Школа современного программирования

Паньгина Нина Николаевна,
Паньгин Андрей Александрович

СТАТИСТИЧЕСКОЕ МОДЕЛИРОВАНИЕ: МЕТОД МОНТЕ-КАРЛО

ВВЕДЕНИЕ

Тема моделирования обычно изучается в школе на примере адекватного представления процессов (например, движения) по выбранным приближенным уравнениям. Статистическое моделирование, наоборот, не предполагает изначально знание математических связей и позволяет получить их на основе многократного наблюдения (компьютерной генерации) возможных событий в представленной модели. Мы предлагаем методическую разработку темы с целью продемонстрировать насколько описываемый метод является мощным и универсальным инструментом для решения различных задач во многих областях знаний.

Изучение данного материала возможно, например, на занятиях по подготовке к олимпиадам по программированию [1]. С этой целью выбраны некоторые задачи, предлагавшиеся на олимпиадах Ленинградской области по информатике.

Подборка задач обеспечивает совместное изучение темы школьниками различных классов, имеет игровой и занимательный характер, иллюстрирует различное применение метода. В ходе разбора задач указаны некоторые приемы программирования, проводится начальное знакомство с важными понятиями теории вероятности, лежащей в основе данного метода. Для освоения темы приводятся задачи для самостоятельного решения с указаниями.

ОБЩАЯ СХЕМА МЕТОДА

Метод статистического моделирования, или метод Монте-Карло, назван так в честь столицы княжества Монако, известной своими многочисленными казино, в которых публика растрачивает или увеличивает свои доходы согласно законам распределения случайных величин.

Этот метод позволяет решать задачи, в условиях которых присутствует элемент неопределенности (например, при подбрасывании монеты может выпасть «орел» или «решка»). Пусть требуется найти некоторую величину (например, долю выпадения «орлов»). На ЭВМ с помощью генератора (датчика) случайных чисел (ДСЧ) имитируются ситуации или процессы, возможные по условию задачи и приводящие к тем или иным исходам. При этом искомая величина принимает некоторые значения (в нашем примере это 0 – если выпал «орел» и 1 – в противном случае). Все или почти все исходы (с учетом, когда монета может упасть на ребро) проявятся, если многократно рассмотреть случайное развитие одного и того же начального состояния (смоделировать некоторое количество N историй).

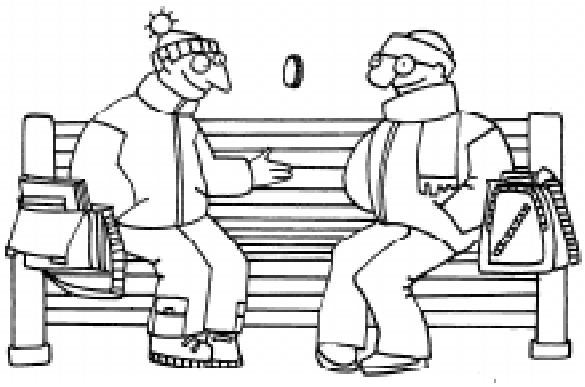
Закон больших чисел «разыгрываемых» историй утверждает, что среднее арифметическое полученных в каждом розыгрыше значений исследуемой величины имеет предел при бесконечном увеличении числа N (в нашей задаче оно равно 1/2).

Это вероятностная сходимость, то есть, чем больше историй, тем достоверней можно утверждать, что наш результат близок к истинному. Для задач с элементами неопределенности – а в реальном мире все задачи такие – это даже естественно. Погрешность определения предельного значения пропорциональна $1/\sqrt{N}$. Таким образом, для увеличения точности результата на один порядок, требуется разыграть в 100 раз больше историй.

Следовательно, требуется получать много случайных чисел так, чтобы переход от одного числа к другому определялся простыми правилами, но чтобы сами числа «производили впечатление случайности» (их называют поэтому псевдослучайными числами). Например, для выбора последовательности случайных цифр можно взять дробную часть числа пи ($\pi = 3.14159265358979323846264338327950288419716939937510582097494459230781640628620899862803482534211 7$... Представляет интерес и обратное утверждение: может ли любая конечная наперед заданная последовательности цифр быть частью бесконечного представления числа π).

Данный способ, однако, мало приемлем для программирования. Как правило, при решении задач методом Монте-Карло используются процедуры, которые с помощью рекуррентных формул генерируют случайные числа, равномерно распределенные на промежутке [0, 1]. В языке Pascal для этого используется стандартная функция RANDOM. Для отладки программы бывает важно уметь воспроизвести псевдослучайные числа, а для генерации другой последовательности случайных чисел используется процедура RANDOMIZE.

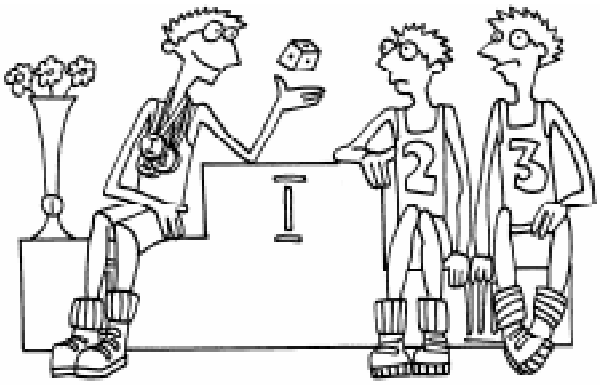
Очень полезно для понимания данного метода моделирование игровых вероятностных ситуаций (бросание монеты или кубика, случайные блуждания). Именно «игровая» или связанная с чем-то знакомым (известным, «бытовым») формулировка задачи помогает лучше усвоить метод, осмыслить понятие вероятности.



...монета может упасть на ребро

Пример 1. (Районная олимпиада 1994).

Три игрока (с номерами 1, 2 и 3), имеющие изначально *X, Y* и *Z* жетонов, соответственно, играют в следующую игру. В каждом раунде каждый игрок ставит на кон один жетон. Затем бросают кубик, на котором цифры 4, 5, 6 заменены на 1, 2 и 3. При выпадении числа *i* игрок с номером *i* забирает с кона все три жетона. Игра заканчивается, когда кто-нибудь из игроков проигрывает все жетоны. Введем функцию $f(X, Y, Z)$ как среднюю длительность игры (среднее количество раундов) при заданных начальных капиталах *X, Y, Z*. Например, $f(2, 2, 2) = 2$. Ваша задача состоит в том, чтобы определить эту функ-



Три игрока (с номерами 1, 2 и 3)... играют в следующую игру.

цию. Для этого необходимо смоделировать игру на компьютере, накопить экспериментальные результаты, проанализировать их, а затем выдвигать гипотезы о виде функции f, проверять их для разных входных значений и, отбросив неподходящие, найти решение.

Моделирование игры не вызывает трудности (программа приведена ниже). Очевидно, что вид функции симметричен относительно порядка задания входных параметров $f(X, Y, Z) = f(Y, X, Z)$ и т. д. Сложность задачи заключается в нахождении вида функции $f(X, Y, Z) = X \cdot Y \cdot Z / (X + Y + Z - 2)$, так как результаты моделирования определяются не точно.

```
var
 i, n, Rounds: LongInt;
 x, y, z, j: Integer;
 a: array[0..2] of Integer;
begin
 Write('x, y, z натуральные: ');
 ReadLn(x, y, z);
 Write('Число историй (игр): ');
 ReadLn(n);
 Rounds:=0;  {число раундов}
             {в одной игре}
 Randomize;
 for i:=1 to n do {разыгрываем}
           {n историй развития}
           {начальной ситуации}
 begin
  a[0]:=x; a[1]:=y; a[2]:=z;
  while a[0]*a[1]*a[2]<>0 do
   {моделируем игру,}
   {пока у всех игроков есть жетоны}
  begin
   Inc(Rounds);
   for j:=0 to 2 do Dec(a[j]);
              {все игроки ставят}
              {по одному жетону}
   Inc(a[Random(3)], 3);
              {а победитель берет}
              {все три жетона}
  end;
 end;
 WriteLn('f(x,y,z) = ', Rounds/n:0:3);
       {среднее количество раундов}
end.
```

Данная задача достаточно хорошо характеризует метод Монте-Карло, а именно:

- **Идею метода:**

ожидаемый результат игры может быть оценен усреднением результатов большого числа игр (это число так и называется – математическим ожиданием или средним значением).

То есть результат приближенно равен числу $\bar{x} = \frac{1}{N} \cdot \sum_{i=1}^{N} x_i$,

где x_i – результат игры i, а N – число всех проведенных игр (испытаний)

- **Достоинство метода:**

незнание *a priori* (до опыта) функциональных зависимостей исследуемой задачи в целом, выявление этих зависимостей *a posteriori* (после опыта).

- **Недостатки метода:**

– неопределенное время расчета (варианты примера 1 при больших числах X, Y, Z);

– приближенное вычисление результата.

Последний недостаток компенсируется тем, что с использованием данного метода вместе со значением $\bar{x}$ может одновременно определяться и его погрешность $S_{\bar{x}}$ по формулам:

$$S_{\bar{x}}^2 = \frac{S^2}{N}, \quad S^2 = \frac{\sum_{i=1}^{N} (x_i - \bar{x})^2}{N-1},$$

При больших N формулу можно упростить:

$$S^2 \approx \overline{x^2} - \bar{x}^2,$$

где $\overline{x^2} = \frac{1}{N} \cdot \sum_{i=1}^{N} x_i^2$.

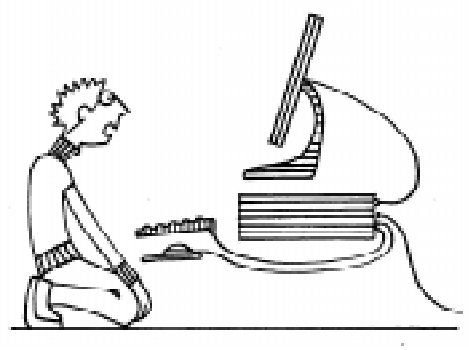
В пределах $[\bar{x} - S_{\bar{x}}, \bar{x} + S_{\bar{x}}]$ с достоверностью 68.27 % находится искомая величина, а в пределах $\bar{x} \pm 2S_{\bar{x}}$ или $\bar{x} \pm 3S_{\bar{x}}$ достоверность уже 95.45% и 99.73% соответственно. Поэтому метод по праву называют порой прецизионным, или точным, в том смысле, что известна точность рассчитываемых величин, и это может служить точкой отсчета для проверки программ, использующих другие приближенные методы.

Иногда, чтобы избежать потери значащих цифр при суммировании, среднее значение определяется в программе после каждого испытания по формуле:

$$\bar{x}_n = \frac{(n-1)\cdot \bar{x}_{n-1}}{n} + \frac{x_n}{n}.$$

Так как программа с использованием метода Монте-Карло порой требует значительных временных затрат, целесообразно выводить на экран некоторую информацию о ходе ее решения во избежание мнимого эффекта «зависания», за исключением тех случаев, когда вывод на экран запрещен по условию задачи. Предпочтительно для описания целочисленных переменных, осуществляющих подсчет историй, использовать тип «длинное целое».

Метод Монте-Карло применяется *для выбора наилучших стратегий* в задачах, где присутствуют много случайных факторов.



...во избежание мнимого эффекта «зависания»,

Задача 1. «Лучшее пари для простаков». (Районная олимпиада 1997).

Игрок А выбирает комбинацию из цифр 0 и 1 длиной 3 знака (например, 001). Игрок В выбирает свою комбинацию (отличную от игрока А). Подбрасывается монета и записываются результаты бросания (например, 101101..., где 0 обозначает «орел», а 1 – «решка»). Игра прекращается в тот момент, когда в последовательности цифр на конце возникает комбинация, выбранная А или В (побеждает А или В, соответственно). Игра повторяется.

а) Оценить шансы на выигрыш каждого из игроков R(A,B) (то есть отношение числа выигрышей игрока В к числу выигрышей игрока А).

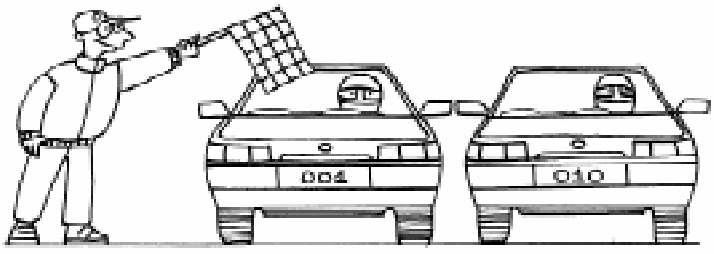
б) Для выбранной игроком А комбинации определить такую комбинацию для игрока В, которая ему дает больше шансов на выигрыш.

В таблице 1 представлены значения R(A,B) для всевозможных выбранных игроками А и В исходных комбинаций при «неограниченном продолжении» игры (выделены наиболее выигрышные ситуации для игрока В).

Пари является беспроигрышным (!) для игрока В. Парадокс заключается в том, что какую бы комбинацию цифр не выбрал игрок А, его соперник В может выб-

Таблица 1

A / **B**	**000**	**001**	**010**	**011**	**100**	**101**	**110**	**111**
000	–	1	2/3	2/3	1/7	5/7	3/7	1
001	1	–	**2**	**2**	1/3	5/3	1	7/3
010	3/2	1/2	–	1	1	1	3/5	7/5
011	3/2	1/2	1	–	1	1	**3**	**7**
100	**7**	**3**	1	1	–	1	1/2	3/2
101	7/5	3/5	1	1	1	–	1/2	3/2
110	7/3	1	5/3	1/3	**2**	**2**	–	1
111	1	3/7	5/7	1/7	2/3	2/3	1	–



«Лучшее пари для простаков»

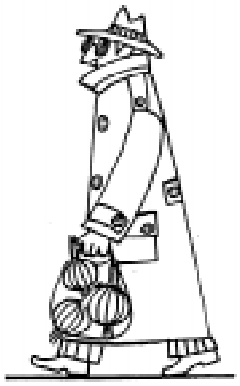
рать другую комбинацию, которая ему дает больше шансов на выигрыш.

Указание. При решении задачи полученные результаты по пункту а) не будут совпадать с данными из таблицы, так как число опытов ограничено, тем не менее, позволяют дать качественный ответ по пункту б).

Метод Монте-Карло используется *для определения вероятности наступления какого-либо события*.

Задача 2.

Пусть дана ось с отмеченными на ней целочисленными точками. Предположим, что Чиполлино спрятался в точке 0, в точке N находится пропасть, и сыщик Моркоу находится в точке k $(0 < k < N)$. Сыщик ищет Чиполлино случайным образом, блуждая по соседним целочисленным точкам. Если он попадет в точку 0, то найдет Чиполлино, а если попадет в точку N, то свалится в пропасть. С какой вероятностью сыщик найдет Чиполлино?



С какой вероятностью сыщик найдет Чиполлино?

Под вероятностью P какого-либо события мы будем понимать предельное значение частоты события, а именно, отношения числа успешных (приведших к появлению данного события) испытаний (N_y) к общему числу проведенных испытаний (N), то есть $P \approx N_y / N$. Чем больше мы проведем испытаний, тем точнее (в идеале) мы определяем численное значение вероятности. Очевидно, что вероятность P удовлетворяет условию: $0 \le P \le 1$.

Указание. Смоделируйте многократный поиск сыщика из точки k. Доля удачных попыток от общего их числа дает приближенную оценку искомой вероятности. На основании этой оценки сформулируйте простую формулу для нахождения вероятности события (обозначим ее $P_0(k)$), указанного в задаче.

Данную задачу можно решить точно, используя рекуррентную формулу:

$$P_0(k) = P_0(k-1)/2 + P_0(k+1)/2$$

при очевидных условиях, что $P_0(0) = 1$ и $P_0(N)=0$, однако использовать рекурсию в такой форме неприемлемо, так как глубина рекурсии не ограничена, что ведет к переполнению стека и краху программы.

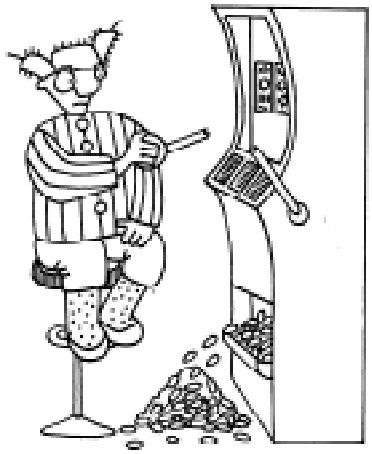
Упражнение 1.

Выведите рекуррентную зависимость $P_0(k+1)$ от $P_0(k)$, начиная с $k = 0$ и, используя известное значение $P_0(N)$, обратным ходом получите общее выражение для $P_0(k)$.

Задача 3.

Пусть имеется «однорукий бандит» – игровой автомат с ручкой, которой его запускают для игры. Считаем (для упрощения), что игра будет типа игры «в орлянку» и игрок имеет начальный капитал в одну монету. Игра ведется до тех пор, пока игрок не обанкротится или выиграет N монет. Промоделировать игру для $N = 10$, определить вероятность (шансы) игрока «сорвать указанный куш» и объяснить, почему автомат назвали «бандитом».

Задача о разорении игрока аналогична задаче блуждания по отрезку $[0, N]$ с той лишь разницей, что требуется определить вероятность $P_N(k)$



...определить вероятность (шансы) игрока «сорвать указанный куш»...

достичь точки N, находясь в точке $k = 1$. Образно говоря, эти модели «связаны одной цепью». Последовательность испытаний, в которой каждое следующее испытание зависит только от исхода предыдущего, называется цепью Маркова. Многие реальные явления (например, броуновское движение частиц, обслуживание телефонных линий) описываются данными вероятностными моделями [2].

Метод Монте-Карло *универсален и применим* как для задач, в условиях которых присутствует элемент неопределенности, так *и для полностью детерминированных задач*.

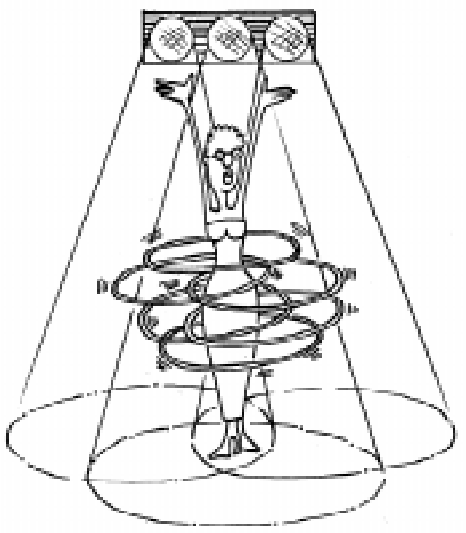
Иногда трудно найти алгоритм или функциональные зависимости для решения сложных задач, однако возможно переформулировать условие задачи таким образом, чтобы использовать для нахождения решения метод Монте-Карло. При этом задача упрощается, но за это приходится «расплачиваться» временем решения и точностью результата.

Пример 2. (Областная олимпиада 2001).

Найти площадь пересечения трех окружностей с заданными радиусами и координатами центров окружностей.

Аналитические выкладки для определения площади пересечения достаточно сложны. Метод Монте-Карло позволяет приближенно вычислить площадь (объем) области, даже в том случае, когда имеется лишь возможность определить, принадлежит ли точка данной области.

Переформулируем условие задачи. Опишем квадрат около одной из окружностей (например, меньшего радиуса). Будем случайным образом кидать точки в этот квадрат. При достаточно большом их количестве (n) они равномерно распределятся по площади квадрата. Часть из них (m) попадет в область пересечения трех окружностей. Можно ожидать, что отношение m/n имеет конечный предел, равный отношению искомой площади к площади описанного квадрата (см. упражнение 2).



...площадь пересечения трех окружностей...

Указание. Наилучший путь – это «использовать геометрию» для анализа частных случаев (когда нет пересечения, одна окружность внутри другой), а метод Монте-Карло – для общего случая.

```
label
 NotInCircle;
var
 i, n, m: LongInt;
 j: Integer;
 x, y, r, rr: array[1..3] of Real;
 xp, yp, xmin, ymin, d: Real;
begin
 for j:=1 to 3 do {вводим координаты}
   {центра и радиус трех окружностей}
 begin
 Write(j, '-я окружность (x y r): ');
 ReadLn(x[j], y[j], r[j]);
 rr[j]:=Sqr(r[j]); {вычислим}
        {квадраты радиусов - }
        {они будут часто}
        {использоваться}
 end;
 Write('Количество историй: ');
                          ReadLn(n);
 xmin:=x[1]-r[1]; {опишем квадрат}
          {около первой окружности}
 ymin:=y[1]-r[1];
 d:=r[1]*2;
 m:=0;
 Randomize;
 for i:=1 to n do {в цикле по числу}
                  {историй}
```

```
begin
  xp:=Random*d+xmin; {бросаем}
                  {случайным образом}
  yp:=Random*d+ymin; {точки}
                {в выбранный квадрат}
  for j:=1 to 3 do {проверяем,}
                {попадает ли точка}
                {в каждый круг}
   if Sqr(xp-x[j])+Sqr(yp-y[j]) > rr[j]
    then goto NotInCircle;
 Inc(m); {считаем количество точек,}
  {попавших сразу во все три круга}
  NotInCircle:
end;
WriteLn('S = ', Sqr(d)*m/n:0:3);
          {результат, тем точнее,}
          {чем больше историй}
end.
```

Задача 4. Два цилиндра одинакового радиуса $R = 1$ пересекаются под прямым углом. Найти объем V их общей части.

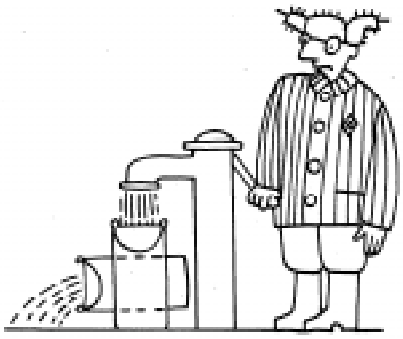
Указание. В журнале «Квант» (№ 2, 1988 г.) приводится точное геометрическое решение задачи: $V = 16R^3/3$.

Задача 5. Оценить, чего больше: несократимых или сократимых дробей.

Фривольное условие задачи предполагает строгую формулировку: какова вероятность того, что наудачу взятая дробь несократима?

Ответ достаточно сложен и равен $6/\pi^2 = 0.6079...$ (Н.Я. Виленкин. В таинственном мире бесконечных рядов. Квант № 10, 1989).

Сформулируем условие иначе. Рассмотрим несократимые дроби вида a/b, где $1 \le a, b \le N$. Количество их $f(N)$. Нужно найти предел $f(N) / N^2$ для больших чисел N. Выберем случайные натуральные числа (не превосходящие фиксированного достаточно большого числа N) для числителя и знаменателя дроби. Повторяем «эксперимент» n раз, подсчитывая количество m несократимых дробей, используя алгоритм Эвклида для нахождения наибольшего общего делителя числителя и знаменателя. Отношение m/n дает оценку доли несократимых дробей.



Два цилиндра одинакового радиуса ... пересекаются под прямым углом.

Строго говоря, мы должны доказать, что характер соотношения не изменится при увеличении числа N. Мы же будем это предполагать во всех задачах, так как математические критерии, гарантирующие сходимость решения, известны в редких случаях. Сходимость можно проверять для различного числа испытаний (например, увеличивая его в 10 раз). Важно, чтобы число историй было «большим».

Упражнение 2.

Свойство равномерного распределения случайных чисел на отрезке [0, 1] означает, что вероятность попасть очередному числу, сгенерированному ДСЧ, в любой выбранный отрезок из [0, 1] равна длине этого отрезка (проверьте моделированием).

Поэтому смоделировать событие (обозначим его С), реализующееся с вероятностью Р, можно так: на единичном отрезке выбирается отрезок длины Р, и если случайная точка попала в заданный отрезок, то считаем, что событие С произошло. Если есть несколько независимых событий, то им следует сопоставить непересекающиеся отрезки с длинами, соответствующими вероятностям событий.

Равномерность распределения точек по отрезку справедлива для большого числа сгенерированных случайных точек. Так, если мы разобьем единичный отрезок на k равных частей, то необходимо сгенерировать более $10 \cdot k$ случайных чисел, чтобы они распределились в каждой части примерно одинаково. Для k случайных точек получим совершенно иную картину распределения.

Пример 3.

На шахматную доску случайным образом бросают 64 зерна. Определить, как зерна по количеству распределятся в клетках.

Указание. Пронумеруем клетки шахматной доски от 0 до 63. Случайное попадание зерна на какую-либо клетку моделируем случайным выбором клетки с помощью датчика случайных чисел RANDOM(64).

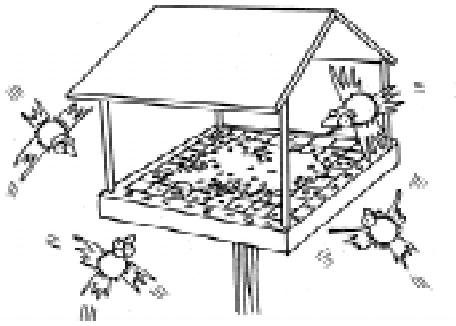
```
const
 Iterations = 10000;
var
 Board: array[0..63] of LongInt;
 Count: array[0..63] of LongInt;
 i, j: LongInt;

begin
 Randomize;
 FillChar(Count, SizeOf(Count), 0);
 for i:=1 to Iterations do
 begin
  FillChar(Board, SizeOf(Board), 0);
  for j:=0 to 63 do
           Inc(Board[Random(64)]);
  for j:=0 to 63 do
             Inc(Count[Board[j]]);
 end;

 for j:=0 to 10 do
 WriteLn(j:2, ' ',
         Count[j]/Iterations:8:5);
end.
```

В результате моделирования оказывается, что вероятнее всего 23 клетки останутся пустыми, на 24 клетках будет по одному зерну, на 12 клетках – по два, на 4 клетках – по три, на одной клетке – четыре.

Генератор случайных чисел можно использовать для построения различных геометрических объектов.

Приведем алгоритм построения простейшего лабиринта. Лабиринты служат основой многочисленных игровых программ и олимпиадных задач.

На плоскости чертится прямоугольник, задающий границы лабиринта. Внутри прямоугольника выбирается точка (координаты которой задаются случайным образом), не лежащая на ранее построенных границах. От точки в случайном направлении (вправо, влево, вверх, вниз) рисуется линия границы до пересечения с какой-либо другой линией. Чтобы проходы в лабиринте были одинаковой ширины, координаты точки задаются с заранее выбранным шагом (например, на целочисленной сетке). Построение лабиринта прекращается по нажатию клавиши <ESC> или когда выбраны все допустимые точки. Такой алгоритм построения не дает циклических путей в лабиринте и, следовательно, в нем всегда можно найти выход. На рис. 1 приведен вариант сгенерированного лабиринта.



На шахматную доску случайным образом бросают 64 зерна.

Пример 4.

Построим лабиринт, используя приведенный выше алгоритм.

```
uses
 Graph, Crt;

const
 Step = 20;    {шаг - размер клетки}
               {в пикселах}
 Width = 30;   {ширина лабиринта}
               {в клетках}
 Height = 20;  {высота лабиринта}
               {в клетках}
 dx: array[0..3] of Integer = (1, 0, -1, 0);
          {смещения по горизонтали}
 dy: array[0..3] of Integer = (0, -1, 0, 1);
{и вертикали для четырех направлений}

var
 Driver, Mode: Integer;
 x, y, n: Integer;
 Wall: Boolean;

begin
 Driver:=Vga; Mode:=VgaHi;
```

```
 InitGraph(Driver, Mode, '');
 {очертим границы лабиринта}
Rectangle(0,0,Width*Step, Height*Step);
 Randomize;
 repeat
  x:=Random(Width)*Step;
       {выбираем случайную точку}
  y:=Random(Height)*Step;
  if GetPixel(x, y)<>0
  then Continue; {если она лежит}
       {на стене, пробуем заново}
  MoveTo(x, y);
  n:=Random(4); {выбираем}
       {случайное направление}
  repeat {рисуется линия}
       {от текущей точки}
   Inc(x, dx[n]*Step);
       {с заданным шагом}
       {и направлением}
   Inc(y, dy[n]*Step);
   Wall:=(GetPixel(x, y)<>0);
       {до ближайшей стенки}
   LineTo(x, y);
  until Wall;
 until KeyPressed;
 ReadKey;
 CloseGraph;
end.
```

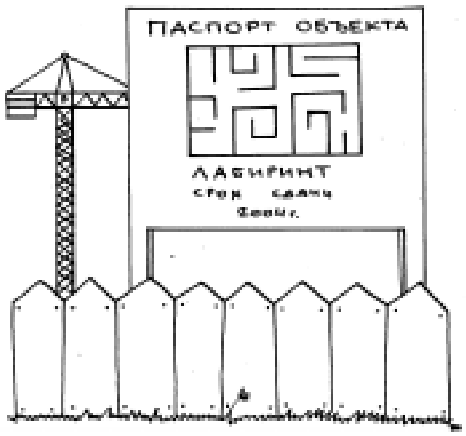
Упражнение 3.

Преобразуйте алгоритм для построения лабиринта на квадратной сетке, где квадрат – есть часть стены (1) или коридора (0). Выведите матрицу лабиринта в файл, который можно использовать далее в задачах отыскания пути в лабиринте (например, на рисунке 1 из левого верхнего поля – в правое нижнее) или для создания игр-стратегий.

Задача 6.

На окружности задана точка, две другие точки выбираются на окружности произвольно. Какова вероятность, что треугольник с вершинами в этих точках – остроугольный? Смоделируйте задачу с помощью метода Монте-Карло.

Указание. Положение случайной точки на окружности можно задавать дугой в радианах от заданной фиксированной точки (например, против часовой стрелки). Тогда угол измеряется половиной дуги между его сторонами.



Построим лабиринт...

Используя геометрическую интерпретацию вероятности, можно, наоборот, свести задачу со случайными параметрами к простому соотношению некоторых величин [2]. Пусть a – дуга (в радианах) до вершины B треугольника от вершины A (рисунок 2), b – дуга до вершины C от точки A ($0 < a, b < 2\pi$). Треугольник будет остроугольным, если:

1) при $b > a$ выполняется: $a < \pi$, $b > \pi$, $b - a < \pi$;

2) при $b < a$ выполняется: $a > \pi$, $b < \pi$, $a - b < \pi$.

Эти условия определяют заштрихованную область на рисунке 3. Следовательно, искомая вероятность равна отношению площади этой области к площади квадрата разброса точек, то есть 1/4.

Необходимо обратить внимание на *выбор подходящей вероятностной модели* для адекватного представления поставленной задачи. Рассмотрим пример.

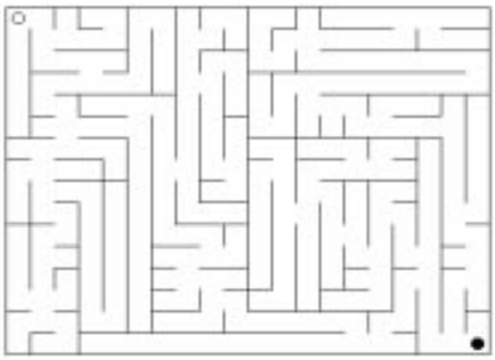


Рисунок 1

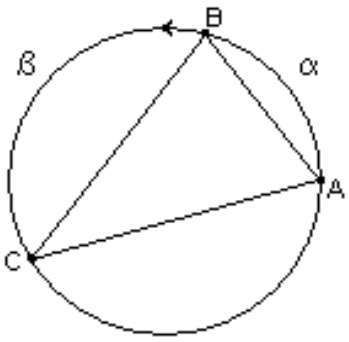


Рисунок 2

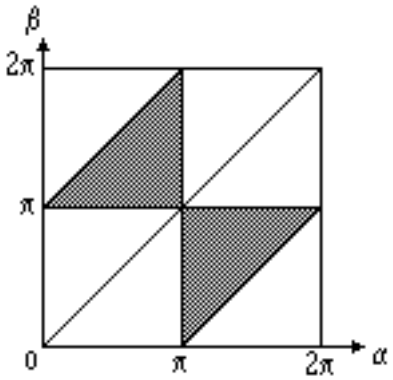


Рисунок 3

Пример 5.

В круге радиуса 1 берется наудачу хорда. Требуется определить вероятность того, что ее длина больше $\sqrt{3}$.

Построим три вероятностные модели:

M1. Положение хорды в круге однозначно задается ее серединой – моделируем случайный выбор середины хорды.

M2. Случайным образом выбираются две точки на окружности, задающие хорду.

M3. Из соображения симметрии фиксируем направление хорды, моделируем положение точки пересечения хорды с диаметром, перпендикулярным этому направлению.

Разные модели дают разные ответы на поставленную задачу (парадокс Бертрана): 1/2, 1/3, 1/4, соответственно для M1, M2, M3 вследствие того, что случайный выбор хорды четко не определен в задаче.

Упражнение 4.

Опишите геометрический способ решения задачи для перечисленных моделей.

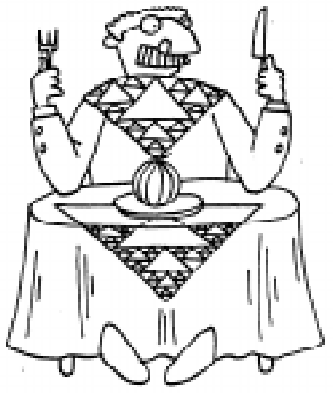
Задача 7. «Салфетка Серпинского».

Возьмем произвольный треугольник и выберем любую точку внутри него. Следующей точкой выберем середину отрезка от заданной точки до произвольно выбранной вершины треугольника. Принимая полученную точку за исходную, продолжим процесс. Изобразите процесс графически.

Салфетку Серпинского можно нарисовать с помощью рекурсивного рисования средних линий треугольника. Оказывается, казалось бы «случайный» разброс точек также создает закономерное кружево, как на рисунке 4.

Указание: Предусмотреть окончание задачи, например, по нажатию любой клавиши или генерации большого, но конечного числа точек.

Небольшое отступление от темы посвятим свойствам полученного объекта.



«Салфетка Серпинского».

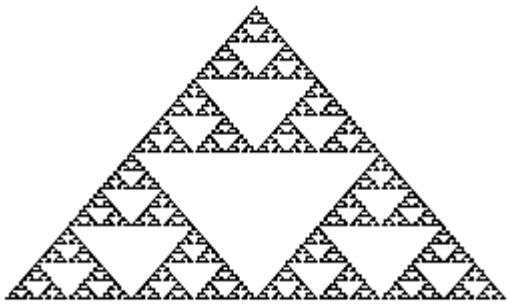


Рисунок 4

Если определить геометрическую размерность как размерность самоподобия по формуле ln (*N*)/ln (*n*), где *N* – число одинаковых частей, на которые разбивается данный объект, имеющий в *n* раз меньший пространственный размер, то для треугольной кривой Серпинского (как это видно из рисунка 4) имеем размерность ln (3)/ln (2) = 1.5849. Для объектов дробной размерности Мандельбротом введено понятие «фрактал» [2].

Красота и простота фракталов завораживает исследователей и даже побуждает использовать его самоподобие в музыке и поэзии – http://sites.netscape.net/rlbtftn/index.html.

С салфеткой Серпинского связана работа клеточного автомата, определяемая в следующей задаче.

Задача 8.

Рассмотрим события, разворачивающиеся во времени в одномерном клеточном пространстве по следующим правилам. Определяется сумма значений пяти клеток (самой клетки и двух ближайших к ней слева и справа). На следующем шаге по времени, в зависимости от суммы, которая может равняться 0, 1, 2, 3, 4, 5, клеткам присваивается значение 0, 1, 1, 1, 0, 0). Постройте на дисплее клеточный автомат, в котором каждая строка соответствует одному моменту времени.

Эволюцию клеточного автомата воспроизводит рисунок 4, подобный салфетке Серпинского.

Одномерные и многомерные клеточные автоматы явились прообразом параллельных вычислений в компьютере, они имитируют сложную структуру связей, которая наблюдается у нервных клеток мозга – нейронов, и породили новую технологию обработки информации – нейрокомпьютинг (Открытые системы, № 4(24), 1997 г.).

Исследуем работу автомата для других правил. Пусть значение в клетке (модель нейрона) определяется по сумме только соседних клеток. Если сумма четна, то считаем воздействия скомпенсированными – значение равно 0 (покой), в противном случае – 1 (возбуждение).

Упражнение 5.

Проверьте, что распространение возбуждения от одиночной клетки дает картину расположения нечетных биномиальных коэффициентов в треугольнике Паскаля.

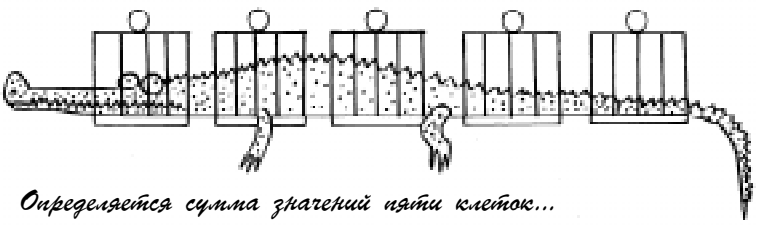
Указание. Треугольником Паскаля называется числовой треугольник, в котором по краям стоят единицы, а каждое число внутри равно сумме двух ближайших стоящих над ним чисел в строке сверху (рисунок 5).

Треугольник Паскаля связан с одним из фундаментальных законов распределения случайных чисел. Для начала, решим следующую задачу.

Задача 9.

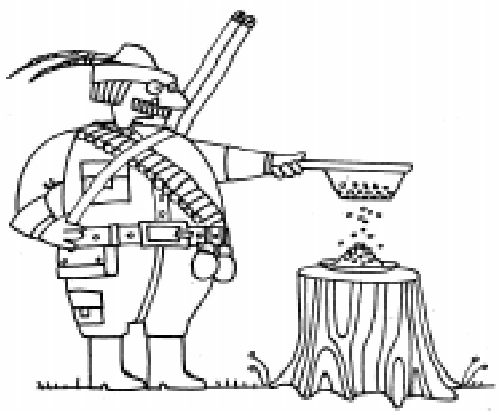
Промоделируйте работу прибора, показанного на рисунке 6. Дробинки, проходя через верхний канал и встречая препятствие, случайным образом продолжают свой путь вправо или влево. Определите среднее число дробинок, попавших в каждую ячейку, расположенную на выходе *m*-уровня препятствий ($m = 9$), если в каждой серии используется 2^m штук дробинок.

На рисунке 6 показано распределение дробинок по ячейкам (сравните с числами Паскаля!) в виде гистограммы, иллюстрирующей вид кривой – кривой Га-




```
          1
        1   1
      1   2   1
    1   3   3   1
  1   4   6   4   1
. . . . . . . . . . . . . . . .
```

Рисунок 5



Продемонстрируйте работу прибора...

усса, которая соответствует данной модели при бесконечно большом числе бесконечно малых препятствий и имеет функциональную зависимость типа $\exp(-x^2)$.

В природе кривая Гаусса характеризует естественный закон распределения («нормальное» распределение) случайных ошибок в серии измерений какой-либо величины. Проверим, насколько хорош наш датчик случайных чисел. Разобьем единичный отрезок на m (= 10000) равных отрезков. Сгенерируем N (= 1000000) случайных чисел. Подсчитаем, сколько точек (в среднем) попадет в каждый отрезок (k_1, k_2, ..., k_m) и насколько отличаются эти числа от идеального равномерного распределения, когда в каждый отрезок попадает в среднем N/m (= 100) точек. Так называемая центральная предельная теорема в теории вероятности утверждает, что ошибки средних (dk_i) распределяются по нормальному закону. Промоделируем равномерное распределение точек на отрезке [0, 1].

Пример 6.

```
const
 N = 1000000;
 M = 10000;
 K = 100; {=N/M}
var
 Segment: array[0..M-1] of LongInt;
 i: LongInt;

begin
 Randomize;
 Assign(Output, 'out.txt');
 Rewrite(Output);
 for i:=1 to N do
  Inc(Segment[Random(M)]);
 for i:=0 to M-1 do
  WriteLn(i,' ', Segment[i]-K);
end.
```

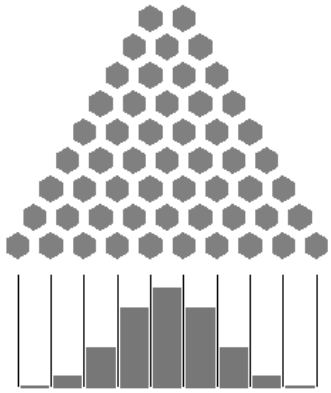


Рисунок 6

В этой программе, как и в примере 3, использовался прием подмены моделируемых величин, применяемый для уменьшения трудоемкости программы: вместо генерирования случайной точки и поиска отрезка, куда она попала, выбирается случайным образом сам отрезок и считается, что точка попала куда-то в него.

В файл «output.txt» выводится номер отрезка и отличия dk_i (со знаком) от общего среднего. На рисунке 7 представлена диаграмма сводной таблицы, построенной с помощью Excel, где по горизонтальной оси отложены значения dk_i, сгруппированные с шагом 4, а по вертикальной оси – количество отрезков, у которых различие от среднего попадает в заданный интервал.

Обычно приводят нормированные значения (делят на общее число точек N). В таком случае говорят о кривой плотности распределения (на рисунке 8 приближенно представлен график кривой Гаусса).

Из рисунка 8 видно, что данные, подчиняющиеся нормальному закону распределения, «кучкуются» около некоторого среднего значения. Чем больше расхождение, тем меньшая доля данных, имеющих такое расхождение. По такому закону, например, распределяется количество школьников (одинакового возраста) по

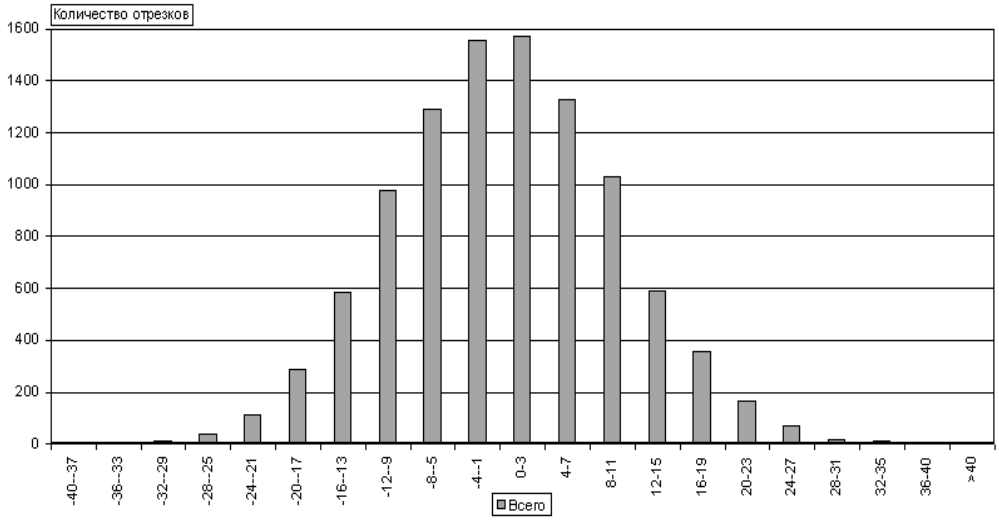


Рисунок 7

росту. Относительно группы, имеющих средний рост, очень мало слишком высоких или низких.

В некоторых задачах (например, моделирования траектории ядерной частицы при исследовании процессов в ядерном реакторе или в задаче поиска внеземных цивилизаций) требуется выбрать случайное пространственное направление от заданной точки (или задать случайную точку на единичной сфере с центром в исходной точке).

Приведем один из способов получения равномерного распределения точек на сфере. Так как три координаты связаны уравнением сферы, то в качестве независимых величин выберем координату Z и угол j, который определяет положение точки на круге, параллельном $X - Y$ плоскости (на высоте Z) от оси X.

Алгоритм по шагам:

1. Выбираем число z, равномерно распределенное на $[-1, 1]$.

2. Выбираем угол j, равномерно распределенный на $[0, 2\cdot\pi)$.

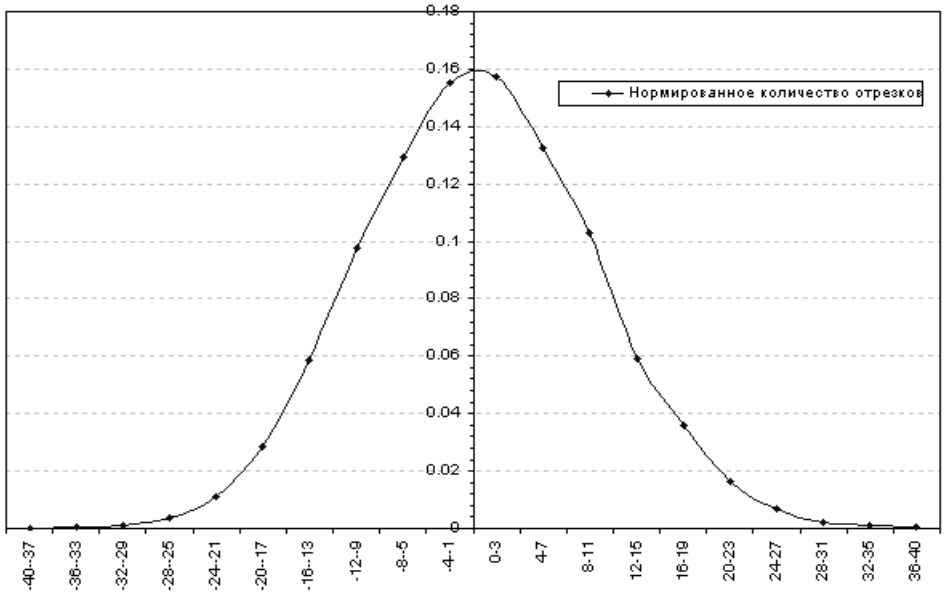


Рисунок 8. График плотности распределения

3. Полагаем $r = \sqrt{1-z^2}$.
4. Полагаем $x = r \cdot \cos(j)$.
5. Полагаем $y = r \cdot \sin(j)$.

Используя вышеизложенный алгоритм, на рисунке 9 графически изображены сгенерированные случайные точки на сфере, где темные точки расположены на «видимой» поверхности сферы ($z \geq 0$), а серые точки – на «невидимой» ($z < 0$).

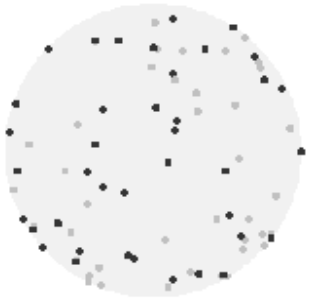
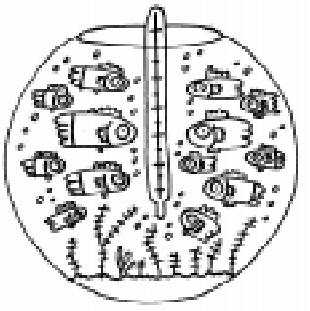


Рисунок 9

Актуальная задача определения температуры во внутренней точке M_0 однородного тела с известной температурой на границе может быть решена методом «блуждания по сферам» (подобно блужданиям по сетке в задаче 2). От заданной точки M_0 последующая точка M_1 выбирается случайным образом на максимальной (внутри исходного тела) сфере с центром в M_0 и т. д. Процесс обрывается, если очередная точка попадает на границу (или ее малую окрестность). Граничное значение температуры добавляется в определение среднего результата для точки M_0 и повторяется очередное моделирование пути.



Определить температуру в центре шара.

Задача 10.

Температура t^o в любой точке (x, y, z) на поверхности однородного единичного шара постоянна и равна

$$t(x,y,z) = 20 \cdot \cos\left(\frac{\pi}{2}x\right) \cdot \cos\left(\frac{\pi}{2}y\right) \cdot \cos\left(\frac{\pi}{2}z\right)$$

Определить температуру в центре шара.

Как видно из приведенных задач, метод Монте-Карло имеет широкое и эффективное применение, и не случайно он выбран многими авторами школьного курса информатики при изучении темы «Моделирование». Не менее важно также включать метод статистического моделирования в курс по подготовке школьников к олимпиадам по программированию.

Литература.

1. *Есипов А.С.*, *Паньгина Н.Н.*, *Громада М.И.* Информатика (задачник). СПб.: «Наука и Техника», 2001.
2. *Севастьянов Б.А.* Вероятностные модели. М.: «Наука», 1992.
3. *Пайтген Х.-О.*, *Рихтер П.Х.* Красота фракталов. М.: «Мир», 1993.

Паньгина Нина Николаевна, учитель информатики лицея № 8 г. Сосновый Бор.

Паньгин Андрей Александрович, студент 4 курса кафедры информатики математико-механического факультета СПбГУ.