Programming Assignment 1: Circular Doubly Linked List

Author: Kaylee Muckerman

Date of Completion: February 23, 2024

Code Description:

Process: Stores data of the process's name and total run time as well as the total time that has elapsed after the process was called.

PUBLIC MEMBER VARIABLES

string ProcessName – Stores the name of the process.

int totalTime – Stores the total time left for the process.

CONSTRUCTORS

Process(string processName, int totalTime) – Sets the processName and totalTime based on the input

PUBLIC MEMBER FUNCTIONS

void updateRunTime()

updateRunTime: Subtracts one second from totalTime everytime a quantum second passes.

Precondition – totalTime has not been corrected.

Postcondition – totalTime is updated to the correct time left.

void print()

print: Prints out the process name and how many quantum seconds are left in the runtime.

Precondition – Current process data is not printed.

Postcondition – Current process data is printed.

Node: Provides the base structure for the nodes within linked lists.

PUBLIC MEMBER VARIABLES

*T *data* – Pointer to node's data.

*Node<T> *next* – Pointer to the next node in the linked list.

*Node<T> *prev* – Pointer to the previous node in the linked list.

CONSTRUCTORS

Node(T data)* – Sets the node's data to the input data and sets next and prev node pointers to nullptr.

PUBLIC MEMBER FUNCTIONS

void print()

print: Prints the data stored in the node.

Precondition: Node's data stored is not printed.

Postcondition: Node's data stored is printed.

CircularDLL: Creates and updates a circular doubly linked list.

PRIVATE MEMBER VARIABLES

*Node<T> *head* – Stores head node of the DLL.

*Node<T> *tail* – Stores tail node of the DLL.

int length – Stores the total length of the DLL.

CONSTRUCTORS

CircularDLL() – Sets the length to 0 and sets the head and tail nodes to nullptr.

*CircularDLL(T *data)* – Sets the length to 0 and sets the head and tail nodes to nullptr. Also inserts the input Process into the DLL.

~CircularDLL() – Deallocates memory when the DLL is destroyed.

PUBLIC MEMBER FUNCTIONS

void printList()

printList: Prints a list of all the running processes in the DLL as well as the Process data.

Precondition: DLL is not printed.

Postcondition: DLL is printed.

*void insertProcess(T *data)*

insertProcess: The input Process is added to the end of the DLL and its prev pointer points to the previous tail node and its next pointer points to the head node. The head node's prev pointer points to the input process and the previous tail node’s next pointer points to the input Process.

Precondition: Input Process is not in the DLL.

Postcondition: Input Process is added to the end of the DLL.

*void deleteProcess(Node<T> *node)*

deleteProcess: Deletes the input Node by fixing it’s surrounding nodes to point to each other and freeing up the space of the deleted node.

Precondition: Input Node is still in the DLL.

Postcondition: Input Node is deleted from the DLL.

void updateTime()

updateTime: Updates the run time for each Process in the DLL.

Precondition: Run times for all Processes have not been updated.

Postcondition: Run times for all Processes have been updated.

bool isEmpty()

isEmpty: Returns true if the DLL is empty and returns false otherwise.

Precondition: State of the list is not accessed.

Postcondition: State of the list has been returned.

void deleteFinishedProcess()

deleteFinishedProcess: Checks for Processes in the DLL whose run times have been completed and deletes them.

Precondition: Processes whose run times have ended are still in the DLL.

Postcondition: Processes that are finished are deleted from the DLL.

int main() – Gets input from the user about the quantum time and which processes they want to add. It also creates a DLL to update accordingly as well as updating the processes.