МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования «Московский Авиационный Институт (Национальный Исследовательский Университет)»

Институт №3 Системы управления, информатика и электроэнергетика

Кафедра 304 «Вычислительные машины, системы и сети»

Отчет по лабораторной работе №1

по учебной дисциплине

«Конструирование компиляторов»

Выполнили:

студенты группы М30-411Б-21

Нуриев Н.Н.

Макаренков М.В.

Принял преподаватель:

доцент каф.304, к.т.н., Охотников Д.А.

Оглавление

[Задание 3](#_Toc192198953)

[Алфавит 4](#_Toc192198954)

[Граф состояний 4](#_Toc192198955)

[Грамматика языка 5](#_Toc192198956)

[Таблица переходов конечного автомата 7](#_Toc192198957)

[Результаты тестирования 8](#_Toc192198958)

[Код программы 10](#_Toc192198959)

# Задание

Разработать модуль приема и обработки данных команд робота. Робот передвигается в горизонтальной плоскости. Каждое сообщение включает в себя набор команд из следующего перечня: вперед, направо, налево.

Реализовать модуль управления роботом со следующим интерфейсом:

moveNorth();

moveSouth();

moveWest();

moveEast();

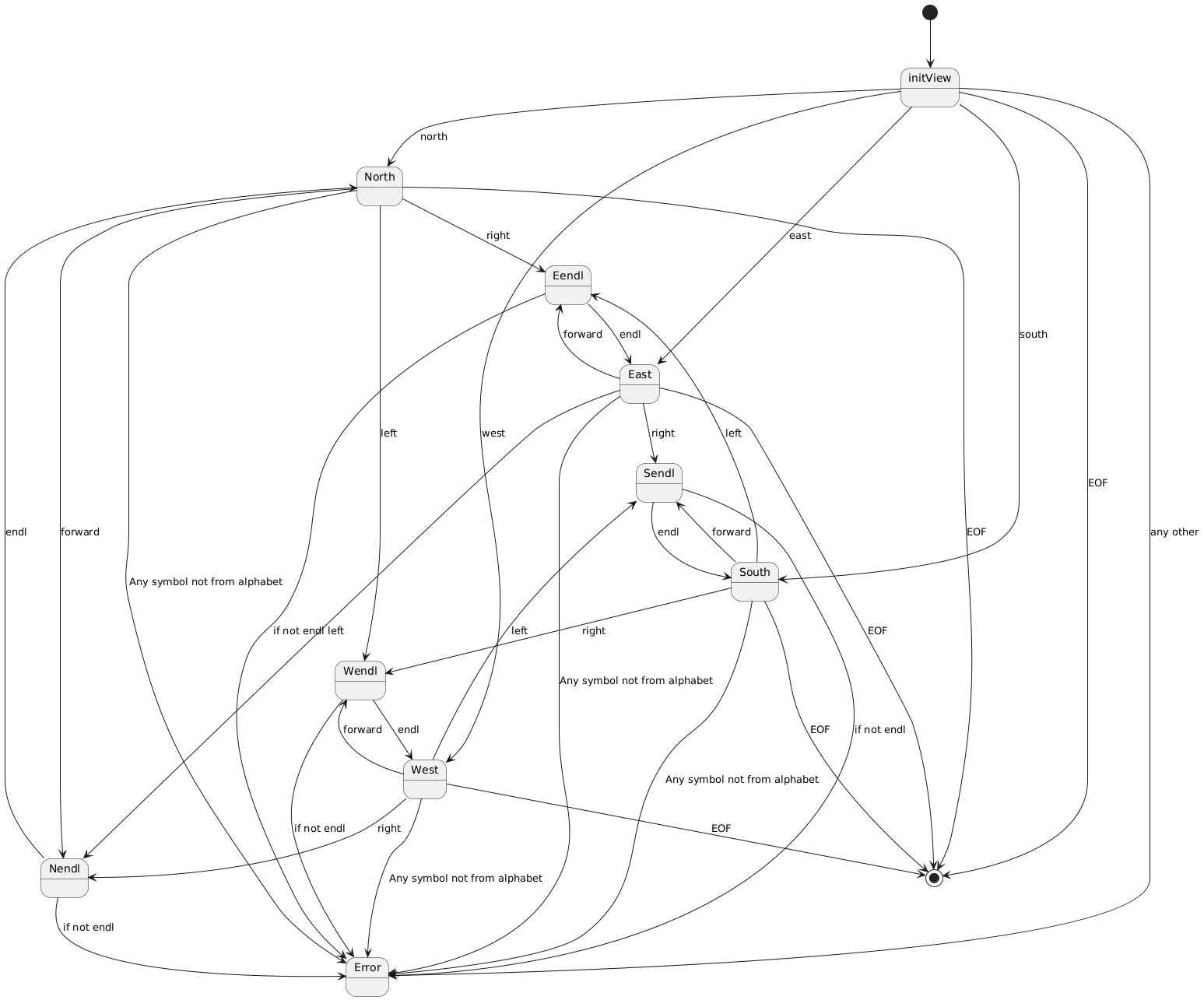
# Алфавит

T = {n, e, w, s, f, r, l, .}

S = { initView, North, Nendl, East, Eendl, West, Wendl, South, Sendl, Error}

N = { initView, North, Nendl, East, Eendl, West, Wendl, South, Sendl, Error}

# Граф состояний



# Грамматика языка

initView → n North

initView → e East

initView → w West

initView → s South

initView → NA Error

North → f Nendl

North → r Eendl

North → l Wendl

North → NA Error

North → n Error

North → e Error

North → w Error

North → s Error

Nendl → endl North

Nendl → NA Error

Nendl → n Error

Nendl → e Error

Nendl → w Error

Nendl → s Error

Eendl → f East

Eendl → l North

Eendl → r South

Eendl → NA Error

Eendl → n Error

Eendl → e Error

Eendl → w Error

Eendl → s Error

Wendl → f West

Wendl → n North

Wendl → r South

Wendl → l East

Wendl → NA Error

Wendl → n Error

Wendl → e Error

Wendl → w Error

Wendl → s Error

South → f Sendl

South → l East

South → r West

South → NA Error

South → n Error

South → e Error

South → w Error

South → s Error

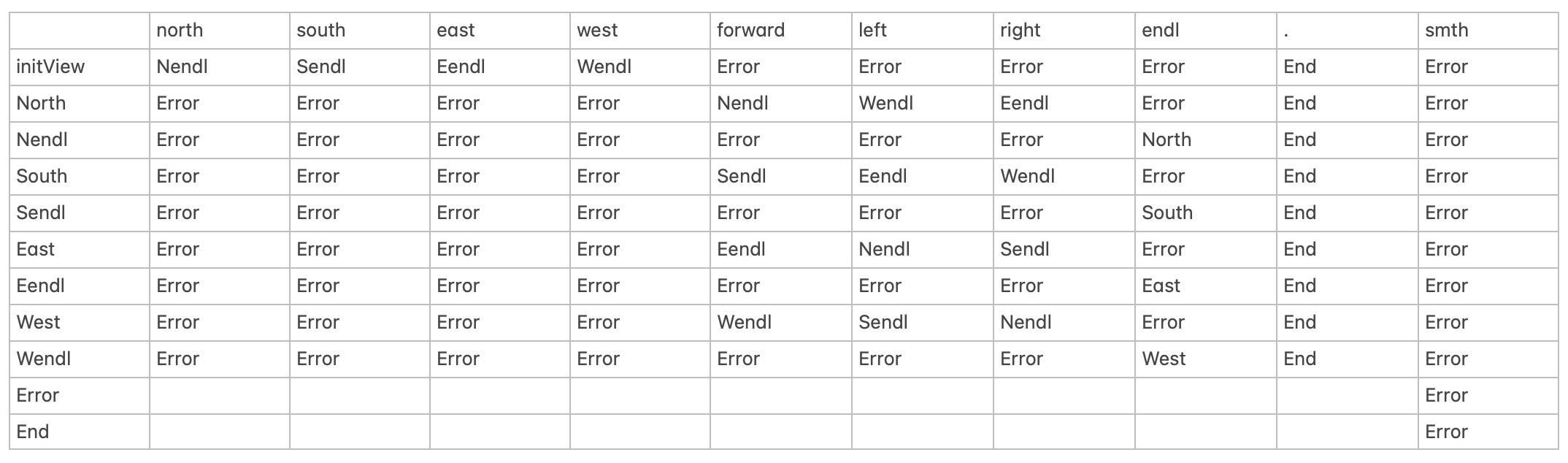
Sendl → Error

Eendl → Error

Nendl → Error

Wendl → Error

# Таблица переходов конечного автомата



# Результаты тестирования

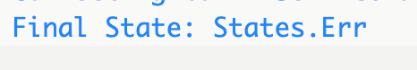
Некорректные тесты:

1. Не задано начальное направление:

Входные данные:

***f f f r f f r r f .***

Результат:



1. Нет символа конца программы – “.”

Входные данные:

***n f f f r f f r r f***

Результат:



1. Внутри входной строки символ начального направления

Входные данные:

***n f f f w r f f r r f .***

Результат:

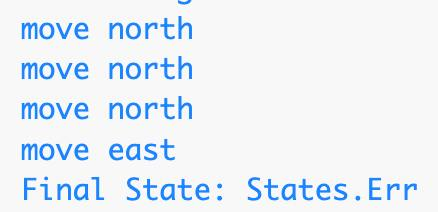


1. Есть символ не из алфавита

Входные данные:

***n f f f r f u f r r f .***

Результат:



Корректные тесты:

1. Входные данные:

***n f f f r f f r r f .***

Результат:



1. Входные данные:

***n l f r f r f l l l f .***

Результат:



# Код программы

import 'dart:io';

enum States {

  InitView,

  North,

  West,

  East,

  South,

  Nendl,

  Wendl,

  Eendl,

  Sendl,

  Err,

  End,

}

void main() {

  var currentState = States.InitView;

  final transitions = {

    States.InitView: {'n': () => States.Nendl, 's': () => States.Sendl, 'e': () => States.Eendl, 'w': () => States.Wendl, '\n': () => States.Err, 'f': () => States.Err,'l': () => States.Err,'r': () => States.Err,'.': () => States.End},

    States.North: {'f': moveNorth, 'l': () => States.Wendl, 'r': () => States.Eendl, '\n': () => States.North, '.': () => States.End,'n': () => States.Err,'s': () => States.Err,'w': () => States.Err,'e': () => States.Err},

    States.South: {'f': moveSouth, 'l': () => States.Eendl, 'r': () => States.Wendl, '\n': () => States.South, '.': () => States.End,'n': () => States.Err,'s': () => States.Err,'w': () => States.Err,'e': () => States.Err},

    States.East: {'f': moveEast, 'l': () => States.Nendl, 'r': () => States.Sendl, '\n': () => States.East, '.': () => States.End,'n': () => States.Err,'s': () => States.Err,'w': () => States.Err,'e': () => States.Err},

    States.West: {'f': moveWest, 'l': () => States.Sendl, 'r': () => States.Nendl, '\n': () => States.West, '.': () => States.End,'n': () => States.Err,'s': () => States.Err,'w': () => States.Err,'e': () => States.Err},

    States.Nendl: {'\n': () => States.North, '.': () => States.End,'f': () => States.Err, 'l': () => States.Err, 'r': () => States.Err,'n': () => States.Err,'s': () => States.Err,'w': () => States.Err,'e': () => States.Err},

    States.Wendl: {'\n': () => States.West, '.': () => States.End,'f': () => States.Err, 'l': () => States.Err, 'r': () => States.Err,'n': () => States.Err,'s': () => States.Err,'w': () => States.Err,'e': () => States.Err},

    States.Eendl: {'\n': () => States.East, '.': () => States.End,'f': () => States.Err, 'l': () => States.Err, 'r': () => States.Err,'n': () => States.Err,'s': () => States.Err,'w': () => States.Err,'e': () => States.Err},

    States.Sendl: {'\n': () => States.South, '.': () => States.End,'f': () => States.Err, 'l': () => States.Err, 'r': () => States.Err,'n': () => States.Err,'s': () => States.Err,'w': () => States.Err,'e': () => States.Err},

    States.End: {},

    States.Err: {}

  };

  final sourceFile = File('${Directory.current.path}/source.txt');

  final sourceFileSymbols = sourceFile.readAsStringSync();

  for (final symbolCode in sourceFileSymbols.runes) {

    final symbol = String.fromCharCode(symbolCode);

    if (!transitions[currentState]!.containsKey(symbol)) {

      currentState = States.Err;

      break;

    }

    currentState = transitions[currentState]![symbol]!();

  }

  if(currentState != States.End) {

    currentState = States.Err;

  }

  print('Final State: $currentState');

}

States moveNorth() {

  print("move north");

  return States.Nendl;

}

States moveSouth() {

  print("move south");

    return States.Sendl;

}

States moveEast() {

  print("move east");

  return States.Eendl;

}

States moveWest() {

  print("move west");

  return States.Wendl;

}