#include<iostream>

#include<assert.h>

using namespace std;

class Vector {

int\* data;

size\_t size = 0;

public:

Vector() :data(nullptr), size(0) {}

Vector(int\* data, size\_t size) {

SetSize(size);

SetData(data);

}

int\* GetData()const {

return data;

}

size\_t GetSize()const {

return size;

}

void SetData(int\* data) {

assert(data != nullptr);

this->data = data;

}

void SetSize(const size\_t& size) {

this->size = size;

}

void Print() const {

for (size\_t i = 0; i < size; i++)

{

cout << data[i] << " ";

} cout << endl;

}

void PushBack(int newdata) {

auto newarray = new int[size + 1]{};

for (size\_t i = 0; i < size; i++)

{

newarray[i] = data[i];

}

newarray[size] = newdata;

if (data != nullptr) {

delete[]data;

}

data = newarray;

newarray = nullptr;

++size;

}

int operator[](int index) {

assert(index >= 0 && index < size && "Our of range error");

return data[index];

}

int operator()(int index) {

assert(index >= 0 && index < size && "Out of range error");

return data[index];

}

bool isReverse(int data1, int data2, bool reverse) {

if (reverse) {

if (data1 < data2)return true;

return false;

}

else {

if (data1 > data2)return true;

return false;

}

}

void Sort(bool reverse = false) {

for (size\_t i = 0; i < size - 1; i++)

{

bool swapped = false;

for (size\_t k = 0; k < size - i - 1; k++)

{

if (isReverse(data[k], data[k + 1], reverse)) {

swapped = true;

int temp = data[k];

data[k] = data[k + 1];

data[k + 1] = temp;

}

}

if (!swapped) {

break;

}

}

}

int Search() {

}

int BinarySearch() {

Sort();

}

void Pop(int index = -1) {

if (index == -1) {

index = size - 1;

}

int\* newarray = new int[size - 1];

for (size\_t i = 0; i < index; i++)

{

newarray[i] = data[i];

}

for (size\_t i = index; i < size - 1; i++)

{

newarray[i] = data[i + 1];

}

if (data != nullptr) {

delete[]data;

}

data = newarray;

newarray = nullptr;

--size;

}

~Vector()

{

delete[]data;

}

};

void main() {

Vector myvector;

myvector.PushBack(20);

myvector.PushBack(10);

myvector.PushBack(1);

myvector.PushBack(18);

myvector.PushBack(-30);

//myvector.Print();

//cout << "Sorted" << endl;

//myvector.Sort(true);

myvector.Print();

myvector.Pop(2);

myvector.Print();

//cout << myvector(1) << endl;

//push back

// push front

// InsertByIndex

//search

//getByIndex

}