// ConsoleApplication18.cpp : Defines the entry point for the console application.

//

#include "stdafx.h"

#include<iostream>

#include<string>

using namespace std;

//------------------ ICoparable -------

//Інтерфейс, який використовується для успадкування у випадку, коли хочемо

// надати можливість класу, що його успадковує порівнювати об"єкти за деякою ознакою

//Для того, щоб його могли успадковувати довільні класи викорисатно шаблонний тип даних

template <class MyType>

\_\_interface IComparable

{

int compareTo(MyType\* obj);

};

//----------- Student ------------

//Клас успадковує інтерфейс IComparable вказуючи, що шаблонний тип для інтерфейсу буде клас Student

class Student :public IComparable<Student>

{

public:

int age;

string name;

Student(int age, string name) :age(age), name(name) {

}

Student() :Student(0, "no name") {

cout << "----" << endl;

}

int compareTo(Student\* obj) {

return age - obj->age;

}

string toString()

{

return name+": "+to\_string(age);

}

};

//------------------- Client -------------------

class Client:public IComparable<Client>

{

public:

string ID;

string name;

double account;

Client(string ID, string name, double account):ID(ID), name(name), account(account)

{}

int compareTo(Client\* obj) {

return account - obj->account;

}

string toString()

{

return ID + ": " +name+ ": " + to\_string(account);

}

};

//----------------------- Array ----------------

//Клас, який може зберігати масив покажчиків довільного типу (використано шаблонний тип)

//Але для того, щоб була можливість сортування необхідно, щоб тип реалізовував інтерфейс IComparable

template<class MyType>

class Array

{

public:

MyType\*\* data;

int length;

Array(MyType\*\* data, int length) {

this->data = data;

this->length = length;

}

Array() {

}

void sort()

{

try {

for (int i = 0; i < length - 1; i++)

{

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

{

IComparable<MyType>\* z1 = (IComparable<MyType>\*)(data[i]);

if (z1->compareTo(data[j])>0)

{

MyType\* t = data[i];

data[i] = data[j];

data[j] = t;

}

}

}

}

catch (...)

{

throw "type must implement IComparable";

}

}

MyType& operator[](int index)

{

return data[index];

}

void show(string message)

{

cout << message << endl;

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

{

cout << data[i]->toString()<<endl;

}

}

};

int main()

{

Student\*\* students = new Student\*[2];

Student s1(222, "Ivan");

Student s2(67, "Petro");

students[0] = new Student(222, "Ivan");

students[1] = new Student (67, "Petro");

Array<Student> arr(students,2); //Створюємо об"єкт - масив студентів

arr.sort();

arr.show("---------------- Sorted students -----------");

//-------------------------------

Client\*\* clients = new Client\*[3];

clients[0]=new Client("Q11","Ivan",21400);

clients[1]=new Client("Q212","Petro",3210);

clients[2] = new Client("Q332", "Olja", 50000);

Array<Client> arr2(clients, 3); //Створюємо об"єкт - масив клієнтів

arr2.sort();

arr2.show("---------------- Sorted clients -------------");

system("pause");

return 0;

}