# О ПРОБЛЕМЕ РЕАЛИЗАЦИИ СЛУЧАЙНЫХ ЧИСЕЛ В C#

***Тазетдинов Б.И., канд. физ.-мат. наук, доцент***

***Мальцев Д.В., канд. хим. наук, доцент***

г. Бирск, ФГБОУ ВПО Бирский филиал БашГУ

**Аннотация.** В работе рассматривается проблема реализации случайных чисел в .NET ( C
#). Рассмотрен генератор псевдослучайных чисел (ГПСЧ). Проведено тестирование разных вариантов применения ГПСЧ.

**Ключевые слова**: генератор случайных чисел, генератор псевдослучайных чисел, криптография, . NET, C#.

Автор: Тазетдинов Б.И., Мальцев Д.В.
28.04.2021 05:00 -

В настоящее время все современные информационные технологии и элементы искусственного интеллекта в той или иной степени используют алгоритмы генерации случайных чисел, а в большей степени это касается теории управления. Так, например случайные числа нужны для генерации паролей, текста капчи, шифрования, генерации сессий. С другой стороны для теории управления описание процессов, которые носят случайный характер и не могут быть заранее точно определены, имеет первостепенное значение [1]. В данной работе рассматривается проблема генерации случайных чисел в наиболее распространенной среде разработке .NET на языке программирования C# [2].

Проблема генерации случайных чисел упирается в аппаратное и программное обеспечение. В языках программирования реализованы генераторы псевдослучайных чисел (ГПСЧ) которые при неумелом использовании могут выдать одинаковый результат. Для получения генератора случайных чисел (ГСЧ) обычно используют ГПСЧ и к нему добавляют источник энтропии.

Рассмотрим как решается описанная выше проблема на примере языка C#. Так в .Net имеется класс
Random
, который является ГПСЧ. Так, например, в нижеприведенном листинге два объекта
Random
создают одинаковую последовательность случайных чисел. В большинстве операционных систем
Windows
созданные во временном интервале до 15 миллисекунд объекты будут иметь одинаковые значения.

```
byte[] bytes1 = new byte[100];
```

```
byte[] bytes2 = new byte[100];
```

```
Random rnd1 = new Random();

Random rnd2 = new Random();

rnd1.NextBytes(bytes1);

rnd2.NextBytes(bytes2);
```

Из вышесказанного следует, что достаточно между инициализациями объектов поставить интервал времени более чем 15 миллисекунд. Тогда наборы, формируемые объектами, будут разными.

```
Random rnd1 = new Random();

Thread.Sleep(20);

Random rnd2 = new Random();
```

Для приложений в крупных системах это не идеальный вариант, так как формирование наборов случайных чисел с задержкой по времени будет замедлять работу алгоритма.

Существуют и другие подходы изменения псевдослучайных чисел. Это привязка к системному времени до миллисекунд (смотрите листинг кода ниже):

```
Random rnd1 = new Random(DateTime.Now.Millisecond);
```

```
Random  rnd2 = new Random(DateTime.Now.Millisecond);
```

Здесь DateTime.Now.Millisecond – свойство, возвращающее текущие миллисекунды объекта DiteTime.

Другой подход – привязка к тактам процессора генератора случайных чисел (ниже представлен листинг кода)

```
Random rnd1 = new  Random((int)DateTime.Now.Ticks);
```

Здесь свойство Ticks возвращает число тактов, который представляет дату и время объекта DateTime. Выведем результат работы свойства Ticks у двух объектов, вызываемых друг под другом на одной электронно-вычислительной машине (смотрите листинг кода ниже):

```
var t1 = DateTime.Now.Ticks;

var t2 = DateTime.Now.Ticks;

Console.WriteLine("{0}t {1}",t1,t2);
```

Результат работы:

```
637519374444572042          637519374444602028
```

Как видно из результата, между двумя присваиваниями прошло около 30000 тактов.

Рассмотрим еще один подход к формированию случайных чисел на основе специализированных библиотек с криптографическими ГСЧ (листинг кода представлен ниже):

```
byte[] randomNumber1 = new byte[1];

byte[] randomNumber2 = new byte[1];

RNGCryptoServiceProvider rnd1 = new RNGCryptoServiceProvider();

rnd1.GetBytes(randomNumber1);

rnd1.GetBytes(randomNumber2);
```

Вышеприведенный код программы формирует два одномерных массива типа byte, состоящих из одного элемента. Класс RNGCryptoServiceProvider реализует механизмы криптографической ГСЧ, предоставляемой поставщиком служб шифрования (cryptographic service provider, CSP). По своей сути этот способ тоже является ГПСЧ, где алгоритмы криптографии вносят энтропию за счет шифрования.

Протестируем время, затрачиваемое на формирование ГПЧС для двух одномерных массивов типа byte, состоящих из 10000000 элементов.

Результаты теста показали следующую картину:

Привязка к тактовой частоте процессора по текущему времени: 182 сек.

Привязка к текущему времени: 367 сек.

Работает криптографическая библиотека: 398 сек.

Из вышеприведенных данных видно, что механизмы формирования случайных чисел с помощью механизмов криптографического ГСЧ наиболее медлительны, по сравнению с остальными. Поэтому их применение актуально в случае необходимости получения защищенных данных, так как алгоритмы шифрования несколько замедляют процесс формирования случайных чисел. Если такой необходимости нет, лучше всего использовать ГПСЧ, привязанный к тактовой частоте процессора.

Список использованных источников:

1. Теория автоматического управления: Учеб. Для вузов по спец. «Автоматика и телемеханика». В 2-х ч. Ч. II. Теория нелинейных и специальных систем автоматического управления./ А.А. Воронов, Д.П. Ким, В.М. Лохин и др.; Под ред. А.А. Воронова. – 2-е изд., перераб. И доп. –М.: Высш. шк., 1986. – 504 с., ил.

2. Шилдт Г. Полный справочник по C#. – Киев: Издательский дом Вильямс, 2004. – 752 с.