**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 №0

дисциплина “ Безопасность ПИС”

по теме “ Игра в имитацию”

Выполнил: ст. группы ПВ-31  
Ковалев Павел Александрович

Проверил: ст.пр. Бондаренко Татьяна Владимировна

**Белгород 2021**

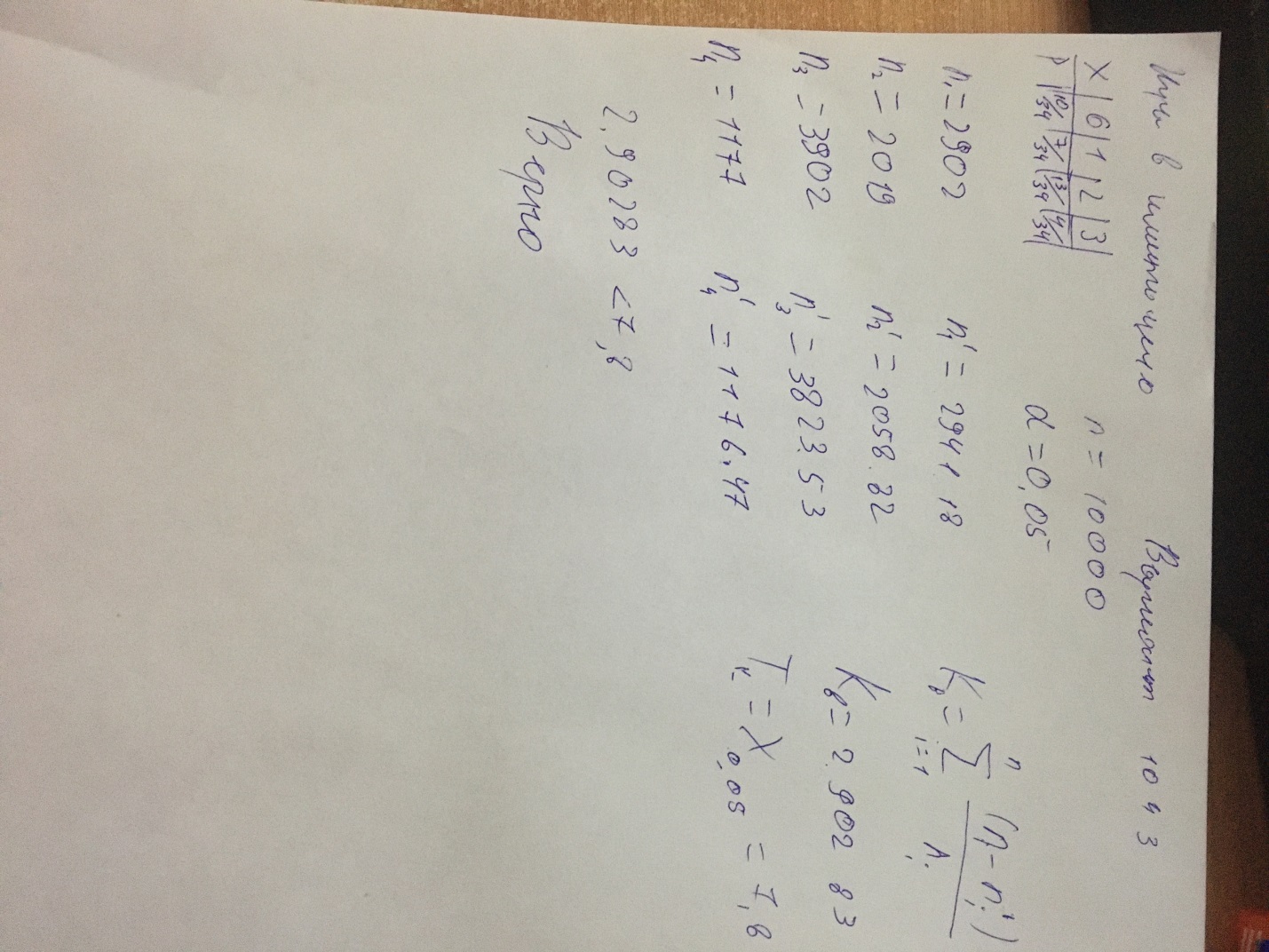
**Задание:**

Дискретная случайная величина ξ принимает значения 0, 1, 2 и 3 с вероятностями соответствующими данным варианта. Используя генератор псевдослучайных чисел, получить 10000 чисел, имитирующих поведение ξ. Проверить на уровне значимости 0,05 соответствие полученных данных распределению ξ, используя критерий сравнения долей. Вероятности: р0=i/2\*(i+j+k), р1=(j+k)/2\*(i+j+k), р2=(i+k)/2\*(i+j+k), р3=j/2\*(i+j+k). Значения i, j, k являются последними тремя цифрами зачетки. Если какая-то из этих трех цифр равна 0, то вместо 0 берется 10.

**Вариант 1043**

**Ход работы**

**Теоретическое решение:**



**Код программы:**

**main.cpp**

#include <QCoreApplication>

#include <stdio.h>

#include <stdlib.h>

#include <iostream>

*/\** *Функция* *возвращает* *случайное* *число* *по* *таблице* *случайных* *величин.*

*\** *Входные* *параметры:* *a(значение),* *p(вероятность),* *n(Количество* *генераций).*

*\** *Выходные* *параметры:* *a(значение),* *p(вероятность).*

*\*/*

double **getRand**(std::vector<double> &a, std::vector<double> &p){

double r = rand() / double(RAND\_MAX);*//Генератор* *псевдослучайных* *чисел*

double temp = 0.f;

*for* ( unsigned i = 0; i < a.size(); i++) {

temp += p[i];

*if*(r<temp)

*return* a[i];

}

*return* 0;

}

*/\** *Функция* *возвращает* *частоты* *выпадения* *при* *n* *бросках.*

*\** *Входные* *параметры:* *a(значение),* *p(вероятность),* *n(Количество* *генераций).*

*\** *Выходные* *параметры:* *a(значение),* *p(вероятность).*

*\*/*

std::vector<double> **getRandFreq**(std::vector<double>& a, std::vector<double>& p, unsigned int n){

std::vector<double> buffer;

*for* (unsigned int i = 0; i < a.size(); i++)

buffer.push\_back(0);

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

double r = getRand(*a*, *p*);

*for* (unsigned int i = 0; i < a.size(); i++)

*if*(a[i] == r)

buffer[i]++;

}

*return* buffer;

}

*/\** *Функция* *возвращает* *истину* *или* *лож* *в* *зависимости* *от* *результата* *проверки* *на* *достоверность* *используя* *критерий*

*\** *сравения* *долей.*

*\** *Входные* *параметры:* *a(значение),* *p(вероятность),* *n(Количество* *генераций)* *alp*

*\** *Выходные* *параметры:* *a(значение),* *p(вероятность).*

*\*/*

bool **checkTrue**(std::vector<double>& a, std::vector<double>& p, unsigned int count, float alp){

std::vector<double> n = getRandFreq(*a*, *p*, count);

*for* (unsigned int i = 0; i < n.size(); i++) {

std::cout << "n[" << i << "]:" << n[i] << std::endl;

}

std::vector<double> np;

double s = 0.f;

*for* (unsigned int i = 0; i < n.size(); i++)

s += n[i];

*for*(*auto* a : p)

np.push\_back(a \* s);

*for* (unsigned int i = 0; i < np.size(); i++) {

std::cout << "np[" << i << "]:" << np[i] << std::endl;

}

double kv = 0.f;

*for* (unsigned int i = 0; i < n.size(); i++)

kv += (n[i]-np[i])\*(n[i]-np[i]) / np[i];

std::cout << "kv = " << kv << std::endl;

*return* kv < 7.8f;

}

void **task**(){

std::vector<double> a = { 0.f, 1.f, 2.f, 3.f}; *//Значения*

std::vector<double> p = {10.f/34.f, 7.f/34.f, 13.f/34.f, 4.f/34.f}; *//Вероятности*

unsigned int n = 10000; *//Количество* *генерация(экспериментов*

*if* (checkTrue(*a*, *p*, n, 0.05f))

std::cout << "True" << std::endl;

*else*

std::cout << "False" << std::endl;

}

#include <QTextCodec>

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

task();

*return* a.exec();

}