ECE 250 - Project 0 – Doubly Linked List – Design Document

Easson Weisshaar, UW UsedID: etcweiss

October 13th

Overview of Classes

Class:

Node – A standard Node that includes both a next and previous pointer allowing interaction with doubly linked list

Private member variables:

***prev**: pointer to previous node, ***next**: pointer to next node, **url**: string that contains node's url, **name**: string that contains node's url-name

Public member functions:

get_next/get_prev: returns either the next node or previous node in the list, **set_next/set_prev**: takes in a pointer to a Node as a parameter, and sets the next or prev variables to said parameter based on whether it's set_next or set_prev, **get_url/get_name**: returns a string containing url or name based on the function used (name for get_name, url for get_url)

Class:

double_ll – a doubly linked list class that includes methods for adding, removing, finding, and printing nodes. Encorporates the Node class

Private member variables: *head:

pointer to the head node of the list, head is a sentinel node ****tail**: pointer to the tail node of the list, tail is a sentinel node

Public member functions:

push_front: adds a new node to the front of the deque (node after head), **push_back:** adds a new node to the back of the deque (node before tail)**, pop_front:** removes the first node (node after head)**, pop_back:** it removes the last node (node before tail)**, clear:** removes all nodes from the list**, front:** returns the first node after head**, back:** returns the first node before tail**, find:** searches for a specific given name through the deque**, print**: prints all nodes between and head and tail if deque is non-empty

Class:

Deque – A class that publicly inherits from *double_ll,* it serves to track the max size and current size of the deque and call functions from *double_ll* based on those parameters

Private member variables:

current_size a int variable that tracks the amount of nodes in the deque minus head and tail **, max_size** an int that stores that max nodes that fit into the deque at a given time

Public member functions:

is_empty: outputs a string based on whether current_size is 0 or greater **, find**: given a name to search it calls the find function from double_ll**, print**: checks whether if current_size is 0 or greater and calls print from double_ll if current_size is greater than zero**, popF**: if current_size is greater than zero it calls pop_front from double_ll, also decreases current_size by 1**, popB**: if current_size is greater than zero it calls pop_back from double_ll, also decreases current_size by 1**, pushF**: given a url and name parameter, it calls push_front from double_ll using the same parameters and increase current_size by 1, if current_size equals max_size, it'll call pop_back from double_ll first**, pushB:** given a url and name parameter, it calls push_back from double_ll using the same parameters and increase current_size by 1, if current_size equals max_size, it'll call pop_front from double_ll first**, front_d:** calls front from double_ll if current_size is greater than 0**, back_d**: calls back from double_ll if current_size is greater than 0**, deque_clear:** calls clear from double_ll and sets current_size to 0**, size_d**: returns current size of deque

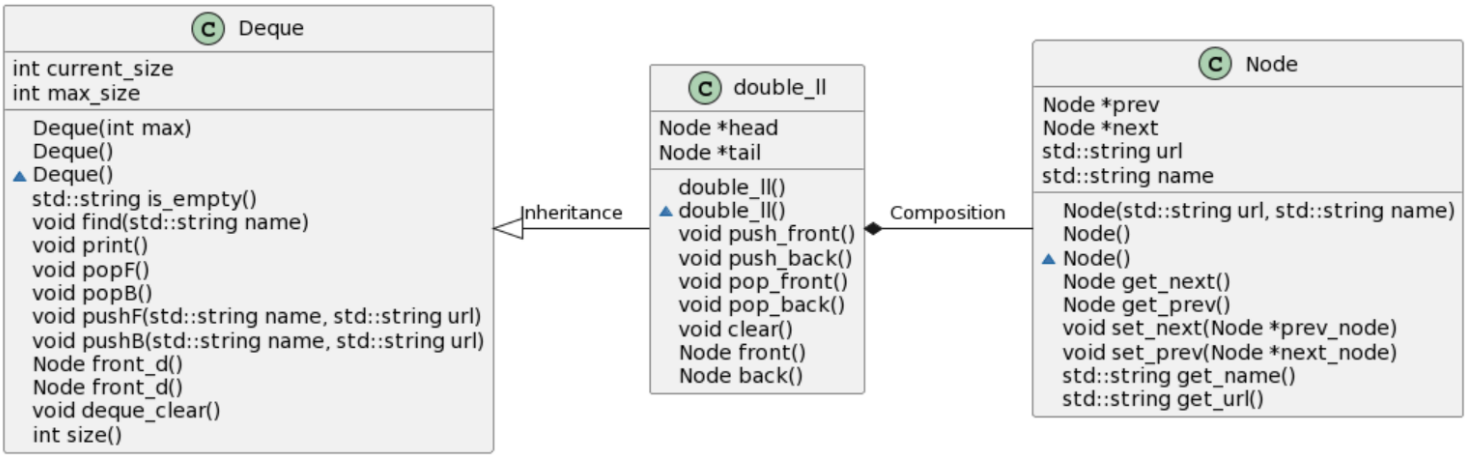
ECE 250 - Project 0 – Doubly Linked List – Design Document

Easson Weisshaar, UW UsedID: etcweiss

October 13th

UML Diagram

Classes - Double linked list deque



Constructor/Destructor:

Node:

- Given a name and url as arguments, the constructor sets the url and name to their respective inputs, also sets next and prev to nullptr
- (default) Given incorrect parameters it sets next and prev to nullptr, and name and url to an empty string
- Sets next and prev pointers to nullptr

double_ll:

- The constructor takes no parameters, all it does is initalizes two new sentinel nodes called head and tail
- The destructor deletes all nodes in the list including head and tail

Deque:

- Takes in a single argument; max, and initializes max_size to that value, also sets current_size to 0
- (default) Sets both current_size and max_size to 0 if given incorrect parameters
- The destructor sets both current_size and max_size to 0

Test Cases

Deque

- Call with any size

push_front/push_back

- Call functions as normal to test basic functionality
- Call until deque is full and see if appropriate end of deque is popped
- Call after deque clear
- Call on empty queue

pop_front/pop_back

ECE 250 - Project 0 – Doubly Linked List – Design Document

Easson Weisshaar, UW UsedID: etcweiss

October 13th

- Call on empty deque
- Call on full deque
- Call on populated deque

clear

- Call on empty
- Call on populated

find

- Search all elements present
- Search non present
- Search recently deleted elements

size

- Test basic function
- Test after list mutations (clear, pushes, pops)

Is_empty

- Test after mutations
- Test after clears
- Test on populated lists

front

- If current_size is 1, it should return same as the back function
- If empty it should fail
- Basic functionality

Back

- If current_size is 1, it should return same as the front function
- If empty it should fail
- Basic functionality

Exit

- Call to see if program ends

Performance Considerations

The following are the only operations to have O(n) runtime. Find, search, and clear. The rest have O(1) runtime. This was achieved by utilizing a head and tail sentinel node. The functions push_front, push_back, pop_front, pop_back, front, back, empty, and size are all constant run time as we always have a way to access the data despite how the list is changed. Push, pop, and front/back commands are always one before or after tail and head nodes respectively, size is stored in a variable so no traversing is required, additionally the empty function benefits from this as we can check whether current_size is greater than 0.