**Project 4: Pager**

Duc Vo, Ty Gehrke

Grand Canyon University

CST-315: Operating Systems Lecture and Lab

Professor Citro

March 10th, 2024

# **Team Members**

- Ty Gehrke

- Duc Vo

# **Project Overview**

Project 4 aims to enhance the memory management capabilities of the existing shell by implementing virtual memory. This addition addresses the limitations of the current system, which lacks mechanisms to handle memory requirements and utilizes only physical memory. By incorporating virtual memory, the project seeks to extend the available memory for processes, thereby improving the overall efficiency and performance of the shell.

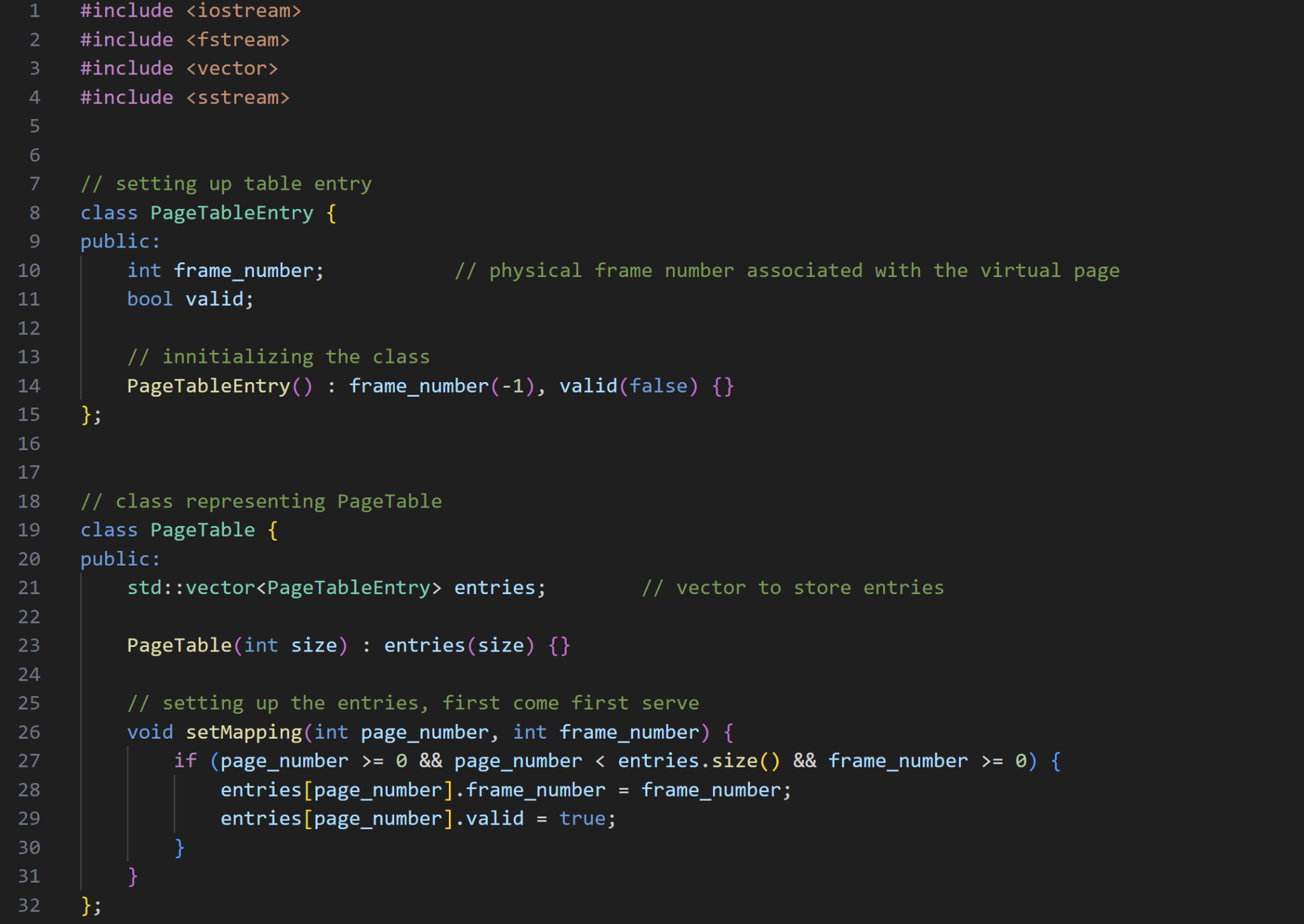
**Project Objectives:**

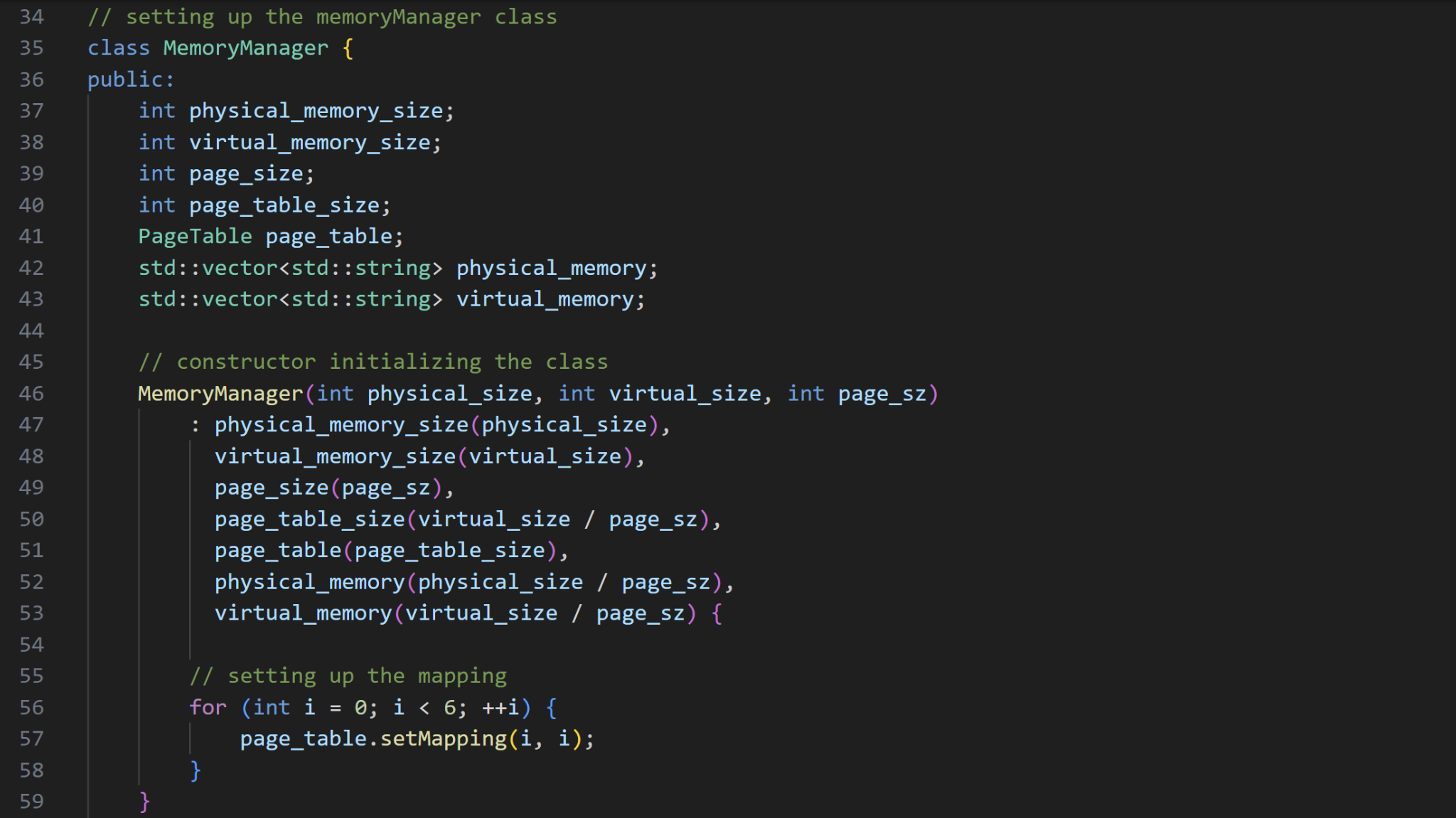
1. **Memory Allocation Enhancement:** Develop mechanisms to efficiently allocate memory for processes, considering their individual memory requirements.
2. **Virtual Address Implementation:** Integrate virtual addressing into the shell to enable processes to use virtual memory addresses, which will be translated to physical addresses by the system.
3. **Page Table Management:** Design and implement data structures and algorithms for managing page tables, ensuring efficient mapping of virtual addresses to physical addresses for each process.
4. **Page Fault Handling:** Implement robust mechanisms to handle page faults, enabling the system to respond effectively when a process accesses a virtual address not currently mapped to physical memory.
5. **Address Translation:** Implement efficient translation of virtual addresses to physical addresses using the page tables, facilitating seamless memory access for processes.

**Algorithm Description**

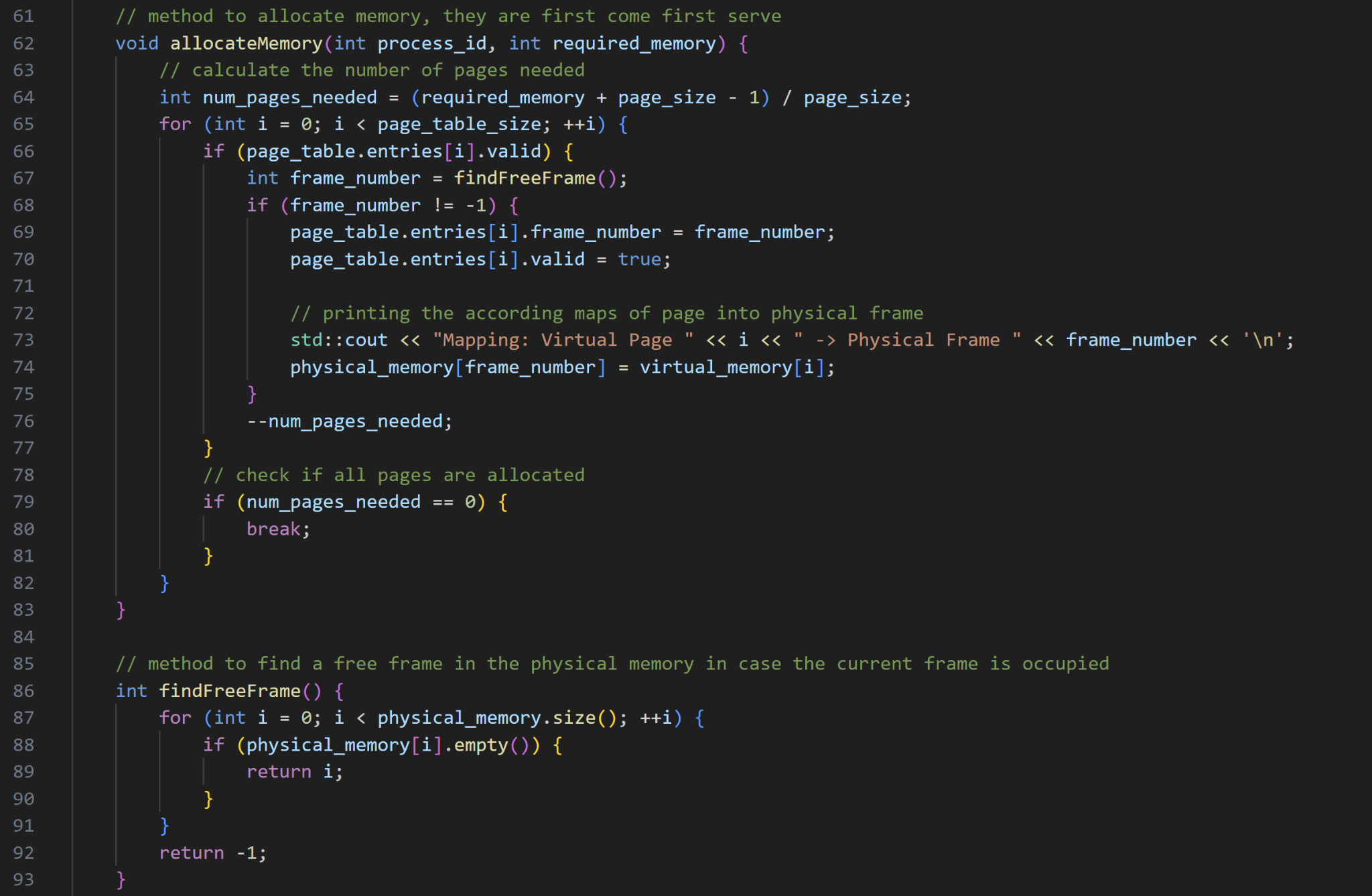
* Initialization:
  + The system initializes the memory manager with the sizes of physical memory, virtual memory, and the page size. It also sets up the page table with initial mappings.
* Memory Allocation:
  + When a process requires memory, the memory manager calculates the number of pages needed based on the process's memory requirements.
  + It iterates through the page table entries to find available pages. If a page is not already mapped to a physical frame, it allocates a free frame in physical memory.
  + It updates the page table entry with the frame number and sets it as valid. Then, it copies the content from the corresponding virtual memory page to the allocated physical frame.
* Page Table Management:
  + The page table manages the mappings between virtual pages and physical frames. Each entry in the page table stores the frame number and a validity flag.
  + The setMapping method allows setting the mapping between a virtual page and a physical frame.
* Page Fault Handling:
  + The system handles page faults implicitly by allocating memory when a process accesses a virtual page that is not mapped to any physical frame.
  + It identifies free frames in physical memory and maps the accessed virtual page to a free frame, copying the content from virtual memory to physical memory.
* Program Loading:
  + Programs are loaded into virtual memory from files. Each word from the file represents data stored in a virtual memory page.
  + The loadProgram method reads data from the file and populates the virtual memory pages accordingly.
* Printing Memory Contents:
  + The system provides methods to print the contents of virtual memory and physical memory for debugging and monitoring purposes.
* Example Usage:
  + In the main function, the system loads data from files into virtual memory and allocates memory for processes.
  + It prints the page table mappings, physical memory contents, and virtual memory contents after each memory allocation.
  + Overall, the algorithm efficiently manages memory allocation, handles page faults, and maintains mappings between virtual addresses and physical addresses, thus simulating a basic Virtual Memory Management System within the shell environment.

**CODE & FLOWCHART**

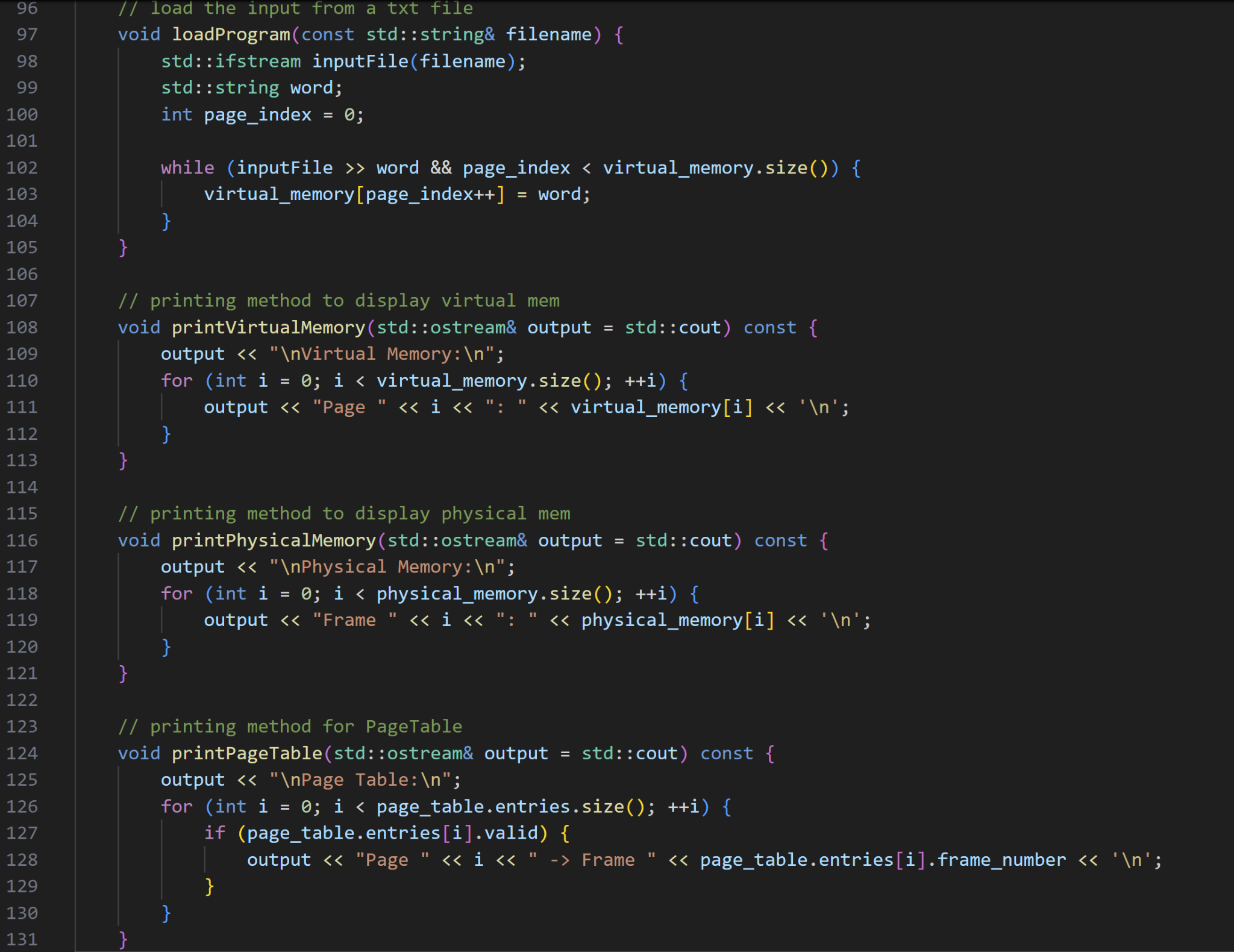
**** First we will be setting up the Page Table Entry using vectors to represent

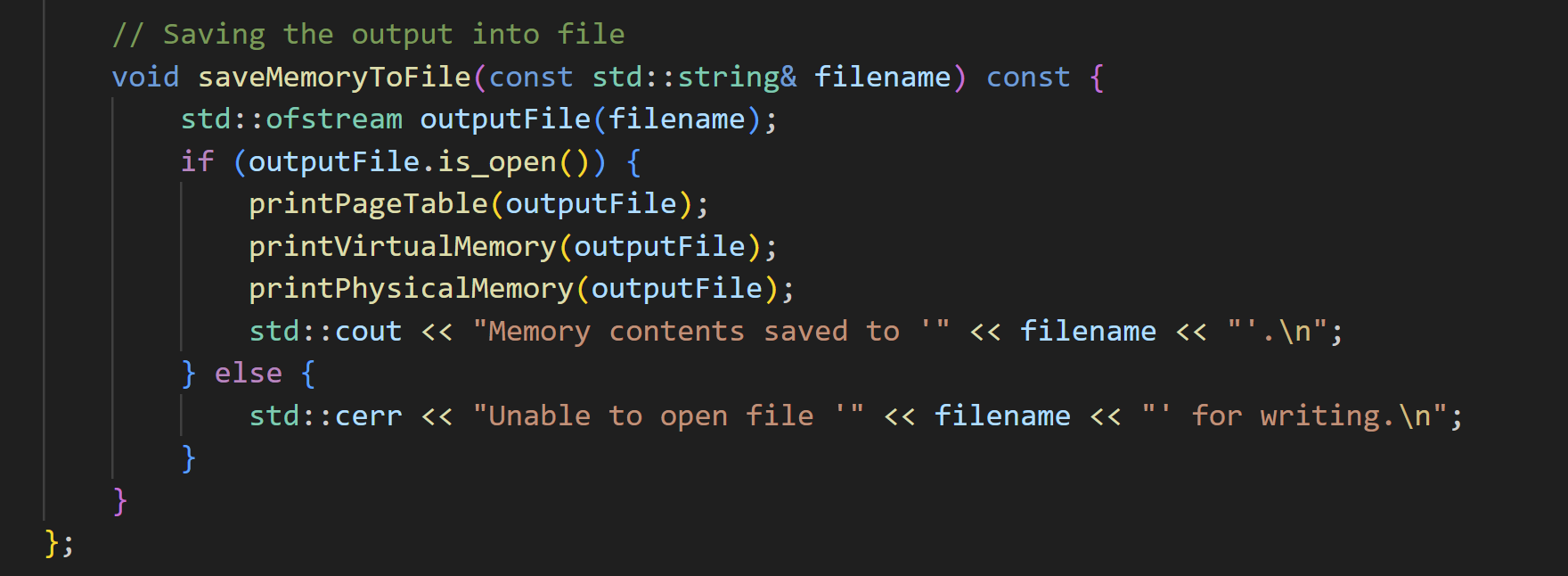


And we will be using the class MemoryManager to simulate how an operating systems manage virtual and physical memory

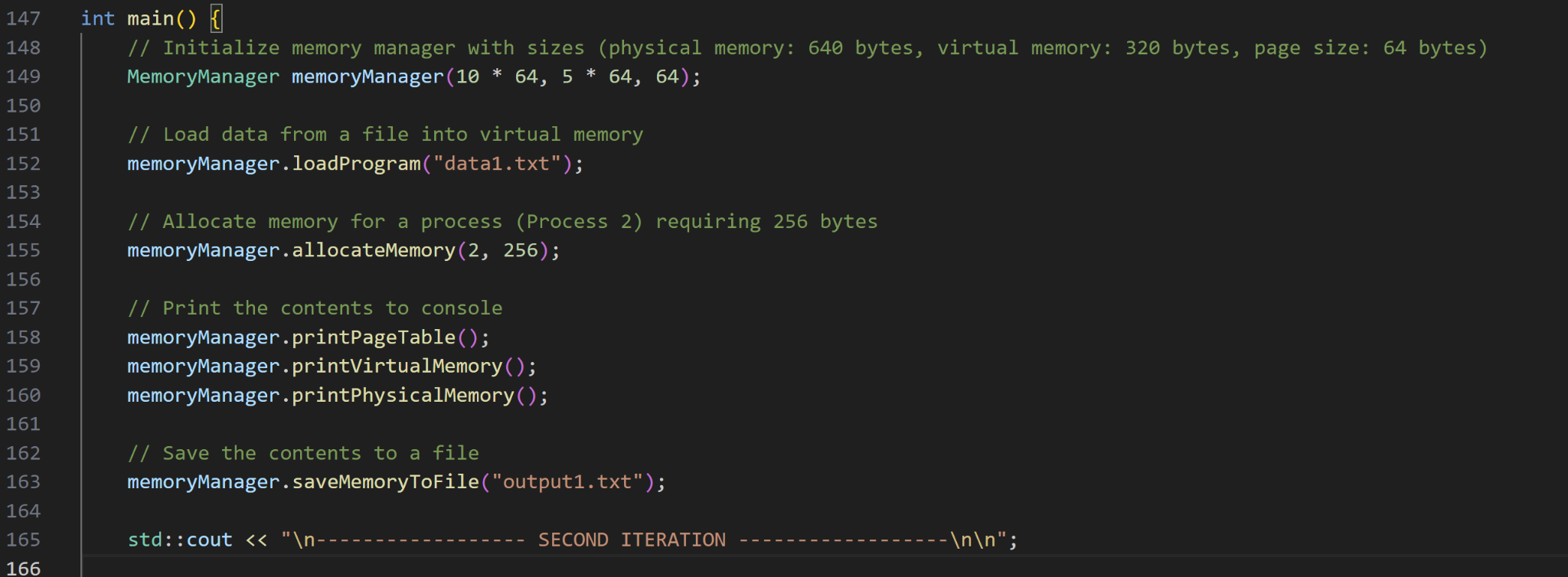


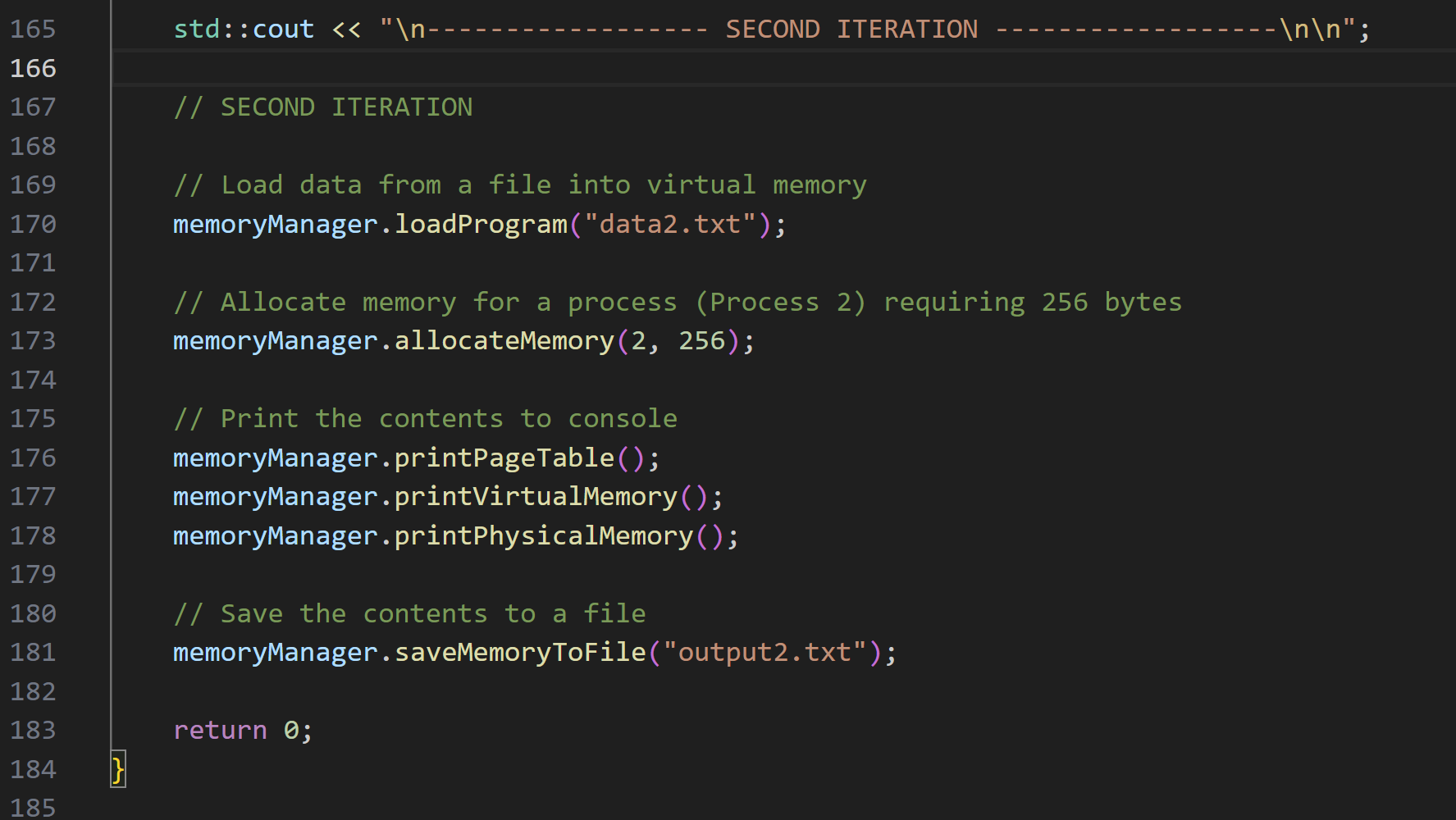
For simplicity, we will just use a queue (first come first out) model for our allocations, the first block of virtual memory will be utilizing the first available block of physical memory



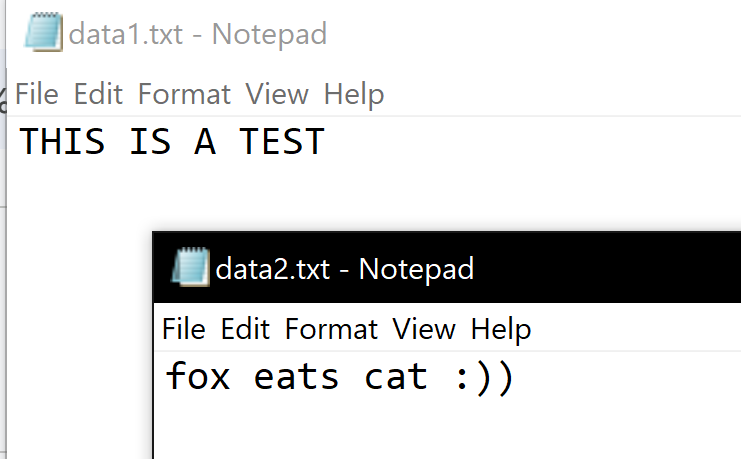
Some helper functions display what is happening on the background

And saving the output to a file

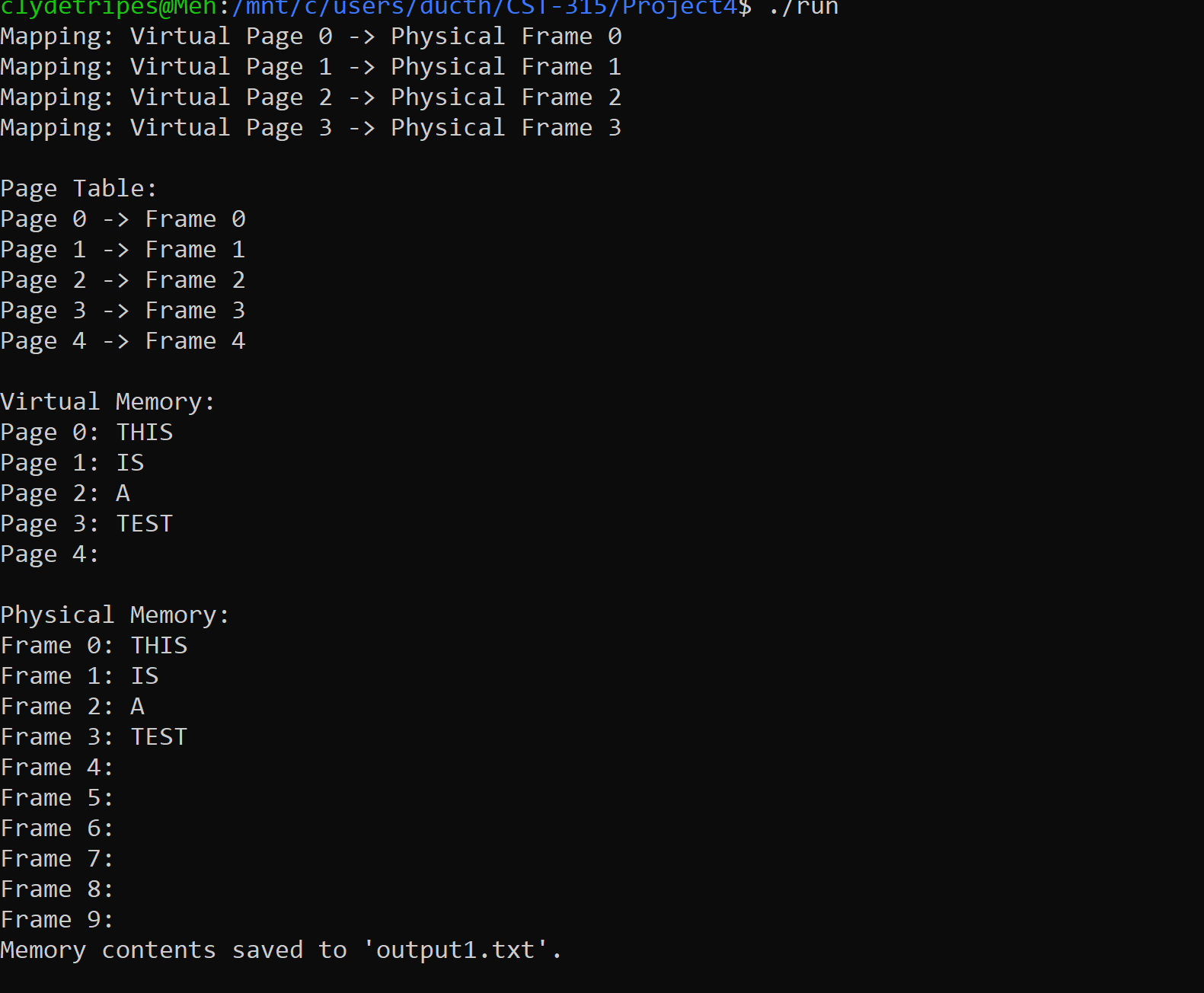




For our simulation, we will be using 2 iterations to simulate the operating systems handling 2 different memory blocks, the first would be in ‘data1.txt’, and the second would be in data2.txt’

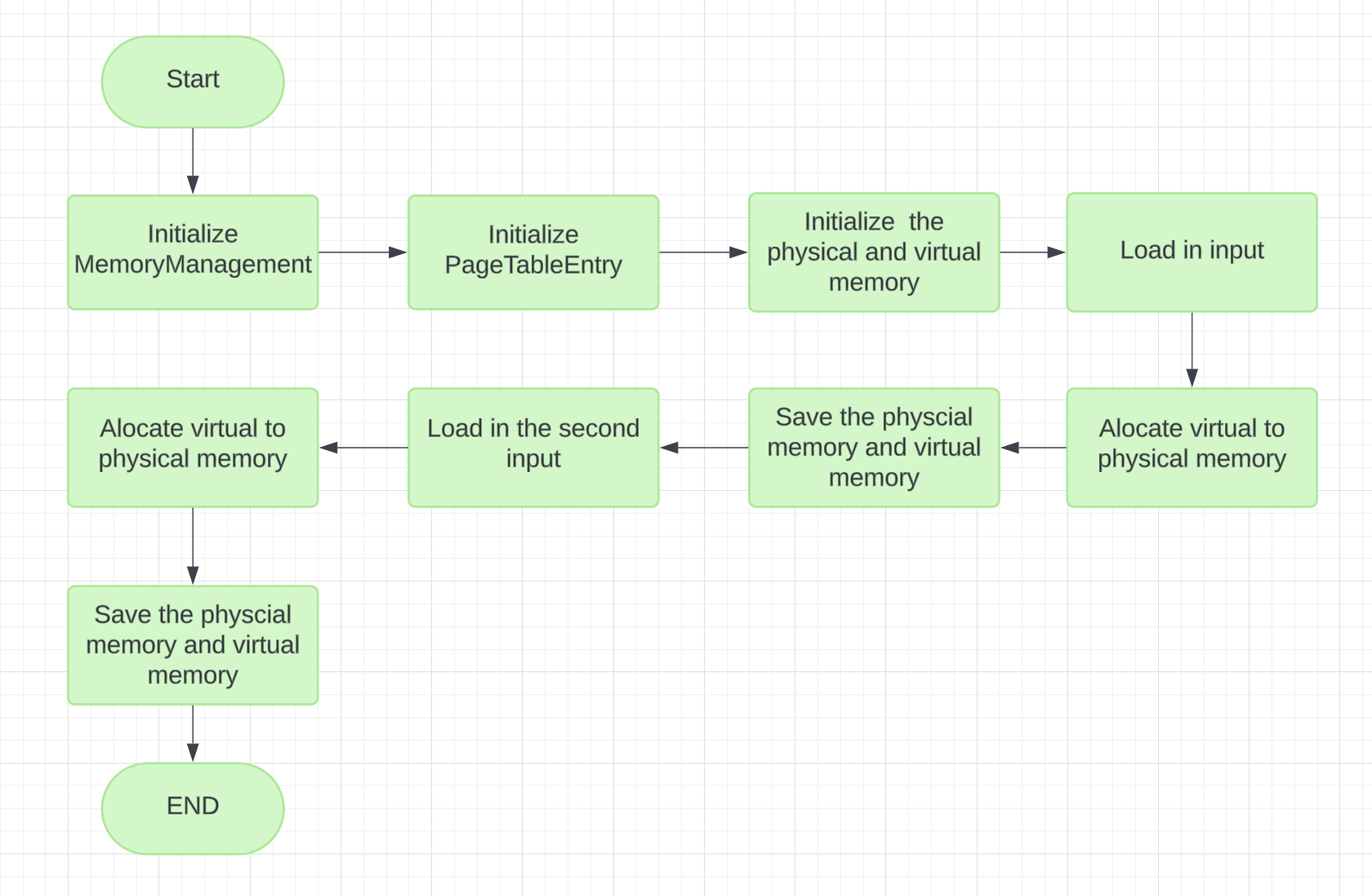


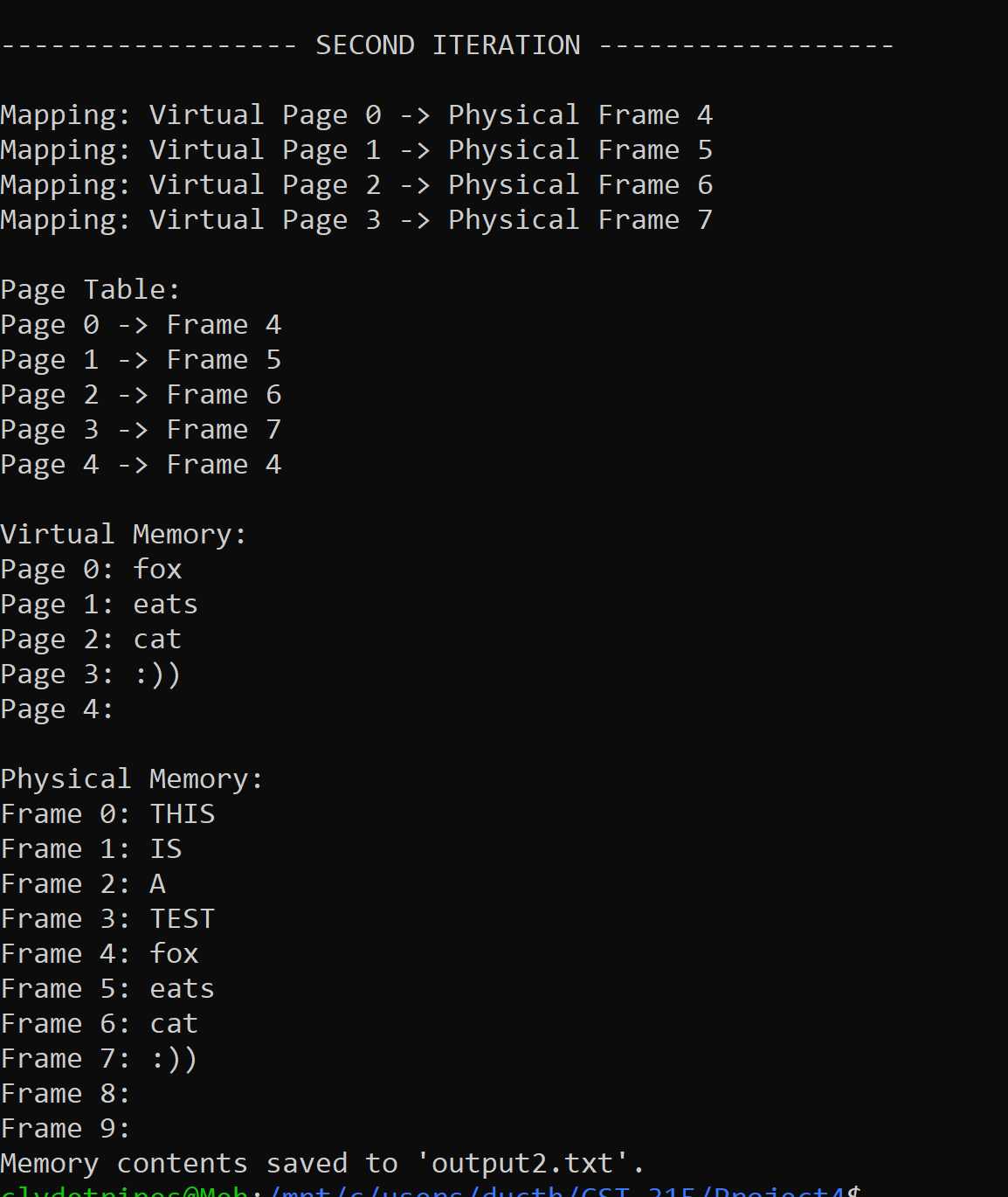
These are the input data



The memoryManagement after the first iteration

**FLOWCHART**

****



**References**

GeeksforGeeks. (2024, January 19). Page Replacement Algorithms in Operating Systems.

https://www.geeksforgeeks.org/page-replacement-algorithms-in-operating-systems/

Stec, A. (2023, May 12). Virtual Memory. Baeldung on Computer Science.

https://www.baeldung.com/cs/virtual-memory