Data Structure  
  
The chosen data structure for the Bluebrook Airport System is a linear structure; a queue. The purpose of a queue is to organise data is a First In, First Out (FIFO) structure. This is implemented in the flight time board, which will display the current system time, the flight ID, flight locations, terminal number, departure time and current system time. There is a fixed amount of flights displayed on screen at one time, previously sorted in the system by departure time. The system/current time exceeds the time on the lowest sorted time, the time dequeues from the timeboard and the row is hidden from the time board, as this happens a later time, previously not displayed will be enqueued on the time board. The process loops throughout the day until all flights for that day have departed the airport.

Enqueue

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Flight ID** | **From** | **To** | **Terminal** | **Departure Time** |
| LDNPRG001 | London | Prague | 2 | 14:29 |
| LDNPRS01 | London | Paris | 1 | 13:59 |
| LDNROM01 | London | Rome | 2 | 13:29 |
| LDNBLN01 | London | Berlin | 1 | 12:59 |
| LDNMAD01 | London | Madrid | 2 | 11:29 |
| LDNADM01 | London | Amsterdam | 1 | 10:59 |
| Current Time | | | | 11:17 |

Dequeue

Advantages   
**Speed** – queue are quick to implement, use and maintain as there is only two main stages to organising the data (enqueuer/dequeuer).  
  
**Memory** - as a queue data structure is linear, it will take up more memory than a non-linear structure.  
  
Disadvantages   
**Sorting** – Before the data can be put in the queue, it must be sorted first, in the case of the Bluebrook airport system, data must be sorted by time closest to system time.  
   
Decision  
The decision to use a queue data structure was an obvious choice, as is used in real time board systems as it is the most accurate, easy to understand method of displaying sorted time data closest to a certain time (current system time).

Dequeue