МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ «ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

Кафедра «Програмна інженерія та інформаційні технології управління»

Звіт з лабораторної роботи №5

З предмету «Алгоритми та структури даних»

Виконав Студент групи КН-36а

Рубан Ю.Д.

Перевірила:

ас. Бородіна І. О.

Харків 2017

КОМБІНАТОРНІ АЛГОРИТМИ

Мета роботи: познайомитися з генераторами випадкових чисел та методами перевірки випадковості.

Завдання

Розробити програму, яка читає з клавіатури число N (1 < N < 256) та параметри генератору випадкових чисел та виводить на екран послідовність з N згенерованих чисел. Програма зберігає до файлу графічну характеристику послідовності згідно завдання та виводить на екран результат одного з тестів NIST

Варіант завдання

Лінійний конгруентний метод.

Автокореляція (користувач задає зсув для копії послідовності).

Частотний побітовий тест.

Хід виконання роботи

Розроблено програму, яка читає з клавіатури число N (1 < N < 256) та параметри генератору випадкових чисел та виводить на екран послідовність з N згенерованих чисел. Програма зберігає до файлу графічну характеристику послідовності згідно завдання та виводить на екран результат одного з тестів NIST

Код програми:

```
#include<iostream>
#include<Math.h>
#include<fstream>
using namespace std;

struct arr
{
	arr() { Arr = 0; size = 0; }
	arr(int *A, int s) { Arr = A; size = s; }
	friend ostream& operator<<(ostream&os, arr& A)
	{
		for (int i = 0; i < A.size; i++)
		{
			os << A.Arr[i];
		}
		if (A.size == 0) { A.Arr[0] = 0; os << A.Arr[0]; }
		return os;
	}
	int*Arr=0;
	int size=0;
};
struct graf
{
	double corel;
	int pushes;
	friend ostream& operator<<(ostream& os, graf A)
	{
		os << A.pushes <<"\t"<< A.corel;
		return os;
	}
};
```

```
class cList
{
private:
	cList *head = 0;
	cList *tail = 0;
	cList *next = 0;
	int count=0;
	int data=0;
public:
	void add(int x)
	{
		cList *temp = new cList;
		temp->data = x;
		temp->next = head;
		if (head == NULL) { tail = head = temp; }
		else
		{
			tail->next = temp;
			tail = temp;
		}
		count++;
	}
	void doCycle(int cycleCount)
	{
		cList * temp = this->head;
		for (int i = 0; i < cycleCount; i++)
		{
			temp = temp->next;
		}
		head = temp;
	}
	int getCount() { return count; }
	double getAvg()
	{
		double sum = 0;
		cList*temp = head;
		for (int i = 0; i < count; i++)
		{
			sum += temp->data;
			temp = temp->next;
		}
		return (sum / (double)count);
	}
	int operator[](int index)
	{
		cList*temp = head;
		for (int i = 0; i < index; i++)
		{
			temp = temp->next;
		}
		return temp->data;
	}
};

unsigned int random(int a, int c, int i);
arr binary(int a);
double binTest(arr* A, int n);
double corel(cList f, cList s);

int main()
{
	int N, a, c;
	cout << "Enter N, a, c (best a,c = 69069, 5)" << endl;
	cin >> N >> a >> c;
	arr *A = new arr[N];
	int z;
	cList first, second;
	cout << endl;
	for (int i = 0; i < N; i++)
	{
		z = random(a, c, i) % 20;
		cout << z<<" ";//Задание 1 - получить последовательность линейно конгруентным методом
		A[i] = binary(z);
		first.add(z);
	}
	second = first;
```

```
	cout << endl << endl;
	for (int i = 0; i < N; i++)
	{
		cout << A[i] << endl;//Массив двоичных чисел, которые взяты с последовательности
	}
	cout << endl;
	cout << "Enter count of pushes " << endl;
	int cycles;
	cin >> cycles;
	graf *G = new graf[cycles];
	for (int i = 0; i < cycles; i++) //Задание 2 - получаем зависимость кореляции от сдвигов (кол-во
сдвигов вводит юзер)
	{
		second.doCycle(1);
		G[i].corel = corel(first, second);
		G[i].pushes = i + 1;
	}
	ofstream o("corels.txt");
	o << "pushes\t" << "corels(pushes)" << endl;
	for (int i = 0; i < cycles; i++)
	{
		o << G[i] << endl;
	}
	cout << endl;
	cout << "The dependance of corels on pushes successfully written in a file corels.txt" << endl <<
endl;
	cout <<"Bite test: "<<binTest(A, N) << endl;//Задание 3 - Частотный побитовый тест

	system("pause");
	return 0;
}

unsigned int random(int a, int c,int i)
{
	static unsigned int X = 1;
	X = (a*X + c) % (int)pow(2,32);
	if (i == 0) {unsigned int n = X; X = 0; return n; }
	return random(a,c,i-1);
}
arr binary(int a)
{
	int x = a;
	int c = 0;
	while (x > 0)
	{
		x /= 2;
		c++;
	}
	x = a;
	int*A = new int[c];
	c = 0;
	while (x > 0)
	{
		A[c] = x % 2;
		x /= 2;
		c++;
	}
	int *B = new int[c];
	if (c == 0) { B[0] = 0; }
	else
	{
		int j = c - 1;
		for (int i = 0; i < c; i++, j--)
		{
			B[i] = A[j];
		}
	}
	arr Arr(B,c);
	return Arr;
}
double binTest(arr* A,int n)
{
	double ones = 0;
	double size=0;
	for (int i = 0; i < n; i++)
	{
		for (int j = 0; j < A[i].size; j++)
```

```
                {
                        if (A[i].Arr[j] == 1)
                        {
                                ones++;
                        }
                        size++;
                }
        }
        double test = ones / size;
        return test;
}
double corel(cList f, cList s)
{
        double coef;
        double upSum=0;
        double downSum1=0, downSum2=0,downSumRes;
        double xAvg = f.getAvg();
        double yAvg = s.getAvg();
        double x, y;
        for (int i = 0; i < f.getCount(); i++)
        {
                x = (f[i] - xAvg);
                y = (s[i] - yAvg);
                upSum += x*y;
                downSum1 += x*x;
                downSum2 += y*y;

        }
        downSumRes = downSum1*downSum2;
        downSumRes = sqrt(downSumRes);
        coef = upSum / downSumRes;
        return coef;
}
```

Висновки

У даній лабораторній роботі я навчився створювати псевдовипадкові послідовності лінійно конгруентним методом, а також досліджувати цю послідовність автокореляційним та частотно побітовими тестами.