//

// main.cpp

// 10c

//

// Created by Jeff on 10/19/16.

// Copyright © 2016 Jeff zhang. All rights reserved.

//

#include <iostream>

#include "stack.h"

#include <vector>

using namespace std;

int main(int argc, const char \* argv[]) {

int catchVar; // To hold values popped off the stack

// Define a stack object to hold 5 values.

Stack<int> stk(5);

// Push the values is, very, interesting, and, fun onto the stack.

for(int i=1; i<=6; i++) {

cout << "...push " << i\*5 << "\n" ;

try{

stk.push(i\*5);

}catch(Stack<int>::Excep\_over){

cout<<" .ERR.The stack is full."<<endl;

}

// stk.push(i\*5);

stk.display();

}

cout << endl;

// Pop the values off the stack.

for(int i=1; i<=6; i++) {

cout << "...pop\n";

try{

stk.pop(catchVar);

}catch(Stack<int>::Excep\_Under){

cout << ".ERR.The stack is empty."<<endl;

}

// stk.pop(catchVar);

stk.display();

}

};

#ifndef StringStack\_h

#define StringStack\_h

//

// InsStack.h

// lab1

//

// Created by Jeff on 10/17/16.

// Copyright © 2016 Jeff zhang. All rights reserved.

//

#include <string>

#include <iostream>

using namespace std;

template<class T>

class Stack{

private:

T \*stackArray;

int stackCapacity;

int top;

public:

class Excep\_over{ };

class Excep\_Under{ };

Stack(int size){

// dynimic assement

stackArray = new T[size];

stackCapacity =size;

top = -1;

}

~Stack(){

delete[]stackArray;

}

// destructor();

void push(T item){

if(isFull())

// cout<<" .ERR.The stack is full."<<endl;

throw Excep\_over();

else

{

top++;

stackArray[top] = item;

}

}

void pop(T &item){

if(isEmpty()){

throw Excep\_Under();

}

else{

item = stackArray[top];

top--;

}

}

bool isFull(){

if(top >= stackCapacity - 1) return true;

return false;

}

bool isEmpty(){

if(top== -1) return true;

return false;

}

void display(){

if(isEmpty())

cout << ".ERR.The stack is empty."<<endl;

else{

cout << ".STK. ";

for(int i =0 ; i <=top;i++){

cout << stackArray[i]<<",";

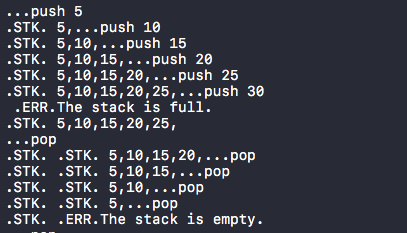
}

}

}

};

#endif /\* stack\_h \*/



//

// main.cpp

// lab1

//

// Created by Jeff on 10/17/16.

// Copyright © 2016 Jeff zhang. All rights reserved.

//

#include <iostream>

#include "IntStack.h"

using namespace std;

int main(int argc, const char \* argv[]) {

int catchVar; // To hold values popped off the stack

// Define a stack object to hold 5 values.

IntStack stk(5);

// Push the values is, very, interesting, and, fun onto the stack.

for(int i=1; i<=6; i++) {

cout << "...push " << i\*5 << "\n" ;

try{

stk.push(i\*5);

}catch(IntStack::Excep\_over){

cout<<" .ERR.The stack is full."<<endl;

}

// stk.push(i\*5);

stk.display();

}

cout << endl;

// Pop the values off the stack.

for(int i=1; i<=6; i++) {

cout << "...pop\n";

try{

stk.pop(catchVar);

}catch(IntStack::Excep\_Under){

cout << ".ERR.The stack is empty."<<endl;

}

// stk.pop(catchVar);

stk.display();

}

};

//

// InsStack.h

// lab1

//

// Created by Jeff on 10/17/16.

// Copyright © 2016 Jeff zhang. All rights reserved.

//

#ifndef IntStack\_h

#define IntStack\_h

#include <iostream>

using namespace std;

class IntStack{

private:

int \*stackArray;

int stackCapacity;

int top;

public:

class Excep\_over{ };

class Excep\_Under{ };

IntStack(int size){

// dynimic assement

stackArray = new int[size];

stackCapacity =size;

top = -1;

}

~IntStack(){

delete[]stackArray;

}

void destructor();

void push(int item){

if(isFull())

// cout<<" .ERR.The stack is full."<<endl;

throw Excep\_over();

else

{

top++;

stackArray[top] = item;

}

}

void pop(int &item){

if(isEmpty())

// cout << ".ERR.The stack is empty."<<endl;

throw Excep\_Under();

else{

cout << ".STK. ";

// for(int i =0 ; i <=top;i++){

//

// cout << stackArray[i]<<",";

//

// }

item = stackArray[top];

top--;

}

}

bool isFull(){

if(top >= stackCapacity - 1) return true;

return false;

}

bool isEmpty(){

if(top== -1) return true;

return false;

}

void display(){

if(isEmpty())

cout << ".ERR.The stack is empty."<<endl;

else{

cout << ".STK. ";

for(int i =0 ; i <=top;i++){

cout << stackArray[i]<<",";

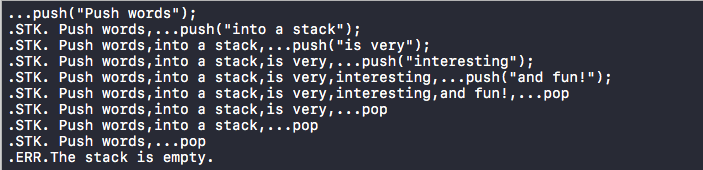
}

}

}

};

#endif /\* InsStack\_h \*/



//

// main.cpp

// lab1

//

// Created by Jeff on 10/17/16.

// Copyright © 2016 Jeff zhang. All rights reserved.

//

#include <iostream>

#include "StringStack.h"

using namespace std;

int main(int argc, const char \* argv[]) {

string catchVar; // To hold values popped off the stack

// Define a stack object to hold 5 values.

StringStack stk(5);

// Push the values is, very, interesting, and, fun onto the stack.

cout << "...push(\"Push words\");\n";

stk.push("Push words"); stk.display();

cout << "...push(\"into a stack\");\n";

stk.push("into a stack"); stk.display();

cout << "...push(\"is very\");\n";

stk.push("is very"); stk.display();

cout << "...push(\"interesting\");\n";

stk.push("interesting"); stk.display();

cout << "...push(\"and fun!\");\n";

stk.push("and fun!"); stk.display();

try {

stk.pop(catchVar);

} catch (StringStack::Excep\_over) {

std::cout << ".EXP. Over !\n";

}

// Pop the values off the stack.

cout << "...pop\n";

stk.pop(catchVar); stk.display();

cout << "...pop\n";

stk.pop(catchVar); stk.display();

cout << "...pop\n";

stk.pop(catchVar); stk.display();

cout << "...pop\n";

stk.pop(catchVar); stk.display();

cout << "...pop\n";

// stk.pop(catchVar); stk.display();

// cout << "...pop\n";

// stk.pop(catchVar); stk.display();

//

try {

stk.pop(catchVar);

} catch (StringStack::Excep\_Under) {

std::cout << ".EXP. Empty Stack!\n";

}

stk.display();

cout << "\n exit program " <<endl;

return 0;

};

//

// StringStack.h

// lab2

//

// Created by Jeff on 10/17/16.

// Copyright © 2016 Jeff zhang. All rights reserved.

//

#ifndef StringStack\_h

#define StringStack\_h

//

// InsStack.h

// lab1

//

// Created by Jeff on 10/17/16.

// Copyright © 2016 Jeff zhang. All rights reserved.

//

#include <string>

#include <iostream>

using namespace std;

class StringStack{

private:

string \*stackArray;

int stackCapacity;

int top;

public:

class Excep\_over{ };

class Excep\_Under{ };

StringStack(int size){

// dynimic assement

stackArray = new string[size];

stackCapacity =size;

top = -1;

}

~StringStack(){

delete[]stackArray;

}

// destructor();

void push(string item){

if(isFull())

// cout<<" .ERR.The stack is full."<<endl;

throw Excep\_over();

else

{

top++;

stackArray[top] = item;

}

}

void pop(string &item){

if(isEmpty()){

throw Excep\_Under();

}

else{

item = stackArray[top];

top--;

}

}

//}

// void pop(int item){

// int item =911;

// if(isEmpty())

// cout << ".ERR.The stack is empty."<<endl;

// else{

// item = stackArray[top];

// top--;

//

// } return item;

// }

bool isFull(){

if(top >= stackCapacity - 1) return true;

return false;

}

bool isEmpty(){

if(top== -1) return true;

return false;

}

void display(){

if(isEmpty())

cout << ".ERR.The stack is empty."<<endl;

else{

cout << ".STK. ";

for(int i =0 ; i <=top;i++){

cout << stackArray[i]<<",";

}

}

}

};

#endif /\* StringStack\_h \*/

