32. Queue

General Description

This is the only true computer science program in the book. It simulates how a 'queue' operates in the computer's memory. To remind you: there are two commonly used short-term storage methods you use in computer memories. By far the most common is the 'stack', which acts like a pile of books. The last book placed on the pile is the first book taken off the pile. It is fairly easy to keep track of what constitutes an empty pile. The 'queue', which is like a bus queue in some respects, works on the basis that the first into the queue is the first out of the queue.
Consider the queue below:

(Bus stop) 1 2 3 4 5 6 7

When the bus comes number 1 joints the bus (only one seat free!). Numbers 8 & 9 join the queue, which if the queue does NOT shuffle forward, looks like this.

(Bus stop) 2 3 4 5 6 7 8 9

Another bus comes and this time numbers 2 & 3 join the bus (not the rush hour). If the queue does not shuffle forward it will now look like this.

(Bus stop) 4 5 6 7 8 9

Effectively, as people join the queue and catch the bus, the queue moves backwards down the street. This can happen in computer memory and can be avoided by 'shuffling forward' which in a long queue is very time consuming, or by keeping a 'notice board' of where the next free piece of pavement in the queue is and who is next on the bus, even if they do not 'seem' to be nearest the bus stop. Anyway, run the program and watch. To make the simulation more realistic the queue has been limited. Obviously it is a little inconvenient if a queue can grow so large in memory that it would over-write the program servicing it.

This program runs on a Model 'A'.

Detailed Description

Lines 10-310 Initialise variables and disable cursor and auto repeat. Enter Mode 7. Initialise queue contents to 0,0 and locate random start point.

320-390 Main structure of program - lines 340-370 - clears the buffer and waits for character.

470-630 PROCqueue displays updated queue with its pointers in double characters and prints the value of pointers.

680-990 PROCadd. If queue full prints overflow message. If not expects hexadecimal number. Sound bleep if number out of range. Head of queue printer is updated. Messages and use data deleted from screen.

1040-1180 PROCdelete. If queue empty, underflow message displayed. If not, display register pointing to tail of queue. Clear old pointer.

1240-1290 Deals with overflow or underflow. Clears error message after eight seconds.

Educational Notes

For the good O-level candidates and A-level candidates this is a very clear graphical demonstration. It assumes a knowledge of hexadecimal, as the values to be entered into the queue should be in ' hex'The ' outputegister' always contains the value of the last item taken out of the queue, even if for several goes you then add to the queue. The coloured arrows indicate the head and tail pointers, and the current position of the pointers is also given numerically. Ten minutes experimenting with the simulation has drummed into some of the A-level candidates what a lesson of teaching failed to do.

Program Listing

```
10  REM ****************************
20  REM **   QUEUE                **
30  REM **   Written by Ian Clarke.**
40  REM **   Jan 83               **
```

```
   50  REM ****************************
   60  ON ERROR GOTO 1330
   70  ENVELOPE 1,3,4,-4,0,5,5,1,1,-1,0,-1,120,80
   80  ENVELOPE 3,4,-10,4,-3,20,3,7,2,0,2,-1,120,120
   90  DIM ML$(8),K$(2)
  100  I=0
  110  A=0
  120  MODE 7
  130  VDU 23;8202;0;0;0
  140  *FX4,1
  150  *FX11,0
  160  FOR X=1 TO 8
  170    PRINT TAB(X*4,9);CHR$(132);X
  180    ML$(X)="00"
  190  NEXT X
  200  MA$="Contents of output register."
  210  PRINT TAB(3,3);CHR$(141);CHR$(129);MA$
  220  PRINT TAB(3);CHR$(141);CHR$(129);MA$
  230  P1=RND(8)
  240  P2=P1
  250  PRINT TAB(4,7);CHR$(133);"Head=";P1;
  260  PRINT TAB(17);CHR$(133);"Tail=";P2
  270  A$="^"
  280  FOR X=1 TO 8
  290  NEXT
  300  PRINT TAB(3,18);CHR$(130);"To add to the queue type '
1'."
  310  PRINT TAB(1);CHR$(131);"To delete from the queue type
 '2'."
  320  PROCqueue
  330  FOR X=1 TO 200:NEXT
  340  *FX15
  350  AD=GET
  360  IF AD<49 THEN VDU 7:GOTO 330
  370  IF AD>50 THEN VDU 7:GOTO 330
  380  IF AD=49 THEN PROCadd ELSE PROCdelete
  390  GOTO 320
  400  REM ----------------------------
  430  REM   Display updated pointers
  440  REM   or updated pointers and
  450  REM   new queue. Also update
  460  REM   head and tail of queue.
  470  DEFPROCqueue
  480  queue$=""
  490  C=0
  500  REPEAT
  510    C=C+1
  520    queue$=queue$+ML$(C)+"  "
  530  UNTIL C=8
  540  PRINT TAB(2,10);CHR$(141);CHR$(134);queue$
  550  PRINT TAB(2);CHR$(141);CHR$(134);queue$
  560  PRINT TAB(P1*4-2);CHR$(141);CHR$(131);A$
  570  PRINT TAB(P1*4-2);CHR$(141);CHR$(131);A$
  580  PRINT TAB(P2*4-1);CHR$(141);CHR$(133);A$
  590  PRINT TAB(P2*4-1);CHR$(141);CHR$(133);A$
  600  PRINT TAB(10,7);P1
  610  PRINT TAB(27,7);P2
  620  REM ............................
  630  ENDPROC
  640  REM ----------------------------
  650  REM  User enters the hexadecimal
  660  REM  number of his/her choice
  670  REM  that will be displayed.
  680  DEFPROCadd
  690  IF I=0 OR I=1 OR P1<>P2 THEN GOTO 730
  700  PRINT TAB(8,16);CHR$(141);CHR$(136);CHR$(133);"OVERFL
OW!!!"
  710  PRINT TAB(8);CHR$(141);CHR$(136);CHR$(133);"OVERFLOW!
!!"
```

```
  720  GOTO 1240
  730  I=2
  740  PRINT TAB(7,20);CHR$(133);"Enter a 2 digit number"
  750  Y=0
  760  REPEAT
  770    Y=Y+1
  780    *FX15
  790    K$(Y)=GET$
  800    IF Y=1 THEN 820
  810  IF K$(Y)=CHR$(13) THEN UNTIL Y>0:GOTO 910
  820  IF K$(Y)>="0" AND K$(Y)<="9" THEN GOTO 870
  830  IF K$(Y)>="A" AND K$(Y)<="F" THEN GOTO 870
  840  VDU 7
  850  FOR X=1 TO 200:NEXT
  860  GOTO 780
  870  PRINT TAB(15+Y);K$(Y);
  880  UNTIL Y=2
  890  ML$(P1)=K$(1)+K$(2)
  900  GOTO 920
  910  ML$(P1)="0"+K$(1)
  920  PRINT TAB(P1*4-1,12);CHR$(141);" "
  930  PRINT TAB(P1*4-1);CHR$(141);" "
  940  P1=P1+1
  950  IF P1=9 THEN P1=1
  960   PRINT TAB(8,20);"                      "
  970  PRINT TAB(16,21);"  "
  980  SOUND 2,-12,160,4
  990  ENDPROC
 1000  REM ----------------------------
 1010  REM   The user wants to delete
 1020  REM   from the queue.  Update
 1030  REM   the value of the pointer.
 1040  DEFPROCdelete
 1050  IF I=2 OR P1<>P2 THEN GOTO 1090
 1060  PRINT TAB(8,16);CHR$(141);CHR$(136);CHR$(133);"UNDERF
LOW!!!"
 1070  PRINT TAB(8);CHR$(141);CHR$(136);CHR$(133);"UNDERFLOW
!!!"
 1080  GOTO 1240
 1090  I=1
 1100  PRINT TAB(16,5);CHR$(141);CHR$(129);ML$(P2)
 1110  PRINT TAB(16);CHR$(141);CHR$(129);ML$(P2)
 1120  PRINT TAB(P2*4,14);CHR$(141);" "
 1130  PRINT TAB(P2*4);CHR$(141);" "
 1140  P2=P2+1
 1150  IF P2=9 THEN P2=1
 1160  SOUND 2,-12,190,4
 1170  REM ..........................
 1180  ENDPROC
 1190  REM ----------------------------
 1200  REM   After the overflow or
 1210  REM   underflow message appears
 1220  REM   on the screen, delete
 1230  REM   it after approx. 8 seconds.
 1240  SOUND 3,3,60,30
 1250  FOR X=1 TO 8000:NEXT
 1260  PRINT TAB(11,16);"            "
 1270  PRINT TAB(11);"           "
 1280  REM ..........................
 1290  ENDPROC
 1300  REM ----------------------------
 1310  REM   The user has pressed the
 1320  REM   escape key. End simulation.
 1330  CLS
 1340  SOUND 1,1,120,75
 1350  PRINT TAB(10,10);CHR$(141);CHR$(136);"End of simulati
on."
 1360  PRINT TAB(10);CHR$(141);CHR$(136);"End of simulation.
"
```

```
1370  FOR X=1 TO 12500:NEXT
1380  *FX4
1390  *FX11,25
1400  MODE 7
1410  REM ..............................
1420  END
```