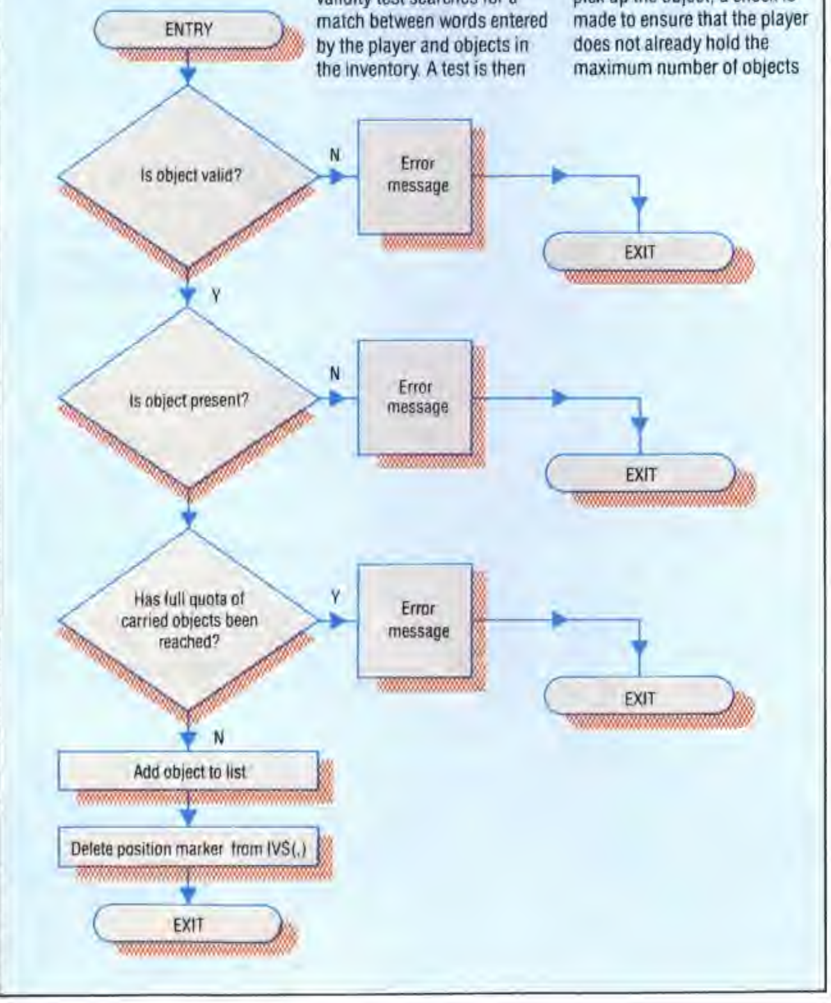
TAKE YOUR PICK

The first thing we did in this project was draw up a map of the imaginary adventure world of our game (see page 766). On this map, we positioned objects that will be of use to the player. Here we show you how to develop the routines needed to allow players to pick up and carry the objects around.

In the last part of the project we looked at command analysis and designated a group of 'normal' commands (see page 813). Included in this group were the commands TAKE and DROP, together with their variations PICK and PUT. Once the appropriate command has been recognised we can construct the routines that obey the command. We will first consider TAKE.

Objective Test

The flowchart for the TAKE routine shows tests used by the routine on the statement entered by the player. The validity test searches for a match between words entered by the player and objects in the inventory. A test is then made to ensure that the object in question is at the player's current location. Finally, before the player is allowed to pick up the object, a check is made to ensure that the player does not already hold the maximum number of objects



To understand the methods employed by the TAKE routine, let's recap on the way that the program keeps track of objects within the adventure world. In the first section of the project we designed DATA statements for each location that contained location descriptions, the names of objects present and information about the possible exits. After the data is read in, the array IV$(,) (used to store the object data for Haunted Forest) has the following contents:

N	IV$(N,1)	IV$(N,2)
1	GUN	10
2	LAMP	9
3	KEY	5

The first column of the array holds the object name, while the second contains its initial location number on the adventure world map. During the description of any location, the second column of this array is scanned to see if any of the objects are at the player's current location, P. When the player wishes to take an object from a location, using a command of the format TAKE THE OBJECT, several factors must be considered:

- Is the object in the command valid; in other words, does it appear in the inventory array, IV$(,)?
- Is the object present at the player's current location?
- Does the player already have the full quota of objects allowed by the game's rules?

If all of these considerations can be answered satisfactorily then the player may take the object. This involves adding the object description to the player's personal object array, IC$(), and deleting the position marker from the relevant entry in IV$(,). Note that the object name does not have to be deleted. If we use a position marker of −1 for each object that has been picked up and carried, then such objects will not appear in location descriptions.It would be rather odd to pick up the GUN from location 10, move to location 9 and then back to location 10, finding on your return that the GUN was still there. Thus the array IV$(,) keeps a record of the positions of all objects *not* being carried by the player. The flowchart for the TAKE routine shows the simple logic that must be applied.

```
3700 REM **** TAKE S/R ****
3710 GOSUB 5300:REM IS OBJECT VALID
3720 IF F=0 THEN SN$="THERE IS NO "+W$:GOSUB5500:
     RETURN
3730 OV=F:GOSUB5450:REM CHECK INVENTORY
3740 IF HF=1 THEN SN$="YOU ALREADY HAVE THE "+IV$
     (F,1):GOSUB5500:RETURN
3750 :
3755 REM ** IS OBJECT HERE ? **
3760 IF VAL(IV$(F,2))<>P THEN SN$=IV$(F,1)+" IS
     NOT HERE":GOSUB5500:RETURN
```