Replacing the subroutine with a procedure in BBC BASIC gives the following code:

```
10 PRINT "FIRST SECTION"
20 PROCdelay
30 PRINT "SECOND SECTION"
40 PROCdelay
50 PRINT "THIRD SECTION"
60 PROCdelay
70 PRINT "FOURTH SECTION"
80 END
100 REM ** DEFINE PROCEDURE **
110 DEF PROCdelay
120 FOR I=1TO100: NEXT I
130 ENDPROC
```

There are a number of similarities between the subroutine construction and the procedure construction; for instance, both come after the END statment but can be called repeatedly from within the main program. The principal advantage of the procedure is that it is called by name rather than by line number. The DEFinition of the PROCedure could start anywhere after the END statement.

If we wanted our example program to be able to wait for different lengths of time before each section, then a more important advantage of procedures can be utilised: the ability to pass values into a procedure definition. Let us say that we wished the pause between the first and second sections to be 100 loops, the pause between the second and third sections to be 200 loops and the pause between the third and fourth sections to be 175 loops. In standard BASIC the value of I would need to be assigned each time before calling the subroutine. Using procedures, the value can be passed by means of a bracket at the end of the calling statement:

```
10 PRINT "FIRST SECTION"
20 PROCdelay(100)
30 PRINT "SECOND SECTION"
40 PROCdelay(200)
50 PRINT "THIRD SECTION"
60 PROCdelay(175)
70 PRINT "FOURTH SECTION"
80 END
100 REM ** DEFINE PROCEDURE **
110 DEF PROCdelay(N)
120 FOR I=1TON: NEXT I
130 ENDPROC
```

More than one value can be passed for use within a procedure if the items are separated by commas. Variable names can also be used in this way to pass the value of the variable at the time when the procedure is called.

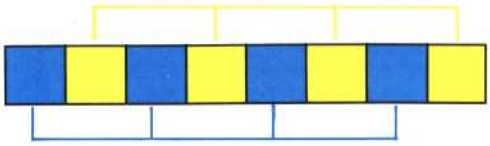
The game we shall be constructing is for one player and uses the keyboard cursor control keys to move a mine detector around a minefield. Points are scored for each mine successfully dealt with. There are, however, several things to slow down your progress around the minefield. The chief consideration is your assistant who mirrors your every move. As you go round dealing with the mines you must make sure that he doesn't tread on one. You are also being sniped at and there is a two minute time limit to the game. In the final version, you will have four willing assistants per game and a choice of skill factors from 0, the easiest, to 9, the most difficult.

Because BBC BASIC uses procedures, flow charts aren't of much use. Instead, a *structure diagram* can be used to illustrate what procedures are required and how they fit together to make the final program. REPEAT . . . UNTIL loops are shown as 'sausage' shapes in our diagram. Decision boxes are like the more usual diamond-shaped box but with the top and bottom cut off to save space. It should be stressed that this diagram was not drawn out in its entirety before programming started, but evolved out of a series of refinements.

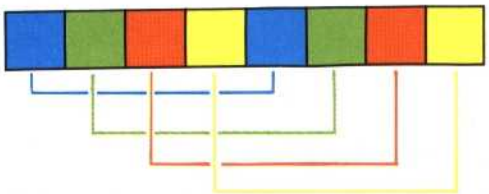
SET-UP PROCEDURES

Brfore we can start defining routines to place objects on the screen, we must decide on the screen display mode we are going to use. There are three main factors to take into account: resolution, colour and memory. In general terms, the more information the screen has to hold, the more memory it will require. So, higher resolutions and more colours mean more memory. If your program is short this may not be important to you, but the program we are designing is fairly lengthy. We do require a few different colours to distinguish the mines and the detector/assistant and to make the game visually appealing. On the Model B we are offered two medium-resolution modes. Mode 2 gives us 16 colours to play with, whereas mode 5 offers only four. By looking at how the BBC interprets bit patterns in each mode we can see why mode 5 uses substantially less memory than mode 2.

Unlike some micros, the BBC holds colour and pixel on/off information within one byte for each small area of the screen. In mode 2, four bits are required to represent the 16 different possible colours. Thus, one byte can hold information about the colour of two pixels only. The bits within the byte are arranged as follows:

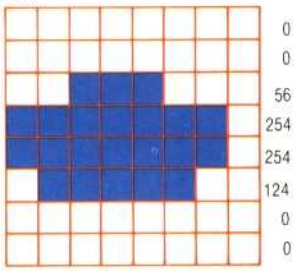


As mode 5 restricts itself to four colours, only two bits are needed to hold the colour information. Thus, one byte can represent four pixels as shown:

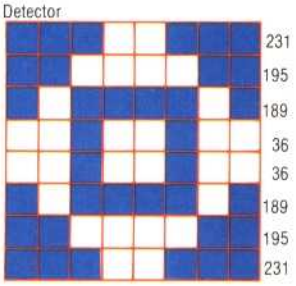


Both modes have 160 by 256 pixel resolution, hence the number of bytes required for mode 2 screen memory is (160×256)/2=20 Kbytes,

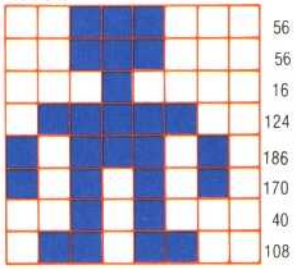
Mine



Detector



Assistant



Picture Points
The user-defined characters representing the mine, the detector and the assistant are described. The pixel information is divided into eight bytes, with each byte being the binary 'picture' of one row of the shape