МИНОБРНАУКИ РОССИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

«ТУЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Кафедра вычислительной техники

Отчет

по лабораторной работе №8

по дисциплине

Моделирование систем

«Исследование имитационной модели склада

механообрабатывающего производства»

Выполнил: \_­­­­­­­­­­­­\_\_\_\_\_\_\_\_\_\_\_ ст. гр. 222151-ПБ Комедин В.А.

Проверил: \_\_\_\_\_\_\_\_\_\_\_\_ доц. кафедры ВТ Семенчев Е.А.

**1. Цель и задачи работы.**

Построение и исследование моделей и алгоритмов производственных систем, организация и проведение машинно­го эксперимента, обработка данных. Развитие творческого подхода к комплексному исследованию.

**2. Задание на работу.**

Находящиеся на складе детали время от времени поставляются для производства некоторого изделия. Ежедневная потребность в деталях не постоянна и описывается как случайная величина D. Эта величина распределена по закону Пуассона:

.

Параметр  - средняя ежедневная потребность в деталях.

Для пополнения запаса деталей в конце каждой недели (неделя пятидневная) склад может сделать заказ на партию из N деталей. Заказанная партия поступает на склад через S дней. Предполагается, что S – случайная величина, закон распределения которой задан.

Для регулирования запаса деталей применяется следующее правило: в конце каждой недели запас сравнивается с фиксированным числом n. Если запас меньше n1, то производится заказ партии деталей, в противном случае заказ не производится.

Моделируя работу склада в течение квартала (13 недель), требуется найти значение n, при котором суммарный ущерб минимален. Начальный запас деталей принять равным М.

Закон распределения дней поставок:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Вариант | Закон распределения | | | | | |
| А | S  P | 1  0,2 | 2  0,2 | 3  0,2 | 4  0.2 | 5  0,2 |

Значения начальных параметров функционирования склада:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Вариант | лямбда | N | M | C1 | C2 |
| 2 | 3 | 20 | 25 | 90 | 60 |

**3. Ход работы.**

**3.1. Схема алгоритма.**

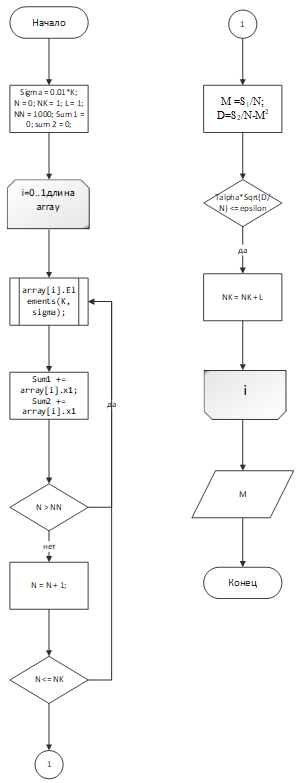


Рисунок 1 – Схема алгоритма получения среднего значения оптимального количества деталей

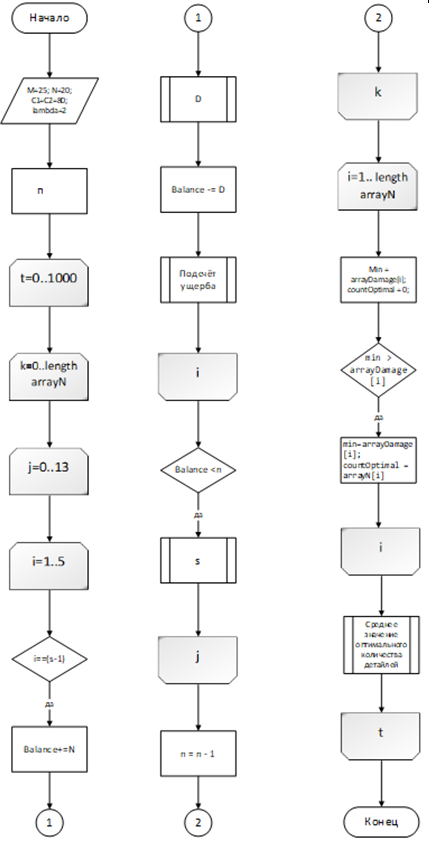


Рисунок 2 - Схема алгоритма функционирования склада механообрабатывающего производства

На рисунке 3 представлен результат работы программы.

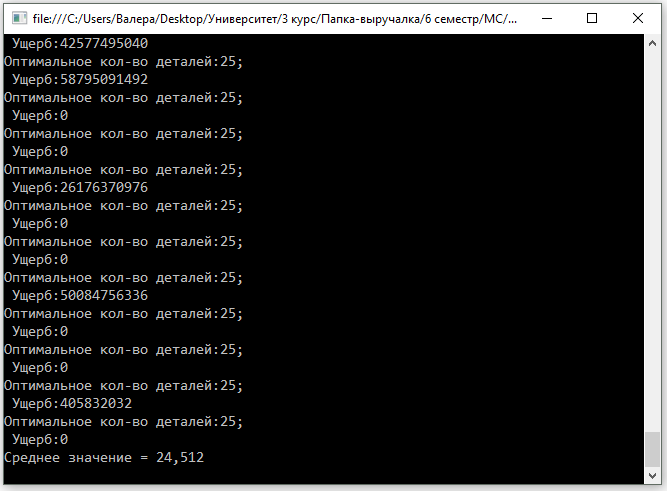


Рисунок 3 – Результат работы программы

**4. Вывод.**

Путем имитационного моделирования работы склада в течение квартала (13 недель) определили значение запаса n, при котором ущерб производству будет минимальным. n = 25.

Модель работы склада механообрабатывающего завода показывает, что оптимальным вариантом выбора количества деталей для сравнения является примерно равной ёмкости склада в конце каждой рабочей недели, иначе ущерб достигает очень больших значений.

Файл Program.cs

using System;

namespace ModelSystem\_7

{

class Program

{

private const double lambda = 3;

private const int N = 20;

private const int M = 25;

static void Main(string[] args)

{

int[] arrayN = new int[M];

double[] arrayDamage = new double[M];

int balance = M;

double sumDamage = 0;

Factorial fact = new Factorial();

int[] D = new int [5];

Puasson K = new Puasson();

DamageCondition damageOneDay = new DamageCondition();

int s = 0;

double sum = 0;

double meanValue = 0;

int n = N; // example for me

for (int t = 0; t < 1000; t++)

{

for (int k = 0; k < arrayN.Length; k++)

{

for (int j = 0; j < 13; j++) // qwarter

{

//one work week

for (int i = 0; i < 5; i++)

{

if (i == (s - 1))

{

balance += N;

}

D[i] = (int)Math.Ceiling(Math.Pow(lambda, K.CountFailure()) / fact.Fact(K.CountFailure()) \* Math.Exp(-1 \* lambda));

balance -= D[i];

sumDamage += damageOneDay.Damage(D[i], M, balance);

}

Random p = new Random();

CountDaysDeliveryOfDetails countDays = new CountDaysDeliveryOfDetails();

if (balance < n) // addDetails

{

s = countDays.CountDays(p.NextDouble()); // get count days

}

else

{

s = 0;

}

}

arrayN[k] = n;

arrayDamage[k] = sumDamage;

n--;

}

double min = 0;

int countOptimal = 0;

for (int i = 0; i < arrayDamage.Length; i++)

{

min = arrayDamage[0];

if (arrayDamage [0] < 0)

{

min = 0;

}

countOptimal = arrayN[0];

if (min > arrayDamage[i])

{

if (arrayDamage[i] > 0)

{

min = arrayDamage[i];

countOptimal = arrayN[i];

}

}

}

if (t > 100)

Console.WriteLine("Оптимальное кол-во деталей:{0};\n Ущерб:{1}", countOptimal, min);

sum += countOptimal;

n = M + 1;

sumDamage = 0;

}

meanValue = sum / 1000;

Console.WriteLine("Среднее значение = {0}", meanValue);

Console.ReadKey();

}

}

}

Файл Puasson.cs

using System;

namespace ModelSystem\_7

{

class Puasson

{

static Random rnd = new Random();

public int CountFailure()

{

int k = 0;

double a = -1.5;

double E = Math.Exp(a);

double s = 0;

do

{

s = rnd.NextDouble();

k++;

}

while ((s - E) > 0);

return k;

}

}

}

Файл Factorial.cs

using System;

namespace ModelSystem\_7

{

class Factorial

{

public double Fact(int parr)

{

if (parr == 0)

return 1;

else

return (parr \* Fact(parr - 1));

}

}

}

Файл DamageCondition.cs

using System;

namespace ModelSystem\_7

{

class DamageCondition

{

private const int C1 = 90;

private const int C2 = 60;

public int Damage(int D, int M, int balance)

{

int damage = 0;

if (balance < D)

{

damage += (C1 \* (D - balance));

}

if (balance > M)

{

damage += (C2 \* (balance - M));

}

return damage;

}

}

}

Файл CountDaysDeliveryOfDetails.cs

using System;

namespace ModelSystem\_7

{

class CountDaysDeliveryOfDetails

{

public int CountDays(double p)

{

int countDays = 0;

if ((p >= 0) && (p < 0.2))

countDays = 1;

else if ((p >= 0.2) && (p < 0.4))

countDays = 2;

else if ((p >= 0.4) && (p < 0.6))

countDays = 3;

else if ((p >= 0.6) && (p < 0.8))

countDays = 4;

else if ((p >= 0.8) && (p <= 1.0))

countDays = 5;

return countDays;

}

}

}