Emma Zajonc

Lab 3 LRU Buffer Pool

CSCN 215\_003

March 28, 2024

Approach Document

In this lab, I was asked to implement two abstract data types: a buffer block data type and a buffer pool data type that operates according to the least-recently used rule. These two data types will be used to process a file in blocks.

On the first day, I will implement the BufferBlock class, starting with the constructor. The constructor must initialize the char array that holds the blocks data. Next, I will complete all of the getter and setter functions. I expect that these will be fairly straightforward. Finally, I will code the function getData(). This function allows the buffer block to read a certain portion of the data in the block of size sz starting from a position pos. The retrieved data is then passed into a char array. This function will use a for loop to process each char character. This will complete BufferBlock and day one of my project.

On day two, I plan to begin work on the class LRUBufferPool, which features the most complex algorithms in the lab. First, I will add a new private variable to the LRUBufferPool class called flName which will hold the name of the file. As stated in the lab, the class will also contain an array of five BufferBlocks.

Next, I will implement the constructor. The constructor will initialize the buffer pool. First, I must open the file in binary. Then, I will use a for loop to iterate through each element in the array of BufferBlocks and read the appropriate data into each block. I will also set the block ID for each initial block. Finally, I will save the name of the file so that I can open it later.

Next, I will work on the functions printBufferBlockOrder() and getLRUBlockID(). These are simple functions I would like to take care of before tackling getBytes().

Finally, I will complete getBytes(). This function takes three parameters: the size of the desired piece of data, the position at which the data is located, and a char array to receive the data. Before proceeding, I will need to define two variables: blockNum and blockPos. I will set blockNum equal to the position of the data divided by the size of each block, leaving me with a value that is equivalent to the block’s ID. I will set blockPos equal to the position of the data modulus the size of each block, giving me the relative position within the block at which the data starts.

Next, I will check if the block that contains the data at the desired position is already in the buffer pool. If it is true, I will pass the data in that block to getBytes()’s char array, move the block to the front of the buffer pool, and shift the other blocks back. If the desired block is not in the buffer pool, I will have to initialize a new BufferBlock. I will open the file again and read in the data in the desired block. I will assign this block to the first index in LRUBufferPool’s BufferBlock array and shift the other blocks down as before.

I will leave a third day to spend debugging. With a project this size, I always expect errors, and I find it wise to leave time to address them.