIdList(identifier);

return cur;

}

private static bool IsKeyword(string identifier)

{

string[] keywords = { "DO", "WHILE", "AND", "OR", "XOR"};

return keywords.Contains(identifier);

}

private static void UpdateIdList(string identifier)

{

if (!id\_list.Contains(identifier))

{

id\_list.Add(identifier);

}

}

static int IsConst(string s, int cur)

{

State status\_const = State.start\_const;

List<string> chars = new List<string>();

int p0 = cur;

bool flag = false;

int p\_dot = 0;

while (status\_const != State.finish\_const && cur != -1 && cur < s.Length)

{

switch (status\_const)

{

case State.start\_const:

if (s[cur] == '-' || s[cur] == '+')

{

status\_const = State.C1;

chars.Add(Convert.ToString(s[cur]));

cur += 1;

}

else if (char.IsDigit(s[cur])) //&& s[cur] != '0')

{

status\_const = State.C1;

chars.Add(Convert.ToString(s[cur]));

cur += 1;

}

else throw new ExceptionWithPosition("[!] СИНТАКСИЧЕСКАЯ ОШИБКА: Ожидалось +|-|1|..|9", cur);

break;

case State.C0:

if (char.IsDigit(s[cur]))

{

status\_const = State.C0;

chars.Add(Convert.ToString(s[cur]));

cur += 1;

}

else throw new ExceptionWithPosition("[!] СИНТАКСИЧЕСКАЯ ОШИБКА: Ожидалось 1|..|9", cur);

break;

case State.C1:

if (char.IsDigit(s[cur]))

{

status\_const = State.C1;

chars.Add(Convert.ToString(s[cur]));

cur += 1;

//p\_dot = cur - 1;

}

else if (s[cur] == '.')

{

status\_const = State.C2;

chars.Add(Convert.ToString(s[cur]));

cur += 1;

p\_dot = cur - 1;

}

else { status\_const = State.finish\_const; }

break;

case State.C2:

if (char.IsDigit(s[cur]))

{

status\_const = State.C3;

chars.Add(Convert.ToString(s[cur]));

cur += 1;

}

else

{

throw new ExceptionWithPosition("[!] СИНТАКСИЧЕСКАЯ ОШИБКА: Ожидалось 0|..|9", cur);

}

break;

case State.C3:

if (cur == s.Length)

{

status\_const = State.finish\_const;

}

else if (char.IsDigit(s[cur]))

{

status\_const = State.C3;

chars.Add(Convert.ToString(s[cur]));

cur += 1;

}

else status\_const = State.finish\_const;

break;

}

}

string ans = string.Join("", chars.ToArray());

double anas = Convert.ToDouble(ans, new NumberFormatInfo { NumberDecimalSeparator = "." });

if (anas < -32768 || anas > 32768)

{

throw new ExceptionWithPosition("[!] СЕМАНТИЧЕСКАЯ ОШИБКА: Число вне диапазона [-32768; 32768]", cur);

}

if (cur != -1)

{

string ind\_str = s.Substring(p0, cur - p0);

if (const\_list.Count == 0 || !const\_list.Contains(ind\_str))

{

const\_list.Add(ind\_str);

}

}

return cur;

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Analyze15

{

internal class ExceptionWithPosition : Exception

{

public int Position;

public ExceptionWithPosition() : base() { }

public ExceptionWithPosition(string message) : base(message) { }

public ExceptionWithPosition(string message, System.Exception inner) : base(message, inner) { }

public ExceptionWithPosition(string message, int pos) : base(message)

{

Position = pos;

}

}

}