УДК 681.3
ББК 32.973
С37

Симонович С. В., Евсеев Г. А.

С37 Занимательное программирование: Visual Basic: Книга для детей, родителей и учителей. — М.: АСТ-ПРЕСС КНИГА: Инфорком-Пресс, 2002. — 320 с.

ISBN 5-7805-0779-1

Книга адресована школьникам 12–15 лет, желающим обучиться составлению программ для персонального компьютера. Прочитав ее, школьник узнает основные понятия программирования, поймет принципы хранения и преобразования данных, освоит стандартные приемы программирования. В качестве языка и среды программирования избрана система Visual Basic, как наиболее доступная и простая в изучении.

Книга рассчитана на самостоятельную работу школьника, не имеющего преподавателя или опытного наставника. Авторы заранее предусмотрели ответы на вопросы, наиболее часто возникающие на ранних этапах обучения. Главная задача книги — увлечь читателя интересными примерами и подвести его к уровню, после которого он сможет расширять знания с помощью учебных и справочных пособий.

УДК 681.3

ISBN 5–7805–0779–1

Становление программирования в России

Создание компьютерных программ в России началось 50 лет назад вместе с производством первых электронных вычислительных машин. Давайте мысленно перенесемся в начало 50-х годов XX века. Для техники это было необычное время. Сейчас нам трудно представить тот энтузиазм, с которым молодое поколение бралось за любые, самые смелые проекты в разных областях: в ракетной, космической и атомной технике, в радиолокации, авиации и судостроении, в вычислительной технике и программировании. Судите сами: что могло остановить людей, прошедших войну, переживших невообразимые трудности и потери? С любыми задачами они справлялись играючи, и не было таких проблем, которые могли бы их надолго остановить.

У новых научных и технических направлений были и замечательные организаторы. Десятки тысяч молодых офицеров, научившихся за годы войны руководить большими массами людей, вернулись в науку и технику, откуда их вырвала война. Им удалось создать сплоченные творческие коллективы.

К середине 60-х годов Советский Союз стал одной из ведущих держав мира в области вычислительной техники. В стране работали десятки тысяч программистов высочайшего класса. Они располагали далеко не лучшей микроэлектроникой, но проявляли незаурядную изобретательность и смекалку. Их трудами были быстро созданы несколько поколений первых ЭВМ, как правило ***превосходивших западные образцы.***

Портрет ЭВМ БЭСМ-6, компьютера, который на 10–15 лет опередил свое время

Так в нашей стране сложилась своя школа программирования. Ее отличало новаторство, умение находить нестандартные решения и, главное, командный дух в работе. Российская школа всегда высоко ценилась в мире. Ценится она и сегодня. Перед нашими программистами открыты двери многих зарубежных компаний по всему миру, особенно в Германии и США. Безработица им не грозит ни при каких кризисах в экономике.

Трудные 80-е годы

Во времена СССР в стране не было и не могло быть «коммерческого» программирования. Все программы создавались либо по государственному заказу, либо в ходе работ над государственными проектами. Созданные программы становились как бы частью ЭВМ или систем, созданных на базе ЭВМ. Такого понятия, как *создание программ на продажу,* не существовало.

Долгое время программы не считались товаром и в западных странах, но там после 1976 г. началось массовое распространение персональных компьютеров. Им были нужны тысячи разнообразных программ, от служебных до игровых. Первое время их создавали энтузиасты, но вскоре для многих это стало профессией.

К коммерческим программам предъявляются особые требования. Они должны быть привлекательны, выразительны и дружественны, иначе их не будут покупать. Программист Либо должен стать художником, мультипликатором, писателем, сценаристом и композитором, либо ему надо собрать команду людей, обладающих этими качествами. Такие коллективы не могут работать только на энтузиазме, и в мире появились компании, профессионально занимающиеся разработкой коммерческих программ.

Когда во второй половине 80-х годов в России возникли рыночные отношения, эмиссары ряда западных фирм приехали сюда набирать программистов и столкнулись с неожиданной проблемой. Из-за того, что в России никогда не было коммерческого программирования, у наших программистов не было ни традиций, ни опыта в оформлении программ. Они могли оригинально и талантливо написать ядро программы, составляющее ее суть, но старательно избегали заниматься оставшимися 90 процентами скучной и нетворческой оформительской работы. В те годы это было серьезной проблемой. Из-за нее образование в стране творческих коллективов, профессионально создающих коммерческие программы, отодвинулось на целое десятилетие.

Выход из тупика

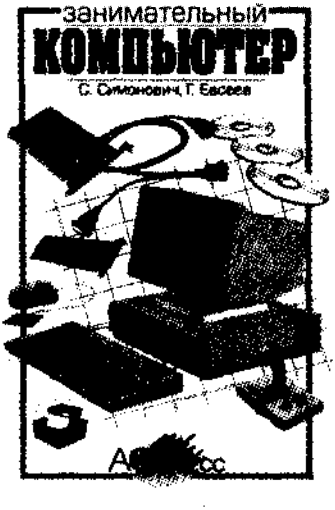
Проблемы с оформлением программ нашли наконец свое решение во второй половине 90-х годов. Тогда в дополнение к обычным языкам программирования

появились так называемые *системы программирования*. Они взяли на себя нелегкую заботу по оформлению программ, и программисты смогли сосредоточиться только на логике работы программ, то есть заниматься тем, что они любят и лучше всего умеют делать.

В этой книге вы научитесь работать с одной из самых простых систем программирования: Visual Basic. Благодаря ей даже школьник может осваивать программирование не «мелом на доске» и не «карандашом в тетради», а так, чтобы любая, даже самая простая программа, имела законченный вид и соблюдала все требования операционной системы *Windows*.

Что вам потребуется

Для работы с этой книгой необходим компьютер с операционной системой *Windows* и установленной системой програмирования Visual Basic, которая может быть любой, начиная с версии Visual Basic 4.0 до самой последней на момент написания книги — Visual Basic 6.0.



Очень хорошо, если вам уже знакомы основные приемы работы с компьютером: установка и запуск программ, создание папок, загрузка и сохранение файлов, работа с мышью и клавиатурой. Если у вас это вызывает трудности, прочитайте, пожалуйста, нашу книгу «Занимательный компьютер». Написана она простым языком, стоит недорого, а приобрести ее можно там же, где вы приобрели эту книгу.

Глава 1
Программирование без компьютера

Из этой главы мы узнаем, что программировать мы уже умеем и что компьютер отличается от простого будильника только тем, что понимает не одну команду, а много. Мы придумаем собственный язык программирования, попробуем составить несколько простых программ и научимся находить в них различия и ошибки

Программировать может каждый

Если ты думаешь, что не умеешь программировать, то, наверное, недооцениваешь свои способности. Вспомни, не приходилось ли тебе когда-нибудь заводить будильник, чтобы не опоздать утром в школу? Приходилось? Давай-ка вспомним, как это делается.

1. Устанавливаем стрелку будильника на нужный час.
2. Заводим пружину будильника.
3. Ждем, когда часовая стрелка подойдет к стрелке будильника. В момент, когда они совпадут, раздастся звонок.

То, что мы сделали, называется программированием. Правда, будильник — слишком простой прибор, и программировать его неинтересно. Единственная операция, которую он способен выполнить по программе, — издать утром противный звонок, который мешает сладко спать. Но, к счастью, будильник — не единственный прибор в нашей жизни. У нас есть еще автоматические стиральные машины, которые можно програм

мировать на выполнение нескольких операций (замачивание, стирка, полоскание, отжим белья). В некоторых семьях есть плиты СВЧ, которые можно программировать на приготовление пищи из сырых продуктов. Интересно, что для разных продуктов (мяса, рыбы, овощей) программы должны быть разными. Та программа, которая хорошо сварит кашу, непременно пережарит омлет.

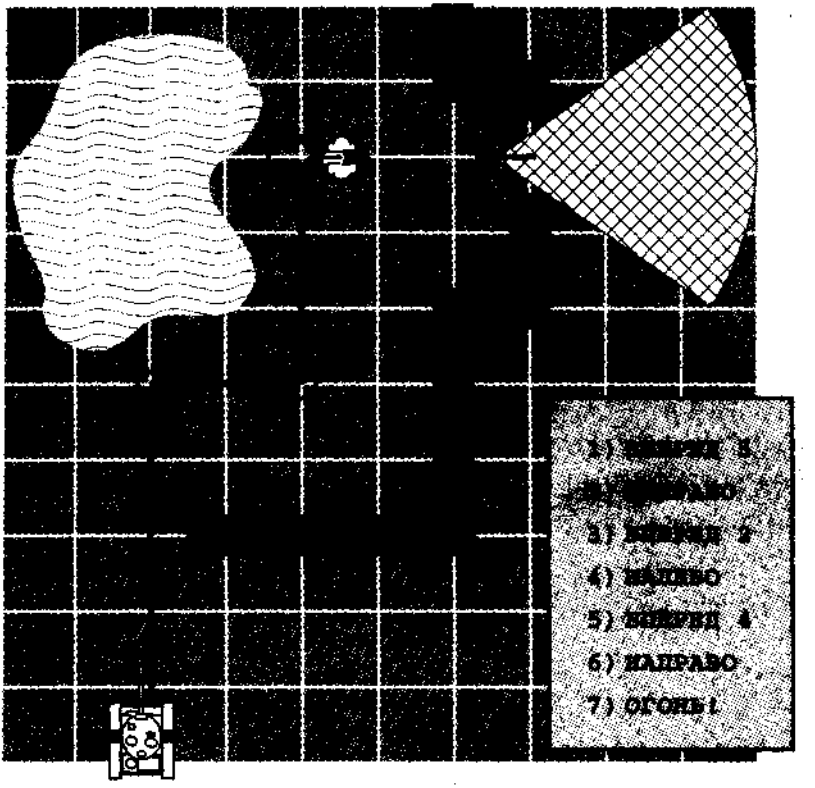
Чем сложнее прибор, чем больше операций он может выполнять, тем интереснее его программировать. Но ничто не может сравниться в этом деле с компьютером. Внутри него есть микросхема, которая называется *процессором*. Процессор компьютера может выполнять более тысячи простейших операций. Это очень много. В русском алфавите всего-навсего 33 буквы, а сколько разных книг можно написать с их помощью!? Теперь представьте себе, сколько разных программ, составленных из тысячи простейших операций, может исполнить компьютер. Это количество бесконечно точно так же, как бесконечно количество книг, которые можно написать русскими буквами.

Изобретаем команды

Всякая программа должна выполнять какую-то задачу. Будильник мы программируем, потому что у нас есть задача: получить в нужное время сигнал тревоги. Стиральную машину программируют, чтобы она в нужное время прекратила стирку и начала полоскание белья, а потом его отжим.

Взгляни на рисунок. Перед экипажем танка стоит задача: проехать по полю и уничтожить вражескую пушку. Пушка защищена валом, и только с одной стороны к ней можно приблизиться на прямой выстрел. А теперь

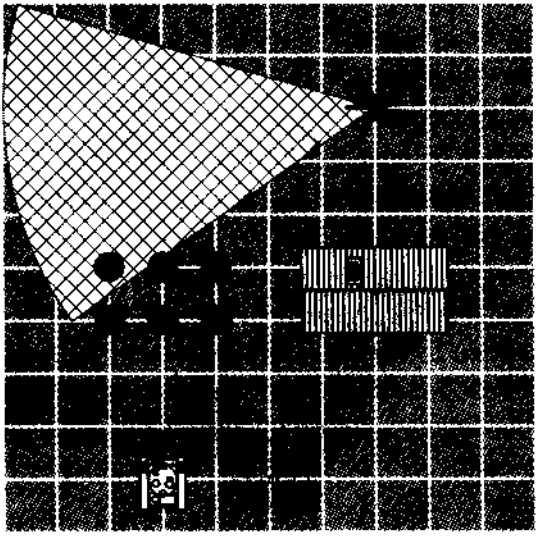
взгляни на программу, которая описывает действия экипажа.



Эта программа состоит из инструкций. Как и вся программа, каждая инструкция тоже выполняет какую-то задачу. Например, инструкции 2, 3, 4 и 5 мы ввели потому, что на маршруте находится озеро, а танки плавать не умеют.

Наверное, ты все уже понял. Мы только что изобрели набор команд. В негр входят четыре команды: ВПЕРЕД, НАПРАВО, НАЛЕВО и ОГОНЬ! С их помощью ты, наверное, сможешь и сам записать программу для движения танка по минному полю. На рисунке показан сектор обстрела противотанковой пушки. Как ты понимаешь,

заезжать в него нельзя. Определи, чем закончится движение танка по каждой из приведенных программ. Вариантов всего четыре: попадание в цель, промах, подрыв на мине и гибель под огнем орудия.



A

Вперед ?
Направо
Огонь

B

Направо
Вперед 2
Налево
Вперед " ?
Направо
Огонь

C

Направо
Вперед 6
Налево
Вперед 1
Налево
Огонь

D

Направо
Вперед 2
Налево
Вперед 5
Направо
Вперед 3
Налево
Огонь

Изобретаем язык программирования

Мы только что придумали несколько команд, с помощью которых можно записывать программы. Мы по-

чти придумали новый язык программирования и можем даже дать ему название, например Атака 1.0. Цифры 1.0 означают, что это первая версия нашего языка программирования. Если мы когда-нибудь введем в него новые команды, то получится новая версия, Атака 2.0, возможности которой будут больше. Если мы внесем в него мелкие изменения, будем менять вторую цифру: Атака 2.1, Атака 2.2 и так далее.

Давайте, например, придумаем команду ДЫМ. По этой команде наш танк создаст дымовую завесу. Вы уже догадались, зачем это нужно? После исполнения такой команды танку разрешается проезжать через сектор обстрела вражеской пушки. Чувствуете, как новая версия языка Атака 2.0 дает танку новые возможности? Так и в языках программирования. Каждая новая версия дает программистам новые возможности в управлении компьютером.

— А зачем программистам управлять танком? Танками управляют люди.

— Программистам все равно, чем управлять. Вместо танка может быть автоматический станок или космический аппарат, это может быть даже птицефабрика, которой управляет компьютер. У птицефабрики другие задачи, поэтому и набор команд будет другим, и язык программирования для нее лучше подойдет другой.

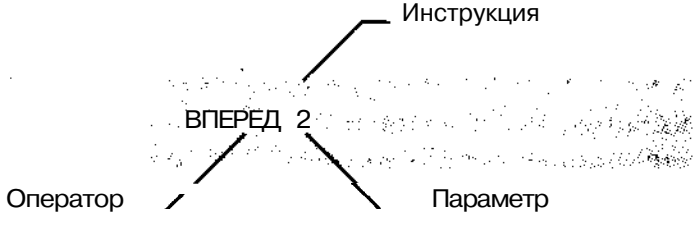
Инструкции, операторы и параметры

Для компьютера все команды являются инструкциями. Получив очередную инструкцию, он должен ее выполнить. Все инструкции процессор компьютера выполняет в том порядке, в котором они в него поступают. Обычно они выполняются по очереди, но среди множества инструкций процессора есть и такие, кото-

рые позволяют менять этот порядок очереди и переставлять инструкции с места на место.

А теперь давайте повнимательнее посмотрим на наш язык программирования Атака 2.0. В него входят команды ВПЕРЕД, ВПРАВО, ВЛЕВО, ОГОНЬ! и ДЫМ. Все эти слова являются *операторами* нашего языка. Это значит, что их (и только их) можно использовать в своих программах. Если в программе появится инструкция НАЗАД, то это будет ошибкой, потому что никакого оператора НАЗАД в нашем языке Атака 2.0 пока нет.

Список операторов — важная часть языка программирования, но это еще не весь язык программирования. Кроме списка операторов в нем должны быть определены правила их использования (они называются *правилами синтаксиса)*. Например, в нашем языке рядом с оператором ВПЕРЕД обязательно должно стоять какое-то число, которое показывает, как далеко должен продвинуться танк. Это число называется ***параметром.***



Из одних и тех же операторов можно получить разные инструкции, например:

```
ВПЕРЕД 5
ВПЕРЕД 3
```

Мы можем договориться, что параметр не обязательно должен быть положительным числом. Допустим, что

нам разрешается записать отрицательный параметр, например:

```
вперед -5
```

Это уже другое правило синтаксиса, причем очень полезное, потому что ВПЕРЕД -5 — это то же самое, что и НАЗАД 5. Вот мы уже и способны выполнить команду НАЗАД 5, даже не имея оператора НАЗАД. У нас на глазах рождается новая версия языка — назовем ее Атака 2.1.

Что такое компилятор?

У нашего языка программирования есть один важный недостаток — он записан символами русского языка. Это значит, что пользоваться им в Австралии или в Гренландии будет невозможно. В программировании есть традиция использовать буквы, символы и слова английского языка. Это не значит, конечно, что не существует российских языков программирования, но они плохо приживаются, и вовсе не потому, что русский язык меньше подходит для программирования, чем английский. Здесь дело в другом. Сейчас объясним.

Чтобы наша программа заработала, нашему языку еще кое-чего не хватает. Дело в том, что процессор компьютера не понимает ни русских, ни английских букв и никаких других. Он понимает только сигналы, которые записываются цифрами. Нам надо перевести слова своего языка программирования в цифровой код, который понимает процессор. Вручную сделать это очень трудно, и такой перевод поручают самому компьютеру. Для этого служат специальные программы, которые называются *компиляторами*. Вот компилятора-то нам и не хватает.

Составить набор операторов для своего языка программирования — дело нетрудное. Придумать правила для

их записи вместе с параметрами — тоже несложно. А вот сделать компилятор — задача неимоверной сложности. Самый первый язык программирования назывался FORTRAN (ФОРТРАН). Его первая версия появилась еще в 1958 г. Набор операторов и правила синтаксиса для него разработала группа программистов всего за несколько недель. А вот разработка компилятора заняла несколько лет и стоила десятки, если не сотни миллионов долларов!

Разработку набора операторов языка и правил синтаксиса называют *идеей* языка программирования. А разработку компилятора называют *реализацией* языка программирования. Пока у нас для языка Атака 2.1 — есть только идея, но реализации, увы, нет.

Реализация языка обходится гораздо дороже разработки его идеи. И это относится не только к первой версии, но и ко всем последующим. Нам нетрудно ввести в идею языка новый оператор, такой, как ДЫМ, но заново переделать компилятор при переходе от версии 1.0 к версии 2.0 — дело очень сложное. Такие сложные дела оправдываются только в том случае, когда языком программирования пользуются не несколько человек, а десятки и сотни тысяч людей. Вот именно для того, чтобы как можно сильнее расширить круг пользователей языка программирования, его операторы и стараются записывать английскими словами и буквами.

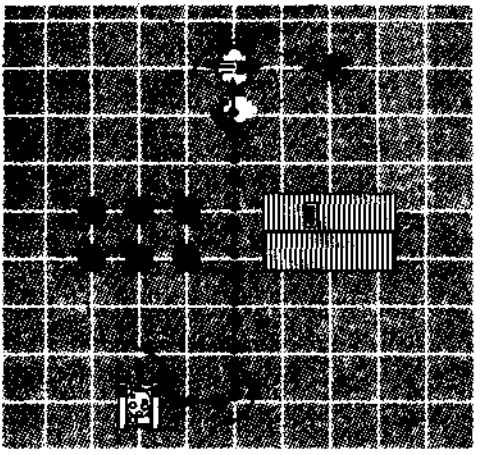
В разных странах не раз были попытки создать свои национальные языки программирования на русском, немецком, французском языках. Но опыт показывает, что из-за ограниченности круга потребителей они медленно развиваются, у них редко выходят новые версии и они быстро начинают отставать по возможностям от своих англоязычных конкурентов.

Нередко национальные языки программирования используют в качестве учебных, и в некоторых школах изучают русскоязычные языки программирования. Однако для того чтобы стать настоящим программистом, лучше все-таки осваивать универсальные языки, признанные во всем мире. Здесь очень поможет знание английского языка.

Делаем язык международным

Самое первое, что должен сделать тот, кто хочет стать программистом — это начать изучать английский язык. Однако дело это хоть и не очень трудное, но все-таки длительное — на это уйдет несколько лет. К тому же английский язык изучают не во всех школах. Так как же нам быть?

Будем постепенно тренироваться на примерах, а где необходимо, мы постараемся вам помочь. Вот, например, взгляните на следующую программу и догадайтесь, что делают команды LEFT, RIGHT, FORWARD, FIRE и SMOKE.



```
RIGHT
FORWARD 2
LEFT
FORWARD 6
SMOKE
FORWARD 1
RIGHT
FIRE
```

А теперь мы сократим наши команды так, чтобы вообще забыть, на каком языке они написаны. Ведь в реа-

лизации языка смысл слов, из которых состоят операторы, никак не используется. Мы ведь собираемся изучать программирование, а не английский язык, не правда ли? Новый язык программирования, который у нас получился, можно назвать по-английски: Attack 3.0.

Определите, на каком рисунке изображен результат действия следующей программы.

```
FWD 4
LFT
RHT
FWD 1
BMK
FWD 2
RHT
FRE
```