Computer Science 335 - File Structures
Programming Assignment 3
DUE: Sunday, Oct 8, 2017 (11:55pm)
CUTOFF: Tuesday, Oct 10, 2017 (11:55pm)
50 Pts

DAFL - Direct Access, Fixed Length

The purpose of this assignment is to create a class called **dataFile** that a program can use to obtain support for a **direct access, fixed length record file**. A **dataFile** object will encapsulate the basic file operations for an application which wishes to read and write persistent data. The file structure assumed by a **dataFile** object is a header record followed by fixed length records. The header record consists of two fields: 1) the size of each fixed length record and 2) the number of records currently in the file. To support the direct access feature we are wanting to implement, the read and write operations will use the (R)elative (R)ecord (N)umber - RRN to help calculate the actual record address of a record in the file. RRN's will begin with record 0.

You are provided two applications to test the dataFile class for correctness and these are found on the class webpage.

dataFile methods to implement

---

createFile - will create a file specified by the user

Prototype: void createFile (char *fn, int n)

- fn = string which contains the physical file name
- n = size of fixed length records to store (in bytes)

Purpose: Create an input/output file and an I/O connection to the file, create header for file, and set the file status indicator to either *fsCreateFail* or *fsSuccess.* This method should not create the file if that file already exists.

openFile - will open a file specified by the user

Prototype: void openFile (char *fn)

Purpose: Connect an I/O stream to a physical file, read in file header for future processing of file, and set the file status indicator to either *fsOpenFail* or *fsSuccess.* This method should fail if the file does not exist.

closeFile - close a file connection

Prototype: void closeFile ()

Purpose: Write the file header back to file , close the file and set the file status indicator to *fsCloseFail* or *fsSuccess*

putRecord - put a record into a file via direct access

Prototype: void putRecord (int k, const void *r)

- k = Relative Record Number (RRN) where record is to be placed in the file, must be 0 <= RRN <= Count

- r = pointer to the memory record to write to file

Purpose: Write the record r into the file at rrn k using direct access. Set the file status indicator to either *fsPutFail* or *fsSuccess*

getRecord - retrieve a record from dataFile directly

Prototype: void getRecord (int k, const void *r)
- k = Relative Record Number(RRN) where the record to be retrieved is located in the file
- r = pointer to a record location in memory where the record is to be read into

Purpose: Read rrn k from file into r. Ensure that the rrn is valid (not negative nor greater than the number of records in the file). Set the file status indicator to either *fsGetFail* or *fsSuccess.*

recordCount - return the number of records on file

Prototype: int recordCount ()

Purpose: Return the count of records in the file. File status is unchanged by this function.

updateRecordCount - inc or dec number of records on file

Prototype: void UpdateRecordCount (int n)
- n = value to increment or decrement record count by

Purpose: Increment or decrement the header record count field by n. File status is unchanged by this function.

FileStatus - return the current status of the file

Prototype: int FileStatus ()

Purpose: Determine the status of the last file operation and return
fsSuccess
fsCreateFail
fsOpenFail
fsCloseFail
fsPutFail
fsGetFail

Please submit **dafl.h and dafl.cpp** using Easel. Even though dafl.h is provided, some may wish to “tweak” it.

dataFile definition along with other information

```
#include <fstream>

using namespace std;

//File status indicators
#define fsCreateFail 0
#define fsSuccess 1
#define fsGetFail 2
#define fsCloseFail 3
#define fsOpenFail 4
#define fsPutFail 5

//Class definition for a dataFile type, this class will give a dataFile object
// the capability to perform basic file I/O operations for a fixed length,
//direct access record file structure.
class dataFile
{
	public:
		void createFile (char *,unsigned int);
		void openFile (char*);
		void closeFile ();
		void putRecord (long,const void*);
		void getRecord (long,const void*);
		int recordCount ();
		void updateRecordCount(int);
		int fileStatus();

	private:
		fstream finOut;
		int recSize;
		int recCount;
		int fs;
};
```