Министерство образования и науки Российской Федерации
Федеральное агентство по образованию
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Вятский государственный университет»

Факультет автоматики и вычислительной техники

Кафедра электронных вычислительных машин

Отчет по лабораторной работе №2 дисциплины «Теория автоматов»

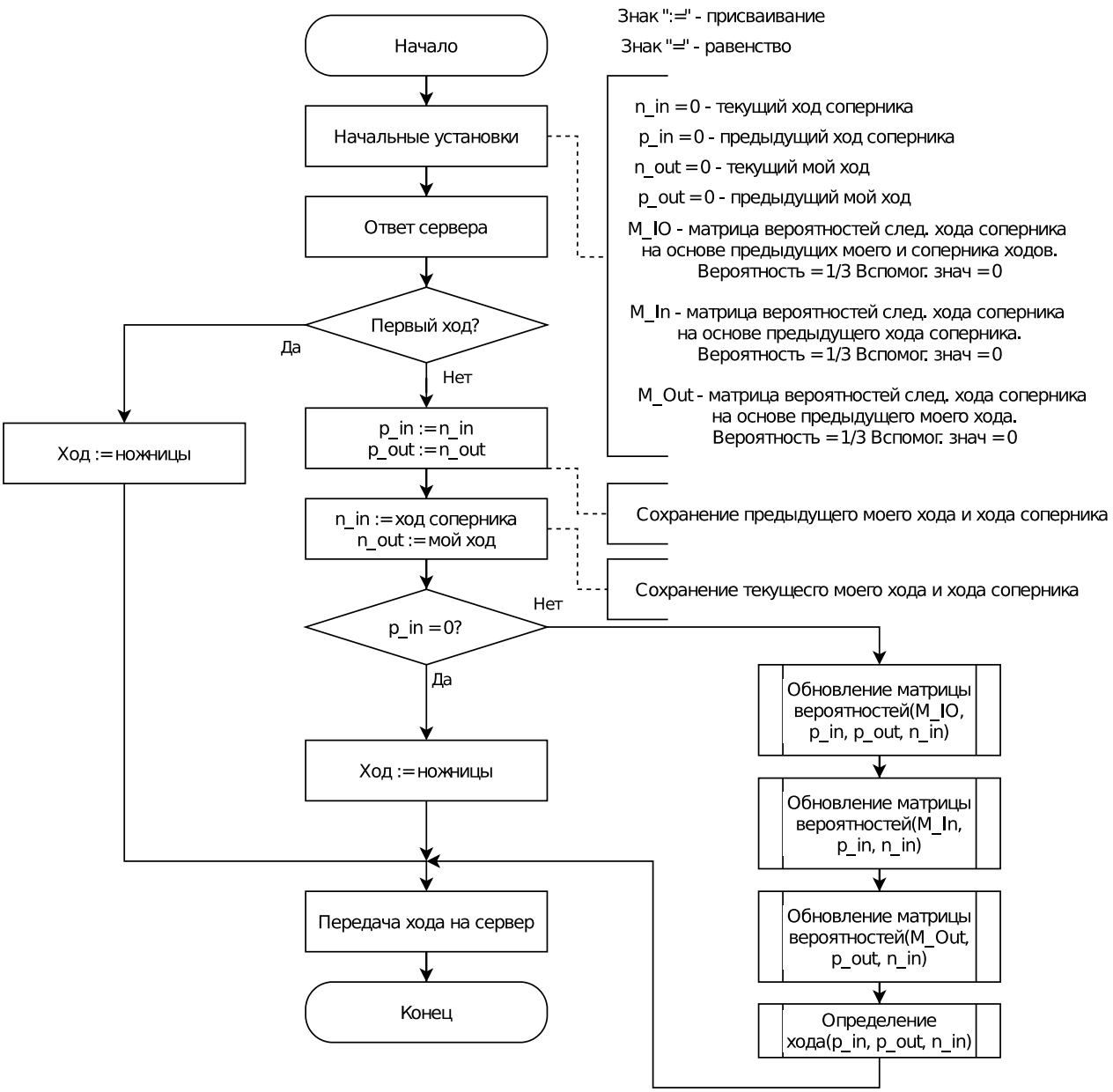
Выполнил студент группы ИВТ-22_____/Крючков И. С/
Проверил_____/Мельцов В. Ю./

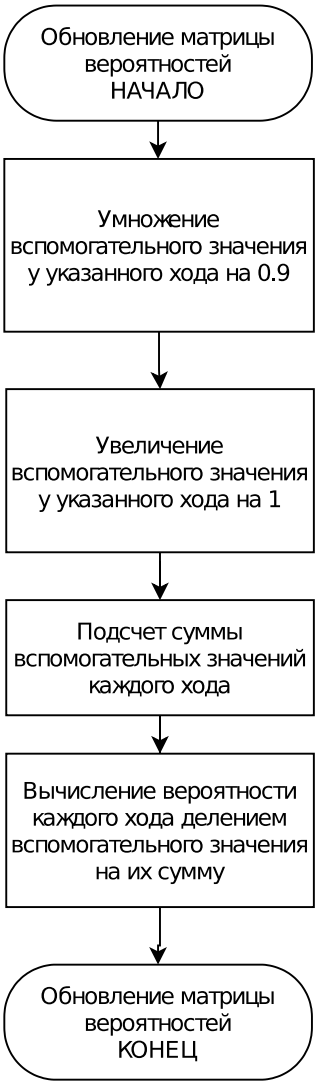
Киров 2021

Задание: разработать бота для игры «Камень, ножницы, бумага»

Словесное описание алгоритма:
Общей идей алгоритма является использование цепей Маркова для прогнозирования следующего хода соперника. В алгоритме используется три модели:
1 – на основе предыдущего хода соперника.
2 – на основе предыдущего моего хода.
3 – на основе предыдущего хода соперника и предыдущего моего хода.
Вспомогательное значение в каждой модели используется для сохранения количества ходов и дальнейшего вычисления вероятности хода по определенной паре (
1. пред. ход соперника – текущий ход соперника
2. пред. мой ход – текущий ход соперника
3. комбинация предыдущих хода соперника и моего хода – текущий ход соперника
).
При пересчете вероятностей в каждой модели производится умножение вспомогательного значения на определенную константу (0.9) (Значение от 0 до 1. Определяет память модели. Чем меньше значение, тем быстрее модель адаптируется к изменениям в поведении соперника).
Для определения следующего хода соперника используется максимальное из средних арифметических значений вероятностей по каждому «знаку» всех трех моделей.
Так как прогнозируется следующий ход соперника, «мой» ход определяется его противоположностью.

Схема алгоритма:




Обновление матрицы вероятностей НАЧАЛО
Умножение вспомогательного значения у указанного хода на 0.9
Увеличение вспомогательного значения у указанного хода на 1
Подсчет суммы вспомогательных значений каждого хода
Вычисление вероятности каждого хода делением вспомогательного значения на их сумму
Обновление матрицы вероятностей КОНЕЦ


Определение хода НАЧАЛО
Вычисление средних арифметических вероятностей каждого хода между матрицами для текущего хода соперника
Определение хода с максимальным значением средней вероятности
Определение противоположного (победного) хода
Определение хода КОНЕЦ

Листинг кода:

```
program v2;

uses math, sysutils;
type
    matr = array[1..9, 1..3, 1..2] of double;
    matrIO = array[1..3, 1..3, 1..2] of double;
    shortMatr = array[1..3, 1..2] of double;
    MarkovChain = object
        var
        matrix: matr;
        memory: double;
        constructor Create(mem:double);
        procedure updateMatrix(pair, inp: integer);
    end;
    IOChain = object
        var
        matrix: matrIO;
        memory: double;
        constructor Create(mem:double);
        procedure updateMatrix(dir, inp: integer);
  end;


var
beat: array[1..3] of integer = (3, 1, 2);
i:integer;
model:MarkovChain;
modelI:IOChain;
modelO:IOChain;
pref1:integer = 0;
pref2:integer = 0;
prefI1:integer = 0;
prefI2:integer = 0;
prefO1:integer = 0;
prefO2:integer = 0;
inp: integer;
out: integer;
wins: integer = 0;
games: integer = 20;

function findMax(mtr:shortMatr):integer;
var preRes:integer;
begin

    if(mtr[1, 2] >= mtr[2,2]) then
        preRes := 1
    else
        preRes := 2;

    if(mtr[preRes, 2] >= mtr[3,2]) then
        findMax := preRes
    else
        findMax := 3;
end;
```

```
function predict(mtrIO, mrtI, mtrO:shortMatr):integer;
var
preRes, i:integer;
sumMas:array[1..3]of double;

begin

    for i := 1 to 3 do
    begin
        sumMas[i] := (mtrIO[i, 1] + mrtI[i, 1] + mtrO[i,1])/3; //
среднее арифметическое коэффициентов каждого хода моделей
    end;

    if(sumMas[1] >= sumMas[2]) then
        preRes := 1
    else
        preRes := 2;

    if(sumMas[preRes] >= sumMas[3]) then
        predict := preRes
    else
        predict := 3;
end;

constructor MarkovChain.Create(mem:double);
var
i, j:integer;
begin

    // Заполнение начальной матрицы
    for i := 1 to 9 do
    begin
        for j := 1 to 3 do
        begin
            matrix[i, j, 1] := 1/3; //chance
            matrix[i, j, 2] := 0; // n
        end;
    end;

    memory := mem;
end;

// Обновление матрицы
procedure MarkovChain.updateMatrix(pair, inp: integer);
var
i, j:integer;
total:double;
begin
    for j := 1 to 3 do
    begin
        matrix[pair, j, 2] := matrix[pair, j, 2] * memory
    end;

    matrix[pair, inp, 2] := matrix[pair, inp, 2] + 1;
```

```
    total := 0;
    for j := 1 to 3 do
    begin
        total := total + matrix[pair, j, 2]
    end;

    for j := 1 to 3 do
    begin
        matrix[pair, j, 1] := matrix[pair, j, 2] / total;

    end;
end;

// I/O
constructor IOChain.Create(mem:double);
var
i, j:integer;
begin

    // Заполнение начальной матрицы
    for i := 1 to 3 do
    begin
        for j := 1 to 3 do
        begin
            matrix[i, j, 1] := 1/3; //chance
            matrix[i, j, 2] := 0; // n
        end;
    end;

    memory := mem;
end;

// Обновление матрицы I/O
procedure IOChain.updateMatrix(dir, inp: integer);
var
i, j:integer;
total:double;
begin
    for j := 1 to 3 do
    begin
        matrix[dir, j, 2] := matrix[dir, j, 2] * memory
    end;

    matrix[dir, inp, 2] := matrix[dir, inp, 2] + 1;

    total := 0;
    for j := 1 to 3 do
    begin
        total := total + matrix[dir, j, 2]
    end;

    for j := 1 to 3 do
    begin
        matrix[dir, j, 1] := matrix[dir, j, 2] / total;
```

```
    end;
end;

function encodePair(out, inp:integer):integer;
var
t:integer;
begin
    t:=inp;
    repeat
    out:=out*10; t:=t div 10;
    until t=0;

    encodePair:= out + inp;
end;

function fromPairToSingle(pair:integer):integer;
begin
    case pair of
        11: fromPairToSingle := 1; //КК
        12: fromPairToSingle := 2; //КН
        13: fromPairToSingle := 3; //КБ
        21: fromPairToSingle := 4; //НК
        22: fromPairToSingle := 5; //НН
        23: fromPairToSingle := 6; //НБ
        31: fromPairToSingle := 7; //БК
        32: fromPairToSingle := 8; //БН
        33: fromPairToSingle := 9; //ББ
    end;
end;


begin

model.Create(0.9); // IO
modelI.Create(0.9); // I
modelO.Create(0.9); // O


    out:= 2;

    pref2 := pref1;
    pref1 := fromPairToSingle(encodePair(out, inp));

    prefI2 := prefI1;
    prefI1 := inp;

    prefO2 := prefO1;
    prefO1 := out;

    if(pref2 <> 0)then
    begin
        model.updateMatrix(pref2, inp);
        modelI.updateMatrix(prefI2, inp);
        modelO.updateMatrix(prefO2, inp);
```

```
        out := beat[predict(model.matrix[pref1], modelI.matrix[prefI1],
modelO.matrix[prefO1])];

    end
    else
        out := 2;

end.
```

Вывод: в ходе выполнения лабораторной работы был реализован алгоритм игры «Камень, ножницы, бумага» на языке Pascal.