



A3.8 Fields, records and file organisation

Data structures · A level · AQA 7517 4.2.1.3, Eduqas A500QS 2.4 · about 25 min

Name _____

Class _____

Date _____

What this lesson is about

Text and binary files, fixed-length records, and serial, sequential, indexed sequential and direct access files.

- Fields, records and files
- Text files and binary files
- Serial files
- Sequential files
- Indexed sequential files
- Direct access files

Questions

6 marks in all

1. What is the difference between a serial file and a sequential file? [1 mark]

- ☐ A A sequential file keeps its records in key order; a serial file keeps them in the order they were added
- ☐ B A serial file keeps its records in key order; a sequential file keeps them in the order they were added
- ☐ C A serial file has an index; a sequential file does not
- ☐ D A sequential file uses a hashing function to place records

2. Which file organisations let a program go straight to, or near, a record without reading from the start of the file? [1 mark]

Tick every answer that is true.

- ☐ A Indexed sequential
- ☐ B Direct access
- ☐ C Serial
- ☐ D Sequential

3. A file holds fixed-length records of 32 bytes each, starting at byte 0. At which byte does record number 5 start, counting records from 0? [1 mark]

-
4. An airline's booking system looks up and changes one seat record at a time as customers call. Which file organisation suits it best? [1 mark]

- ☐ A Direct access
- ☐ B Serial
- ☐ C Sequential
- ☐ D A text file read line by line

5. Put the steps for updating a sequential master file from a transaction file in order.

[1 mark]

Number the lines 1 to 5 to put them in the right order.

- ___ When the keys match, apply the transaction and write the updated record
- ___ Sort the transaction file into the same key order as the master file
- ___ When one file runs out, copy the rest of the other file
- ___ Write whichever current record has the smaller key to the new master file, and read the next from that file
- ___ Read the first record from each file

6. What is an advantage of a binary file over a text file for storing records?

[1 mark]

- ☐ A Values are stored in their internal form, so records are smaller and need no conversion from characters
- ☐ B A person can read and edit it in any text editor
- ☐ C It does not need the program to know the record layout
- ☐ D It can only hold strings

The task: update the master file

`master.txt` is a sequential master file of robots, in ascending order of `id`, with the header `id,name,best_cm`. `results.txt` is a transaction file with the header `id,name,cm`, also in ascending `id` order, holding at most one new run per robot. Both files are shown below. Merge them in one pass, without sorting, into a new file `new_master.txt` : - The first line is the header `id,name,best_cm`. - Then one line per robot, in ascending `id` order, in the form `id,name,best_cm`. - A robot in `master.txt` with no result is copied unchanged. - A robot in both files keeps the **larger** distance, written exactly as it appears in whichever file it came from (for example `41.0`). - A robot only in `results.txt` is added with its `cm` as its `best_cm`. Finally print `<improved>` records `improved`, `<added>` `added`, where `improved` counts robots whose best distance went up and `added` counts new robots. The robot does not move.

```
# the two lines every program starts with: the commands, then the robot
from bugbot import *
connect()

with open("master.txt") as f:
    master = f.read().splitlines()
with open("results.txt") as f:
    results = f.read().splitlines()
```

Plan your program here, then type it in and press Run.



Do it on the robot

www.bugbotlab.com/learn/a3-8-file-organisation/

The simulator checks it and tells you when it passes. Nothing to install, no account.

Challenges

1. Change `results.txt` so a transaction has an `id` that is lower than every `id` in the master file. Does your merge still work?
2. Why must the transaction file be sorted before this kind of update? What would go wrong with a serial transaction file?
3. `find` in the indexed sequential example reads a whole block for a key that is not there. How could it stop sooner?