**IntList.h**

|  |
| --- |
| #ifndef \_INTLIST\_H\_ |
|  | #define \_INTLIST\_H\_ |
|  |  |
|  | #include <iostream> |
|  | using namespace std; |
|  |  |
|  | struct IntNode { |
|  | int data; |
|  | IntNode \*prev; |
|  | IntNode \*next; |
|  | IntNode(int data) : data(data), prev(0), next(0) {} |
|  | }; |
|  |  |
|  | class IntList { |
|  | private: |
|  | 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 |

|  |
| --- |
|  |
| #include "IntList.h" |
|  | #include <iostream> |
|  |  |
|  | // INITIALIZES AN EMPTY LIST W/ DUMMY HEAD AND DUMMY TAIL |
|  | IntList::IntList() { |
|  | dummyHead = 0; |
|  | dummyTail = 0; |
|  | } |
|  |  |
|  | // DEALLOCATES ALL REMAINING DYNAMICALLY ALLOCATED MEMORY (ALL REMAINING INTNODES) |
|  | IntList::~IntList() { |
|  | while (dummyHead != 0) { |
|  | pop\_front(); |
|  | } |
|  | } |
|  |  |
|  | //INSERTS A DATA VALUE (WITHIN A NEW NODE) AT THE FRONT END OF THE LIST |
|  | void IntList::push\_front(int value) { |
|  | IntNode\* temp = new IntNode(value); |
|  | if (dummyHead == 0) { |
|  | dummyHead = temp; |
|  | dummyTail = temp; |
|  | } |
|  |  |
|  | else { |
|  | temp->next = dummyHead; |
|  | dummyHead->prev = temp; |
|  | dummyHead = temp; |
|  | } |
|  | } |
|  |  |
|  | void IntList::pop\_front() { |
|  | if(dummyHead != 0) { |
|  | IntNode\* temp = dummyHead; |
|  | dummyHead = dummyHead->next; |
|  | if(dummyHead != 0) { |
|  | dummyHead->prev = 0; |
|  | } |
|  |  |
|  | else { |
|  | dummyTail = 0; |
|  | } |
|  | delete temp; |
|  | } |
|  | } |
|  |  |
|  | void IntList::push\_back(int value) { |
|  | IntNode\* temp = new IntNode(value); |
|  | if(dummyTail == 0) { |
|  | dummyHead = temp; |
|  | dummyTail = temp; |
|  | } |
|  | else { |
|  | dummyTail->next = temp; |
|  | temp->prev = dummyTail; |
|  | dummyTail = temp; |
|  | } |
|  |  |
|  | } |
|  |  |
|  | //REMOVES THE NODE AT THE BACK END OF THE LIST (THE NODE BEFORE THE DUMMY TAIL) |
|  | //DOES NOTHING IF LIST IS ALR EMPTY, MUST BE O(1) OPERATION |
|  | void IntList::pop\_back() { |
|  | if (!empty()) { |
|  | IntNode\* temp = dummyTail; |
|  |  |
|  | if (dummyTail == dummyHead) { |
|  | delete temp; |
|  | dummyHead = 0; |
|  | dummyTail = 0; |
|  | } |
|  |  |
|  | else { |
|  | dummyTail = dummyTail->prev; |
|  | dummyTail->next = 0; |
|  | delete temp; |
|  | } |
|  | } |
|  | } |
|  |  |
|  | // RETURNS TRUE IF THE LIST DOES NOT STORE ANY DATA VALUES (ONLY HAS DUMMY) |
|  | bool IntList::empty() const { |
|  | if (dummyHead == 0 && dummyTail == 0) { |
|  | return true; |
|  | } |
|  |  |
|  | else { |
|  | return false; |
|  | } |
|  | } |
|  |  |
|  | // GLOBAL FRIEND FUNC, OUTPUTS TO THE STREAM ALL OF THE INT VALUES WITHIN |
|  | // THE LIST ON A SINGLE LINE, EA. SEPARATED BY A SPACE. NO SPACE/NEWLINE AT END |
|  | ostream & operator<<(ostream &out, const IntList &rhs) { |
|  | IntNode \*info = rhs.dummyHead; |
|  | while (info != 0) { |
|  | out << info->data; |
|  | if (info->next != 0) { |
|  | out << ' '; |
|  | } |
|  | info = info->next; |
|  | } |
|  | return out; |
|  | } |
|  |  |
|  | // PRINTS TO SINGLE LINE ALL INT VALS STORED IN LIST IN REVERSE, NO SPACE/NEWLINE AT END |
|  | void IntList::printReverse() const { |
|  | for (IntNode \*temp = dummyTail; temp != 0; temp = temp->prev) { |
|  | cout << temp->data; |
|  | if(temp->prev != 0) { |
|  | cout << " "; |
|  | } |
|  | } |
|  | } |
|  |  |

Main.cpp

|  |
| --- |
| #include "IntList.h" |
|  | #include <iostream> |
|  | using namespace std; |
|  |  |
|  | int main() { |
|  | IntList Sample; |
|  |  |
|  | cout << "IntList push\_front() called with value of 4" << endl; |
|  | Sample.push\_front(4); |
|  | cout << Sample << endl; |
|  |  |
|  | cout << "IntList push\_back() called with value of 2" << endl; |
|  | Sample.push\_back(2); |
|  | cout << Sample << endl; |
|  |  |
|  | cout << "IntList push\_back() called with value of 3" << endl; |
|  | Sample.push\_back(3); |
|  | cout << Sample << endl; |
|  |  |
|  | cout << "IntList push\_front() called with value of 4" << endl; |
|  | Sample.push\_front(2); |
|  | cout << Sample << endl; |
|  |  |
|  | cout << "IntList pop\_back() called" << endl; |
|  | Sample.pop\_back(); |
|  | cout << Sample << endl; |
|  |  |
|  | cout << "IntList pop\_front() called" << endl; |
|  | Sample.pop\_front(); |
|  | cout << Sample << endl; |
|  |  |
|  | cout << "IntList printReverse() called" << endl; |
|  | Sample.printReverse(); |
|  | cout << endl; |
|  |  |
|  | return 0; |
|  | } |