Національний Технічний Університет України

«Київський Політехнічний Інститут»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Лабораторна робота №1

**Генератор псевдовипадкових двійкових послідовностей**

|  |  |
| --- | --- |
| Прийняв  Доц. Марковський О.П.  «\_\_»\_\_\_\_\_\_\_\_ 2015 р. | Виконав  Студент 2-ого курсу ФІОТ  групи ІО-41  Смішний Д.М. |

|  |  |  |  |
| --- | --- | --- | --- |
| Тип генератора | Розрядність | Об’єм вибірки | Складність |
| Т(р6) | 8+ | 10000 | Н |

Код програми:

**Клас LFSR**

**public** **class** LFSR {

/\*\*

\* **@param** args

\*/

**public** **int** args[];

**public** **int** arr[];

**public** **int** exit;

**public** **int** buffArr[];

**public** **int** arr1[];

**public** **int** xor\_exit;

LFSR(**int** ...args){

**this**.args = args;

}

**public** **int**[] getArgsLength() {

**return** buffArr = **new** **int**[args.length];

//return polynom\_arr=new int[args.length];

}

**public** **void** changeArgs(){

**for** (**int** i=0;i<args.length;i++){

buffArr[i]=args[i]-1;

**if** (buffArr[i]<=-1){

System.***out***.println("One of argument equally 0");

}

}

}

**public** **void** main(String[] args) {

// **TODO** Auto-generated method stub

}

**public** **void** arrRandom(**int** length){

arr = **new** **int**[length];

**for** (**int** i=0; i<arr.length; i++){

arr[i]=(**int**)(Math.*random*()\*2);

}

}

**public** **void** polynom(){

exit=arr[0];

**for** (**int** i=0; i<arr.length; i++){

**for** (**int** j=0; j<buffArr.length;j++){

**if** ((arr.length-1-i)==buffArr[j]&&buffArr[j]!=arr.length-1){

XOR(arr[0],arr[i]);

arr[i]=xor\_exit;

}

}

}

}

**public** **void** XOR (**int** param1, **int** param2) {

**if** (param1==param2){

xor\_exit=0;

} **else** {

xor\_exit=1;

}

}

**public** **int** shiftLeft() {

**for** (**int** i=0; i<arr.length-1; i++){

arr[i]=arr[i+1];

}

arr[arr.length-1]=exit;

**return** exit;

}

}

**Клас RandomTable**

**public** **class** RandomTable {

**public** **int** arr[];

**public** String itemBinLength;

**public** String bufferStr;

**public** **int** lastResult;

**public** OutputingLFSR output1= **new** OutputingLFSR ();

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

}

**public** **void** arr\_random(**int** length){

arr = **new** **int**[length];

**for** (**int** i=0; i<arr.length; i++){

arr[i]=(**int**)(Math.*random*()\*2);

}

}

**public** **void** binNum(**int** width){

**int** length=(**int**)Math.*pow*(2,width);

String tableItemLength="";

**for** (**int** i=0; i<arr.length; i++){

itemBinLength=Integer.*toBinaryString*(i);

**if** (itemBinLength.length()<length){

**for** (**int** j=0;j<width-itemBinLength.length();j++){

tableItemLength+="0";

}

itemBinLength=tableItemLength+Integer.*toBinaryString*(i);

tableItemLength="";

}

**if** (itemBinLength.equals(bufferStr)){

lastResult=arr[i];

}

}

}

**public** **void** doMethod(){

bufferStr="";

bufferStr=output1.result();

arr\_random(64);

binNum(6);

}

**public** **int** getLastResult(){

**return** lastResult;

}

}

**Клас OutputingLFSR**

**public** **class** OutputingLFSR {

LFSR lfsr1 = **new** LFSR(8,4,3,2);

LFSR lfsr2 = **new** LFSR(9,4);

LFSR lfsr3 = **new** LFSR(10,3);

LFSR lfsr4 = **new** LFSR(11,2);

LFSR lfsr5 = **new** LFSR(12,6,4);

LFSR lfsr6 = **new** LFSR(13,4,3,1);

**public** String outputLFSRs="";

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

}

**public** **void** toDoLFSR(LFSR name, **int** capacity){

name.getArgsLength();

name.changeArgs();

name.arrRandom(capacity);

name.polynom();

outputLFSRs += String.*valueOf*(name.shiftLeft());

}

**public** String result(){

outputLFSRs="";

toDoLFSR(lfsr1,8);

toDoLFSR(lfsr2,9);

toDoLFSR(lfsr3,10);

toDoLFSR(lfsr4,11);

toDoLFSR(lfsr5,12);

toDoLFSR(lfsr6,13);

**return** outputLFSRs;

}

}

**Клас WorkingClass**

**public** **class** WorkingClass {

/\*\*

\* **@param** args

\*/

**public** **static** RandomTable *element* = **new** RandomTable();

**public** **static** **int** *sampleArr*[];

**public** **final** **static** **int** ***SAMPLE\_LENGTH***=10000;

**static** FileWriter *write*;

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

*formSampleArr*();

}

**public** **static** **void** formSampleArr(){

*sampleArr* = **new** **int**[***SAMPLE\_LENGTH***];

**for**(**int** i = 0; i<*sampleArr*.length; i++) {

*element*.doMethod();

*sampleArr*[i]=*element*.lastResult;

}

/\*for (int i=0; i<sampleArr.length;i++){

System.out.print(sampleArr[i]+" ");

}\*/

}

}

**Клас Tests**

**public** **class** Tests {

**public** **static** **double** *finalTest1result*;

**public** **static** **double** *finalTest2Result*;

**public** **static** **double**[] *finalTest3ResultArr*;

**public** **static** **int** *finalTest4ResultArr*[];

**public** **static** **int** *complexity*=0;

**public** **static** String *stringSample*="";

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

WorkingClass.*formSampleArr*();

System.***out***.println("1st test: "+*test1*());

System.***out***.println("2nd test: "+*test2*());

*test3*(5);

*test4*();

}

**public** **static** **double** test1(){

*finalTest1result*=0.;

**for** (**int** i=0;i<WorkingClass.*sampleArr*.length;i++){

*finalTest1result*+=WorkingClass.*sampleArr*[i];

}

**return** (*finalTest1result*/(**double**)WorkingClass.***SAMPLE\_LENGTH***);

}

**public** **static** **double** test2(){

*finalTest2Result*=0.;

**for** (**int** i=0;i<WorkingClass.*sampleArr*.length-1;i++){

**if** (WorkingClass.*sampleArr*[i]==WorkingClass.*sampleArr*[i+1]){

*finalTest2Result*+=1.;

} **else**{

*finalTest2Result*+=0;

}

}

**return** (*finalTest2Result*/((**double**)WorkingClass.***SAMPLE\_LENGTH***-1));

}

**public** **static** **void** test3(**int** width){

String tableTruth[];

String tableItemLength="";

**int** length=(**int**)Math.*pow*(2,width);

tableTruth = **new** String[length];

*finalTest3ResultArr* = **new** **double** [length];

**for** (**int** i=0;i<tableTruth.length; i++){

tableTruth[i]=Integer.*toBinaryString*(i);

**if** (tableTruth[i].length()<length){

**for** (**int** j=0;j<width-tableTruth[i].length();j++){

tableItemLength+="0";

}

tableTruth[i]=tableItemLength+Integer.*toBinaryString*(i);

tableItemLength="";

}

}

**for** (**int** i=0; i<WorkingClass.*sampleArr*.length;i++){

*stringSample*+=WorkingClass.*sampleArr*[i];

}

**for**(**int** i=0; i<*finalTest3ResultArr*.length;i++){

*finalTest3ResultArr*[i]=0.;

}

**for** (**int** i=0;i<*stringSample*.length()-width;i++){

**for** (**int** j=0;j<tableTruth.length;j++){

**if** (tableTruth[j].equals(*stringSample*.substring(i,width+i))){

*finalTest3ResultArr*[j]+=1.;

}

}

}

**for** (**int** i=0;i<*finalTest3ResultArr*.length;i++){

*finalTest3ResultArr*[i]=*finalTest3ResultArr*[i]/(WorkingClass.***SAMPLE\_LENGTH***-(**double**)width);

}

**for** (**int** i=0; i<*finalTest3ResultArr*.length;i++){

System.***out***.println("3rd test: "+*finalTest3ResultArr*[i]);

}

}

**public** **static** **void** test4(){

*complexity*= 0;

**boolean** isComplexityFound = **false**;

**int** maxNum;

**for** (**int** j = 0; j < WorkingClass.*sampleArr*.length - 1 && !isComplexityFound; j++) {

*complexity*+=1;

maxNum = 1 << *complexity*;

*finalTest4ResultArr* = **new** **int**[maxNum];

isComplexityFound = **true**;

**for** (**int** i = 0; i < *finalTest4ResultArr*.length; i++) {

*finalTest4ResultArr*[i] = -1;

}

**for** (**int** i = 0; i < WorkingClass.*sampleArr*.length - *complexity*; i++) {

**int** numInWindow = *toIntFromBit*(WorkingClass.*sampleArr*, i, i + *complexity* - 1);

**int** valueForNum = *finalTest4ResultArr*[numInWindow];

**if** (valueForNum == -1) {

*finalTest4ResultArr*[numInWindow] = WorkingClass.*sampleArr*[i + *complexity*];

} **else** **if** (valueForNum != WorkingClass.*sampleArr*[i + *complexity*]) {

isComplexityFound = **false**;

**break**;

}

}

}

System.***out***.println("4th test: " + *complexity*);

}

**private** **static** **int** toIntFromBit(**int**[] arr, **int** start, **int** end) {

**int** num = 0;

**int** twoInPower = 1;

**for** (**int** i = end; i >= start; i--) {

num += arr[i]\*twoInPower;

twoInPower \*= 2;

}

**return** num;

}

}