МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ государственное БЮДЖЕТНОЕ

образовательное учреждение

высшего образования

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кафедра автоматизированных систем управления



**КУРСОВАЯ РАБОТА**

**по дисциплине:** Теория формальных языков и компиляторов

**на тему:** **Комментарии языка FORTRAN**

Выполнил:Проверил:

Студент гр. АВТ-912, АВТФ д.т.н., профессор

*Беккер А.К. Шорников Ю.В.*

«17» мая 2022 г.«\_\_\_» \_\_\_\_\_\_ 20\_\_ г.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись) (подпись)

Новосибирск 2022

Оглавление

[ПОСТАНОВКА ЗАДАЧИ 3](#_Toc103680900)

[ГРАММАТИКА 4](#_Toc103680901)

[КЛАССИФИКАЦИЯ ГРАММАТИКИ 5](#_Toc103680902)

[МЕТОД АНАЛИЗА 6](#_Toc103680903)

[ДИАГНОСТИКА И НЕЙТРАЛИЗАЦИЯ ОШИБОК 7](#_Toc103680904)

[ТЕСТИРОВАНИЕ 8](#_Toc103680905)

[ЛИТЕРАТУРА 11](#_Toc103680906)

[ЛИСТИНГ 12](#_Toc103680907)

# ПОСТАНОВКА ЗАДАЧИ

Выполнить программную алгоритма синтаксического анализа комментариев языка FORTRAN.

Комментарии в языке FORTRAN начинаются с восклицательного знака (!), все символы после этого (кроме строки символов) игнорируются компилятором.

Многострочные комментарии в данном языке отсутствуют.

# ГРАММАТИКА

G[Z] = {N, T, P, Z};

Z – начальный символ

N {Z, <символ>, <цифра>, <буква>} – множество нетерминальных символов

S – множество терминальных символов, куда входят все буквы и цифры

P:

1) Z → !{<символ>}

2) <символ> → <буква> | <цифра>

3) <буква> → a | b | c | … z| A | B | C … |Z

4) <цифра> → 0 | 1 | 2 | … | 9

Граф состояний

A = {S, ∑, δ, So, F}, где

S = {A, B, C, Err},

So = А,

F = {С, Err}

∑ = {!, digit|letter, /n}

δ – таблица переходов

# КЛАССИФИКАЦИЯ ГРАММАТИКИ

Грамматика G[Z] по классификации Хомского [2] относится к автоматной и имеет вид:

A→ aB | a | Λ, a∈, A,B∈

В левой части допускаются только нетерминальные символы, а в правой в части могут присутствовать как символы с терминального словаря, так и символы с нетерминального.

# МЕТОД АНАЛИЗА

Для грамматики G[E] был выбран метод анализа с помощью графа состояний, потому что он подходит для автоматных грамматик. Для реализации выбран язык C#, так как он является объектно-ориентированным.

Описание графа состояний:

A = {S, ∑, δ, So, F}, где

S = {A, B, C, Err},

So = А,

F = {С, Err},

∑ = {!, digit|letter, /n}

δ – таблица переходов

|  |  |  |  |
| --- | --- | --- | --- |
| Состояние | Функции | | |
| ! | digit|letter | /n |
| A | B | - | - |
| B | - | B | C |
| C | - | - | - |
| Err | - | - | - |

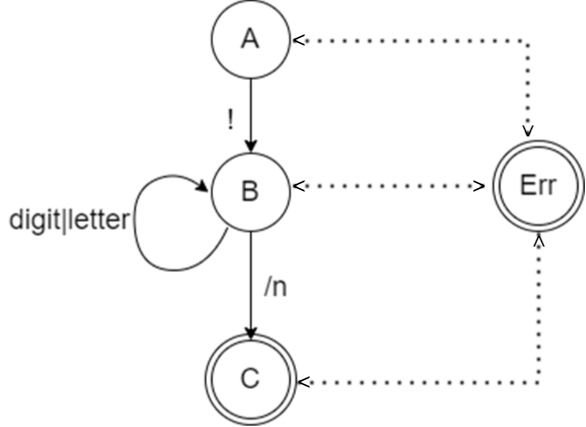


Рисунок 1 - Граф G[E]

# ДИАГНОСТИКА И НЕЙТРАЛИЗАЦИЯ ОШИБОК

Для данной грамматики производится диагностика и нейтрализация ошибок, с исправлениями. Нейтрализация ошибок осуществляется по методу Айронса, то есть, спускаясь по синтаксическому дереву без возврата по контексту, при обнаружении тупиковой ситуации отбрасываются те литеры (символы), которые привели в тупиковую ситуацию и разбор продолжается. Также есть возможность исправления следующих ошибок: ввод неверных символов; пропуск символа начала комментария «!»; пропуск символа конца комментария «\n». Программа выводит исправленную версию

ТЕСТИРОВАНИЕ

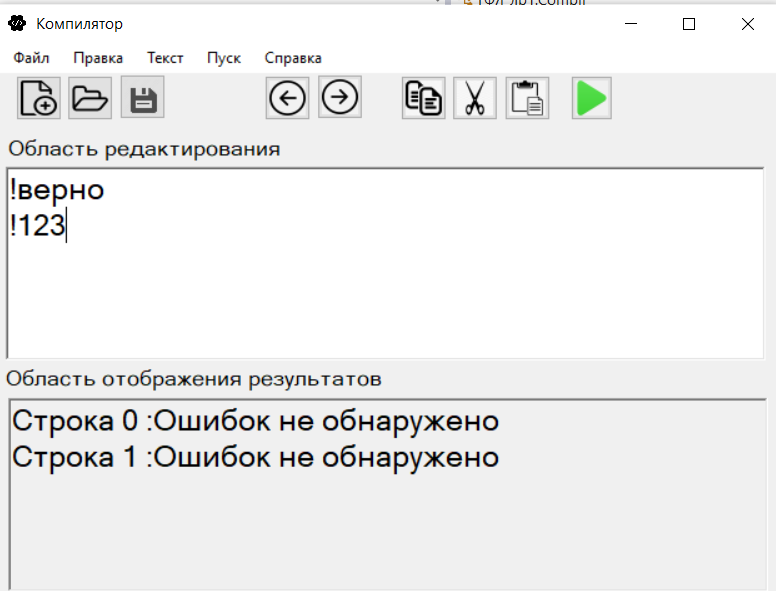


Рисунок 2 – Работа без ошибок

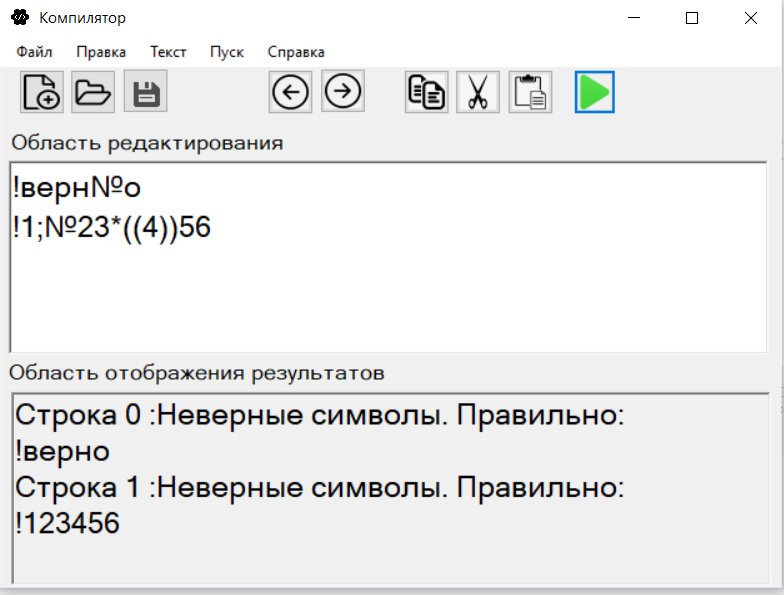


Рисунок 3 – Ошибка: ввод неверных символов

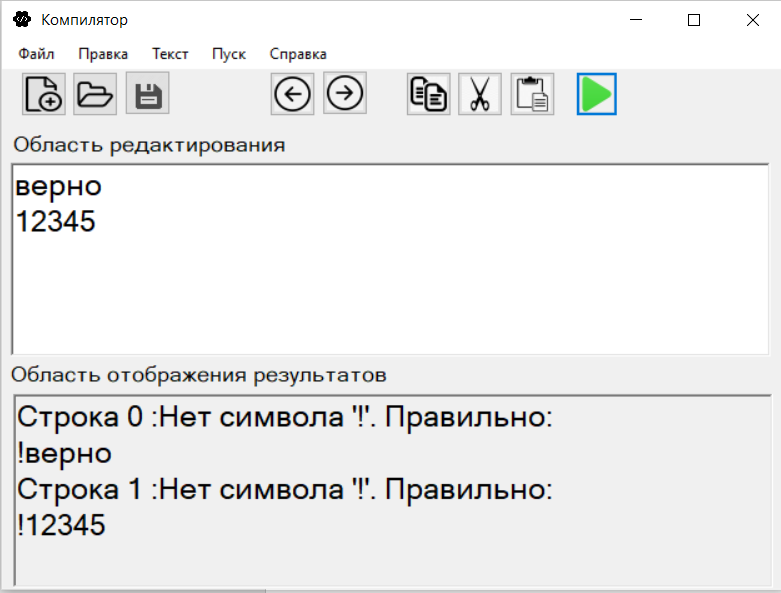


Рисунок 4 – Ошибка: пропущен символ «!»

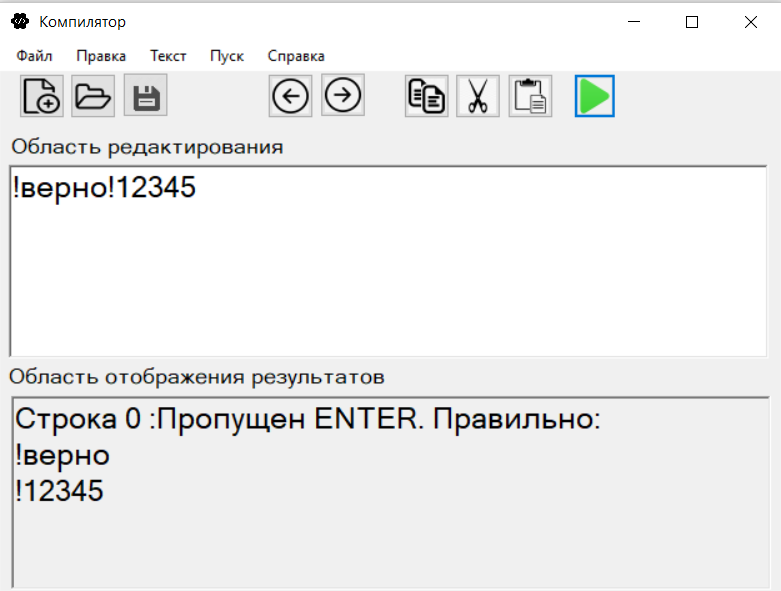


Рисунок 5 – Ошибка: пропущен ENTER

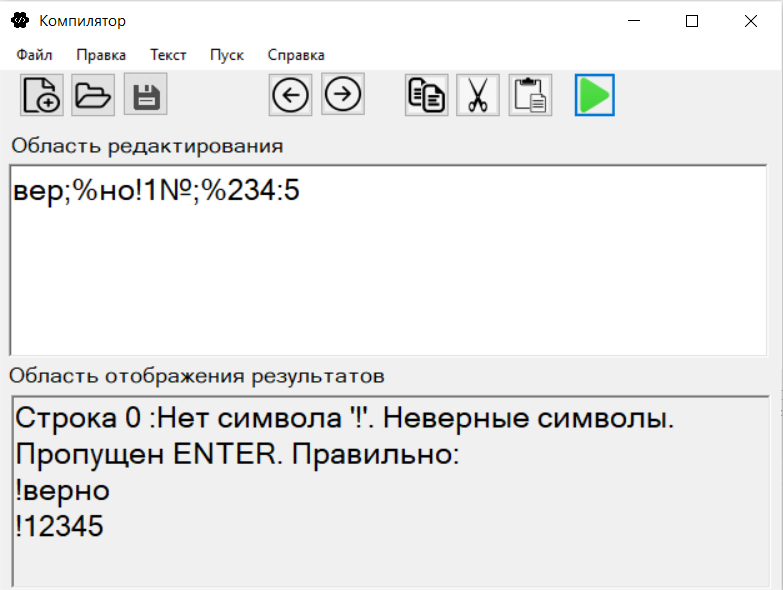


Рисунок 6 – Допущены все возможные ошибки

# ЛИТЕРАТУРА

1. Шорников Ю.В. Теория и практика языковых процессоров: Учеб. пособие. – Новосибирск: Изд-во НГТУ, 2004. – 208 с.
2. Пентус А. Е., Пентус М. Р. Теория формальных языков: Учебное пособие. – М.: Изд-во ЦПИ при механико-математическом ф-те МГУ, 2004 — 80 с.

# ЛИСТИНГ

Функция разбора входной строки

private void comment()

{

char[] prox;

bool flEr = false, fVoskl=false, fSimb=false, fEnter=false;

int count = 0, enter=0;

string[] str = editing.Text.Split('\n');

for (int r = 0; r < str.Length; r++)

{

flEr = false;

fVoskl = false;

fSimb = false;

fEnter = false;

prox = str[r].ToCharArray();

if (prox[0] != '!')

{

count++;

flEr = true;

fVoskl = true;

prox = sdvig(prox);

//continue;

}

for (int i = 1; i < prox.Length; i++)

{

if(char.IsLetter(prox[i]) == false && char.IsDigit(prox[i]) == false )

{

count++;

if ((prox[i]=='!') && (i!=0))

{

fEnter = true;

}

else

{

fSimb = true;

}

flEr = true;

}

}

result.Text += "Строка " + r + " :";

if (flEr == false)

{

result.Text += "Ошибок не обнаружено\n";

}

else

{

if (fVoskl == true)

{

result.Text += "Нет символа '!'. ";

}

if (fSimb == true)

{

count++;

result.Text += "Неверные символы. ";

prox = ErrSimbol(prox);

}

if (fEnter == true)

{

count++;

result.Text += "Пропущен ENTER. ";

result.Text += "Правильно:\n";

ErrEnter(prox);

}else

{

result.Text += "Правильно:\n";

for (int i = 0; i < prox.Length; i++)

{

result.Text += prox[i];

}

result.Text += "\n";

}

}

}

}

Функции нейтрализации ошибок

private char[] ErrSimbol(char[] prox)

{

Array.Resize<Char>(ref prox, prox.Length);

for (int i = 0; i < prox.Length; i++)

{

if(char.IsLetter(prox[i]) == false && char.IsDigit(prox[i]) == false && prox[i]!='!')

{

Array.Clear(prox, i, 1);

}

}

return prox;

}

private void ErrEnter(char[] prox)

{

int enter = 0, count = 0, n = 0;

for (int i = 0; i < prox.Length; i++)

{

if ((prox[i] == '!') && (i != 0))

{

enter = i;

count++;

for (int j = n; j < enter; j++)

{

result.Text += prox[j];

n = enter ;

}

result.Text += '\n';

// ErrEnter(prox,i);

}

}

if (count > 0)

{

for (int j = enter; j < prox.Length; j++)

{

result.Text += prox[j];

}

}

}

private char[] sdvig(char[] prox)

{

char prev = prox[0];

char[] Nprox= new char[prox.Length+1];

char next;

for (int i = 0; i < prox.Length -1; ++i)

{

next = prox[i + 1];

prox[i + 1] = prev;

prev = next;

}

prox[0] = '!';

for (int i = 0; i < prox.Length ; i++)

{

Nprox[i] = prox[i];

}

Nprox[Nprox.Length-1] = prev;

return Nprox;

}