МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ

Лабораторная работа 2

Выполнил:
Анищенко Арсений
4 курс 3 группа
Преподаватель:
Кирлица Валерий Петрович

М и н с к
2019

Условие

Вариант 9 (задание 6.8 страница 179)

По 1000 реализаций оценить характеристики наращенной суммы C: C_{min}, C_{max}, E{C}, D{C}, P{13000 <= C <= 13200}

$n_1 = n_2 = 2$ года
P = 10000
$i_1 = R(0.09, 0.102)$
$i_2 = Tr(0.08, 0.12, 0.09, 0.1)$
$h_i = 2.6 + 0.4i + \varepsilon_i$
$\varepsilon_i = x + \eta$
η: $P(\eta = -0.1) = 0.1$, $P(\eta = 0.2) = 0.1$, $P(\eta = 0) = 0.7$

Ход работы

Теория

Базовую случайную величину можем смоделировать используя генератор Макларена — Марсальи, основанного на мультипликативном конгруэнтном методе:

$$\alpha_i = \alpha_i' / M$$

$$\alpha_i' = \beta\alpha\_i - 1' \bmod M$$

$$i = 1, 2, \ldots$$

$$\alpha_0' = \beta = 65539$$

$$M = 2147483648$$

Методом обратного преобразования получим равномерно распределенную случайную величину:

$$R(a, b) = y(b - a) + a$$

Методом обратного преобразования получим трапецеидально распределенную случайную величину:

$$Tr(a,\ b,\ c,\ d) = \begin{cases} a + \sqrt{y(c-a)(b+d-a-c)},\ y \le \frac{c-a}{b+d-a-c} \\ \frac{y(b+d-a-c)+a+c}{2},\ \frac{c-a}{b+d-a-c} < y \le 1 - \frac{b-d}{b+d-a-c} \\ b - \sqrt{(1-y)(b-d)(b+d-a-c)},\ 1 - \frac{b-d}{b+d-a-c} < y \end{cases}$$

Наращенную сумму без учета инфляции получим следующим образом:

$$S = P(1+i_1)^{n_1}(1+i_2)^{n_2}$$

Инфляцию посчитаем как:

$$j = \prod_{i=1}^{16}(1\ +\ (hi)\ /\ 100)$$

Итоговая наращенная сумма:

$$C = \frac{S}{j}$$

Реализация

```
/// <summary>
/// MacLaren-Marsaglia generator for base random variable
/// </summary>
3 references
public class MGenerator
{
    private double _alpha;
    private double _beta;
    private double _m;

    1 reference
    public MGenerator(int seed = 65539)
    {
        _alpha = seed;
        _beta = 65539;
        _m = 2147483648;
    }

    1 reference
    public double NextRand()
    {
        _alpha = (_alpha * _beta) % _m;

        return _alpha / _m;
    }
}
```

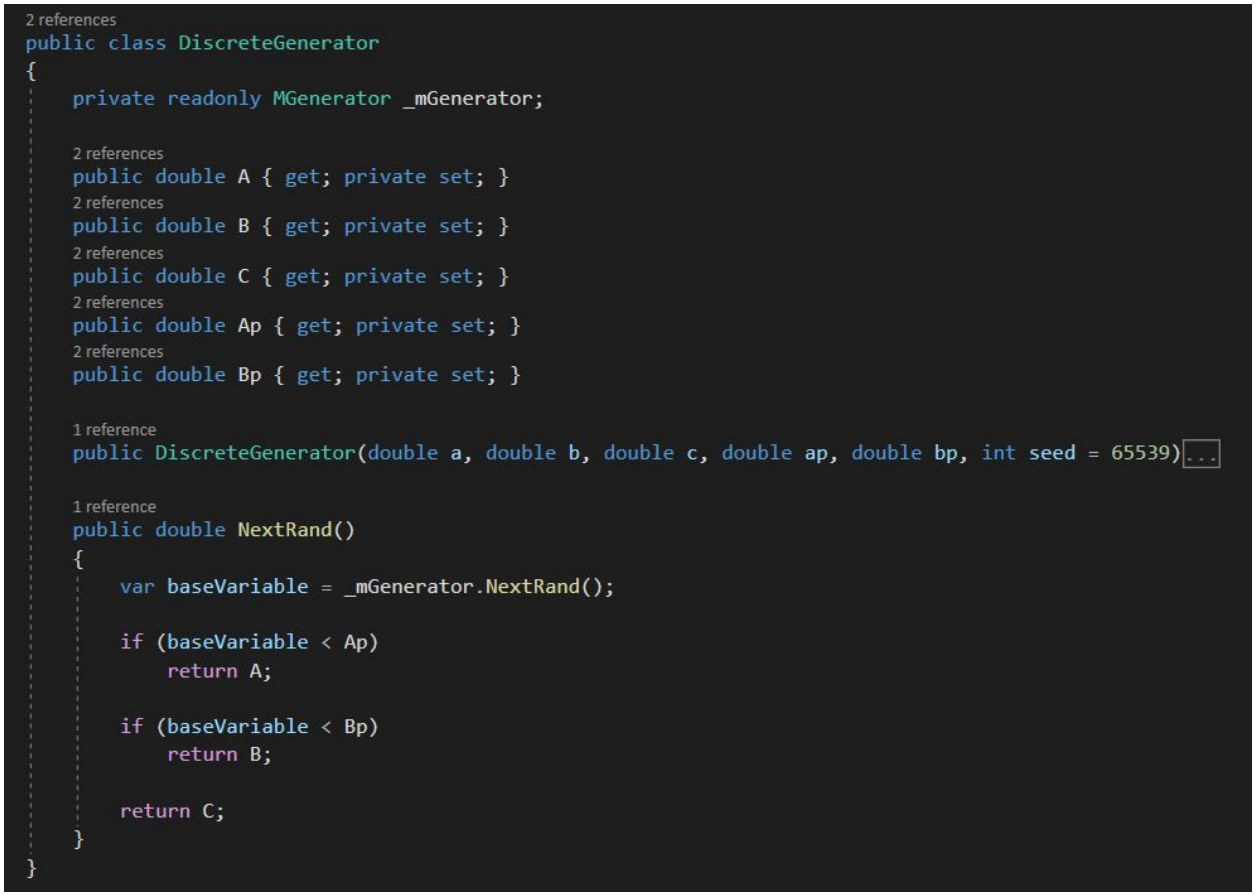
Класс реализующий генерацию базовой случайной величины методом Макларена — Марсальи.

```
/// <summary>
/// Uniform distributed random generator
/// </summary>
2 references
public class UniformGenerator
{
    private readonly MGenerator _mGenerator;

    3 references
    public double A { get; private set; }
    2 references
    public double B { get; private set; }

    1 reference
    public UniformGenerator(double a, double b, int seed = 65539)
    {
        _mGenerator = new MGenerator(seed);

        A = a;
        B = b;
    }

    1 reference
    public double NextRand()
    {
        var baseVariable = _mGenerator.NextRand();

        return baseVariable * (B - A) + A;
    }
}
```

Класс реализующий генерацию св с равномерным распределением.

```
/// <summary>
/// Trapezoidal distributed random generator
/// </summary>
2 references
public class TrapezoidalGenerator
{
    private readonly MGenerator _mGenerator;

    6 references
    public double A { get; private set; }
    4 references
    public double B { get; private set; }
    5 references
    public double C { get; private set; }
    5 references
    public double D { get; private set; }
    5 references
    public double Density...

    1 reference
    public TrapezoidalGenerator(double a, double b, double c, double d, int seed = 65539)...

    1 reference
    public double NextRand()
    {
        var baseVariable = _mGenerator.NextRand();

        if (baseVariable < (C - A) / Density)
        {
            var d = Math.Sqrt(baseVariable * (C - A) * Density);
            return A + d;
        }

        if (baseVariable > 1 - (B - D) / Density)
        {
            var d = Math.Sqrt((1 - baseVariable) * (B - D) * Density);
            return D - d;
        }

        return (baseVariable * Density + A + C) / 2;
    }
}
```

Класс реализующий генерацию св с трапецеидальным распределением.



```
2 references
public class DiscreteGenerator
{
    private readonly MGenerator _mGenerator;

    2 references
    public double A { get; private set; }
    2 references
    public double B { get; private set; }
    2 references
    public double C { get; private set; }
    2 references
    public double Ap { get; private set; }
    2 references
    public double Bp { get; private set; }

    1 reference
    public DiscreteGenerator(double a, double b, double c, double ap, double bp, int seed = 65539)...

    1 reference
    public double NextRand()
    {
        var baseVariable = _mGenerator.NextRand();

        if (baseVariable < Ap)
            return A;

        if (baseVariable < Bp)
            return B;

        return C;
    }
}
```

Класс реализующий генерацию св с дискретным распределением.

```
class Program
{
    private const int P = 10000;
    private const int N1 = 2;
    private const int N2 = 2;

    1 reference
    static List<double> ConductExperiment()
    {
        var uniformGenerator = new UniformGenerator(9, 10.2);
        var trapezoidalGenerator = new TrapezoidalGenerator(8, 12, 9, 10);
        var discreteGenerator = new DiscreteGenerator(-0.1, 0.1, 0, 0.1, 0.2);

        var res = new List<double>();

        for (int it1 = 0; it1 < 1000; ++it1)
        {
            var i1 = uniformGenerator.NextRand();
            var i2 = trapezoidalGenerator.NextRand();
            var s = P * Math.Pow(1 + i1 / 100, N1) * Math.Pow(1 + i2 / 100, N2);
            var h = 2.6;
            var ksi = 0.0;
            var j = 1.0;
            for (int it2 = 0; it2 < 16; ++it2)
            {
                j *= (1 + (h + ksi) / 100);
                ksi += discreteGenerator.NextRand();
                h += 0.4;
            }
            res.Add(s / j);
        }

        return res;
    }

    0 references
    static void Main(string[] args)
    {
        var c = ConductExperiment();

        Console.WriteLine($"Min(C) = {c.Min()}");
        Console.WriteLine($"Max(C) = {c.Max()}");
        var avg = c.Average();
        Console.WriteLine($"E(C) = {avg}");
        Console.WriteLine($"D(C) = {Math.Sqrt(c.Select(x => Math.Pow(x - avg, 2)).Average())}");
        Console.WriteLine($"P(13000 <= C <= 13200) = {c.Count(x => 13000 <= x && x <= 13200)}");

        Console.ReadKey();
    }
}
```

Тестовый класс

Результаты

Min(C) = 5465,50907621395
Max(C) = 6337,44071657794
E(C) = 5914,8702244978
D(C) = 130,038812911029
P(13000 <= C <= 13200) = 0