

Exercise 8.1

1. File is a structured collection of data ie a collection of records related to an object of interest eg file of temperatures or file of students
2. A record is a structured collection of data relating to an individual object eg a student record
3. A field denotes a data part of a record eg student date of birth within the student record.
4. Fields normally have a data type in the same way that variables have a data type. This is required to enable a program to know how to treat the data eg does the field contain a real or an integer. This is needed when the data field of the record is read into RAM.
5.

Field Name	Data Type
Student ID	integer
firstname	String
surname	String
address	String (could break this up in Street Number, Street, Suburb, Zip etc)
phone	String
Contact	String
email	String
6. Question meant to ask determine file size
$$\text{record length} = 4 + 1 + 8 + 10 = 23\text{bytes}$$
$$\text{file size} = 28 * 1000000 = 28,000,000 = 28\text{MB}$$
7. Database is a logical collection of related files eg School Database

Exercise 8.2

1. In the terms of the IB a sequential file is ordered eg a file of surnames would be in alphabetic order, whereas a serial file has no logical order ie a file of surnames would not be sorted.
2. Access is the mechanism available to read or write the record. Organisation is the method used to arrange the records eg in linear ordered fashion one after the other with no gaps between records, or randomly scattering the records either in any order in the file or scattered over the disk in random order.
3. Most languages allow the use of append, otherwise you need to read through the file until the end is reached and then write.

Open file
Append record
Close file
4. Open file
While not end of file
 Read record
 if record.age > fixed value display record
EndWhile
close file
5. Open file for reading
while not end of file
 read records into array structure
edit array position
close file
Open file for writing
set array index = 1
while not end of array

```

        write array[index] to disk file
        index = index +1
    endwhile
close file

```

6. You need to read the data into a list structure and sort this structure and then write the file out to disk.
7. The basic algorithm is that which is used by operating utility software that makes a copy of a file.

```

Open fileA for reading
Open fileB for writing
While not end of fileA
    read record
    write record to fileB
Close fileA, fileB

```

Exercise 8.3

NOTE: there is no Exercise 8.3

Exercise 8.4

1. A partially index file means that not all records have an index entry in the index. To access a record the index is searched and then the location of the first record in the disk block is accessed directly and the records read sequentially until the desired record is found.
2. A fully indexed file means that each record in the file has a corresponding entry in the index. To read a record the index is searched and then the record accessed directly using the index entry, which is the relative record position.
3. Advantage of partly indexed access is that the index file is smaller and thus takes less space in RAM and the index file itself takes less space on disk. The disadvantage is that disk blocks are needed to enable groups of records to be stored this can waste space or require the file to be re-built to re-group the record blocks to correspond to the partial index entry.

Advantage of the fully indexed approach is that each record is indexed and hence there is no need for record blocks. Once located in the index the record is directly accessed and no sequential searching of the disk block is required.

Disadvantage is that the index itself can be large and take up disk space and RAM when it is in use.

Both have the overhead of maintaining the index ie adding and deleting entries which requires additional CPU resources.

4. The index for a fully indexed file would require entries for each record as shown below.

Index	file
Andrews (1)	1. Andrews
Brown (2)	2. Brown

Zork (3) 3. Zrok

A partly indexed approach would require alphabetic disk blocks ie one for each starting letter of a surname. In this case we allow three spots for each letter. If a fourth name was added starting with A the file would need to be re-built along with the index.

Index	file
Andrews (1)	1. Andrew 2. 3.
Brown (4)	4. Brown 5. 6
Zork(100)	(gaps for the other letters here). 100 Zork

110. End of File

Exercise 8.5

1. Direct access file organisation implies that the records in the file can be accessed directly without reference to the other records in the file, which is different to sequential access and provides much faster access. The records can be scattered over the disk space in any order, hence the term random file, or random access file.
2. Magnetic tape can only be accessed sequentially because the tape is wound past the read heads. It is not possible to directly access records in the tape without using this mechanism.
3. Because the disk's surface or platter are divided into tracks and these are further divided into sectors. Thus it is possible to determine directly the location of a record by calculating its track and sector.
4. The PURE moveto command indicates to move the reader to the byte position specified in the byte sequence that constitutes the file.
5. for you to do.
6. Unlike sequential file processing, direct access allows bytes to be written directly onto the disk. Again this is possible because of the track and sector mechanism.
7. The record position is used because we use fixed length records. Thus if a record is fixed to be 100 bytes in length we can say $\text{moveto}(\text{recordNumber} * \text{recordLength})$. Here record 1 is thought of as in position 0, otherwise the formula is $\text{moveto}([\text{recordPosition} - 1] * \text{record length})$.
8. for you to do.
9. We could keep an index of the key in sorted order ie key could be surname for instance. OR, we could read through the file repeatedly to locate the records - you would not do this!
10. An ATM uses the customers bank account number to directly access the main file. Only a directly access system would provide the necessary speed of access.

11. There is probably very little need to gain direct access to a temperature, thus the most sensible structure would be a sequentially organised and accessed file.

Exercise 8.6

1. Similar to an earlier exercise. The difference is that the hash total is used to generate a record position. You could handle clashes by setting aside an overflow part to be located after the hash space size.

File Hash Space (records 1 to n)

File overflow part (record n+m)

2. It can waste space and thus take up a lot of disk space.
3. No hashing does not allow records to be stored in order. A hashing process simply generates a set of values with the limits of the hash space. The keys used to generate the sequence will be in random order.
4. Hashing requires no index, thus saves file space ie no index file. There is no need to read the index to retrieve records hence the file processing will be faster.
5. The advantage an index has is that we can access the records in order and no overhead is lost in handling clashes.
6. for you to do.

Exercise 8.7

1. The algorithm requires reading through the file in the order the records are stored. This is essentially how a basis file query is conducted that requires inspecting each record. It is the same process someone would use if they were asked to read through a filing cabinet or worker records to create a report that listed all those workers whose birthday was in the following week.

The basic algorithm is as follows:

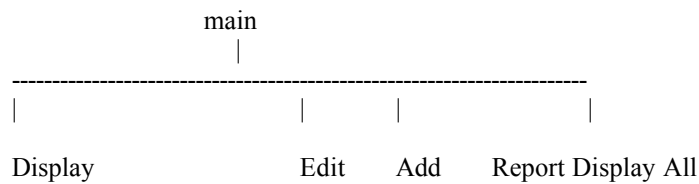
```
procedure DISPLAYREPORT2(val daysCheck integer)  
  declare J integer  
  for J = 1 upto filesize (DATAFILE) do  
    if (RECLIST[J].DAYSOUTSTANDING > daysCheck)  
      output(RECLIST[J].CUSTID)  
    enddo  
endprocedure DISPLAYREPORT2
```

2. For your to do!
3. for you to do.

4. pass as an argument
5. for you to do.

Exercise 8.8

1. The basic structure chart would need a main module and then below modules to perform the required functions. The locate function would be required by all modules.



Exercise 8.9

1. For you to do.
2. For you to do.
3. For you to do.

Exercise 8.10

1. The basic idea is to store the key+record position as a pair of values in. A composite record structure would be needed and an array of that type used to store each composite record.

The file would be a fully indexed.

Index Array []

key 1, record address n
 key 2, record address m
 etc.

Exercise 8.11

1. to 5. for you to do.