***Министерство образования Республики Беларусь***

***Учреждение Образования***

***«Брестский Государственный Технический Университет»***

***Кафедра ИИТ***

**Лабораторная работа №9**

**По дисциплине «Традиционные и интеллектуальные**

**информационные технологии»**

**Тема:**

**«Основы имитационного моделирования»**

**Выполнил:**

Студент 1-го курса

Группы ИИ-15(1)

Волк И. А.

**Проверил:**

Анфилец С. В.

Брест 2018

Цель: познакомиться с основами имитационного моделирования.

***Задание:***

Построить двухключевую систему с использованием алгоритма RSA и выполнить в ней операцию шифрования и дешифрования трех первых букв фамилии студента (при количестве букв меньше 3, недостающие буквы берутся из имени). Пара простых чисел P и Q выбирается из диапазона ближайших к количеству букв в фамилии и имени студента.

Код программы:

#include "stdafx.h"

#include <Windows.h>

#include <iostream>

#include <cmath>

using namespace std;

struct pavilion

{

char name;

int paths[4];

int numOfBuys = 0;

};

void buyer( int\*\*, int\*, int\*\*,

int\*, int\*, int,

pavilion\*, int\*, int);

int choose(int\*, int\*, int);

bool isOnWay(pavilion, int, int);

bool toBuy(int);

void result(pavilion\*, int);

int main()

{

/\* map \*/

int numOfPaths = 4;

int \*\* paths = new int\*[numOfPaths];

int \* sizeOfPath = new int[numOfPaths];

sizeOfPath[0] = 4;

paths[0] = new int[sizeOfPath[0]];

paths[0][0] = 0; paths[0][1] = -1; paths[0][2] = 2; paths[0][3] = 1;

sizeOfPath[1] = 4;

paths[1] = new int[sizeOfPath[1]];

paths[1][0] = 1; paths[1][1] = 0; paths[1][2] = 2; paths[1][3] = 3;

sizeOfPath[2] = 3;

paths[2] = new int[sizeOfPath[2]];

paths[2][0] = 1; paths[2][1] = 0; paths[2][2] = -1;

sizeOfPath[3] = 3;

paths[3] = new int[sizeOfPath[3]];

paths[3][0] = 1; paths[3][1] = 3; paths[3][2] = -1;

int\*\* chancesToWay = new int\*[numOfPaths];

chancesToWay[0] = new int[sizeOfPath[0]];

chancesToWay[0][0] = 20; chancesToWay[0][1] = 10; chancesToWay[0][2] = 40; chancesToWay[0][3] = 30;

chancesToWay[1] = new int[sizeOfPath[1]];

chancesToWay[1][0] = 20; chancesToWay[1][1] = 30; chancesToWay[1][2] = 10; chancesToWay[1][3] = 40;

chancesToWay[2] = new int[sizeOfPath[2]];

chancesToWay[2][0] = 50; chancesToWay[2][1] = 10; chancesToWay[2][2] = 40;

chancesToWay[3] = new int[sizeOfPath[3]];

chancesToWay[3][0] = 80; chancesToWay[3][1] = 10; chancesToWay[3][2] = 10;

// pavilions

char names[] = "ABCDEFGHIJKLM";

int numOfPavs = sizeof(names) / sizeof(\*names);

pavilion \* pavs = new pavilion[numOfPavs];

for (int i = 0; i < numOfPavs; i++)

pavs[i].name = names[i];

pavs[0].paths[0] = 2; pavs[0].paths[1] = -1; pavs[0].paths[2] = 1; pavs[0].paths[3] = -1; // A

pavs[1].paths[0] = 2; pavs[1].paths[1] = -1; pavs[1].paths[2] = 1; pavs[1].paths[3] = -1; // B

pavs[2].paths[0] = -1; pavs[2].paths[1] = 0; pavs[2].paths[2] = -1; pavs[2].paths[3] = -1; // C

pavs[3].paths[0] = 0; pavs[3].paths[1] = -1; pavs[3].paths[2] = -1; pavs[3].paths[3] = -1; // D

pavs[4].paths[0] = 3; pavs[4].paths[1] = 1; pavs[4].paths[2] = -1; pavs[4].paths[3] = -1; // E

pavs[5].paths[0] = 3; pavs[5].paths[1] = 1; pavs[5].paths[2] = -1; pavs[5].paths[3] = -1; // F

pavs[6].paths[0] = -1; pavs[6].paths[1] = 3; pavs[6].paths[2] = -1; pavs[6].paths[3] = 0; // G

pavs[7].paths[0] = -1; pavs[7].paths[1] = 3; pavs[7].paths[2] = -1; pavs[7].paths[3] = 0; // H

pavs[8].paths[0] = -1; pavs[8].paths[1] = 3; pavs[8].paths[2] = -1; pavs[8].paths[3] = 0; // I

pavs[9].paths[0] = -1; pavs[9].paths[1] = 3; pavs[9].paths[2] = -1; pavs[9].paths[3] = 0; // J

pavs[10].paths[0] = -1; pavs[10].paths[1] = 3; pavs[10].paths[2] = -1; pavs[10].paths[3] = 0; // K

pavs[11].paths[0] = -1; pavs[11].paths[1] = -1; pavs[11].paths[2] = -1; pavs[11].paths[3] = 2; // L

pavs[12].paths[0] = -1; pavs[12].paths[1] = -1; pavs[12].paths[2] = -1; pavs[12].paths[3] = 2; // M

// entry

int numOfEntry = 3;

int \*entry = new int[numOfEntry];

entry[0] = 0;

entry[1] = 2;

entry[2] = 3;

int\* chances = new int[numOfEntry];

chances[0] = 20; chances[1] = 30; chances[2] = 50;

/\*\*/

int numOfBuyers = 10000;

int class1 = (int)((double)numOfBuyers \* 0.2);

int class2 = (int)((double)numOfBuyers \* 0.6);

int class3 = numOfBuyers - class1 - class2;

int chances1[13] = { 30 ,10, 10, 5, 5, 5, 5, 5, 5, 5, 5, 0, 0 };

int chances2[13] = { 0, 0, 5, 5 ,10, 10, 40, 10 ,10, 5, 5, 0 ,0 };

int chances3[13] = { 0, 0 ,5 ,5 ,5 ,5, 5, 5, 5, 10, 10, 20 ,25 };

for (int i = 0; i < class1; i++)

buyer(paths, sizeOfPath, chancesToWay, entry, chances, numOfEntry, pavs, chances1, numOfPavs);

for(int i = 0; i < class2; i++)

buyer(paths, sizeOfPath, chancesToWay, entry, chances, numOfEntry, pavs, chances2, numOfPavs);

for (int i = 0; i < class3; i++)

buyer(paths, sizeOfPath, chancesToWay, entry, chances, numOfEntry, pavs, chances3, numOfPavs);

result(pavs, numOfPavs);

system("pause");

return 0;

}

void buyer(int\*\* paths, int\* sizeOfPath, int\*\* chancesToWay,

int\* entryArr, int\* chancesToEntry, int numOfEntry,

pavilion\* pavs, int\* chancesToBuy, int numOfPavs)

{

int pos = choose(entryArr, chancesToEntry, numOfEntry);

bool wasBuy = false;

for (int b = 0; b < 13, !wasBuy; b++)

{

int \* nums = new int[sizeOfPath[pos]];

for (int i = 0; i < sizeOfPath[pos]; i++)

nums[i] = i;

int way = choose(nums, chancesToWay[pos], sizeOfPath[pos]);

for (int i = 0; i < numOfPavs; i++)

if (isOnWay(pavs[i], pos, way) && toBuy(chancesToBuy[i]))

{

wasBuy = true;

pavs[i].numOfBuys++;

break;

}

if (paths[pos][way] == -1)

break;

pos = paths[pos][way];

}

}

bool toBuy(int chance)

{

return (rand() % 100) < chance;

}

int choose(int\* arr, int\* chances, int size)

{

int now = -1;

for (int i = 0; i < size - 1; i++)

{

bool thisEl = (rand() % 100) < chances[i];

if (thisEl)

now = arr[i];

}

if (now == -1)

now = arr[size - 1];

return now;

}

bool isOnWay(pavilion pav, int pos, int way)

{

if (pav.paths[pos] == way) return true;

return false;

}

void result(pavilion\* pavs, int numOfPavs)

{

for (int i = 0; i < numOfPavs; i++)

{

cout.width(5);

cout << pavs[i].name;

}

cout << endl;

for (int i = 0; i < numOfPavs; i++)

{

cout.width(5);

cout << pavs[i].numOfBuys;

}

cout << endl;

}

Output:



Вывод: по ходу лабораторной работы познакомился с основами имитационного моделирования.