===================== опис інтерфейсу ===============

#pragma once

\_\_interface IElementOperation {

void add(int element); //Додавання у колекцію нового елемента

bool remove(int element); //Видалення першого входження (false - якщо не видалив жодного)

void removeAt(int index); //Видаляє елемент у вказаній позиції

int indexOf(int value); //Пошук першого входження елемента (якщо немає то повертає -1)

int removeAll(int element); //Видаляє усі вказані значення (повертає кількість видалених)

void insert(int index, int element); //Вставляє елемент всередину

};

================= клас,що реалізовує інтерфейс ===========

#pragma once

#include "IElementOperation.h"

#include <string>

using namespace std;

class IntArray :

public IElementOperation

{

private:

int\* elements=0;

int maxLength; //Максимальна можлива на даний момент кількість

int length; //Справжня кількість елементів

void set\_maxLength(int value);

bool isIndexCorrect(int index);

public:

int getMaxLength();

int getLength();

void set\_element(int index, int value);

int get\_element(int index);

IntArray(int maxLength);

IntArray(int length, int firstElement,...);

IntArray();

void add(int element); //Додавання у колекцію нового елемента

bool remove(int element); //Видалення першого входження (false - якщо не видалив жодного

void removeAt(int index); //Видаляє елемент у вказаній позиції

int indexOf(int value); //Пошук першого входження елемента (якщо немає то повертає -1)

int removeAll(int element); //Видаляє усі вказані значення (повертає кількість видалених)

void insert(int index, int element); //Вставляє елемент всередину

int& operator[](int index);

string toString();

~IntArray();

};

#include "stdafx.h"

#include "IntArray.h"

void IntArray::set\_maxLength(int value)

{

if (value <= 0)

throw "error length < 0";

if (elements==0)

{

maxLength = value;

elements = new int[maxLength];

}

else

{

int\* temp = new int[maxLength \* 2];

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

{

temp[i] = elements[i];

}

delete[]elements;

elements = temp;

maxLength = maxLength\*2;

}

}

bool IntArray::isIndexCorrect(int index)

{

return index>=0 && index<length;

}

int IntArray::getMaxLength()

{

return maxLength;

}

int IntArray::getLength()

{

return length;

}

void IntArray::set\_element(int index, int value)

{

if (!isIndexCorrect(index))

throw "index incorrect";

elements[index] = value;

}

int IntArray::get\_element(int index)

{

if (!isIndexCorrect(index))

throw "index incorrect";

return elements[index];

}

IntArray::IntArray(int maxLength)

{

set\_maxLength(maxLength);

length = 0;

}

IntArray::IntArray(int length, int firstElement, ...):IntArray(length)

{

this->length = length;

int\* temp = &firstElement;

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

{

elements[i] = temp[i];

}

}

IntArray::IntArray() :IntArray(20)

{

}

void IntArray::add(int element)

{

if (length == maxLength)

set\_maxLength(maxLength \* 2);

elements[length] = element;

length++;

}

bool IntArray::remove(int element)

{

int index = indexOf(element);

if (index==-1)

return false;

removeAt(index);

return true;

}

void IntArray::removeAt(int index)

{

if (!isIndexCorrect(index))

throw "index incorrect";

for (int i = index+1; i < length; i++)

{

elements[i - 1] = elements[i];

}

length--;

}

int IntArray::indexOf(int value)

{

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

{

if (elements[i] == value)

return i;

}

return -1;

}

int IntArray::removeAll(int element)

{

int removedElementsCount = 0;

int index;

while ( (index=indexOf(element)) != -1 )

{

removeAt(index);

removedElementsCount++;

}

return removedElementsCount;

}

void IntArray::insert(int index, int element)

{

if (!isIndexCorrect(index))

throw "index incorrect";

if (length == maxLength)

set\_maxLength(maxLength \* 2);

for (int i = length-1; i >= index; i--)

{

elements[i + 1] = elements[i];

}

elements[index] = element;

length++;

}

int & IntArray::operator[](int index)

{

if (!isIndexCorrect(index))

throw "index incorrect";

return elements[index];

}

string IntArray::toString()

{

if (length == 0)

return "array is empty";

string s = to\_string(elements[0]);

for (int i = 1; i < length; i++)

{

s += ", " + to\_string(elements[i]);

}

return s;

}

IntArray::~IntArray()

{

delete[]elements;

}

// ConsoleApplication38.cpp : Defines the entry point for the console application.

//

#include "stdafx.h"

#include <iostream>

#include "IElementOperation.h"

#include "IntArray.h"

using namespace std;

//================

void addRandomNumbers(IElementOperation\* collections, int min, int max, int numbersCount)

{

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

{

collections->add(min + rand() % (max - min + 1));

}

}

int main()

{

IntArray\* array = new IntArray();

addRandomNumbers(array, 1, 10, 5);

printf("Array: %s\n",array->toString().data());

printf("Remove element:");

int element;

cin >> element;

array->remove(element);

printf("Array: %s\n", array->toString().data());

system("pause");

return 0;

}