№ 47 (240) декабрь 1999

еженедельное
ПРИЛОЖЕНИЕ К ГАЗЕТЕ
«ПЕРВОЕ СЕНТЯБРЯ»

# ИНФОРМАТИК

## Системотехника как научное направление

**30 лет назад, в 1969 году, в Московском энергетическом институте была организована первая в нашей стране кафедра системотехники**

Термин *системотехника* (от английского *system engineering*) введен редактором известной книги Г.Х. Гуда и Р.Э. Макола *Системотехника. Введение в проектирование больших систем*. Пер. с англ. (М., 1962). Системотехникой часто называют "направление в кибернетике, изучающее вопросы планирования, проектиро-

**Окончание читайте на с. 32**

## Читайте в номере

# РЕШЕНИЕ совещания по обсуждению концепции курса информатики в 12-летней школе

(Москва, Минобразования России, Департамент общего среднего образования, 17 ноября 1999 г.)






На совещании присутствовали:

• члены секции информатики ФЭС Минобразования России;

• авторы учебников и методической литературы по информатике из Перми, Екатеринбурга, Санкт-Петербурга, Новосибирска, Переславля-Залесского, Москвы и др.;

• учителя, методисты и ученые, стоящие у истоков становления курса информатики в общеобразовательной школе.

Всего — 42 человека.

Участники совещания обращаются к Министерству образования РФ и лично к министру В.М. Филиппову с предложением: учесть при доработке концепции 12-летней школы важную роль информатики как принципиально важной составляющей общего образования и не допустить снижения ее роли в российской школе начала XXI века.

При указанной оценке роли информатики в сфере общего образования участники совещания исходят из основных проблем, стоящих перед перспективной системой образования: фундаментализации образования, опережающего характера всей системы образования, доступности образования. Изучение информатики именно на уровне общеобразовательной школы является необходимым условием реализации этих принципов. Такая точка зрения отражена, в частности, в рекомендациях ЮНЕСКО, ряде решений Минобразования России, Российской академии образования.

Присутствие и усиление роли информатики в сфере общего образования — тенденция, крепнущая в развитых странах. Почти во всех странах Западной Европы, США, Японии, многих странах Юго-Восточной Азии информатика является дисциплиной, изучаемой большинством школьников.

Крайне важно при подготовке решений и организации практических действий сохранить и усилить достигнутое согласованное понимание Минобразования России, научной и педагогической общественностью места школьной информатики.

Участники совещания выразили тревогу в связи с наметившейся тенденцией недооценки научного ядра информатики в сфере общего образования, вытеснения ее из образования. Примерами таких действий могут служить:

• отсутствие в новой редакции базисного учебного плана общеобразовательных учебных заведений РФ образовательной области “Информатика” и отнесение информатики к образовательной области “Математика”;

• проект макетов обновляемых государственных образовательных стандартов высшего профессионального образования, в котором не усматривается места для специальности “учитель информатики”;

• практические действия некоторых региональных органов управления образованием, вытесняющих информатику из сферы общего образования (вразрез с четко выраженной позицией Минобразования России).

При выработке решений относительно места и содержания информатики в школе целесообразно принять во внимание следующее:

• Школьная информатика формировалась в России на протяжении последних 15 лет как дисциплина, ответственная и за общее интеллектуальное развитие и мировоззрение, и за формирование практических навыков, необходимых современному человеку.

• В ее становление вложены огромные материальные и человеческие ресурсы.

• К настоящему времени разработаны и используются в отечественной школе современные учебники и другие методические материалы.

• Стала свершившимся фактом массовая система подготовки, переподготовки и повышения квалификации учителей информатики, имеющая значительное научно-методическое и учебно-методическое оснащение.

• Российские школьники побеждают на международных олимпиадах по информатике.

• Авторитет российского высшего образования по информатике, основы которого закладываются в школе, в мире велик.

Участники совещания считают необходимым принятие следующих неотложных мер.

1. Включить информатику в федеральный базисный учебный план 12-летней школы в качестве самостоятельной образовательной области с выделением учебных часов на уровне основной школы.

2. При утверждении содержания образования по информатике гармонично сочетать теоретические основы науки, формирующие интеллектуальный уровень и мировоззрение и создающие фундамент образования во многих сферах, и прикладные информационные технологии с акцентом на ведущую роль фундаментальной части школьной информатики.

3. Рекомендовать региональным органам управления образованием включать при наличии в школах региона материально-технической базы информатику и в региональный, и в школьный компоненты учебных планов.

4. При разработке и утверждении новых классификаторов и госстандартов высшего профессионального образования сохранить специальность “учитель информатики” и соответственно учесть эту специальность в системе повышения квалификации и переподготовки работников образования.

Участники совещания надеются, что Министерство образования РФ примет необходимые меры по сохранению и развитию достижений школьной информатики как важнейшего фактора современного образования.

# Электронные таблицы и базы данных

В.М. Нечаев

*Продолжение. Начало в № 36, 41, 43, 46/99*

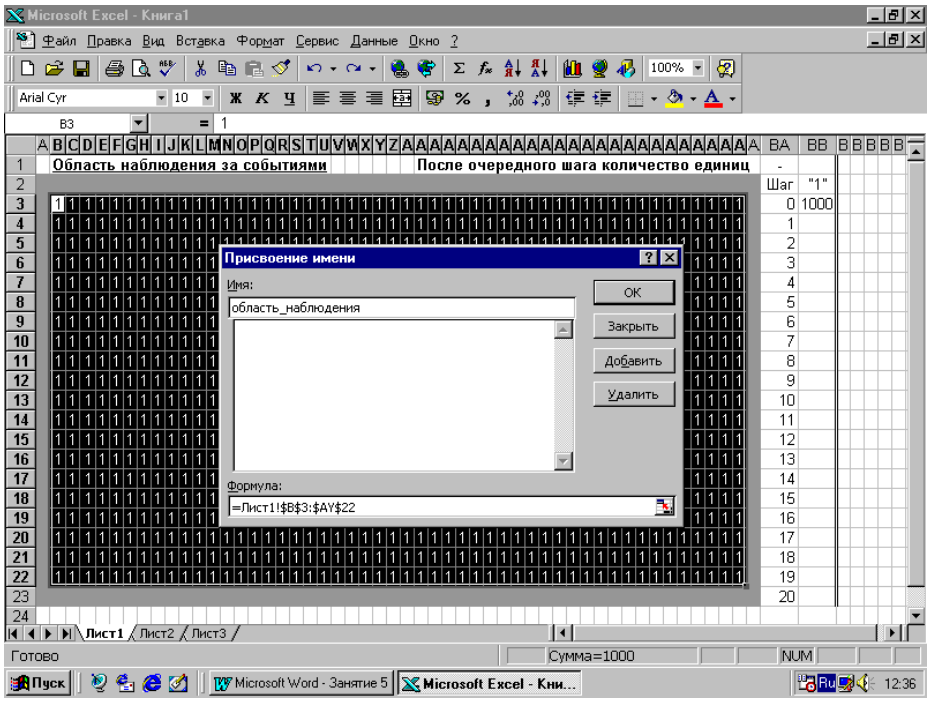
## Занятие 5. Модель радиоактивного распада атомных ядер

### Имена диапазонов

В школьном курсе математики изучаются разные функции: степенные, тригонометрические, показательные. Все они очень важны и используются при описании природных явлений, когда требуется выразить в виде формулы или графически зависимость одной величины от другой. Так, например, траекторией снаряда является парабола, поскольку высота его полета над землей есть квадратичная функция дальности, колеблющаяся на морской волне лодка дает синусоиду и т.д. Речь, конечно, идет об упрощенных моделях, не учитывающих множества обстоятельств — сопротивления воздуха, кривизны Земли, трения в воде и скорости ее течения, силы ветра, формы лодки — всего не перечислишь. Однако эти факторы, пока их можно считать второстепенными, лишь дополняют идеальную картину, показывающую главное.

Сейчас мы рассмотрим одну такую модель, несколько абстрактную, но тем не менее отражающую количественную сторону известного природного процесса — самопроизвольного распада атомных ядер, наблюдаемого у радиоактивных химических элементов — урана, радия, тория и других. Каждый атом будет представлен клеточкой таблицы, в которой может стоять единичка, означающая, что атом еще не распался, или нолик — в противном случае. Поначалу все клетки, а их должно быть достаточно много, заполнены единицами: это исходная ситуация. Спустя минуту часть атомов, но каких именно — неизвестно, распадется, т.е. единицы в некоторых клетках сменятся нулями. Через минуту единиц станет еще меньше, затем еще. Нас интересует зависимость их общего количества от времени. Предполагается, что она будет экспоненциальной.

На чистом листе уменьшим сразу ширину всех столбцов до значения 1 вместо установленного по умолчанию 8,43 (Формат⇒Столбец⇒Стандартная ширина⇒1⇒ОК). Два столбца — BA и BB — следует все же расширить для вывода итоговых результатов.




Заполним единицами всю область (B3:AY22) — как раз получится тысяча клеток (всегда приятнее иметь дело с круглыми числами). А чтобы в дальнейшем было удобно ссылаться на всю их совокупность, скажем для подсчета количества единиц, присвоим этой области какое-нибудь имя. Делается это так: сначала область выделяется с помощью мыши или через клавиатуру, затем используется команда Вставка⇒Имя⇒Присвоить⇒(вводим желаемое имя без пробелов)⇒Добавить⇒ОК.

Теперь в ячейке BB1 мы можем использовать функцию СЧЕТЕСЛИ, указывая на ее диалоговой карточке в соответствующем поле уже не адреса просматриваемых ячеек, а имя всего их диапазона: Вставка⇒Имя⇒Вставить⇒(выбираем из списка имен нужное)⇒ОК

Получится формула =СЧЕТЕСЛИ(область_наблюдения;1), согласно которой в ячейку BB1 будет выводиться интересующее нас количество нераспавшихся на момент наблюдения атомов.

## Случайные числа

Исходное состояние подготовлено, теперь смоделируем динамику событий. В этом нам поможет встроенная функция СЛЧИС, которая аргументов не имеет, а просто вырабатывает случайное число в диапазоне от нуля до единицы. Если же эту функцию использовать не в чистом виде, а вложив в функцию ОКРУГЛ, то получать будем два значения при каждом новом вычислении: =ОКРУГЛ(СЛЧИС();0). При вложении функций, как и во многих других случаях, заполнять поля лучше в обратном порядке — снизу вверх, причем на карточке округления следует сначала указать второй аргумент, выбрав ноль для количества десятичных знаков после запятой.

Но где же разместить такую формулу, вот вопрос! Прямо в ту ячейку, за которой мы хотим наблюдать, например B3, помещать ее нельзя. Ведь по условиям задачи единица может в дальнейшем поменяться на ноль, но никак не наоборот: распавшиеся атомы не возрождаются! И усложнить выражение, указав =ЕСЛИ(B3=1;ОКРУГЛ(СЛЧИС();0);0), тоже нельзя, поскольку ссылка оказывается циклической.

условие — если истина — ложь

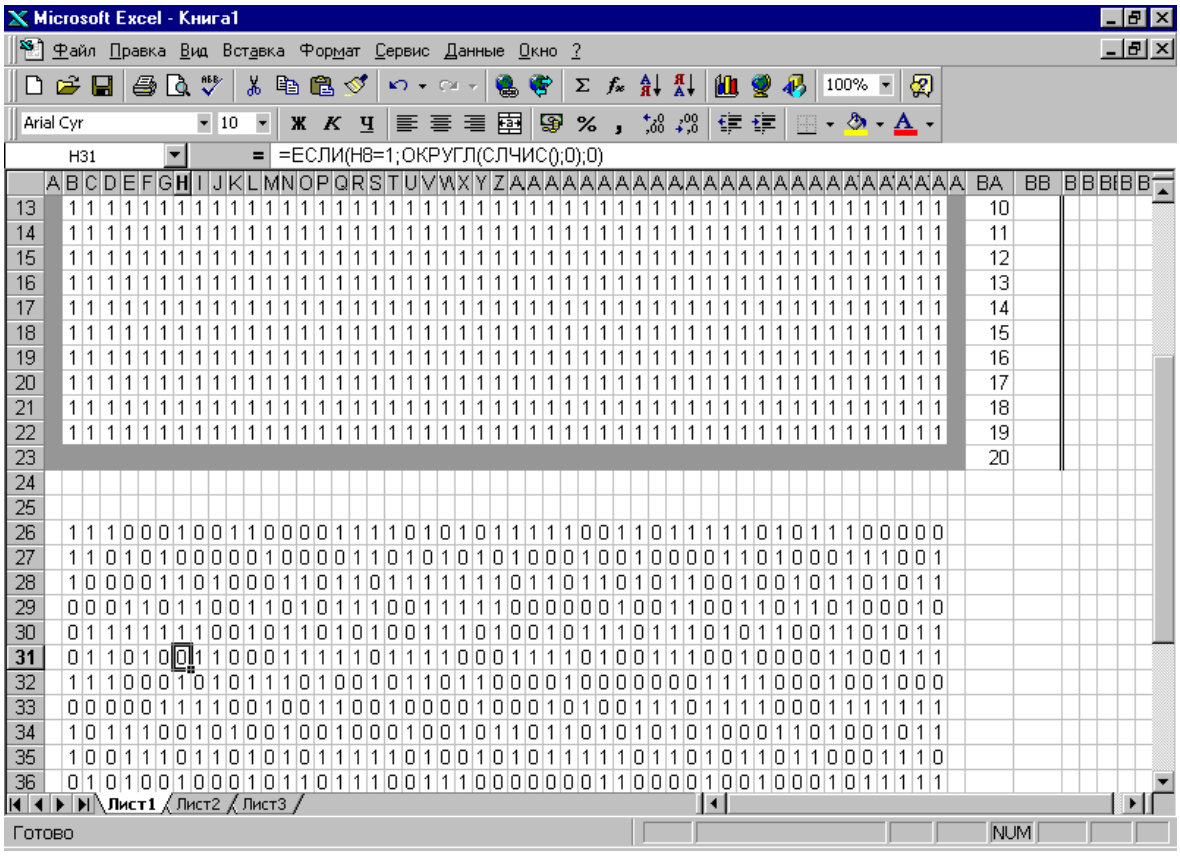
Однако выход из ситуации есть.

Нужно создать еще одну область для проведения опытов, вспомогательную, расположенную, допустим, ниже. Внешний вид у нее будет, как у основной арены действий:

Правка⇒Перейти⇒область_наблюдения⇒ОК⇒Копировать⇒(переместить курсор на B26)⇒Вставить⇒Esc

Вот оно, кстати, удобство использования имен вместо повторных выделений клеток.

Теперь уже и формулы можно вводить беспрепятственно. Собственно, ввести надо именно указанную формулу, причем один только раз, в соответствующую B3 ячейку B26, после чего следует просто растиражировать ее по всему полю (B26:AY45). Между прочим, сейчас самое подходящее время и этой вновь созданной области присвоить свое имя — "лаборатория".




План эксперимента таков. Две картинки представляют собой как бы мгновенные отчеты о состоянии системы в последовательные моменты времени. Верхнее поле вначале заполнено "целыми атомами", на нижнем примерно половина атомов уже "распалась". Если мы теперь скопируем нижние нули и единицы (причем именно сами числа, а не формулы, с помощью которых они получились) "вверх", то новое вычисление случайных величин внизу даст следующий шаг процесса.

Так, например, в ячейке H31 получился ноль, а в H32 оказалась единица. Это при первом шаге, при одинаковых начальных условиях. А при втором, когда в соответствующих ячейках H8 и H9 уже будут стоять 0 и 1, "атом" в ячейке H9 снова подвергнется испытанию судьбы в отличие от своего соседа, для которого все уже закончилось. Нам останется лишь повторять шаги один за другим, перемещая каждый раз результаты подсчета единиц из ячейки BB1 в клетки (BB3:BB23).

## Макросы

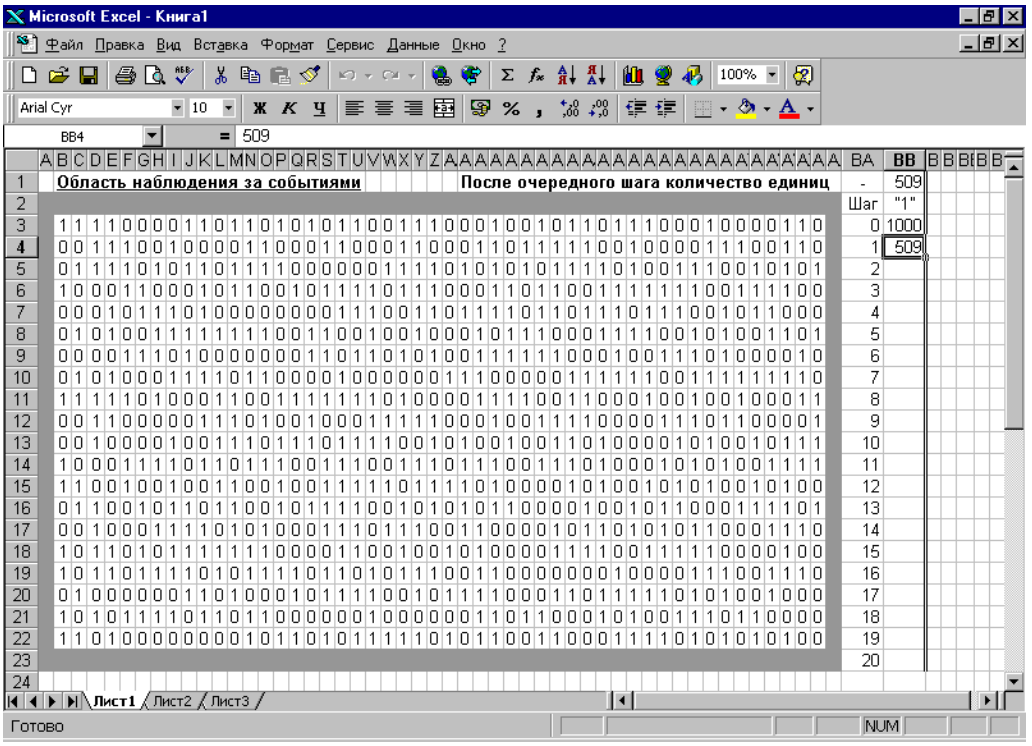
В принципе уже сейчас можно было бы приступить к сбору данных для статистики. Впрочем, и так понятно, что после первого шага "в живых" останется около половины "атомов", после второго — половина от половины, т.е. четверть, далее — половина от четверти. Обычная геометрическая прогрессия — стоило ли огород городить?

Однако все так ясно, когда вероятность распада равна 1/2. Но ведь нам никто не мешает брать для своей вычислительной лаборатории другие, более сложные законы, чтобы из сравнения результатов делать, быть может, и не вполне очевидные заключения. При этом неплохо было бы автоматизировать сам способ проведения эксперимента: копировать значения из нижнего поля в верхнее, перемещать текущие показания счетчика в столбец статистики, восстанавливать исходное состояние системы для повторных опытов. Такую рутинную работу, выполняемую многократно, мы организуем с помощью макросов, запоминая последовательность "мышиных" щелчков и клавиатурных нажатий. И прежде всего займемся копированием значений.

Сервис ⇒ Макрос ⇒ Начать запись ⇒
(вводим имя: перенос_снизу_вверх) ⇒ Tab ⇒ (вводим букву для вызова: латинскую Q) ⇒
ОК ⇒ Правка ⇒ Перейти ⇒ лаборатория ⇒ ОК ⇒
Копировать ⇒ Правка ⇒ Перейти ⇒ область_наблюдения ⇒ ОК ⇒
Правка ⇒ Специальная вставка ⇒ значение ⇒ ОК ⇒ Esc ⇒
(вертикальная прокрутка листа вверх для щелчка по клетке BB1) ⇒
(правая клавиша мыши — "протяжка" BB1 за рамку в BB4) ⇒
Копировать только значения ⇒ Сервис ⇒ Остановить запись.

} Тело макроса

Все, макрос записан. Его можно вызвать в любой момент, стоит лишь нажать Ctrl + Q. Экран мигнет черным — это сработает выделение верхней области при переходе к ней (нижняя тоже мигает, но почти незаметно). Во время вставки в "область_наблюдения" числовых значений, скопированных из "лаборатории", ячейки последней будут заново пересчитаны по случайному закону и появится свежий материал для последующих операций.




Аналогично записываем еще один макрос — для восстановления исходного состояния:

Сервис ⇒ Макрос ⇒ Начать запись ⇒ (имя: в_исходное_состояние) ⇒
Tab ⇒ (буква: Z) ⇒ ОК ⇒ (щелчок по клетке B3 ⇒ Ввод в нее числа 1) ⇒
(протяжка с помощью левой клавиши мыши за уголок вправо до AY3 включительно) ⇒
(протяжка с помощью левой клавиши мыши за уголок вниз до строки 22 включительно) ⇒
(выделение диапазона BB4:BB23) ⇒ Del ⇒ (щелчок по клетке BB1) ⇒
Сервис ⇒ Остановить запись.

} Тело второго макроса

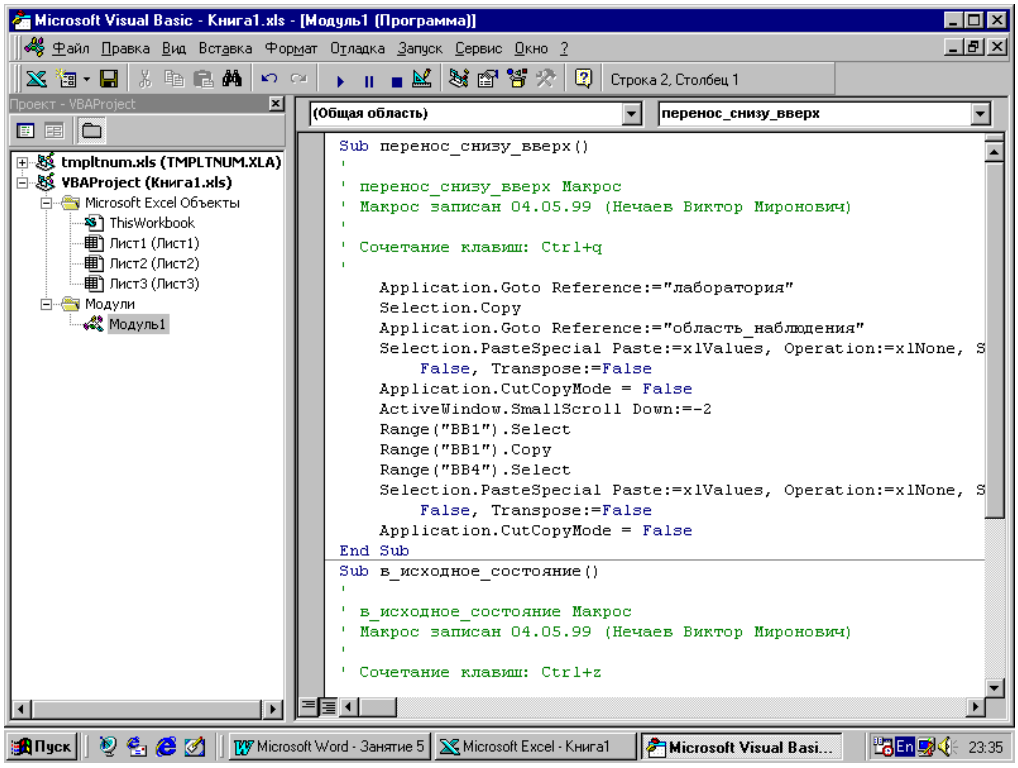
## Язык VBA

Проверим работу созданных макросов — нажмем раза три-четыре подряд клавиши Ctrl + Q, посмотрим, как единицы постепенно сменяются нулями, а потом восстановим исходное состояние, используя комбинацию клавиш Ctrl + Z. Все хорошо, кроме одного. Промежуточные результаты подсчета, начиная со второго шага, требуется каждый раз вручную перетаскивать из ячейки BB4 в следующие за ней клетки BB5, BB6, BB7... ("цепляясь" правой клавишей мыши за курсорную рамку и выбирая в контекстном меню пункт "Копировать только значения"). Почему бы эту буксировку тоже не включить в первый макрос?

К сожалению, не все так просто. Макрос, для создания которого запоминалось описание какого-то фиксированного набора действий, по определению не может приводить то к одной, то к другой ячейке. Чтобы все же достичь желаемого, необходимо пообщаться с программой Excel на языке Visual Basic for Applications. Общение будет, впрочем, довольно непродолжительным, поскольку уже готовый макрос можно взять за основу и лишь немного подправить.

Сервис ⇒ Макрос ⇒ Макросы ⇒ (выбираем имя перенос_снизу_вверх) ⇒ Изменить

Открывается еще одно окно, а в нем видна запись на языке VBA двух наших макросов; тот, который нам нужен, как раз умещается целиком. Сначала указано имя этой, как теперь видно, подпрограммы и даны соответствующие комментарии. Затем следуют программные строки, в которых мы сейчас хоть и поверхностно, но разберемся. Более того, добавим новые!




```
For n = 0 To 19                                  — Начало цикла
Application.Goto Reference:="лаборатория"        — Переход в "лабораторию"
Selection.Copy                                   — Копирование ее содержимого в буфер
Application.Goto Reference:="область_наблюдения" — Переход к "области_наблюдения"
Selection.PasteSpecial Paste:=xlValues, …        — Специальная вставка, только значения

Application.CutCopyMode=False                    — Больше копировать не надо (Esc)

ActiveWindow.SmallScroll Down:=-2                — Прокрутка вверх на две клетки
Range("BB1").Select                              — Переход к клетке BB1
Range("BB1").Copy                                — Копирование ее содержимого в буфер
Range("BB4").Offset(n,0).Select                  — Переход к клетке BB4 (и вниз на n)
Selection.PasteSpecial Paste:=xlValues, …        — Специальная вставка, только значения
Application.CutCopyMode=False                    — Больше копировать не надо
Next                                             — Конец цикла
```

Всего лишь в трех местах нам нужно внести изменения в текст программы — для наглядности они выделены. Две крайних строки обеспечивают организацию цикла, а вставка внутри третьей как бы говорит: "Перейдя к BB4, сместись вниз на *n* строк и только потом выделяй".

## Сбор статистических данных

Вернувшись в Excel и запустив на выполнение макрос, мы получим теперь целую серию из двух десятков миганий, по окончании которой уже вся "область_наблюдения" будет покрыта нулями, а в правом столбце будет представлен числовой ход процесса. Второй макрос при восстановлении исходного состояния предыдущие результаты просто сотрет. Нам же эти данные после каждого опыта следует сохранять для статистики на другом рабочем листе. Для этого потребуется записать еще один, уже третий, макрос.

Но сначала присвоим области BB3:BB23, где размещены нужные нам числа, а также области D3:D23 (на Листе 2), куда их предстоит перенести, свои собственные имена; допустим, "результат_опыта" и "для_усреднения". Тогда формировать новый макрос будет гораздо удобнее.

Сервис ⇒ Макрос ⇒ Начать запись ⇒ (имя: добавить_к_статистике) ⇒ Tab ⇒ (буква: S) ⇒ ОК ⇒ (щелчок по ярлыку Листа 1, а затем по ячейке B3) ⇒ (набор с клавиатуры цифры 1) ⇒ (щелчок по кнопке ввода в строке формул, т.е. по зеленой галочке слева от знака равно) ⇒ Копировать ⇒ (выбор "области_наблюдения" в раскрывающемся еще левее списке имен) ⇒ Вставить ⇒ Esc ⇒ (выбор "результат_опыта" в списке имен) ⇒ Копировать ⇒ (выбор "для_усреднения" в списке имен) ⇒ Вставить ⇒ (выбор "результат_опыта" в списке имен) ⇒ Del ⇒ (щелчок по BB1) ⇒ Сервис ⇒ Остановить запись.

Теперь можно попробовать, как работает весь комплекс. Стартуем с Листа 1:

а) Ctrl + Z ("область_наблюдения" заполнилась единицами, а колонка результатов очистилась);

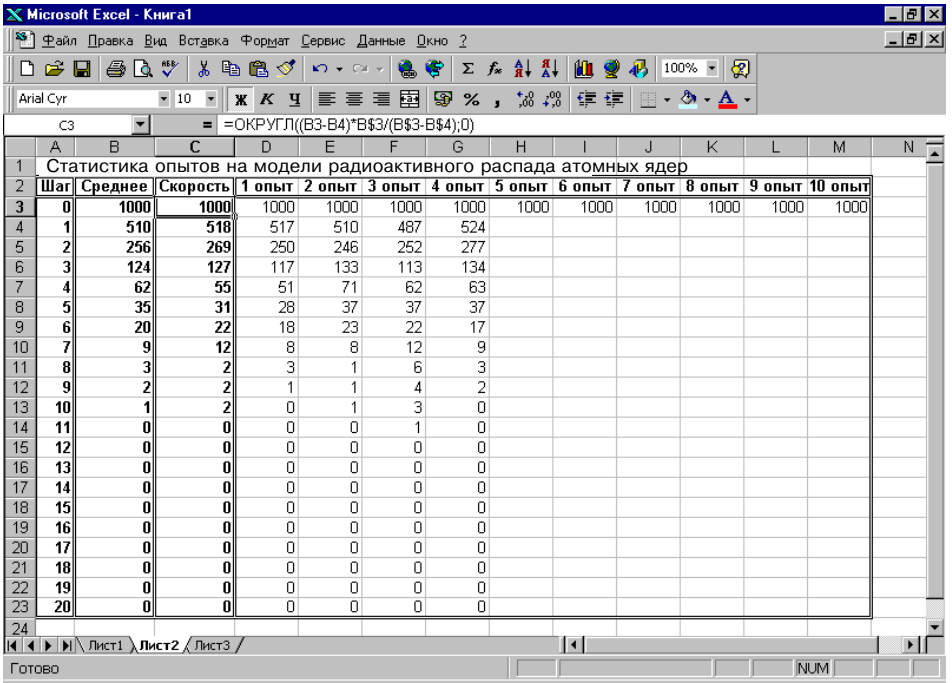
б) Ctrl + Q (длинная серия миганий экрана и в итоге одни нули, но результаты есть);

в) Ctrl + S (мгновенный перескок на Лист 2 и затем обратно, исходная ситуация восстановлена).

Только что созданный нами макрос тоже нуждается в небольших дополнениях на языке Visual Basic. Дело в том, что переносить результат на Лист 2 надо не в один и тот же столбец D, а с увеличивающимся смещением: первый раз в D, второй — в E, третий — в F. Вот эти добавки:

Static n. В самой первой строке (после комментариев) объявляется переменная-счетчик, сохраняющая свое значение от одного вызова подпрограммы до следующего. И далее, в теле макроса, между строками Application.Goto Reference:="для_усреднения" и ActiveSheet.Paste, вставляем еще одну, Selection.Offset(0, n). Select, реализующую смещение по столбцам таблицы.

А двумя последними операторами (перед End Sub) этот счетчик "накручивается", n=n+1: If n = 10 Then n = 0, доходя до того предельного значения, которое нас устраивает.




| Шаг | Среднее | Скорость | 1 опыт | 2 опыт | 3 опыт | 4 опыт | 5 опыт | 6 опыт | 7 опыт | 8 опыт | 9 опыт | 10 опыт |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1000 | 1000 | 1000 | 1000 | 1000 | 1000 | 1000 | 1000 | 1000 | 1000 | 1000 | 1000 |
| 1 | 510 | 518 | 517 | 510 | 487 | 524 | | | | | | |
| 2 | 256 | 269 | 250 | 246 | 252 | 277 | | | | | | |
| 3 | 124 | 127 | 117 | 133 | 113 | 134 | | | | | | |
| 4 | 62 | 55 | 51 | 71 | 62 | 63 | | | | | | |
| 5 | 35 | 31 | 28 | 37 | 37 | 37 | | | | | | |
| 6 | 20 | 22 | 18 | 23 | 22 | 17 | | | | | | |
| 7 | 9 | 12 | 8 | 8 | 12 | 9 | | | | | | |
| 8 | 3 | 2 | 3 | 1 | 6 | 3 | | | | | | |
| 9 | 2 | 2 | 1 | 1 | 4 | 2 | | | | | | |
| 10 | 1 | 2 | 0 | 1 | 3 | 0 | | | | | | |
| 11 | 0 | 0 | 0 | 0 | 1 | 0 | | | | | | |
| 12 | 0 | 0 | 0 | 0 | 0 | 0 | | | | | | |
| 13 | 0 | 0 | 0 | 0 | 0 | 0 | | | | | | |
| 14 | 0 | 0 | 0 | 0 | 0 | 0 | | | | | | |
| 15 | 0 | 0 | 0 | 0 | 0 | 0 | | | | | | |
| 16 | 0 | 0 | 0 | 0 | 0 | 0 | | | | | | |
| 17 | 0 | 0 | 0 | 0 | 0 | 0 | | | | | | |
| 18 | 0 | 0 | 0 | 0 | 0 | 0 | | | | | | |
| 19 | 0 | 0 | 0 | 0 | 0 | 0 | | | | | | |
| 20 | 0 | 0 | 0 | 0 | 0 | 0 | | | | | | |

Лист 2, разумеется, надо подготовить заранее, так чтобы результаты отдельных опытов усреднялись. Впоследствии по этим средним значениям будут строиться графики.

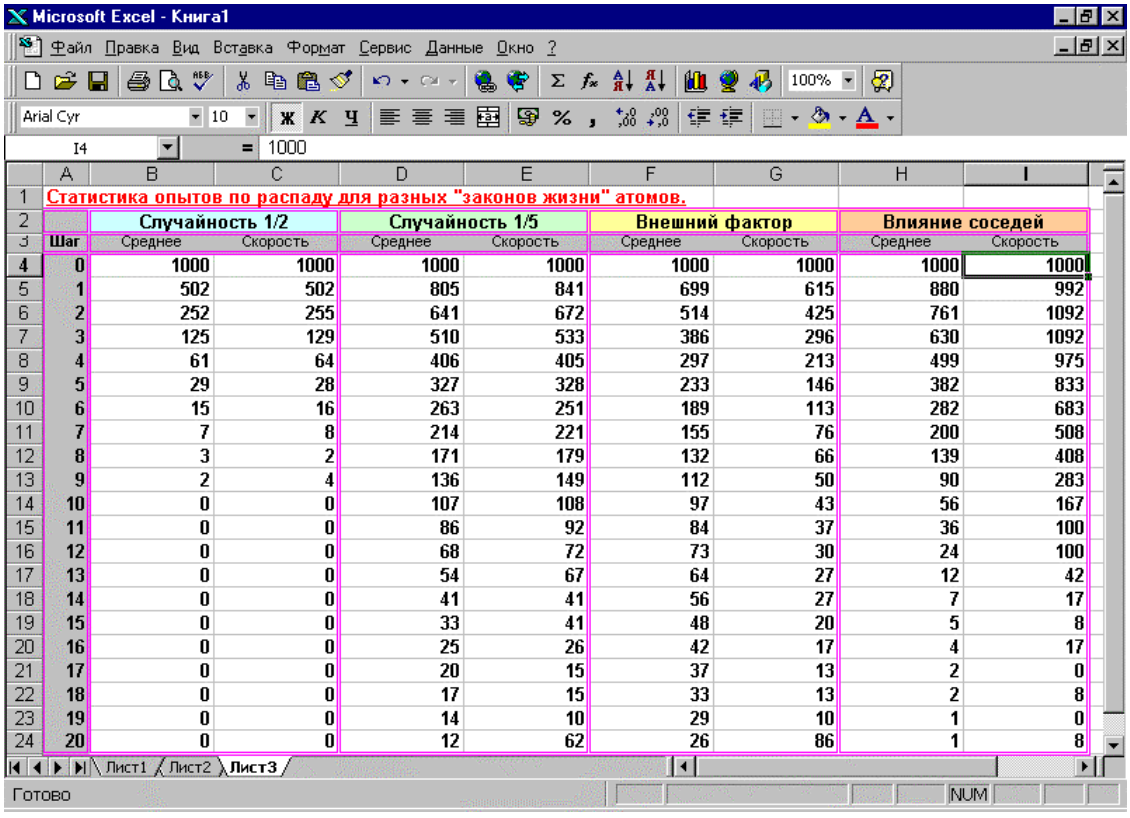
## Случай или тенденция

Помимо самого количества “нераспавшихся атомов”, нам важно знать еще и насколько быстро это количество уменьшается с каждым шагом. В формуле под заголовком “Скорость” (см. экран на с. 7) указано, как ее следует подсчитывать. Скоростью будет просто разница между предыдущим и последующим значениями, но только умноженная на некоторый постоянный коэффициент, специально подобранный так, чтобы оба будущих графика, для количества и для скорости, начинались в одной точке. Будут ли они и в других точках для следующих шагов совпадать или, наоборот, разойдутся, и можно ли найти какое-то объяснение такому их поведению — вот, собственно, главное, что представляет интерес во всем исследовании.

Первый же проведенный опыт дает явное совпадение данных в колонках “Среднее” и “Скорость” (тут даже не понадобились графики, использовались одни только числа), а его повторение лишь подтверждает этот факт — при усреднении случайные отклонения сглаживаются. Попробуем тогда изменить сам закон, по которому во вспомогательной области “лаборатория” определяется, “распадется” ли во время очередного шага атом или останется “в живых”. Формула в ячейке `B26` по-прежнему будет ссылаться на соответствующую ей клетку `B3` и, встретив там рано или поздно ноль, дальше так его нулем и оставит, отслеживая печальное событие, случившееся с атомом. Но вот саму вероятность его распада мы можем уменьшить, добавив к первому аргументу функции `ОКРУГЛ` несколько десятых: `=ЕСЛИ(B3=1;ОКРУГЛ(СЛЧИС()+0,3;0);0)`. Раньше исход отдельного испытания был пятьдесят на пятьдесят, т.е. 1/2, теперь же будем иметь двадцать на восемьдесят, или 1/5. Формулу, конечно, надо растиражировать на все ячейки “лаборатории”.

Проведя серию опытов, усредненные их результаты перенесем уже на третий лист, копируя только значения. Очевидно, что от предыдущих, которые также надо поместить в этот итоговый отчет, они заметно отличаются. Однако и теперь сама исследуемая величина и скорость ее изменения с большой степенью точности равны друг другу. Ну а если еще сильнее изменить закон жизни для атомов?

Пусть они будут распадаться не только сами по себе, но еще и под воздействием дополнительного внешнего фактора. Мы снова добавим к аргументу `СЛЧИС` несколько десятых, но только не для всех клеток одинаково: для первых четырех рядов “лаборатории” — ни одной, для следующих четырех — одну, для следующих — две и т.д. Чем ниже лежат “атомы”, тем у них, выходит, больше вероятность остаться нераспавшимися, как будто они подвергаются какому-то разрушающему облучению сверху и верхние слои экранируют нижние.




Статистика опытов по распаду для разных "законов жизни" атомов.

| Шаг | Случайность 1/2 | | Случайность 1/5 | | Внешний фактор | | Влияние соседей | |
|---|---|---|---|---|---|---|---|---|
| | Среднее | Скорость | Среднее | Скорость | Среднее | Скорость | Среднее | Скорость |
| 0 | 1000 | 1000 | 1000 | 1000 | 1000 | 1000 | 1000 | 1000 |
| 1 | 502 | 502 | 805 | 841 | 699 | 615 | 880 | 992 |
| 2 | 252 | 255 | 641 | 672 | 514 | 425 | 761 | 1092 |
| 3 | 125 | 129 | 510 | 533 | 386 | 296 | 630 | 1092 |
| 4 | 61 | 64 | 406 | 405 | 297 | 213 | 499 | 975 |
| 5 | 29 | 28 | 327 | 328 | 233 | 146 | 382 | 833 |
| 6 | 15 | 16 | 263 | 251 | 189 | 113 | 282 | 683 |
| 7 | 7 | 8 | 214 | 221 | 155 | 76 | 200 | 508 |
| 8 | 3 | 2 | 171 | 179 | 132 | 66 | 139 | 408 |
| 9 | 2 | 4 | 136 | 149 | 112 | 50 | 90 | 283 |
| 10 | 0 | 0 | 107 | 108 | 97 | 43 | 56 | 167 |
| 11 | 0 | 0 | 86 | 92 | 84 | 37 | 36 | 100 |
| 12 | 0 | 0 | 68 | 72 | 73 | 30 | 24 | 100 |
| 13 | 0 | 0 | 54 | 67 | 64 | 27 | 12 | 42 |
| 14 | 0 | 0 | 41 | 41 | 56 | 27 | 7 | 17 |
| 15 | 0 | 0 | 33 | 41 | 48 | 20 | 5 | 8 |
| 16 | 0 | 0 | 25 | 26 | 42 | 17 | 4 | 17 |
| 17 | 0 | 0 | 20 | 15 | 37 | 13 | 2 | 0 |
| 18 | 0 | 0 | 17 | 15 | 33 | 13 | 2 | 8 |
| 19 | 0 | 0 | 14 | 10 | 29 | 10 | 1 | 0 |
| 20 | 0 | 0 | 12 | 62 | 26 | 86 | 1 | 8 |

И, наконец, рассмотрим вариант, который отразит не внешнее вмешательство в судьбы атомов, а влияние их друг на друга. Для каждой клетки к тому же аргументу `СЛЧИС` будем прибавлять столько десятых, сколько в ее ближайшем окружении (сверху, снизу, справа, слева) стоит единиц. “Живые” будут как бы поддерживать “в живых” своих соседей.

## Тиражирование графиков

Результаты удобнее обсуждать, когда они представлены в графической форме: построим отдельные диаграммы для всех четырех вариантов. Различаться они будут незначительно, поэтому, выполнив построение первой, мы просто размножим ее в нужном количестве экземпляров, после чего останется лишь внести коррективы в ссылки на диапазоны отображаемых данных. Выделяем область B3:C24 и вызываем мастера диаграмм:

Стандартные ⇒ График ⇒ (Вид — самый первый) ⇒ Далее ⇒ Ряд ⇒ (щелкаем в поле для Подписи по оси X и проводим курсором мыши по ячейкам A4:A24) ⇒ Далее ⇒ (в поле для Названия диаграммы вводим "Случайность 1/2") ⇒ Далее ⇒ Готово

Диаграмма получилась неказистой, но у нас есть все возможности, чтобы привести ее к надлежащему виду, и делать это лучше уже на новом, четвертом по счету листе. Ножницами вырезаем диаграмму и делаем две вставки подряд — сначала вставку нового листа, затем вставку из буфера. Вставленную диаграмму смещаем в левый верхний угол, размер у нее как раз подходящий — четверть полного окна. Но вот размеры внутренних элементов надо, конечно, менять.

**Оси координат**. Щелчок по ним правой клавишей мыши ⇒ Формат оси ⇒ Шрифт ⇒ 7 ⇒ снимаем галочку автомасштаба ⇒ (для оси X еще Шкала ⇒ снимаем галочку пересечение...) ⇒ OK.

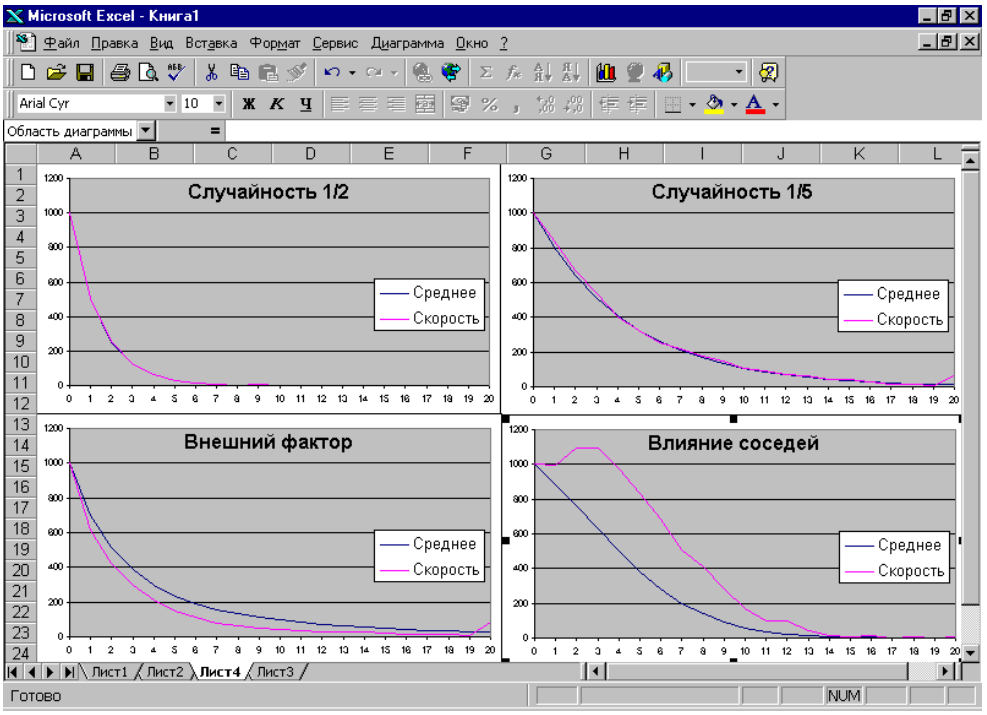
**Область построения**. Щелчок левой клавишей по серому фону ⇒ растягиваем, "цепляясь" за средние черные квадратики вправо, влево, вверх, вниз до предела.

Чтобы во время работы с диаграммами не ошибиться и указать мышью именно тот элемент изображения, который требуется изменить, очень важно внимательно приглядываться к всплывающим подсказкам. Как, например, сейчас, когда мы хотим выделить всю диаграмму целиком и скопировать ее в буфер для дальнейшего тиражирования. При этом косвенным подтверждением правильного выбора будет доступность соответствующей кнопки на панели инструментов. Щелкнув по ней, мы увидим мигающий по периметру пунктир; но только перед тем, как вставлять диаграмму из буфера на новое место, скажем справа, следует переместить туда, в клетку G1, саму курсорную рамку. Затем помещаем курсорную рамку в клетку A13 и вновь вставляем диаграмму, а потом выполняем те же операции для клетки G13.

Теперь поменяем "содержимое" второй диаграммы и займемся заголовком. Щелкаем по нему, используя левую клавишу мыши, а затем еще раз, но не быстро (это не двойной щелчок, а два отдельных). Дальше работаем, как с обычным текстом, — 1/2 стираем, а 1/5 вписываем.

Настала очередь самого главного — исходных данных:

Щелчок правой клавишей мыши по области диаграммы ⇒ Исходные данные ⇒ Ряд ⇒ Среднее ⇒ щелчок по красной кнопочке справа от поля ввода для Значения ⇒ выделяем мышью вместо прежнего новый диапазон ячеек D4:D24 и снова щелкаем по той же красной кнопочке ⇒ Скорость ⇒ то же самое, но ячейки, соответственно, E4:E24 ⇒ OK.




Для двух нижних диаграмм выполняются аналогичные действия, только диапазоны ячеек другие.

## Лист диаграммы

В дополнение к построенным четырем диаграммам неплохо бы было иметь еще одну, совмещенную, чтобы сравнивать графики друг с другом. В качестве объекта действий выделим сразу все ячейки B2:I24 и повторим почти тот же самый путь, что и в предыдущих случаях. Единственным различием будет наше пожелание (на последнем этапе диалога с мастером) поместить диаграмму на отдельном листе, тут же, кстати, и назвав его "Общая картина".

На отдельном листе диаграмма выглядит гораздо лучше, и это еще в самом простом и скромном оформлении. Подбирать же разнообразные цветовые сочетания — довольно увлекательное занятие. Попробуйте, например, использовать готовые образцы для фона (текстур) как области самой диаграммы, так и области ее построения:

Щелчок правой клавишей мыши по соотв. области ⇒ Формат ⇒ Вид ⇒ Способы заливки ⇒ Текстура ⇒ Образец ⇒ OK

Впрочем, для фона области построения лучше оставить обычную заливку, слегка, пожалуй, усилив затенение. Для области же легенды подходит, наверное, градиентная горизонтальная заливка, опять-таки серым, — хотя и велико желание употребить яркие цвета, но надо знать меру. Разве что на мелких деталях, вроде рамок той и другой области, а также осей координат, можно дать себе некоторую свободу:

Щелчок правой клавишей мыши ⇒ Формат ⇒ Вид ⇒ Рамка другая ⇒ Цвет (красный) ⇒ Толщина (большая) ⇒ с тенью ⇒ OK

Щелчок правой клавишей мыши ⇒ Формат оси ⇒ Вид ⇒ Ось другая ⇒ Цвет (красный) ⇒ Толщина (средняя) ⇒ OK

Что же касается самих линий, то здесь как раз целесообразно по возможности разнообразить палитру, а если покажется, что оттенков не хватает, следует призвать на помощь пунктиры. Заодно постараемся избавиться от некоторой угловатости графиков:

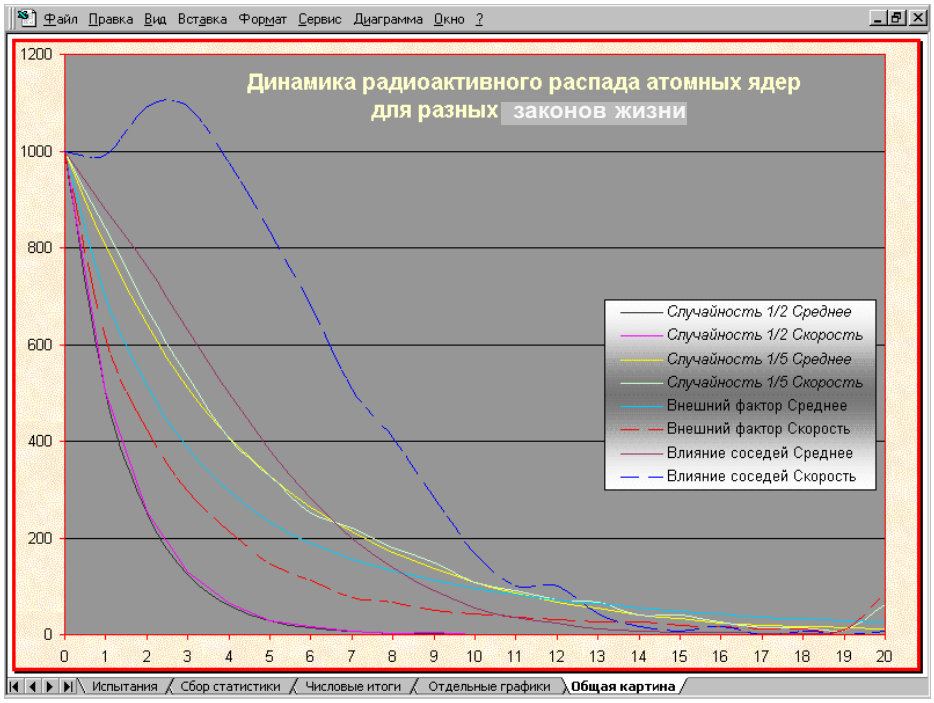
Щелчок правой клавишей мыши по линии ⇒ Формат ряда ⇒ Вид ⇒ Линия другая ⇒ Цвет (Тип) ⇒ Сглаженная ⇒ OK

Из двух самых нижних кривых сглаживать надо только одну (любую): эти линии, так тесно прилегающие друг к другу, не должны сливаться в одно целое. Между прочим, во время сглаживания, подстраиваясь к нему, программа не совсем удачно сместит вниз еще и точку отсчета вертикальной шкалы, и мы ее вежливо поправим:

Щелчок правой клавишей мыши по цифрам оси Y ⇒ Формат оси ⇒ Шкала ⇒ Минимальное значение 0 ⇒ OK

Чтобы не обойти вниманием ось X, щелкнем правой клавишей и по ней:

Щелчок правой клавишей мыши по цифрам X ⇒ Формат оси ⇒ Шкала ⇒ Число категорий между подписями 2 ⇒ OK




Такая красивая диаграмма достойна того, чтобы быть развернутой во весь экран, на этот раз уже с помощью левой клавиши мыши:

Вид ⇒ Во весь экран и затем еще раз Вид ⇒ По размеру окна.

Все! Можно записывать таблицу на диск в файл "Модель распада ядер".

## Вопросы для проверки

1. Каким образом избежать мигания экрана при исполнении первого макроса?
2. Сколько строк программного текста, не считая первой строки (`Sub в_исходное_состояние`), последней строки (`End Sub`) и комментариев, содержит второй макрос?
3. Какая формула записана в ячейке `C4` на Листе 2?
4. Какое изменение надо внести в таблицу, чтобы на графиках не фигурировал явно не относящийся к делу скачок скорости в конце кривой?
5. Какая формула описывает третий вариант закона жизни ядра?
6. Какая формула описывает четвертый вариант закона?
7. Сколько элементов разного рода имеется в построенных диаграммах?
8. Где можно наблюдать дублирование всплывающих подсказок диаграммы?
9. В чем проявляется на общей диаграмме положительный характер взаимодействия ядер — поддержка живыми живых?

## Задания для самостоятельной работы

С помощью нашей числовой модели были испытаны четыре разных закона распада атомных ядер. Несмотря на очень упрощенную форму, исследование дает повод высказать одно важное предположение о механизме самого явления. Известно, что единственной функцией, производная которой равна ей самой, является экспонента. Но и у нас в двух случаях получилось так, что скорость совпадала с самой величиной! Подбор масштаба отвечал наличию множителя $\lambda$ в показателе экспоненты $e^{\lambda x}$, в нашем случае отрицательного (им определяется так называемый период полураспада $T = \ln 2/\lambda$). Совпадения графиков происходили, когда каждое ядро распадалось независимо от внешнего влияния (каким бы оно ни было — глобальным или местным), просто потому, что "судьба постучала в дверь". Отсюда и наше предположение: если распад отдельного ядра есть следствие исключительно внутренних процессов, протекающих в нем, и для внешнего наблюдателя данный распад представляется случайным событием, то динамика коллективного поведения таких ядер обязательно будет экспоненциальной.

Это очень смелое предположение*, но достаточно ли у нас имеется оснований, чтобы его принять? Вообще говоря, надо осуществить проверку на новом экспериментальном материале, отражающем другие возможные законы жизни ядер. Целесообразно сделать следующее.

1. Подобно тому, как вводилось неоднородное в пространственном отношении внешнее воздействие, исследовать эффект переменного во времени, т.е. меняющегося от опыта к опыту воздействия.
2. Рассмотреть случай взаимного влияния атомов, выражающегося не в "поддержке друг друга", а, наоборот, в конкуренции, когда для каждого ядра вероятность остаться нераспавшимся будет уменьшаться пропорционально количеству имеющихся вокруг него целых соседей.
3. Повторить опыт с экранируемым внешним воздействием, но учесть то обстоятельство, что в экранировании могут участвовать только не распавшиеся еще ядра.

*Продолжение следует*

* Его можно высказать даже в еще более сильной форме: если некий объект, существуя исключительно по своим собственным внутренним законам, рано или поздно проявляет определенное свойство, то динамика проявления этого свойства большим коллективом таких объектов обязательно будет экспоненциальной.

Но и это еще не все. А справедливо ли обратное утверждение: если динамика проявления определенного свойства большим коллективом одинаковых объектов экспоненциальная, то сами объекты являются сугубыми индивидуалами, существующими по своим внутренним законам? Ведь если это так, хотя бы даже отчасти, мы получаем в свое распоряжение мощное средство для исследования свойств отдельных представителей какого-либо класса по их коллективному поведению!

# Игра "Жизнь"

Ю.А. Соколинский

## Введение

Популярная игра “Жизнь” является, с одной стороны, увлекательным развлечением, а с другой — моделью биологической эволюции. Для учащихся классов с углубленным изучением информатики предлагается проект: разработка программы игры “Жизнь” и на этой основе знакомство с указанной игрой. Составление такой программы вполне по силам достаточно подготовленным учащимся. В основном требуется умение работать с циклами, массивами строк и символами. Отметим, что в тех случаях, когда требуется формализованная запись алгоритма, используется нотация, близкая к школьному алгоритмическому языку.

## 1. Игра “Жизнь” — развлечение и средство моделирования биологической эволюции

Игру “Жизнь” придумал в конце 60-х годов американский математик Конуэй (Conway), и она сразу же оказалась весьма популярной, поскольку позволяет выполнять интересные исследования и получать удивительные, неожиданные результаты. Для этой игры не требуется партнер — в нее можно играть и одному. Возникающие в процессе игры ситуации очень похожи на реальные процессы, происходящие при зарождении, развитии и гибели колоний (популяций) живых организмов. По этой причине “Жизнь” можно отнести к категории моделирующих игр, которые в той или иной степени имитируют процессы, происходящие в реальной жизни.

Для игры “Жизнь”, если не пользоваться компьютером, понадобятся большая (строго говоря, бесконечная) доска, разбитая на клетки, и много фишек двух цветов. Основная идея игры состоит в том, чтобы, начав с какого-нибудь простого расположения фишек (организмов, особей), расставленных по различным клеткам доски, проследить за эволюцией исходной позиции под действием “генетических законов” Конуэя, которые управляют рождением, гибелью и выживанием фишек. Конуэй тщательно подбирал свои правила и долго проверял их на практике, добиваясь, чтобы они удовлетворяли ряду условий, главные из которых:

1. Не должно быть ни одной исходной конфигурации, для которой существовало бы простое доказательство возможности неограниченного роста популяции.
2. Должны существовать простые начальные конфигурации, которые в течение значительного промежутка растут, претерпевают разнообразные изменения и заканчивают свою эволюцию одним из трех способов:

- полностью исчезают либо из-за перенаселенности, либо, наоборот, из-за разреженности жизненного пространства, т.е. популяция вырождается;
- переходят в устойчивую конфигурацию и перестают изменяться вообще — эволюция вышла на стационарный режим;
- выходят на колебательный режим, при котором совершают некий бесконечный цикл превращений с определенным периодом.

Короче говоря, правила игры должны быть такими, чтобы поведение популяции было интересным, а главное, непредсказуемым.

Прежде чем сформулировать законы жизни популяции, обратим внимание на то, что каждую клетку бесконечной доски окружают восемь соседних клеток: четыре имеют с ней общие стороны, а четыре другие — общие вершины.

Теперь приведем указанные законы (правила игры). Они довольно просты.

1. *Правило выживания.* Каждая фишка, у которой имеются две или три соседних, выживает и переходит в следующее поколение.
2. *Правило рождения.* Если число фишек, с которыми граничит какая-либо пустая клетка, равно трем, то на ней происходит рождение нового организма, т.е. следующим ходом на нее ставится одна фишка.
3. *Правило гибели.* Каждая фишка, у которой оказывается более трех соседних, погибает от перенаселения. Каждая фишка, вокруг которой свободны все клетки или занята только одна, погибает от одиночества.

Гибель и рождение всех организмов происходит одновременно. Выжившие и вновь рожденные организмы образуют одно поколение, или ход (шаг), эволюции начальной конфигурации.

При игре “вручную” (без использования компьютера) полезно использовать следующие рекомендации:

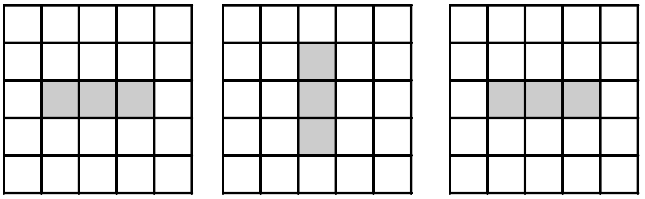
1) начать с конфигурации из черных фишек;
2) положить на каждую обреченную фишку по одной черной фишке;
3) найти все свободные клетки, на которых должен произойти акт рождения, и поставить на них белую фишку;
4) снять погибшие (столбики из двух фишек), а новорожденные белые заменить черными фишками.

Пора привести примеры.

### ПРИМЕР 1

Начальная конфигурация состоит из трех фишек, расположенных подряд, горизонтально, см. *рис.* 1*а*. Ясно, что крайние фишки погибают, так как у них только один сосед. Зато в свободных клетках, находящихся выше и ниже средней фишки, происходит рождение новых фишек, поэтому следующее поколение

состоит из трех фишек, расположенных подряд, но уже вертикально. Начальная конфигурация повернулась на 90°, см. *рис.* 1*б*. По аналогичным причинам следующее, второе, поколение опять повернется на 90° и совпадет с исходным, см. *рис.* 1*в*. Мы получили пульсирующий, периодический режим эволюции с периодом, составляющим два поколения. Данная начальная конфигурация имеет название "мигалка". (Среди любителей игры "Жизнь" принято давать характерным конфигурациям не слишком научные, но зато выразительные названия.)



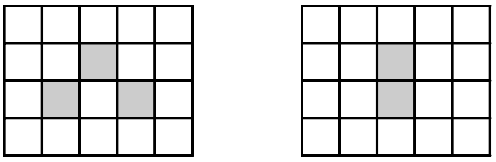
*а* *б* *в*

*Рис.* 1. "Мигалка"

А если взять начальную конфигурацию из трех вертикальных фишек, как на *рис.* 1*б*? Ясно, что она ведет себя аналогичным образом, и ее мы также будем называть "мигалкой".

**ПРИМЕР 2**

В начальной конфигурации, изображенной на *рис.* 2*а*, выживает лишь верхняя фишка, а рождение происходит в клетке, расположенной ниже, — она окружена тремя фишками. Следующее поколение (см. *рис.* 2*б*) состоит из двух фишек, из которых одна находится выше другой. Каждая из них имеет лишь одного соседа и, следовательно, погибает. Это пример вырождения популяции.

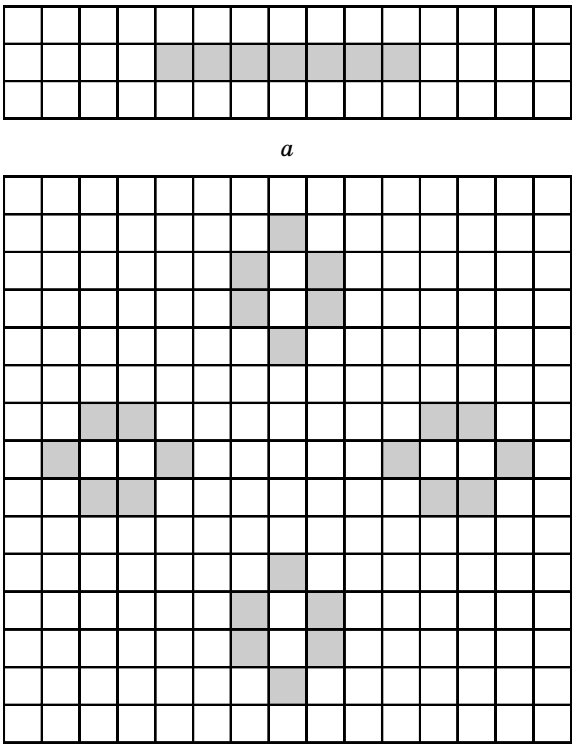


*а* *б*

*Рис.* 2. Вырождение популяции

**ПРИМЕР 3**

На *рис.* 3*а* изображена начальная конфигурация, состоящая из семи горизонтальных фишек, а на *рис.* 3*б* — ее 14-е поколение. Оно имеет название "улей", поскольку состоит из четырех "пасек". Попробуйте самостоятельно получить пропущенные 13 поколений, а также 14-е и 15-е. Если вы не запутаетесь, то убедитесь, что 15-е поколение ничем не отличается от предыдущего и является таким же "ульем". Значит, в данном случае эволюция на 14-м ходу вышла на стационарный режим. Сам "улей" и образующие его "пасеки" — это примеры устойчивых, самовоспроизводящихся конфигураций.



*а*

*б*

*Рис.* 3. "Пасека" (пример стационарного режима)

Последний пример говорит о том, что, как правило, играть в "Жизнь" "вручную" — это занятие, требующее времени и внимания. Ясно, что для получения удовольствия от игры и возможности изучать сложные ситуации требуется компьютер. И автор игры, Конуэй, и его многочисленные последователи разработали целый ряд вариантов программ игры "Жизнь", с помощью которых было получено множество интересных результатов. Давайте и мы прежде всего разработаем свою версию этой программы. Этой задаче посвящены следующие разделы.

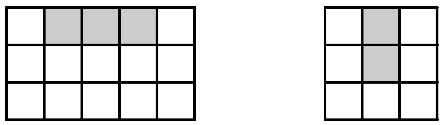
## 2. Организация программы игры "Жизнь"

Прежде всего остановимся на вопросах сценария программы, структуры основных данных, этапах разработки программы.

Ясно, что реализация концепции бесконечной доски (другими словами, бесконечного жизненного пространства) весьма затруднительна. Более того, от нее следует отказаться не только по техническим причинам, но и по соображениям принципиального характера. Пространство, которое занимает биологическая популяция, как правило, ограничено, и его нехватка — это одна из важнейших причин, препятствующих развитию популяции.

Отметим, что если фишки находятся на краю доски, то нельзя гарантировать корректность процесса эволюции. Приведем соответствующий пример. Возьмем в качестве начальной конфигурации три горизонтальные фишки ("мигалку"), но расположим их на верхнем краю доски, см. *рис.* 4*а*. На следующем шаге останется средняя фишка, а под ней окажется

новорожденная, см. *рис. 4б*. Как и в примере 2, обе эти фишки погибают, т.е. получилось вырождение популяции, а вовсе не колебательный режим.



*а* *б*

*Рис.* 4. "Мигалка" на краю доски

Поэтому при формировании начальной конфигурации запретим ставить фишки на край доски. Каждую новую конфигурацию будем располагать по центру доски. Если же это не поможет избежать появления фишек на граничных строках и/или столбцах доски, то, значит, достигнуты предельные размеры жизненного пространства — процесс эволюции завершен.

Доску конечных размеров можно моделировать либо двухмерным символьным массивом, либо одномерным массивом строк. Последний подход оказывается более удобным, и именно его мы будем придерживаться. Каковы размеры этого массива, т.е. длина строк и их количество? Будем исходить из следующего соображения: когда мы просматриваем эволюцию популяции, то каждое поколение должно целиком умещаться на экране монитора.

Изображением фишки ("организма") будет служить звездочка, т.е. символ "*". Конечно, можно выбрать любой другой символ, который кажется подходящим.

Видимо, представленные выше рисунки свидетельствуют, что нет необходимости рисовать сетку, разбивающую доску на клетки. Вместо этого пустые клетки будем помечать точкой, что повышает наглядность изображения доски с фишками (т.е. очередного поколения).

Запоминать образы поколений будем в текстовом файле. При этом первые и последние "свободные" строки (они содержат только точки) в файл не выводим. Поколения в файле отделяются друг от друга строкой, которая содержит слово "Поколение" и его порядковый номер. Номер начального поколения — нулевой. Естественно дать указанному файлу имя LIFE (без расширения).

Работу программы следует прекратить в следующих случаях:

— Популяция выродилась, т.е. очередное поколение оказалось пустым.
— Достигнуты предельные размеры (ширина или высота) жизненного пространства.
— Обнаружена периодичность эволюции, т.е. очередное поколение совпало с одним из предшествующих. При этом если период оказался равным единице, то популяция вышла на стационарный режим: все поколения, начиная с некоторого, совпадают.

Перед прекращением работы программы желательно вывести на экран соответствующее сообщение.

Просмотр файла можно выполнить после завершения программы с помощью любого текстового редактора. Например, подойдет редактор Edit, входящий в состав DOS 6.x, или редактор файловой системы Norton Commander, который вызывается клавишей F4.

Разработку программы выполним в два этапа, т.е. составим две версии: 1 и 2. Отличаться они будут лишь одной процедурой: генератором начального поколения (начальной конфигурации) и параметрами экрана (об этом см. ниже).

На первом этапе генератор будет выполнен предельно просто. После инициализации массива строк нового поколения установка фишек (организмов) производится с помощью операторов присваивания. Каждой фишке соответствует свой оператор. Конечно, такой вариант генератора начального поколения является лишь отладочным. Он позволит проверить работоспособность всех компонент программы и получить первые впечатления о самой игре "Жизнь".

На втором этапе мы разработаем усовершенствованный, наглядный генератор. Он позволит вывести на экране доску и расставлять на ней фишки с помощью соответствующих клавиш.

Теперь обсудим вопрос о структуре основных данных программы. Начнем с размеров доски, а точнее, с параметров массива строк, который ее моделирует. Обозначим через `NSscr`, `NCscr` число строк и столбцов (колонок) экрана. Конечно, длина строки должна быть максимально возможной, т.е. равной `NCscr`. Введем тип строки, на языке Паскаль он записывается как

```
type tStr = string[NCscr]
```

и аналогично на других алгоритмических языках.

Количество строк массива обозначим через `NSmax`. Как мы условились, поколение должно целиком поместиться на экране. Кроме того, на экране должна находиться разделительная строка, а также служебные строки редактора, который используется для просмотра (они содержат меню и подсказки). Пусть `Nhelp` — количество служебных строк. Можно принять `Nhelp=3`. Следовательно,

```
NSmax = NSscr − 1 − Nhelp
```

Создаем тип *поколение*, который на Паскале имеет вид:

```
type tGen = array[1..NSmax] of tStr
```

Основными переменными программы будут `Gen1` и `Gen2` — *старое* и *новое* поколения, они имеют тип `tGen`. Кроме того, используются следующие глобальные переменные:

`Count` — счетчик поколений;

`CountP` — номер поколения, начиная с которого эволюция носит периодический характер;

`S0` — строка, состоящая из точек ("свободная" строка);

`Fout` — логический текстовый файл, в котором хранятся образы поколений.

В Паскале эта переменная объявляется как `Fout: text`, в Си — как `FILE *Fout`. В Бейсике роль такой переменной играет порядковый номер файла, который указывается при его открытии.

Выше мы использовали параметры текстового экрана `NSscr`, `NCscr`, но не указали их значений. Восполним этот пробел. Обычные значения указанных параметров составляют: `NSscr = 25`, `NCscr = 80`. Это означает, что по умолчанию текстовый экран состоит

из 25 строк и 80 колонок. Чтобы изменить эти характеристики, надо предпринять специальные усилия. Спрашивается, а зачем их изменять? Попробуйте с помощью какого-либо текстового редактора изобразить квадрат из символов типа звездочек, точек и т.п. На экране этот квадрат будет выглядеть как вытянутый по вертикали прямоугольник. Объясняется такое явление тем, что высота области символа вдвое больше ширины. Для нас этот эффект нежелателен, поскольку он затрудняет зрительное восприятие конфигураций, особенно сложных, см., например, *рис.* 10 (с. 21).

Как сделать размеры символа одинаковыми? Надо либо удвоить число строк экрана, либо уменьшить вдвое число его колонок. Первый способ удобен, когда мы пользуемся текстовым редактором для просмотра файла поколений. Обычно такие редакторы имеют средства переключения в режим "50 строк": для редактора Edit надо при его вызове указать опцию `/H`, а для редактора Norton Commander, вызываемого клавишей F4, можно воспользоваться меню команд или нажать клавиши Alt F9.

При разработке программы, в которой изображения выводятся на текстовый экран, удобнее второй способ: перейти в режим "40 колонок". Мы воспользуемся им на втором этапе, чтобы обеспечить наглядность генератора начального поколения. Как перейти в режим "40 колонок"? В Паскале для этого надо выполнить команду `TextMode(CO40)`, в Си — `textmode(C40)`, в Бейсике (версия QBasic) — `SCREEN 1`.

Итак, на первом этапе (программа 1) параметры экрана составляют `NSscr = 25`, `NCscr = 80`, а на втором (программа 2) число строк не меняется, а число колонок уменьшается вдвое: `NCscr = 40`. Обратите внимание: после выхода из программы 2 восстанавливается текстовый режим 80 × 25, поэтому и здесь редактор, с помощью которого просматривается файл поколений, надо перевести в режим "50 строк". Однако длина строки, равная 40, разумеется, не изменится и на экране поколения будут занимать половину его ширины.

## 3. Вспомогательные функции

Разделим процедуры и функции на два класса: основные и вспомогательные. Основные процедуры и функции — это те, которые вызываются из основного алгоритма, остальные — вспомогательные. Как мы увидим ниже, основными являются только процедуры, а вспомогательными — функции. С них и начнем.

**3.1.** Функция `Near(i,j,Gen)` выдает число соседних фишек. Входы функции:

— индексы клетки `i`, `j`, для которой находится число соседних фишек, т.е. `i` — номер строки, `j` — номер столбца, в которых находится заданная клетка;

— текущее поколение `Gen` типа `tGen` (в дальнейшем мы не будем оговаривать тип поколений).

Алгоритм заключается в следующем.

Вводится счетчик `n`, вначале равный нулю. Просматривается по очереди каждая соседняя клетка, и если она содержит фишку, то наращивается значение счетчика. Его последнее значение и выдает функция `Near`. Запишем этот алгоритм подробнее и более формально:

```
n:=0
нц для ii от i-1 до i+1
| нц для jj от j-1 до j+1
| | если
| | | (ii>0) и (ii<=NSmax) и (jj>0)
| | | и (jj<=NCscr) и ((ii<>i)
| | | или (jj<>j)) и (Gen[ii][jj]='*')
| | | то n:=n+1
| | кон если
| кц
кц
Near:=n
```

**3.2.** Функция `EmptyLine(i, Gen)` логического типа выдает значение **истина**, если `i`-я строка текущего поколения "свободная", т.е. не содержит фишек, а иначе **ложь**. Входные переменные: номер строки `i` и текущее поколение `Gen`.

Алгоритм достаточно прост.

Полагаем `EmptyLine:=`**истина**. Затем в цикле просматриваем заданную строку от начала до конца, т.е. от `1` до `NCscr`. Если очередная позиция `j`-й строки содержит звездочку `(Gen[i][j]='*')`, то полагаем `EmptyLine:=`**ложь** и выходим из функции.

**3.3.** Функция `EmptyCol(j, Gen)` логического типа выдает значение **истина**, если `j`-я колонка текущего поколения "свободная", т.е. не содержит фишек, а иначе **ложь**. Входные переменные: номер колонки `j` и текущее поколение `Gen`.

Алгоритм аналогичен предыдущему. Полагаем `EmptyLine:=`**истина**. Затем в цикле просматриваем заданную колонку от начала до конца, т.е. от `1` до `NSmax`. Если очередная позиция `i`-й колонки содержит звездочку `(Gen[i][j]='*')`, то полагаем `EmptyLine:=`**ложь** и выходим из функции.

## 4. Основные процедуры

Здесь мы рассмотрим основные процедуры, за исключением генератора начального поколения.

**4.1.** Процедура `NewGen(Gen1, Gen2)` осуществляет переход к новому поколению. Ее вход — старое поколение `Gen1`, выход — новое поколение `Gen2`.

Алгоритм реализует "законы жизни популяции" Конуэя. Сначала происходит инициализация нового поколения:

```
нц для i от 1 до NSmax Gen2[i]:=S0 кц
```

Как мы помним, `S0` — строка, состоящая из точек ("свободная" строка).

Затем отслеживаются фишки, которые выживают и переходят в новое поколение. Это те фишки, для которых число соседних фишек составляет два или три. Формальная запись этого правила:

```
нц для i от 1 до NSmax
| нц для j от 1 до NCscr
| | если (Gen1[i][j]='*') и
| | |    ((Near(i, j, Gen1) =2) или
| | |    (Near(i, j, Gen1) =3))
| | | то Gen2[i][j] := '*'
| | кон если
| кц
кц
```

Осталось осуществить "рождение" новых фишек. Согласно Конуэю, фишка рождается на свободной клетке, если число ее соседних фишек равно 3. Запишем этот процесс в виде:

```
нц для i от 1 до NSmax
| нц для j от 1 до NCscr
| | если (Gen1[i][j]='.') и
| | |    (Near(i, j, Gen1)=3)
| | | то Gen2[i][j]:='*'
| | кон если
| кц
кц
```

**4.2.** Процедура

`Transform(Gen2,Gen1,I1,H,SignEnd)`

преобразует новое поколение `Gen2` в старое `Gen1`.

*Вход процедуры* — новое поколение `Gen2`.

*Выходные параметры:*

`Gen1` — старое поколение;

`I1` — номер первой строки `Gen1`, содержащей фишки;

`H` — "высота" `Gen1`, т.е. число строк, начиная с `I1` и кончая `I2`, где `I2` — номер первой снизу строки `Gen1`, содержащей фишки.

*Выходной параметр* `SignEnd` — признак конца. Он имеет значения:

1 — при вырождении нового поколения `Gen2`,

2 — при достижении предельной высоты,

3 — при достижении предельной ширины.

В остальных случаях его значение — ноль.

Идея преобразования заключается в том, что новая конфигурация (если она не вырождена) переносится в `Gen1` таким образом, чтобы она расположилась по центру. Если конфигурация не достигла своих предельных размеров, то это обеспечит "свободу" граничных строк и столбцов, т.е. они не будут содержать фишек.

Изложим сущность преобразования более подробно. Находятся номера "окаймляющих" строк `Imin`, `Imax` и столбцов `Jmin`, `Jmax`. Это означает, что `Imin` — номер первой сверху строки `Gen2`, содержащей фишки, а `Imax` — номер первой снизу такой же строки. Аналогично, `Jmin`, `Jmax` — номера первой слева и справа колонки `Gen2`, содержащей фишки. Если поиск `Imin` оказался безрезультатным, то это означает вырожденность `Gen2`. Иначе прямоугольник, окаймленный указанными строками и столбцами, переносится в `Gen1` так, чтобы он расположился по центру, насколько это возможно. Если окажется, что высота этого прямоугольника $H \geq NSmax-1$, то это означает существование фишек в граничных строках (в первой и/или последней). Аналогично, если ширина прямоугольника больше либо равна $NCscr-1$, то существуют фишки в граничных столбцах. В первом случае признак конца `SignEnd` получает значение 2, во втором — 3.

Теперь приведем формальное описание алгоритма.

```
SignEnd:=0; Imin:=1
нц пока (EmptyLine(Imin,Gen2)) и (Imin<NSmax)
| Imin:=Imin+1
кц
если (Imin=NSmax) и (EmptyLine(Imin,Gen2))
| то
| SignEnd:=1 |Вырождение популяции
| выход из процедуры
кон если
Imax:=NSmax
нц пока EmptyLine(Imax,Gen2)
| Imax:=Imax-1
кц
Jmin:=1
нц пока EmptyCol(Jmin,Gen2)
| Jmin:=Jmin+1
кц
Jmax:=NCscr;
нц пока EmptyCol(Jmax,Gen2)
| Jmax:=Jmax-1
кц
H:=Imax-Imin+1; W:=Jmax-Jmin+1
если H>=NSmax-1 то SignEnd:=2 кон если
если W>=NCscr-1 то SignEnd:=3 кон если
нц для i от 1 до NSmax Gen1[i]:=S0 кц
Чистка Gen1
| I1, J1 — номер строки и столбца левого
| верхнего угла центрированной конфигурации
I1:=(NSmax-H) div 2+1
J1:=(NCscr-W) div 2+1
| Копируем Gen2 в Gen1, центрируя новую
| конфигурацию
нц для i от Imin до Imax
| нц для j от Jmin до Jmax
| | Gen1[I1+i-Imin][J1+j-Jmin]:=Gen2[i][j]
| кц
кц
```

**4.3.** Процедура `Out(Gen, I1, H, Count)` выводит очередное поколение `Gen` в файл поколений `Fout`. Как уже говорилось, первые и последние "свободные" строки в файл не выводятся. Входные параметры процедуры:

`Gen` — очередное поколение;

`I1` — номер первой строки `Gen`, содержащей фишки;

`H` — число строк `Gen`, содержащих фишки;

`Count` — номер поколения.

Алгоритм работы процедуры довольно прост:

```
вывод нс (Fout,'Поколение', Count)
нц для i от I1 до I1+H-1
| вывод нс (Fout, Gen[i])
кц
```

**4.4.** Процедура `Period(Gen, I1, H, CountP)` ищет в файле поколений "близнеца" очередного поколения `Gen`, т.е. такое поколение, которое совпадает с `Gen`.

*Входные параметры процедуры:*

Gen — очередное поколение;

I1 — номер первой строки Gen, содержащей фишки;

H — число строк Gen, содержащих фишки.

*Выходной параметр:*

CountP — номер поколения-"близнеца". Если "близнец" не нашелся, то

CountP=–1.

Нахождение "близнеца" означает, что, начиная с поколения CountP, режим эволюции — периодический.

*Алгоритм* поиска "близнеца":

Закрываем файл поколений Fout и открываем его на чтение. Читаем первую строку файла, содержащую заголовок начального поколения. Полагаем:

Count=0; CountP=–1.

(Здесь Count не глобальная, а локальная переменная данной процедуры.)

Начинаем цикл анализа поколений:

- Полагаем счетчик строк поколения i=0.
- Выполняем цикл чтения очередного поколения. Он заканчивается, когда прочитанная строка является заголовком следующего поколения либо последней строкой файла. Поколение записывается в массив Gen0, число записанных строк хранит счетчик i.
- Если i=H, то проверяем совпадение массивов Gen0 и Gen. В случае совпадения полагаем CountP=Count.
- Закрываем файл Fout и открываем его на дозапись (append).
- Выходим из процедуры.

Если i<>H или Gen0>Gen, то наращиваем счетчик поколений Count.

Конец цикла анализа поколений, когда прочитана последняя строка файла.

Закрываем файл Fout и открываем его на дозапись.

Конец алгоритма.

Запишем его подробнее и более формально:

```
закрыть файл(Fout)
открыть файл(Fout, 'r')
  | файл открывается на чтение
ввод нс(Fout); Count:=0; CountP:=-1
нц
| i:=0
| нц
| | ввод нс(Fout, S)
| | |S хранит очередную строку
| | если S[1]='П'
| | | | заголовок поколения имеет вид
| | | "Поколение <номер>"
| | то NewGen:=истина
| | иначе
| | NewGen:=ложь
| | i:=i+1
| | Gen0[i]:=S
| кон если
| кц при (конец файла(Fout)) или (NewGen)
| если i=H
| | то
| | j:=1
| | нц пока (j<i) и (Gen0[j]=Gen[I1+j-1])
| | | j:=j+1
| | кц
| | если (j=i) и (Gen0[j]=Gen[I1+j-1])
| | | то
| | | закрыть файл(Fout)
| | | открыть файл(Fout, 'a')
| | | | файл открывается на дозапись
| | | CountP:=Count
| | | выход из процедуры
| | кон если
| кон если
| Count:=Count+1
кц при конец файла(Fout)
закрыть файл(Fout)
открыть файл(Fout, 'a')
```

Здесь функция конец файла(F) выдает значение **истина**, когда прочитана последняя строка файла F, а иначе **ложь**.

## 5. Этап 1. Генератор начальной конфигурации и основной алгоритм

Все приведенные выше процедуры и функции будут использоваться как в первом, так и во втором варианте программы. Сейчас мы составим вариант генератора начальной конфигурации, который предназначен для первого этапа, т.е. для программы 1. Как уже указывалось, он весьма прост и, можно даже сказать, примитивен. Естественно, при этом пользователь имеет минимум удобств, а главное, он должен ориентироваться в данной программе, так что этот вариант надо рассматривать как отладочный.

Процедура StartGen1(Gen) создает первоначальное новое поколение (начальную конфигурацию) Gen. Конфигурация фишек, рассматриваемая как единое целое, может располагаться внутри массива произвольно. Ведь процедура Transform, которая будет вызвана вслед за StartGen1, преобразуя первоначальное новое поколение в старое, расположит его по центру массива.

Алгоритм процедуры заключается в том, что сначала массив Gen чистится:

нц для i от 1 до NSmax Gen2[i]:=S0 кц,

— а затем расстановка фишек производится соответствующими операторами присваивания, количество которых определяется числом фишек. Разумеется, предыдущие операторы присваивания надо стереть или превратить в комментарий. Как мы помним, фишки не желательно ставить на край доски, однако в процедуре это не контролируется.

Например, возьмем в качестве начальной конфигурации "мигалку", т.е. три фишки, расположенные подряд горизонтально.

Надо написать три оператора присваивания, которые могут иметь вид:

```
Gen[2][2]:='*';
Gen[2][3]:='*';
Gen[2][4]:='*'
```

С таким же успехом их можно записать и так:

```
Gen[5][10]:='*';
Gen[5][11]:='*';
Gen[5][12]:='*'
```

— и т.д.

Теперь мы в состоянии сформулировать основной алгоритм программы 1.

- Присваиваем значения параметрам экрана:
  NSscr:=25; NCscr:=40;
  Nhelp:=3; NSmax:=NSscr-(NHelp+1)
- Присваиваем начальное значение счетчику поколений Count:=0
- Формируем "чистую" строку S0:
  S0 := '';
  нц для i от 1 до NCscr
  | S0:=S0+ '.'
  кц
- Связываем логический файл поколений Fout с физическим файлом Life и открываем файл Fout на запись.
- Вызываем генератор начальной конфигурации StartGen1(Gen2)
- Начало основного цикла
- Вызываем процедуру преобразования новой конфигурации Gen2 в старую Gen1:
  Transform(Gen2,Gen1,I1,H,SignEnd).
- Напомним, что
  I1 — номер первой строки Gen, содержащей фишки,
  H — число строк Gen, содержащих фишки,
  SignEnd — признак конца.
  | если поколение не начальное и не
  | вырожденное
  если (Count>0) и (SignEnd<>1)
  |то Period(Gen1,I1,H,CountP)
  | | вызов процедуры поиска "двойника"
  | | если нашелся "двойник" (режим стал
  | | периодическим), то признаку конца
  | | даем значение 4
  | если CountP>=0 то SignEnd:=4 кон если
  кон если
- Если очередное поколение не вырожденное (SignEnd <>1), то вызываем процедуру записи очередного поколения в файл поколений Out(Gen1,I1,H,Count) и наращиваем счетчик поколений Count.
- Если достигнут конец процесса эволюции (SignEnd > 0), то выводим на экран соответствующее сообщение и приостанавливаем работу программы до нажатия любой клавиши (либо клавиши Enter), а иначе вызываем процедуру создания нового поколения NewGen(Gen1,Gen2).
- Конец основного цикла при SignEnd>0.
- Закрываем файл поколений Fout.

Конец основного алгоритма программы 1.

Составив программу 1, можно проверить работоспособность всех ее компонент и убедиться, что для простых начальных конфигураций (можно использовать примеры раздела 1 и, в частности, пример 3) программа действительно порождает новые поколения в соответствии с законами Конуэя.

## 6. Этап 2. Наглядный генератор начальной конфигурации и уточнение основного алгоритма

Переходим ко второму, основному, этапу. Здесь нам надо разработать усовершенствованный, наглядный генератор, позволяющий вывести на экран доску и расставлять на ней фишки с помощью соответствующих клавиш, а также уточнить основной алгоритм.

Начнем с *наглядного генератора начальной конфигурации.*

Естественно использовать привычные управляющие клавиши: стрелки для сдвига курсора по горизонтали и вертикали, клавиши End и Home для установки курсора в конец и начало строки, клавиши Page Down и Page Up для установки курсора в конец и начало массива. Устанавливать фишку будем нажатием клавиши Enter. Если она уже установлена, то нажатие клавиши Enter, наоборот, снимает фишку. Для выхода из генератора используем клавишу Esc.

Доску расположим в верхней части экрана. Внизу останутся Nhelp+1 (т.е. 4) не использованные строки. В трех нижних строках можно разместить подсказку, например, такую:

```
"стрелки — управление курсором"
"Enter — уст/снять фишку Esc — выход"
"Home,End,PgUp,PgDn — нач/кон строки/поля".
```

Мы знаем, как в программу ввести с клавиатуры тот или иной символ. Для этого в Паскале используется функция ReadKey, в Си — getch( ), в Бейсике — INKEY$. А как передать в программу сообщение о том, что нажата какая-то специальная клавиша? Прежде всего клавиши Enter и Esc не отличаются в этом отношении от обычных. Для них в таблице символов предусмотрены индексы: 13 для Enter и 27 для Esc, и, следовательно, указанные функции позволят узнать, что нажата какая-то из этих клавиш. Для остальных специальных клавиш их нажатие моделируется последовательным нажатием двух клавиш. Первой из них соответствует нулевой индекс, а так как ни один символ клавиатуры не имеет такого индекса, то нулевой индекс служит признаком нажатия какой-то специальной клавиши. Индекс второго "прочтенного" символа (так называемый скан-код) и говорит, какая специальная клавиша нажата. Приведем таблицу скан-кодов тех специальных клавиш, которые нам понадобятся:

| ↑ | ↓ | → | ← | End | Home | Page Up | Page Down |
|---|---|---|---|---|---|---|---|
| 72 | 80 | 77 | 75 | 79 | 71 | 81 | 73 |

Запретим ставить фишки в граничные строки и столбцы. Это гарантирует, что начальная конфигурация не будет занимать все жизненное пространство.

Нам понадобится вспомогательная процедура `Star(Xc,Yc,Gen)`, которая устанавливает или снимает фишку. Алгоритм этой процедуры:

- Устанавливаются черный (Black) цвет текста и светло-серый (LightGray) цвет фона.
- Курсор устанавливается в позицию Xc,Yc. (В Паскале для этого используется команда gotoXY, в Си — gotoxy, в Бейсике — LOCATE.)
- Если символ Gen[Yc][Xc] — это звездочка, то в позицию Xc,Yc массива Gen и экрана заносится точка, и наоборот. Точнее:

```
если Gen[Yc][Xc]='*'
| то
|    Gen[Yc][Xc]:='.'
|    вывод ('.')
| иначе
|    Gen[Yc][Xc]:='*'
|    вывод ('*')
кон если
```

- Курсор снова устанавливается в позицию Xc,Yc (поскольку он сдвинулся на одну позицию вправо).

Конец алгоритма.

Теперь мы можем составить наглядный генератор начальной конфигурации, т.е. процедуру `StartGen2(Gen)`.

Изложим алгоритм работы этого генератора.

- Выполняем чистку экрана.
- Устанавливаем светло-серый цвет фона и черный цвет текста.
- Производим чистку массива Gen и первых NSmax строк экрана:

```
нц для i от 1 до NSmax
| Gen[i]:=S0 |Чистка Gen
| вывод (S0)   |Чистка экрана
кц
```

- Выводим подсказку в нижние три строки.
- Даем координатам курсора начальные значения: Xc=2; Yc=2 — и устанавливаем курсор в начальную позицию.
- Выполняется цикл клавиатуры, который запишем в формальном стиле:

```
нц
| Ich:=индекс(ввод символа)
| | Ich — индекс символа, введенного
|    с клавиатуры
| | выбор Ich
|    27: | Esq
|    чистка экрана
| выход из процедуры
| 13: | Enter
| Star(Xc,Yc,Gen)
| | установка/снятие фишки
| 0: | специальная клавиша
|    SC:=индекс(ввод символа)
|    | SC — скан-код специальной клавиши
|    выбор SC
|    72: | Курсор вверх
|    если Yc>2 то Yc:=Yc-1 кон если
|    75: | Курсор влево
|    если Xc>2 то Xc:=Xc-1 кон если
|    77: | Курсор вправо
|    если Xc<NCscr-1 то Xc:=Xc+1 кон если
|    80: | Курсор вниз
|    если Yc<NSmax-1 то Yc:=Yc+1 кон если
|    71: | Home Курсор к началу строки
|    Xc:=2
|    79: | End Курсор к концу строки
|    Xc:=NCscr-1
|    73: | PgUp Курсор к началу поля
|    Xc:=2; Yc:=2
|    81: | PgDn Курсор к концу поля
|    Xc:=NCscr-1; Yc:=NSmax-1
|    кон выбор
|    Установка курсора в позицию Xc,Yc
|    кон выбор
кц
```

Конец алгоритма наглядного генератора.

Нам осталось сформулировать *основной алгоритм программы* 2. Но поскольку он мало отличается от своего предшественника — *основного алгоритма программы* 1, то мы укажем лишь, в чем заключаются эти различия. Их всего два.

1. Число колонок экрана теперь не 80, а 40, т.е. параметр `NCscr` имеет значение `NCscr=40`.

2. В начале основного алгоритма надо поместить команду перехода в режим "40 колонок". Как уже указывалось, в Паскале она имеет вид `TextMode(CO40)`, в Си — `textmode(C40)`, в Бейсике (версия QBasic) — `SCREEN 1`.

## 7. Играем в "Жизнь"

Итак, мы составили и отладили программу 2, позволяющую удобным способом формировать начальную конфигурацию и следить за ее эволюцией. Давайте воспользуемся этой программой и посмотрим, на что способна игра "Жизнь".

Сейчас мы приведем несколько задач, выполнив которые, вы познакомитесь с некоторыми достижениями в этой области и интересными конфигурациями. Сначала о терминологии: триплетами называют конфигурации из трех фишек, каждая из которых имеет хотя бы одного соседа. Тетрамино и пентамино — это аналогичные конфигурации из четырех или пяти фишек, связанных между собой "ходом ладьи".

### ЗАДАЧА 1

Исследуйте эволюцию всех возможных триплетов. (В их число входят начальные конфигурации примеров 1 и 2 раздела 1.) Покажите, что эволюция триплетов, симметричных относительно вертикальной или горизонтальной оси, протекает одинаково.

## ЗАДАЧА 2

Исследуйте эволюцию пяти тетрамино, изображенных на *рис.* 5. (Если не учитывать симметричные конфигурации, то это все возможные тетрамино.)

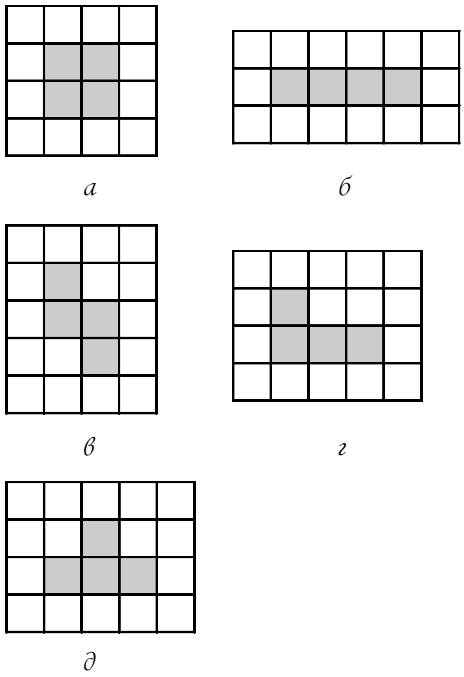
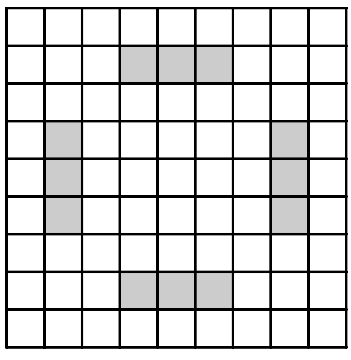



*Рис.* 5. Пять вариантов тетрамино

Покажите, что:

- Первая из них, имеющая название "блок", является устойчивой конфигурацией.
- Следующие три на некотором шаге (каком именно?) переходят в уже знакомую нам устойчивую конфигурацию "пасеку" (пример 3 раздела 1).
- Пятая на каком-то шаге (опять-таки, каком?) переходит в конфигурацию, изображенную на *рис.* 6*а*. Она называется "навигационные огни" и состоит из четырех "мигалок". На следующем шаге "мигалки" поворачиваются на 90°, см. *рис.* 6*б*, далее они опять поворачиваются на тот же угол, т.е. восстанавливаются "навигационные огни", *рис.* 6*в*. Эволюция стала периодической с периодом два.



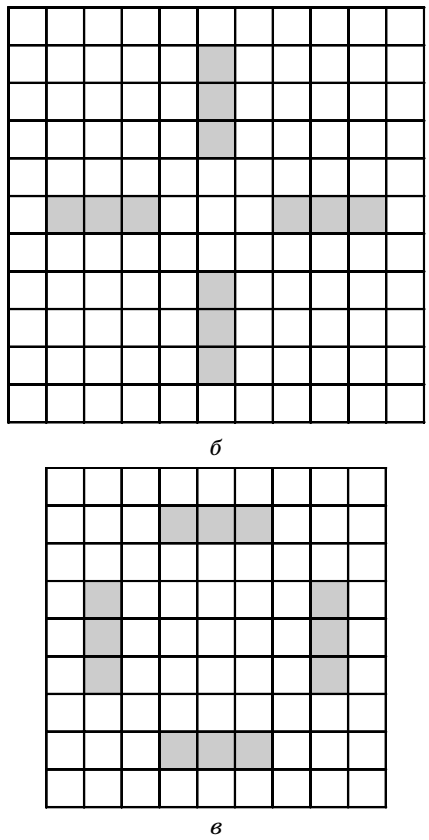
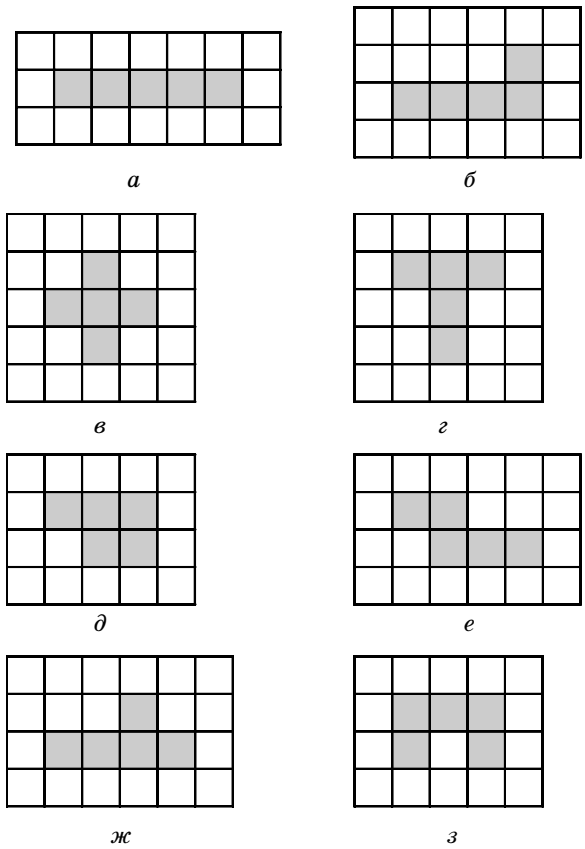
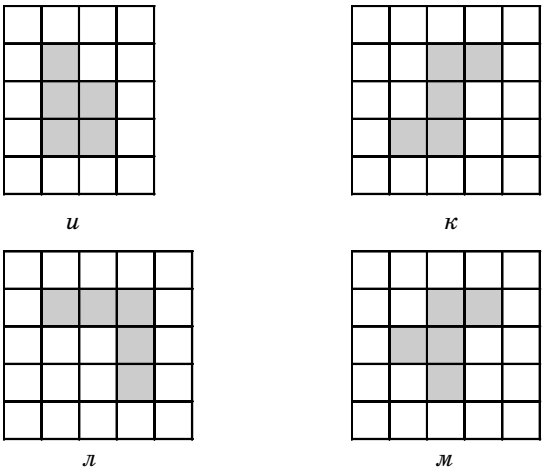




*Рис.* 6. "Навигационные огни"

## ЗАДАЧА 3

Исследуйте эволюцию 12 пентамино, изображенных на *рис.* 7.





*и* *к* *л* *м*

*Рис.* 7. 12 вариантов пентамино

Покажите, что:

- Первые четыре варианта дают "навигационные огни".
- Варианты 5—10 приводят к вырождению.
- Вариант 11 приводит к устойчивой конфигурации "каравай", изображенной на *рис.* 8.
- Для последнего варианта, известного как *r*-пентамино, на 70-м ходу достигается предельная ширина жизненного пространства.

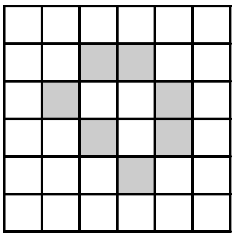
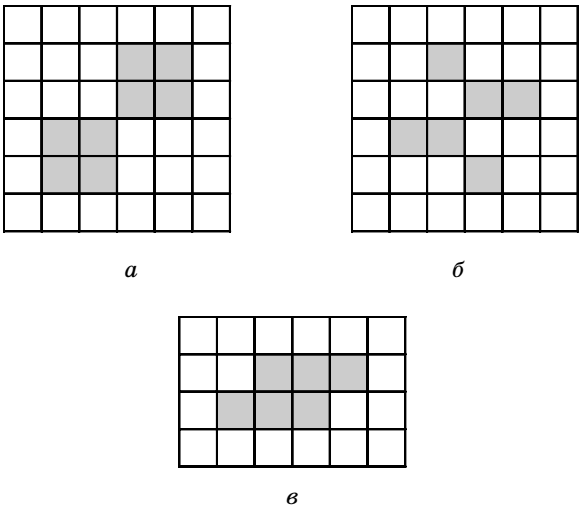


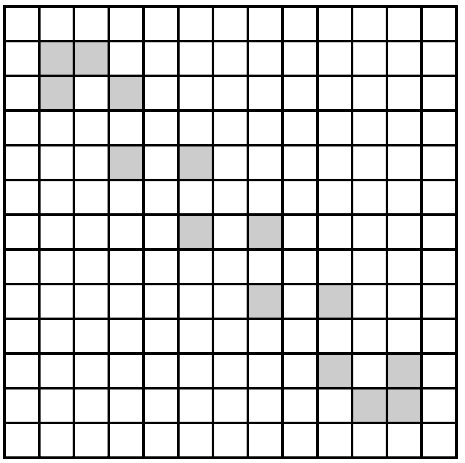
*Рис.* 8. "Каравай"

## ЗАДАЧА 4

Покажите, что конфигурации, показанные на *рис.* 9: "бакен" (*рис.* 9*а*), "часы" (*рис.* 9*б*), "жаба" (*рис.* 9*в*), "палка" (*рис.* 9*г*), — являются "флип-флопами", т.е. повторяются через каждые два шага.



*а* *б* *в*



*г*

*Рис.* 9. "Флип-флопы"

## ЗАДАЧА 5

Покажите, что внутренняя часть конфигурации "вертушка", изображенной на *рис.* 10, поворачивается на 90° по часовой стрелке на каждом ходу, а все блоки остаются на месте, т.е. эволюция этой конфигурации периодическая с периодом 4.

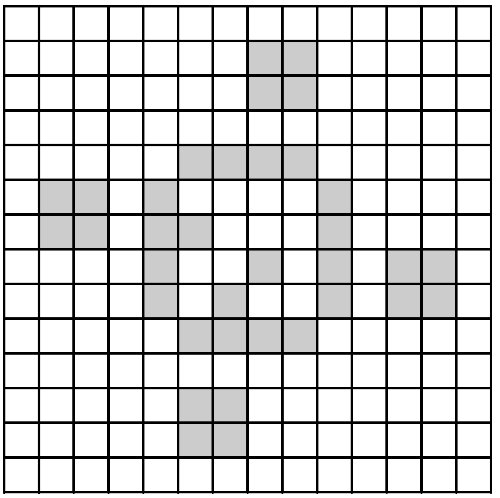


*Рис.* 10. "Вертушка"

## ЗАДАЧА 6

Покажите, что конфигурация "восьмерка" (см. *рис.* 11), не только внешне напоминает эту цифру, но и повторяется циклически через каждые 8 шагов.

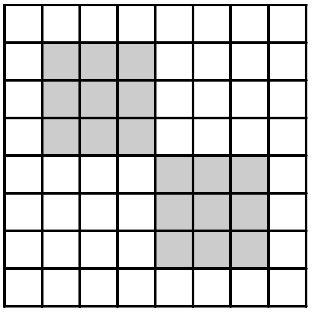


*Рис.* 11. "Восьмерка"

### ЗАДАЧА 7

Исследуйте эволюцию "глайдера", см. *рис.* 12. Покажите, что он воспроизводится через каждые 4 хода (английское слово *glider* означает планер).

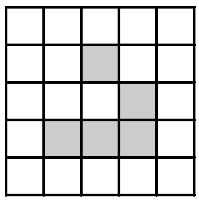


*Рис.* 12. "Глайдер"

### ЗАДАЧА 8

Проверьте, что *П*-гептамино (*рис.* 13) на некотором шаге (определите его номер) превращается в "пульсар CP 48-56-72", т.е. конфигурацию, изображенную на *рис.* 14, который, в свою очередь, повторяется после следующих трех шагов. Другими словами, эволюция *П*-гептамино на некотором шаге выходит на пульсирующий режим с шагом 3.

Откуда взялись такое название и обозначение? Пульсары — это космические объекты, обладающие строго периодическими характеристиками. Их открытие в 60-х годах было настоящей научной сенсацией. В названии обыгрывается не только тема пульсаров, но и способ их обозначения. Число 48 — это количество фишек пульсара, а 56 и 72 — числа фишек следующих двух поколений (проверьте!).

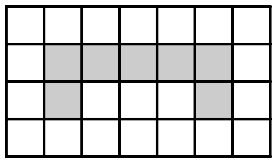


*Рис.* 13. *П*-гептамино

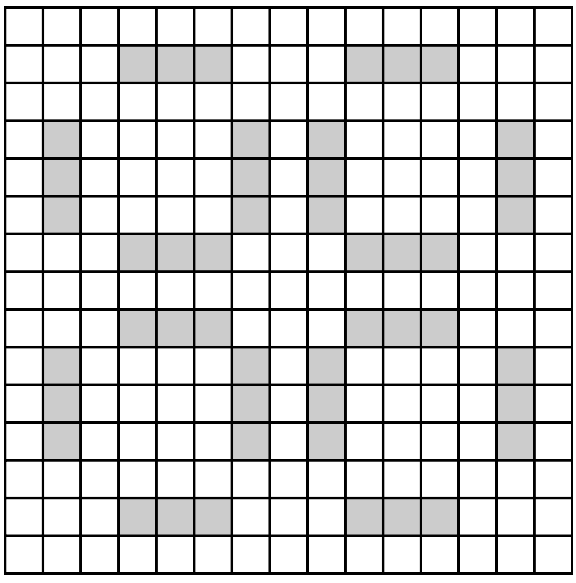


*Рис.* 14. "Пульсар CP 48-56-72"

### ЗАДАЧА 9

Проследите за судьбой "Чеширского кота", *рис.* 15. Убедитесь, что на шестом ходу от него остается только улыбка (*рис.* 16), которая затем пропадает, и остается лишь след лапы (блок из четырех фишек). (Напомним, что Чеширский кот — это персонаж сказки Л.Кэрролла "Алиса в стране чудес". Он обладает чудесной способностью исчезать, оставляя свою улыбку.)

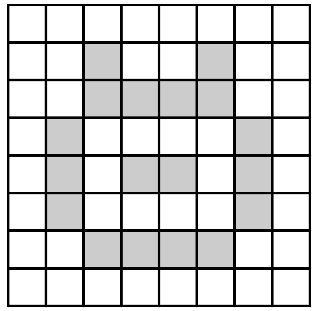


*Рис.* 15. "Чеширский кот"

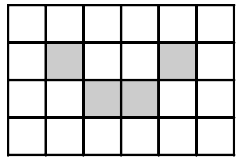


*Рис.* 16. "Улыбка кота"

Мы рассказали лишь малую часть того, что удалось открыть и исследовать любителям игры "Жизнь". Еще много интересного об этой игре можно найти в прекрасных книгах М.Гарднера [1, 2], рекомендуем ознакомиться с ними.

Но самый главный совет: самим попробовать эту игру, отыскать интересные, неожиданные конфигурации, проследить за их превращениями. Без сомнения, вы получите много удовольствия, погрузившись в "Жизнь".

## Литература


1. *Гарднер М.* Крестики-нолики. М.: Мир, 1988.
2. *Гарднер М.* Математические новеллы. М.: Мир, 1974.
3. *Фаронов В.В.* Программирование на персональных ЭВМ в среде Турбо Паскаль. М.: изд-во МГТУ, 1991.
4. *Каспер Е.* Освоим QBasic играючи! М.: Горячая линия — телеком, Радио и связь, 1999.
5. *Болски М.И.* Язык программирования Си. М.: Радио и связь, 1988.

# Задачи о взвешиваниях

Д.М. Златопольский

**В газете "Информатика" № 19/99 была опубликована задача олимпиады института Вейцмана, связанная со взвешиванием монет. Как показывает опыт, такие задачи вызывают у учащихся большой интерес, и поэтому их целесообразно рассматривать в контексте разработки алгоритмов, их анализа, при изучении основных алгоритмических конструкций (ветвлений и др.) и т.п. В связи с этим предлагаем еще несколько задач на эту тему.**

## ЗАДАЧА 1

**Даны 80 монет, одна из них фальшивая, более легкая, чем все остальные, имеющие одинаковый вес. Необходимо за 4 взвешивания найти фальшивую монету.**

Эта задача, выражаясь спортивным языком, для разминки. Алгоритм ее решения достаточно хорошо известен, поэтому здесь приведем несколько заданий, которые можно предложить учащимся при его обсуждении.

***Задание 1.*** Дано $n$ монет. Необходимо поделить монеты на 3 примерно равные группы (записать формулы для определения числа монет в каждой из трех групп).

***Решение***

Если число монет в группах обозначить соответственно $K(1)$, $K(2)$ и $K(3)$, то искомые значения можно рассчитать, рассуждая следующим образом.

Число монет в первой группе должно составлять треть от общего числа:

$$K(1)=n \text{ div } 3, \qquad (1)$$

— где div — целочисленное деление.

Оставшиеся монеты необходимо разделить пополам, т.е. число монет во второй группе равно:

$$K(2)=(n - K(1)) \text{ div } 2, \qquad (2)$$

а в третьей:

$$K(3)= n - K(1) - K(2) \qquad (3)$$

**Задание 2.** Доказать, что при разделении $n$ монет в соответствии с формулами (1)—(3) как минимум две группы будут иметь одинаковое число монет и число монет в группах будет отличаться не более чем на 1.

***Решение***

Очевидно, что значение $n$ может соответствовать одному из трех случаев: быть кратным трем ($n = 3m$, где $m = 1, 2, 3, ...$), быть на 1 меньше, чем в первом случае, и быть меньше на 2.

Требуемое доказательство следует из таблицы:

| Рассчитываемые значения | Варианты значения $n$ | | | Примечание |
|---|---|---|---|---|
| | $3m$ | $3m - 1$ | $3m - 2$ | |
| $K(1)$ | $m$ | $m - 1$ | $m - 1$ | По формуле (1) |
| $n - K(1)$ | $2m$ | $2m$ | $2m-1$ | |
| $K(2)$ | $m$ | $m$ | $m - 1$ | По формуле (2) |
| $K(3)$ | $m$ | $m$ | $m$ | По формуле (3) |

**Задание 3.** $N$ монет разделены на три группы в соответствии с формулами (1)—(3). Определить номера двух любых групп с одинаковым числом монет, а также номер оставшейся группы.

***Решение***

Если номер первой из групп с одинаковым числом монет обозначить $F$, второй — $S$, а номер оставшейся группы — $T$, то можно записать (см. таблицу в решении задания 2):

```
если K(1)=K(2)
| то
|  F:=1;  S:=2; T:=3
| иначе
|  F:=2;  S:=3; T:=1
все
```

— где $K(1)$ и $K(2)$ — число монет соответственно в первой и второй группах (см. задание 1).

Алгоритм определения искомых номеров среди трех имеющихся групп может быть оформлен как алгоритм с двумя аргументами и тремя результатами (здесь и далее будем использовать термины школьного алгоритмического языка [1, 2]; синтаксис этого языка используем и для записи алгоритмов):

```
алг Определение_номеров_групп_монет
    (арг цел k1, k2, рез цел f, s, t)
    | k1, k2 — число монет соответственно
    | в первой и второй группах
    | f, s, t — искомые номера групп
нач
| если k1=k2
|  | то
|  | f:=1;  s:=2; t:=3
|  | иначе
|  | f:=2;  s:=3; t:=1
| все
кон
```

**Задание 4.** Составить алгоритм нахождения номера фальшивой монеты в группе из трех монет. Фальшивая монета более легкая, чем все остальные, имеющие одинаковый вес. (В результате выполнения алгоритма должен быть определен номер фальшивой монеты.)

Обсуждая это и последующие задания с учащимися, целесообразно ввести понятие исполнителя разрабатываемого алгоритма[1]. Назовем его "Определитель фальшивых монет". Примем, что в системе команд[2] такого исполнителя имеется команда Сравнить на весах две любые монеты, а также что имеется функция для определения веса монеты G(НомерМонеты). С учетом этого алгоритм действий исполнителя по нахождению фальшивой монеты в группе из трех монет может быть представлен в виде:

```
алг Нахождение_номера_фальшивой_монеты_3
нач цел Fal
  | Fal — номер искомой фальшивой монеты
  | Сравнить на весах две любые монеты
  | Назовем их № 1 и № 2, а оставшуюся — № 3
  если G(1)=G(2) | весы в равновесии
    | то | фальшивая монета № 3
    |    Fal :=3
    | иначе | одна из чашек весов перетянула
    |    | фальшивая монета — та из монет № 1
    |    | или № 2, которая легче
    | если G(1)<G(2)
    |   | то
    |   |    Fal:=1
    |   | иначе
    |   |    Fal:=2
    | все
  все
  вывод нс, "В этой группе фальшивая
  |          монета №", Fal
кон
```

**Задание 5.** Составить алгоритм нахождения фальшивой монеты в группе из 4—9 монет. Фальшивая монета более легкая, чем все остальные, имеющие одинаковый вес.

Здесь также целесообразно использовать исполнителя "Определитель фальшивых монет". Будем считать, что в системе его команд имеются следующие:

— Разделить все монеты на три примерно равные группы (арг цел n) (команда с аргументом $n$ — количеством монет в группе);

— Сравнить на весах вес двух групп с одинаковым числом монет (арг цел f, s) (команда с аргументами $f$ и $s$ — номерами двух групп с одинаковым количеством монет).

Кроме того, в качестве вспомогательных будем использовать:

1) функцию G(НомерГруппы) определения веса группы монет;
2) функцию K(НомерГруппы) определения числа монет в группе;
3) алгоритм Определение_номеров_групп_монет, возвращающий номера двух любых групп с одинаковым числом монет, а также номер оставшейся группы (см. задание 3);
4) алгоритм Нахождение_номера_фальшивой монеты_3 (см. задание 4), описав его с аргументом, характеризующим номер исследуемой группы: Нахождение_номера_фальшивой монеты_3 (арг цел НомерГруппы).

***Решение для задания 5***

```
алг Нахождение_номера_фальшивой_монеты_4_9 (арг цел n) | 4≤n≤9
нач цел f, s, t
  | Разделить все монеты на три примерно равные группы(n)
  | Количество монет по группам (см. задание 1):
  K(1):=n div 3; K(2):=(n — K(1)) div 2; K(3):= n — K(1)— K(2)
  | Определяем номера двух любых групп с одинаковым числом монет,а также номер оставшейся группы.
  | Определение номеров групп монет (K(1), K(2), f, s, t)
  | Сравнить на весах вес двух групп с одинаковым числом монет — группы F и S
  если G(f)= G(s)  | весы в равновесии
    | то | Фальшивая монета находится в группе t
    |    | Исполнитель определяет ее
    | если K(t)=1 | Если в группе t одна монета
    |   | то | то она — фальшивая
    |   |   вывод нс, "Фальшивая монета — в группе", t
    |   | иначе | Исполнитель ищет фальшивую монету в группе t
    |   |    | по алгоритму Нахождение_номера_фальшивой монеты_3
    |   |   Нахождение_номера_фальшивой_монеты_3(t)
    | все
```

[1] По мнению авторов книги [3], понятие исполнителя является ключевым в курсе программирования. Вопросы, связанные с понятием исполнителей алгоритмов и системы команд исполнителя, отражены в Обязательном минимуме содержания образования по информатике [4]. К сожалению, имеющиеся учебники, как правило, не учитывают это обстоятельство. Исключение составляет учебник [1]. Важное место роли исполнителя алгоритма при изучении основ программирования отводится и в [5].

[2] Система команд исполнителя — совокупность действий, которые может выполнять этот исполнитель.

```
    иначе | одна из чашек весов перетянула
    | Фальшивая монета находится в той из групп f или s, которая легче
    если G(f)< G(s)
      то | Фальшивая монета находится в группе f
      вывод нс, "Фальшивая монета — в группе", f
        | Исполнитель ищет ее по алгоритму Нахождение_номера_фальшивой_монеты_3
        Нахождение_номера_фальшивой_монеты_3(f)
        | Случай, когда в группе одна монета, не рассматриваем
      иначе | Фальшивая монета находится в группе s
      вывод нс, "Фальшивая монета — в группе", s
      Нахождение_номера_фальшивой_монеты_3(s)
    все
  все
кон
```

***Примечание.*** Не вызывает сомнения тот факт, что алгоритм Нахождение_номера_фальшивой_монеты_3 работает и в случае, если в группе $t$ имеются 2 монеты.

**Задание 6.** Составить алгоритм нахождения фальшивой монеты среди $n$ монет. Остальные условия аналогичны заданию 5.

***Решение***

В данном случае алгоритм решения опишем как алгоритм с аргументом $n$ — числом монет в исследуемой группе.

```
алг Нахождение_номера_фальшивой_монеты(арг цел n)
нач цел f, s, t
  | Разделить все монеты на три примерно равные группы...  см. алгоритм в п. 5
  | Определяем номера двух любых групп с одинаковым числом монет, а также номер оставшейся группы.
  | См. алгоритм в п. 5
  | Сравнить на весах вес двух групп с одинаковым числом монет (группы f и s)
  | Кладем на одну чашку весов монеты группы f, на другую — группы s
  если G(f)= G(s)
    то | Фальшивая монета находится в группе t
       | Находим ее
    если K(t)=1
      то
         вывод нс, "Фальшивая монета — в группе", t
      иначе | Определяем фальшивую монету в группе t, рекурсивно используя данный алгоритм
         Нахождение_номера_фальшивой_монеты(K(t))
    все
    иначе
       | Фальшивая монета находится в той из групп f или s, которая легче
    если G(f)< G(s)
      то
         Нахождение_номера_фальшивой_монеты(K(f))
      иначе
         Нахождение_номера_фальшивой_монеты(K(s))
    все
  все
кон
```

**Задание 7.** Даны $n$ монет, одна из них фальшивая, более легкая, чем все остальные, имеющие одинаковый вес. За какое наименьшее число взвешиваний всегда можно найти фальшивую монету?

***Решение***

Пусть $k$ — натуральное число, удовлетворяющее неравенствам $3^k \geq n$, $3^{k-1} < n$. Покажем, что это число $k$ удовлетворяет условию задания.

Прежде всего покажем, что при помощи $k$ взвешиваний всегда можно определить фальшивую монету. Разделим наши монеты на три группы так, чтобы в двух равных группах было по $3^{k-1}$ (или меньше) монет, а число монет в третьей группе было бы не больше $3^{k-1}$ (это возможно, ибо $n \leq 3^k$). Положив на чашки весов две группы из равного числа монет, мы определим, в какой из трех групп содержится фальшивая монета (см. алгоритм Нахождение_номера_фальшивой_монеты задания 6). Таким образом, после первого взвешивания мы выделим группу из $3^{k-1}$ монет,

среди которых содержится фальшивая (если окажется, что фальшивая монета содержится в группе, содержащей меньше $3^{k-1}$ монет, то мы можем дополнить эту группу монет до $3^{k-1}$). При каждом последующем взвешивании будем аналогично делить монеты на три группы и определять, в какой из них находится искомая монета. Следовательно, после $k$ взвешиваний мы придем к группе из одной монеты, т.е. выделим фальшивую монету.

Теперь осталось показать, что $k$ есть *минимальное* число взвешиваний, с помощью которых всегда можно выделить фальшивую монету, т.е. что при любых способах взвешивания результаты взвешиваний могут сложиться таким неблагоприятным для нас образом, что после $k-1$ взвешиваний фальшивая монета не будет выделена.

При каждом взвешивании монеты распадаются на три группы: монеты, попавшие на одну чашку, попавшие на другую чашку и не попавшие ни на одну из чашек. Если на чашки весов было положено одинаковое число монет и весы уравновесились, то фальшивая монета заведомо находится в группе монет, не попавших при взвешивании ни на одну из чашек. Если одна из чашек перетянет (при равном числе монет на чашках), то фальшивая монета заведомо находится на второй чашке. Наконец, если на чашки весов было положено разное число монет, то в случае, когда перетянула чашка, где монет больше, фальшивая монета может оказаться в любой из трех групп и такое взвешивание вообще не даст нам никаких сведений о местонахождении фальшивой монеты. Пусть теперь при произвольно производимых взвешиваниях результат взвешивания каждый раз оказывается наиболее неблагоприятным, т.е. фальшивая монета каждый раз оказывается в той из трех групп, которая содержит наибольшее число монет. Тогда при каждом взвешивании число монет группы, содержащей фальшивую монету, убывает не более чем в 3 раза (ибо при делении некоторого числа монет на три группы всегда по крайней мере одна из трех групп содержит не менее чем треть от общего числа монет — см. решение заданий 1 и 2). Поэтому после $k-1$ взвешиваний число монет группы, содержащей фальшивую монету, останется не меньшим, чем $n/3^{k-1}$, и так как $n > 3^{k-1}$, то после $k-1$ взвешиваний фальшивая монета не будет выделена.

***Примечание.*** Можно коротко записать ответ задачи в такой форме: минимальное число взвешиваний, необходимое для выделения фальшивой монеты из группы в $n$ монет, есть $\mathrm{INT}(\log_3(n-0{,}5))+1$, где INT — функция, возвращающая целую часть числа.

## ЗАДАЧА 2

**Среди 12 монет имеется одна фальшивая. Известно, что фальшивая монета отличается по весу от настоящих, но не известно, легче она настоящих или тяжелее. Настоящие монеты все одного веса. С помощью трех взвешиваний на чашечных весах без гирь выделить фальшивую монету и одновременно установить, легче она или тяжелее остальных.**

***Решение***

Разделим наши монеты на три группы по четыре монеты в каждой. Монеты первой группы обозначим $m_1$, $m_2$, $m_3$, $m_4$, второй — $m_5$, $m_6$, $m_7$, $m_8$, третьей — $m_9$, $m_{10}$, $m_{11}$, $m_{12}$. При первом взвешивании поместим на каждую чашку весов по группе из четырех монет, например, первую и вторую. Возможны два варианта:

1. Чашки весов уравновесились.
2. Одна из чашек перевесила.

Рассмотрим оба варианта по отдельности.

**1.** *При первом взвешивании чашки весов уравновесились.* Следовательно, фальшивая монета находится в оставшейся, третьей, группе, а 8 монет на весах — настоящие. Положим при втором взвешивании на одну чашку три монеты из третьей группы (например, монеты $m_9$, $m_{10}$, $m_{11}$), а на другую — три монеты из числа восьми заведомо настоящих (например, монеты $m_1$, $m_2$, $m_3$). Возможны два случая:

1.1. Чашки весов уравновесились. Тогда монета $m_{12}$ — фальшивая. Сравнивая третьим взвешиванием ее с настоящей ($m_1$), мы находим, легче она или тяжелее, чем настоящая.

1.2. Одна из чашек перетянула. В этом случае фальшивой является одна из монет $m_9$, $m_{10}$ или $m_{11}$. При этом если перетянула чашка с настоящими монетами, то фальшивая монета легче настоящих; одним взвешиванием мы без труда выделяем более легкую из трех монет: 1, 2 и 3 (см. решение задания 4). Если же перетянула чашка с монетами $m_9$, $m_{10}$, $m_{11}$, то фальшивая монета тяжелее настоящих; но и в этом случае ее также легко определить одним взвешиванием.

**2.** При первом взвешивании одна из чашек весов перетянула.

Здесь возможны 2 случая:

а) перетянула чашка весов с монетами $m_1$, $m_2$, $m_3$, $m_4$;

б) перетянула чашка весов с монетами $m_5$, $m_6$, $m_7$, $m_8$.

В обоих случаях все монеты в оставшейся группе — настоящие. А если фальшивая монета — одна из монет $m_1$, $m_2$, $m_3$, $m_4$, то она тяжелее настоящих, если одна из монет $m_5$, $m_6$, $m_7$, $m_8$ — то она легче настоящих.

Рассмотрим эти два случая по отдельности.

**2а.** Перетянула чашка весов с монетами $m_1$, $m_2$, $m_3$, $m_4$. При втором взвешивании поместим на одну чашку монеты $m_1$, $m_2$ и $m_5$, а на другую — монеты $m_3$, $m_4$ и $m_6$. Возможны опять-таки различные случаи:

2*а*.1. Чашки уравновесились. Тогда фальшивая одна из монет $m_7$ или $m_8$ (и при этом она легче настоящих). При третьем взвешивании поместим на одну чашку весов монету $m_7$, а на вторую — монету $m_8$; та из этих монет, которая окажется легче другой, и будет фальшивой.

2*а*.2. Перетянула чашка с монетами $m_1$, $m_2$ и $m_5$. В этом случае монеты $m_3$, $m_4$ и $m_5$ — настоящие; в самом деле, если бы одна из монет $m_3$, $m_4$ была бы тяжелее остальных или монета $m_5$ была бы легче остальных, то при втором взвешивании чашка, на которой лежат монеты $m_3$, $m_4$ и $m_6$, должна была бы перетянуть, чего на самом деле не случилось. Итак, фальшивой является одна из монет $m_1$, $m_2$ (в этом случае фальшивая монета тяжелее настоящих) или $m_6$ (в этом случае фальшивая монета легче настоящих).

Положим при третьем взвешивании на одну чашку монету $m_1$, а на другую — монету $m_2$. Если чашки уравновесились, то фальшивая монета $m_6$, а если одна из чашек перетянула, то на перетянувшей чашке лежит фальшивая монета.

2*а*.3. Перетянула чашка с монетами $m_3$, $m_4$ и $m_6$. Рассуждая аналогично предыдущему, мы заключаем, что монеты $m_1$, $m_2$ и $m_6$ — настоящие и что либо одна из монет $m_3$, $m_4$ фальшивая и тяжелее настоящих, либо монета $m_5$ фальшивая и легче настоящих.

При третьем взвешивании положим на одну чашку монету $m_3$, а на другую — монету $m_4$. Если весы уравновесились, то фальшивая монета $m_5$. Если же одна из чашек перетянула, то на ней и находится фальшивая монета.

**2б.** Перетянула чашка весов с монетами $m_5$, $m_6$, $m_7$, $m_8$.

В этом случае поступаем аналогично с той разницей, что роль монеты $m_1$ будет играть монета $m_5$ (и наоборот), монеты $m_2$ — монета $m_6$ (и наоборот), монеты $m_3$ — монета $m_7$ (и наоборот), монеты $m_4$ — монета $m_8$ (и наоборот).

**Задание для учеников.** Составить сводку результатов (аналогично задаче в № 19/99).

***Решение***

**1.**

```
Если m1 + m2 + m3 + m4 = m5 + m6 + m7 + m8, то:
   если m9 + m10 + m11 = m1 + m2 + m3, то
      фальшивая монета — m12;
   если m12 > m1, то
      она тяжелее настоящих;
   если m12 < m1, то
      она легче настоящих.

Если m9, m10, m11 < m1, m2, m3, то
      фальшивая монета легче настоящих;
   если m9 > m10, то
      фальшивая монета — m10;
   если m9 = m10, то
      фальшивая монета — m11;
   если m9 < m10, то
      фальшивая монета — m9;
   если m9, m10, m11 > m1, m2, m3, то
      фальшивая монета тяжелее настоящих;
   если m9 > m10, то
      фальшивая монета — m9;
   если m9 = m10, то
      фальшивая монета — m11;
   если m9 < m10, то
      фальшивая монета — m10.
```

**2а.**

```
Если m1 + m2 + m3 + m4 > m5 + m6 + m7 + m8, то:
   если m1 + m2 + m5 = m3 + m4 + m6, то
      фальшивая одна из монет — m7 или m8
      (и при этом она легче настоящих).
Если m7 < m8, то
      фальшивая монета — m7;
   если m7 > m8, то
      фальшивая монета — m8;
   если m1 + m2 + m5 > m3 + m4 + m6, то
      фальшивой является одна из монет — m1,
      m2 (в этом случае фальшивая монета
      тяжелее настоящих) или m6 (в этом
      случае фальшивая монета легче настоящих).

Если m1 = m2, то
      фальшивая монета — m6;
   если m1 < m2, то
      фальшивая монета — m2;
   если m1 > m2, то
      фальшивая монета — m1;
   если m1 + m2 + m5 < m3 + m4 + m6, то
      фальшивой является одна из монет — m3,
      m4 (и в этом случае она тяжелее
      настоящих) либо монета m5 (в этом
      случае она легче настоящих).

Если m3 = m4, то
      фальшивая монета — m5;
   если m3< m4, то
      фальшивая монета — m4;
   если m3 > m4, то
      фальшивая монета — m3.
```

**2б.**

```
Если m1 + m2 + m3 + m4 < m5 + m6 + m7 + m8, то:
   если m5 + m6 + m1 = m7 + m8 + m2, то
      фальшивая одна из монет — m3 или m4
      (и при этом она легче настоящих).
Если m3 < m4, то
      фальшивая монета — m3;
   если m3 > m4, то
      фальшивая монета — m4;
   если m5 + m6 + m1 > m7 + m8 + m2, то
      фальшивой является одна из монет — m5,
      m6 (в этом случае фальшивая монета
      тяжелее настоящих) или m2 (в этом
      случае фальшивая монета легче настоящих).
```

```
Если m5 = m6, то
     фальшивая монета — m2;
   если m5 < m6, то
     фальшивая монета — m6;
   если m5 > m6, то
     фальшивая монета — m5;
   если m5 + m6 + m1  < m7 + m8 + m2, то
     фальшивой является одна из монет — m7,
     m8 (и в этом случае она тяжелее
     настоящих) либо монета m1 (в этом
     случае она легче настоящих).

Если m7 = m8, то
     фальшивая монета — m1;
   если m7< m8, то
     фальшивая монета — m8;
   если m7 > m8, то
     фальшивая монета — m7.
```

## ЗАДАЧА 3

**Имеются 20 металлических кубиков, одинаковых по размеру и внешнему виду. Некоторые из них алюминиевые, остальные — дюралевые (более тяжелые). При помощи не более чем 11 взвешиваний на чашечных весах без гирь определить число дюралевых кубиков.**

***Примечание.*** В задаче предполагается, что все кубики могут быть алюминиевыми, но дюралевыми все они быть не могут (иначе если бы все кубики оказались одного веса, то без этого условия мы никак не смогли бы определить, алюминиевые они или дюралевые).

***Решение***

Положим на чашки весов по одному кубику (первое взвешивание). При этом могут иметь место два различных случая.

**1.** *При первом взвешивании одна из чашек весов перетянула.*

В таком случае из двух взвешиваемых кубиков один обязательно является алюминиевым, а второй — дюралевым. Далее кладем эти два кубика на одну чашку весов, а на вторую — последовательно по паре оставшихся кубиков (разбиение 18 оставшихся кубиков на 9 пар проводим произвольно). Если какая-нибудь пара кубиков перетягивает нашу пару, то это значит, что оба кубика во второй паре дюралевые; если перетягивает первая пара, то оба кубика второй пары — алюминиевые; если обе пары имеют одинаковый вес, то вторая пара также содержит один алюминиевый и один дюралевый кубики. Таким образом, в случае 1 мы можем определить число дюралевых кубиков при помощи 10 взвешиваний (первое взвешивание и еще 9 взвешиваний пар кубиков).

**2.** *При первом взвешивании чашки весов остались в равновесии.*

В таком случае кубики первой пары или оба алюминиевые, или оба дюралевые. Далее кладем эти два кубика на одну чашку весов, а на вторую — последовательно по паре из числа 18 оставшихся. Пусть первые $k$ из этих пар оказались одного веса с первоначальной, а $(k+1)$-я пара — другого веса. (Если $k = 9$, то все кубики имеют одинаковый вес и, следовательно, дюралевых кубиков нет вовсе; случай $k = 0$ ничем не отличается от общего случая.) Предположим для определенности, что $(k+1)$-я пара оказалась более тяжелой, чем первоначальная (рассуждение мало изменилось бы, если $(k+1)$-я пара оказалась бы более легкой). В таком случае первые два кубика, а следовательно, и кубики первых $k$ пар, которые оказались с ними одинакового веса, — алюминиевые. Итак, мы провели пока $1 + (k+1) = k+2$ взвешиваний и выделили при этом $k+1$ пар у алюминиевых кубиков. Теперь положим на чашки весов по кубику из последней взвешенной пары ($(k+3)$-е взвешивание). Если оба кубика окажутся одинакового веса, то они оба должны быть дюралевыми; в противном случае — один из них алюминиевый, а второй — дюралевый. В обоих случаях мы можем после $k+3$ взвешиваний указать на пару из двух кубиков, один из них алюминиевый, а второй — дюралевый. С помощью этой пары мы $8 - k$ взвешиваниями определим число дюралевых кубиков среди оставшихся $20 - 2(k+2) = 16 - 2k$ кубиков так, как мы поступали в случае 1. Общее число взвешиваний в случае 2 будет равно $k+3+(8-k) = 11$.

### Литература


1. *Кушниренко А.Г., Лебедев Г.В., Сворень Р.А.* Основы информатики и вычислительной техники. М.: Просвещение, 1990.
2. *Эпиктетов М.Г.* Почему школьный алгоритмический? Информатика, 1995, № 33.
3. *Кушниренко А.Г., Лебедев Г.В.* Программирование для математиков. М.: Наука, 1988.
4. Оценка качества подготовки выпускников основной школы по информатике (сборник материалов). Информатика, 1999, № 38.
5. *Сенокосов А.И.* Информатика 7—9 (учебник). Информатика, № 24, 28, 34/99.

# Информационные технологии в образовании

М.Е. Белиловская

**С 9 по 12 ноября в Москве проходила IX международная конференция-выставка "Информационные технологии в образовании" ("ИТО-99"). Работала она в Московском городском физико-математическом лицее № 1511 при МИФИ (директор — Евгений Федорович Семенов). Организатором этой крупнейшей в России конференции по информатизации образования выступило научно-производственное предприятие "БИТ про" при поддержке Министерства образования России, Института ЮНЕСКО по информационным технологиям в образовании, Института проблем информатики Российской академии наук и Московского комитета образования.**

В Москву съехались преподаватели информатики школ и вузов, разработчики программного обеспечения и другие специалисты более чем из 90 населенных пунктов России, Украины, Беларуси, Казахстана, Киргизии, Польши, США. Кто-то не смог приехать — им пришлось ограничиться публикацией доклада в сборнике "ИТО-99". Активное участие в конференции приняли жители Москвы (35 процентов от общего числа участников), Московской области (5 процентов), Санкт-Петербурга, Екатеринбурга и Волгограда.

Тем для обсуждения набралось множество. Больше всего докладов собрала вторая секция — "Интеграция информационных технологий в образование". Третья — "Технологии открытого образования" — это самое модное направление на сегодняшний день, по словам председателя программного комитета "ИТО-99" Сергея Александровича Христочевского, заведующего лабораторией "Проблемы информатизации образования" Института проблем информатики РАН. Хотя об эффективности этих технологий еще спорить и спорить. Пятая, самая маленькая, секция — "Информационные технологии в образовании для людей со специальными потребностями" — появилась впервые. "На самом деле, — комментирует Сергей Христочевский, — ей нужно придавать наибольшее значение, и не только на конференции, но и в повседневной жизни. Информационные технологии могут вернуть этих людей к полноценной активной жизни, тогда как у остальных всего лишь увеличивают производительность труда". И, конечно, большой интерес вызвали дискуссии на первой секции — "Информатика: стандарты и содержание".

## В XXI век — с "Агатами"?

Обеспечение техникой — общая проблема. Научный директор лицея информационных технологий № 1533 и главный редактор журнала "Компьютер в школе" Александр Гиглавый заметил, что "единственным постоянным свойством компьютера является его изменчивость". Школу это постоянное движение вперед, как правило, благополучно минует. Мы не так далеко ушли от логарифмической линейки и светлой памяти КУВТ (комплект учебной вычислительной техники). Многие учителя, с которыми я разговаривала, задавали риторический вопрос: с чем наши дети войдут в XXI век? С "Агатами"? И почти все с горечью вспоминали летние обещания премьер-министра и министра образования: каждой школе — современный компьютер с выходом в Интернет.

У многих детей хорошие компьютеры появляются дома. Приходят они потом в класс на информатику и спрашивают презрительно: "А зачем мы будем работать за *этим*? Зачем нам это надо?" А что может сказать учитель? "Объясняем, — говорят, — что вроде так надо, а сами понимаем, что это — даже не вчерашний день, а позавчерашний..."

С точки зрения министерства хорошо учить можно на любой технике, но здесь встает вопрос: чему учить? Программированию — да. Но сегодня большинство учителей предпочитает обучать не ему, а технологиям, работе в офисных приложениях. Светлана Ивановна Золотова, начальник отдела компьютерного обучения Фонда новых технологий в образовании "Байтик", многолетний организатор летней конференции по информатизации образования в Троицке, об этом жалеет. "Скоро некому будет участвовать в российской олимпиаде по информатике, — говорит она. — Осталось лишь несколько сильных лицеев и других центров, которые дают достаточно знаний, чтобы ребята становились программистами... Спор "программирование или информационные технологии" закончился. Хороший ПК покупают, чтобы поставить на него графический редактор".

И — Интернет. Конечно, он есть еще далеко не у всех. Это только в Иркутске, говорят, провайдеры соревнуются — кто больше школ к Интернету подключит. Остальные получают гранты или платят свои кровные.

Но вот проблема доступа в Интернет решена. Что делать дальше? Евгения Семеновна Полат, заведующая лабораторией дистанционного образования РАО, отметила два варианта образовательной деятельности в Интернете, которые активно пропагандируются сейчас в странах, более информационно развитых. Первый вариант — интеграция: Интернет и некоторые формы дистанционного образования интегрируются в очное образование. Это же мечта ученика — не приходить каждый день в класс! Самостоятельная познавательная деятельность сочетается с работой с учителем в школе. Второй вариант — дистанционное обучение по следующей схеме: обычные курсы дистанционного образования, виртуальные классы, чаты, где группы учеников общаются не только с учителем, но и друг с другом, плюс электронный учебник.

## Нужно ли учителю дистанционное образование?

Про дистанционное образование с использованием компьютерных технологий спорили, пожалуй, больше всего. Дистанционное обучение — это не только технический вопрос, но и педагогический. Какова в этой ситуации роль педагога? Можно ли научить чему-либо, когда ученик и учитель находятся по разные стороны экрана?

"Зачем учителю дистанционное обучение?" — спрашивали из зала во время заседания секции "Открытое образование". "Мне его плюсы кажутся совершенно очевидными, — поглаживая седую бороду, говорил директор

"Роботландии" Юрий Абрамович Первин. — Дистанционное образование многократно увеличивает активность учителя во время повышения квалификации и переподготовки. Если во время очного обучения есть возможность спрятаться за спину товарища, то при дистанционном присутствует постоянный контроль. Во-вторых, появляется потрясающая возможность привлечь квалифицированных людей различных областей. Например, чтобы создавать курс "Базы данных", я могу пригласить не человека из соседнего подъезда, а более крупного специалиста, допустим, из Перми. Третье достоинство — экономия средств. Можно не посылать учителей в районные и областные центры, а учить на местах, — документ, удостоверяющий повышение квалификации, они получают тот же. Кроме того, учитель не отрывается от учеников, они вместе с ним участвуют в процессе дистанционного образования, осваивая новые формы работы и новые учебные пособия непосредственно после их получения".

## Сделай сам: web-квесты и сайты

В рамках "ИТО-99" прошло несколько семинаров; один из самых многочисленных — "Квест-технологии как новые формы образовательной работы средней и высшей школы", подготовленный некоммерческой американской организацией "Проект Гармония" (*Project Harmony*). Этот проект стремится объединить российские школы в их сетевых инициативах, привлечь внимание к использованию информационных технологий учителями-предметниками. С этими целями "Гармония" активно пропагандирует развитие различных конкурсов, создание web-квестов и образовательных сайтов. "Результатом такой совместной деятельности учителя-предметника и информатика может быть вполне качественный образовательный продукт", — считает сотрудница проекта Елена Николаевна Ястребцева.

Термин *web-quest* появился лет пятнадцать назад. Оказалось, что так презираемые родителями игры типа "ходилка-стрелялка" могут нести в себе мощный образовательный потенциал. Ведь это же такая удобная форма: ребенок выбирает себе роль и доводит ее до конца игры, двигаясь по индивидуальной траектории, приобретая собственные знания и опыт. Квест — это как путешествие рыцарей в поисках приключений. В данном случае — познавательных.

Образовательный web-квест — это сайт в Интернете, с которым работают учащиеся, выполняя ту или иную учебную задачу. Особенность web-квестов — часть или даже вся необходимая для работы информация находится на других сайтах или различных носителях информации. В финале квеста может стоять вполне реальная цель: например, составить доклад о проблемах Байкала и направить его именно тому депутату Государственной думы, от которого может зависеть принятие закона об охране уникального озера.

Пока у нас проходят только первые попытки создания квестов. "В Интернете масса материалов, но все это создано не нами, — говорит Галина Григорьевна Мангазеева, заведующая кафедрой информатики экспериментальной школы № 47 города Иркутска, которая собирается предложить своим ученикам создать свой web-квест, — они дают возможность сделать самим то, что будет интересно другим".

В этом году "Проект Гармония" провел конкурс на создание сайта, посвященного литературным героям, — "Сайт д'Артаньяна". Жюри долго обсуждало поступившие 20 работ, и победителями стали Саша Кубарев и Надя Мельникова, восьмиклассники из гимназии № 1522, с сайтом про Пиноккио. Награду — компьютер — им вручили на конференции. Саша, державшийся с большим достоинством, заявил: "Прочтя эту книгу заново, я понял, что главной человеческой добродетелью является благодарность...".

Пиноккио, а не Буратино, обратите внимание! Ребята вдохновлялись аналогичным итальянским сайтом, разместили у себя оригинальный текст Коллоди (с языком помогала Сашина сестра, которая "болеет" итальянским и Интернетом одновременно) и русско-итальянский словарь, — удивительно, как много в наших языках похожих слов! И еще — Клуб деревянных человечков, в котором состоят все друзья Пиноккио, в том числе Буратино и российская матрешка. Ведь она — тоже деревянная и тоже — человечек.

Ресурсов для учителей в русском Интернете пока не так много — кому же создавать их, как не преподавателям-энтузиастам? Маловероятно, что в эту деятельность сейчас будут вкладываться серьезные деньги.

## На "Выступе ловких бегемотов"

Выставка собрала практически все российские фирмы, работающие в нише образования, — участие стоит недорого, а аудитория очень широкая. Были представлены мультимедийные программы для детей и взрослых, комплекты учебных программ для школ и вузов, программы для автоматизации делопроизводства, составления расписания и т.д. и т.п.

Конечно, "компьютерный" учебник не заменит "бумажного", но сейчас появляется новое качество: мультимедиатехнологии позволяют предъявлять материал в другом виде, сочетать воздействие видеоизображения, звука и текста.

Фирма "1С" подвела промежуточные результаты своей акции "1С:Репетитор" — в школу!". С 1 октября по 5 ноября учителям было бесплатно передано около 1500 коробок с программами серии "1С:Репетитор", а на выставке "ИТО-99" учителя получили еще более 500 копий. По словам представителей компании, акция направлена не на школы в целом, а на учителей-предметников: важно, чтобы программный продукт использовался по назначению, а не лежал мертвым грузом в кабинете директора.

НПП "Эрикос" презентовало свою новинку — образовательную игру "Грамотей-ЭВЕРЕСТ", предназначенную для того, чтобы научить детей правильно писать. Построена она по принципу "Тетриса": сверху падают слова, в которые нужно вставить правильные буквы. На конкурс "грамотеев" были приглашены все участники выставки. Я героически дошла до 4-го уровня, который назывался "Выступ ловких бегемотов" (всего уровней 21, каждый отвечает за пять правил из школьной программы), — на большее не хватило времени. Победил же и покорил Эверест преподаватель Брянского государственного педагогического университета Владимир Николаевич Алдушонков.

Вход на конференцию и выставку был свободным. Каждый посетитель, заполнив анкету, бесплатно получал официальный каталог и набор из последних номеров компьютерных журналов. По предварительным результатам, "ИТО-99" посетило более 1500 человек. По их анкетам был составлен рейтинг компьютерных программ, используемых в различных областях образования, а также рейтинг лучших образовательных web-сайтов. На первом месте в этом рейтинге оказался сайт газеты "Первое сентября". Спасибо всем, кто проголосовал за нас, — мы постараемся стать еще лучше.

*Подробная информация о конференции и тезисы представленных докладов опубликованы в Интернете по адресу*: **http://ITO.BITpro.ru/1999**.

Для того чтобы приобрести заинтересовавшие вас книги наложенным платежом, нужно:

- **указать в купоне ваши фамилию, имя, отчество, почтовый адрес и индекс;**
- **рядом с названием заказываемых книг, в квадрате, поставить количество экземпляров;**
- **вырезать и выслать купон по адресу:**

**121165, Москва, ул. Киевская, д. 24,**
**«Первое сентября»**
**Деньги вы заплатите только при получении книг на почте. Указанная в купоне цена не включает в себя расходы за почтовую пересылку и авиатариф.**

**Телефон для справок** (095) 249-33-86.

## КНИГА — ПОЧТОЙ
# КУПОН

**Первое сентября**
**Объединение педагогических изданий**

**почтовый индекс**

**город (поселок, деревня, район, область)**

**улица (квартал, проспект, проезд, переулок, бульвар, тупик, шоссе, линия и т.п.)**

**дом** **корп./строение/** **квартира** **в/ч** **а/я** **п/я**

**фамилия** **имя, отчество**

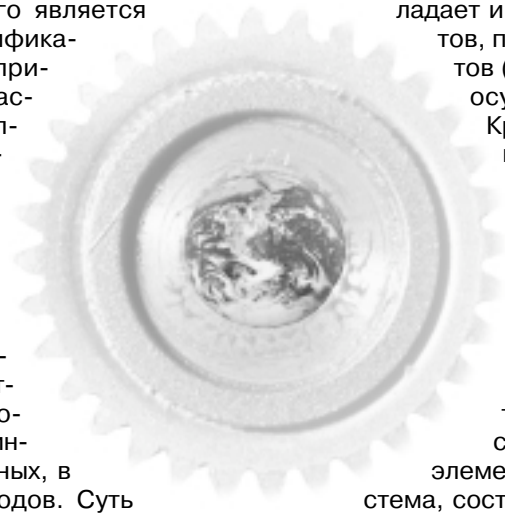
- ☐ **С.Л. Соловейчик. Пушкинские проповеди (40 руб.)**
- ☐ **С.Л. Соловейчик. Последняя книга (50 руб.)**
- ☐ **Школа сотрудничества** (Сборник статей по педагогике сотрудничества) **(35 руб.)**
- ☐ **Д.В. Растимешин. Как получить заработанные деньги** (Практическое руководство для учителей) **(12 руб.)**
- ☐ **А.Ю. Головатенко. Социализм. Теория и практика** (В 2-х частях) **(40 руб.)**
- ☐ **Я иду на урок в начальную школу. Русский язык (22 руб.)**
- ☐ **Я иду на урок в начальную школу. Тесты по русскому языку (10 руб.)**
- ☐ **Я иду на урок в начальную школу. Природоведение (22 руб.)**
- ☐ **Я иду на урок в начальную школу. Математика (22 руб.)**
- ☐ **Я иду на урок физики. 7-й класс** (В 3-х частях) **(50 руб.)**
- ☐ **Я иду на урок химии. 8–11-е классы (22 руб.)**
- ☐ **Я иду на урок химии. Летопись важнейших открытий. XVII—XIX вв. (22 руб.)**
- ☐ **Я иду на урок математики. 5-й класс (22 руб.)**
- ☐ **Я иду на урок математики. Тесты. 5-й класс (8 руб.)**
- ☐ **Я иду на урок биологии. Зоология. Беспозвоночные (22 руб.)**
- ☐ **Я иду на урок истории. Древнейшая и древняя история (22 руб.)**
- ☐ **Л.А. Кацва. Россия в 1917–1918 годах. 10–11-е классы (15 руб.)**
- ☐ **Т.С. Троицкая, О.Е. Петухова. Учебно-методический комплект по начальному литературному образованию для учителей и родителей. 1–2-е классы** (В 3-х книгах) **(60 руб.)**

# Системотехника как научное направление

**Окончание. Начало на с. 1**

вания, конструирования и поведения сложных информационных систем", основу которых составляют компьютеры [1]. (Причем тут следует сразу указать, что дать определение системотехники, устраивающее всех специалистов, использующих данный термин, сегодня не представляется возможным.)

В целом системотехника представляет собой сформировавшееся научное направление. Определенным подтверждением этого является введение еще в 1977 году квалификации "инженер-системотехник", присваиваемой специалистам в области автоматизированных систем управления и электронных вычислительных машин.



Считается, что основным методом системотехники является так называемый *системный подход* [2, 3, 4]. Однако четкого определения понятий *сложная система* и *большая система* не существует, и системный подход является, вообще говоря, не чем-то самостоятельным, а лишь некоторым синтезом здравого смысла и известных, в том числе математических, методов. Суть здесь — применительно к большим системам — хорошо поясняется следующим простым примером. Для решения систем линейных алгебраических уравнений существует метод определителей Крамера, позволяющий (теоретически) решить систему с любым числом неизвестных и уравнений. Однако если система состоит из очень большого количества уравнений с таким же количеством неизвестных, то, употребляя метод Крамера, решение задачи нельзя довести до конца ни на одном компьютере, и приходится пользоваться другими (тоже известными) методами.

Для системного подхода очень важно понимание того, что система представляет собой не просто объединение своих частей: она имеет новое (системное) свойство, не сводящееся к свойствам отдельных элементов. Например, система "компьютер" обладает свойством, которым ни одна из ее частей не обладает, — обрабатывать информацию и выдавать результаты. Но таким свойством не обладает и простая "сумма" тех же элементов, поскольку подбор и связь элементов (для технических систем) должны осуществляться целенаправленно. Кроме того, надо понимать, что элемент системы, то есть ее часть, выполняющая определенные функции, не подлежит расчленению только при данной степени детализации системы. При другой степени детализации элемент системы сам может рассматриваться как система. К примеру, тот же компьютер, включенный в сеть, в одном случае может рассматриваться как элемент этой сети, а в другом — как система, состоящая из ряда элементов.

### Литература


1. *Скурихин В.И.* Системотехника // Энциклопедия кибернетики. Киев: Гл. редакция Украинской советской энциклопедии, 1975. Т. 2.
2. *Губанов В.А., Захаров В.В., Коваленко А.Н.* Введение в системный анализ. Л.: ЛГУ, 1988.
3. *Николаев В.И., Брук В.М.* Системотехника: методы и приложения. Л.: Машиностроение, 1985.
4. *Перегудов Ф.И., Тарасенко Ф.П.* Введение в системный анализ. М.: Высшая школа, 1989.



**Гл. редактор**
С.Л. Островский
**Зам. гл. редактора**
Е.Б. Докшицкая
**Редакция:**
И.Н. Фалина,
Н.Л. Беленькая,
Н.П. Медведева
**Дизайн и компьютерная верстка:**
Н.И. Пронская
**Корректоры:**
Е.Л. Володина,
С.М. Подберезина




выходит четыре раза в месяц
При перепечатке ссылка на ИНФОРМАТИКУ обязательна, рукописи не возвращаются

121165, Киевская, 24
тел. 249 4896
Отдел рекламы
тел. 249 9870

**Учредитель: ООО "Чистые пруды"**
**Регистрационный номер 012868**
Отпечатано в типографии ОАО ПО "Пресса-1".
125865, ГСП, Москва, ул. "Правды", 24.
Тираж 5500 экз.
Заказ №

**ИНДЕКС ПОДПИСКИ**
**для индивидуальных подписчиков 32291**
**комплекта приложений 32744**

**Тел. (095)249 3138, 249 3386. Факс (095)249 3184**

**Internet: inf@1september.ru**
**Fidonet: 2:5020/69.32**
**WWW: http://www.1september.ru**