#include <iostream>

#include <vector>

#include <stack>

using namespace std;

int main() {

vector<int> vec;

vec.push\_back(50);

vec.push\_back(81);

cout<<vec.size();

for (vector<int>::iterator it = vec.begin(); it != vec.end(); it++) {

cout << '\t' << \*it;

}

stack<int> stk;

stk.push(50);

stk.push(81);

while (!stk.empty()) {

cout << '\t' << stk.top();

stk.pop();

}

cout<<endl;

system("pause");

return 0;

}

#include <iostream>

#include <list>

using namespace std;

class myList: public list<int>{

public:

void join(){

for (myList::iterator iter1 = begin(); iter1 != end(); ++iter1) {

myList::iterator iter2 = iter1;

for (++iter2; iter2 != end(); ++iter2) {

if (\*iter1 == \*iter2) {

iter2 = erase(iter2);

if (iter2 == end()) {

break;

}

}

}

}

}

};

int main()

{

myList ls1;

ls1.push\_back(12);

ls1.push\_back(2);

ls1.push\_back(5);

ls1.push\_back(2);

ls1.push\_back(11);

ls1.push\_back(3);

ls1.push\_back(2);

ls1.push\_back(11);

for (myList::iterator iter = ls1.begin(); iter != ls1.end(); iter++) {

cout << \*iter << '\t';

}

ls1.join();

cout << endl;

for (myList::iterator iter = ls1.begin(); iter != ls1.end(); iter++) {

cout << \*iter << '\t';

}

return 0;

}

#include <iostream>

#include "List.h"

using namespace std;

class ListArray : public List{

public:

Link \* operator[](int index) const{

if (index < 0) throw ("Error - negative argument!");

Link \* current = first;

while (true) {

if (current == NULL) throw ("Error - the list does not have this index!");

if (!index) return current;

current = current->next;

--index;

}

}

};

void main()

{

int element;

ListArray ls1,ls2;

try

{

for(int i=0;i<5;i++)

{

ls1.add(i);

cout << i << " ";

}

cout << endl;

ls1.removeFirst();

for(int i=0;i<4;i++)

{

element=ls1[i]->value;

cout << element << " ";

ls2.add(element);

}

cout << endl;

cout << ((ls2.includes(4))? "ls2 includes 4" : "ls2 doesn't include 4") << endl;

cout << ((ls2.includes(0))? "ls2 includes 0" : "ls2 doesn't include 0") << endl;

ls2.removeFirst();

cout << ((ls2.includes(0))? "ls2 includes 0" : "ls2 doesn't include 0") << endl;

cout << ls2[1]->value;

ls2.removeFirst();

cout << ls2[1]->value;

}

catch (char\* problem)

{

cout<< problem;

}

}