Федеральное государственное автономное образовательное учреждение высшего образования «**Уральский федеральный университет** имени первого Президента России Б. Н. Ельцина»

Отчет по лабораторной работе №3.5

дисциплины «Алгоритмы и анализ сложности»

Студент: Волков Илья Евгеньевич  
Группа: РИ-200004 (АТ-08)  
Преподаватель: Рыбкина Мария Николаевна

2022 год.

Оглавление

[Задание 3](#_Toc101813509)

[Теоретическая часть 3](#_Toc101813510)

[Инструкция пользователя 4](#_Toc101813511)

[Инструкция программиста 4](#_Toc101813512)

[Тестирование 5](#_Toc101813513)

[Листинги 6](#_Toc101813514)

[Вывод 11](#_Toc101813515)

# Задание

Лабораторная работа №3.5

Задание 5. Реализация нечеткого вывода  
  
Предлагается создать систему нечеткого вывода для управления автомобилем роботом.  
Составьте лингвистические переменные с базовыми термами:  
1. Скорость автомобиля. Термы: «Стоит», «Малая», «Средняя», «Высокая», «Очень высокая»  
2. Расстояние до впереди идущего автомобиля. Термы: «Малое», «Среднее», «Большое», «Очень большое».  
3. Управление одной педалью «Тормоз/Газ», от -100% до 100%. Термы: «Полный тормоз» (-100%), «Полный газ» (100%), другие термы составьте по своему усмотрению.  
  
Постройте 10 правил нечеткого вывода. Например,  
«Если скорость робота мала, расстояние до впереди идущего автомобиля среднее и скорость этого автомобиля высокая, то сильно газовать».  
Используя систему нечеткого вывода, определите величину управления педалью Тормоз/Газ для ситуации:  
• скорость робота равна 50км/ч,  
• расстояние до следующей машины 100 м.,  
• скорость следующей машины 20 км/ч

# Теоретическая часть

Есть несколько способов решения данной задачи:

1. Найти готовую библиотеку.

2. Найти готовое решение и, основываясь на нем, написать свое.

3. Полностью написать все с нуля самостоятельно.

Я выбрал третий вариант, ибо написать с нуля было куда проще, чем искать что-то в интернете.

# Инструкция пользователя

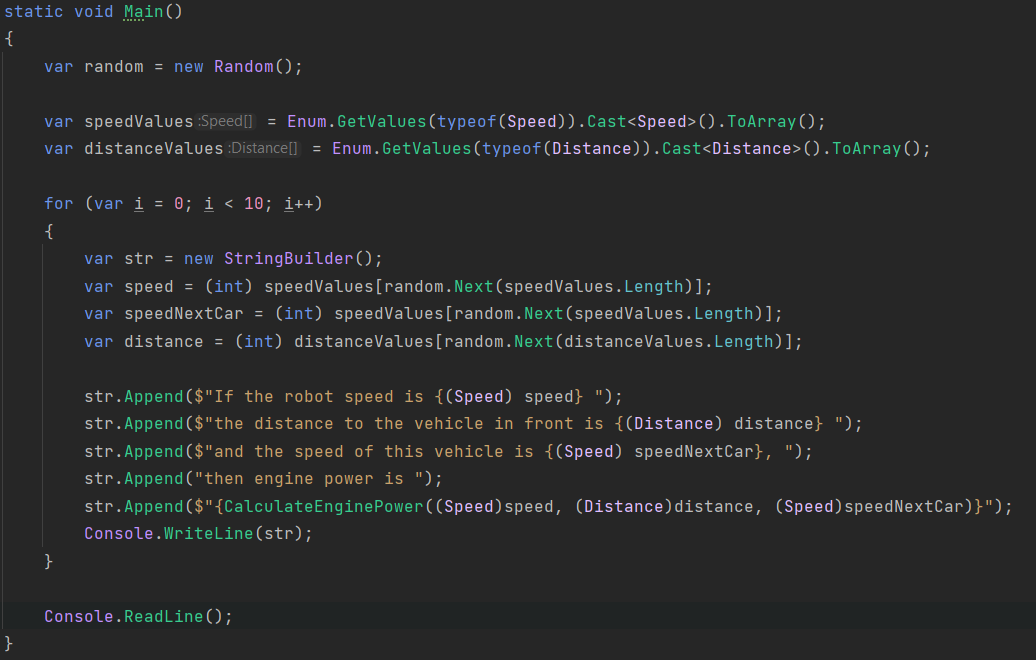
Для запуска софта используется файл Autopilot.exe.  
Откроется окно с примером работы программы. В ПО нет пользовательского интерфейса, т.е. кроме как кодом управлять им нельзя.

# Инструкция программиста

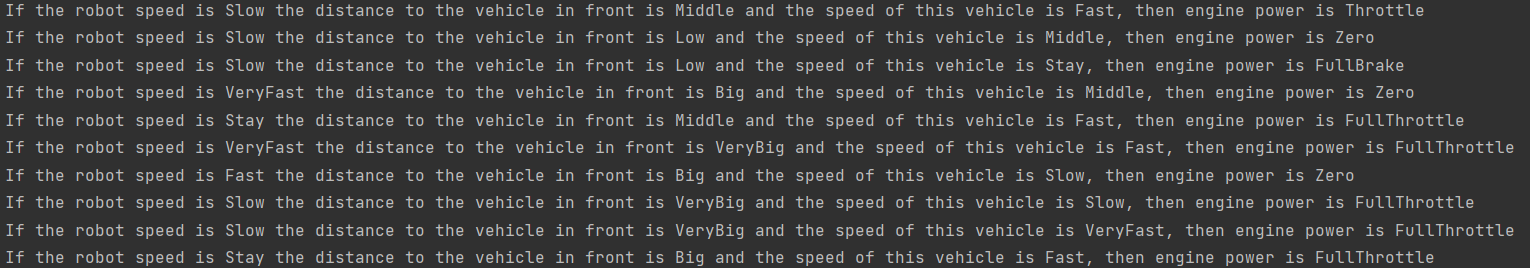
Для запуска софта используется файл Autopilot.exe.   
Для запуска решения используется файл Autopilot.sln (или Autopilot.csproj). Основной алгоритм реализован в файле Program.cs. Пример работы с методом представлен в Program.cs.

# Тестирование

1.



Вывод:



# Листинги

Листинг Program.cs:

using System.Text;  
  
namespace Autopilot  
{  
 class Autopilot  
 {  
 static void Main()  
 {  
 var random = new Random();  
  
 var speedValues = Enum.GetValues(typeof(Speed)).Cast<Speed>().ToArray();  
 var distanceValues = Enum.GetValues(typeof(Distance)).Cast<Distance>().ToArray();  
  
 for (var i = 0; i < 10; i++)  
 {  
 var str = new StringBuilder();  
 var speed = (int) speedValues[random.Next(speedValues.Length)];  
 var speedNextCar = (int) speedValues[random.Next(speedValues.Length)];  
 var distance = (int) distanceValues[random.Next(distanceValues.Length)];  
  
 str.Append($"If the robot speed is {(Speed) speed} ");  
 str.Append($"the distance to the vehicle in front is {(Distance) distance} ");  
 str.Append($"and the speed of this vehicle is {(Speed) speedNextCar}, ");  
 str.Append("then engine power is ");  
 str.Append($"{CalculateEnginePower((Speed)speed, (Distance)distance, (Speed)speedNextCar)}");  
 Console.WriteLine(str);  
 }  
  
 Console.ReadLine();  
 }  
   
 private static EnginePower CalculateEnginePower(Speed speed, Distance distance, Speed speedNextCar)  
 {  
 switch (speed)  
 {  
 case (int) Speed.**Stay**:  
 switch (speedNextCar)  
 {  
 case Speed.**Stay**:  
 return distance == Distance.**Low** ? EnginePower.**Zero** : EnginePower.**Throttle**;  
 case Speed.**Slow**:  
 return distance == Distance.**Low** ? EnginePower.**Zero** : EnginePower.**Throttle**;  
 case Speed.**Middle**:  
 return distance == Distance.**Low** ? EnginePower.**Zero** : EnginePower.**Throttle**;  
 case Speed.**Fast**:  
 case Speed.**VeryFast**:   
 return distance == Distance.**Low** ? EnginePower.**Zero** : EnginePower.**FullThrottle**;  
 }  
 break;  
   
 case Speed.**Slow**:  
 switch (speedNextCar)  
 {  
 case Speed.**Stay**:  
 switch (distance)  
 {  
 case Distance.**Low**:  
 return EnginePower.**FullBrake**;  
 case Distance.**Middle**:  
 return EnginePower.**Zero**;  
 case Distance.**Big**:  
 return EnginePower.**Throttle**;  
 case Distance.**VeryBig**:  
 return EnginePower.**FullThrottle**;  
 }  
 break;  
   
 case Speed.**Slow**:  
 switch (distance)  
 {  
 case Distance.**Low**:  
 return EnginePower.**Zero**;  
 case Distance.**Middle**:  
 return EnginePower.**Zero**;  
 case Distance.**Big**:  
 return EnginePower.**Throttle**;  
 case Distance.**VeryBig**:  
 return EnginePower.**FullThrottle**;  
 }  
 break;  
   
 case Speed.**Middle**:  
 switch (distance)  
 {  
 case Distance.**Low**:  
 return EnginePower.**Zero**;  
 case Distance.**Middle**:  
 return EnginePower.**Throttle**;  
 case Distance.**Big**:  
 return EnginePower.**Throttle**;  
 case Distance.**VeryBig**:  
 return EnginePower.**FullThrottle**;  
 }  
 break;  
 case Speed.**Fast**:  
 case Speed.**VeryFast**:   
 switch (distance)  
 {  
 case Distance.**Low**:  
 return EnginePower.**Zero**;  
 case Distance.**Middle**:  
 return EnginePower.**Throttle**;  
 case Distance.**Big**:  
 return EnginePower.**FullThrottle**;  
 case Distance.**VeryBig**:  
 return EnginePower.**FullThrottle**;  
 }  
 break;  
 }  
 break;  
   
 case Speed.**Middle**:  
 switch (speedNextCar)  
 {  
 case Speed.**Stay**:  
 switch (distance)  
 {  
 case Distance.**Low**:  
 return EnginePower.**FullBrake**;  
 case Distance.**Middle**:  
 return EnginePower.**Brake**;  
 case Distance.**Big**:  
 return EnginePower.**Zero**;  
 case Distance.**VeryBig**:  
 return EnginePower.**Throttle**;  
 }  
 break;  
   
 case Speed.**Slow**:  
 switch (distance)  
 {  
 case Distance.**Low**:  
 return EnginePower.**FullBrake**;  
 case Distance.**Middle**:  
 return EnginePower.**Zero**;  
 case Distance.**Big**:  
 return EnginePower.**Throttle**;  
 case Distance.**VeryBig**:  
 return EnginePower.**Throttle**;  
 }  
 break;  
   
 case Speed.**Middle**:  
 switch (distance)  
 {  
 case Distance.**Low**:  
 return EnginePower.**Zero**;  
 case Distance.**Middle**:  
 return EnginePower.**Throttle**;  
 case Distance.**Big**:  
 return EnginePower.**Throttle**;  
 case Distance.**VeryBig**:  
 return EnginePower.**FullThrottle**;  
 }  
 break;  
 case Speed.**Fast**:  
 case Speed.**VeryFast**:   
 switch (distance)  
 {  
 case Distance.**Low**:  
 return EnginePower.**Zero**;  
 case Distance.**Middle**:  
 return EnginePower.**Throttle**;  
 case Distance.**Big**:  
 return EnginePower.**FullThrottle**;  
 case Distance.**VeryBig**:  
 return EnginePower.**FullThrottle**;  
 }  
 break;  
 }  
 break;  
   
 case Speed.**Fast**:  
 case Speed.**VeryFast**:  
 switch (speedNextCar)  
 {  
 case (int) Speed.**Stay**:  
 switch (distance)  
 {  
 case Distance.**Low**:  
 return EnginePower.**FullBrake**;  
 case Distance.**Middle**:  
 return EnginePower.**Brake**;  
 case Distance.**Big**:  
 return EnginePower.**Zero**;  
 case Distance.**VeryBig**:  
 return EnginePower.**Throttle**;  
 }  
 break;  
   
 case Speed.**Slow**:  
 switch (distance)  
 {  
 case Distance.**Low**:   
 return EnginePower.**Brake**;  
 case Distance.**Middle**:  
 return EnginePower.**Brake**;  
 case Distance.**Big**:  
 return EnginePower.**Zero**;  
 case Distance.**VeryBig**:  
 return EnginePower.**Throttle**;  
 }  
 break;  
   
 case Speed.**Middle**:  
 switch (distance)  
 {  
 case Distance.**Low**:  
 return EnginePower.**Brake**;  
 case Distance.**Middle**:  
 return EnginePower.**Brake**;  
 case Distance.**Big**:  
 return EnginePower.**Zero**;  
 case Distance.**VeryBig**:  
 return EnginePower.**Throttle**;  
 }  
 break;  
 case Speed.**Fast**:  
 case Speed.**VeryFast**:   
 switch (distance)  
 {  
 case Distance.**Low**:  
 return EnginePower.**Brake**;  
 case Distance.**Middle**:  
 return EnginePower.**Zero**;  
 case Distance.**Big**:  
 return EnginePower.**FullThrottle**;  
 case Distance.**VeryBig**:  
 return EnginePower.**FullThrottle**;  
 }  
 break;  
 }  
 break;  
 }  
  
 return EnginePower.**FullBrake**;  
 }  
  
 *// 1. Скорость автомобиля.  
 // Термы: «Стоит», «Малая», «Средняя», «Высокая», «Очень высокая»* private enum Speed  
 {  
 **Stay** = 0,  
 **Slow** = 20,  
 **Middle** = 50,  
 **Fast** = 70,  
 **VeryFast** = 100  
 }  
  
 *// 2. Расстояние до впереди идущего автомобиля.  
 // Термы: «Малое», «Среднее», «Большое», «Очень большое».* private enum Distance  
 {  
 **Low** = 10,  
 **Middle** = 50,  
 **Big** = 100,  
 **VeryBig** = 200  
 }  
  
 *// 3. Управление одной педалью «Тормоз/Газ», от -100% до 100%.  
 // Термы: «Полный тормоз» (-100%), «Полный газ» (100%), «Тормоз» (-50%), «Газ» (50%), «Ноль» (0%),* private enum EnginePower  
 {  
 **FullThrottle** = 100,  
 **Throttle** = 50,  
 **Zero** = 0,  
 **Brake** = -50,  
 **FullBrake** = -100  
 }  
 }  
}

# Вывод

Я сумел, как мне кажется, реализовать алгоритм нечеткого вывода на примере автопилота робота. Программа работает очень быстро, хотя есть способы оптимизации. В процессе выполнения я узнал определение нечеткого вывода. Эта лабораторная работа показалась мне достаточно легкой.