**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИИ**  
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ  
ВЫСШЕГО ОБРАЗОВАНИЯ  
**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ**  
**ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В.Г.ШУХОВА»**  
**(БГТУ им. В.Г.Шухова)**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

Практика № 2

дисциплина “ Безопасность ПИС”

по теме “ Пьяница на утёсе”

Выполнил: ст. группы ПВ-31  
Ковалев Павел Александрович

Проверил: Кабалянц П.С.

**Белгород 2021**

**Задание:**

**Пьяница на утёсе**

Пьяница стоит между двумя пропастями, с одной стороны река, с другой копья.

В начальный момент времени пьяница стоит на левой ноге. Его поведение задается графом марковского процесса (i, j, k - последние три цифры номера студенческого).

1) Необходимо определить среднее время жизни пьяницы и вероятность упасть в реку.

2) Написать программу, которая имитирует поведение пьяницы и выводит среднее количество переходов до падения с утеса и долю падений в реку.

3) Сравнить теоретическую вероятность падению в реку с долей падения в реку критерием сравнения долей.

**Решение**

Вычислим вероятности:

Составим матрицу переходных вероятностей:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Р** | **К** | **Л** | **О** | **П** |
| **Р** | 1 | 0 | 0 | 0 | 0 |
| **К** | 0 | 1 | 0 | 0 | 0 |
| **Л** | 0.17 | 0 | 0 | 0.83 | 0 |
| **О** | 0 | 0 | 0.1 | 0.5 | 0.4 |
| **П** | 0 | 0.44 | 0 | 0.55 | 0 |

Вычислим фундаментальную матрицу:

это правый нижний блок матрицы переходов, который отвечает за переходы

из непоглощающих состояний в непоглощающие

это левый нижний блок матрицы переходов, который отвечает за переходы

из непоглощающих состояний в поглощающие

Элементы матрицы В – это вероятность поглощения в том или ином состоянии в зависимости от того, из какого состояния начали.

Проверим гипотезу

p = 0,09

Пьяница упал в реку 84 раза из 1000

Выборочное значение критерия:

Результаты имитации поведения пьяницы соответствуют теоретическим результатам.

**Код программы:**

**main.cpp**

#include <QCoreApplication>

#include <QTextCodec>

#include <iostream>

*using* *namespace* std;

#define N 1000

#define M 5

*using* *namespace* std;

float randNum = 0;

float sum = 0;

float ivar = 10;

float jvar = 4;

float kvar = 3;

int condition = 1;

int lifeTime[N];

int river = 0;

int stakes = 0;

int longestLifeTime = 0;

int **main**(int argc, char \*argv[])

{

QCoreApplication a(*argc*, argv);

#ifdef Q\_OS\_WIN32

QTextCodec::setCodecForLocale(QTextCodec::codecForName("IBM 866"));

#endif

#ifdef Q\_OS\_LINUX

QTextCodec::setCodecForLocale(QTextCodec::codecForName("UTF-8"));

#endif

float P[M][M] = { 0 };

P[0][0] = 1;

P[1][1] = 1;

P[2][0] = ivar / (ivar + jvar);

P[2][3] = jvar / (ivar + jvar);

P[3][2] = ivar / (ivar + jvar + kvar);

P[3][3] = jvar / (ivar + jvar + kvar);

P[3][4] = kvar / (ivar + jvar + kvar);

P[4][1] = kvar / (jvar + kvar);

P[4][3] = jvar / (jvar + kvar);

cout << QString::fromUtf8("Матрица переходов:").toLocal8Bit().data() << endl;

*for* (int i = 0; i < M; i++) {

*for* (int j = 0; j < M; j++)

printf ("%.2f\t", P[i][j]);

cout << endl;

}

*for* (int i = 0; i < N; i++) { *//обнуляем* *массив*

lifeTime[i] = 0;

}

srand(time(0));

*for* (int i = 0; i < N;)

{

*while* ((condition != 3 || condition != 4) && i < N)

{

randNum = rand() % 99;

randNum /= 100;

*switch* (condition) {

*case* 0: *//на* *левой* *ноге*

*if* (randNum < P[2][0])

{

condition = 3;

lifeTime[i]++;

}

*if* (randNum >= P[2][0])

{

condition = 1;

lifeTime[i]++;

}

*break*;

*case* 1: *//на* *двух* *ногах*

*if* (randNum < P[3][2]) {

condition = 0;

lifeTime[i]++;

}

*if* (randNum >= P[3][2] && randNum < (P[3][2] + P[3][3]))

{

lifeTime[i]++;

}

*if* (randNum >= (P[3][2] + P[3][3])) {

condition = 2;

lifeTime[i]++;

}

*break*;

*case* 2: *//на* *правой* *ноге*

*if* (randNum < P[4][3])

{

condition = 1;

lifeTime[i]++;

}

*if* (randNum >= P[4][3])

{

condition = 4;

lifeTime[i]++;

}

*break*;

*case* 3:

condition = 1;

i++;

river++;

*break*;

*case* 4:

condition = 1;

i++;

stakes++;

*break*;

}

}

}

*//среднее* *арифметическое*

*for* (int i = 0; i < N; i++)

{

sum += lifeTime[i];

*if* (lifeTime[i] > longestLifeTime)

longestLifeTime = lifeTime[i];

}

sum /= N;

cout << endl << QString::fromUtf8("Продолжительность жизни пьяницы на утесе: ").toLocal8Bit().data() << sum << endl;

cout << endl << QString::fromUtf8("Количество падений в реку: ").toLocal8Bit().data() << river << endl;

cout << endl << QString::fromUtf8("Количество падений на копья: ").toLocal8Bit().data() << stakes << endl;

cout << endl << QString::fromUtf8("Максимальная продолжительность жизни: ").toLocal8Bit().data() << longestLifeTime;

*return* a.exec();

}