УДК 004.4

ИСПОЛЬЗОВАНИЕ ВНЕШНЕЙ ЭНТРОПИИ В ГЕНЕРАТОРАХ СЛУЧАЙНЫХ ЧИСЕЛ

И.Гнатюк
Северо-Казахстанский государственный университет им.М.Козыбаева, Петропавловск

Научный руководитель - В.П.Куликова

Всем правит случай. Знать бы еще, кто правит случаем.
Жан Кокто
Всем правит бал Его Величество Случай, которым управляет Ее Высочество Закономерность.
Леонид С. Сухоруков
Счастливому случаю нужно помочь, если уж удалось его организовать.
Правило Дюшарма

Предлагается разнообразить идеи алгоритмов ГСЧ [1] внешними источниками энтропии. Для реализации проекта выбрана интегрированная среда разработки Microsoft Visual Studio 2010, и язык Visual Basic.NET.

На форме проекта размещены рядом друг с другом 56 кнопок-квадратиков (рисунок 1). К каждой кнопке привязано системное событие *MouseEnter* (действие, которое срабатывает, когда курсор попадает в зону видимости кнопки, иначе говоря – когда курсор «задевает» кнопку любой своей частью).

Рисунок1. Окно программы с выводом статистических данных

В обработчике событий каждой кнопки находится код, который получает системное время в миллисекундах. При наведении курсора на кнопку в памяти сохраняется некоторое число в пределах 0 ÷ 999. При повторном проведении курсором над той же кнопкой сохраняется уже новое число. Таким образом, пользователь произвольно проводит курсором над полем кнопок, формируя серию чисел. Рекомендуется «поработать» (провести курсором над кнопками) минимум над 1/2 поля, чтобы получить «лучшую» последовательность.

Вне зависимости от того, над сколькими квадратиками был проведен курсор, в результате всегда будем получать 56 цифр. «Недостающие» цифры появляются следующим образом:

- Создается 2 массива b и c размерностью в 56 элементов каждый. Поочередно рандомно (встроенной функцией Basic *Rnd*) заполняются массивы b и c; каждый раз, до и после заполнения очередного элемента, инициируется сброс счетчика рандомизации, иначе каждое следующее значение зависело бы от предыдущего.

- Фрагмент программного кода (рисунок 2) иллюстрирует алгоритм заполнения массива.

```
For j = 0 To 55
    Randomize()
    znak = Rnd() * 5

    If znak = 0 Then
        a(ii) = aAA(j) + b(j)
        If a(ii) > 999 Then
            While a(ii) > 999
                a(ii) = (a(ii) / 10) + Rnd() * 42
            End While
        End If

    ElseIf znak = 1 Then
        a(ii) = aAA(j) - b(j)
        If a(ii) < 0 Then
            a(ii) = a(ii) * (-1)
        End If
        If a(ii) > 999 Then
            While a(ii) > 999
                a(ii) = (a(ii) / 10) + Rnd() * 42
            End While
        End If

    ElseIf znak = 2 Then
        a(ii) = aAA(j) * b(j)
        If a(ii) > 999 Then
            While a(ii) > 999
                a(ii) = (a(ii) / 10) + Rnd() * 42
            End While
        End If

    ElseIf znak = 3 Then
        a(ii) = aAA(j) / (b(j) + Rnd() * 9)
        If b(j) = 0 Then
            b(j) = Rnd() * 999
            If a(ii) > 999 Then
                While a(ii) > 999
                    a(ii) = (a(ii) / 10) + Rnd() * 42
                End While
            End If
        End If

    ElseIf znak = 4 Then
        a(ii) = Now.Millisecond
    End If
Next
```

Рисунок 1. "Резервное" заполнение массива

Каждый раз, при очередном «резервном» заполнении массива, вызывается функция *Randomize()*, которая в качестве исходной точки берет значение системного таймера, а не предыдущее значение *Rnd*. То есть каждый раз, нажимая красную кнопку, пользователь инициирует заполнение массивов b и c заново, разными значениями, не зависящими друг от друга.

После того, как пользователь достаточно «поелозил» по полю, он нажимает на продолговатую кнопку (рисунок 1) и получает результат.

Программа позволит пользователю увидеть результат работы только в том случае, если он «задействовал», как минимум, половину кнопок поля, в противном случае он получит сообщение о необходимости «задействовать» большее число «реагирующих» кнопок. Реализованная программа позволяет скопировать полученные данные в буфер обмена. Также пользователь может получить статистические данные по сгенерированной последовательности, такие как число запусков генерации, распределение чисел по интервалам, сумму данной серии чисел, нажав на кнопку «Получить стат. данные».

Кнопки-квадратики расположены в хаотическом порядке, то есть первая кнопка (если ее задействовать курсором) необязательно определяет первое число последовательности, и т.д., что дополнительно улучшает «случайность», поэтому даже если построчно проводить по полю, с допустимо большой скоростью, нельзя будет определить системность полученной последовательности. Невозможно предсказать, как поведет себя пользователь, в какой последовательности он будет водить курсором над квадратиками, какие задействует, какие нет. Следует заметить, что при повторе одного и того же алгоритма движений курсора, будем получать разные цифры: невозможно попасть в тот же временной промежуток, в котором «рисовалась» предыдущая траектория курсора, так как человеческая реакция просто не способна на такое. Таким образом, качество предлагаемого генератора обусловлено, в том числе, внешним источником энтропии:

- произвольностью расположения кнопок на форме;
- произвольностью их выбора пользователем;
- непредсказуемостью временного интервала этого выбора.

Данный генератор тестировался на качество работы [2]:

- случайность;
- равномерную распределенность;
- статистическую независимость.

Тестирование проведено неоднократно, разными людьми, в разное время. Применялись тесты: критерий пиков, тестирование равномерного распределения: математическое ожидание, дисперсия, среднеквадратическое отклонение, частотный тест, проверка по критерию «χ-квадрат», проверка на статистическую независимость, отсутствие автокорреляции. Большинство тестов показали надежность данного генератора (не менее 90%), χ-квадрат тест показал 99% надежности, тестирование отдельных статистик всегда показывало результаты, близкие к эталонным для равномерного распределения.

Перспектива: усовершенствование алгоритма, путем усложнения «резервного способа генерации» (работа массивов b и c).

Литература

1. Дональд Э. Кнут. Глава 3. Случайные числа // Искусство программирования = The Art of Computer Programming. — 3-е изд. — М.: Вильямс, 2000. — Т. 2. Получисленные алгоритмы. — 832 с.

2. М. А. Иванов, И. В. Чугунков. Методика оценки качества генераторов ПСП // Теория, применение и оценка качества генераторов псевдослучайных последовательностей. — М.: КУДИЦ-ОБРАЗ, 2003. — 240 с.