Обзор читательских откликов на статьи, посвященные игре «Бой в памяти», экологической войне на планете Аква-Тор и охоте с компьютером на бобра-работягу

А. К. ДЬЮДНИ

Когда в прошлом году появилась статья, посвященная игре «Бой в памяти»*, я никак не ожидал, что поднятые в ней вопросы привлекут к себе столь широкое внимание. Приведенные в этой статье описания программ, перемещающихся по памяти компьютера и пытающихся уничтожить друг друга, вызвали широкий резонанс. По свидетельству многих читателей, письма которых я рассмотрю ниже, действительность изобилует примерами, когда в компьютерах при самых разнообразных обстоятельствах поселяются «червяки», «вирусы» и другие программные «вредители». В результате в мире компьютера могут возникать настолько ужасные ситуации, что я даже не решаюсь о них писать.

Выпущенная недавно во Франции книжка шпионско-детективного жанра Т. Бретона и Д. Бенеша "Softwar: La Guerre Douce" («Война программ») дает нам фантастический пример подобного рода. Сюжет книги построен на паутине заговора, которую плетут американские поставщики супер-компьютера, собираясь продать его другой стране. Американские власти после некоторого притворного колебания соглашаются на эту сделку. Компьютер был тайно запрограммирован специальными «программными бомбами». Закупленная для расчетов прогноза погоды машина, а точнее ее программное обеспечение, скрывает в себе некий скрытый механизм — как только Национальная служба погоды США сообщит, что в г. Сан-Томасе на Виргинских островах наблюдается определенная температура, машина должна немедленно приняться за искажение и уничтожение всех доступных ей программ в сети ЭВМ страны-импортера. Предположив реальность подобных сценариев, хочется отметить: «Если и говорить о войнах, то пусть они лучше будут программными». С другой стороны, при мысли о том, что борьба программ в компьютерах, которыми располагают военные ведомства, может привести к трагической случайности, тоже становится как-то не по себе.

Прежде чем перейти к описанию разнообразных случаев, связанных с

КОМАНДА	ОБОЗНАЧЕНИЕ	КОД	ОПЕРАНДЫ		ПОЯСНЕНИЯ
Переместить	MOV	1	A	B	Переместить содержимое ячейки A в ячейку B
Сложить	ADD	2	A	B	Сложить содержимое ячеек A и B
Вычесть	SUB	3	A	B	Вычесть содержимое ячейки A из содержимого ячейки B
Перейти	JMP	4	A		Передать управление на ячейку A
Перейти, если ноль	JMZ	5	A	B	Передать управление на ячейку A, если в ячейке B находится нуль
Перейти, если больше	JMG	6	A	B	Передать управление на ячейку A, если в ячейке B находится величина бóльшая нуля
Уменьшить и перейти, если 0	DJZ	7	A	B	Вычесть 1 из содержимого ячейки B и передать управление по адресу A, если содержимое B становится нулем
Сравнить	CMP	8	A	B	Сравнить величины в ячейках A и B, если они не равны, пропустить следующую команду
Разветвить	SPL	9	A		Разветвить выполнение команд: выполнить следующую команду и команду в ячейке A
Определить	DAT	0		B	Невыполняемая команда; B — значение элемента данных

Система команд языка программирования для игры «Бой в памяти»

* «В мире науки», 1984, № 7. — *Прим. ред.*

действиями всевозможных враждебных программ, напомним вкратце правила игры «Бой в памяти». Двое играющих пишут каждый свою программу на языке низкого уровня Редкод. Эти программы помещаются в обширную замкнутую область памяти, или просто Память. В действительности Память представляет собой массив из нескольких тысяч ячеек с последовательными адресами, причем за адресом последней ячейки вновь следует адрес первой. Каждая команда боевой программы занимает одну ячейку Памяти. Программа монитор под названием Марс (Mars — сокращение от слов A Memory Array Redcode Simulator) управляет боевыми программами, выполняя по одной очередной команде из каждой программы по простой системе разделения времени. Программы атакуют друг друга и одновременно пытаются избежать ущерба или восстановить себя, если ущерб все же был нанесен. Простейший вид атаки может представлять собой команда MOV. Например, команда

MOV # 0 1000

приведет к тому, что в ячейку, отстоящую на 1000 адресов от данной команды, будет помещено число, равное нулю. Информация, содержавшаяся ранее в ячейке, будет уничтожена. В частности, если нуль попадет в начальную ячейку, содержащую команду из программы противника, то эта команда исчезнет, программа не сможет больше выполняться и это приведет к поражению противника. Поскольку ни один компьютер, будь то персональный или большой универсальный, не имеет таких встроенных средств, как язык Редкод и массив памяти для сражающихся программ, эти средства должны моделироваться.

```
1   IF PEEK (104) = 134 GOTO 10
2   POKE 104, 134: POKE 134 * 256,0
3   PRINT CHR$(4) "RUN APPLE WORM"
10  HOME : POKE - 16302,0: POKE - 16304,0: POKE 1023,160
20  FOR I = 0 TO 94: READ D: POKE 1024 + I, D: NEXT I
30  POKE - 16368,0
40  IF PEEK ( - 16384)<128 GOTO 40
50  CALL 1024
100 DATA 160,225,200,185,255,3,153,127,4,192,95,208,245,
    160,18,190,76,4,24,189,128,4,105,128,157,128,4,189,129,
    4,105,0,157,129,4,192,13,208,18,238,23,4,173,23,4
200 DATA 141,151,4,206,31,4,173,31,4,141,159,4,136,208,211,
    173,167,4,72,173,176,4,141,167,4,104,141,176,4,76,128,
    4,7,20,25,28,33,46,55,61,65,68,72,75,4,16,40,43,49,52
```

Программа «червяк», обитающая в компьютерах

Вдохновленный статьей Л. Пенроуза о самовоспроизводящихся механизмах, появившейся в июньском номере журнала "Scientific American" за 1959 г., Ф. Стал из Честерфилда, шт. Миссури, создал миниатюрную линейную вселенную, населенную незамысловатыми существами, которые жили, передвигались по своему миру, подчиняясь (если так можно выразиться) воле судеб. Стал пишет:

«Как и в игре «Бой в памяти», я выделил замкнутый линейный сегмент памяти, где с помощью несколько модифицированного машинного языка моделировались существа. Тогда в моем распоряжении была машина IBM 650 с памятью на магнитном барабане. Существо было запрограммировано, чтобы ползать в пределах своего мира, питаться ненулевыми словами и, наевшись досыта, производить на свет свою копию. Аналогично игре «Бой в памяти» у меня была программа монитор, которая хранила сведения о живых существах и распределяла между ними машинное время. Эту программу я называл шутливо «Левой рукой бога».

В письме Стал рассматривает способность программы к воспроизводству и описывает интересный механизм мутаций. В процессе создания своей копии программа может внести в нее несколько случайных изменений. Однако Стал пишет: «Я исключил это свойство после одного из прогонов системы, когда стерильный мутант сожрал единственное способное к воспроизводству существо. Было очевидно, что для достижения сколько-нибудь интересных результатов потребовался бы чрезвычайно большой объем памяти и очень много машинного времени».

Можно рассказать также историю, связанную с игрой «Животное», в которой программа, последовательно задавая играющему с ней человеку двадцать наводящих вопросов, пытается определить, какое животное он задумал. Д. Кларк из лаборатории вычислительной техники Массачусетского технологического института сообщил нам, что сотрудники одной фирмы увлеченно играли в «Животное». Хотя эта программа совсем не напоминает боевую программу и даже не имеет ничего общего с простыми существами Стала, она приобрела способность к воспроизводству в памяти компьютера благодаря усилиям программиста, направленным на то, чтобы повысить эффективность игры. Когда программа ошибается, называя не то животное, которое имел в виду человек, она просит его посоветовать, какой вопрос ей в будущем следует взять на вооружение, чтобы играть успешнее. Это свойство программы подсказало программисту одно техническое решение, благодаря которому у всех играющих все время была бы одна и та же версия «Животного».

«Работая с одной из очень ранних систем, в которой отсутствовали развитая структура каталогов коллективного пользования и средства защиты данных, программист изобрел весьма оригинальный способ сделать игру доступной сразу для нескольких пользователей. Допустим, у одного из пользователей библиотека содержит игру. Всякий раз, когда этот пользователь поиграл с программой, она создает свою копию в библиотеке другого пользователя. Если в этой библиотеке ранее уже имелась копия игры, старая версия стиралась и на ее место записывалась новая. Благодаря этому поведение программы становилось для играющего неожиданным. Если же библиотека ранее не содержала версии «Животного», то игра предлагалась еще одному пользователю».

Кларк вспоминает, что игра «Животное» была весьма популярна и в каждой программной библиотеке компании обязательно содержался экземпляр этой игры. «Более того, при переходе сотрудников фирмы из одного отдела в другой. . . они брали с собой и игру, и со временем она распространилась на все машины компании». Ситуация в общем-то не была опасной, но многочисленные копии этой сравнительно безобидной игры стали существенно засорять дисковую память. И только когда была изобретена еще более «жизнеспособная» версия игры, создавшаяся проблема была решена. Во время игры с новой версией «Животного» программа создавала не одну, а две свои копии в двух других библиотеках. Через некоторое время эта программа должна была заменить все старые

версии «Животного». Через год, по наступлении определенной даты, все копии новой программы изменили свое поведение. «Вместо того чтобы воспроизводить себя в двух экземплярах при каждом обращении к программе, она играла теперь одну заключительную партию, говорила пользователю «до свидания» и уничтожала себя. Вот так «Животное» было изгнано из системы.

Рут Льюорт из Холмдела, шт. Нью-Джерси, однажды создала своеобразного монстра, даже не написав программы как таковой. Компьютер, принадлежавший компании, сотрудником которой она была, работал в режиме разделения времени. Рут подготавливала демонстрационную версию обучающей программы и решила сделать запасную копию на другой системе с разделением времени. Когда система, с которой она обычно работала, стала перегруженной и реагировала на директивы слишком медленно, Рут переключилась на другую систему. «Эта система реагировала очень живо в течение тех трех минут, пока она работала, а потом мой графический терминал перестал реагировать на команды, а на его экране наблюдался полный хаос. Ни один пользователь не мог ни начать, ни нормально закончить выполнение своей задачи. Вывод был однозначным — в моей программе что-то было не так! Несмотря на охвативший меня страх, я внезапно вспомнила, что указала знак амперсанд в качестве символа, разделяющего поля данных, вводимых с терминала. Но амперсанд является также символом, означающим в системе порождение процесса, выполняемого в фоновой зоне! Когда компьютер принял первое сообщение с терминала, он, должно быть, перехватил амперсанды, предназначенные для терминала, и запустил несколько процессов, которые в свою очередь порождали новые процессы, и так до бесконечности». Срочный междугородный звонок проинформировал администратора системы о причинах внезапного «заболевания» машины, она была остановлена и запущена заново. Разумеется, Льюорт заменила амперсанд более безобидным символом, и ее программа «с тех пор всегда работала благополучно и весело».

Хотя программы для игры «Бой в памяти» и не размножаются подобным образом, дополнительные копии могут повысить их выживаемость. Несколько читателей считают, что следует всегда иметь три копии программы, с тем чтобы экземпляр программы, выполняющейся в каждый данный момент, мог определить с помощью двух других, не повреждены ли команды в данном экземпляре. При этом выполняющаяся программа может даже заменить ошибочную команду правильной. Эти соображения лежат в основе программы Scavenger, функции которой заключаются в защите файлов от случайных ошибок, появляющихся при их копировании на магнитной ленте. А. Хадсон из Ньютона, шт. Массачусетс (название фирмы, в которой он работает, не указано), пишет: «Всякий, кто часто пользуется магнитной лентой, должно быть, испытал на себе силу закона суммирования вероятностей, поистине представляющего собой враждебную стихию». В своем письме Хадсон рассказывает о многочисленных ошибках, возникающих при работе с магнитными лентами, и доказывает, что, хотя каждая из этих ошибок имеет сравнительно небольшую вероятность, если ее рассматривать в отдельности, какая-нибудь одна из них может произойти уже с довольно большой вероятностью, внушающей серьезное беспокойство. Он продолжает:

«Однако не волнуйтесь, программа Scavenger вам поможет. Если вы доверите ей свой файл, она скопирует его сразу на три магнитные ленты, не беспокоя вас никакими подробностями. Даже если в работе системы произойдет программный сбой (что случается несколько раз в день), список невыполненных или прерванных заданий обычно остается невредимым, и когда нормальный режим работы машины восстанавливается, все задания, связанные с действием программы Scavenger, выполняются своим чередом. Запись на каждую ленту производится в ходе отдельного прогона программы, запланированного операционной системой».

Тем, кто работает с компьютерами Apple (буквально — «яблоко». — *Ред.*), следует остерегаться маленькой вредной программы Apple Worm (буквально — «яблочный червяк». — *Ред.*), созданной Дж. Хаузером и У. Бакли из Калифорнийского политехнического университета в Сан-Луи-Обиспо. Написанная для компьютера Apple II на языке ассемблера 6502, эта разновидность червяка размножается в памяти машины, пронизывая ее насквозь. Сначала загружается специальная программа, написанная на Бейсике (см. рисунок на с. 71), которая в свою очередь загружает «червяка» в область памяти с малыми значениями адресов. Программа на Бейсике, наоборот, занимает противоположную часть памяти (с большими значениями адресов).

«Благодаря тому что «червяк» первоначально загружается в область памяти графического представления данных, вы можете наблюдать на экране, как он устремляется (между прочим, двигаясь задом наперед) в соседние области памяти с бóльшими адресами. . . После того как «червяк» покидает окно графического изображения на экране. . . проходит некоторое время и он стирает всю память, в том числе и программу на Бейсике, и с ходу врезается в постоянную память машины».

Хаузер и Бакли намереваются в недалеком будущем выпустить публикацию, посвященную разновидностям подобного «червяка». Они уже спроектировали Червячную Операционную Систему и даже создали видеоигру, одним из главных персонажей которой является Червяк.

Еще одна программная забава была изобретена Р. Черутти и М. Морокутти из Брешиа, Италия. Прочитав перевод статьи, посвященной игре «Бой в памяти», в "Le Scienze" — итальянском издании журнала "Scientific American", они стали размышлять о том, как заразить персональные компьютеры Apple II, но не «червяками», а «вирусом». Вот что пишет Черутти:

«Марко (Морокутти. — *Ред.*) задумал написать программу, способную переселяться из одного компьютера в другой подобно вирусу, заражая таким образом компьютеры Apple. Однако мы долго не могли придумать, каким способом можно осуществить эту идею. В конце концов я понял, что на самом деле программа должна была заражать флоппи-диски, пользуясь памятью машины, лишь как промежуточной средой между одним диском и другим. Так наш вирус начал обретать форму.

```
OAT                    0     /указатель на защищаемый адрес
ADD     #1            —1     /увеличить защищаемый адрес
PCT     @—2                  /защитить адрес
CMP     #102          —3     /если все 102 команды защищены...
JMP     2                    /...то выйти из цикла
JMP     —4                   /иначе выполнить его еще раз
                      .
                      .
                      .
ОСНОВНАЯ ЧАСТЬ «ВОЮЮЩЕЙ» ПРОГРАММЫ
                      .
                      .
                      .
```

Цикл, защищающий программу от случайного попадания бомбы

Как известно, на каждом дискете компьютера Apple имеется копия дисковой операционной системы, загружаемой в машину при включении питания. Вирус должен был представлять собой некое изменение в этой операционной системе. При каждой операции записи на диск он должен был проверять, содержится ли там его копия. Если нет, то «вирус» вносил бы аналогичное изменение в операционную систему, записанную на диске. Таким образом, он мог поселиться на всех дискетах, которые вставлялись бы в дисковод данной машины после первоначальной загрузки исходной системы, содержавшей «вирус». Мы подумали, что если записать такую операционную систему на один из дискетов, используемых в самом крупном магазине, торгующем персональными компьютерами, то в Брешиа возникнет эпидемия, которая распространится на весь город.

Но что это за эпидемия при таком безобидном вирусе? Нет, наш вирус должен быть злокачественным! Поэтому мы решили, что после 16 циклов воспроизводства, подсчитываемых на самом дискете, программа должна стереть и заново проинициировать дискет сразу же после загрузки операционной системы. Но теперь мы уже сами ужаснулись своей зловещей идее и решили не только отказаться от ее реализации, но и никому не говорить о ней ни слова».

Ну что ж, это было очень любезно со стороны Черутти и Морокутти. В персональных компьютерах дисковая операционная система (ДОС) — это царь и бог, вершащий судьбами программ, данных и всего остального. Согласно описанному выше замыслу, дисковая операционная система стирает диск, с которого она была загружена в машину, и потому уже не может быть загружена снова с этого диска. Заболевшая операционная система могла бы даже время от времени выдавать на экран удручающее сообщение:

ВАШ ДИСК БАРАХЛИТ?
Самое время приобрести
ДОКТОРА ДОС
Диски с ДОКТОРОМ можно
купить
в ближайшем магазине,
торгующем компьютерами

Вирусное заражение, описанное выше, уже имело место в действительности, правда в небольших масштабах. Р. Скрента-младший, учащийся из Питсбурга, написал программу, обладающую следующими свойствами. Вместо того чтобы стирать записи на диске или выводить на экран объявления, эта инфекция приводила к возникновению тонких, трудно уловимых ошибок в операционной системе.

«Все это кажется теперь детской игрой, — пишет Скрента, — но боже мой! Я никак потом не мог избавиться от этой электронной чумы. Она заразила все мои диски и диски всех моих приятелей. Она даже умудрилась проникнуть на графические диски учителя математики». Скрента сочинил программу, которая должна была уничтожить вирус, однако она оказалась не столь эффективной, как сам вирус.

На основании сказанного возникает неплохая задачка, и я поистине проявил бы недостаток воображения и даже своего рода безответственность, если бы не сформулировал ее своим читателям. На одной или двух страничках опишите программу ДОКТОР ДОС, которая хранилась бы на диске и каким-то образом защищала бы компьютеры от подобных электронных эпидемий. На многих дискетах, предназначенных для персональных компьютеров, содержатся копии ДОС. После включения машины она получает свою операционную систему, считывая ее с дискета. Загруженная в память машины ДОС будет оставаться в силе, когда машина будет работать и с другими дискетами, также содержащими копии ДОС. Если загруженная в память операционная система заражена, то она может внести изменения и в другие копии ДОС или даже полностью заменить их собою. Как можно воспрепятствовать подобному заражению?

В ПЕРВОНАЧАЛЬНОЙ версии «Бой в памяти» главной целью было защитить программу А от случайных попаданий «бомб», разбрасываемых боевой программой противника В. Если бы такая защита была более или менее гарантирована, то на следующем этапе эволюции игры можно было бы приступить к созданию программ, способных отыскивать друг друга и проводить уже «прицельные» атаки, если можно так выразиться.

Пытаясь обеспечить гарантию защиты от случайных попаданий, я предложил в прошлой статье ввести еще одну команду

РСТ *A*

где *A* — относительный адрес (прямой или косвенный) команды, которая должна быть защищена. Однократная попытка изменить содержимое ячейки с этим адресом будет заблокирована системой Марс, осуществляющей управление игрой. Однако следующая попытка сделать то же самое уже будет успешной. Мне кажется, что при помощи простого цикла каждая боевая программа может защитить свои команды от случайных бомб в течение достаточно длительного промежутка времени, чтобы успеть провести какое-то активное действие по отношению к программе-противнику. Такой защищающий программу цикл изображен на рисунке на с. 72. Он состоит из шести команд, четыре из которых выполняются при каждом прохождении через цикл. Таким образом, программа, состоящая из *n* команд (включая команды, составляющие цикл), потребует выполнения 4 × *n* команд, чтобы полностью обезопасить себя от однократного попадания. Однако такая защита едва ли будет эффективной против программы Карлик, которая производит два выстрела подряд по каждой ячейке памяти.

Эту команду можно применять и с другой целью, не предвиденной в первой статье, посвященной игре «Бой в памяти». С. Петерс из Тимару, Новая Зеландия, и М. Дэрем из Уинстон-Салема, шт. Северная Дакота, независимо друг от друга придумали, как можно воспользоваться командой РСТ для проведения атакующих действий. Программа под названием Карлик-ловушка разбрасывает по памяти нули, как обычно, но затем защищает каждый нуль, так что в соответствующую ячейку нельзя записать ничего другого. Это означает, что ничего не подозревающая программа противника может попасться в одну из ловушек в ходе перемещения на новое место. Команда, которая должна быть записана в ячейку, содержащую защищенный нуль, не окажет на эту ячейку никакого влияния. Позднее, когда процесс выполнения программы достигнет этой ячейки, программа погибнет, потому что нуль — это невыполняемая команда. Возможно, команду РСТ стоит включить в систему команд какой-нибудь из будущих версий игры «Бой в памяти», но пока я склонен воздержаться от этого в интересах простоты, которая является, пожалуй, пробным камнем любой игры.

Читатели поделились и многими другими идеями, среди которых можно отметить двумерный массив Памяти, предложенный Р. Нортоном из Мэдисона, шт. Висконсин, и правило ограничения дальности «стрельбы», предложенное У. Митчелом, преподавателем математического факультета Пенсильванского университета. Идея Нортона, пожалуй, не нуждается в пояснениях, а вот по поводу предложения Митчела следует сказать несколько слов. Программе дозволяется производить изменения в любой ячейке памяти, расстояние до которой не превышает определенного установленного количества адресов.

Благодаря этому правилу программа Карлик автоматически не сможет причинить никакого ущерба противнику, не находящемуся в его окрестности. Из этого правила следует и многое другое, включая акцент на перемещение. Каким еще образом программа сможет подойти на расстояние выстрела к противнику? У этого правила много достоинств, и я полагаю, что некоторые читатели из числа многих, обладающих своей собственной системой «Бой в памяти», изучат это предложение более глубоко и подробно.

Нортон предлагает также, чтобы каждой стороне предоставлялось более одной команды, выполняемой при каждом обращении к программе. Та же идея пришла в голову и многим другим читателям. На самом деле я решил принять это предложение. Теперь игра «Бой в памяти» приобретает более открытый характер.

Предложенное изменение можно реализовать, добавив следующую команду с условным названием «разветвить» к системе команд игры, приведенной на рисунке на с. 70:

SPL *A*

Когда управление достигает ячейки, в которой содержится эта команда, дальнейший процесс выполнения программы разветвляется на две части, а именно идет на команду, следующую за SPL, и команду, отстоящую на *A* адресов. Поскольку это нововведение сразу позволяет игроку иметь несколько программ, выполняющихся одновременно, необходимо уточнить правила, согласно которым система управления Марс распределяла бы время между программами. Существует две возможности.

Чтобы проиллюстрировать их, предположим, что один игрок имеет три программы A_1, A_2 и A_3, в то время как у другого — две программы, B_1 и B_2. Один из возможных вариантов заключается в том, чтобы сначала выполнить все три программы первого игрока, а затем две программы второго. Тогда порядок выполнения программ будет следующим: A_1, A_2, A_3, а затем B_1 и B_2. Далее эта процедура будет циклически повторяться. Другой вариант заключается в том, чтобы чередовать выполнение программ, принадлежащих различным игрокам. В этом случае последовательность выполнения будет такой: A_1, B_1, A_2, B_2, A_3, B_1 и т.д. Эти две схемы принципиально различны, если судить по производимому ими эффекту. Первая схема поощряет стратегию неограниченного распространения и потому, по-видимому, снижает роль интеллекта в игре. Вторая схема, наоборот, означает, что, чем большим количеством программ располагает игрок, тем реже каждая из них будет выполняться. Здесь, кажется, работает закон «уменьшающейся прибыли», поэтому я принимаю вторую схему. Цель же игры заключается в том, чтобы так или иначе вывести из строя все программы противника.

Новая команда создает богатые возможности. Чтобы проиллюстрировать простейшую из них, рассмотрим боевую программу Чертенок-стрелок:

```
SPL 2
JMP −1
MOV 0 1
```

Посмотрим, что произойдет, когда управление попадет в начало программы. Команда SPL 2 означает, что программе будет предоставлено право выполнить сразу две команды: JMP −1 и MOV 0 1. Первая из этих команд начинает выполнение программы сначала, а вторая приводит Чертенка в движение. Он будет двигаться все время в одном и том же направлении — «вниз», поскольку операндом команды MOV всегда будет следующий адрес, на что указывает (положительное) число 1. Таким образом, в каждом программном цикле возникают новые Чертята. Бесконечным потоком они устремляются в соседние области памяти, полные решимости уничтожить по пути любую программу неприятеля. На первый взгляд кажется, что против этой армады Чертят невозможна никакая защита, однако защита существует. Рассмотрим еще более простую программу Чертенок-ловушка, которая выполняется по команде SPL, содержащейся в основном теле программы, защищающей свой «верхний» фланг:

```
MOV # 0 −1
JMP −1
```

В каждом цикле выполнения Чертенок-ловушка записывает нуль в ячейку, непосредственно прилегающую к ней «сверху» в расчете поразить приближающегося чертенка противника. Здесь правило очередности выполнения играет уже решающую роль. Если Чертенок-стрелок принадлежит A, а Чертенок-ловушка принадлежит B, то для A потребуется *n* циклов для выполнения *n* Чертят, в то время как лишь один Чертенок может достичь ячейки, прилегающей к программе Чертенок-ловушка. При прочих равных условиях для B требуется лишь один цикл выполнения для Чертенка-ловушки, чтобы остановить приблизившегося Чертенка-противника.

Можно представить себе расширенный вариант игры «Бой в памяти», где обе стороны порождают и разворачивают целые армии программ, каждая из которых имеет свою специализацию — обнаружение, атака, защита и даже ремонт. Предстоит изучить еще много тонких приемов игры, подобных тому, что предложил Дж. Мак-Лин из Вашингтона, у которого возникла идея создать специализированную программу-ловушку, разбрасывающую команды перехода JMP по различным адресам памяти в надежде на то, что одна из таких команд попадет в расположение программы противника. Каждая такая команда JMP уведет управление из программы противника в программу-ловушку, тем самым программа противника как бы перейдет на сторону программы-ловушки.

В связи с тем что программ теперь становится много, возникает дополнительная проблема: как исключить случаи нападения на программы, принадлежащие одной и той же стороне, т.е. не нападать на «своих»? Здесь представляется необходимой система распознавания типа «свой — чужой».

Среди многих читателей, построивших свои системы игры «Бой в памяти», особо следует отметить троих: Ч. Годфри из Уилтона, шт. Коннектикут; Г. Мак-Рэя из Монмут-Джанкшина, шт. Нью-Джерси, и М. Роузинга из Литлтона, шт. Колорадо, которые постарались четко определить свои системы и документировать их программное обеспечение. Мне кажется, что читателям особенно интересно было бы познакомиться с документами, составленными Роузингом. Но по этому поводу и по многим другим соображениям, касающимся вопросов поддержания контактов среди энтузиастов игры, у меня возникла заманчивая идея. Если кто-нибудь из читателей, обладающих уже действующей системой «Бой в памяти», согласится стать директором ассоциации любителей этой игры, то можно будет наладить отправку документации на различные системы, предложений, касающихся новых правил, интересных программ и записей отдельных партий всем членам ассоциации. Один из добровольцев будет избран директором, остальные же будут помогать, выпуская регулярный листок новостей, участвуя в работе комиссий по утверждению правил и т.п. в зависимости от того, кому какая работа покажется интересной.

ПРОДОЛЖАЮТ поступать отклики читателей, пробовавших играть с обитателями океана на планете Аква-Тор (см. статью в рубрике «Занимательный компьютер» в журнале «В мире науки» № 2 за 1985 г. — *Ред.*).

Мы сможем обсудить лишь некоторые из многих интересных писем. Вообще говоря, при подборе правильных параметров наблюдаются естественные флуктуации численности акул и рыб. Некоторые читатели, которым хотелось сделать планету Аква-Тор более похожей на Землю, ввели в свои программы кое-какие дополнительные факторы. Ну что же, в игре напрашиваются усложнения, и, наверное, они нужны. Однако измененным системам присущ тот недостаток (при прочих равных условиях), что проводить какие-либо параллели со стандартной системой иногда уже становится опасно.

Среди читателей, которые за основу брали исходную версию системы, были Дж. Андерсон из Лодердейла, шт. Миннесота; С. Берггрен из Сателлит-Бича, шт. Флорида; М. Бойд из Амхерста, шт. Нью-Гемпшир; Дж. Коннет из Миннеаполиса, шт. Миннесота; Э. Куда из Парк-Ридж, шт. Иллинойс; Д. Хопкинс из Чэмпейна, шт. Иллинойс; Дж. Лемон из Эль-Сегундо, шт. Калифорния и К. Райт из Грейлинга, шт. Мичиган.

Среди вопросов, поднятых этими и другими читателями, был, в частности, вопрос об измерении продолжительности выживания. Разумеется, для «вечных» популяций никаких проблем не возникает, но было бы полезно иметь шкалу измерения для невечных сценариев. Измерение хрононами, как отмечает Стивенс, может привести к заблуждениям, если для продолжительности жизни и периода способности к размножению выбрать сравнительно большие величины. При попытках измерения циклами также возникают проблемы: прежде всего как определить цикл? Стивенс сделал интересное наблюдение: если акулы и рыба выживают в течение достаточно большого количества повторений основного, построенного на случайных числах цикла, то одна из начальных конфигураций повторится, и тем самым вечная жизнь после этого становится гарантированной.

Несколько читателей, включая Д. Эммануэла из Оук-Брука, шт. Иллинойс, Р. Файзела из Форт-Вашингтона, шт. Мэриленд, и Дж. Лью из Исследовательского центра Т. Уотсона фирмы IBM в Йорктаун-Хайтсе, шт. Нью-Йорк, описали в своих письмах современные теории, помогающие анализировать экологию планеты Аква-Тор. Пока у меня еще нет окончательного вывода по вопросу о том, помогут ли нам стохастические матрицы в выводе частных вероятностей выживания при произвольных комбинациях параметров. Однако интересно отметить, что дополнительно были исследованы уравнения Лотки-Вольтерра (сформулированные в 1931 г.), чтобы рассмотреть роль диффузии как фактора, влияющего как на хищника, так и на его жертву. Диффузия превращает гладко меняющиеся решения Лотки — Вольтерра в кривые более сложной формы. Исторические сведения, приведенные в письме Лью, свидетельствуют, что Альфред Дж. Лотка был американским математиком, который десятью годами раньше, чем Вольтерра, сформулировал эти уравнения.

Бойд воспользовался фазовой диаграммой, чтобы проанализировать динамику популяций акулы/рыбы. В каждый момент времени t откладываются текущие значения величин x — численности рыб и y — численности акул, как координаты точки на плоскости. По мере увеличения времени популяции описывают цикл, и точки на диаграмме рисуют замкнутую кривую причудливой формы вокруг фиксированного центра или фокуса. Бойд применил этот метод, чтобы изучить влияние размеров океана на выживаемость популяций. Он сообщает, что «в мирах более прямоугольной формы кривые на диаграмме утрачивают ярко выраженный фокус, траектории становятся более неровными и в конце концов приобретают совершенно случайный характер». Квадратная форма океана, очевидно, является предпочтительной.

Среди новшеств, введенных читателями, были такие, как жизнеспособность акул, мутации, двойные популяции рыб и планктон. В предыдущей статье, посвященной этой игре, я забыл упомянуть, что рыбы планеты Аква-Тор питаются вездесущим и изобилующим океаническим планктоном. Лемон ввел это свойство явным образом, поместив планктон в каждую точку океана, не занятую рыбой или акулой. Планктон размножается в свободных областях и играет ту же роль по отношению к рыбе, что рыба играет по отношению к акулам. Здесь также существуют вечные популяции.

Согласно предложению Э. Куда, акулы приобретают или теряют баллы, оценивающие их жизнеспособность, в зависимости от того, насколько хорошо они питаются. Таким образом, жизнеспособные акулы могут выжить без еды значительно дольше, чем примитивные акулы стандартной версии программы АКВАТОР. Э. Куда прислал графики (как и многие другие читатели, составившие программы по экологии планеты Аква-Тор), весьма сходные с данными, публикуемыми рыболовной компанией залива Гудзон.

Коннет в своей модели использует два вида рыб. Один вид представляет собой стандартную для планеты Аква-Тор разновидность. Другой вид размножается в любой свободной области к югу или востоку. Благодаря большей подвижности второй вид часто выживает дольше первого. Р. Иваско из Сакраменто, Калифорния, предлагает наделить акул и рыб такими характеристиками, как размер, скорость и подвижность. Эти характеристики должны управляться генетическими факторами. Берггрен разработал свою систему под названием Evolve (Эволюция) два года назад. Она напоминает АКВАТОР, но отличается тем, что виды эволюционируют под воздействием факторов окружающей среды. Таким образом, как рассуждал Берггрен, популяции достигнут равновесия, способствующего длительному выживанию.

Никто не смог решить тороидную задачу преследования. Сейчас я открою половину решения, чтобы не лишать читателей удовольствия, которое они получат при поиске другой половины. Вспомним, что при каждом ходе сначала двигается рыба, затем двигаются две акулы. Оставаться на месте не разрешается. Представим себе четыре луча, исходящих от одиночной рыбы. Каждый луч идет по диагонали и изгибается вокруг тора, рано или поздно встречаясь с самим собой. Если обе акулы находятся на паре противоположных лучей, то не важно, в каком направлении будет двигаться рыба — одна акула преследует ее на постоянном расстоянии, а другая все время приближается. Таким образом, рыба обречена. Я предоставляю читателям подумать над тем, как акулы охотятся за лучами, если можно так выразиться.

НОВАЯ попытка построить автомат типа «бобер-работяга с пятью состояниями» была предпринята Дж. Уингом из Бронкса, шт. Нью-Йорк. В ходе испытаний, проведенных 21 декабря 1984 г., машина Тьюринга, созданная Уингом, начала с чистой ленты и напечатала 1915 единиц прежде, чем остановилась. Этот результат был независимо подтвержден А. Брейди из Университета штата Невада и Р. Робинсоном из Калифорнийского университета в Беркли. Эта, по словам Брейди, «поразительная» машина Уинга, кажется, оправдывает скептицизм, с которым оба математика отнеслись к известию о том, что автомат У. Шульта (см. статью в рубрике «Занимательный компьютер в журнале «В мире науки» № 10 за 1984 г. — *Ред.*) представляет собой бобра-работягу с пятью состояниями. Этот автомат выдал лишь 501 единицу.