IntList.H

|  |
| --- |
| #ifndef \_\_INTLIST\_H\_\_ |
|  | #define \_\_INTLIST\_H\_\_ |
|  |  |
|  | #include <iostream> |
|  |  |
|  | using std::ostream; |
|  |  |
|  | 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 value); |
|  | void pop\_front(); |
|  | void push\_back(int value); |
|  | void pop\_back(); |
|  | bool empty() const; |
|  | friend ostream & operator<<(ostream &out, const IntList &rhs); |
|  | void printReverse() const; |
|  | }; |
|  |  |
|  | #endif |

IntList.CPP

|  |
| --- |
| #include "IntList.h" |
|  |  |
|  | using std::cout; |
|  | using std::endl; |
|  |  |
|  | IntList::IntList() : dummyHead(new IntNode(0)), dummyTail(new IntNode(0)) { |
|  | dummyHead->next = dummyTail; |
|  | dummyHead->prev = dummyTail; |
|  | dummyTail->next = dummyHead; |
|  | dummyTail->prev = dummyHead; |
|  | } |
|  |  |
|  | IntList::~IntList() { |
|  | IntNode\* curr = dummyHead; |
|  | while (curr != dummyTail) |
|  | { |
|  | IntNode\* temp = curr; |
|  | curr = curr->next; |
|  | delete temp; |
|  | } |
|  | delete curr; |
|  | } |
|  |  |
|  | void IntList::push\_front(int value) { |
|  | IntNode\* newNode = new IntNode(value); |
|  | IntNode\* temp = dummyHead->next; |
|  | newNode->prev = dummyHead; |
|  | newNode->next = temp; |
|  | dummyHead->next = newNode; |
|  | temp->prev = newNode; |
|  | } |
|  |  |
|  | void IntList::pop\_front() { |
|  | if (!empty()) { |
|  | IntNode\* temp = dummyHead->next; |
|  | dummyHead->next = temp->next; |
|  | temp->next->prev = dummyHead; |
|  | delete temp; |
|  | } |
|  | } |
|  |  |
|  | void IntList::push\_back(int value) { |
|  | IntNode\* newNode = new IntNode(value); |
|  | IntNode\* temp = dummyTail->prev; |
|  | newNode->prev = temp; |
|  | newNode->next = dummyTail; |
|  | dummyTail->prev = newNode; |
|  | temp->next = newNode; |
|  | } |
|  |  |
|  | void IntList::pop\_back() { |
|  | if (!empty()) { |
|  | IntNode\* temp = dummyTail->prev; |
|  | dummyTail->prev = temp->prev; |
|  | temp->prev->next = dummyTail; |
|  | delete temp; |
|  | } |
|  | } |
|  |  |
|  | bool IntList::empty() const { |
|  | return dummyHead->next == dummyTail; |
|  | } |
|  |  |
|  | ostream & operator<<(ostream &out, const IntList &rhs) { |
|  | if (!rhs.empty()) { |
|  | IntNode\* curr = rhs.dummyHead->next; |
|  | while (curr->next != rhs.dummyTail) { |
|  | out << curr->data << " "; |
|  | curr = curr->next; |
|  | } |
|  | out << curr->data; |
|  | } |
|  |  |
|  | return out; |
|  | } |
|  |  |
|  | void IntList::printReverse() const { |
|  | if (!empty()) { |
|  | IntNode\* curr = dummyTail->prev; |
|  | while (curr->prev != dummyHead) { |
|  | cout << curr->data << " "; |
|  | curr = curr->prev; |
|  | } |
|  | cout << curr->data; |
|  | } |
|  | }  MAIN.CPP   |  | | --- | | #include "IntList.h" | |  |  | |  | #include <iostream> | |  |  | |  | using std::cout; | |  | using std::endl; | |  |  | |  | int main() { | |  | cout << "Creating new IntList." << endl; | |  | IntList\* list = new IntList(); | |  | cout << "Printing empty list forward: " << \*list << endl; | |  | cout << "Printing empty list in reverse: "; | |  | list->printReverse(); | |  | cout << endl << endl; | |  |  | |  | cout << "push\_front(5)" << endl; | |  | list->push\_front(5); | |  | cout << "Forward: " << \*list << endl; | |  | cout << "Reverse: "; | |  | list->printReverse(); | |  | cout << endl << endl; | |  |  | |  | cout << "push\_back(8)" << endl; | |  | list->push\_back(8); | |  | cout << "Forward: " << \*list << endl; | |  | cout << "Reverse: "; | |  | list->printReverse(); | |  | cout << endl << endl; | |  |  | |  | cout << "push\_front(9)" << endl; | |  | list->push\_front(9); | |  | cout << "Forward: " << \*list << endl; | |  | cout << "Reverse: "; | |  | list->printReverse(); | |  | cout << endl << endl; | |  |  | |  | cout << "push\_back(100)" << endl; | |  | list->push\_back(100); | |  | cout << "Forward: " << \*list << endl; | |  | cout << "Reverse: "; | |  | list->printReverse(); | |  | cout << endl << endl; | |  |  | |  | cout << "pop\_front()" << endl; | |  | list->pop\_front(); | |  | cout << "Forward: " << \*list << endl; | |  | cout << "Reverse: "; | |  | list->printReverse(); | |  | cout << endl << endl; | |  |  | |  | cout << "pop\_back()" << endl; | |  | list->pop\_back(); | |  | cout << "Forward: " << \*list << endl; | |  | cout << "Reverse: "; | |  | list->printReverse(); | |  | cout << endl << endl; | |  |  | |  | cout << "pop\_front()" << endl; | |  | list->pop\_front(); | |  | cout << "Forward: " << \*list << endl; | |  | cout << "Reverse: "; | |  | list->printReverse(); | |  | cout << endl << endl; | |  |  | |  | cout << "pop\_back()" << endl; | |  | list->pop\_back(); | |  | cout << "Forward: " << \*list << endl; | |  | cout << "Reverse: "; | |  | list->printReverse(); | |  | cout << endl << endl; | |  |  | |  | cout << "pop\_back() empty list" << endl; | |  | list->pop\_back(); | |  | cout << "Forward: " << \*list << endl; | |  | cout << "Reverse: "; | |  | list->printReverse(); | |  | cout << endl << endl; | |  |  | |  | cout << "pop\_front() empty list" << endl; | |  | list->pop\_front(); | |  | cout << "Forward: " << \*list << endl; | |  | cout << "Reverse: "; | |  | list->printReverse(); | |  | cout << endl; | |  |  | |  | cout << "Destructor (deleting list)" << endl; | |  | delete list; | |  | cout << "Printing forward deleted pointer. Program will crash!" << endl; | |  | cout << "Forward: " << \*list << endl; | |  | return 0; | |  | } | |