Graphical user interface

Description automatically generated

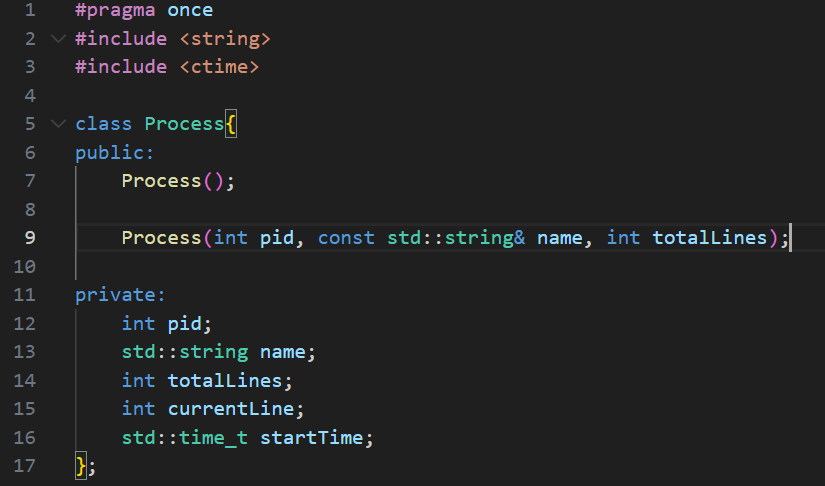
**CSOPESY Seatwork**

**Instructor:** Neil Patrick Del Gallego, Ph.D.

*Use Calibri Font Size 11 for texts.*

|  |  |
| --- | --- |
| **NAME:** | **Brianna B. Salvador** |
| **SECTION:** | **S13** |
| **DATE OF SEATWORK** | **05/30/25** |

1. **Provide a mockup C++ header code snippet that shows how a process can be represented in your OS emulator project. Describe its attributes.**



**Variables**

pid = PID of the running process

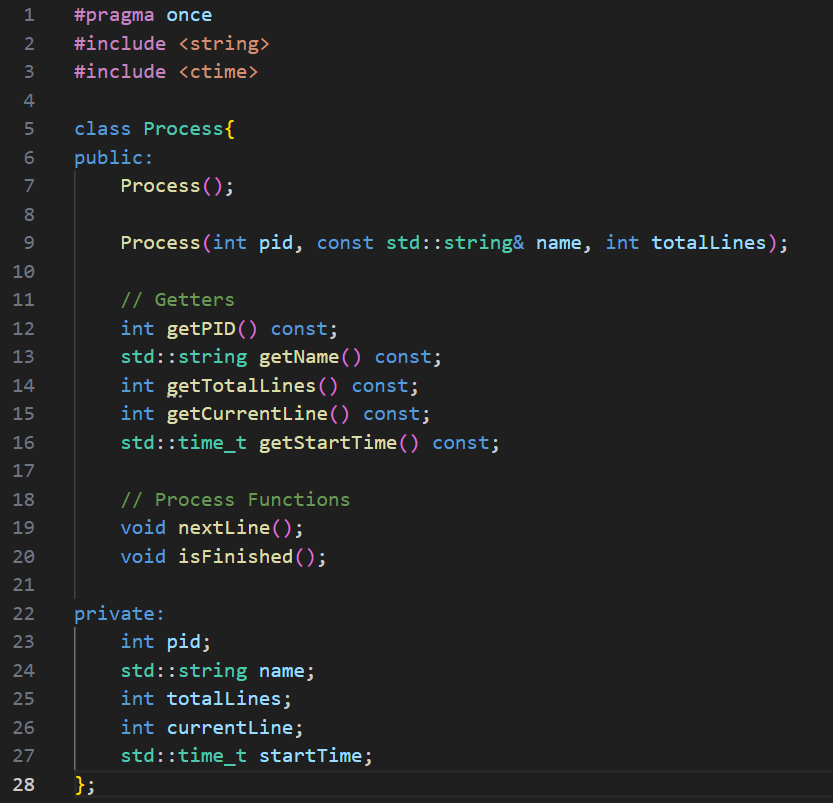
name = Name of the process

totaLines = Total number of instructions / lines

currentLine = The current line number (starts at 0)

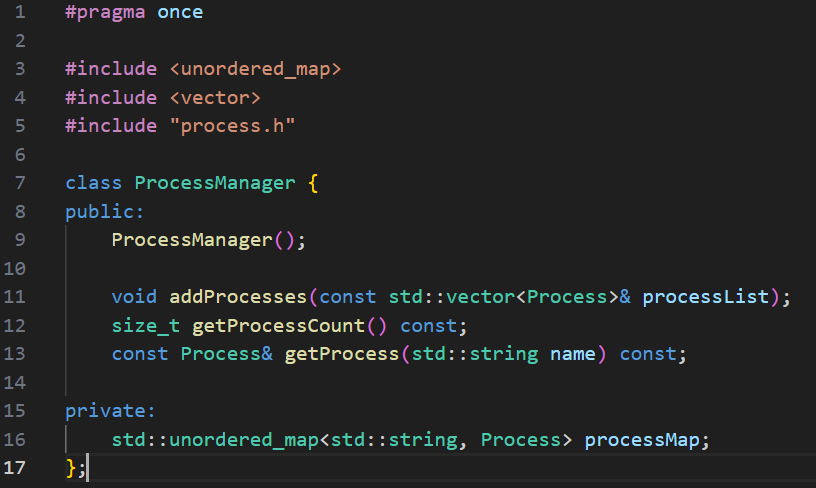
startTime = Timestamp of when the process have run

1. **Provide a mockup C++ header code snippet that shows what attributes + functions are needed for tracking which instruction line is being executed by a process. E.g. Given a process with X number of instructions, how do we print the "current line number" being executed in our console?**

****

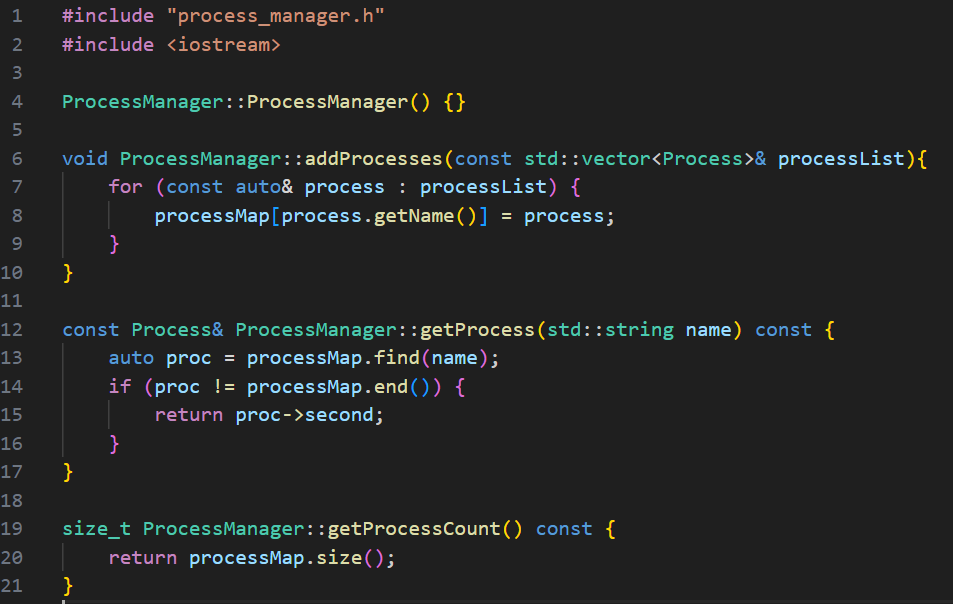
In the code, the functions related to tracking the instruction line being executed are the nextLine() and isFinished() functions. The nextLine() function simulates the process moving line per line by incrementing the current line until it reaches the total number of instructions and returns the current line through the getCurrentLine() function. The getStartTime() function stores the timestamp of when the process ran.

1. **Provide a mockup C++ header code snippet that allows one to store N processes effectively, that can be traversed in linear time.**

****

This header code serves as the process manager that keeps track of all the other processes. It uses an unordered map which allows processes to be stored using key value pairs. Adding of processes is done by passing a vector list with N processes into the addProcesses() function. The getProcessCount() function allows the program to access the current total number of processes. The map can be traversed using the total number of processes to create a basic search loop which gives us a time complexity of O(n). The getProcess() function is used to access the stored processes by name.

1. **Related to the previous question, provide a code snippet that allows one to store N processes and retrieve 1 process in O(1) time.**



Using the same file header as the previous question, we are able to store any N processes into the list using the addProcesses() function. In this function, it iterates through every element in the given vector and uses the name of the process as the *key* while the Process object is the *value*. The getProcess() function handles retrieving 1 process by name. Since we are using an unordered map, the time complexity to retrieve 1 process is O(1).