numbers of the locations lying to the north, east, south and west of the current location. In order to determine which exits are possible, the program first splits the eight-digit string into the four numbers that describe which location lies in each direction.

```
2300 REM **** DESCRIBE EXITS S/R ****
2310 EX$=EX$(P)
2320 NR=VAL(LEFT$(EX$,2))
2330 EA=VAL(MID$(EX$,3,2))
2340 SO=VAL(MID$(EX$,5,2))
2350 WE=VAL(RIGHT$(EX$,2))
```

If there is no exit in a given direction, the value assigned is zero — and this is a great help with the description of the exits. A preliminary check must be made to see if any exits are possible before starting to construct the sentence 'There are exits to the...'. This can be done by performing a logical OR on all four direction variables, and this will only produce a zero result if all four direction variables are zero. If this is not the case, then the routine continues to test each direction variable in turn. If the variable is non-zero then the corresponding direction is added to the sentence.

```
2355 IF(NR OR EA OR SO OR WE)=0 THEN RETURN
2360 PRINT:SN$="EXITS ARE TO THE "
2370 IF NR <>0 THEN SN$=SN$+"NORTH "
2380 IF EA <>0 THEN SN$=SN$+"EAST "
2390 IF SO <>0 THEN SN$=SN$+"SOUTH "
2400 IF WE <>0 THEN SN$=SN$+"WEST "
2410 GOSUB 5500:REM FORMAT
2415 PRINT
2420 RETURN
```

Now that we have developed routines that describe each location, we can develop procedures that will allow the player to do things within the world we have created. In a future instalment of the project, we shall be considering more detailed algorithms that analyse instructions. For now, we will deal with the movement instructions the player can issue by simply entering a one word direction command, such as 'NORTH' or 'SOUTH'. If such an instruction is passed to a movement subroutine as the variable NN$, then the movement routine is as follows:

```
3500 REM **** MOVE S/R ****
3510 MF=1:REM SET MOVE FLAG
3520 DR$=LEFT$(NN$,1)
3530 IF DR$<>"N"ANDDR$<>"E"ANDDR$<>"S"ANDDR$<>"W"
  THEN GOTO3590
3540 IF DR$="N"AND NR<>0 THEN P=NR:RETURN
3550 IF DR$="E"AND EA<>0 THEN P=EA:RETURN
3560 IF DR$="S"AND SO<>0 THEN P=SO:RETURN
3570 IF DR$="W"AND WE<>0 THEN P=WE:RETURN
3580 PRINT:PRINT"YOU CAN'T ";IS$
3585 MF=0:RETURN
3590 REM ** NOUN NOT DIRECTION **
3600 PRINT"WHAT IS ";NN$;" ?"
3610 MF=0:RETURN
```

This routine actually uses only the first letter of the direction command passed to it. It begins by checking that the command is, in fact, a direction. If so, the direction specified in the command is acted upon. After ensuring that there is an exit in that direction, P — the variable that keeps track of the player's position — is changed to the value of NR, EA, SO or WE.

Before we can use the subroutines that we have developed here, however, we need to tie them all together to form a repeating loop. The flowchart shows the logical structure of this main calling loop. Although this is not the final structure of the main program loop it serves to demonstrate the aspects of the program covered so far. To use the subroutines given here, insert the following lines, which form a part of the main loop.

```
200 GOSUB6000:REM READ ARRAY DATA
210 P=INT(RND(TI)*10+1):REM START POINT
230 REM **** MAIN LOOP STARTS HERE ****
240 MF=0:REM MOVE FLAG
245 PRINT
250 GOSUB2000:REM DESCRIBE POSITION
255 GOSUB2300:REM DESCRIBE EXITS
260 PRINT:INPUT"INSTRUCTIONS";IS$
```

Also include the following lines in the main calling loop:

```
270 NN$=IS$:GOSUB 3500:REM MOVE
280 GOTO 230:REM RESTART MAIN LOOP
```