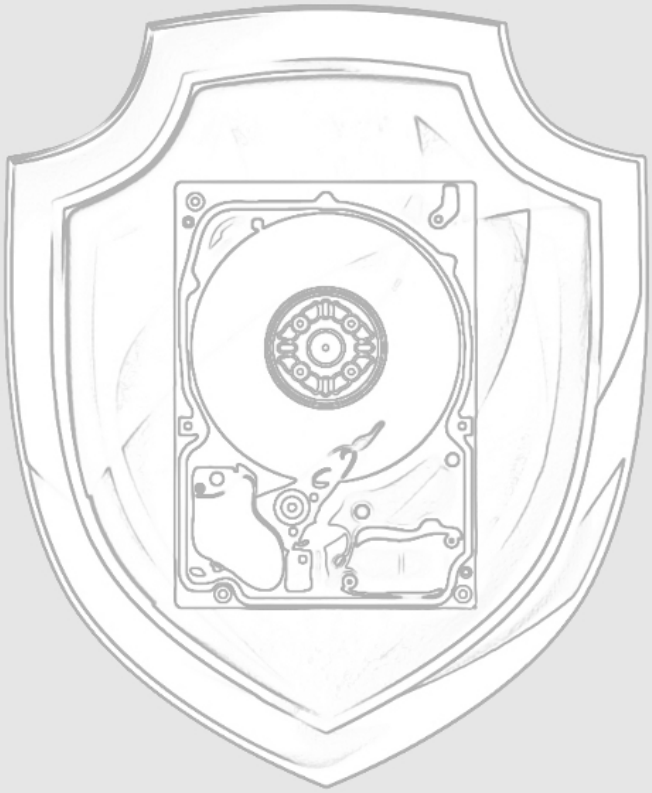
И. Б. ПЕТРОВ



ПРОГРАММНЫЕ АЛГОРИТМЫ ГЕНЕРАЦИИ ПСЕВДОСЛУЧАЙНЫХ ЧИСЕЛ.

2015

Содержание:

От автора.

В данной публикации представлены три несложных авторских программных алгоритма генерации псевдослучайных чисел, которые могут быть реализованы любым, даже начинающим, программистом в своем приложении, в частности, в публикации рассмотрен вариант реализации алгоритмов в такой программной среде разработки, как Microsoft® Visual Basic® .NET.

Все алгоритмы приведенные здесь были апробированы автором и на их основе было создано несколько приложений для генерации паролей и шифрования информации. Также данные алгоритмы были использованы в нескольких криптографических программных продуктах.

Особенность данных алгоритмов заключается в простоте их реализации, надежности и эффективной результативности, достаточной для их применения на практике.

АЛГОРИТМ ГЕНЕРАЦИИ ПСЕВДОСЛУЧАЙНЫХ ЧИСЕЛ НА ОСНОВЕ ПРОСТОЙ ВЫБОРКИ ПО СИСТЕМНОМУ ТАЙМЕРУ (Алгоритм IoaTR).

Данная концепция реализуется при помощи простого программного алгоритма, заключающегося в следующем:

Некая целочисленная переменная i меняет свое значение с «1» на «0» с частотой не менее 100 Гц. Через некоторый интервал времени, отсчитываемый системным таймером, происходит выборка значений данной переменной. При этом величины частоты выборки значений переменной и частоты смены значений переменной i — не являются кратными друг другу, а величина частоты выборки является числом иррациональным. Из-за несовершенства и погрешности измерения математического процессора и таймера, используемого в современных процессорах - итоговая выборка значений будет давать случайный равномерный ряд.

Опытным путем установлено, что оптимальными значениями частот, являются:

– частота смены значений переменных i = 100 Гц или время смены значений = 0,01 сек.

– частота выборки = 1,14... Гц или время выборки = 0,877 сек.

Пример реализации алгоритма на VB.NET:

```
Imports System
Imports System.IO
Public Class Main
   Dim i As Integer ' Задаем переменную как целое число
   Private Sub Main_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
MyBase.Load
      T1.Interval = 10 ' Задаем интервал первому таймеру (0,01 сек.)
      T1.Enabled = True ' Запускаем первый таймер
      T3.Interval = 877 ' Задаем интервал третьему таймеру (0,877 сек.) - значение установлено
опытным путем. Однако главное чтобы частное от деления интервала первого таймер и третьего
таймера было числом иррациональным.
      T3.Enabled = True ' Запускаем третий таймер
   End Sub

   Private Sub T1_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles T1.Tick
      i = 0 ' Задаем переменной значение "0"
      T2.Interval = 10 ' Задаем интервал второму таймеру (0,01 сек.)
      T2.Enabled = True
      T2.Start() ' Запускаем второй таймер
      T1.Stop() ' Останавливаем первый таймер
   End Sub

   Private Sub T2_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles T2.Tick
      i = 1 ' Задаем переменной значение "1"
      T1.Interval = 10
      T1.Enabled = True
      T1.Start() ' Запускаем первый таймер
      T2.Stop() ' Останавливаем второй таймер
   End Sub
```

```
    Private Sub T3_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles T3.Tick
        T1.Stop()
        T2.Stop()
        Dim f As New StreamWriter("Random.itr", True, System.Text.Encoding.GetEncoding(1251)) '
Открываем файл "Random.itr", если поставить не False а True, то запись будет в конец файла. Т.е.
если в файле уже есть текст, то этот текст стираться не будет, а новый будет добавляться в конец.
Если файла не существует, то он создаваться не будет. А если же поставить False, то если в файле
был текст, то он стирается, и запись идет в пустой файл. Если файла не существует, то он
создается. Дальше идет тип кодировки, в нашем случае это стандартная 1251
        f.Write(i) ' Записываем в файл Random.itr текст
        f.Close() ' Закрываем файл
        T1.Start() ' Запускаем первый таймер
    End Sub
End Class
```

АЛГОРИТМ ГЕНЕРАЦИИ ПСЕВДОСЛУЧАЙНЫХ ЧИСЕЛ МЕТОДОМ ВЫБОРКИ ИЗ ЕСТЕСТВЕННОГО РЯДА НАТУРАЛЬНЫХ ЦЕЛЫХ СЛУЧАЙНЫХ ЧИСЕЛ (Алгоритм Мишень).

В основе данного метода лежит принцип ручной (при помощи пользователя) случайной выборки числа из естественного ряда натуральных целых случайных чисел, полученных методом вычисления математических случайных последовательностей (число **e** и **Pi**).

Основу программного алгоритма составляет ряд натуральных целых случайных чисел, из которого по нажатию клавиши проводится случайная выборка числа. В коде VB.NET это выглядит таким образом:

```
Dim sar = "15108657...." ' задаем естественный ряд натуральных целых случайных чисел
  Do
      For i = 0 To sar.Length - 1 ' объявляем цикл вывода данных
          If Not (Me.InvokeRequired) Then
              TTmpG.Text = sar(i) ' выводим в поле по одному символу из ряда по порядку
          Else
              Me.Invoke(New stTtext(AddressOf slt), sar(i).ToString)
          End If
          System.Threading.Thread.Sleep(1) ' задаем временной интервал
      Next
  Loop
```

Как видно из кода, функция sar(i) принимает в течении 1 миллисекунды попеременно значение каждого символа (числа) заданного ряда случайных чисел. Для выборки необходимого символа можно воспользоваться таким алгоритмом:

```
' по нажатию клавиши
Dim n As Integer
   n = Int(26 * Cos(TTmpG.Text) + 26) ' сводим полученное число к интервалу от 0 до 26
   GenGp.Text = GenGp.Text & Chr(65 + n) ' записываем посимвольно латинские буквы в  с их
значением ASCII
```

То есть, при нажатии клавиши пользователем происходит выборка случайного числа из текущего ряда (ведь точно не известно когда нажмет клавишу пользователь и какое при этом будет значение sar(i)). Следующей строкой преобразуем полученное число в значение ASCII и записываем полученный символ таблицы в поле вывода.

Данный числовой ряд, состоящий из целых натуральных чисел, получен при помощи метода ручной обработки числовых рядов, состоящих из первых 1000 знаков (цифр) после запятой, числа e и числа Pi. Метод ручной обработки заключался в случайном перемешивании цифровых последовательностей исходных рядов. Полученный числовой ряд составляет 500 знаков (цифр), случайным образом (методом ручной обработки) выбранных, из полученного путем смешения последовательностей числа e и Pi, числового ряда.

Так как известно, что последовательности чисел e и Pi, представляют собой естественные ряды натуральных целых случайных чисел, то и итоговый ряд, также будет являться множеством случайных чисел, обладающим большей энтропией, чем исходные множества, за счет случайных биологических процессов, протекающих в мозге оператора, производившим перемешивание числовых рядов.

МОДАЛЬНО-ИТЕРАЦИОННЫЙ АЛГЕБРАИЧЕСКИЙ АЛГОРИТМ ГЕНЕРАЦИИ ПСЕВДОСЛУЧАЙНЫХ ЧИСЕЛ (Алгоритм iFk).

Шаг 1. Первоначально определяем псевдослучайные натуральные величины в заданном или неограниченном интервале.

Источником энтропии математического множества величин, из которого производится выборка, может служить среда технико-программного обеспечения вычислительной техники со своим множеством псевдослучайных процессов. Примером является: состояние таймера, случайный ввод символов с клавиатуры, координаты текущего положения курсора и т. п. Однако для упрощения и автоматизации процесса генерации случайного числа методы «ручного» ввода неприменимы. В практике используют уже готовые генераторы псевдослучайных чисел, предоставляемые многими средами программирования.

Рассмотрим пример генерации псевдослучайных величин на примере MS Visual Basic:

```
Randomize() ' определяем начальное значение оператора Randomize для инициализации
            ' генератора случайных чисел
    x = Int(Rnd(Rnd) * 1000000) ' определяем первоначальные значения исходного
                                ' случайного числа от 0 до 1000000
    y = Int(Rnd(Rnd) * 1000000) ' определяем первоначальные значения исходного
                                ' случайного числа от 0 до 1000000
```

В данном случае используются две переменные со случайными величинами в интервале от 0 до 1 000 000. Примечательно, что оператор Randomize в данном случае использует значение системного таймера (исходя из спецификации среды программирования MS Visual Basic), а функция Rnd использует в качестве первоначального значения, генерированное случайное число, как производную от первоначального состояния оператора Randomize.

Шаг 2. Определяем количество будущих итераций как псевдослучайное натуральное число в заданном или неограниченном интервале.

Также используем готовый генератор псевдослучайных чисел на примере MS Visual Basic:

```
Itr = Int(Rnd(Rnd) * 100) ' определяем кол-во иттераций от 0 до 100
```

Шаг 3. Запускаем цикл итераций генерации случайного числа.

```
For i = 1 To Itr ' запускаем цикл итераций генерации случайного числа
```

Шаг 3.1. Определяем по формуле промежуточные переменные.

```
    a = Int((x + x ^ 3 * x ^ Log10(x)) Mod x) ' определяем по ф-ле первую
                                              ' промежуточную переменную
    b = Int((y + y ^ 3 * y ^ Log10(y)) Mod x) ' определяем по ф-ле вторую
                                              ' промежуточную переменную
```

Общая формула для определения промежуточных переменных имеет вид:

$$(x + x^n * Log10(x)) \bmod x$$

где n – любая положительная натуральная величина.
Данная формула позволяет получить равномерное распределение псевдослучайных величин с большей энтропией.

Шаг 3.2. Определяем по формулам промежуточные переменные, как остатки делений и суммы значений переменных всех итераций.

```
c = Int(a Mod b) ' находим промежуточное значение как остаток деления a и b
sum1 = s1 + c ' подсчитываем общую сумму всех итераций значений c
sum2 = s2 + (a + b) ' подсчитываем общую сумму всех итераций значений a и b
```

Шаг 3.4. Определяем по формулам за каждый последующий шаг итераций (начиная со второй итерации) новые случайные значения исходных псевдослучайных величин.

```
x = Int(x + Log10(x) Mod y) ' за каждый последующий шаг итераций (начиная со
                            ' второго шага) определяем по ф-ле новое значение числа x
y = Int(y + Log10(y) Mod x)  ' за каждый последующий шаг итераций (начиная со
                            ' второго шага) определяем по ф-ле новое значение числа y
Next
```

Данная формула позволяет получить равномерное распределение псевдослучайных величин с большей энтропией, что позволяет значительно увеличить энтропию выходных значений всего алгоритма.

Шаг 4. Генерируем итоговое случайное число.

```
num = Int(sum1 Mod sum2) ' генерируем итоговое случайное число
```

Таким образом итоговое значение определяется как остаток от деления сумм итераций. Учитывая то, что в данном алгоритме на разных его шагах энтропия исходных псевдослучайных натуральных величин, определенная функцией среды программирования, возрастала, то следовательно — итоговое значение обладает энтропией во много раз большей, что увеличивает его качественные значения для использования получения псевдослучайной величины в криптографии.

Итоговая величина num принимает вид равномерного ряда псевдослучайных значений с очень высокой степенью энтропии.