WRITE YOUR OWN PROGRAM

CREATING A DATABASE

ADVENTURE GAME

WRITE YOUR OWN PROGRAM

CREATING A DATABASE

ADVENTURE GAME

You are in a corridor with a strong
current going west. Your way is blocked
by the arms of a huge rainbow colored
octopus.

Steve Rodgers and Marcus Milton

Foreword

Adventure games are played in the world of your imagination. They are like a story and can take any form you like. You can be the last survivor of a space expedition marooned on a distant planet, or shrunk in size and trapped within the circuits of an alien computer. The game explained in this book takes place in an undersea world with sharks, sea horses and the bleached bones of previous explorers.

In computer terms, an adventure game requires a large *database* to store all the descriptions used in the game. A database is a collection of information which the computer retrieves and acts on according to the instructions within the program. Since the player uses verbs and nouns to describe the actions he or she wants to take, the game relies on the computer's ability to recognize and process these accordingly. Text data is processed as *strings* of characters, and adventure games of this sort rely on the way that strings and parts of strings can be identified and operated upon. In this book, the adventure game has been divided into two sections. The first section gives the main program and the data statements. The second section explains how the verbs which the player uses, call up the relevant sections of program and direct the progress of the game.

Contents

```
20  D=18000:S=400:H=2000:F=5
30  C$="          "
40  PRINT  TAB(2,1)CHR$129"HEIGHT    "CHR$132"DESCENT    "
    CHR$131"SPEED    "CHR$133"DISTANCE"
50  PRINT  TAB(35,23)CHR$146CHR$255CHR$255CHR$255
60  X=0:Y=4
70  PRINT  TAB(X,Y)CHR$150CHR$255CHR$255CHR$255
80  PRINT  TAB(3,2)C$
90  PRINT  TAB(3,2)CHR$129;H
100 PRINT  TAB(13,2)C$
110 PRINT  TAB(13,2)CHR$132;F
120 PRINT  TAB(22,2)C$
130 PRINT  TAB(22,2)CHR$131;S
140 PRINT  TAB(30,2)C$
150 PRINT  TAB(30,2)
160 IF
```

The plan

The first stage of writing an adventure game is to plan the
territory which the player will explore. This is usually a
series of "rooms" or "locations." In our game, there are 24
rooms arranged on a four by six grid. Each room has a door
or doors leading to other rooms, and the player is told which
exits are available as the game is played. In some rooms
there are objects which may help later on in the game and in
others there are obstacles which have to be overcome. The
object of the game is to replace the plug in the plughole in
room 19, but first you must pass the octopus in room 20.

To play the game, you must give the computer instructions, using a number of verbs and nouns that the computer understands. You can move from one room to another by typing 'GO NORTH" (or any other of the four directions). The verb "LOOK" is useful, because it may reveal things in the room that are not apparent at first sight. If you type "HELP" at any stage, you will be shown a list of the verbs you can use, in case you've forgotten them. To pick up any of the objects that you encounter on the way, type "GET KNIFE" (or any other object). When planning your game, it's a good idea to play it on paper first, to make sure that it works.

5 SEAWEED COVERED WINDOW

6 FLIPPERS

WHITE CLIFF

11 BLUE HANDCUFFS

12

17 RED HANDCUFFS

18

23

24

EAST

LIST

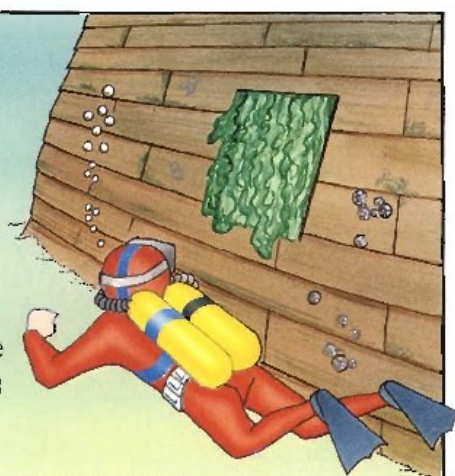| NOUNS	VERBS	OBJECTS
NORTH	GO	KNIFE
SOUTH	GET	PAIR OF FLIPPERS
EAST	DROP	
WEST	CUT	KEY
CHEST	WEAR	CLUMP OF SEAWEED
WINDOW	GIVE	
KNIFE	UNLOCK	ROTTEN OLD BONE
FLIPPERS	USE	
KEY	INVENTORY	MAGIC PLUG
SEAWEED	HELP	YELLOW PAIR OF HANDCUFFS
BONE	LOOK	
PLUG		GREEN PAIR OF HANDCUFFS
HANDCUFFS		RED PAIR OF HANDCUFFS
		BLUE PAIR OF HANDCUFFS

The data statements

APPLE IIe

Type in the **DATA** on the opposite page, ignoring the black marker lines. You should *format* the **DATA** at this point so that the words are not illegibly broken in two when printed on the screen. The best way to begin is to count the number of characters you have typed within the inverted commas. The Apple will allow a maximum of 40 characters across the screen; if you find that a word would be broken in two at the fortieth character, then type blank spaces up to the end of the line and begin the next word at the 41st character.

In text only games like this, you have to rely on your imagination to set the scenes, so your adventure will be more fun to play if your location **DATA** is long and descriptive. Your **DATA** statements can be much longer than ours given opposite, but you must remember that the number of characters that you use will depend upon the particular limitations set by your computer. Your **DATA** statement could read something like this: 6050 **DATA** "You are standing on the seabed. In front of you, to the south, there is an unnaturally square shaped patch of seaweed that clings to the barnacled hull of an ancient pirate ship."

COMMODORE 64

The Commodore 64 will not allow you to type more than 80 characters to a program line. Therefore, to enable you to use the **DATA** on the opposite page you must split each line into two lines of **DATA**. This is best illustrated by the following example. . .

Type in the **DATA** at line 6010 as far as the black marker line, and end it at this point by typing a double quote ("); then press the **RETURN** key.

Start a new line number and continue with the **DATA** after the black marker, to give line 6011 **DATA** "BLOCKED BY A HIGH CORAL REEF". Commodore users should treat the rest of the **DATA** in the same way, ending at line 6201, which should read: 6201 **DATA** "FACE THE JAWS OF A GREAT WHITE SHARK". Later on in the book, you will see that the various room descriptions are **READ** from the **DATA** in pairs (labeled **T$** and **D$**) and added together in line 5150 to make a complete description.
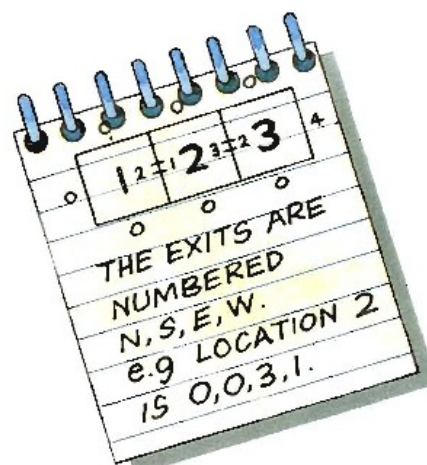
At the end of this book, the complete program listing is given in full. If you turn to page 41, you'll find the Commodore **DATA** split into its appropriate line numbers.

```
6000  REM   **********DESCRIPTIONS**********
6010  DATA  "YOU ARE ON THE SEABED.THE WAY WEST IS|
BLOCKED BY A HIGH CORAL REEF."
6020  DATA  "YOU ARE ON THE SEABED.TO THE SOUTH A|
BARNACLED WALL TOWERS ABOVE YOU."
6030  DATA  "YOU ARE IN FRONT OF A WOODEN DOOR. YOU|
CAN SEE NO HANDLE."
6040  DATA  "YOU ARE ON THE SEABED.TO THE SOUTH A|
BARNACLED WALL TOWERS ABOVE YOU."
6050  DATA  "YOU ARE ON THE SEABED. TO THE SOUTH A|
BARNACLED WALL HAS A SQUARE PATCH OF SEAWEED GROWING
ON IT."
6060  DATA  "YOU ARE ON THE SEABED,TO THE SOUTH|IS A
BARNACLED WALL.A CLIFF BLOCKS THE WAY EAST."
6070  DATA  "YOU ARE IN A LONG, LOW CAVERN. AT THE
FAR|END A LARGE BONEFISH IS SWIMMING AROUND."
6080  DATA  "YOU ARE IN A BRIGHTLY LIT CHAMBER.THE|
WALLS,FLOOR AND ROOF GLOW IN SHIMMERING LIGHT."
6090  DATA  "YOU ARE IN A DIMLY LIT CAVERN WITH A|
HUGE DOOR AT THE FAR END.YOU CAN SEE NO HANDLE."
6100  DATA  "YOU ARE IN A ROOM FULL OF HUNGRY|
SEAHORSES! THEY NUZZLE YOUR HAND IN A FRIENDLY MANNER"
6110  DATA    "YOU ARE IN A SMALL ROOM.THE NORTH WALL|
   HAS A SMALL WINDOW IN IT THROUGH WHICH YOU CAN SEE
   THE SEABED"
6120  DATA    "YOU ARE IN AN AM-A-Z-NGLY SQUARE ROOM.|
THE WALLS,FLOOR AND ROOF ARE ALL SQUARE AS ARE ALL THE
   EXITS"
6130  DATA    "YOU ARE IN A TINY LITTLE ROOM THAT IS|
   OCCUPIED BY A CHEST INSCRIBED WITH THE INITIALS D.J."
6140  DATA   "YOU ARE IN A COLD,MURKY ROOM,GREY MUD|
   SWIRLS AROUND YOU AND YOU FEEL A FAINT CURRENT EAST"
6150  DATA    "YOU ARE IN A GLOOMY AND EERIE PLACE,ALL|
AROUND YOU ARE THE BONES OF LONG DEAD EXPLORERS!"
6160  DATA    "YOU ARE IN A SQUARE ROOM.THE SOUTH EXIT
   HAS THE WORDS 'DO NOT ENTER'|ABOVE IT.THE NORTH AND
   WEST DOORWAYS ARE CRAWLING WITH SEA URCHINS"
6170  DATA    "YOU ARE IN A CIRCULAR ROOM WITH A VERY
   STRONG CURRENT THAT SWIRLS|AROUND THE ROOM AND DOWN A
   HOLE IN THE FLOOR"
6180  DATA "YOU ARE IN A CORRIDOR WITH A STRONG
CURRENT GOING WEST.YOUR WAY|IS BLOCKED BY THE ARMS OF
A HUGE RAINBOW COLORED OCTOPUS."
6190  DATA    "YOU'RE IN A SHIPWRECKED CAPTAIN'S CABIN
   AND FEEL THE FLOW OF WATER TO THE WEST"
6200  DATA  "YOU SEE A RUSH OF SWIRLING WATER AND|FACE
   THE JAWS OF A GREAT WHITE SHARK."
```

Data statements continued

Lines 6210 to 6270 store the exits initially available from each location as a series of four numbers in the order north, south, east and west. If a number is zero, there is no exit in the corresponding direction. For example, location 8 (the BRIGHT CHAMBER) has exits 0,14,0,7. This means that there is no exit from this location going north or east, but south leads into room 14 and west leads to room 7. The following lines contain a list of every object that the player will be able to pick up. After each object description is a number which represents the room number in which the object is found. If the number is zero, then the object will only appear later on in the game. If the number is 99, then the player is carrying the object.

Line 6510 is a list of the nouns that the computer will recognize. These are the directions and the names of objects. Only the first three letters are stored, since this is how the program will use them. Finally, line 6530 stores the verbs that the computer will recognize. They are stored in their entirety since they will be required to appear on the screen when the player types HELP.
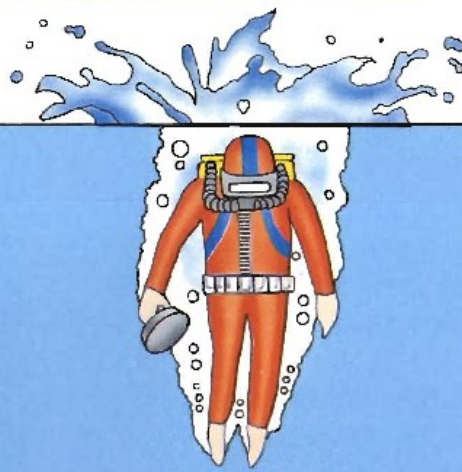
```
6210   REM *************EXITS*********************
6220   DATA  0,0,7,0,  0,0,5,1,  0,0,4,2,  0,0,5,3
6230   DATA  0,0,6,4,  0,0,0,5,  0,0,8,0,  0,14,0,7
6240   DATA  0,15,0,0,  0,16,0,0,  5,17,12,0,  0,18,0,11
6250   DATA  7,0,0,0,  0,0,15,0,  7,21,16,14,  10,22,17,15
6260   DATA  11,23,18,16,  12,24,0,17,  0,0,20,0,  0,0,21,0
6270   DATA  15,0,0,20,  0,0,0,0,  17,0,24,0,  18,0,0,23
6300   REM ***********OBJECTS****************
6310   DATA   KNIFE,2
6320   DATA   PAIR OF FLIPPERS,6
6330   DATA   KEY,0
6340   DATA   CLUMP OF SEAWEED,0
6350   DATA   ROTTEN OLD BONE,15
6360   DATA   MAGIC PLUG,99
6370   DATA   YELLOW PAIR OF HANDCUFFS,0
6380   DATA   GREEN PAIR OF HANDCUFFS,9
6390   DATA   RED PAIR OF HANDCUFFS,17
6400   DATA   BLUE PAIR OF HANDCUFFS,11
```

```
6500   REM    ***************NOUNS*************
6510   DATA   NOR,SOU,EAS,WES,DOG,CHE,WIN,KNI,FLI,KEY,
SEA,BON,PLU,HAN
6520   REM ****************VERBS***************
6530   DATA   GO ,GET,DROP,CUT,WEAR,GIVE,UNLOCK,USE,INVE
NTORY,HELP,LOOK
```

THE MAIN PROGRAM

You are about to enter the undersea kingdom of Neptune's Caverns. In this section of the program the initial screen which presents the player with instructions is given. The variables used in the game are given their initial values. These operations are all called up by the control program which is only seven lines long!

```
             NEPTUNE'S CAVERNS
   YOU HAVE FOUND THE MAGIC PLUG THAT
BELONGS AT THE BOTTOM OF THE SEA, AND
DECIDE TO REPLACE IT BEFORE THE WATER
DRAINS AWAY. WITH YOUR SCUBA GEAR YOU
DIVE INTO THE OCEAN AND BEGIN YOUR
ADVENTURE.
(IF YOU NEED ASSISTANCE TYPE HELP)

YOU ARE IN FRONT OF A WOODEN DOOR.YOU
CAN SEE NO HANDLE.

EXITS:   EAST   WEST

WHAT DO YOU DO NEXT?
```

The first part of the listing is the CONTROL PROGRAM contained in lines 10 to 70. The program is structured so that it is grouped into subroutines. The control program calls these subroutines in the order they are required – enabling you to see at once how the program works. Each subroutine is given later on in the listing. On a call to each subroutine the computer goes to the relevant subroutine – for example, **GOSUB 5000** – the initialize subroutine – takes control to line 5000.

```
10   REM    NEPTUNE'S CAVERNS
20   GOSUB 5000: REM   INITIALIZE
30   GOSUB 1000: REM   INPUT
40   GOSUB 2000: REM   SORT
50   IF WF = 0 AND LF = 0 THEN GOTO 30
60   GOSUB 5500: REM   END
70   END
```

Line 20 calls the initialize subroutine, where all the variables and arrays needed for the game are set up. The next two subroutines, **INPUT** and **SORT**, do most of the work of the program. Line 50 tests for the end of the game by examining the win flag **WF** and the lose flag **LF**. If neither flag is set, the instruction **GOTO 30** sends the program back to line 30, ready to repeat the process. At line 60 the game is over, and the **END** subroutine will display a suitable message on the screen, followed by a question to play again. If the player wishes to have another go, he or she replies and the game is **RUN** again, otherwise line 70 **END**s the program. By using subroutines, the program is essentially written in seven lines. The computer has been told what to do, and in which order to do it. All that now remains is to write each subroutine so that the computer can jump to the relevant line number when the subroutine is called. The first subroutine is **INITIALIZE**. The first eight lines clear the screen (using the **HOME** command in line 5010) and then **PRINT** an introduction to the game.

```
5000   REM    *********INITIALIZE*********
5010   HOME : PRINT  TAB( 10);"NEPTUNE'S CAVERNS"
5020   PRINT : PRINT "   YOU HAVE FOUND THE MAGIC PLUG"
5030   PRINT " THAT BELONGS AT THE BOTTOM OF THE SEA"
5040   PRINT " AND DECIDE TO REPLACE IT BEFORE THE"
5050   PRINT " WATER DRAINS AWAY. WITH YOUR SCUBA GEAR"
5060   PRINT " YOU DIVE INTO THE OCEAN AND BEGIN YOUR"
5070   PRINT " ADVENTURE...."
5080   PRINT  TAB( 3);"IF YOU NEED ASSISTANCE TYPE
       'HELP'"
```

It is a good plan to give the variables used in the game their initial values while the player is reading the instructions. Initialization of variables can take time in complex games.

```
5090  NN = 14:VV = 11:CP = 3:IN = 1:HC = 0:WF = 0:LF =
0:UF = 0:FF = 0:CF = 0
5100   DIM LO$(24): DIM EX(24,4): DIM NO$(NN): DIM VB$
(VV): DIM OB$(10): DIM OB(10)
5110   FOR I = 1 TO 24
5120   IF I = 17 OR I = 18 OR I = 23 OR I = 24 THEN LO$
(I) = LO$(12): GOTO 5140
5130   READ D$:LO$(I) = D$
5140   NEXT I
5150   FOR I = 1 TO 24: FOR J = 1 TO 4: READ EX(I,J):
NEXT J: NEXT I
5160   FOR I = 1 TO 10: READ OB$(I): READ OB(I): NEXT I
5170   FOR I = 1 TO NN: READ NO$(I): NEXT I
5180   FOR I = 1 TO VV: READ VB$(I): NEXT I
5190   GOSUB 4000: RETURN : REM  LOOK
```

Line 5090 gives the variables (**NN, VV, CP, IN, HC**) their initial values, and sets the flags (**WF, LF, FF, CF**) to zero. The next line **DIM**ensions six arrays, reserving space for descriptions of rooms, **LO$**; exits from each room, **EX**; nouns, **NO$**; verbs, **VB$**; objects, **OB$**; and the position of each object **OB**. For example, **DIM LO$ (24)** sets aside memory space for 24 descriptions and **DIM EX (24,4)** reserves space for 24 room numbers, each with four exits. Lines 5110 to 5180 **READ** information from the **DATA** statements into the various arrays, using a **FOR ... NEXT** loop. Note that line 5120 tells the computer that locations 17, 18, 23 and 24 all use the same **DATA**. A list of the variables used is given below. Line 5190 tells the computer to **GOSUB 4000** as this subroutine uses the end of the "LOOK" section of the **SORT** subroutine.

VARIABLES

NN = NUMBER OF NOUNS   VV = NUMBER OF VERBS

CP = CHARACTER POSITION (ROOM)   IN = NUMBER OF ITEMS IN INVENTORY

HC = NUMBER OF HANDCUFFS

OB $ (I) = OBJECT   NO $ (I) = NOUN

OB (I) = OBJECT NUMBER   NO(I) = NOUN NUMBER

LO$ (I) = LOCATION DESCRIPTION VB $ (I) = VERB

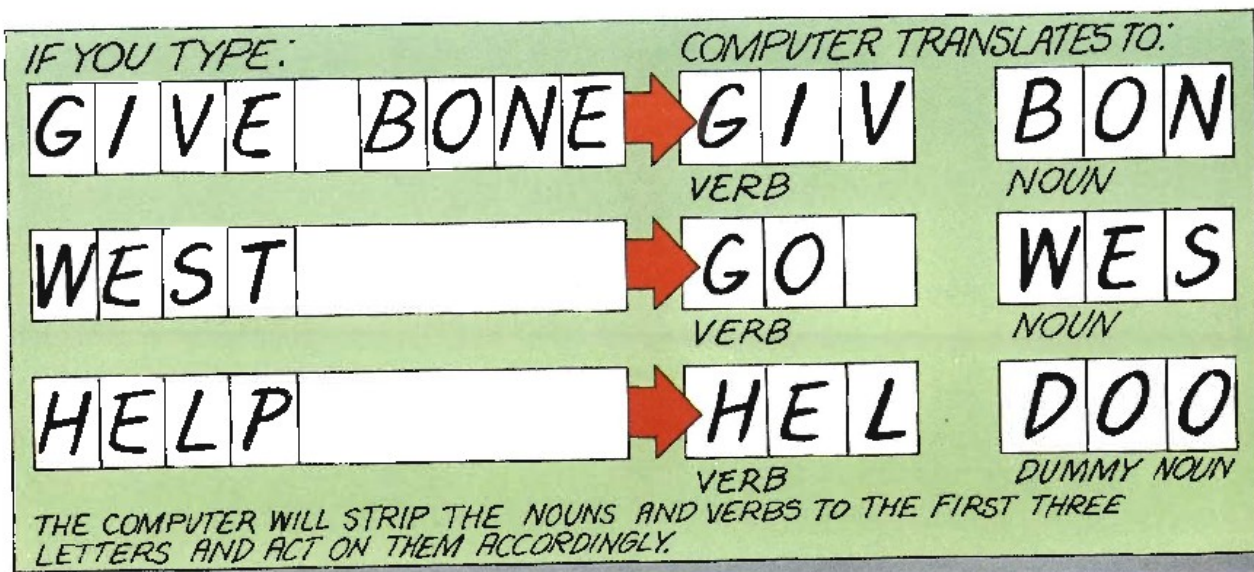EX (CP,NO) = EXIT FROM LOCATION VB (I) = VERB NUMBER

FLAGS

WF = WIN FLAG   UF = UNLOCK FLAG   CF = CUT FLAG

LF = LOST FLAG   FF = FLIPPER FLAG

```
1000   REM    INPUT
1010   VB$ = "":NO$ = ""
1020   INPUT "WHAT DO YOU DO NEXT ?";R$
1030   FOR I = 1 TO LEN (R$)
1040   IF  MID$ (R$,I,1) = " " THEN VB$ = LEFT$ (R$,3)
:NO$ = MID$ (R$,I + 1,3):I = LEN (R$)
1050   NEXT I
1060   IF NO$ < > "" THEN  RETURN
1070 R$ = LEFT$ (R$,3)
1080   IF R$ = "NOR" OR R$ = "SOU" OR R$ = "EAS" OR R$
= "WES" THEN VB$ = "GO ":NO$ = R$: RETURN
1090   IF R$ = "HEL" OR R$ = "INV" OR R$ = "LOO" THEN N
O$ = "DOO":VB$ = R$: RETURN
1100   PRINT "I DON'T UNDERSTAND THAT "
1110   GOTO 1010
```

The next subroutine is **INPUT**. It accepts a two-word "sentence" in the form verb/noun, or a one-word instruction. The computer tells the player if it doesn't recognize the sentence and returns to the beginning for another attempt. Acceptable one word instructions are "HELP," "INVENTORY," "LOOK" and the four directions. The latter is to save the player having to type "GO NORTH," etc. It is enough to type "NORTH" on its own, or even "NOR." The subroutine puts "GO" into **VB$** (verb) and "NOR" into **NO$** (noun). "HELP," "INVENTORY" and "LOOK" are treated similarly as verbs with a dummy noun "DOO" inserted. Line 1010 ensures that **VB$** and **NO$** are empty before starting and line 1020 puts the player's response into **R$**. Lines 1030 to 1050 split **R$** into **VB$** and **NO$** so that verbs and nouns can be handled separately.

IF YOU TYPE: COMPUTER TRANSLATES TO:

| G | I | V | E | | B | O | N | E | → | G | I | V | | B | O | N |
VERB   NOUN

| W | E | S | T | | | | | | → | G | O | | W | E | S |
VERB   NOUN

| H | E | L | P | | | | | | → | H | E | L | | D | O | O |
VERB   DUMMY NOUN

THE COMPUTER WILL STRIP THE NOUNS AND VERBS TO THE FIRST THREE LETTERS AND ACT ON THEM ACCORDINGLY.

A loop is started in line 1030 and the loop limit is set to the character LENgth of the sentence. Line 1040 checks IF the next character in the sentence is a space – if it is, VB$ is set to the first three characters and NO$ to the three characters after the space. Finally, the loop counter is set to a maximum to stop the search going further. If NO$ contains something, then line 1080 ends the subroutine, since the player must have typed a two word sentence. If the sentence contains no spaces, it may still be an acceptable single word. It is then stripped to its first three characters and lines 1080 and 1090 test to see if it is a direction or a command. If the computer recognizes the word, VB$ and NO$ are arranged and the subroutine is ended. If the computer doesn't recognize the word, line 1100 PRINTs an appropriate message. Line 1110 loops back to the beginning of the subroutine to allow the player to give another instruction.
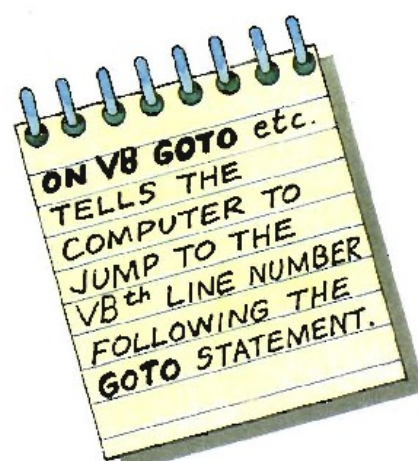
```
2000    REM    SORT
2010    VB = 0:NO = 0: FOR I = 1 TO VV
2020    IF VB$ =  LEFT$ (VB$(I),3) THEN VB = I:I = VV
2030    NEXT I
2040    FOR I = 1 TO NN
2050    IF NO$ = NO$(I) THEN NO = I:I = NN
2060    NEXT I
2070    IF NO = 0 OR VB = 0 THEN  PRINT "I DON'T
UNDERSTAND THAT ": RETURN
2080    ON VB GOTO 3000,3100,3200,3300,3400,3500,3620,37
00,3800,3900,4000
```

The SORT subroutine decides if the three letter words are valid - a technique often called "parsing." The loop between lines 2010 and 2030 tries to identify the verb. If VB$ is the same as the first three letters of one of the verbs stored in the VB$ array, then line 2020 sets VB equal to the value of the loop variable. The loop is ended by setting the loop variable to a maximum. In the same way, lines 2040 to 2060 check for a valid noun, except that the nouns are already stored as three letters so they can be compared directly.

Line 2070 checks if NO or VB are still zero. If so, a message is printed and the subroutine ends here. If the noun and verb pass all these tests, they are valid words and the relevant action is taken in line 2080. For example, if the value of VB is four (verb is "CUT") then the computer will GOTO the fourth line number in the list and carry out the instructions given from that line onwards, until the subroutine is ended with the RETURN instruction. These sections of the SORT subroutine are explained in the next chapter.

ON VB GOTO etc. TELLS THE COMPUTER TO JUMP TO THE VBth LINE NUMBER FOLLOWING THE GOTO STATEMENT.
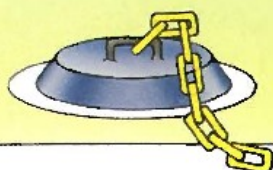
```
5500   REM   END
5510   IF CP = 22 THEN  PRINT "WITH ONE SWIFT SNAP THE
SHARK BITES OFF YOUR HEAD": GOTO 5540
5520   IF CP = 16 THEN  PRINT "THE SPINES ON THE SEA-UR
CHINS ARE VERY  POISONOUS! YOU DIE A HORRIBLE DEATH
   NEXT TIME TAKE PRECAUTIONS!": GOTO 5540
5530   PRINT "WITH A 'THUNK' THE PLUG DROPS INTO THE  H
OLE AND THE SWIRLING WATERS GROW STILL  CONGRATULATIONS
! YOU SAVED THE SEAS !
5540   INPUT "DO YOU WISH TO PLAY AGAIN? ";R$
5550   IF  LEFT$ (R$,1) = "Y" THEN  RUN
5560   PRINT "BYE-BYE": RETURN
```

The last subroutine is **END** This is called when either the win or lose flag is set - the only two conditions when the game can end. The player dies if he or she enters the SHARK ROOM (character position number 22), so line 5510 tests if the player is there. If the player is in the SHARK ROOM the computer displays the gruesome message "WITH ONE SWIFT SNAP THE SHARK BITES OFF YOUR HEAD." Control will pass to line 5540, jumping over lines 5520 and 5530.

If the character position is not equal to 22 (the SHARK ROOM), line 5520 tests if **CP** (the character position) is equal to 16 (the SEA URCHIN ROOM). The bad news message is displayed on the screen. If the player is in neither of these rooms, he or she must have won the game. Line 5530 tells the player of his or her success, and lines 5540 and 5550 offer the player another go.
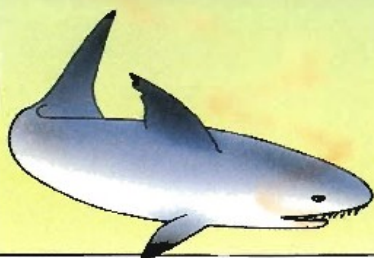
YOU WIN IF YOU PUT THE PLUG IN THE HOLE IN BOX 19.

WITH A "THUNK" THE PLUG DROPS INTO THE HOLE AND THE SWIRLING WATERS GROW STILL. **CONGRATULATIONS,** YOU SAVED THE SEAS

YOU LOSE IF YOU ENTER BOX 22.

WITH ONE SWIFT SNAP THE SHARK BITES OFF YOUR HEAD.

YOU LOSE IF YOU EXIT BOX 16 NORTH OR WEST WITHOUT WEARING FLIPPERS.

THE SPINES ON THE SEA URCHINS ARE VERY POISONOUS! YOU DIE A HORRIBLE DEATH. NEXT TIME TAKE PRECAUTIONS.

The first part of the listing is the CONTROL PROGRAM in lines 10 to 70. The program is grouped into subroutines, which are called in the order they are required – enabling you to see at once how the program works. Each subroutine is called by the **GOSUB** instruction followed by a program line number. When the subroutine has been completed a signal to **RETURN** to the control program is given

```
10 REM NEPTUNE'S CAVERNS
20 GOSUB 5000: REM INITIALIZE
30 GOSUB 1000: REM INPUT
40 GOSUB 2000: REM SORT
50 IF WF<>1 AND LF<>1 THEN GOTO 30
60 GOSUB 5700: REM END
70 END
```

Line 20 calls the **INITIALIZE** subroutine where all the variables and arrays needed for the game are set up. The next two subroutines, **INPUT** and **SORT**, do most of the work of the program. Line 50 tests for the end of the game by examining the win flag **WF** and the lose flag **LF**. If neither flag is set, the instruction **GOTO 30** sends the program back to line 30 ready to repeat the process. At line 60 the game is over, and the **END** subroutine will display a suitable message on the screen, followed by a question to play again. If the player wishes to have another go, he or she replies and the game is **RUN** again, otherwise line 70 **END**s the program. The first subroutine is **INITIALIZE**. In line 5010, the screen is cleared using the reversed heart symbol and then the game instructions are displayed.

```
5000 REM ***********INITIALIZE***********
5010 PRINT""
5020 PRINTSPC(12)"NEPTUNE'S  CAVERNS"
5030 PRINT "  YOU  HAVE  FOUND  THE  MAGIC  PLUG
      THAT"
5040 PRINT"BELONGS  AT  THE  BOTTOM  OF  THE
      SEA.  AND"
5050 PRINT"DECIDE  TO  REPLACE  IT  BEFORE  THE
      WATER"
5060 PRINT"DRAINS  AWAY.  WITH  YOUR  SCUBA
      GEAR  YOU"
5070 PRINT"DIVE  INTO  THE  OCEAN  AND  BEGIN
      YOUR"
5080 PRINT"ADVENTURE...."
5090 PRINT "<IF  YOU  NEED  ASSISTANCE  TYPE
      HELP>"
```

It is a good plan to set the variables to their initial values after
printing the instructions on the screen (lines 5100 to 5210).
This allows the player time to read the instructions while the
computer is busy with the **DATA**.

```
5100  NN=14:VV=11:CP=3:IN=1:HC=0:WF=0:LF=0:UF=0:
      FF=0:CF=0
5110  DIM T$(24):DIM D$(24):DIM EX(24,4):DIM NO$(NN):
      DIM VB$(VV)
5120  DIM OB$(10):DIM OB(10)
5130  FOR I=1 TO 24
5140  IF I=17 OR I=18 OR I=23 OR I=24 THEN T$(I)=T$(13):
      D$(I)=D$(13):GOTO 5160
5150  READ T$,D$:T$(I)=T$:D$(I)=D$
5160  NEXT I
5170  FOR I=1 TO 24:FOR J=1 TO 4:READ EX(I,J):NEXT J,I
5180  FOR I=1 TO 10:READ OB$(I),OB(I):NEXT I
5190  FOR I=1 TO NN:READ NO$(I):NEXT I
5200  FOR I=1 TO VV:READ VB$(I):NEXT I
5210  GOSUB 4000: RETURN: REM LOOK
```

Line 5100 gives the variables (**NN, VV, CP, IN, HC**) their
initial values, and sets the flags (**WF, LF, UF, FF, CF**) to zero.
The next lines DIMension seven arrays reserving space for
descriptions of rooms, **T$** and **D$**; exits **EX**; **NO$** and **VB$**
(nouns and verbs), objects; **OB$**; and starting location of each
object **OB**. For example, **DIM OB$ (10)** sets aside memory
space for 10 objects to be described. Lines 5130 to 5200
READ DATA into the various arrays. Note that line 5140 tells
the computer that locations 17,18,23 and 24 all use the same
descriptive **DATA**. Line 5210 instructs the computer to
GOSUB line 4000 because **INITIALIZE** shares its end
with the "LOOK" section of the **SORT** subroutine.

VARIABLES

NN = NUMBER OF NOUNS   VV = NUMBER OF VERBS
CP = CHARACTER POSITION (ROOM)   IN = NUMBER OF ITEMS IN INVENTORY
HC = NUMBER OF HANDCUFFS

OB$ (I) = OBJECT   NO$(I)=NOUN
OB(I) = OBJECT NUMBER   NO(I)=NOUN NUMBER
LO$ (I) = LOCATION DESCRIPTION   VB$(I) = VERB
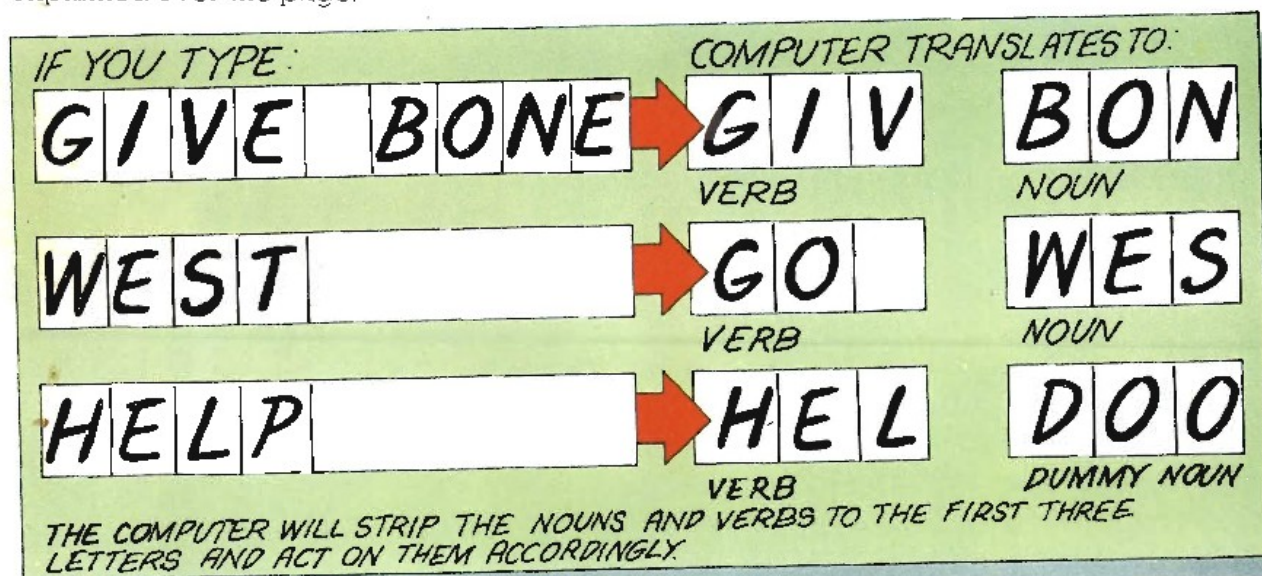EX (CP,NO) = EXIT FROM LOCATION   VB(I)=VERB NUMBER

FLAGS

WF = WIN FLAG   UF = UNLOCK FLAG   CF = CUT FLAG
LF = LOST FLAG   FF = FLIPPER FLAG

```
1000  REM *********INPUT ***************
1010  VB$="":NO$="":R$=""
1020  INPUT"WHAT DO YOU DO NEXT";R$
1030  FOR I=1 TO LEN(R$)
1040  IF MID$(R$,I,1)=" " THEN VB$=LEFT$(R$,3):
      NO$=MID$(R$,I+1,3):I=LEN(R$)
1050  NEXT I
1060  IF NO$<>"" THEN RETURN
1070  R$=LEFT$(R$,3)
1080  IF R$="NOR"OR R$="SOU"OR R$="EAS" OR
      R$="WES" THEN VB$="GO ":NO$=R$:RETURN
1090  IF R$="HEL" OR R$="INV" OR R$="LOO"
      THEN NO$="DOO":VB$=R$:RETURN
1100  PRINT"I DON'T UNDERSTAND THAT"
1110  GOTO 1010
```

The next subroutine is **INPUT**. It accepts a two-word "sentence" in the form verb/noun, or a one-word instruction. If neither of these are recognized, the computer **PRINT**s a message and returns to the beginning for another attempt. Acceptable one word instructions are "HELP," "INVENTORY," "LOOK" and the four directions. The latter is to save the player having to type "GO NORTH," etc. It is enough to type "NORTH" on its own, or even "NOR." The subroutine puts "GO" into **VB$** (verb) and "NOR" into **NO$** (noun) for processing later. "HELP," "INVENTORY" and "LOOK" are treated similarly as verbs with a dummy noun "DOO" inserted. Line 1010 ensures that **VB$**, **NO$** and **R$** are empty before starting, and line 1020 puts the player's response into **R$**. Lines 1030 to 1050 contain the loop that splits **R$** into **VB$** and **NO$**. The rest of this subroutine is explained over the page.

IF YOU TYPE: COMPUTER TRANSLATES TO:

GIVE BONE → GIV (VERB) BON (NOUN)

WEST → GO (VERB) WES (NOUN)

HELP → HEL (VERB) DOO (DUMMY NOUN)

THE COMPUTER WILL STRIP THE NOUNS AND VERBS TO THE FIRST THREE LETTERS AND ACT ON THEM ACCORDINGLY.
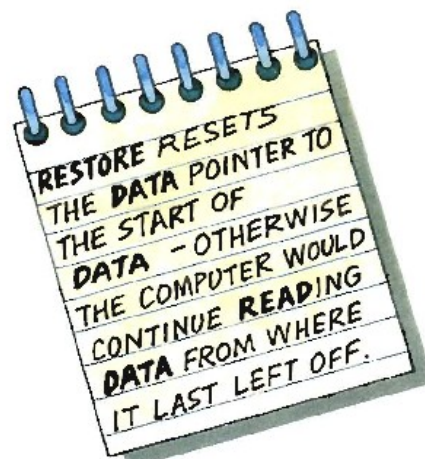
A loop is started in line 1030 and its limit is set to the character **LEN**gth of the sentence. Line 1040 checks **IF** the next character in the sentence is a space – if so, **THEN VB$** is set to the first three characters, and **NO$** to the three characters after the space. Finally, the loop counter is set to a maximum to stop the search going further. If **NO$** contains something, then line 1060 ends the subroutine, as a two word sentence must have been recognized. If the program has reached line 1070 the sentence contains no spaces, but may still be an acceptable single word. It is then stripped to its first three characters. Lines 1080 and 1090 see if it is a direction or a command. If the word is valid, **VB$** and **NO$** are arranged and the subroutine is ended. The sentence will have made no sense at line 1100, and a message is **PRINT**ed. Line 1110 loops back to the start to give the player another attempt to enter a valid instruction.

```
2000  REM*******SORT****************************
2010  VB=0: NO=0
2020  FOR I=1 TO VV
2030  IF VB$=LEFT$(VB$(I),3) THEN VB=I:I=VV
2040  NEXT
2050  FOR I=1 TO NN
2060  IF NO$=NO$(I) THEN NO=I:I=NN
2070  NEXT
2080  IF NO=0 OR VB=0 THEN PRINT"I DON'T
         UNDERSTAND THAT":RETURN
2090  ON VB GOTO 3000,3100,3200,3300,3400,3500,3620
         ,3700,3800,3900,4000
```

The **SORT** subroutine decides if the three – letter words are valid – a technique often called "parsing." The loop between lines 2020 and 2040 tries to identify the verb. If **VB$** is the same as the first three letters of one of the verbs stored in the **VB$** array, then line 2030 sets **VB** equal to the value of the loop variable. The loop is ended by setting the loop variable to a maximum. In the same way, lines 2050 to 2070 check for a valid noun, except that the nouns are already stored as three letters so they can be compared directly. Line 2080 checks if **NO** or **VB** are still zero. If so, a message is printed and the subroutine ends here. If the noun and verb pass all these tests, they are valid words and the relevant action is taken in line 2090. For example, if the value of **VB** is four (verb is "CUT") then the computer will **GOTO** the fourth line number in the list and carry out the instructions given from that line onwards until the subroutine is ended with the **RETURN** instruction. These sections of the **SORT** subroutine are explained in the next chapter.
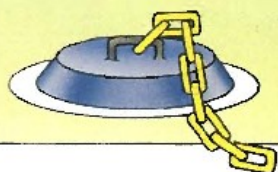
RESTORE RESETS THE **DATA** POINTER TO THE START OF **DATA** – OTHERWISE THE COMPUTER WOULD CONTINUE **READING DATA** FROM WHERE IT LAST LEFT OFF.

```
5500  REM********<<END>>********************
5510  IF CP=22 THEN PRINT"WITH A SNAP, THE SHARK
      BITES OFF YOUR HEAD.":GOTO 5570
5520  IF CP=16 THEN PRINT"YOU HAVE STEPPED ON A
      POISONOUS SEA-URCHIN."
5530  IF CP=16 THEN PRINT"YOU DIE A HORRIBLE
      DEATH.":GOTO 5570
5540  PRINT"WITH A 'THUNK' THE PLUG DROPS
      INTO THE"
5550  PRINT"HOLE AND THE SWIRLING WATERS
      GROW STILL."
5560  PRINT"CONGRATULATIONS! YOU SAVED THE
      SEAS!"
5570  INPUT"DO YOU WISH TO PLAY AGAIN";R$
5580  IF LEFT$(R$,1)="Y" THEN RUN
5590  PRINT"BYE-BYE":RETURN
```

The last section to be defined is the **END** subroutine. If either the win or lost flag is found to be set in line 50, then control falls through to line 60 and there is a call to **GOSUB 5500**. Line 5510 tests to see if the player is in the SHARK ROOM (character position 22). If so, a death message is printed and control **GO**es **TO** line 5570, where the player is asked if he or she would like another game.

If the character position is not 22, lines 5520 and 5530 test **IF CP** is equal to 16 (the SEA URCHIN room), and print a message. If neither of these conditions are met the player has won the game. A message is printed in line 5540, and lines 5570 and 5580 deal with the "play again" routine.
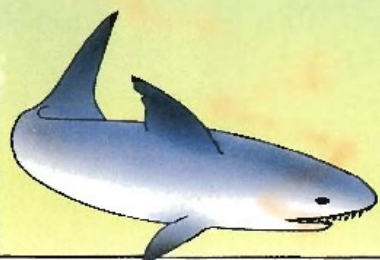
YOU WIN IF YOU PUT THE PLUG IN THE HOLE IN BOX 19

WITH A "THUNK" THE PLUG DROPS INTO THE HOLE AND THE SWIRLING WATERS GROW STILL. **CONGRATULATIONS** YOU SAVED THE SEAS.
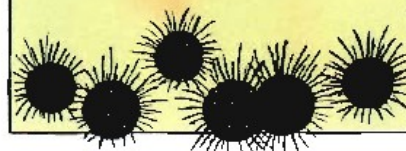
YOU LOSE IF YOU ENTER BOX 22

WITH ONE SWIFT SNAP THE SHARK BITES OFF YOUR HEAD.

YOU LOSE IF YOU EXIT BOX 16 NORTH OR WEST WITHOUT WEARING FLIPPERS.

THE SPINES ON THE SEA URCHIN ARE VERY POISONOUS! YOU DIE A HORRIBLE DEATH. NEXT TIME TAKE PRECAUTIONS.

Testing your program

Before you carry on with the next section of the program, it is a good idea to save your work so far on tape or disk. You'll find the instructions for this in your user's manual. To test that the program is correct, turn to page 31 or 37 and type in the "LOOK" section beginning at line 4000. Then follow the instructions in the box below. You will be able to explore the first six rooms in the game.

APPLE IIe: 1. TYPE **2065 IF NO<=4 THEN CP=EX(CP,NO): GOTO 4000** HIT [RETURN]. 2. TYPE **20 6 ENDPROC** HIT [RETURN]. 3. TYPE **RUN**, HIT [RETURN]. WHEN YOU ARE SATISFIED EVERYTHING **RUNS** CORRECTLY, DELETE LINES 2065 AND 2066 BEFORE PROCEEDING.

```
EXITS:  EAST   WEST

WHAT DO YOU DO NEXT ?GO EAST

YOU ARE ON THE SEABED.TO THE SOUTH IS A
BARNACLED WALL.A CLIFF BLOCKS THE WAY
EAST.
A PAIR OF FLIPPERS IS HERE

EXITS:  WEST

WHAT DO YOU DO NEXT ?
```

```
EXITS:  EAST   WEST

WHAT DO YOU DO NEXT ?GO WEST

YOU ARE ON THE SEABED.TO THE SOUTH A
BARNACLED WALL TOWERS ABOVE YOU.
A KNIFE IS HERE

EXITS:  EAST   WEST

WHAT DO YOU DO NEXT ?
```

COMMODORE: 1. ENTER **2075 IF NOK=4 THEN LET CP=EX(CP,NO): GOTO 4000** HIT [ENTER] 2. ENTER **2076 RETURN** HIT [ENTER]. 3. HIT [RUN]. WHEN YOU ARE SATISFIED EVERYTHING **RUNS** AS IT SHOULD, DELETE LINES 2075 AND 2076 BEFORE PROCEEDING.

```
YOU ARE ON THE SEABED.TO THE SOUTH A
BARNACLED WALL AND A SQUARE PATCH OF SEA
WEED GROWING ON IT.
EXITS:  EAST   WEST

WHAT DO YOU DO NEXT ?GO EAST

YOU ARE ON THE SEABED.TO THE SOUTH
BARNACLED WALL A CLIFF BLOCKS THE
WAY EAST.
A PAIR OF FLIPPERS IS HERE

EXITS:  WEST

WHAT DO YOU DO NEXT ?
```

```
YOU ARE IN FRONT OF A WOODEN DOOR.YOU
CAN SEE AN HANDLE.
EXITS:  EAST   WEST

WHAT DO YOU DO NEXT? GO WEST

YOU ARE ON THE SEABED.TO THE SOUTH A
BARNACLED WALL TOWERS ABOVE YOU.
A KNIFE IS HERE

EXITS:  EAST   WEST

WHAT DO YOU DO NEXT ?
```

THE VERB ROUTINES

This section of the program is actually part of the
subroutines which begin at line 2000. It contains each of the
verbs that the player can use to direct his or her actions
during the game, and makes sure that the computer gives
the appropriate response. Since the computer automatically
places each line in its numerical order, this section will
appear under the relevant subroutine in the program
listings.

```
EXITS:    EAST    WEST

WHAT DO YOU DO NEXT? GO EAST
YOU ARE ON THE SEABED. TO THE SOUTH A
BARNACLED WALL TOWERS ABOVE YOU.
EXITS:    EAST    WEST

WHAT DO YOU DO NEXT? GO EAST
YOU ARE ON THE SEABED. TO THE SOUTH A
BARNACLED WALL HAS A SQUARE PATCH OF SEA
WEED GROWING ON IT.
EXITS:    EAST    WEST

WHAT DO YOU DO NEXT? CUT SEAWEED
THE SEAWEED FALLS AWAY TO REVEAL AN
OPEN WINDOW.
```

```
3000    REM  *************GO*****************
3010    IF NO > 4 THEN  PRINT "GO WHERE ? ": RETURN
3020    IF EX (CP,NO) = 0 THEN  PRINT "NO EXIT THAT WAY "
: RETURN
3030    IF CP = 16 AND (NO = 1 OR NO = 4) AND FF = 0 THE
N LF = 1: RETURN
3040 CP = EX (CP,NO): GOTO 4000
```

Control goes to line 3000 when the verb identified in the
SORT subroutine is "GO" (verb 1). Line 3010 **PRINT**s a
message if the noun wasn't one of the four directions. Line
3020 tests if an attempt is made to move in a direction with no
exit that is, if the exit array **EX (CP,NO)** equals zero. Line
3030 tests if the player has stepped on a sea urchin and died!
It does this by checking the character position (16=SEA
URCHIN ROOM), then the direction (north or west is
dangerous); then it checks the FLIPPERS flag, **FF (FF = 1**
means that the player is wearing flippers). If line 3040 is
reached, all is well and **CP** is changed to its new location.
GOTO 4000 sends control to the "LOOK" section which
PRINTs the new position.

```
3100    REM  ***********GET***************
3110    IF NO = 6 AND CP = 13 THEN  PRINT "IT'S TOO
HEAVY !": RETURN
3120    IF NO < 8 THEN  PRINT "DON'T BE SILLY": RETURN
3130    IF OB(NO - 7) = 99 AND NO$ < > "HAN" THEN
PRINT "YOU'VE ALREADY GOT IT !": RETURN
3140    FOR I = 7 TO 10
3150    IF OB(I) = CP AND NO = 14 THEN NO = I + 7:HC = H
C + 1:I = 10
3160    NEXT I
3170    IF OB(NO - 7) = CP THEN  PRINT "O.K.":OB(NO - 7)
 = 99:IN = IN + 1: RETURN
3180    PRINT "IT ISN'T HERE !": RETURN
```

If the verb number is two, then control is sent to the "GET"
section. Lines 3110 and 3120 anticipate any attempt to do the
impossible. Line 3130 checks if the object is already carried
i.e. has the value 99 . . . **AND** that the noun input is not "HAN."
Lines 3140 to 3160 check if a pair of handcuffs is being taken
by comparing the location of each handcuff **OB (I)**, with
character position **CP**. If **OB (I)=CP** then the handcuff
inventory **HC** is increased. Line 3170 compares the object
location **OB (NO-7)** with the character position **CP** and
PRINTs the "O.K." message. The object location takes the
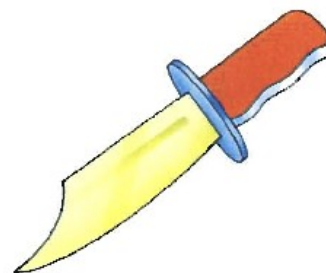value 99 to indicate the object has been picked up.

```
3200    REM ***************DROP***************
3210    IF NO < 8 THEN  PRINT "DON'T BE SILLY !": RETURN
3220    FOR I = 7 TO 10
3230    IF OB(I) = 99 AND NO = 14 THEN NO = I + 7:HC = HC - 1:I = 10
3240    NEXT I
3250    IF OB(NO - 7) < > 99 THEN  PRINT "YOU HAVEN'T GOT IT!":RETURN
3260    PRINT "O.K.":OB(NO - 7) = CP:IN = IN - 1
3270    IF CP = 19 AND NO = 13 THEN WF = 1
3280    RETURN
```

In the "DROP" section, line 3210 prevents the player from putting down anything that can't be carried. If the item to be dropped is a pair of handcuffs, a **FOR . . . NEXT** loop in lines 3220 to 3240, decreases the value of the **HC** by 1. Should the object location not equal 99, line 3250 **PRINT**s a message to inform the player that the object is not carried. If the control runs to line 3260, then the object location becomes equal to the character position and the inventory is decreased by 1 because the object has been dropped. Finally, line 3270 tests for a win, i.e. if the player has dropped the magic plug in the plug hole.

Lines 3300 to 3340 of the "CUT" section prevent the player from cutting anything but the seaweed on the barnacled wall in location five. At any illegal attempt a message is printed and the subroutine is ended. Only if all conditions are satisfied will the message in line 3350 be **PRINT**ed. Information is then updated – A CLUMP OF SEAWEED will be put into location five; an exit will become visible and the cut flag **CF** will be set to 1. In addition, the location description **LO$ (5)** is altered before the subroutine is ended.

```
3300    REM ***************CUT***************
3310    IF OB(1) < > 99 THEN  PRINT "YOU'VE NOTHING SHARP ENOUGH !": RETURN
3320    IF NO < > 11 THEN  PRINT "YOU CAN'T CUT THAT !": RETURN
3330    IF CP < > 5 THEN  PRINT "YOU CAN'T DO THAT !": RETURN
3340    IF CF = 1 THEN  PRINT "YOU'VE ALREADY DONE THAT !": RETURN
3350    PRINT "THE SEAWEED FALLS AWAY TO REVEAL AN OPEN WINDOW"
3360    OB(4) = 5:EX(5,2) = 11:CF = 1
3370    LO$(5) = LEFT$ (LO$(5),59) + " AN OPEN WINDOW IN IT ": RETURN
```
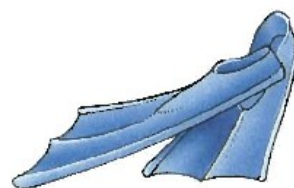
```
3400   REM   ************WEAR*************
3410   IF FF = 1 AND NO$ = "FLI" THEN   PRINT "YOU'VE
   ALREADY GOT THEM ON !": RETURN
3420   IF OB(2) = 99 AND NO$ = "FLI" THEN FF = 1:OB(2)
   = 0: PRINT "THEY FIT NICELY !": RETURN
3430   IF NO = 14 AND MC > 0 THEN   PRINT "THAT'S REALLY
   SILLY !": RETURN
3440   PRINT "YOU CAN'T WEAR THAT !": RETURN
```

The verb "WEAR" allows the player to put on the FLIPPERS
found in location six. If the player moves west or north from
location 16 without wearing flippers, then he or she will be
killed by the poisonous sea urchins The possibility of the
flippers already being worn is considered in line 3410. In
line 3420 the player is allowed to wear the flippers if they are
being carried, i.e. **OB (2) = 99** and **IF** the noun input is 'FLI.'
The flipper flag (**FF**) is then set to 1 and the FLIPPERS are
removed from the items carried with the instruction
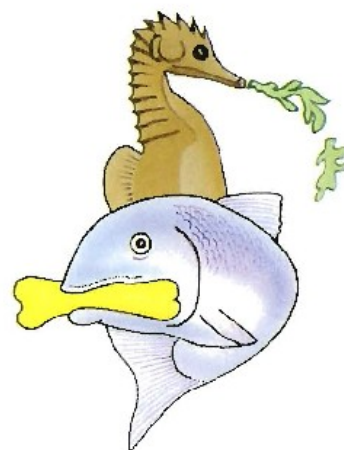**OB (2) = 0**. A suitable message is then **PRINT**ed.

```
3500   REM   ************GIVE*************
3510   IF CP = 10 AND NO = 11 THEN 3550
3520   IF CP = 7 AND NO = 12 THEN 3580
3530   IF CP = 20 THEN   PRINT "THAT WON'T DO ANY
   GOOD!": RETURN
3540   PRINT "NOTHING HERE WANTS IT!": RETURN
3550   IF OB(4) < > 99 THEN   PRINT "YOU HAVEN'T GOT
   IT": RETURN
3560 OB(4) = 0:OB(3) = 10
3570   PRINT "THEY CROWD ROUND YOU AND SOMETHING
   GLINTS IN THE CORNER ": RETURN
3580   IF OB(5) < > 99 THEN   PRINT "YOU HAVEN'T GOT
   IT": RETURN
3590 OB(5) = 0:EX(7,2) = 13
3600   PRINT "THE FISH SNATCHES THE BONE AND RETIRES TO
   A CORNER"
3610 LO$(7) = "YOU ARE IN A LOW CAVERN. A BONEFISH IS
   IN THE CORNER GNAWING ON A THIGH BONE": RETURN
```

The "GIVE" section enables seaweed to be given to the sea
horses in location ten and a bone to be offered to the
bonefish in location seven. Line 3510 tests for the first
possibility, and sends control to line 3550 if the player is
in the correct room (**CP = 10**) and the noun is "SEAWEED."
At this line, the seaweed is tested to discover **IF** it is not
being carried by the player, i.e. **OB (4) < > 99** and if this is
so,then a suitable message is **PRINT**ed. Otherwise, the
player must have satisfied all conditions.

Line 3560 removes the seaweed OB (4) from the room by setting it to zero and the key OB (3) is put into location ten. The following line PRINTs a clue-filled message and the subroutine ends.

If conditions in line 3510 are not met, control falls to 3520 where tests are made to discover if the player is giving a bone to the bonefish in location seven – if so, then control goes to line 3580. Otherwise a "catch all" message is PRINTed. Line 3580 tests if the bone is being carried and if it isn't then a message is PRINTed. However, IF OB (5) = 99 the player has satisfied the bonefish and bone conditions, and control drops through to line 3590. The bone is removed from the room by setting OB (5) to 0. An exit is then opened to the south when EX (7,2) is made equal to 13. A message is PRINTed in line 3600 that implies that something has happened. Finally, the location description LO$ is changed, ready for when the player types "LOOK"!

```
3620   REM *********UNLOCK**************
3630   IF OB(3) < > 99 THEN   PRINT "YOU HAVEN'T EVEN
GOT A KEY !": RETURN
3640   IF NO = 5 AND (CP = 3 OR CP = 9) THEN   PRINT "TH
ERE ISN'T EVEN A KEYHOLE !": RETURN
3650   IF CP < > 13 OR NO < > 6 THEN   PRINT "YOU
CAN'T DO THAT": RETURN
3660   IF UF = 1 THEN   PRINT "IT'S ALREADY UNLOCKED !":
 RETURN
3670 UF = 1:OB(7) = 13: PRINT "THE KEY TURNS EASILY "
3680 LO$(13) = "YOU ARE IN A ROOM THAT HAS A LARGE
OPEN CHEST IN THE MIDDLE": RETURN
```

Only Davy Jones' Locker (the CHEST) can be unlocked with the key. Line 3630 tests that the player has in fact picked this up. It may be that the player will try to unlock the door in locations three and nine. However, the door is something of a red herring (all good adventure games have these!), and can never be opened. Line 3640 deals with this situation and tells the player that the key cannot be used

Control will only pass from line 3650 if the player is in the room, has the key, and the noun input is "CHEST." IF everything is in order THEN the unlock flag UF is tested to ensure that the player has not previously unlocked the chest. Line 3670 sets UF to 1 and the key is located in the room i.e. OB (7) is made equal to 13, telling the computer that the key stays in the room. A message is PRINTed to guide the player, and the subroutine is ended.
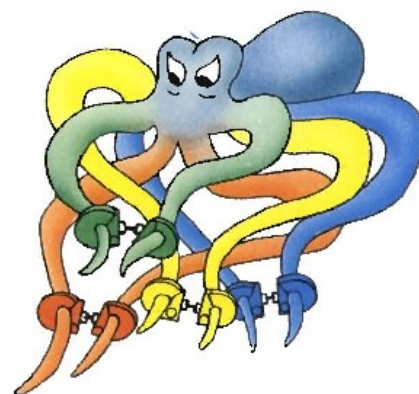
```
3700    REM ************USE*****************
3710    IF CP < > 20 OR NO < > 14 THEN  PRINT "YOU
CAN'T DO THAT HERE": RETURN
3720    IF HC < 4 THEN  PRINT "YOU HAVEN'T GOT ENOUGH
PAIRS!": RETURN
3730    PRINT "THE OCTOPUS CAN'T MOVE, HE ISN'T AMUSED"
3740    FOR I = 7 TO 10:OB(I) = 0: NEXT I
3750 HC = 0:EX(20,4) = 19:LO$(20) =  LEFT$ (LO$(20),53
) + "A MANACLED OCTOPUS SITS SULKING": RETURN
```

The "USE" verb is designed to allow the player to immobilize the octopus in location 20 with four sets of handcuffs, i.e. "USE HANDCUFFS."

Line 3710 checks **IF** the player is not in location 20 (the octopus room) **OR** the noun number is not 14 (corresponding to handcuffs – this prevents the player from trying to use any other object in this location). In other words, the message "YOU CAN'T DO THAT HERE" appears if the player types anything but USE HANDCUFFS in the octopus room. Line 3720 ensures that the player can do nothing with anything less than four pairs of handcuffs.

If control drops through to line 3730, a message is **PRINT**ed that indicates the octopus has been successfully manacled. The handcuffs are removed from the player using a **FOR . . . NEXT** loop. The handcuff positions are set to zero in line 3740. Finally, the number of handcuffs is made zero; a new exit becomes available to the west and the description of the room **LO$ (20)**, is changed accordingly.

```
3800    REM **********INVENTORY************
3810    PRINT : PRINT "YOU ARE CARRYING:-"
3820    IF IN = 0 THEN  PRINT "NOTHING !": RETURN
3830    FOR I = 1 TO 10
3840    IF OB(I) = 99 THEN  PRINT "A ";OB$(I)
3850    NEXT I: RETURN
```

The final three parts of the program make up the very useful commands of "INVENTORY," "HELP" and "LOOK," without which the game would be extremely difficult to play. If **IN** contains zero then a command to "INVENTORY" would produce the words "YOU ARE CARRYING:– NOTHING!" on the screen, **PRINT**ed by lines 3810 and 3820. However, if **IN** is not equal to zero, a loop starting at line 3830 will check each object to see if it is carried by the player, and **PRINT**s a list of those that are.

At the command "HELP," the four lines of program 3900 to 3930 **PRINT** each of the verbs on the screen to enable the player to construct acceptable two word sentences. A **FOR** ... **NEXT** loop is used to do this with the maximum value of I set to the maximum number of verbs available **VV**.
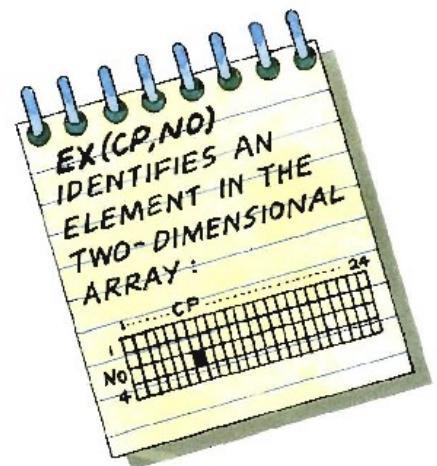
```
3900   REM  **************HELP**************
3910   PRINT : PRINT "THESE ARE THE VERBS YOU MAY USE"
3920   FOR I = 1 TO VV: PRINT VB$(I) ,: NEXT
3930   PRINT " (YOU NEED ONLY TYPE THE FIRST THREE
LETTERS)": RETURN
```

The last section of the whole program concerns the verb "LOOK." If you have followed the instructions on page 24 (in the **RUN** box), you will have already typed this part of the program.

The LOOK section displays the location description. This is very useful to the player. During the game, the screen is continually scrolling, so the location description may disappear off the top of the screen. Typing LOOK will allow the player to examine his or her surroundings again.

This section is a little peculiar because it is used as the end part of the **INITIALIZE** subroutine, linked by a **GOSUB** statement. Line 4010 **PRINT**s the location description **LO$ (CP)**. The lost flag **LF** is set if the player has ignored the warning above the south exit of location 16 and entered the shark room (location 22). Lines 4020 to 4050 **PRINT** the object found in the location of the player. Note that the mention of the BONE is sidestepped in line 4030 to make the player think harder about what is happening. Lastly, the available exits are **PRINT**ed in lines 4060 to 4100

EX(CP,NO) IDENTIFIES AN ELEMENT IN THE TWO-DIMENSIONAL ARRAY:

```
4000   REM  *************LOOK*************
4010   PRINT : PRINT LO$(CP): IF CP = 22 THEN LF = 1: R
ETURN
4020   FOR I = 1 TO 10
4030   IF (I = 5 AND CP = 15) THEN 4050
4040   IF OB(I) = CP THEN  PRINT "A ";OB$(I);" IS HERE"
4050   NEXT I
4060   PRINT : PRINT "EXITS:  ";
4070   IF EX(CP,1) > 0 THEN  PRINT "NORTH  ";
4080   IF EX(CP,2) > 0 THEN  PRINT "SOUTH  ";
4090   IF EX(CP,3) > 0 THEN  PRINT "EAST  ";
4100   IF EX(CP,4) > 0 THEN  PRINT "WEST  ";
4110   PRINT : PRINT : RETURN
```

```
3000  REM***********GO***************
3010  IF NO>4 THEN PRINT "GO  WHERE?":RETURN
3020  IF EX(CP,NO)=0 THEN PRINT"NO  EXIT  THAT
         WAY":RETURN
3030  IF CP=16 AND (NO=1 OR NO=4) AND FF=0 THEN LF=1:
         RETURN
3040  CP=EX(CP,NO) : GOSUB 4000
3050  RETURN
```

Control is sent from the **SORT** subroutine to line 3000 if the
verb is 'GO' (verb 1). Line 3010 **PRINT**s a message if the
noun wasn't one of the four directions. Line 3020 tests if an
attempt is made to move in a direction with no exit, that is if
EX (CP,NO) equals zero. Line 3030 tests if the player has
stepped on a sea urchin and died! It does this by checking
the character position (16 = SEA URCHIN room), then the
direction (north or west is dangerous); then it checks the
FLIPPERS flag, **FF** (this takes the value 1 if the player is
wearing flippers). If these conditions are met then the game
is lost so **LF** becomes 1 and the subroutine ends.

```
3100  REM***********GET***************
3110  IF NO=4 AND CP=13 THEN PRINT"IT'S  TOO
         HEAVY!":RETURN
3120  IF NO>P THEN PRINT "DON'T  BE  SILLY":RETURN
3130  IF OB(NO-7)=99 AND NO$<>"HAN"  THEN
         PRINT"YOU'VE  ALREADY  GOT  IT!":RETURN
3140  FOR I=7 TO 10
3150  IF (OB(I)=CP AND NO=14)  THEN NO=I+7:HC=HC+1:I=10
3160  NEXT I
3170  IF OB(NO-7)=CP THEN PRINT"O.K.,"  : OB(NO-7)=99:
         IN=IN+1:RETURN
3180  PRINT"IT  ISN'T  HERE!":RETURN
```

If the verb number is 2, then control is sent to the "GET"
section. Lines 3110 and 3120 anticipate any attempt to do the
impossible. Line 3130 checks **IF** the object is already
carried, i.e. has the value **99** . . . **AND** that the noun input is
not "HAN." Lines 3140 to 3160 check if a pair of handcuffs is
being taken by comparing the location of each handcuff **OB
(I)**, with character position **CP**. If **OB (I) = CP** the handcuff
inventory **HC** is increased and I is set to its maximum to end
the loop. Line 3170 checks the object location **OB (NO-7)** with
the character position **CP** and **PRINT**s the "C.K." message if
they agree. The object location then takes the value **99** to
indicate the object has been picked up and the inventory **IN**
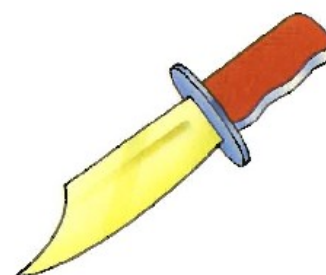is increased by 1.

```
3200  REM************DROP*****************
3210  IF NO<6 THEN PRINT"DON'T BE SILLY":RETURN
3220  FOR I=7 TO 10
3230  IF (OB(I)=99 AND NO=14) THEN NO=I-7:HC=HC-1:I=10
3240  NEXT I
3250  IF OB(NO-7)<>99 THEN PRINT"YOU HAVEN'T GOT
          IT!":RETURN
3260  PRINT"O.K.":OB(NO-7)=CP:IN=IN-1
3270  IF CP=19 AND NO=15 THEN WF=1
3280  RETURN
```

In the "DROP" section, line 3210 prevents the player from putting down anything that can't be carried. If the item to be dropped is a pair of handcuffs, a **FOR . . . NEXT** loop in lines 3220 to 3240 decreases the value of **HC** by 1. Should the object location not equal **99** then line 3250 **PRINT**s a message to inform the player that the object is not carried. If control reaches line 3260 the object location becomes equal to the character position and the inventory is decreased by **1**. Finally, line 3270 tests for a win, i.e. if the player has dropped the magic plug into the plug hole.

Lines 3300 to 3340 of the "CUT" section prevent the player from cutting anything but the seaweed on the barnacled wall in location five. At any illegal attempt a message is **PRINT**ed and the subroutine is ended. Only if all conditions are satisfied will the message in line 3350 be **PRINT**ed. Information is then updated – A CLUMP OF SEAWEED will be put into location five, an exit will become visible and the cut flag **CF** will be set to 1. In addition, the location description **D$ (5)** is altered before the subroutine is ended.

```
3300  REM************CUT*****************
3310  IF OB(1)<>99 THEN PRINT"YOU'VE NOTHING
          SHARP ENOUGH!":RETURN
3320  IF NO<>11 THEN PRINT"YOU CAN'T CUT THAT!":
          RETURN
3330  IF CP<>5 THEN PRINT"YOU CAN'T DO THAT":
          RETURN
3340  IF CF=1 THEN PRINT"YOU'VE ALREADY DONE
          THAT!":RETURN
3350  PRINT"THE SEAWEED FALLS AWAY TO REVEAL
          AN OPEN WINDOW."
3360  OB(4)=5:EX(5,2)=11:CF=1
3370  D$(5)=LEFT$(D$(5),19)+"AN OPEN WINDOW IN
          IT":RETURN
```
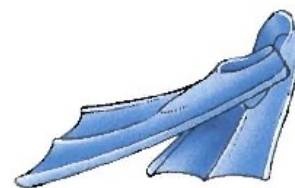
```
3400 REM ***********WEAR*************
3410 IF FF=1 AND NO$="FLI" THEN PRINT"YOU'VE
     ALREADY GOT THEM ON!":RETURN
3420 IF OB(2)=99 AND NO$="FLI" THEN FF=1:OB(2)=0:
     PRINT"THEY FIT NICELY!":RETURN
3430 IF NO=14 AND H<>0 THEN PRINT"THAT'S REALLY
     SILLY!":RETURN
3440 PRINT"YOU CAN'T WEAR THAT!":RETURN
```

The verb "WEAR" allows the player to put on the FLIPPERS found in location six. If the player moves west or north from location 16 without protecting his or her feet, then instant death will occur on the poisonous spines of the sea urchins. The possibility of the flippers already being worn is covered in line 3410. In line 3420 the player is allowed to wear the flippers if they are being carried, i.e **OB (2) = 99, AND** the noun input is "FLI." The flipper flag (**FF**) is then set to 1 and the FLIPPERS are removed from the items carried with the instruction **OB (2) = 0**. The last two lines prevent the player from wearing any of the other objects.
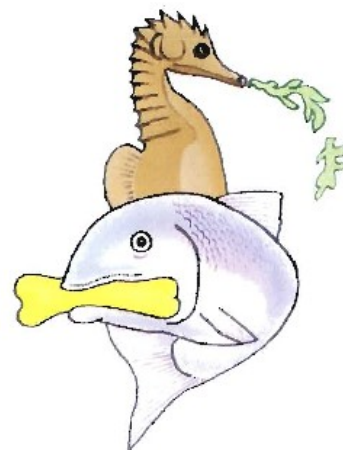
```
3500 REM ***********GIVE*************
3510 IF CP=10 AND NO=11 THEN 3550
3520 IF CP=7 AND NO=12 THEN 3580
3530 IF CP=20 THEN PRINT"THAT WON'T DO ANY
     GOOD!":RETURN
3540 PRINT"NOTHING HERE WANTS IT":RETURN
3550 IF OB(4)<>99 THEN PRINT"YOU HAVEN'T GOT
     IT":RETURN
3560 OB(4)=0:OB(3)=10
3570 PRINT"SOMETHING GLINTS IN THE CORNER":
     RETURN
3580 IF OB(5)<>99 THEN PRINT"YOU HAVEN'T GOT
     IT":RETURN
3590 OB(5)=0:EX(7,2)=13
3600 PRINT"THE FISH SNATCHES THE BONE AND
     RETIRES TO A CORNER"
3610 T$(7)="YOU ARE IN A CAVERN. IN THE
     CORNER A"
3615 D$(7)="BONEFISH IS CHEWING A THIGH-
     BONE":RETURN
```

The "GIVE" section enables seaweed to be given to the sea horses and a bone to be offered to the bonefish. Line 3510 tests for the first possibility, and sends control to line 3550 if **CP = 10** and the noun is "SEAWEED." Otherwise, the player must have satisfied all conditions. The following line prints a clue-filled message ending the subroutine.

If conditions in line 3510 are not met then control falls to line 3520 where tests are made to discover if the player is giving a bone to the bonefish in location seven – if it is, control goes to line 3580. Otherwise a "catch all" message is **PRINT**ed and the subroutine ends.

Line 3580 tests if the bone is being carried, and if it isn't a message is **PRINT**ed. However **IF OB (5) = 99** the player has satisfied both the bonefish and the bone conditions, and control drops through to line 3590. The bone is removed from the room by setting **OB (5)** to zero. An exit is then opened to the south when **EX (7,2)** is made equal to 13.

A message is **PRINT**ed in line 3590 that tells the player about the cavern and the bonefish. Finally, the location T$ and D$ are changed, ready for when the player types "LOOK!"

```
3620  REM**********UNLOCK**********
3630  IF OB(3)<>97 THEN PRINT"YOU HAVEN'T EVEN
         GOT A KEY!":RETURN
3640  IF NO=5 AND (CP=3 OR CP=9) THEN PRINT"THERE
         ISN'T EVEN A KEYHOLE!":RETURN
3650  IF CP<>13 OR NO<>6 THEN PRINT"YOU CAN'T DO
         THAT":RETURN
3660  IF UF=1 THEN PRINT"IT'S ALREADY
         UNLOCKED!":RETURN
3670  UF=1:OB(7)=13:PRINT"THE KEY TURNS EASILY"
3680  T$(13)="YOU ARE IN A ROOM WITH AN OPEN
         CHEST IN THE MIDDLE."
3690  D$(13)="":RETURN
```

Nothing but Davy Jones' Locker (the CHEST) can be unlocked with the key. Line 3630 tests that the player has, in fact, picked this up. It may be that the player will try to UNLOCK the door in locations three and nine. However, the door is something of a red-herring (all good adventure games have these!), and can never be opened. Line 3640 deals with this situation and tells the player that the key cannot be used.

Control will only pass from line 3650 if the player is in the room with the locker and the noun input is "CHEST." If everything is in order the unlock flag **UF** is tested to ensure that the player has not previously unlocked the chest. Line 3670 sets **UF** to 1 and the key is located in the room, i.e. **OB (7)** is made equal to 13, telling the computer that the key stays in that room. Finally, a message is **PRINT**ed to hint that the action was successful.
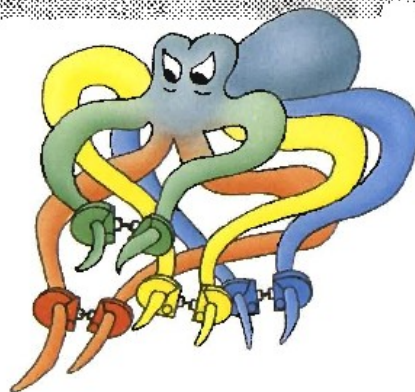
```
3700 REM************USE*********************
3710 IF CP<>20 OR NO<>14 THEN PRINT"YOU CAN'T DO
     THAT HERE":RETURN
3720 IF HC<4 THEN PRINT"YOU HAVEN'T GOT ENOUGH
     PAIRS!":RETURN
3730 PRINT"THE OCTOPUS CAN'T MOVE.HE ISN'T
     AMUSED!"
3740 FOR I=7 TO 10:OB(I)=0:NEXT
3750 HC=0:EX(20,4)=19
3760 T$(20)="YOU ARE IN A CORRIDOR WITH A
     CURRENT GOING WEST"
3770 D$(20)="A COLORFUL, MANACLED OCTOPUS
     SITS SULKING.":RETURN
```

The "USE" verb is designed to allow the player to immobilize the octopus in location 20 with four sets of handcuffs – the instruction is "USE HAN."

Line 3710 checks that the player is in the correct position and that the noun number corresponding to the handcuffs (NO = 14) is valid. Line 3720 ensures that the player has all four pairs of handcuffs. If control drops through to line 3730, a message indicates that the octopus has been manacled. The handcuffs are removed from the player using a FOR ... NEXT loop to set the object position to zero in line 3740. Finally, the number of handcuffs is made zero, a new exit becomes available to the west and the location description of the room T$ (20) and D$ (20), is changed.

The final three parts of the program make up the very useful commands of "INVENTORY," "HELP" and "LOOK," without which the game would be extremely difficult to play.

```
3800 REM ********INVENTORY***********
3810 PRINT:PRINT "YOU ARE CARRYING:-"
3820 IF IN=0 THEN PRINT"NOTHING!":RETURN
3830 FOR I=1 TO 10
3840 IF OB(I)=99 THEN PRINT"A ";OO$(I)
3850 NEXT: RETURN
```

The INVENTORY routine tells the player which objects he or she is carrying. If IN contains 0 then a command to "INVENTORY" would produce the words "YOU ARE CARRYING – NOTHING!" on the screen, printed by lines 3810 and 3820. However, if IN is not equal to 0, a loop starting at line 3830 checks each object to see if it is carried and will PRINT these on the screen if this is so.

```
3900 REM **********HELP************
3910 PRINT:PRINT"THESE ARE THE VERBS YOU MAY
     USE:-"
3920 FOR I=1 TO VV:PRINT VB$(I),:NEXT I
3930 PRINT:PRINT "(YOU NEED ONLY TYPE THE
     FIRST THREE LETTERS)":RETURN
```
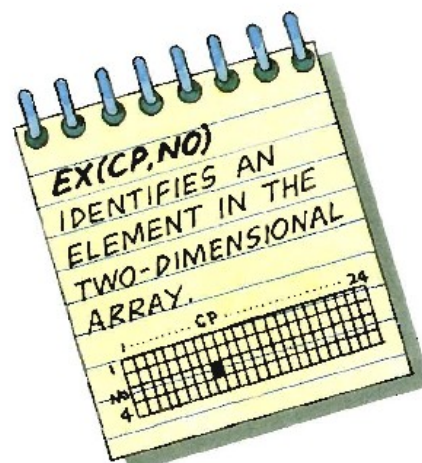
At the command "HELP," the three lines of program 3910 to
3930 display each of the verbs on the screen to enable the
player to construct acceptable two-word display sentences.
A FOR ... NEXT loop in line 3920 is used to do this with the
maximum value of I set to the maximum number of verbs
available – in this case, the value of VV.

The last section of the program to be dealt with concerns the
verb "LOOK", which allows the player to call up the location
descriptions of the room he or she is in. If you have followed
the instructions on page 24 (in the RUN box), you will have
already typed this part of the program. This section is
different from the other verb routines because it is used as
the end of the INITIALIZE subroutine, linked by a GOSUB
statement in line 5190.

Line 4010 PRINTs the location descriptions T$ (CP) and D$
(CP). The lost flag LF is set if the player has ignored the
warning above the south exit of location 16 and entered the
SHARK ROOM (location 22).

Lines 4020 to 4050 PRINT the object found in the location
occupied by the player. Note that the mention of the OLD
BONE is carefully sidestepped in line 4030 to make the game
more difficult. Lastly, the available exits from the room the
player is in, are PRINTed in lines 4060 to 4100.

EX(CP,NO) IDENTIFIES AN ELEMENT IN THE TWO-DIMENSIONAL ARRAY. 24

```
4000 REM *********LOOK*************
4010 PRINT:PRINT T$(CP):PRINT D$(CP):IF CP=22
     THEN LF=1:RETURN
4020 FOR I=1 TO 10
4030 IF I=5 AND CP=15 THEN 4050
4040 IF OB(I)=CP THEN PRINT"A ";OB$(I);" IS HERE"
4050 NEXT I
4060 PRINT:PRINT"EXITS:   ";
4070 IF EX(CP,1)>0 THEN PRINT"NORTH  ";
4080 IF EX(CP,2)>0 THEN PRINT"SOUTH  ";
4090 IF EX(CP,3)>0 THEN PRINT"EAST  ";
4100 IF EX(CP,4)>0 THEN PRINT"WEST  ";
4110 PRINT:PRINT:PRINT:RETURN
```
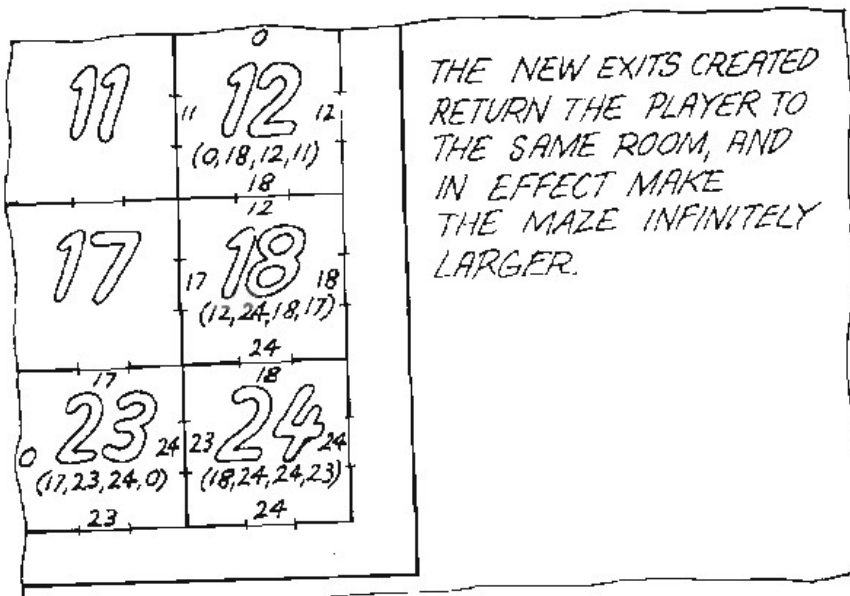
Improve your program

The program that you have keyed in so far will give you a
challenging adventure game. If you or your friends get stuck,
look back to the game plan on page 8 to find your way
through. But once you have stored the basic game on tape or
disk, you can make changes and improvements. For
example, you could clear the screen when the game is won
or lost and design a more effective message to appear. The
additions highlighted below create a maze from the rooms
already given in the game plan.

Creating a maze

Many adventure games contain a maze as one more obstacle
for the player to overcome. In this program, all you have to
do is add extra exits to rooms 12, 18, 23, and 24. The revised
DATA statements given below do this. In effect, all that
happens is that when you take the wrong exits from these
rooms, you end up where you started from. If you refer
to the game plan and add these additional exits to the
east and south you should be able to discover the only
way out. Why not try to create a maze with some of the
other rooms?

```
6240  DATA  0,15,0,0,  0,16,0,0,  5,17,12,0,  0,18,12,11
6250  DATA  7,0,0,0,  8,0,15,0,  9,21,16,14,  10,22,17,15
6260  DATA  11,23,18,16,  12,24,18,17,  0,0,20,0,  0,0,21,0
6270  DATA  15,0,0,20,  16,0,0,0,  17,23,24,0,  18,24,24,23
```


THE NEW EXITS CREATED
RETURN THE PLAYER TO
THE SAME ROOM, AND
IN EFFECT MAKE
THE MAZE INFINITELY
LARGER.

The complete listing

The complete listing for our adventure game - Neptune's Caverns - is given below, for both the Apple IIe and the Commodore 64 computers. These listings include all the additional sections of program given on the opposite page, so they contain the complete version of Neptune's Caverns.

APPLE IIe

```
10   REM    NEPTUNE'S CAVERNS
20   GOSUB 5000: REM   INITIALIZE
30   GOSUB 1000: REM   INPUT
40   GOSUB 2000: REM   SORT
50   IF WF = 0 AND LF = "" THEN GOTO 30
60   GOSUB 5500: REM   END
70   END
1000 REM    INPUT
1010 VB$ = "":NO$ = ""
1020 INPUT "WHAT DO YOU DO NEXT ?";R$
1030 FOR I = 1 TO LEN (R$)
1040 IF  MID$ (R$,I,1) = " " THEN VB$ =  LEFT$ (R$,I):
     :NO$ =  MID$ (R$,I + 1,3):I =  LEN (R$)
1050 NEXT I
1060 IF NO$ < > "" THEN  RETURN
1070 R$ =  LEFT$ (R$,3)
1080 IF R$ = "NOR" OR R$ = "SOU" OR R$ = "EAS" OR R$
     = "WES" THEN VB$ = "GO ":NO$ = R$: RETURN
1090 IF R$ = "HEL" OR R$ = "INV" OR R$ = "LOO" THEN
     NO$ = "DOO":VB$ = R$: RETURN
1100 PRINT "I DON'T UNDERSTAND THAT"
1110 GOTO 1010
2000 REM    SORT
2010 VB = 0:NO = 0: FOR I = 1 TO VV
2020 IF VB$ =  LEFT$ (VB$(I),3) THEN VB = I:I = VV
2030 NEXT I
2040 FOR I = 1 TO NN
2050 IF NO$ = NO$(I) THEN NO = I:I = NN
2060 NEXT I
2070 IF NO = 0 OR VB = 0 THEN  PRINT "I DON'T
     UNDERSTAND THAT ": RETURN
2080 ON VB GOTO 3000,3100,3200,3300,3400,3500,3600,37
     00,3800,3900,4000
3000 REM ************GO***************
3010 IF NO > 4 THEN  PRINT "GO WHERE ?": RETURN
3020 IF EX(CP,NO) = 0 THEN  PRINT "NO EXIT THAT WAY"
     : RETURN
3030 IF CP = 16 AND (NO = 1 OR NO = 4) AND FF = 0
     THEN LF = 1: RETURN
3040 CP = EX(CP,NO): GOTO 4000
3100 REM ***********GET*************
3110 IF NO = 6 AND CP = 13 THEN  PRINT "IT'S TOO
     HEAVY !": RETURN
3120 IF NO < 8 THEN  PRINT "DON'T BE SILLY": RETURN
3130 IF OB(NO - 7) = 99 AND NO$ < > "HAN" THEN
     PRINT "YOU'VE ALREADY GOT IT !": RETURN
3140 FOR I = 7 TO 10
3150 IF OB(I) = CP AND NO = 14 THEN NO = I + 7:HC = H
     C + 1:I = 10
3160 NEXT I
3170 IF OB(NO - 7) = CP THEN  PRINT "O.K.":OB(NO - 7)
     = 99:IN = IN + 1: RETURN
3180 PRINT "IT ISN'T HERE !": RETURN
3200 REM ***********DROP***********
3210 IF NO < 8 THEN  PRINT "DON'T BE SILLY !": RETURN
3220 FOR I = 7 TO 10
3230 IF OB(I) = 99 AND NO = 14 THEN NO = I + 7:HC = H
     C - 1:I = 10
3240 NEXT I
3250 IF OB(NO - 7) < > 99 THEN  PRINT "YOU HAVEN'T
     GOT IT":RETURN
3260 PRINT "O.K.":OB(NO - 7) = CP:IN = IN - 1
3270 IF CP = 19 AND NO = 13 THEN WF = 1
3280 RETURN
3300 REM ***********CUT***********
3310 IF OB(1) < > 99 THEN  PRINT "YOU'VE NOTHING
     SHARP ENOUGH !": RETURN
3320 IF NO < > 11 THEN  PRINT "YOU CAN'T CUT THAT !"
     : RETURN
3330 IF CP < > 5 THEN  PRINT "YOU CAN'T DO THAT !":
     RETURN
3340 IF CF = 1 THEN  PRINT "YOU'VE ALREADY DONE THAT
     !": RETURN
```

COMMODORE 64

```
10 REM NEPTUNE'S CAVERNS
20 GOSUB 5000: REM INITIALIZE
30 GOSUB 1000: REM INPUT
40 GOSUB 2000: REM SORT
50 IF WF<>1 AND LF<>1 THEN GOTO 30
60 GOSUB 5500: REM END
70 END
1000 REM *******INPUT **************
1010 VB$="":NO$="":R$=""
1020 INPUT"WHAT DO YOU DO NEXT";R$
1030 FOR I=1 TO LEN(R$)
1040 IF MID$(R$,I,1)=" " THEN VB$=LEFT$(R$,3):
     NO$=MID$(R$,I+1,3): I=LEN(R$)
1050 NEXT I
1060 IF NO$<>"" THEN RETURN
1070 R$=LEFT$(R$,3)
1080 IF R$="NOR"OR R$="SOU"OR R$="EAS" OR
     R$="WES" THEN VB$="GO ":NO$=R$:RETURN
1090 IF R$="HEL" OR R$="INV" OR R$="LOO"
     THEN NO$="DOO":VB$=R$:RETURN
1100 PRINT"I DON'T UNDERSTAND THAT"
1110 GOTO 1010
2000 REM******SORT***************
2010 VB=0: NO=0
2020 FOR I=1 TO VV
2030 IF VB$=LEFT$(VB$(I),3) THEN VB=I:I=VV
2040 NEXT
2050 FOR I=1 TO NN
2060 IF NO$=NO$(I) THEN NO=I:I=NN
2070 NEXT
2080 IF NO=0 OR VB=0 THEN PRINT"I DON'T
     UNDERSTAND THAT":RETURN
2090 ON VB GOTO 3000,3100,3200,3300,3400,3500,3600
     ,3700,3800,3900,4000
3000 REM***********GO*****************
3010 IF NO>4 THEN PRINT "GO WHERE?":RETURN
3020 IF EX(CP,NO)=0 THEN PRINT"NO EXIT THAT
     WAY":RETURN
3030 IF CP=16 AND (NO=1 OR NO=4) AND FF=0 THEN LF=1:
     RETURN
3040 CP=EX(CP,NO): GOSUB 4000
3050 RETURN
3100 REM***********GET***************
3110 IF NO=6 AND CP=13 THEN PRINT"IT'S TOO
     HEAVY!":RETURN
3120 IF NO<8 THEN PRINT "DON'T BE SILLY":RETURN
3130 IF OB(NO-7)=99 AND NO$<>"HAN" THEN
     PRINT"YOU'VE ALREADY GOT IT!":RETURN
3140 FOR I=7 TO 10
3150 IF (OB(I)=CP AND NO=14) THEN NO=I+7:HC=HC+1:I=10
3160 NEXT I
3170 IF OB(NO-7)=CP THEN PRINT"O.K.": OB(NO-7)=99:
     IN=IN+1:RETURN
3180 PRINT"IT ISN'T HERE!":RETURN
3200 REM*********DROP***************
3210 IF NO<8 THEN PRINT"DON'T BE SILLY":RETURN
3220 FOR I=7 TO 10
3230 IF (OB(I)=99 AND NO=14) THEN NO=I+7:HC=HC-1:I=10
3240 NEXT I
3250 IF OB(NO-7)<>99 THEN PRINT"YOU HAVEN'T GOT
     IT!":RETURN
3260 PRINT"O.K.": OB(NO-7)=CP: IN=IN-1
3270 IF CP=19 AND NO=13 THEN WF=1
3280 RETURN
3300 REM**********CUT**************
3310 IF OB(1)<>99 THEN PRINT"YOU'VE NOTHING
     SHARP ENOUGH!":RETURN
3320 IF NO<>11 THEN PRINT"YOU CAN'T CUT THAT!":
     RETURN
3330 IF CP<>5 THEN PRINT"YOU CAN'T DO THAT!":
     RETURN
3340 IF CF=1 THEN PRINT"YOU'VE ALREADY DONE
     THAT!":RETURN
```

APPLE IIe:

```
3350 PRINT "THE SEAWEED FALLS AWAY TO REVEAL AN
     OPEN WINDOW"
3360 OB(4) = 5:EX(5,2) = 11:CP = 1
3370 LO$(5) = LEFT$ (LO$(5),59) + "AN OPEN WINDOW IN
     IT ": RETURN
3400 REM ***********WEAR************
3410 IF FF = 1 AND NO$ = "FLI" THEN PRINT "YOU'VE
     ALREADY GOT THEM ON!": RETURN
3420 IF OB(2) = 99 AND NO$ = "FLI" THEN FF = 1:OB(2)
     = 0: PRINT "THEY FIT NICELY!": RETURN
3430 IF NO = 14 AND HC > 0 THEN PRINT "THAT'S REALLY
     SILLY!": RETURN
3440 PRINT "YOU CAN'T WEAR THAT!": RETURN
3500 REM ***********GIVE************
3510 IF CP = 10 AND NO = 11 THEN 3550
3520 IF CP = 7 AND NO = 12 THEN 3580
3530 IF CP = 20 THEN PRINT "THAT WON'T DO ANY
     GOOD!": RETURN
3540 PRINT "NOTHING HERE WANTS IT": RETURN
3550 IF OB(4) <> 99 THEN PRINT "YOU HAVEN'T GOT
     IT": RETURN
3560 OB(4) = 0:OB(3) = 10
3570 PRINT "THEY CROWD ROUND YOU AND SOMETHING
     GLINTS IN THE CORNER": RETURN
3580 IF OB(5) <> 99 THEN PRINT "YOU HAVEN'T GOT
     IT": RETURN
3590 OB(5) = 0:EX(7,2) = 11
3600 PRINT "THE FISH SNATCHES THE BONE AND RETIRES TO
     A CORNER"
3610 LO$(7) = "YOU ARE IN A DIM CAVERN, A BONEFISH IS
     IN THE CORNER GNAWING ON A THIGH BONE": RETURN
3620 REM ***********UNLOCK************
       .  .  .
3800 REM ***********INVENTORY************
3810 PRINT : PRINT "YOU ARE CARRYING:-"
3820 IF IN = 0 THEN PRINT "NOTHING!": RETURN
3830 FOR I = 1 TO 10
3840 IF OB(I) = 99 THEN PRINT "A ";OB$(I)
3850 NEXT I: RETURN
3900 REM ***********HELP************
3910 PRINT : PRINT "THESE ARE THE VERBS YOU MAY USE"
3920 FOR I = 1 TO VV: PRINT VB$(I);: NEXT
3930 PRINT "(YOU NEED ONLY TYPE THE FIRST THREE
     LETTERS)": RETURN
4000 REM ***********LOOK************
4010 PRINT : PRINT LO$(CP): IF CP = 22 THEN LF = 1:
     RETURN
4020 FOR I = 1 TO 10
4030 IF VI = 5 AND CP = 15 THEN 4050
4040 IF OB(I) = CP THEN PRINT "A ";OB$(I);" IS HERE"
4050 NEXT I
4060 PRINT : PRINT "EXITS: ";
4070 IF EX(CP,1) > 0 THEN PRINT "NORTH   ";
4080 IF EX(CP,2) > 0 THEN PRINT "SOUTH   ";
4090 IF EX(CP,3) > 0 THEN PRINT "EAST    ";
4100 IF EX(CP,4) > 0 THEN PRINT "WEST    ";
4110 PRINT : PRINT : RETURN
5000 REM ***********INITIALIZE************
5010 HOME : PRINT TAB(10);"NEPTUNE'S CAVERNS"
5020 PRINT : PRINT "   YOU HAVE FOUND THE MAGIC PLUG"
5030 PRINT " THAT BELONGS AT THE BOTTOM OF THE SEA"
5040 PRINT " AND DECIDE TO REPLACE IT BEFORE THE"
5050 PRINT " WATER DRAINS AWAY. WITH YOUR SCUBA GEAR"
5060 PRINT " YOU DIVE INTO THE OCEAN AND BEGIN YOUR"
5070 PRINT " ADVENTURE..."
5080 PRINT TAB(5);"IF YOU NEED ASSISTANCE TYPE
     HELP"
5090 NN = 14:VV = 11:CP = 3:IN = 1:HC = 0:WF = 0:LF =
     0:UF = 0:FF = 0:LF = 0
5100 DIM LO$(24): DIM EX(24,4): DIM NO$(NN): DIM VB$(
     VV): DIM OB$(10): DIM OB(10)
5110 FOR I = 1 TO 24
5120 IF I = 17 OR I = 18 OR I = 22 OR I = 24 THEN LO$
     (I) = LO$(12): GOTO 5140
5130 READ D$:LO$(I) = D$
5140 NEXT I
5150 FOR I = 1 TO 24: FOR J = 1 TO 4: READ EX(I,J):
     NEXT J: NEXT I
5160 FOR I = 1 TO 10: READ OB$(I): READ OB(I): NEXT I
5170 FOR I = 1 TO NN: READ NO$(I): NEXT I
5180 FOR I = 1 TO VV: READ VB$(I): NEXT I
5190 GOSUB 4000: RETURN : REM LOOK
5500 REM END
5510 IF CP = 22 THEN PRINT "WITH ONE SWIPE, SNAP THE
     SHARK BITES OFF YOUR HEAD": GOTO 5540
```

COMMODORE 64:

```
3350 PRINT"THE SEAWEED FALLS AWAY TO REVEAL
     AN OPEN WINDOW."
3360 OB(4)=5: EX(5,2)=11:CP=1
3370 D$(5)=LEFT$(D$(5),19)+"AN OPEN WINDOW IN
     IT":RETURN
3400 REM ***********WEAR************
3410 IF FF=1 AND NO$="FLI" THEN PRINT"YOU'VE
     ALREADY GOT THEM ON!":RETURN
3420 IF OB(2)=99 AND NO$="FLI" THEN FF=1:OB(2)=0:
     PRINT"THEY FIT NICELY!":RETURN
3430 IF NO=14 AND HC>0 THEN PRINT"THAT'S REALLY
     SILLY!":RETURN
3440 PRINT"YOU CAN'T WEAR THAT!":RETURN
3500 REM ***********GIVE************
3510 IF CP=10 AND NO=11 THEN 3550
3520 IF CP=7 AND NO=12 THEN 3580
3530 IF CP=20 THEN PRINT"THAT WON'T DO ANY
     GOOD!":RETURN
3540 PRINT"NOTHING HERE WANT'S IT":RETURN
3550 IF OB(4)<>99 THEN PRINT"YOU HAVEN'T GOT
     IT":RETURN
3560 OB(4)=0:OB(3)=10
3570 PRINT"SOMETHING GLINTS IN THE CORNER":
     RETURN
3580 IF OB(5)<>99 THEN PRINT"YOU HAVEN'T GOT IT":
     RETURN
3590 OB(5)=0:EX(7,2)=11
3600 PRINT"THE FISH SNATCHES THE BONE AND
     RETIRES TO A CORNER"
3610 T$(7)="YOU ARE IN A CAVERN, IN THE
     CORNER A"
3615 D$(7)="BONEFISH IS CHEWING A THIGH
     BONE":RETURN
3620 REM***********UNLOCK************
3630 IF OB(1)<>99 THEN PRINT"YOU HAVEN'T EVEN
     GOT A KEY":RETURN
3640 IF NO=5 AND (CP=5 OR CP=9) THEN PRINT"THERE
     ISN'T EVEN A KEYHOLE!":RETURN
3650 IF CP=13 OR NO=6 THEN PRINT"YOU CAN'T DO
     THAT":RETURN
3660 IF UF=1 THEN PRINT"IT'S ALREADY
     UNLOCKED!":RETURN
3670 UF=1:OB(7)=13:PRINT"THE KEY TURNS EASILY"
3680 T$(13)="YOU ARE IN A ROOM WITH AN OPEN
     CHEST IN THE MIDDLE."
3690 D$(13)="":RETURN
3700 REM***********USE************
3710 IF CP=20 OR NO=14 THEN PRINT"YOU CAN'T DO
     THAT HERE":RETURN
3720 IF HC<4 THEN PRINT"YOU HAVEN'T GOT ENOUGH
     PAIRS!":RETURN
3730 PRINT"THE OCTOPUS CAN'T MOVE,HE ISN'T
     AMUSED!"
3740 FOR I=7 TO 10:OB(I)=0:NEXT
3750 HC=0:EX(20,4)=19
3760 T$(20)="YOU ARE IN A CORRIDOR WITH A
     CURRENT GOING WEST"
3770 D$(20)="A COLORFUL, MANACLED OCTOPUS
     SITS SULKING.":RETURN
3800 REM ***********INVENTORY************
3810 PRINT:PRINT "YOU ARE CARRYING:-"
3820 IF IN=0 THEN PRINT"NOTHING!":RETURN
3830 FOR I=1 TO 10
3840 IF OB(I)=99 THEN PRINT"A ";OB$(I)
3850 NEXT: RETURN
3900 REM ***********HELP************
3910 PRINT:PRINT"THESE ARE THE VERBS YOU MAY
     USE:-"
3920 FOR I=1 TO VV:PRINT VB$(I),,:NEXT I
3930 PRINT:PRINT "(YOU NEED ONLY TYPE THE
     FIRST THREE LETTERS)":RETURN
4000 REM ***********LOOK************
4010 PRINT:PRINT T$(CP):PRINT D$(CP):IF CP=22
     THEN LF=1:RETURN
4020 FOR I=1 TO 10
4030 IF I=5 AND CP=15 THEN 4050
4040 IF OB(I)=CP THEN PRINT"A ";OB$(I);" IS HERE"
4050 NEXT I
4060 PRINT:PRINT"EXITS:   ";
4070 IF EX(CP,1)>0 THEN PRINT"NORTH   ";
4080 IF EX(CP,2)>0 THEN PRINT"SOUTH   ";
4090 IF EX(CP,3)>0 THEN PRINT"EAST    ";
4100 IF EX(CP,4)>0 THEN PRINT"WEST    ";
4110 PRINT:PRINT:PRINT:RETURN
5000 REM ***********INITIALIZE************
5010 PRINT"..."
5020 PRINT:SPC(12)"NEPTUNE'S CAVERNS"
5030 PRINT" YOU HAVE FOUND THE MAGIC PLUG THAT"
5040 PRINT"BELONGS AT THE BOTTOM OF THE SEA, AND"
5050 PRINT"DECIDE TO REPLACE IT BEFORE THE WATER"
5060 PRINT"DRAINS AWAY. WITH YOUR SCUBA GEAR YOU"
5070 PRINT"DIVE INTO THE OCEAN AND BEGIN YOUR"
5080 PRINT"ADVENTURE..."
5090 PRINT "(IF YOU NEED ASSISTANCE TYPE HELP)"
5100 NN=14:VV=11:CP=3:IN=1:HC=0:WF=0:LF=0:UF=0:
     FF=0:CF=0
5110 DIM T$(24):DIM D$(24):DIM EX(24,4):DIM NO$(NN):
     DIM VB$(VV)
5120 DIM OB$(10):DIM OB(10)
5130 FOR I=1 TO 24
5140 IF I=17 OR I=18 OR I=23 OR I=24 THEN
     T$(I)=T$(12):D$(I)=D$(12): GOTO 5160
5150 READ T$,D$:T$(I)=T$:D$(I)=D$
```

APPLE IIe continued

```
5520 IF CP = 16 THEN PRINT "THE SPINES ON THE SEA-
URCHINS ARE VERY POISONOUS! YOU DIE A HORRIBLE DEATH
NEXT TIME TAKE PRECAUTIONS!": GOTO 5540
5530 PRINT "WITH A THUNK THE PLUG DROPS INTO THE H
OLE AND THE SWIRLING WATERS GROW STILL CONGRATULATIONS
! YOU SAVED THE SEAS !
5540 INPUT "DO YOU WISH TO PLAY AGAIN ";R1
5550 IF LEFT$ (R1,1) = "Y" THEN RUN
5560 PRINT "BYE-BYE": RETURN
6000 REM ********DESCRIPTIONS*********
6010 DATA "YOU ARE ON THE SEABED.THE WAY WEST IS
BLOCKED BY A HIGH CORAL REEF."
6020 DATA "YOU ARE ON THE SEABED.TO THE SOUTH A
BARNACLED WALL TOWERS ABOVE YOU."
6030 DATA "YOU ARE IN FRONT OF A WOODEN DOOR. YOU
CAN SEE NO HANDLE."
6040 DATA "YOU ARE ON THE SEABED.TO THE SOUTH A
BARNACLED WALL TOWERS ABOVE YOU."
6050 DATA "YOU ARE ON THE SEABED. TO THE SOUTH A
BARNACLED WALL HAS A SQUARE PATCH OF SEAWEED GROWING
ON IT."
6060 DATA "YOU ARE ON THE SEABED.TO THE SOUTH IS A
BARNACLED WALL.A CLIFF BLOCKS THE WAY EAST."
6070 DATA "YOU ARE IN A LONG, LOW CAVERN. AT THE
FAR END A LARGE BONEFISH IS SWIMMING AROUND."
6080 DATA "YOU ARE IN A BRIGHTLY LIT CHAMBER.THE
WALLS,FLOOR AND ROOF GLOW IN SHIMMERING LIGHT."
6090 DATA "YOU ARE IN A DIMLY LIT CAVERN WITH A
HUGE DOOR AT THE FAR END.YOU CAN SEE NO HANDLE."
6100 DATA "YOU ARE IN A ROOM FULL OF HUNGRY
SEAHORSES! THEY NUZZLE YOUR HAND IN A FRIENDLY MANNER"
6110 DATA "YOU ARE IN A SMALL ROOM.THE NORTH WALL
HAS A SMALL WINDOW IN IT THROUGH WHICH YOU CAN SEE
THE SEABED"
6120 DATA "YOU ARE IN AN AM-A-Z-INGLY SQUARE ROOM.
THE WALLS,FLOOR AND ROOF ARE ALL SQUARE AS ARE ALL THE
EXITS"
6130 DATA "YOU ARE IN A TINY LITTLE ROOM THAT IS
OCCUPIED BY A CHEST INSCRIBED WITH THE INITIALS D.J."
6140 DATA "YOU ARE IN A COLD,MURKY ROOM.GREY MUD
SWIRLS AROUND YOU AND YOU FEEL A FAINT CURRENT EAST"
6150 DATA "YOU ARE IN A GLOOMY AND EERIE PLACE.ALL
AROUND YOU ARE THE BONES OF LONG DEAD EXPLORERS!"
6160 DATA "YOU ARE IN A SQUARE ROOM.THE SOUTH EXIT
HAS THE WORDS 'DO NOT ENTER' ABOVE IT.THE NORTH AND
WEST DOORWAYS ARE CRAWLING WITH SEA URCHINS"
6170 DATA "YOU ARE IN A CIRCULAR ROOM WITH A VERY
STRONG CURRENT THAT SWIRLS AROUND THE ROOM AND DOWN A
HOLE IN THE FLOOR"
6180 DATA "YOU ARE IN A CORRIDOR WITH A STRONG
CURRENT GOING WEST.YOUR WAY IS BLOCKED BY THE ARMS OF
A HUGE RAINBOW COLORED OCTOPUS."
6190 DATA "YOU'RE IN A SHIPWRECKED CAPTAIN'S CABIN
AND FEEL THE FLOW OF WATER TO THE WEST"
6200 DATA "YOU SEE A RUSH OF SWIRLING WATER AND FACE
THE JAWS OF A GREAT WHITE SHARK."
6210 REM *********EXITS************
6220 DATA 0,0,2,0, 0,0,3,1, 0,0,4,2, 0,0,5,3
6230 DATA 0,0,6,4, 0,0,0,5, 0,0,8,0, 0,14,0,7
6240 DATA 0,15,0,0, 0,16,0,0, 5,17,12,0, 0,18,12,11
6250 DATA 7,0,0,0, 8,0,15,0, 9,21,16,14, 10,22,17,
15
6260 DATA 11,23,18,16, 12,24,18,17, 0,0,20,0, 0,0
,21,0
6270 DATA 15,0,0,20, 0,0,0,0, 17,23,24,0, 18,24,
24,23
6300 REM *******OBJECTS************
6310 DATA KNIFE,2
6320 DATA PAIR OF FLIPPERS,6
6330 DATA KEY,0
6340 DATA CLUMP OF SEAWEED,0
6350 DATA ROTTEN OLD BONE,15
6360 DATA MAGIC PLUG,99
6370 DATA YELLOW PAIR OF HANDCUFFS,0
6380 DATA GREEN PAIR OF HANDCUFFS,9
6390 DATA RED PAIR OF HANDCUFFS,17
6400 DATA BLUE PAIR OF HANDCUFFS,11
6500 REM *********NOUNS***********
6510 DATA NOR,SOU,EAS,WES,DOO,CHE,WIN,KNI,FLI,KEY,
SEA,BON,PLU,HAN
6520 REM *********VERBS***********
6530 DATA GO,GET,DROP,CUT,WEAR,GIVE,UNLOCK,USE,
INVENTORY,HELP,LOOK
```

COMMODORE 64 continued

```
5160 NEXT I
5170 FOR I=1 TO 24:FOR J=1 TO 4:READ EX(I,J):NEXT J,I
5180 FOR I=1 TO 10:READ OB$(I),OB(I):NEXT I
5190 FOR I=1 TO NN:READ NO$(I):NEXT I
5200 FOR I=1 TO VV:READ VB$(I):NEXT I
5210 GOSUB 4000: RETURN: REM LOOK
5500 REM ***********END**************
5510 IF CP=22 THEN PRINT"WITH A SNAP.THE SHARK BITES
OFF YOUR HEAD":GOTO5570
5520 IF CP=16 THEN PRINT"YOU HAVE STEPPED ON A
POISONOUS SEA-URCHIN"
5530 IF CP=16 THEN PRINT"YOU DIE A HORRIBLE
DEATH.": GOTO 5570
5540 PRINT"WITH A 'THUNK' THE PLUG DROPS INTO THE"
5550 PRINT"HOLE AND THE SWIRLING WATERS GROW STILL"
5560 PRINT"CONGRATULATIONS, YOU SAVED THE SEAS!"
5570 INPUT"DO YOU WISH TO PLAY AGAIN";R1
5580 IF LEFT$(R1,1)="Y" THEN RUN
5590 PRINT"BYE-BYE":RETURN
6000 REM*********DESCRIPTIONS***********
6010 DATA "YOU ARE ON THE SEABED.THE WAY WEST IS"
6011 DATA "BLOCKED BY A HIGH CORAL REEF"
6020 DATA "YOU ARE ON THE SEABED.TO THE SOUTH A"
6021 DATA "BARNACLED WALL TOWERS ABOVE YOU."
6030 DATA "YOU ARE IN FRONT OF A WOODEN DOOR.YOU"
6031 DATA "CAN SEE NO HANDLE."
6040 DATA "YOU ARE ON THE SEABED.TO THE SOUTH A"
6041 DATA "BARNACLED WALL TOWERS ABOVE YOU."
6050 DATA "YOU ARE ON THE SEABED.TO THE SOUTH A"
6051 DATA "BARNACLED WALL HAS A SQUARE PATCH OF
SEAWEED GROWING ON IT."
6060 DATA "YOU ARE ON THE SEABED.TO THE SOUTH"
6061 DATA "IS A BARNACLED WALL.A CLIFF BLOCKS THE
WAY EAST."
6070 DATA "YOU ARE IN A LONG,LOW CAVERN.AT THE FAR"
6071 DATA "END A LARGE BONEFISH IS SWIMMING AROUND."
6080 DATA "YOU ARE IN A BRIGHTLY LIT CHAMBER.THE"
6081 DATA "WALLS,FLOOR AND ROOF GLOW IN SHIMMERING
LIGHT."
6090 DATA "YOU ARE IN A DIMLY LIT CAVERN WITH A"
6091 DATA "HUGE DOOR AT THE FAR END.YOU CAN SEE NO
HANDLE."
6100 DATA "YOU ARE IN A ROOM FULL OF HUNGRY"
6101 DATA "SEAHORSES.THEY NUZZLE YOUR HAND IN A
FRIENDLY MANNER."
6110 DATA "YOU ARE IN A SMALL ROOM.THE NORTH WALL"
6111 DATA "HAS A SMALL WINDOW IN IT THROUGH WHICH
YOU CAN SEE THE SEABED"
6120 DATA "YOU ARE IN AN AM-A-Z-E-NGLY SQUARE ROOM"
6121 DATA "THE WALLS,FLOOR AND ROOF ARE ALL SQUARE
AS ARE ALL THE EXITS."
6130 DATA "YOU ARE IN A TINY LITTLE ROOM THAT IS"
6131 DATA "OCCUPIED BY A CHEST INSCRIBED WITH THE
INITIALS D.J."
6140 DATA "YOU ARE IN A COLD,MURKY ROOM.GREY MUD"
6141 DATA "SWIRLS AROUND YOU AND YOU FEEL A FAINT
CURRENT TO THE EAST."
6150 DATA "YOU ARE IN A GLOOMY AND EERIE PLACE.ALL"
6151 DATA "AROUND YOU ARE THE BONES OF LONG DEAD
EXPLORERS!"
6160 DATA "YOU ARE IN A SQUARE ROOM.THE SOUTH EXIT
HAS THE WORDS 'DO NOT ENTER'"
6161 DATA "ABOVE IT.THE NORTH AND WEST DOORWAYS ARE
CRAWLING WITH SEA URCHINS."
6170 DATA "YOU ARE IN A CIRCULAR ROOM WITH A VERY
STRONG CURRENT THAT SWIRLS"
6171 DATA "AROUND THE ROOM AND DOWN A HOLE IN THE
FLOOR."
6180 DATA "YOU ARE IN A CORRIDOR WITH A STRONG
CURRENT GOING WEST. YOUR WAY"
6181 DATA "IS BLOCKED BY THE ARMS OF A LARGE
RAINBOW COLORED OCTOPUS."
6190 DATA "YOU'RE IN A SHIPWRECKED CAPTAIN'S CABIN"
6191 DATA "YOU FEEL THE FLOW OF WATER TO THE WEST."
6200 DATA "YOU SEE A RUSH OF SWIRLING WATER AND"
6201 DATA "FACE THE JAWS OF A GREAT WHITE SHARK."
6210 REM*********EXITS************
6220 DATA 0,0,2,0, 0,0,3,1, 0,0,4,2, 0,0,5,3
6230 DATA 0,0,6,4, 0,0,0,5, 0,0,8,0, 0,14,0,7
6240 DATA 0,15,0,0, 0,16,0,0, 5,17,12,0, 0,18,0,11
6250 DATA 7,0,0,0, 8,0,15,0, 9,21,16,14, 10,22,17,15
6260 DATA 11,23,18,16, 12,24,0,17, 0,0,20,0, 0,0,21,0
6270 DATA 15,0,0,20, 0,0,0,0, 17,0,24,0, 18,0,0,23
6300 REM*********OBJECTS************
6310 DATA KNIFE,2
6320 DATA PAIR OF FLIPPERS,6
6330 DATA KEY,0
6340 DATA CLUMP OF SEAWEED,0
6350 DATA ROTTEN OLD BONE,15
6360 DATA MAGIC PLUG,99
6370 DATA YELLOW PAIR OF HANDCUFFS,0
6380 DATA GREEN PAIR OF HANDCUFFS,9
6390 DATA RED PAIR OF HANDCUFFS,17
6400 DATA BLUE PAIR OF HANDCUFFS,11
6500 REM*********NOUNS***********
6510 DATA NOR,SOU,EAS,WES,DOO,CHE,WIN,KNI,FLI,KEY,SEA,
BON,PLU,HAN
6520 REM*********VERBS***********
6530 DATA GO,GET,DROP,CUT,WEAR,GIVE,UNLOCK,USE,
INVENTORY,HELP,LOOK
```

Glossary

Array   An array is a set of data, held together and identified by one variable name (see also the entry for *variable*). One way of imagining an array is as a series of boxes within the computer's memory, with each separate piece of data held in a separate box.

DATA   A list of information that is required by a program. **DATA** can consist of words or numbers, or both together. A program is sent to the **DATA** with the instruction **READ**.

ON N GOTO   This instruction followed by a series of line numbers tells the computer to jump to the Nth line number in the series given. For example if the value of **VB** is 2, then **ON VB GOTO** 1500, 1750, 3900 would send control of the program to line number 1750.

FOR ... NEXT   This is a sequence of commands that are used to make the computer repeat an operation a certain number of times. For example, the instructions **FOR X = 1 TO 5:PRINT 2*X:NEXT X** would cause the computer to display the two times table.

GOSUB   GOSUB XXXX sends control of the program to a subroutine starting at line XXXX. The search for line XXXX starts at line 0 – so the program will run faster if subroutines that are called most often are placed near the start of the listing.

GOTO   This instruction tells the computer to jump directly to the specified line number, missing out any lines in between. It is often used with **IF ... THEN** (see below). Be careful when using **GOTO**s, as it is easy to have the program jumping backwards and forwards so much that the program is difficult to follow.

IF ... THEN   This is used as a way of telling the computer to do something only when certain conditions are true. This instruction often looks something like this: **IF** score = 10 **THEN PRINT** "WELL DONE, YOU'VE WON!!!"

INPUT   This instruction allows the computer to be given information while a program is running. When the computer comes to an **INPUT** instruction it prints a question mark (on some computers a different symbol) to prompt the user and waits for the input to be given.

LEFT$ In BASIC this instruction is used to copy the left part of a string starting at the left hand end. It is followed in brackets

by figures which indicate the particular string and the number of letters to be copied. **RIGHT$** does the same, but starts from the right hand end and **MID$** is used to copy from the middle of the string.

LEN   This a BASIC instruction which counts the number of characters in a string.

LET   This is one way of giving the computer information. In some programs there may be statements such as: X=10
This simply means that the number ten is stored under the label X. It is often clearer to write:
LET X=10
The **LET** statement also gives rise to something that at first sight seems illogical, if not impossible. In many programs you will see things like:
LET X=X+1
Of course, in mathematical terms X can't equal X+1. All this type of statement means is "increase the value of whatever is stored in X by one."

LIST   This makes the computer display whatever program is has in its memory. You can **LIST** single lines, or parts of a program by following the **LIST** command with the appropriate line numbers.

PRINT   This tells the computer to display something on the screen. Letters and symbols that are to be displayed should be enclosed in quotation marks, but numbers need not be.

RETURN   This is the signal to end a subroutine. **RETURN** causes control of the program to go back to the statement following the most recently executed **GOSUB**.

Subroutine   A subroutine is a collection of program instructions making up part of a program and used to perform a specific task. The same subroutine may be called to do the same task at different stages of the main program.

String   A string represented by the symbol **$**, is a series of letters and/or numbers that are enclosed within double quotes. The computer will not try to interpret any information given within a string, and in most cases will simply print it on the screen when required.

Variables   When you give the computer information you have to give it a label under which it is stored. This label is called a variable since the information it contains may change during the course of the program. When you want the computer to

do something with the information, you must refer to it by its label – its variable name. For example, the statement **LET A=6** places 6 under the variable name **A**.

There are two types of variable. A *numeric variable* is one in which the information stored will always be numbers. If the data to be stored consists of letters or words then a *string variable* must be used. The variable name must then be followed by the string sign – **$**. So, for example, if you wanted a name stored the statement would read: **LET N$ – "JAMES"**. String variable information must always be in quotes.

Index

WRITE YOUR OWN PROGRAM

THIS NEW SERIES INTRODUCES THE ART OF PROGRAMMING YOUR COMPUTER. EACH BOOK SHOWS HOW TO STRUCTURE A PROGRAM INTO ROUTINES, AND AT THE SAME TIME EXPLAINS AND ANALYZES WHAT EXACTLY YOU ARE ASKING THE COMPUTER TO DO AND WHY.

AND THERE'S FUN TOO! EACH BOOK CONTAINS ONE OR MORE EXCITING AND ORIGINAL COMPUTER GAMES. THESE CAN BE ADAPTED AND EXTENDED TO DO BIGGER AND BETTER THINGS.

THE BOOKS ARE SPECIFIC TO THE COMMODORE 64 AND APPLE IIe.