|  |
| --- |
| #include <iostream> |
|  | #include <string> |
|  | #include "IntList.h" |
|  |  |
|  | using namespace std; |
|  |  |
|  | IntList::IntList() { //initialize empty list |
|  | head = new IntNode(0); |
|  | tail = new IntNode(0); |
|  | tail->prev = head; |
|  | head->next = tail; |
|  | } |
|  |  |
|  | IntList::~IntList() { |
|  | if (!head && !tail) { |
|  | return; |
|  | } |
|  |  |
|  | if (head == tail) { |
|  | free(head); |
|  | head = NULL; |
|  | tail = NULL; |
|  | } |
|  | else { |
|  | IntNode\* temp = head; |
|  | while (head) { |
|  | head = temp->next; |
|  | free(temp); |
|  | temp = head; |
|  | } |
|  | tail = NULL; |
|  | } //Deallocates all remaining dynamically allocated memory(all remaining IntNodes). |
|  | } |
|  |  |
|  | void IntList:: push\_front(int value) { |
|  | //Inserts a data value(within a new node) at the front end of the list.This must be an O(1) operation. |
|  | IntNode\* newNode = new IntNode(value); |
|  | if (head->next == tail) { //list empty |
|  | head->next = newNode; |
|  | newNode->prev = head; |
|  | newNode->next = tail; |
|  | tail->prev = newNode; |
|  | } |
|  | else { |
|  | IntNode\* temp = head->next; //previous first element |
|  | head->next = newNode; |
|  | newNode->next = temp; |
|  | newNode->prev = head; |
|  | temp->prev = newNode; |
|  | } |
|  | } |
|  | void IntList::pop\_front() { |
|  | if (head->next == tail) { //list empty |
|  | return; |
|  | } |
|  | else{ |
|  | IntNode\* item1 = head->next; |
|  | IntNode\* item2 = item1->next; |
|  | head->next = item2; |
|  | item2->prev = head; |
|  | delete item1; |
|  | } |
|  | } |
|  |  |
|  | void IntList::push\_back(int value) { |
|  | IntNode\* newNode = new IntNode(value); |
|  | if (head->next == tail){ //list empty |
|  | head->next = newNode; |
|  | newNode->prev = head; |
|  | newNode->next = tail; |
|  | tail->prev = newNode; |
|  | } |
|  | else { |
|  | IntNode\* temp = tail->prev; |
|  | temp->next = newNode; |
|  | newNode->prev = temp; |
|  | newNode->next = tail; |
|  | tail->prev = newNode; |
|  | } |
|  | } |
|  | void IntList::pop\_back() { |
|  | if (head->next == tail) { //list empty |
|  | return; |
|  | } |
|  | else { |
|  | IntNode\* lastItem = tail->prev; |
|  | IntNode\* newLastItem = lastItem->prev; |
|  | newLastItem->next = tail; |
|  | tail->prev = newLastItem; |
|  | delete lastItem; |
|  | } |
|  | } |
|  |  |
|  | bool IntList::empty() const { |
|  | if (head->next == tail) { |
|  | return true; |
|  | } |
|  | else { |
|  | return false; |
|  | } |
|  | } |
|  |  |
|  |  |
|  | void IntList::printReverse() const { |
|  | IntNode\* item1 = head->next; |
|  | IntNode\* curItem = tail->prev; |
|  | if (head->next == tail) { |
|  | return; |
|  | } |
|  | while (curItem != item1){ |
|  | cout << curItem->data << " "; |
|  | curItem = curItem->prev; |
|  | } |
|  | cout << curItem->data; |
|  | } |
|  | ostream& operator<<(ostream& out, const IntList& rhs) { |
|  | if (rhs.head->next == rhs.tail) { //empty list |
|  | return out; |
|  | } |
|  | IntNode\* it = rhs.head->next; |
|  | /\*if (it->next == rhs.tail) { |
|  | out << it->data; |
|  | return out; |
|  | }\*/ |
|  | while (it != rhs.tail->prev){ |
|  | out << it->data << " "; |
|  | it = it->next; |
|  | } |
|  | out << it->data; |
|  |  |
|  | return out; |
|  | } |