MY
ATARI XL
AND ME

JACK WALKER

My Atari XL and Me

Jack Walker

Duckworth

INTRODUCTION

This book is for children and for total beginners. It does not attempt to go beyond the absolute minimum necessary to understand the principles of programmming. I have tried to explain things as thoroughly and as simply as I can. I hope that I have succeeded in avoiding the assumption that after the first three or four pages the reader is miraculously transformed into an expert.

The difficulty with programming languages is not one of essence but of detail. There are a lot of words to understand and learn and put together. That is a good thing, because a good vocabulary can produce powerful sentences. But a collection of words is not necessarily a vocabulary. This book considers the most important words which happen (as is the case in any sort of language) to be the simplest. It deals not only with words but with the ideas that inform programming:

Actions can happen one after another, in sequence.

Actions can be repeated. They can be repeated a fixed number of times. Or they can be repeated until something happens to make them stop. If that something never happens they will keep repeating.

Actions can be aimed in one direction or another depending on some condition.

Actions act on objects.

1

A computer program is a collection of actions on objects. The objects are given to the actions. The actions process the objects to produce new objects. These objects may be processed by other actions in the program or they may be shown as the end results of the program.

One of the major ideas that this book does not deal with is that of recursion. It is not because this idea is particularly difficult, but because it comes into its own for programs that are more complex than any in this book.

An hour a day with the book, reading and acting on it, is a program for grasping how really simple and powerful the ideas of programming are.

The best possible readers I could hope for are children and parents reading the book together.
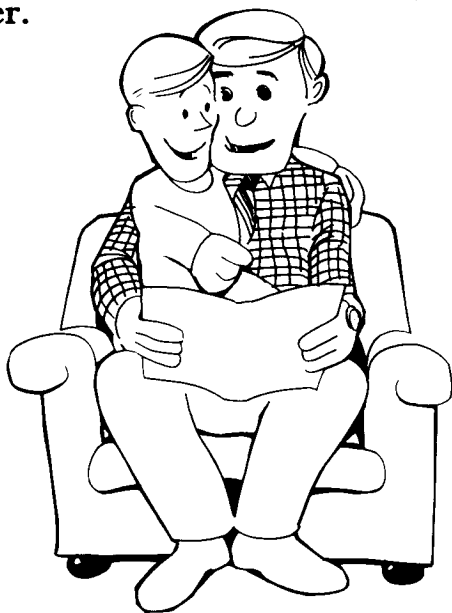
IT LOOKS FAMILIAR!

When you first look at your Atari, you might tell yourself, "This looks just like a typewriter!"

Have you ever used a typewriter? You first put a blank sheet of paper in it. You type and you see that words get PRINTED on the paper. Does the sheet of paper have to be white? Not really. Does the typewriter ribbon have to be black? Not really. You could have red ink on black paper, black ink on white paper, and so on. What would happen if you had ink the same colour as the paper? If you typed your name in red ink on red paper, would you be able to see your name?

Have you thought what happens when you type the letter A on a typewriter?
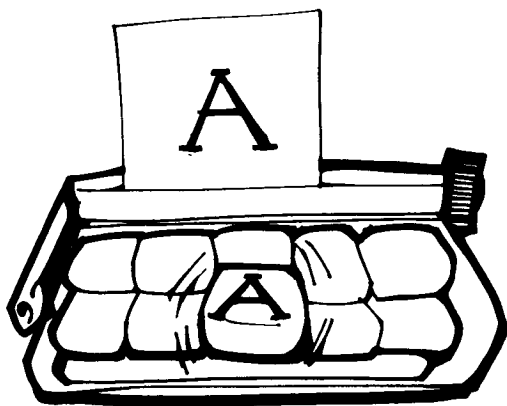
When you hit the A key, this makes the typewriter move levers and springs, an arm comes up and strikes the ribbon and the letter A is printed on the paper.

If you type the letter A on the keyboard of your Atari, the Atari obeys your command to print the letter A on the screen. But, of course, it uses a different method to the one used by the typewriter. The Atari has electronic chips inside it instead of springs and levers. It shows things on the screen in the same way as a TV set does.

Is your Atari switched on and connected?

Notice the READY sign. It tells you that the Atari is ready for you to type instructions and orders into it.
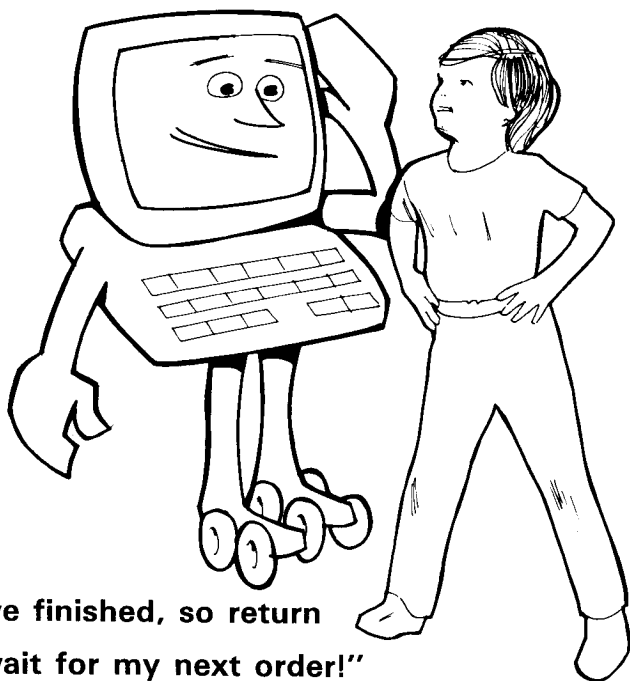
"I've finished, so return and wait for my next order!"

Type in your first name. Now, the way you tell the Atari that you have finished an instruction is by pressing the RETURN key. So press RETURN.

The Atari gives you a message:

ERROR—"NAME"

That's its rather cheeky way of telling you that it doesn't understand what you want it to do.

That's not really surprising, when you think that your Atari is really only a machine. Your TV set gives marvellous moving pictures and sound, but it needs to be switched on by you. You have to change the channels. You have to turn the brightness, volume, contrast and sound controls.

So you also have to give your Atari instructions in a special language that it can understand. This language is called BASIC.

One of the words in this language is PRINT.

Type in:

PRINT"ATARI"

Press RETURN to tell the Atari that you've finished your instruction. What do you see on the screen? What would you do if you wanted to PRINT your first name instead of ATARI?

DOING THINGS AT ONCE AND DOING THINGS LATER

When you are home at the weekend, you may suddenly decide to go out and play football. After that, you may decide to read a book. You can keep on deciding to do things as you think of them, without having a definite plan. But you can instead think ahead and make a list of what you want to do, one thing after another. If you keep a list, you can look at it later and see what you have to do.

Suppose this is your list:

    10 PLAY FOOTBALL
    20 READ A BOOK
    30 WATCH TELEVISION
    40 HAVE DINNER
    50 PLAY WITH THE
       ATARI COMPUTER
    60 END
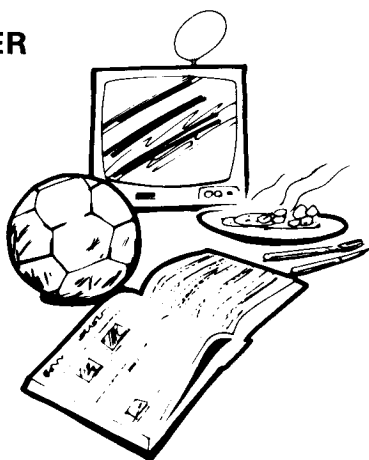
Why do you think the numbers go up ten at a time? Let's try and think why.

Suppose you want to have a shower after playing football. Then you can write:

15 HAVE A SHOWER

You can see that having the numbers go up by ten makes it easier to put in something else in your list of things to do.

Notice how you tell yourself to finish doing the things on your list. You put the instruction in Line 60.

Now, you can command the Atari to do things at once:

PRINT"ATARI"

(Don't forget to press the RETURN key after the command.)

You can, instead, make a list of commands for the Atari to obey:

```
10 SETCOLOR 2,3,14
20 PRINT"ATARI"
30 END
```

Don't forget to press the RETURN key after typing line 10 and after typing lines 20 and 30.

When you make a list of instructions for the computer to follow, the computer will remember them. But it won't act on them immediately. It will follow the instructions only when you order it to.

You give it the order by typing RUN.

The list of instructions is called a PROGRAM.

Now press the RESET key (The silver one—top right of keyboard) to return screen colour to normal.

If you want to tell the computer that you are going to give it a new program, type NEW.

So, now type NEW.

Now type in this program:

```
10 SET COLOUR 2,2,0
20 PRINT"ATARI"
30 END
```

Notice that Line 10 is different to the Line 10 you typed in before.

Now type RUN.

FAT LETTERS
AND THIN LETTERS

Now we are going to look at fat letters and thin letters. How do we make fat and thin letters? By changing the screen MODES of the Atari using the command—GRAPHICS.
   Why is that?
   It's because of the word GRAHPICS.
   GRAPHICS tells the Atari to divide the screen up into little blocks. Of course, you can't actually see these little blocks, but the Atari divides the screen up in its memory. Each of these is called a 'graphics mode'.
   When you tell the Atari to divide the screen up into GRAPHICS 2 blocks, the letters it shows on the screen are fatter than the letters it shows when it is in GRAPHICS 1.
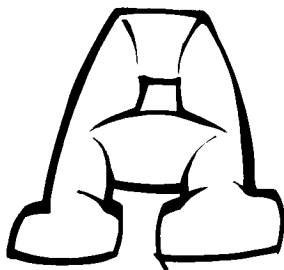
   Let's look at the way the letter A is shown in GRAPHICS 2.
   First, type NEW. To print on the screen in MODES 1 and 2, you have to use PRINT # (# is on key 3).

Now, enter this new program:

```
10 GRAPHICS 2
20 PRINT #6, "A"
30 END
```

Now RUN. You'll see the screen split and the READY is a blue strip at the foot of the screen and "A", in colour orange near the top. Notice how the A shape is made up of little dots.

Now let's look at the way the letter A is shown in GRAPHICS 1.

First, we have to command the Atari to go to GRAPHICS 1.

Does this mean we have to type in a whole new program?

Not really.

The only line you want to change is Line 10.
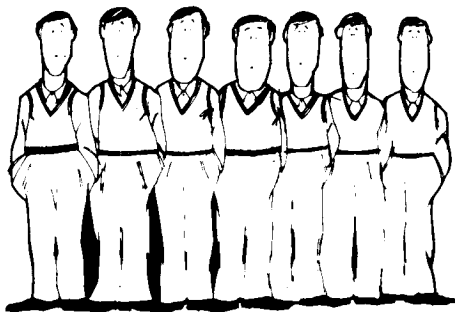
So, all you have to do is type in:

```
10 GRAPHICS 1
```

Now type RUN.

Notice how the letter A is not as fat as the letter A you saw for GRAPHICS (mode) 2.

That's because the little dots making up the letter A in GRAPHICS (mode) 1 are thinner than the dots in GRAPHICS (mode) 2 so it looks thinner.

Suppose some fat men try to squeeze into a telephone box. Then suppose some thin men do the same. You can see that more thin men will fit in. In the same way, more letters will fit on the screen in GRAPHICS (mode) 1 than in GRAPHICS (mode) 2. In GRAPHICS (mode) 1 the Atari can fit 24 characters down the screen. It can only fit 12 characters in GRAPHICS (mode) 2.
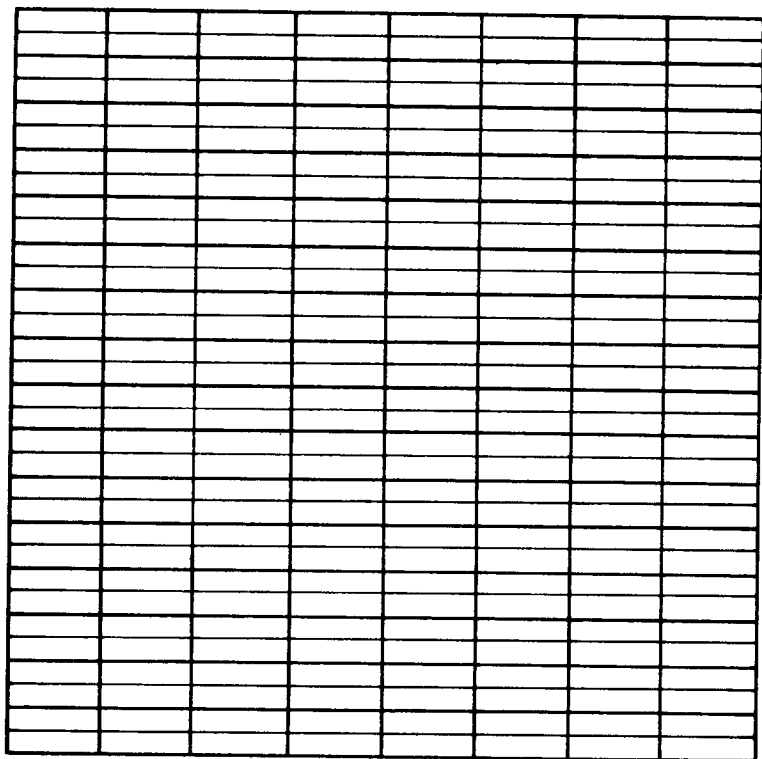
By the way, the little dots that make up a letter or a picture on the screen are called PIXELS. The fatter the pixels a letter is made up of, the fatter the letter is. The fatter the letters are, the fewer of them will fit onto the screen. It's just like the fat and thin men and the telephone box.

You can not only put letters on the screen. The pixels are dots. So you can order the Atari to join the dots together to draw lines, and triangles, and circles; and even pictures. Of course, you have to give the Atari careful instructions to do these things. As you practise more and more, you will learn how to write bigger and bigger programs.

The Atari can draw letters and pictures on the screen in sixteen different MODES as TEXT MODES.
You've looked at MODE 1 and MODE 2.

You can easily see what the letter A will look like in the other TEXT MODE number 0 (normal mode).

First, type LIST 10 to look at Line 10.

10 GRAPHICS 1

You just want to change the 1 to the number 0, (By the way, don't confuse the number 0 with the letter O. The number 0 is near the top right-hand corner of the keyboard next to 9 key.)

So, press the CONTROL key and hold down.

Use the upward-arrow key that's near the RETURN key to shift the ☐ sign (it's called the EDITING CURSOR) to go on the 1 of Line 10. Now, use the right-arrow key carefully until you see the editing cursor go on the 1 at the end of the word graphics.
Then release the CONTROL key and type in 0. Then keep pressing RETURN until the cursor is back to a blank line.

Now LIST and change line 20 to __ PRINT "A" by deleting # 6,.

Now you can RUN your program and see what the letter A looks like in 0 MODE (The normal screen MODE).

NUMBERS AND CHARACTERS

ONE NINE EIGHT FOUR
1984
NINETEEN EIGHTY FOUR

Let's go back to one of the programs we started with. Type in:

```
10 SETCOLOR 2,2,0
20 PRINT"ATARI"
30 END
```

Look again at line 20:

```
20 PRINT"ATARI"
```

Why can't we just say ATARI, without the '' marks? If you want, try it and you'll see that the Atari thinks it is a number and prints a zero.

That's because the Atari has to know whether you are talking about a *character* or a *number*.

Numbers can be added, or subtracted or multiplied or divided.

    1 + 2 = 3
    4 − 2 = 2
    4 * 2 = 8 (* means *multiply* for the
                Atari. Look for it on your
                keyboard.)

    4 / 2 = 2 (/ means *divide* for the Atari.
                Look for it on your keyboard.)

Characters are all the things that are not treated as if they are numbers.

"ATARI" is a collection of characters. You can't treat it the way you can a number. You can't do things like addition and subtraction to it.

The " marks tell the Atari that it is dealing with characters.

But let's look at 1984.

1984 looks like a number but it can also be treated as if it is a collection of characters. It depends on what you mean when you say 1984.

If you mean the year 1984, then it's a collection of characters. If it is how many pounds you have in the bank, then it's a number.

Your name is a set of characters. You can't multiply your first name by your surname to give a new name. But you can multiply ten by two to give twenty.

15

WHAT'S HAPPENING INSIDE?

Here, once again, is the list of instructions
for what you might do on a Saturday:

    10 PLAY FOOTBALL
    15 HAVE A SHOWER
    20 READ A BOOK
    30 WATCH TELEVISION
    40 HAVE DINNER
    50 PLAY WITH THE ATARI
    60 END

You write these instructions on a piece of paper. You can't write them on air! You need somewhere — a piece of paper — to put the instructions. If you write down even more instructions, you will need more paper.

In the same way, when you enter a program into the Atari, it is kept inside the space the Atari uses to store the instructions you give it in your program. The larger your program is, the more space it will need.

This space is in the computer's memory. You can think of it as being made up of tiny little brain-boxes.

Does that mean that if you could look into the tiny brain-boxes in the computer's memory, you would see the same sort of letters and numbers as the ones you write on a piece of paper?

No, not at all.

Take two little bits of blank paper. On the first piece, write a 1 on one side and a 0 on the other side. Do exactly the same thing on the other piece of paper.

Now put the two pieces side-by-side. If you do this and then turn one piece over, and then the other, you will get the following patterns:

11
10
01
00

If you tell yourself that you are an Atari, and that each of the patterns means something to you, you will get an idea of how the Atari stores information in its memory.

For example, you can tell yourself that the pattern 11 means the letter P.

Every time you press a key, the Atari changes the letters into patterns like the ones above. Of course, you've only got four patterns above. That's because you used two pieces of paper. But if you took *eight* pieces of blank paper and then did exactly what you did above, you would get 256 different patterns! That's more than enough for the Atari to change what you type in into the patterns of 0 and 1 that it can understand.

Of course, you can't understand the language that the Atari uses inside itself. So the Atari very kindly changes it to language you can understand, when it shows things on the screen.
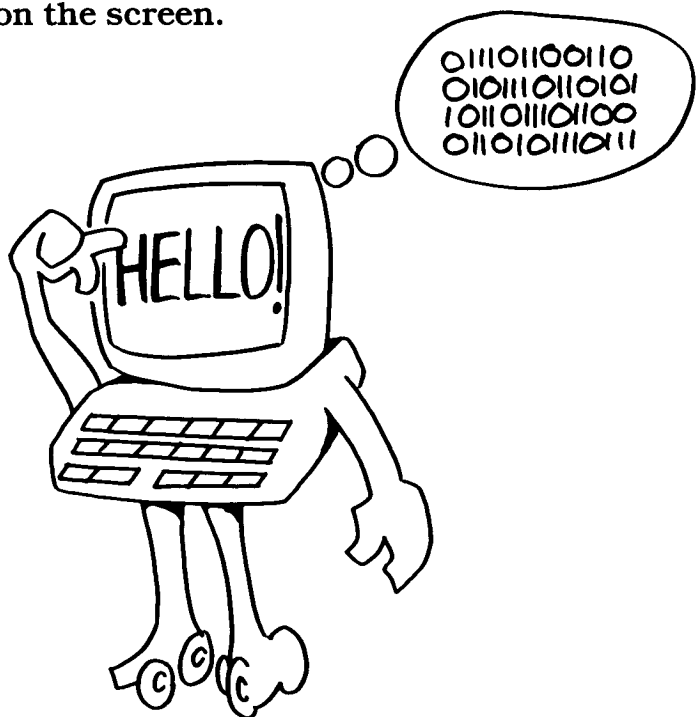
SOME THINGS REMAIN THE SAME AND SOME THINGS CHANGE

You were born with one nose. You will have one nose all your life (I hope!) When something doesn't change, it is called a CONSTANT. Can you think of other things that you could call constants?

See what's in one of your pockets right now. Maybe your pocket's empty! Or maybe it's got something in it — money perhaps, or a telephone number or the name of your favourite pop star. The contents of your pocket can change. Things that can change are called VARIABLES.

If your pocket contains things that can be added, subtracted, multiplied or divided, it contains NUMBER variables. For example, if your pocket contains five pence, you can add two pence to it to make seven pence. Or you can put 15 pence — three times five — into it.

If your pocket contains changeable things like telephone numbers or names or addresses, it contains what the Atari calls STRING variables. Remember that you can't treat these as if they are numbers. You can't take away your address from somebody else's address to get a new address!

You can imagine that the Atari has many, many pockets in its memory. It keeps number variables or string variables in its memory pockets. Of course, it can also keep things that don't change in its memory pockets.
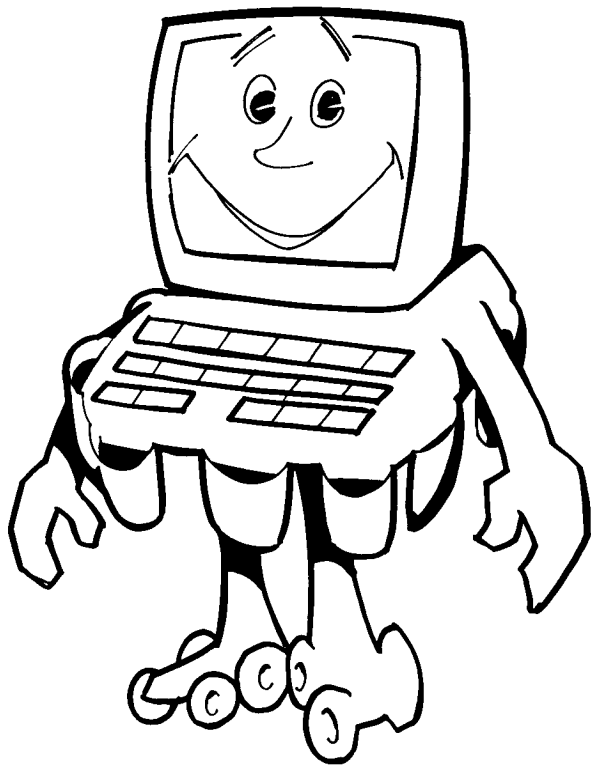
A BIGGER PROGRAM

Press RESET key, type NEW.

Now let's type in the same program that we used before:

10 SETTT

I did that wrong, but I realised it before I pressed RETURN. I had to correct it, of course, but all I had to do was keep pressing the DELETE (top right hand side of computer) key to get rid of the last two letters T.

Next I typed:

20 PRINT AN "ATARI"

Oh no! I didn't want the AN there. If I left it, the Atari would think I had typed in rubbish. So I had to correct it. I pressed RETURN and an ERROR message is produced. I placed cursor at the start of the ERROR message by pressing the up-arrow and CONTROL keys. I then released these keys and pressed the SHIFT and DELETE keys together. This removed the error message. I then held down the CONTROL key and used the up-arrow key (at right hand side of keyboard) to get on the 2 in 20. Now I used the right-arrow key to move the cursor to just past the N in AN. I released the CONTROL and right-arrow keys and pressed the DELETE/BACKSPACE key until AN was removed. Then I pressed RETURN until cursor was moved back to a blank line.

Now I pressed the SHIFT and CLEAR keys together and then typed LIST. That was tiring. But we all make mistakes, that's why it's so useful to be able to edit program lines, instead of having to type them out completely again. That would be even more tiring.

```
10 SETCOLOR 2,3,14
20 PRINT"ATARI"
80 END
```

I typed end in line 80 to leave room for further line entries.

We are now going to change this little program a lot, and make it bigger.

First, we want to get rid of line 20. All we have to do is type:

```
20
```

Now press RETURN. That's how we can get rid of a whole line in a program.

Press the SHIFT and CLEAR keys together and then type LIST.

We want to tell the Atari to reserve one of its memory-pockets for a NUMBER variable. Now, one pocket looks just like another. So how can we tell if the pocket should contain a number or a string variable in it?

Suppose you wanted to reserve one of your pockets for money only. One way of doing it (but don't really!) would be to get a sticky label, write CASH on it, and stick it on a pocket.

If you want to reserve a pocket for putting names only into it, you could get a sticky label and write NAME$ on the sticky label,

then put this on the pocket. The $ sign tells you that you are not dealing with numbers but with characters. You are dealing with STRING variables when you store names. You can't do arithmetic with names!

What would you do if you wanted to reserve a pocket for putting telephone numbers in? Is a telephone number a number that people add to other telephone numbers? Would PHONE$ be a nice sticky label to put over a pocket?

NUMBER VARIABLES

Let's pretend that your CASH pocket starts
off with no money in it. Your CASH is 0.
  We can say:

  CASH = 0

Now suppose that you add 5 pence to what's inside your cash pocket.

We can now say:

CASH = CASH + 5

How much CASH is in your pocket now? If CASH started off by being 2, and you added 7 to it, how much CASH would there then be in the pocket?

(What could you do if you wanted to reserve a pocket for MARBLES?) Type in a new Line 20:

20 CASH = 0

You've now ordered the Atari to reserve one of its memory pockets for a NUMBER variable called CASH. Inside this variable memory pocket, it puts 0.

How can we know what the Atari has in its variable memory pocket called CASH? After all, because it is a number variable, we could change it by adding, subtracting, multiplying or dividing. We don't want to do the hard work of remembering what's inside the variable pocket, especially if what's inside it keeps changing. We would prefer the Atari to do the donkey work and tell us what we want.

It's easy. All we have to do is to command the Atari to PRINT what CASH is on the screen.

So let's type at Line 40:

40 PRINT CASH

LIST then RUN the program. It won't surprise you to see 0 on the screen. After all, that's what CASH starts off as.

Suppose your friend came along just now and saw the 0, and asked you what it meant. You could say, "That's CASH". But why should you waste your breath when the Atari can give a message for you?

So type in Line 30:

30 PRINT "CASH = "

LIST the program. Remember that if the screen gets messy you can always clean it up with SHIFT/CLEAR.

See the difference between Line 30 and Line 40? The " marks in Line 30 tell the Atari that it is dealing with characters not numbers.

Here's our program:

```
 5 GRAPHICS 0
10 SETCOLOR 2,3,14
20 CASH = 0
30 PRINT "CASH = "
40 PRINT CASH
80 END
```

Now RUN the program. You can see how the Atari prints a message on the screen.

Now, say we want to increase the contents of the CASH memory pocket by 5.
Let's type in the fresh line, Line 50:

```
50 CASH  =  CASH + 5
```

Let's tell the Atari to print a message after this. Type Line 60:

```
60 PRINT "CASH = "
```

Let's order the Atari to also tell us what the CASH variable has changed to after we added 5 to it. Type:

```
70 PRINT CASH
```

Now LIST it.

```
 5 GRAPHICS 0
10 SETCOLOR 2,3,14
20 CASH=0
30 PRINT "CASH = "
40 PRINT CASH
50 CASH = CASH + 5
60 PRINT "CASH = "
70 PRINT CASH
80 END
```

What does the Atari put into the CASH number variable pocket when it comes to Line 20?

What does the Atari put into the CASH number variable pocket when it comes to Line 50?

Now RUN the program. Don't forget that whenever you want to see the lines of your program after you run it, you can use LIST again.

Let's go back to your own pocket. Let's suppose that you start off, once again, with no cash in it. Now, suppose you keep adding two pence to it, three times.

CASH = 0   Nothing at the beginning

CASH = CASH + 2 2 pence inside now
FIRST TIME

CASH = CASH + 2 4 pence inside now
SECOND TIME

**CASH = CASH + 2** 6 pence inside now
THIRD TIME.

Now, suppose you wanted to be able to tell yourself to add to what's in your pocket, without having to remember *how many* times you want to do this.

Suppose you took a piece of paper and wrote these instructions on it:

```
FOR K = 1 TO 3
CASH = CASH + 2
NEXT K
```

Now suppose you took a sticky label and wrote **K** on it and stuck it on another pocket. Now start reading the instructions you wrote.
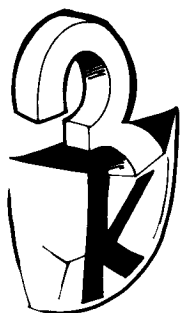
The instruction **FOR K = 1 TO 3** means that, the FIRST TIME , there will·be 1 in this K pocket. So you add 2 pence to the CASH pocket. When you see the instruction **NEXT K**, you know that the K pocket now has got 2 in it, for the SECOND time that you have to add 2 pence to the CASH pocket. Now you come to **NEXT K** again, and now the K pocket contains 3, for the THIRD TIME that you have to add 2 pence to the CASH pocket.

As soon as K becomes more than 3, you stop adding 2 pence to the CASH pocket. That's because the instruction **FOR K = 1 TO 3** tells you to do the action only three times.

Let's try this again, with different numbers.

Suppose you wanted to add 3 pence each time to your CASH pocket. Suppose you wanted to do this 4 times.

Here are the instructions you could write for yourself to follow:

```
CASH = 0
FOR K = 1 TO 4
CASH = CASH + 3
NEXT K
```

How many times are you adding 3 pence to the CASH pocket? How many pence will there be in the CASH pocket after you've added 3 pence to it for 4 times?

Remember our program so far:

```
 5 GRAPHICS 0
10 SETCOLOR 2,3,14
20 CASH = 0
30 PRINT "CASH = "
40 PRINT CASH
50 CASH = CASH + 5
60 PRINT "CASH = "
70 PRINT CASH
80 END
```

At Line 50, the Atari is adding 5 to its CASH number variable pocket. Let's order it to do this 20 times!

Type in Line 45:

```
45 FOR K = 1 TO 20
```

Now type in line 55:

```
55 NEXT K
```

LIST your program.
Now we have:

```
 5 GRAPHICS 0
10 SETCOLOUR 2, 3, 14
20 CASH = 0
```

```
30 PRINT "CASH = "
40 PRINT CASH
45 FOR K = 1 TO 20
50 CASH = CASH + 5
55 NEXT K
60 PRINT "CASH = "
70 PRINT CASH
80 END
```

The Atari will do everything between Line 45 and Line 70 as many times as Line 45 tells it to. Line 45 tells the Atari that K will go up to 20. You can change K to go up to 10 or 30 or any number you like. Try K going up to 1000!
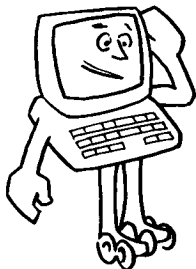
Suppose you wanted the Atari to add 7 to its CASH number variable pocket 40 times. What would you change Line 45 to? Try it now, and remember that at Line 50 you are ordering the Atari to add 7 to its CASH pocket.

SAVE IT!

Remember our list of things we might do at the weekend? You know, we might decide to scrap this list.
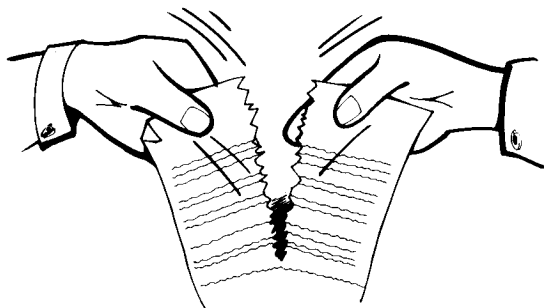
Suppose we wanted to use the list again some other weekend! We would want to save it and store it in a safe place. It would be pretty boring if we had to write the instructions in the list again and again. So we save the list.

Our Money Program for the Atari is something we should save before we switch off the Atari. Remember, although we sometimes treat this pretty little machine as if it's human, that's only because we have

imagination. The Atari is still just a machine, and as soon as we switch it off it goes to sleep and forgets everything we've taught it. So we should SAVE our Money Program before sending the little Atari to sleep.

But not yet! Just once more, look at the Money Program again. If it isn't already in the Atari, type it in. Change Line 45 to:

45 FOR K = 1 TO 1000

Now LIST it.

But this time, after you type RUN and press RETURN, press the BREAK key quickly. The Atari escapes from doing the Money Program instructions!
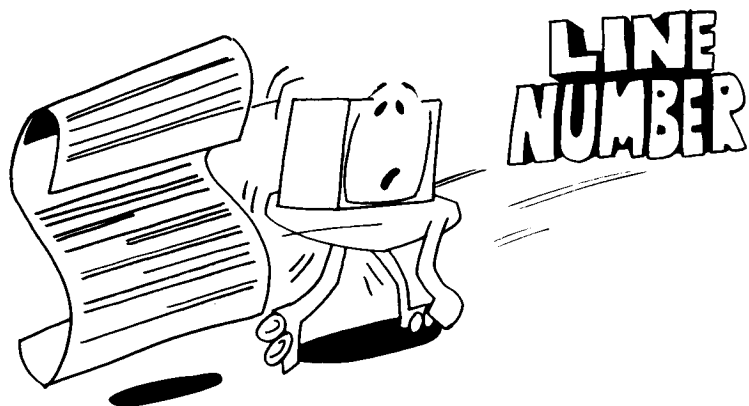
Notice that the Atari is really trying to look after you. It tells you at what line number the program stopped. Not bad for something that's really a collection of chips!

Before we let ourselves get carried away, let's remember that our little Atari can't think for itself. It tells you where it stopped because there's a program inside it that comes ready-made when you get the Atari. This ready-made program also does things like allowing you to save your programs on tape.

The BREAK key allows you to stop a program that's running. But the program itself comes to no harm. You can RUN it again or LIST it and even change it. Changing a program is called EDITING a program. We've already done some editing. BREAK gives you a chance to change your mind when you are in the middle of running a program.

Just a minute! Sometimes, people who write programs don't want you to be able to use the BREAK key in the way we just talked about. So you'll find that they have stopped the BREAK key from being able to do what it usually does. They have disabled this key. Sorry to have kept you waiting to find out how to SAVE your MONEY Program. Here's how:

First, make sure that your cassette recorder is connected both to the computer and to the mains switch. Be very careful with electricity! Really, it's better to ask your parents or someone older to do all the connections for you.

Put a tape — one of the 15-minute ones — into the recorder. Make sure that the tape is at its beginning, then wind it just a little way forward so that there really is magnetic tape to record your program and not a little

35

bit of dead plastic!
   Now type:

   SAVE "C"

   Remember, our little Atari doesn't know
you've finished giving it a command until
you press the RETURN key. So press
RETURN. The computer will honk twice at
you. Remember the sound comes from the
TV loudspeaker, so if you have the volume
turned down you wont hear it!
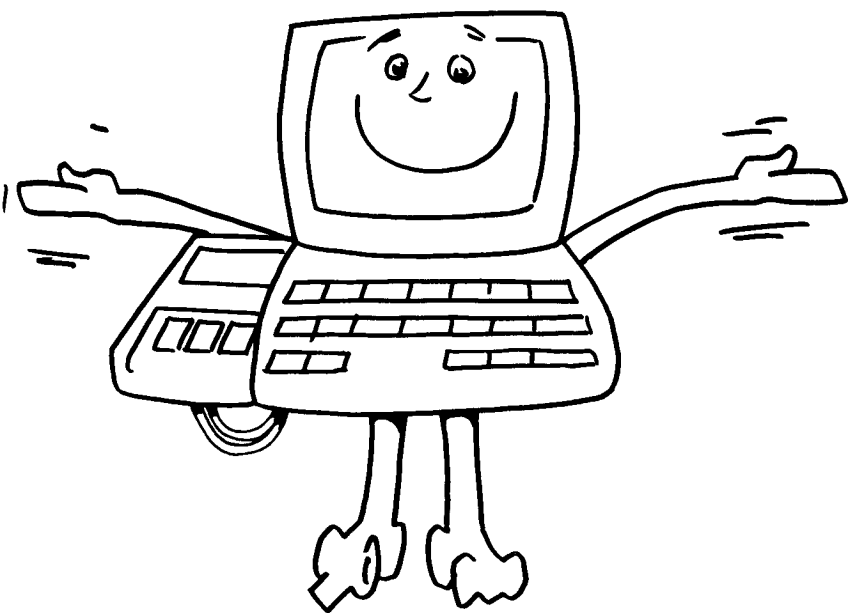
   Now press the PLAY and RECORD keys on
the recorder. Press them firmly so they
lock in place, ask someone to explain what
buttons to press to record something on a
cassette recorder. After that, press RETURN
and the recorder reels will turn.

While your Atari is recording your MONEY program, a high-pitched sound comes from the loudspeaker of the TV changing to a rough note. If it records it successfully, you'll see the little READY message sign coming onto the screen.

Sometimes, your program may not be recorded successfully. When that happens, ask someone older for advice.

Now that you've learnt how to save a program, you will want to know how you can bring it back into the Atari from where it is on the tape. Of course, you should have an idea as to where on the tape you saved your program. If you have a counter on your tape recorder, you can note down at what numbers your programs begin and end.

Suppose a program begins at 150 on your tape counter. You can wind the tape forward or backward to just before 150. Then you can start loading in your program.

1. Get as close as possible to just before the beginning of your program. You may have to wind your tape to do this.

2. Type:

LOAD"C"

3. Now press RETURN.

4. The computer will honk, just once this time.

Now press the PLAY button and then the
RETURN key.

Then when you see the READY message
sign, you know the MONEY program is
safely back inside the memory of the Atari.

If you wanted to, you could now LIST the
program and change it around — in other
words, EDIT it. Instead, you could order the
Atari to obey the instructions of the
program, by using RUN.

At last. We've saved the MONEY program.
We can load it in later any time we like.
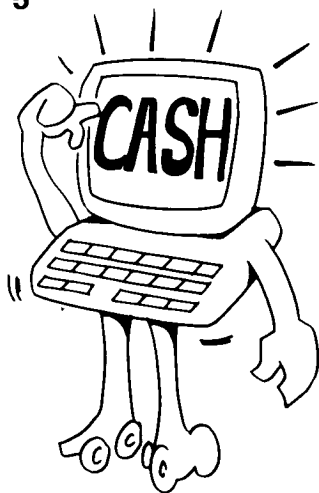Let's have a bit of a break.

SHOWING OFF

I hope you saved your Money Program on
tape. If you did, load it in and LIST it.
   Anyway, here it is again:

```
 5 GRAPHICS 0
10 SETCOLOR 2,3,14
20 CASH = 0
30 PRINT "CASH ="
40 PRINT CASH
45 FOR K= 1 TO 20
50 CASH = CASH + 5
55 NEXT K
60 PRINT "CASH = "
70 PRINT CASH
80 END
```

   RUN the program.
   Let's look carefully at what the Atari
displays, or shows, on the screen. Notice
that the message CASH appears on one line
of the screen, and the number on the next
line below.

Maybe we prefer to have the number on the *same* line of the screen as CASH. How can we do this?

Of course, we'll have to change or EDIT the program, because we now want it to do something different.

Press the CONTROL key and hold down. Use the up-arrow key to move the EDITING CURSOR to get on the 3 in Line 30.

Press the right-arrow key gently until the editing cursor gets just past the end of Line 30, press the SPACE BAR (the long bar at the bottom of the keyboard) once. This gives a little space to separate one thing from another.

Now, what we want to do is have the Atari print what's inside the CASH pocket, just after the message, on the same line. So, if Line 30 now became

```
30 PRINT "CASH = " ; CASH
```

it would do the trick. Release the CONTROL key. Now type a ; then press the SPACE BAR to get a separating space.

Now type CASH. Press RETURN to tell the Atari that you've finished typing in line 30. Then continue pressing to move the cursor to a clear line.

What about Line 40? We don't need it anymore, because Line 30 is doing its job. To get rid of it, just type 40 and then press RETURN. If you like, clean the screen with CLEAR/SHIFT, then LIST.

Now RUN the Money Program and notice the display.

Just for fun, LIST Line 30 and EDIT it again, but this time use a , instead of a ;

RUN this version of the program and notice what the display looks like this time. Now LIST line 30 and EDIT it back to what it was before.

Can't we do the same things to Line 60 and Line 70? Of course we can. The ; or the , tells the Atari to print what comes behind it, on the same line of the display. Why don't you now change Line 60 and get rid of line 70?

Your new Money Program should look like this:

```
 5 GRAPHICS 0
10 SETCOLOR 2,3,14
20 CASH = 0
30 PRINT "CASH = " ; CASH
45 FOR K = 1 TO 20
50 CASH = CASH + 5
55 NEXT K
60 PRINT "CASH =" ; CASH
80 END
```

POSITION DANCING

I wonder if you've seen any old film musicals where someone danced across a floor that was divided up into squares just like a chess-board is.

Imagine that the screen is like that chess-board dance floor.

In MODE 2 the screen "dance floor" is divided up into 20 squares when you look from left to right. When you look from top to bottom, the "dance floor" is divided into 12 squares. If you want to, you can divide a big piece of paper into squares. If you do, make it 20 squares broad and 12 squares deep. This may help you to imagine what I'm saying much better. Each square can fit a single thing into it. This can be a single character like A or M, or a single number like 5 or 9.
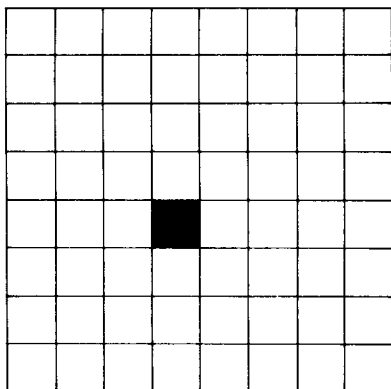
Each square is itself divided up into 8 PIXELS across and 8 PIXELS down.

Remember that we said that PIXELS are little blocks. Remember that we also said that PIXELS can be fat or thin, depending on what MODE the Atari is in.

The fatter the PIXELS are, the fatter the squares of the screen "dance floor" are. Just like fewer fat men could fit into the telephone box, this means that the "dance floor" will have fewer squares in it in MODE 2 than it will have in MODE 0.

In MODE 2, there are 20 squares across the screen. In MODE 0, there are 40 squares across the screen. The PIXELS of MODE 2 are twice as fat as the PIXELS of MODE 0.

Let's go back to the Money Program and imagine that we want to move the word CASH over the squares of the screen.

The instruction to move things over the screen is POSITION.

Suppose we want to move CASH five squares across the screen. POSITION 5,0 will do this for us.

You can imagine that there is a dancer called TEXT CURSOR who is carrying CASH across the screen dance floor. The numbers 5,0 tells TEXT CURSOR to move CASH by five squares across.

EDIT Line 30 of the Money Program:

30 POSITION 5,0: PRINT "CASH = " ; CA-SH

Before we carry on, let's think about how we should EDIT Line 30.

First, press the CONTROL key and hold down.

Now use the up-arrrow key to get the EDIT CURSOR to get on the 3 in 30.

Now use the right-arrow key to move just after the 30.

We now have to put something new in. So press the INSERT key (top right), then type 15 times then release the CONTROL key. POSITION 5,0:

Now press RETURN several times to move cursor down to blank line.

Now LIST and RUN the program.

Suppose we want TEXT CURSOR to POSITION dance across the screen, but not just five squares across but also ten squares down?

POSITION 5,10 will do that for us. The first number is 5. It tells TEXT CURSOR how many squares to tab dance across the screen. The second number is 10. It tells the TEXT CURSOR how many squares to tab dance down the screen.

Where does TEXT CURSOR begin its dance?

Look at the top left-hand corner of the
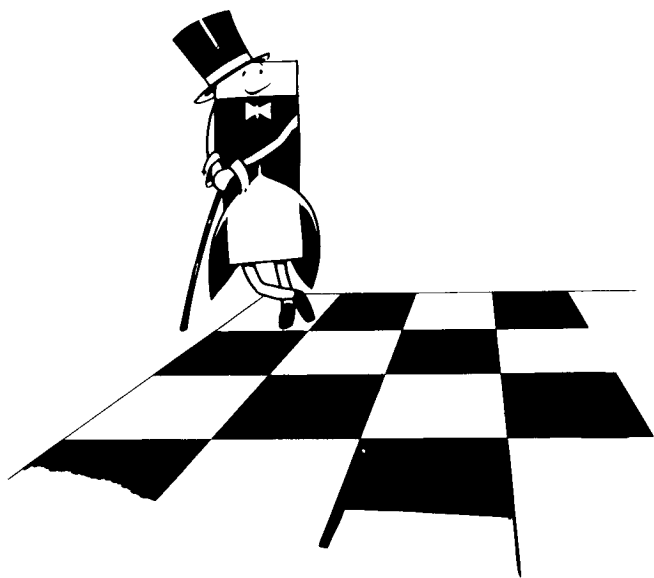
screen. Imagine TEXT CURSOR standing
there, dressed in top-hat and tails and shiny
shoes!

If you tell it to POSITION 5,0 it will dance
five places across the screen. It will then put
down whatever it's "carrying".

If, instead, you tell it to POSITION 5,10 it
will dance 5 squares across and ten squares
down the screen. It will then put down, on
the screen, whatever it is "carrying".

Now let's EDIT Line 30 to be:

30 POSITION 5,10: PRINT "CASH = " ; CASH

Now LIST and RUN the program.
Let's now EDIT Line 60:

60 POSITION 5,12: PRINT "CASH =" ; CASH

Where do you think POSITION 5,12 makes
TEXT CURSOR dance to?
LIST and RUN the program.

45

THE COLOURS OF THE RAINBOW

So far, we've been writing on the screen in white ink on blue background. As the white ink is on top of the blue background we call the characters on top the FOREGROUND and the colour underneath the BACKGROUND COLOUR. The colours around the edge of the screen are called BORDER colours.

Remember, whatever is on top is called the FOREGROUND. Whatever is underneath is called the BACKGROUND and around the edge, BORDER.

In MODE 0 (the normal MODE), if we want to write on a red background we can tell the Atari what we want by saying SETCOLOR 2,3,14. (Note the commas.)

Where SETCOLOR is the command to select a colour, 2 is the register number for background or screen, 3 is the number of the selected colour, red and 14 is the brightness of the colour. (This number must be an even number ranging from zero to 14).

To change the border in MODE 0, see the following example command: SETCOLOR 4, 6, 12 (Note the register number is now 4)

To change the brightness of the characters see the following example command: SETCOLOR 1, 0, 0 (Note the register number is now 1)

Colour No.	Colour	Colour No.	Colour
0	Grey	8	Dark Blue
1	Gold	9	Light Blue

2	Orange	10	Green
3	Pink	11	Yellow-Green
4	Violet	12	Light Green
5	Purple	13	Olive Green
6	Blue	14	Orange

Suppose we want our program to appear on a green background with a gold border. Change line 1Ø:

1Ø SETCOLOR 4,1,14

This gives us a Gold colour BORDER. Type in new line, Line 15:

15 SETCOLOR 2, 12, 14

This gives us GREEN paper (BACKGROUND).

LIST the program, then run it.

Now RUN the program, and see what happens.

Suppose you change Line 15 to SETCOLOR 2, 12, 1Ø to give the same brightness to characters and background. What happens when you use the same brightness? Why not try it and see what happens when you RUN the program.

If you do try it, try also listing the program!

If you want to change the background colour directly, without writing it in a program line, all you have to do is type the command SETCOLOR 2 and the background and brightness number you want.

So, if you find that you can't see the listing because both the character and background are the same brightness, just
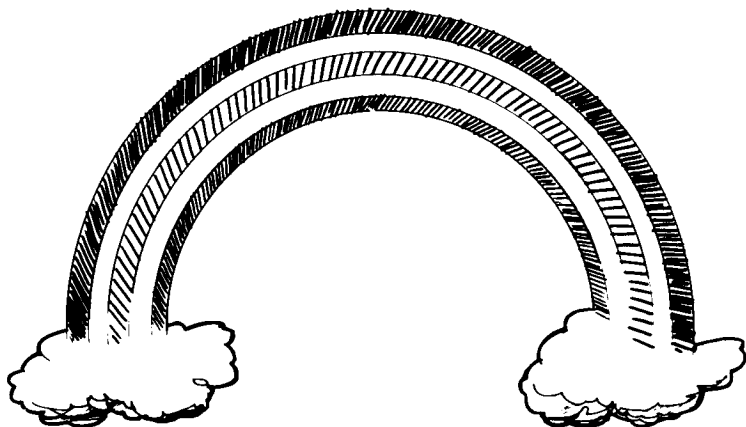
change the brightness number. You can do
this directly by typing SETCOLOR and a
brightness number that's different to the
normal brightness number of 10. Try
SETCOLOR 2, 12, 0.
Now you can LIST the program and see it
again.

But, if you want to have different ink and
paper colours while the program is
RUNNING, you will have to make sure that
Lines 10 and 15 give different colours.

Try different screen, border and
brightness values.

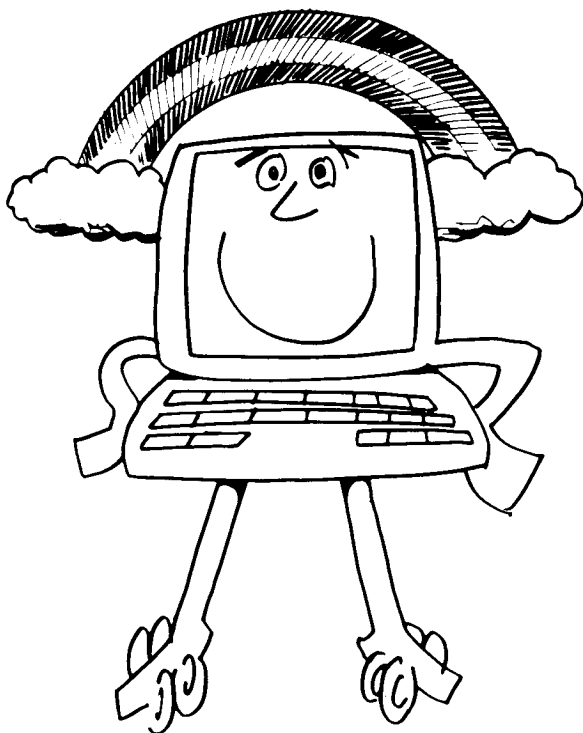# DIFFERENT MODES

Of course, the Atari has other MODES.
In MODE Ø, the Atari has 1 colour available
In MODE 1, the Atari has 5 colours available
In MODE 2, the Atari has 5 colours available
In MODE 3, the Atari has 4 colours available
In MODE 4, the Atari has 2 colours available

In MODE 5, the Atari has 4 colours available

In MODE 6, the Atari has 2 colours available

In MODE 7, the Atari has 4 colours available.

In MODE 8, the Atari has 1 colour available.

In MODE 9, the Atari has 1 colour available.

In MODE 10, the Atari has 9 colours available.

In MODE 11, the Atari has 16 colours available.

In MODE 12, the Atari has 5 colours available.

In MODE 13, the Atari has 5 colours available.

In MODE 14, the Atari has 2 colours available.

In MODE 15, the Atari has 4 colours available.

Only the first three are classified as TEXT MODES the rest are GRAPHICS MODES and are used for drawing and patterns on the screen.

We are not going to use them now but they would be used in a more advanced book. We are mainly using the 'everyday' working MODE, MODE Ø.

SWEET SIXTEEN

MODE 11 is very interesting, because it can have 16 screen colours available.

Of course, you'll have to EDIT your program if you want to try screen (background) and border colours.

If you do want to EDIT your program, first LIST it. It should be:

```
 5 GRAPHICS 0
10 SETCOLOR 4,1,14
15 SETCOLOR 2,12,14
20 CASH = 0
30 POSITION 5,10: PRINT "CASH = ";CASH
45 FOR K = 1 TO 20
50 CASH = CASH + 5
55 NEXT K
60 POSITION 5,12: PRINT "CASH = ";CASH
80 END
```

Now put a tape into your cassette player and SAVE this program.

Now change Line 10 to read:

```
10 SETCOLOR 4,4,14
```

Change Line 15 to:

```
15 SETCOLOR 2,10,14
```

Now RUN the program.

You can have a lot of fun by using border and changing different COLOUR background and the brightness numbers valves.

BACK INSIDE THE ATARI'S POCKETS

If you remember, the pockets of the Atari's memory can have STRINGs inside them. To show that what's inside the pocket is a string, we can imagine that the sticky label on the pocket has the name of the pocket written on it, with a $ at the end of the name.

So, suppose we have a string pocket that we label A$. Suppose we want to put the string "NUT" inside the A$ pocket.

Now, we are writing something new, so don't use the Money Program. If the Money Program is already in the Atari, press RESET key and type in NEW.

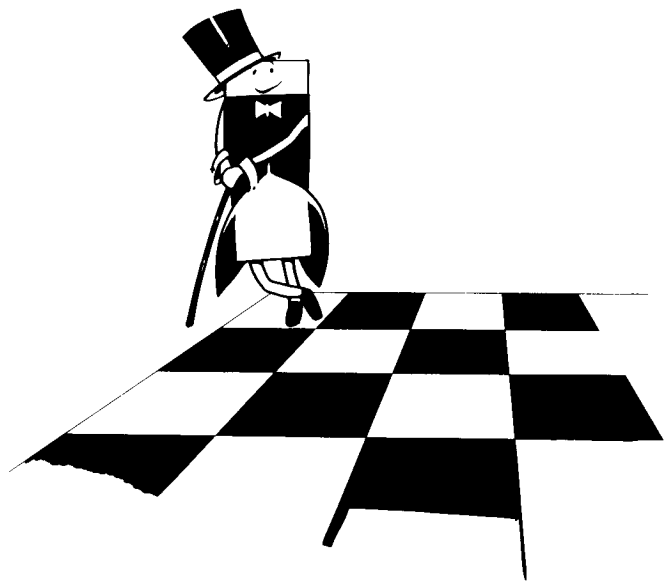Type:

```
10 DIM A$(10)
40 A$ = "NUT"
```

This puts the "NUT" character string inside the variable string pocket that is labelled A$.

Type:

```
50 PRINT A$
```

RUN this little program.

Let's not forget our friend TEXT CURSOR, the POSITION dancer.

LIST the program and change Line 5Ø to:

5Ø POSITION 5,1Ø: PRINT A$

Line 5Ø shows on the screen what the variable string pocket A$ contains. What does A$ contain?

Why not try and add Lines to change screen, border and colours and the character brightness?

Try getting a blue border with a green screen (background).

Did you manage? Here's what I got:

```
 5 GRAPHICS Ø
1Ø DIM A$ (1Ø)
2Ø SETCOLOR 2,12,14
3Ø SETCOLOR 4,1Ø,14
4Ø A$ = "NUT"
5Ø POSITION 5,1Ø: PRINT A$
6Ø END
```

Why not type in this program press SHIFT/CLEAR keys and try it?

Now let's add:

6Ø POSITION 5,12: PRINT "CRACKER"

Let's clear the screen with SHIFT/CLEAR and RUN the program. It looks like an advertisement for a ballet!

Well, so far the contents of the A$ variable pocket haven't changed. It's still got "NUT" in it. But try this:

```
7Ø A$ (LEN(A$)+1) = "CRACKER"
8Ø POSITION 5,14: PRINT A$
9Ø END
```

Now let's LIST it then press SHIFT/CLEAR and then RUN it.

What does A$ contain now?

SAVE the program in the usual way. Let's save it under the name "SUITE".

What does the + sign in the line A$ (LEN(A$) + 1) = "CRACKER" mean? All it means is that "NUT" and "CRACKER" are put together to make up "NUTCRACKER". Then "NUTCRACKER" is put inside the A$ string variable pocket. The A$ pocket used to have "NUT" in it. But now what's inside has changed to "NUTCRACKER".
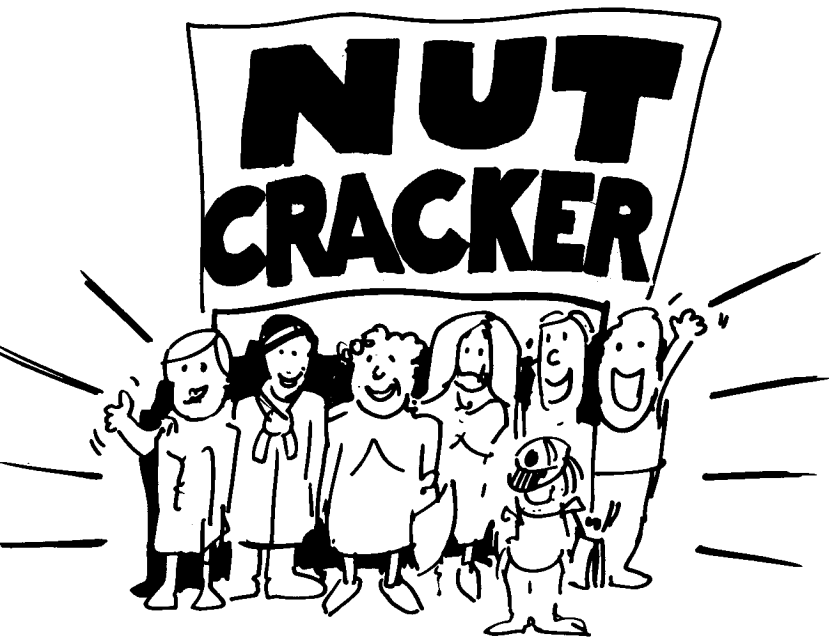
Putting two strings together to make a new string is called CONCATENATION.

Try this by typing it in directly and not as a program line:

```
PRINT "NUT";"CRACKER"
```

Now press the RESET key and type NEW and try this as a little program:

```
 5 GRAPHICS 0
10 DIM A$(8), B$(3), C$(8)
20 A$ = "ELECT"
30 B$ = "ION"
40 LET C$ (LEN(C$) + 1) = A$
45 LET C$ (LEN(C$) + 1) = B$
50 POSITION 10,12: PRINT C$
60 END
```

If you look at Lines 10, 20, 30 and 45 you will see that this time we have three string variable pockets. These pockets are called A$, B$ and C$.

Line 10 instructs the computer to prepare space for the strings. We use the instruction word DIM (dimension) with character reserve numbers in brackets.

Line 20 puts "ELECT" into pocket A$
Line 30 puts "ION" into pocket B$.

Line 40 looks and sees what's inside C$ and A$ and adds them together for a new C$

Line 45 then looks and sees what's inside C$ and B$ and adds them together for a new C$. It copies "ELECT" and "ION" and puts them together to make "ELECTION". It then puts "ELECTION" into pocket C$.

Then, at last, Line 50 shows us, on the screen, what pocket C$ has got inside it.

# IF YOU DON'T TELL IT WHAT YOU MEAN, THE ATARI WON'T KNOW WHAT YOU MEAN
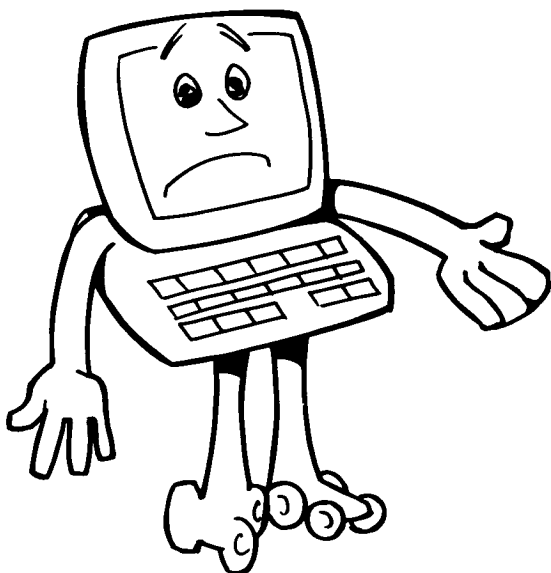
Just to see what happens, first type NEW and then type directly:

```
PRINT CASH$
```

The Atari doesn't understand you because you haven't first dimensioned the string and told it to create the string variable pocket CASH$. You have to first type in something like Line 1Ø in the program above. So you would be alright if you typed, for example:

```
DIM CASH$(4)
CASH$ = "CASH"
PRINT CASH$
```
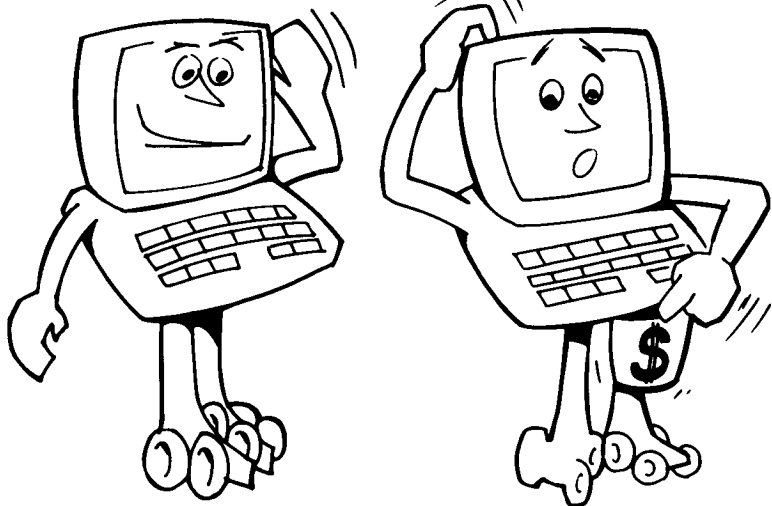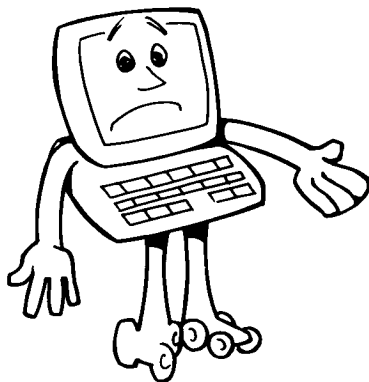
Again, just to see what happens, type this in:

PRINT CASH

The Atari treats CASH as a number and prints a 0.

Remember, to be a character string, CASH must have " marks around it. "CASH" is a character string.

To be a string variable pocket that can contain a string, CASH must have the $ sign at the end of it. CASH$ is a string variable.

So if you only type PRINT CASH the Atari thinks it should look for a NUMBER variable pocket called CASH. then prints a 0. But if we first tell the Atari that there is a CASH number variable pocket, then it will be quite happy. So type:

```
CASH = 30
PRINT CASH
```
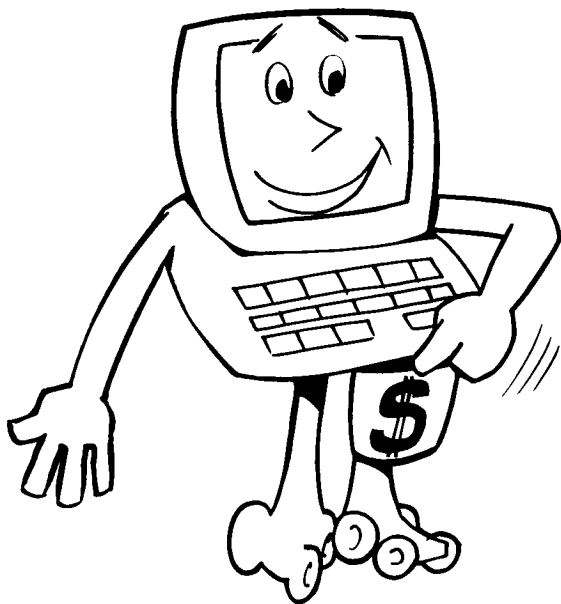


What do you see on the screen?

By the way, the messages the Atari gives you when it can't understand something you've typed in, are called ERROR MESSAGES.

# CONCATENATING STRINGS AND ADDING NUMBERS

CONCATENATION is a very, very big word. It's a big mouthful, but its meaning is very simple. It just means putting characters side-by-side. So, if you put "DON" next to "KEY", you'll get "DONKEY".
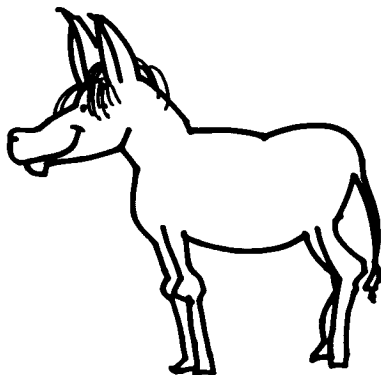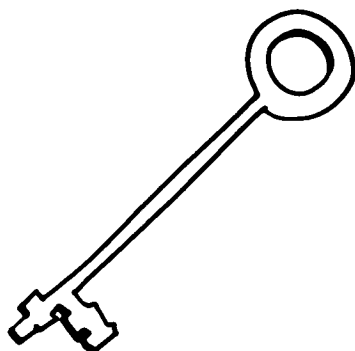
Remember that our little Atari has to be told exactly what to do. To put strings side-by-side, you have to use the LEN command with + 1 to add "KEY" on to the end of "DON". Otherwise, the Atari won't understand what you mean.

So, to tell the Atari to put "DON" and "KEY" side by side, you would have to use something like D$ = "DON". D$(LEN(D$) + 1) = "KEY". Then the D$ pocket would contain the character string "DONKEY" inside it.

Of course, if you wanted the Atari to show what it had inside the D$ pocket, you would have to tell it to PRINT D$.

Now, here's something interesting. Type in:

```
PRINT 3 + 4
```

The answer you see is 7. Here, the + sign doesn't put the 3 and the 4 side-by-side to give 34. It gives 7 instead.

Why is that? Well, because there are no " marks around the 3 and the 7, the Atari knows that it is looking at NUMBERS and not at strings. It ADDS the numbers 3 and 4 to give 7.

Ahah! Can you guess what would happen if you put the " marks around the 3 and the 4 ? Why not try it and see. Type:

```
PRINT "3" ; "4"
```

Don't forget to press RETURN!

CUTTING THE STRING

I sincerely hope you SAVED "SUITE" on tape earlier. If you did, you can LOAD "SUITE" now in the usual way.
  Here's SUITE:

```
 5 GRAPHICS 0
10 DIM A$ (10)
20 SETCOLOR 2,12,14
30 SETCOLOR 4,10,14
40 A$ = "NUT"
50 POSITION 5,10: PRINT A$
60 POSITION 5,12: PRINT "CRACKER"
70 A$ (LEN(A$) + 1) = "CRACKER"
80 POSITION 5,14: PRINT A$
90 END
```

Take a piece of paper and write on it: NUTCRACKER.
  Now, start from the left, from the letter N. Count one-two-three until you come to the letter T. What have you got? That's right, NUT.
  If you cut the paper just after T, you'll get two pieces of paper. The left piece of paper will have NUT on it.
  (By the way, be careful if you use scissors. Ask someone older before you use scissors. Sharp things are dangerous.)
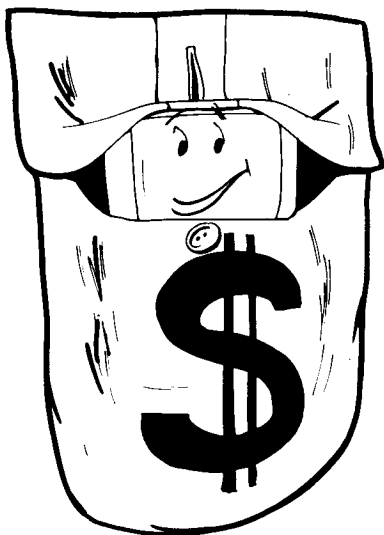 We can tell the Atari to do the same thing with character strings. We can tell it to cut up a string in the way we cut-up NUTCRACKER from the left.

Add these lines to the program SUITE:

85 L\$ = A\$ (1,3)
86 POSITION 5,16: PRINT L\$

Now edit line 1Ø and add L\$(3) to that line.
   Let's look at line 85. The Atari looks
inside the A\$ pocket and sees
"NUTCRACKER" inside it. Then it counts 3
starting from the leftmost character, N. It
gets NUT after counting 3 characters
forwards, starting from N. It then puts the

string "NUT" into the string variable pocket L$.

Then line 86 shows us, on the screen, what's inside L$.

Just for fun, try counting from the right with the word DONKEY.

Write DONKEY on a piece of paper. Start from the right, from the letter Y. Go backwards, counting one-two-three. You come to the letter K. What have you got? That's right, KEY.

If you cut the paper just before K, you'll get two pieces. The right piece will have KEY on it.

Now edit line 10 again, and add R$(7). Let's add Lines 87 and 88 to SUITE:

```
87 L$ = A$(4,10)
88 POSITION 5,20: PRINT R$
```

Let's look at line 87. The Atari looks inside the A$ pocket and sees the character string "NUTCRACKER" there. Then it counts four characters forewards, starting from N. It sees 7 characters in the DIM statement for R$ and gets the string "CRACKER". It then puts the string "CRACKER" inside the string variable pocket R$.

Line 88 shows us, on the screen, what's inside R$.

The program SUITE has changed quite a lot, so why not SAVE it again? Call it SUITE1 if you like.

WHAT MUST I DO, IF . . . .

Remember our list of things to do at the weekend?

```
10 PLAY FOOTBALL
20 READ A BOOK
30 WATCH TELEVISION
40 HAVE DINNER
50 PLAY WITH THE ATARI
60 END
```

Let's add Line 15:

```
15 HAVE A SOFT DRINK
```

Now, when you read your list of things to do, you'll have a soft drink after playing football. But, maybe sometimes you're not thirsty! Then you'll only really want a soft drink if you're thirsty.
Let's change Line 15 to:

```
15 IF I'M THIRSTY, THEN
   I'LL HAVE A SOFT DRINK
```

Before, you would always have a soft drink. Now, because you've used IF and THEN, you'll have a soft drink only if you're thirsty.
IF and THEN, you won't be surprised to learn, are words the Atari can understand.

RICH MAN, POOR MAN

Imagine you start off with 2 pounds in your CASH pocket. Suppose some very generous person keeps giving you 5 pounds to put inside your CASH pocket. Suppose he does this 1Ø times. As soon as your CASH pocket has more than 2Ø pounds inside it, you shout, "I'M RICH! I'M RICH!"

Let's write this out carefully:
MY CASH POCKET HAS 2 POUNDS IN IT.
Another pocket has 10 in it, to show how many times the generous person will give me money to add to CASH. I'll call this pocket KOUNT.

I'll keep adding 5 to CASH. I'll do this 10 times, because KOUNT contains 10.

IF CASH becomes more than 20, I'll shout, "I'M RICH! I'M RICH!"

But anyway, whether I've got more than 20 pounds or not, I'll tell you what's inside my CASH pocket.

Of course, I'll keep checking to see whether I've added money 10 times. Now let's try to type in a program the Atari will understand:

```
10 SETCOLOR 2,15,14
15 SETCOLOR 4,9,14
20 CASH = 2
30 FOR KOUNT = 1 TO 10
40 CASH = CASH + 5
50 IF CASH > 20 THEN PRINT "I'M RICH!
   I'M RICH!"
60 PRINT "CASH = " ; CASH
70 NEXT KOUNT
80 END
```

Let's look at line 50. Can you guess what the > sign in IF CASH > 20 means? That's right, the > sign simply means *more than.*

(There's a < sign as well, just next to the *more than* sign. It means *less than.*)

RUN the program.

How much money does the CASH pocket end up containing?

Well, CASH ends up with 52 in it.

What's the use of IF? It's very useful. It allows you to do something only if there's a reason. It allows you to make a decision. IF you're thirsty, then you decide to have a soft drink. You'll only follow Line 15 of your weekend list if you have the right reason to do so.

Similarly, because of the IF in Line 50, you'll only shout that you're rich when you have more than 20 pounds in your pocket.

To see this, change Line 40 to:

```
40 CASH = CASH + 1
```

Now, the CASH pocket is being increased by 1 each time, instead of 5.

RUN the program now and see what happens.

BUT ONLY UNTIL....

Now suppose the generous person didn't want to be *too* generous.

Suppose he only wanted to repeat giving you 5 pounds until your CASH pocket had more than 20 pounds in it. Then he would stop.

Just imagine it for yourself. You've got 2 pounds in your CASH pocket. He gives you 5 pounds. You add this to the 2 in your CASH pocket. Have you got more than 20 pounds now? No. So he gives you another 5 pounds. And so on. He will repeat what he's doing. But only until you have more than 20 pounds.

Let's write a little program using IF and THEN. Remember to use again the RESET key and NEW to show the Atari that you're writing a new program.
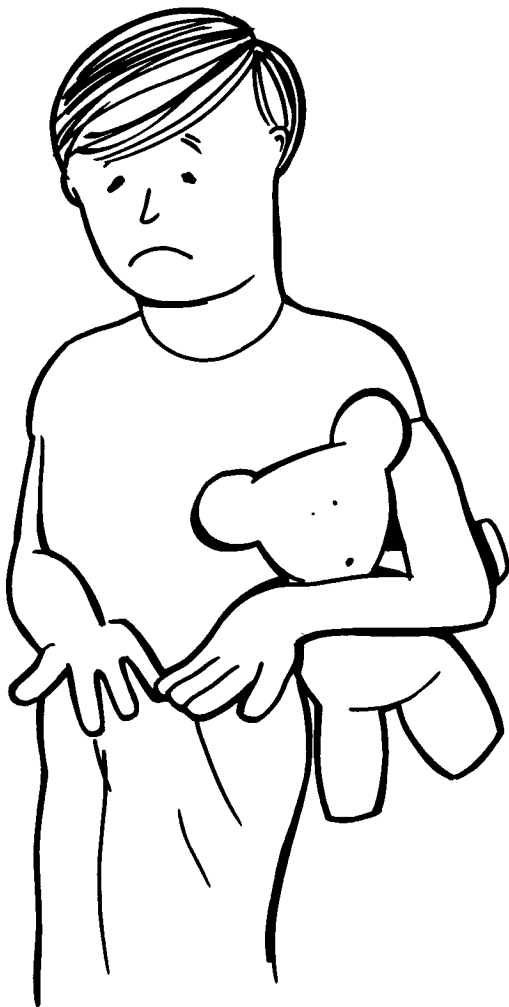
Type in the following program. You'll see big gaps in it, but don't try to copy the gaps. Just type the lines in as usual, one after the other. Of course, when you finish typing a line, you have to use RETURN to let the Atari know. The command in line 10 clears the screen.

```
10 PRINT CHR$ (125)
20 SETCOLOR 2,14,14
30 SETCOLOR 4,10,14

40 PRINT "I'M POOR!"
```

```
 50 CASH = 2
 80 CASH = CASH + 5
 90 PRINT "CASH = "; CASH
100 IF CASH < 20 THEN GOTO 80
```

```
110 PRINT
120 PRINT
130 PRINT "YOU'RE RICH."
140 PRINT "SO NO MORE MONEY
    FOR YOU!"
```

```
150 END
```


Look at Lines 110 and 120 of the program.
These program lines just print out blank
lines. The blank lines separate the "YOU'RE
RICH" message from the other messages on
the screen. RUN the program and see. Then,
if you like, leave out Lines 110 and 120 and
see what happens when you now RUN the
program.

Remember how to get rid of a complete
line? Just type its number again and then
press RETURN.

By the way, I'm sure you've noticed that:
FOR always goes together with NEXT.
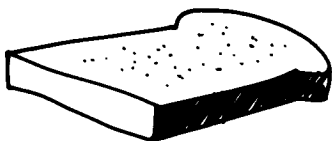IF always goes together with THEN.

They go together like the two slices of a
sandwich! And, just as a sandwich usually
has something in between its two slices,
they usually have something between them!

FOR

NEXT

IF

THEN

When you typed the last program you might have noticed the funny word GOTO. Your Atari usually does things in the order of its line numbers, but sometimes you might want it to jump around in a different order. This word does that and says "when you have done that go to somewhere else." So, GOTO 80 in line 100 means if you have less than 20 pounds go to line 80 again.
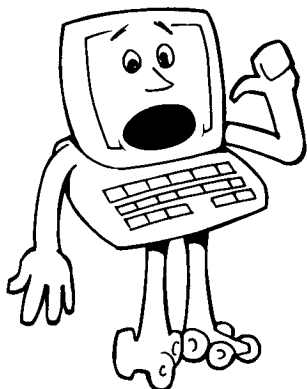
THE ATARI
EXPECTS....
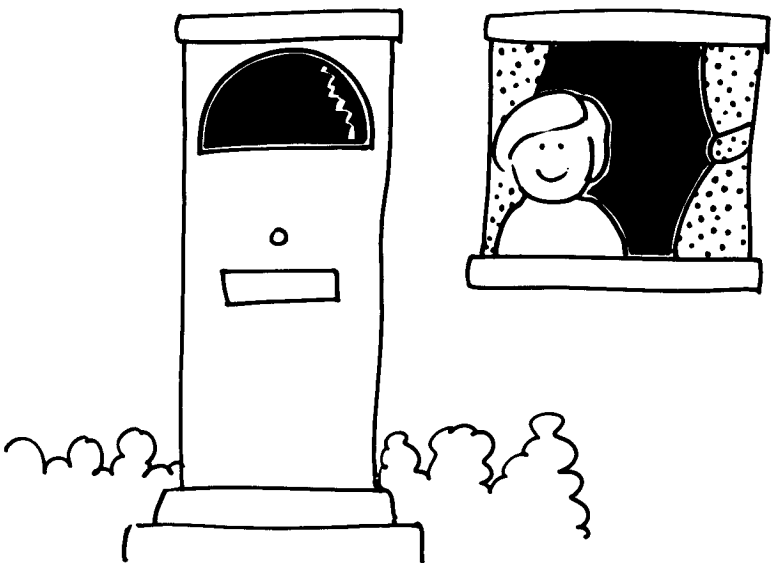
If you're expecting a letter on your birthday,
maybe you'll get up early. Then maybe you'll
stand by the door and wait for the postman
to put a letter through the letter box.

You can also make the Atari wait and expect something to be put into one of its memory pockets.

What word will the Atari need to INPUT something into a pocket?

The word INPUT, of course.

The Atari can wait and expect you to give it a NUMBER to put into a number variable pocket. Or the Atari can wait and expect you to put a character STRING variable into a string variable pocket.

How will it know what sort of variable to input into a pocket?

Well, once again, it all has to do with whether there is a $ sign to tell the Atari to expect a character string variable.

For example, if you tell the Atari to INPUT N$, it will think that what you type in is a character string. It will put this string into the string variable pocket N$.

But if you tell the Atari to INPUT N, it will think that what you type in is a number. It will put this number into the number variable pocket N.

THINK OF A NUMBER!

Now press the RESET key and type in NEW,
then type this little new program:

```
10 PRINT CHR$ (125)
20 SETCOLOR 2,8,14
30 POSITION 3,2: PRINT "TYPE IN"
40 POSITION 3,3: PRINT "YOUR FAVOURITE"
50 POSITION 3,4; PRINT "NUMBER"
60 POSITION 3,5: PRINT "FROM 1 TO 9"
70 INPUT NUM
80 POSITION 1,7: PRINT "YOUR FAVOURITE
   IS "
90 POSITION 8,10: PRINT NUM
100 NUM = NUM * 9
110 ANSWER = NUM * 12345679
120 POSITION 3,12: PRINT "HERE'S A
    LOT OF"
130 SETCOLOR 4,12,14
140 POSITION 6,14: PRINT ANSWER
150 SETCOLOR 2,6,14
160 POSITION 8,16: PRINT "!!!!"
170 END
```

Run the program. Now LIST it.
Let's look at Line 70. If you had typed
NUM = 8 at Line 70, the number 8 would
already be in the NUM number variable
pocket. You'd be stuck with it, and you
wouldn't be able to choose a number from 1
to 9.
But, because Line 70 has the word INPUT,
the Atari waits for you to type in a number

to put into its NUM pocket.

To show you that it's expecting an input from you, the Atari shows a ? on the screen.

Let's look at Line 100. Here, what's inside the NUM pocket will be multiplied by 9.
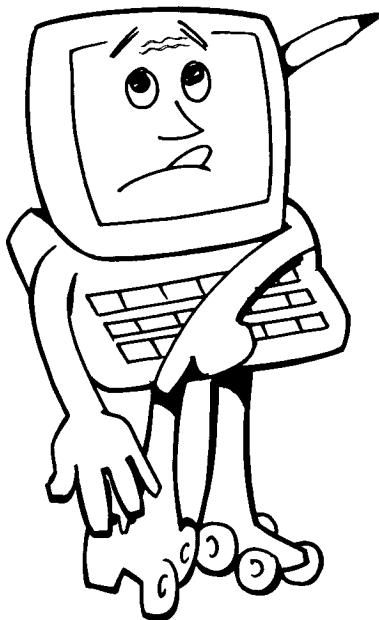
Suppose you type in 4 as your favourite number when you see the ? mark. The NUM pocket will then have 4 inside it. After the Atari does what Line 100 says, the NUM pocket will have 36 inside it.

Let's look at Line 110. Here, NUM is multiplied by a very large number. We don't have to worry, though, because the Atari will do this big job for us. It will then put the new number into the ANSWER pocket.

By now, you should understand what the other lines of the program do. But, let's just one more time go through the program line-by-line. LIST the program. Then look at the listing on the screen as you read what follows.

Line 10 tells the Atari to clear the screen.

Line 20 gives BACKGROUND colour number of 8.

Line 30 uses POSITION to print a part of a message. POSITION dances 3 across and 2 down the screen.

Line 40 uses POSITION for another part of the message. It dances 3 across and 3 lines down.

Line 50 uses POSITION for another part of the message. It dances 3 across and 4 down.

Line 60 uses POSITION for the last part of the message. It dances 3 across and 5 down.

Line 70 shows a ? mark and makes the Atari wait for a number to be typed in. It puts it into the number variable NUM.
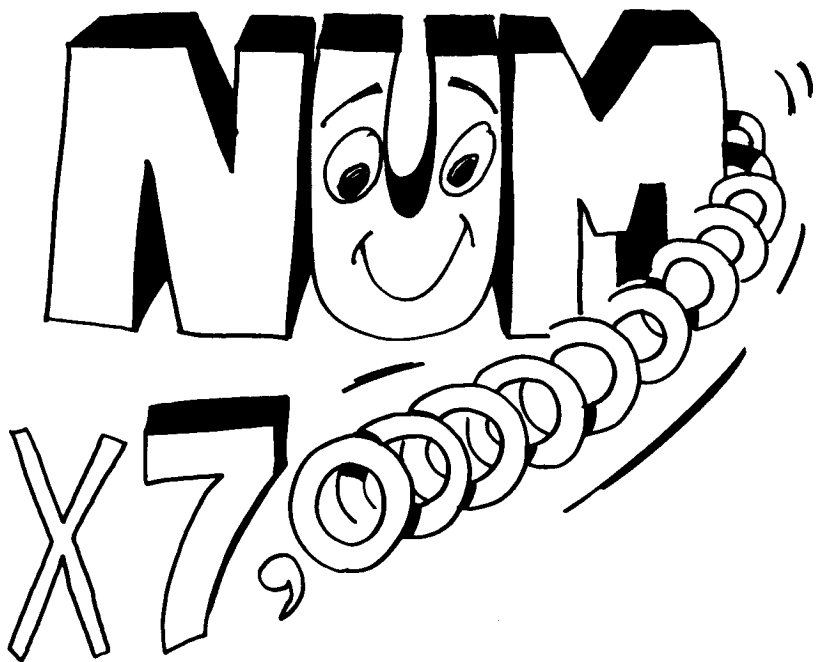
Line 80 uses POSITION to show another message on the screen.

We use different numbers for POSITION because we want to place things in different places on the screen.

Line 90 uses PRINT and POSITION to show us, on a part of the screen, what NUM contains at this moment.

Line 100 multiplies what NUM contains by 9. So now NUM has changed to containing 9 times what it did before.

Line 110 multiplies NUM by a large number. We don't have to bother about how, because the Atari does that for us. It puts this number into the number variable ANSWER.

Line 120 places another message on the screen, using POSITION.

Line 130 changes the BORDER colour number to 12.

Line 140 uses PRINT and POSITION to show what's inside ANSWER, on a part of the screen.

Line 150 changes the background colour number to 6.

Line 160 just prints four ! signs on a part of the screen, using POSITION.

HOW REMARKABLE!

We've gone through all the lines of this program but, of course, we will get to know things better and need explanations less.

If we want to put in little explanations inside a program itself, so that the program can be understood by someone else, we can use REM. As you can guess, REM simply tells the Atari that what follows it on a line is a remark. A remark just tells you something about a part of a program. The Atari does nothing when it comes to the REM while it is RUNning a program. The REM line is only shown when you LIST the program.

For example, we could add Line 145, just before Line 15Ø:

        145 REM NEW BACKGROUND
            COLOUR

We can add other REM lines in the same way, if we want to explain some important part of a program. This is really useful in a long, long program, which could be hard to read and understand if it isn't explained a little at important points.

But, remember, too many REM lines can also make a program difficult to read. You have to use your common sense.

For example, one nice place to put a remark would be just before Line 1Ø. We could add:

5 REM FAVOURITE NUMBER PROGRAM

Here we are just reminding ourselves what the program is about. Three months later we can look at Line 5 and easily know what the program is about.
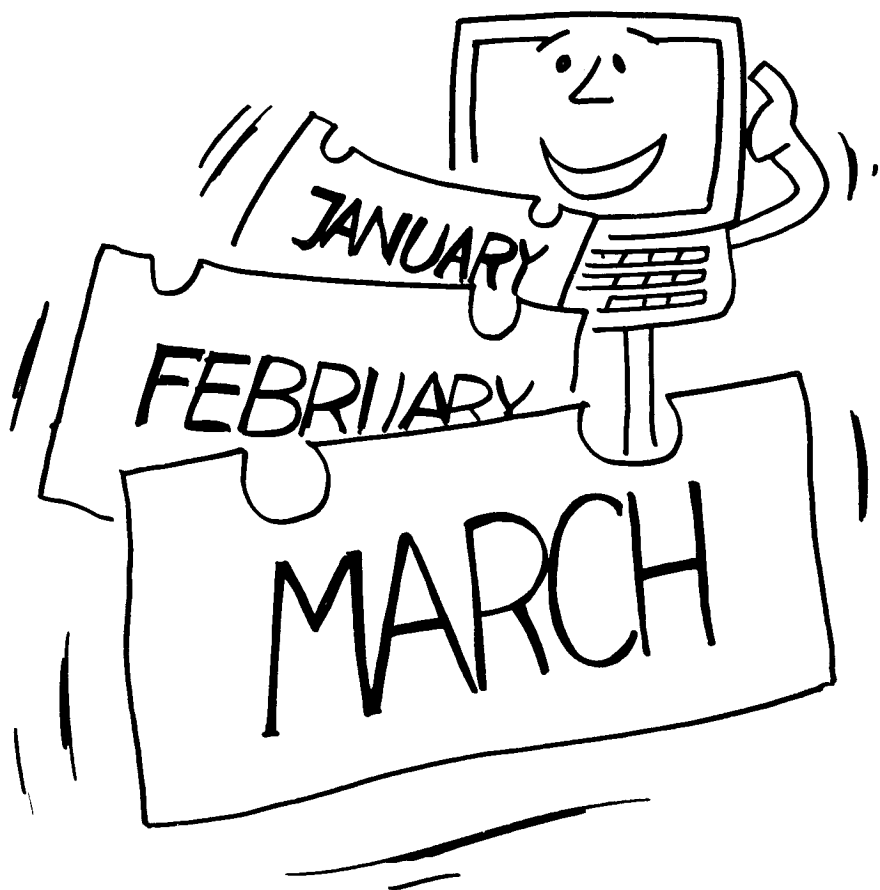
HUFF AND PUFF

I have to admit it, it's a long time since I was at school. But I seem to remember a story about three little piggies who each built a house. I can't remember the story completely, but I'm certain one of the little pigs built his house of straw and the other one built his of bricks. And there was a big bad wolf who huffed and puffed and blew down the house of straw. But he completely ran out of huff and puff when he tried to blow down the house of bricks.
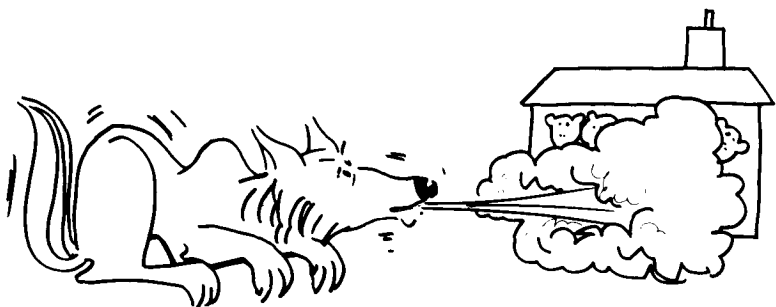
What's that got to do with programs?
A very great deal.

A house of straw is untidy. Bits of it keep falling off and you have to keep patching it. If part of a wall needs repair, you have to take out bits of good wall too, and the straw you put in is very difficult to put exactly where you want it to go.

If your house is built of bricks, then it's easy to take out a bad brick and put in a fresh one. It's easy to replace one part by another.

In the same way, you should write your programs so that they are neat, and built up of small parts. Each part should have a clear purpose. A brick has a solid, clear shape. It fits neatly with other bricks.

Is that easier to say than do? At the beginning, yes. But as you practise, it will become as easy to do as to say.

I wanted to write a game.

1. First, I wanted to PREPARE THE SCREEN.

2. Then I wanted to tell the Atari to create a string variable pocket and a number variable pocket.

3. I wanted to give some messages to the player, to TELL HER OR HIM ABOUT THE GAME.

4. I wanted to INPUT whether the player wanted to play the game or not. As soon as he wanted to stop, the game would END. Otherwise I would carry on with the game.

5. Then I wanted to give a problem to the player. I wanted him to guess whether a wolf was "huffing" or "puffing". This is really what the game is about.

6. While the player was trying to guess what the wolf was doing, I also wanted to check whether he had guessed right. Had he WON? If he won, I would do this:

I would tell him he had won and then ask him if he wanted to play again or stop.

7. I wanted to tell him how many tries he had so far. I also wanted to check whether he had more than 3 guesses. If he had, then the wolf had huffed and puffed enough times to blow his house down. So he had LOST. I would give him a message and ask him if he wanted to play again or stop.

So long as the player didn't have more than 3 goes, he could keep playing until he wanted to stop.

If you think about writing any kind of program, you'll find that it uses the same sort of program bricks:

A brick to show things on the screen at

the beginning.

A brick to create certain variables.

A brick to give messages about the program.

A brick to input information for the program to look at or to use.

A brick to create something out of information the program is given. This can be a problem for the user, as in a game. But this can instead be a solution, as when the computer adds a lot of numbers together and tells you the result on the screen.

A brick to see if the program has got a result that the user wants. In this case, has the player won?

A brick to see if the program has a result the user doesn't want. In this case, has the user lost because he's used up all his tries?

A brick to keep the user informed about what's happening. In this case, how many guesses.

Each PROGRAMMING brick is called a SUBROUTINE. Notice that we haven't actually done any programmming. But we've done something more important than writing program lines. We have planned a program to be built out of SUBROUTINE bricks.

THE HUFF-PUFF GAME

1. We'll describe or DEFINE the PREPARE SCREEN SUBROUTINE brick. Let's give it a name to remember it by — let's call it REMPREPARE. So our description can be called REMPREPARE. Here it is:

REMPREPARE

The border colour is green-blue. The background colour is pink.

RETURN

Notice the RETURN. What do you think it means? That's right, it means that we've finished the definition of a subroutine.

2. Let's DEFINE the SUBROUTINE to create the variables we need for Huff Puff. Let's call it VARIABLES.

REM VARIABLES

Ask for the player's name. That is, INPUT the name as a character string. Put it into a a string variable pocket called NAME$.

Tell the Atari to create a variable number pocket called TRIES with 0 inside it. (TRIES will be increased by 1 each time the player has a go.) TRIES will tell the player how many goes he or she has had so far.

RETURN

3. DEFINE the SUBROUTINE to TELL the player about the game.

REMTELL

Print" This is the Huff and Puff game
Print" Do I huff or do I puff?"
Print" Type HUFF or PUFF and press Return"
Print" Yours sincerely, Wolf!"

RETURN

Now we have to keep repeating the problem (which is the procedure brick number 4 below) until the player types "S" for stopping the game. Number 8 below does that. We'll keep looking at A$. So long as it is not equal to "S" when the IF line is reached, the game will be repeated. We'll also tell the player to press S to stop the game if that's what the player wants. If he wants to keep playing, we'll tell the player to press P. As soon as S is pressed, we'll end the game.

4. This is the PROBLEM procedure. Let's call it REMPROBLEM. As soon as the player tries this problem, he's having a go, so we'll increase TRIES by 1.

The problem is, is the Wolf doing a huff or a puff? How do we tell the Atari to choose whether the Wolf is doing a huff or a puff?

There is a very useful word called RND. If you have the numbers 1, 2, 3, and so on, written on separate pieces of paper and close your eyes and pick one, you are choosing a number at RANDOM. You can't be sure what number you will choose. RND tells the Atari to close its eyes and choose a number.

If you say INT(RND(1) * 12) + 1, it will choose a number from 1 to 12.
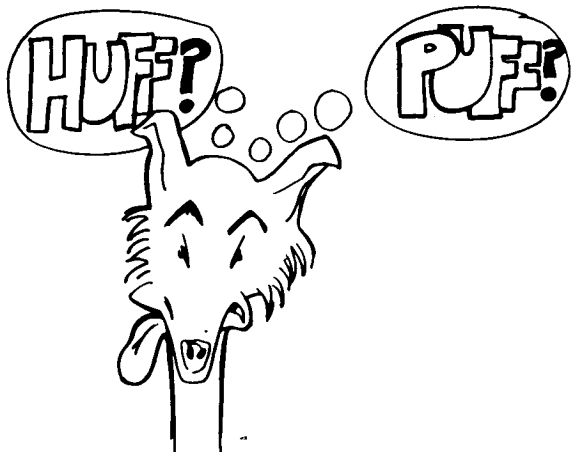
If you say INT(RND(1)*2) + 1 it will choose 1 or 2. There we have it. INT(RND(1)*2) + 1. If the number that comes up is 1, then we'll put "HUFF" into a string variable pocket R$.

If the RND function chooses 2, then we'll put "PUFF" into R$.

REMPROBLEM

First, add 1 to the number variable TRIES. So, TRIES = TRIES + 1.

Make a number variable equal to INT(RND(1)*2) + 1. Call this number variable R. So R = INT(RND(1)*2) + 1.

Now if R = 1, then we say R$ = "HUFF". If R = 2 then R$ = "PUFF".

Now we'll ask the player:" Do I huff or do I puff?"

What has the player typed in? We tell the Atari to expect HUFF or PUFF. We'll say something like, INPUT B$. If it is "HUFF", we'll check R$ to see if it's equal to B$. If it is, we'll tell the player he's won, using the

87

WON subroutine. If R$ is not equal to B$, we'll just carry on with telling the player how many goes he or she has had. We'll do the same kind of thing if the player has typed in "PUFF". If the player types in something totally different, he just loses another go.

   RETURN

   5. The WON subroutine prints the player's name, says he has won, and gives a message like "I can't blow your house down. Yours disappointedly, Wolf." Then, ask the player if he wants to play or stop.
   6. REM GOES subroutine. We tell the player how many goes he has. If it's more than 3, we do a subroutine that we can call the HUFFPUFF subroutine. Otherwise, we repeat the problem.
   7. REM HUFFPUFF can print out messages like "it's your try number four, your house is made of straw. I've huffed and puffed and blown your house down. Yours in anticipation, Wolf!" Then we ask the player if he wants to play or stop.
   If the player hasn't said he wants to finish, or hasn't won, or hasn't used up all his goes, this means that A$ is certainly not "S". So the game is repeated from number 4 above.
   8. REM PLAY or STOP subroutine asks the player if he wants to play or stop.
   I've left it up to you to make items 5, 6, 7, and 8 more like subroutines.

PROCEED WITH YOUR PROGRAMS!

Let's make a list of the subroutines, together with other bits of the Huff and Puff game:

```
10 PRINT CHR$ (125)
15 DIM NAME$ (15), R$(4), B$(5), A$(6)
20 GOSUB 140
30 GOSUB 190
40 GOSUB  240
100 GOSUB 340
110 GOSUB 550
130 GOTO 100
```

That's the whole of the Huff Puff game, written out as a list of Subroutines. Each time the Atari comes TO A GOSUB it goes to that line and returns when it finds RETURN command.

So, when the Atari COMES TO LINE 20 and sees GOSUB 140, it looks for this:

```
140 REMPREPARE
150 SETCOLOR 2,3,14
160 SETCOLOR 4,10,14
170 PRINT CHR$ (125) : SETCOLOR 1,0,0
180 RETURN
```

So now let's carry on with all the definitions of the subroutines.

```
190 REM  VARIABLES
200 PRINT "WHAT IS YOUR NAME?"
210 INPUT NAME$
```

```
220 TRIES = 0
230 RETURN

240 REMTELL
250 PRINT "THIS IS THE HUFF AND PUFF
    GAME"
260 PRINT "TYPE HUFF IF YOU THINK I'M
    HUFFING"
270 PRINT"TYPE PUFF IF YOU THINK I'M
    PUFFING"
280 PRINT "THEN PRESS RETURN"
290 PRINT "YOURS SINCERELY,WOLF"
330 RETURN

340 REM PROBLEM
350 TRIES = TRIES + 1
360 R = INT(RND(1)*2) + 1
370 IF R = 1 THEN R$ = "HUFF"
380 IF R = 2 THEN R$ = "PUFF"
390 PRINT "DO I HUFF?"
400 PRINT "DO I PUFF?"
410 INPUT B$
420 IF B$ = R$ THEN GOTO 460
440 RETURN

450 REM WON
460 PRINT NAME$
465 SETCOLOR 2,13,14
470 PRINT "YOU WON!"
480 PRINT "I CAN'T BLOW YOUR HOUSE
    DOWN!"
490 PRINT "YOURS IN DISAPPOINTMENT"
500 SETCOLOR 1,0,0
510 PRINT "WOLF"
520 GOTO 850
540 RETURN
```

```
550 REM GOES
560 PRINT "YOUR NUMBER OF TRIES
    SO FAR IS"
570 PRINT TRIES
580 IF TRIES > 3 THEN GOTO 610
590 RETURN

600 REM HUFFPUFF
610 SETCOLOR 2,5,14
620 PRINT CHR$ (125)
630 PRINT "YOUR TRY NUMBER FOUR"
640 PRINT "MEANS YOUR HOUSE IS OF
    STRAW"
650 PRINT "I'VE HUFFED AND I'VE
    PUFFED AND BLOWN YOUR HOUSE
    DOWN"
660 PRINT "YOURS IN ANTICIPATION"
680 PRINT "WOLF!"
700 GOTO 850
840 REM PLAY OR STOP
850 PRINT "TYPE S TO STOP"
860 PRINT "TYPE P TO PLAY"
870 INPUT A$
880 IF A$ = "P" THEN 20
890 IF A$ = "S" THEN END
```
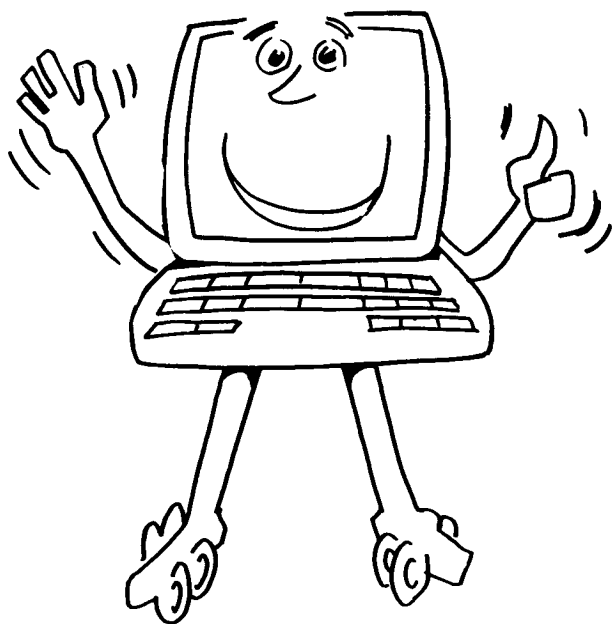
Type the program in. If you want to list it,
it's a useful trick type LIST 190, 340 the
computer would only list lines between line
190 and line 340 inclusive. You'll see part of
the listing on the screen. To get more, type
LIST and another range of line numbers.

If you run the program, you'll notice that
the screen doesn't look very neat. It's
important that people can read what's on
the screen easily. So work on this program
to make it look better. Use everything you
have learnt.

If you get stuck, just ask somebody who knows a bit of programming to help you. There are many things I don't understand, and I often ask other people for help. But first I try to do the things myself. So try improving this program yourself first. If you practise whenever you can, you will soon be able to write your own programs!

INDEX

DUCKWORTH
HOME COMPUTING

All books written by Peter Gerrard, former editor of *Commodore Computing International,* author of two top-selling adventure games for the Commodore 64, or by Kevin Bergin. Both are regular contributors to *Personal Computer News, Which Micro? and Software Review* and *Popular Computing Weekly.*

EXPLORING ADVENTURES ON THE ATARI 48K
by Peter Gerrard
£6.95

This is a complete look at the fabulous world of Adventure Games for the Atari Computer. Starting with an introduction to adventures, and their early history, it takes you gently through the basic programming necessary on the Atari before you can start writing your own games.

Inputting information, room mapping, movement, vocabulary – everything required to write an adventure game is explored in detail. There follow a number of adventure scenarios, just to get you started, and finally three complete listings written specially for the Atari, which will send you off into wonderful worlds where almost anything can happen.

The three games listed in this book are available on one cassette.

Other titles in the series include *Sprites & Sound on the 64, 12 Simple Electronic Projects for the VIC, Will You Still Love Me When I'm 64, Advanced Basic & Machine Code Programming on the VIC, Advanced Basic & Machine Code Programming on the 64,* as well as *Pocket Handbooks for the VIC, 64, Dragon, Spectrum* and *BBC Model B.*

Write in for a catalogue.

DUCKWORTH
The Old Piano Factory, 43 Gloucester Crescent, London NW1
Tel: 01-485 3484

Duckworth Home Computing

MY ATARI XL AND ME
Jack Walker

Playing with a computer should be fun as well as instructive. The purpose of this book is to help both children and parents to understand how the Atari XL computer works and what it can do.

The book adopts a simple, friendly style with emphasis on learning as fun. A major feature is the use of short, self-contained sections which encourage children and parents to work at a steady, unhurried pace. An hour a day with book and computer should enable a child to grasp the simple but powerful ideas behind programming. This is not a text for learning BASIC but rather a way to get to know the Atari XL using simple BASIC commands. Children as young as 6 or 7 should be able to understand the text and operate the computer with or without parental help.

Jack Walker has written and edited numerous computer books. He runs his own editorial and marketing company, J.S.W. Publishing Services.

ISBN 0-7156-1963-2

9 780715 619636