IntList.H

|  |
| --- |
| // |
|  | // Created by Cristina Lawson on 2019-04-06. |
|  | // |
|  |  |
|  | #include <iostream> |
|  | using namespace std; |
|  |  |
|  | #ifndef DOUBLY\_LINKEDLIST\_INTLIST\_H |
|  | #define DOUBLY\_LINKEDLIST\_INTLIST\_H |
|  |  |
|  | struct IntNode { |
|  | int data; |
|  | IntNode \*prev; |
|  | IntNode \*next; |
|  | IntNode(int data) : data(data), prev(0), next(0) {} |
|  | }; |
|  |  |
|  | class IntList { |
|  |  |
|  | private: |
|  | IntNode\* head; |
|  | IntNode\* tail; |
|  | IntNode\* dummyHead; |
|  | IntNode\* dummyTail; |
|  |  |
|  | public: |
|  | IntList(); |
|  | ~IntList(); |
|  | void push\_front(int); |
|  | void pop\_front(); |
|  | void push\_back(int); |
|  | void pop\_back(); |
|  | bool empty() const; |
|  | friend ostream & operator<<(ostream &out, const IntList &rhs); |
|  | void printReverse() const; |
|  | }; |
|  |  |
|  |  |
|  | #endif //DOUBLY\_LINKEDLIST\_INTLIST\_H |

IntList.CPP

|  |
| --- |
| // |
|  | // Created by Cristina Lawson on 2019-04-06. |
|  | // |
|  |  |
|  | #include <iostream> |
|  | using namespace std; |
|  |  |
|  | #include "IntList.h" |
|  |  |
|  | IntList::IntList() { |
|  | dummyHead = new IntNode(0); |
|  | dummyTail = new IntNode(0); |
|  |  |
|  | head = dummyHead; |
|  | tail = dummyTail; |
|  |  |
|  | head->prev = nullptr; |
|  | head->next = tail; |
|  |  |
|  | tail->prev = head; |
|  | tail->next = nullptr; |
|  | } |
|  |  |
|  | IntList::~IntList() { |
|  | while (!empty()) { |
|  | pop\_front(); |
|  | } |
|  | delete head; |
|  | delete tail; |
|  | } |
|  |  |
|  | void IntList::push\_front(int value) { |
|  | IntNode\* temp = new IntNode(value); |
|  |  |
|  | if (empty()) { |
|  | head->next = temp; |
|  | temp->prev = head; |
|  | temp->next = tail; |
|  | tail->prev = temp; |
|  | } |
|  | else { |
|  | temp->next = head->next; |
|  | head->next->prev = temp; |
|  | head->next = temp; |
|  | temp->prev = head; |
|  | } |
|  | } |
|  |  |
|  | void IntList::pop\_front() { |
|  | IntNode\* temp = head->next; |
|  |  |
|  | if (empty()) { |
|  |  |
|  | } |
|  | else { |
|  | head->next = temp->next; |
|  | temp->next->prev = head; |
|  | delete temp; |
|  | temp = nullptr; |
|  | } |
|  | } |
|  |  |
|  | void IntList::push\_back(int value) { |
|  | IntNode\* temp = new IntNode(value); |
|  |  |
|  | if (empty()) { |
|  | push\_front(value); |
|  | } |
|  | else { |
|  | temp->next = tail; |
|  | temp->prev = tail->prev; |
|  | tail->prev = temp; |
|  | temp->prev->next = temp; |
|  | } |
|  | } |
|  |  |
|  | void IntList::pop\_back() { |
|  | IntNode\* prevNode = tail->prev; |
|  |  |
|  | if (!empty()) { |
|  | tail->prev = prevNode->prev; |
|  | prevNode->prev->next = tail; |
|  | delete prevNode; |
|  | prevNode = nullptr; |
|  | } |
|  | } |
|  |  |
|  | bool IntList::empty() const { |
|  | if (head->next != tail && tail->prev != head) { |
|  | return false; |
|  | } |
|  | else { |
|  | return true; |
|  | } |
|  | } |
|  |  |
|  | ostream & operator<<(ostream &out, const IntList &rhs) { |
|  | IntNode\* i = nullptr; |
|  | IntNode\* outNode = rhs.head; |
|  |  |
|  | for (i = outNode->next; i != rhs.tail; i = i->next) { |
|  | out << i->data; |
|  | if (i->next != rhs.tail) { |
|  | out << " "; |
|  | } |
|  | } |
|  | return out; |
|  | } |
|  |  |
|  | void IntList::printReverse() const { |
|  | IntNode\* i = nullptr; |
|  |  |
|  | for (i = tail->prev; i != head; i = i->prev) { |
|  | cout << i->data; |
|  | if (i->prev != head) { |
|  | cout << " "; |
|  | } |
|  | } |
|  | }  MAIN.CPP   |  | | --- | | #include <iostream> | |  | using namespace std; | |  |  | |  | #include "IntList.h" | |  |  | |  | int main() { | |  | cout << "Enter a test number(1-3): "; | |  | int test; | |  | cin >> test; | |  | cout << endl; | |  | //tests constructor, destructor, push\_front, pop\_front, display | |  | if (test == 1) { | |  | cout << "\nlist1 constructor called" << flush; | |  | IntList list1; | |  | cout << "\npushfront 10" << flush; | |  | list1.push\_front(10); | |  | cout << "\npushfront 20" << flush; | |  | list1.push\_front(20); | |  | cout << "\npushfront 30" << flush; | |  | list1.push\_front(30); | |  | cout << "\nlist1: " << flush; | |  | cout << list1; | |  | cout << "\nlist1 reverse: " << flush; | |  | list1.printReverse(); | |  | cout << "\npop" << flush; | |  | list1.pop\_front(); | |  | cout << "\nlist1: " << flush; | |  | cout << list1; | |  | cout << "\npop" << flush; | |  | list1.pop\_front(); | |  | cout << "\nlist1: " << flush; | |  | cout << list1; | |  | cout << "\npop" << flush; | |  | list1.pop\_front(); | |  | cout << "\nlist1: " << flush; | |  | cout << list1; | |  | cout << endl; | |  | } | |  | if (test == 1) { | |  | cout << "list1 destructor called" << endl; | |  | } | |  |  | |  | //tests push\_back | |  | if (test == 2) { | |  | cout << "\nlist2 constructor called" << flush; | |  | IntList list2; | |  | cout << "\npushback 10" << flush; | |  | list2.push\_back(10); | |  | cout << "\npushback 20" << flush; | |  | list2.push\_back(20); | |  | cout << "\npushback 30" << flush; | |  | list2.push\_back(30); | |  | cout << "\nlist2: " << flush; | |  | cout << list2; | |  | cout << "\nlist2 reverse: " << flush; | |  | list2.printReverse(); | |  | cout << "\npop" << flush; | |  | list2.pop\_front(); | |  | cout << "\nlist2: " << flush; | |  | cout << list2; | |  | cout << "\npop" << flush; | |  | list2.pop\_front(); | |  | cout << "\nlist2: " << flush; | |  | cout << list2; | |  | cout << "\npop" << flush; | |  | list2.pop\_front(); | |  | cout << "\nlist2: " << flush; | |  | cout << list2; | |  | cout << endl; | |  | } | |  | if (test == 2) { | |  | cout << "list2 destructor called" << endl; | |  | } | |  |  | |  | //tests selection\_sort | |  | if (test == 3) { | |  | cout << "\nlist3 constructor called" << flush; | |  | IntList list3; | |  | cout << "\npushfront 10" << flush; | |  | list3.push\_front(10); | |  | cout << "\npushfront 20" << flush; | |  | list3.push\_front(20); | |  | cout << "\npushfront 30" << flush; | |  | list3.push\_front(30); | |  | cout << "\nlist3: " << flush; | |  | cout << list3; | |  | cout << "\nlist3 reverse: " << flush; | |  | list3.printReverse(); | |  | cout << "\npop" << flush; | |  | list3.pop\_front(); | |  | cout << "\npop" << flush; | |  | list3.pop\_front(); | |  | cout << "\npop" << flush; | |  | list3.pop\_front(); | |  | cout << "\nlist3: " << flush; | |  | cout << list3; | |  | cout << "\npushfront 10" << flush; | |  | list3.push\_front(10); | |  | cout << "\nlist3: " << flush; | |  | cout << list3; | |  | cout << "\npushfront 20" << flush; | |  | list3.push\_front(20); | |  | cout << "\nlist3: " << flush; | |  | cout << list3; | |  | cout << "\nlist3 reverse: " << flush; | |  | list3.printReverse(); | |  | cout << endl; | |  | } | |  | if (test == 3) { | |  | cout << "list3 destructor called" << endl; | |  | } | |  |  | |  |  | |  | return 0; | |  | } | |