CPU Scheduler (CircularDLL.cpp)

Amy Tran

2/23/24

Class Process: Acts as the data class. Represents the type of data each node will hold. The object requires a name (string value) and a time (int value) to be created.

Public Member variables for the Process class

string processName: holds data of the process's object name.

int totalTime: hold data of the processes total time.

Parameterized Constructor*:* initializes the values imputed by the user into the variables of processName and totalTime.

Public member functions

void updateRunTime(int qTime): Function is to update the preexisting total time a process object has by the quantity indicated by qTime. The total time is to be reduced by qTime.
Precondition: input must be an int
Postcondition: total time is decreased

void print(): prints a process objects name and totaltime.
Preconditions: process object just exist
Postconditions: statement is displayed

int getTime(): returns the total time for a process object
Preconditions: must be called with a pre existing process object
Postconditions: returns a processes objects total time.

template <typename t> class Node: is a template to be used for the DoublyLinkedClass. Contains data and can be linked to other nodes.

Public member variables for the Node class

T *data: is used to hold data values within the node.

Node<T> *next: used as a pointer for the current nodes. Is used to point the current node to its next node (the node next to it).

Node <T> *prev: used a pointer for the current node to point to its previous node.

Parameterized constructor: assigns data to a singular node. Also initializes both next and prev variables with nullptr.

Public member functions

void print(): prints current nodes data value
Preconditions: A node must be created and exists before printing.
Postconditions: data value is printed

template <typename T> class CircularDDL: acts as the container for the nodes. Creates the list for which the nodes should be linked up.

Variables for CircularDLL:

Node <T> *curr: can be used as a placeholder to see which node the user is currently manipulating

int length: indicates the length of the whole circular doubly linked list

Node <T> *head: indicates which node is at the beginning of the list

Node <T> *tail: indicates which node is at the end of the list

Parameterized constructor: creates a new node by using a data value imputed by the user. Assigns the node to a current value and assigns the length of the doubly linked list to 1. Sets head and tail node both to nullptr.

Deconstructor: deallocates memory when a node is removed or destroyed.

Public member functions

void printList(): prints the entire doubly linked list
Preconditions: at least one node must exist in the list
Postconditions: list is printed

void get(int index): returns returns the node at a certain index value
Preconditions: index must be equal or greater than 0. The linked list and nodes must exist.
Postconditions: Node at the indicated index will be returned.

void insertProcess(T *data): inserts a new node with a user imputed data value to the end of the doubly circular linked list.
Preconditions: a data value must be imputed
Postconditions: preexisting doubly linked lists length is increased by 1 and a new node is added to the end of the list

void deleteProcess(): deletes a node whose data value is less than or equal to 0.
Preconditions: list must have at least one node whose data value is less than or equal to 0.
Postconditions: preexisting list is shortened by one node.

int LLength():returns the length of the preexisting linked list
Preconditions: a circular linked list must exist
Postconditions: lists length is displayed