Design of program

For implementing the page replacement algorithms, I have modeled each cache using arrays. For each algorithm, I am maintaining a separate array.

First in the code, I am reading the number of physical pages and page size. Then I am reading and storing the addresses being accessed.

After reading the inputs, I am converting each address to the page number by performing integer division by page size.

After that, for each page being accessed, first I find out if that page is present in the caches for all three algorithms by scanning the caches.

If page is not present in the cache, it means there is page fault. If cache is partially full and page fault happens, then it is first time access. I am handling this case for all algorithms while handling page fault for fifo. I am simply adding this page to end of array of each cache. Page faults for all three algorithms is increased by 1.

For lru, if the page is found, we have to move the page to end of array.

Then for fifo and lru, if cache is full, we just need to pop the first element and push the new page to the cache. Page faults for both these algorithms is increased by 1.

For opt, if cache is full, we have to search over the later pages and find the first next access of each page in cache. From these, whichever has last next access is our target for replacement. If there is no next access for a page in cache, that page is the target. After finding the next access for each page in cache, we set the target and replace the target. Page fault for opt is increased by 1.

After this, I am printing the total number of the page faults for all three algorithms.