МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Кафедра ЕОМ



Звіт до лабораторних робіт

з ООП

Виконав:

ст. групи КІ-15

Ковальський П.А.

Прийняв:

викл. Козак Н.Б.

Львів-2020

ЛАБОРАТОРНА РОБОТА № 7.

МНОЖИННЕ СПАДКУВАННЯ. ПОЛІМОРФІЗМ

**Мета:** познайомитися із множинним спадкуванням класів та поліморфізмом.

**ТЕОРЕТИЧНІ ВІДОМОСТІ**

**Множинне спадкуування**

Якщо спадкування здійснюється від декількох батьківських класів одночасно, тоді воно називається множинним спадкуванням. Визначальним для похідного класу породженого множинним спадкуванням є те, що він явно чи неявно повинен успадковувати характеристики декількох базових класів. Основні принципи одинарного спадкування, зокрема спадкування членів, модифікаторів доступу до членів базових класів, розширення та обмеження характеристик, без жодних доповнень можуть бути перенесені на множинне спадкування. Неявним множинним спадкуванням можуть бути випадки змішаного спадкування. Результатом цих спадкуваннь є ієрархія, в якій похідний клас неявно (через один проміжний) успадкував характеристики двох базових класів. Якщо похідний клас породжується від декількох базових, то в декларації класу треба вказати усі базові класи, розділяючи їх комою, разом зі специфікаторами спадкування. У загальному випадку синтаксис множинного спадкування має вигляд:

class Ім'яПохідногоКласу : \*модифікатор+ Ім'яБазовогоКласу1, ..., \*модифікатор+ Ім'яБазовогоКласуN ,тіло класу-;

Розглянемо узагальнений приклад множинного спадкування:

class A ,оголошення класу-; class B , оголошення класу -; class C : public A , оголошення класу -; class D: public B, public C , оголошення класу -; // аналогічно оголошенню class D: public B, C, оголошення класу -;

Конструктори при множинному спадкуванні

Розглянемо алгоритм роботи конструкторів при множинному спадкуванні. При створенні об'єкта класу, який множинно породжений, після виклику конструктора похідного класу викликатиметься конструктор найпершого батьківського класу. Якщо він є похідним від ще одного класу, то викликатиметься і буде виконуватись конструктор останнього. По закінченню роботи усіх конструкторів по гілці дерева від найпершого класу, розпочне виконуватись гілка від другого батьківського класу і т.д. Після того, як відпрацюють конструктори усіх батьківських класів, виконається тіло конструктора похідного класу.

**Варіант-12**



**Код програми:**

**Source.cpp:**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include "CSmartPhone.h"

int main()

{

cout << "----- Creating objects -----" << endl;

CPhone a1((char\*)("Garold"), (char\*)("Nomi"));

a1.SetWeight(0.450);

a1.ShowFullInformation();

a1.Call("+38(073)-999-99-99");

a1.Rington("Lil Peep - Switchblades");

CComputer a2((char\*)("Garold"), (char\*)("Mac"));

a2.SetWeight(2.5);

a2.ShowFullInformation();

a2.PlayGame("CS:GO");

a2.UpdateOS(1.2);

CSmartPhone a3((char\*)("Garold"), (char\*)("Mac"), (char\*)("Meizu"), 75, 1234);

a3.SetWeight(0.150);

a3.ShowFullInformation();

a3.UnlockWithPassword(1234);

a3.UnlockWithPassword(2222);

a3.FindInBrowser("http\\nulp.com.ua");

CSmartPhone a4 = a3;

a4.ShowFullInformation();

CSmartPhone a5(a4);

a5.ShowFullInformation();

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

cout << "----- Creating reference -----" << endl;

cout << "----- CDevice &ref1 = a1; -----" << endl;

CDevice &ref1 = a1;

ref1.ShowFullInformation();

cout << "----- ref1 = a2; -----" << endl;

ref1 = a2;

ref1.ShowFullInformation();

cout << "----- CDevice &ref2 = a2; -----" << endl;

CDevice &ref2 = a2;

ref2.ShowFullInformation();

cout << "----- CDevice &ref3 = a2; -----" << endl;

CDevice &ref3 = a3;

ref3.ShowFullInformation();

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

cout << "----- Creating dynamic objects -----" << endl;

cout << "----- CDevice\* ptr1 = new CPhone((char\*)(\"Robert\"), (char\*)(\"Simens\")) -----" << endl;

CDevice\* ptr1 = new CPhone((char\*)("Robert"), (char\*)("Simens"));

ptr1->SetWeight(0.050);

ptr1->ShowProperty();

ptr1->ShowFullInformation();

cout << "----- CDevice\* ptr2 = new CComputer((char\*)(\"Pit\"), (char\*)(\"eMachines\")); -----" << endl;

CDevice\* ptr2 = new CComputer((char\*)("Pit"), (char\*)("eMachines"));

ptr2->SetWeight(5.6);

ptr2->ShowProperty();

ptr2->ShowFullInformation();

cout << "----- CDevice\* ptr3 = new CSmartPhone((char\*)(\"Jack\"), (char\*)(\"Hp\"), (char\*)(\"Nokia\"), 65); -----" << endl;

CDevice\* ptr3 = new CSmartPhone((char\*)("Jack"), (char\*)("Hp"), (char\*)("Nokia"), 65, 4321);

ptr3->SetWeight(0.225);

ptr3->ShowProperty();

ptr3->ShowFullInformation();

cout << "----- CPhone \*ptr4 = new CPhone((char\*)(\"Top\"), (char\*)(\"Xiaomi\")); -----" << endl;

CPhone \*ptr4 = new CPhone((char\*)("Top"), (char\*)("Xiaomi"));

ptr4->SetWeight(0.125);

ptr4->ShowProperty();

ptr4->ShowFullInformation();

cout << "----- CComputer \*ptr5 = new CComputer((char\*)(\"Doms\"), (char\*)(\"SimpsonPC\")); -----" << endl;

CComputer \*ptr5 = new CComputer((char\*)("Doms"), (char\*)("SimpsonPC"));

ptr5->SetWeight(4.3);

ptr5->ShowProperty();

ptr5->ShowFullInformation();

cout << "----- CSmartPhone \*ptr6 = new CSmartPhone((char\*)(\"Vadym\"), (char\*)(\"Dell\"), (char\*)(\"Iphone\"),38); -----" << endl;

CSmartPhone \*ptr6 = new CSmartPhone((char\*)("Vadym"), (char\*)("Dell"), (char\*)("Iphone"), 38, 1111);

ptr6->SetWeight(0.185);

ptr6->ShowProperty();

ptr6->ShowFullInformation();

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

cout << "delete ptr1 " << endl;

delete ptr1;

cout << "-------------------------" << endl << "delete ptr2 " <<

endl;

delete ptr2;

cout << "-------------------------" << endl << "delete ptr3 " <<

endl;

delete ptr3;

cout << "-------------------------" << endl << "delete ptr4 " <<

endl;

delete ptr4;

cout << "-------------------------" << endl << "delete ptr5 " <<

endl;

delete ptr5;

cout << "-------------------------" << endl << "delete ptr6 " <<

endl;

delete ptr6;

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

system("pause");

return 0;

}

**CSmartPhone.cpp:**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include "CSmartPhone.h"

CSmartPhone::CSmartPhone() : CDevice(), CPhone(), CComputer()

{

cout << "CSmartPhone Default Constructor" << endl;

}

CSmartPhone::CSmartPhone(char\* name, char\* computerName, char\*phoneName, int perfomance, int password) : CDevice(name), CPhone(name, phoneName), CComputer(name, computerName)

{

m\_Perfomance = perfomance;

m\_Password = password;

cout << "CSmartPhone Constructor" << endl;

}

CSmartPhone::CSmartPhone(const CSmartPhone& obj) : CDevice(obj),

CPhone(obj), CComputer(obj)

{

cout << "CSmartPhone Copy Constructor" << endl;

this->m\_Perfomance = obj.m\_Perfomance;

}

CSmartPhone::~CSmartPhone()

{

cout << "CSmartPhone Destructor" << endl;

}

void CSmartPhone::ShowProperty()

{

cout << "Phone: " << m\_PhoneName << " and Computer: " << m\_ComputerName << endl;

cout << "My perfomance is: " << m\_Perfomance << "\%" << endl;

}

void CSmartPhone::ShowFullInformation()

{

cout << "My name is " << m\_CreatorName << ", my weight is " <<

m\_Weight << " kg" << endl;

ShowProperty();

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

}

CSmartPhone& CSmartPhone::operator=(CSmartPhone& obj)

{

if (this != &obj)

{

cout << "CSmartPhone operator=" << endl;

CDevice::operator=(obj);

CPhone::operator=(obj);

CComputer::operator=(obj);

this->m\_Perfomance = obj.m\_Perfomance;

this->m\_Password = obj.m\_Password;

}

return \*this;

}

void CSmartPhone::UnlockWithPassword(int password)

{

{

if (password == m\_Password)

{

cout << "Unlocked" << endl;

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

}

else

{

cout << "Wrong password" << endl;

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

}

}

}

void CSmartPhone::FindInBrowser(const char\* link)

{

cout << "Opening page by link " << link << endl;

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

}

**CSmartPhone.h:**

#include "CPhone.h"

#include "CComputer.h"

#ifndef SMART

#define SMART

class CSmartPhone : public CPhone, CComputer

{

private:

int m\_Perfomance;

int m\_Password;

public:

CSmartPhone();

CSmartPhone(char\* creatorName, char\* computerName, char\* phoneName, int perfomance, int password);

CSmartPhone(const CSmartPhone& obj);

virtual ~CSmartPhone();

void ShowProperty();

virtual void ShowFullInformation();

CSmartPhone& operator=(CSmartPhone&);

void UnlockWithPassword(int password);

void FindInBrowser(const char\*);

};

#endif

**CPhone.cpp:**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include "CPhone.h"

CPhone::CPhone() : CDevice()

{

cout << "CAutoRacer Default Constructor" << endl;

m\_PhoneName = new char[20];

strcpy(m\_PhoneName, "No name");

}

CPhone::CPhone(char\* creatorName, char\* phoneName) : CDevice(creatorName)

{

cout << "CPhone Constructor" << endl;

m\_PhoneName = new char[strlen(phoneName) + 1];

strcpy(m\_PhoneName, phoneName);

}

CPhone::CPhone(const CPhone& obj) : CDevice(obj)

{

cout << "CPhone Copy Constructor" << endl;

m\_PhoneName = new char[strlen(obj.m\_PhoneName) + 1];

strcpy(m\_PhoneName, obj.m\_PhoneName);

}

CPhone::~CPhone()

{

cout << "CPhone Destructor" << endl;

delete[] m\_PhoneName;

}

void CPhone::ShowProperty()

{

cout << "Phone name: " << m\_PhoneName << endl;

}

void CPhone::ShowFullInformation()

{

cout << "My name is " << m\_CreatorName << ", my weight is " <<

m\_Weight << " kg" << endl;

ShowProperty();

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

}

CPhone& CPhone::operator=(CPhone& obj)

{

if (this != &obj)

{

cout << "CPhone operator=" << endl;

delete[] m\_PhoneName;

m\_CreatorName = new char[strlen(obj.m\_PhoneName) + 1];

strcpy(m\_PhoneName, obj.m\_PhoneName);

CDevice::operator=(obj);

}

return \*this;

}

void CPhone::Call(const char\* number)

{

cout << "Calling number " << number << endl;

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

}

void CPhone::Rington(const char\* rington)

{

cout << "Playing rington " << rington << endl;

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

}

**CPhone.h:**

#include "CDevice.h"

#include <cstring>

#ifndef PHONE

#define PHONE

class CPhone : virtual public CDevice

{

protected:

char\* m\_PhoneName;

public:

CPhone();

CPhone(char\* creatorName, char\* phoneName);

CPhone(const CPhone& obj);

virtual ~CPhone();

void ShowProperty();

virtual void ShowFullInformation();

void Call(const char\*);

void Rington(const char\*);

CPhone& operator=(CPhone&);

};

#endif

**CComputer.cpp:**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include "CComputer.h"

CComputer::CComputer() : CDevice()

{

cout << "CComputer Default Constructor" << endl;

m\_ComputerName = new char[20];

strcpy(m\_ComputerName, "No name");

}

CComputer::CComputer(char\* creatorName, char\* computerName) : CDevice(creatorName)

{

cout << "CComputer Constructor" << endl;

m\_ComputerName = new char[strlen(computerName) + 1];

strcpy(m\_ComputerName, computerName);

}

CComputer::CComputer(const CComputer& obj) : CDevice(obj)

{

cout << "CComputer Copy Constructor" << endl;

m\_ComputerName = new char[strlen(obj.m\_ComputerName) + 1];

strcpy(m\_ComputerName, obj.m\_ComputerName);

}

CComputer::~CComputer()

{

cout << "CComputer Destructor" << endl;

delete[] m\_ComputerName;

}

void CComputer::ShowProperty()

{

cout << "Computer name:" << m\_ComputerName << endl;

}

void CComputer::ShowFullInformation()

{

cout << "My name is " << m\_CreatorName << ", my weight is " << m\_Weight << " kg" << endl;

ShowProperty();

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

}

CComputer& CComputer::operator=(CComputer& obj)

{

if (this != &obj)

{

cout << "CComputer operator=" << endl;

delete[] m\_ComputerName;

m\_CreatorName = new char[strlen(obj.m\_ComputerName) + 1];

strcpy(m\_ComputerName, obj.m\_ComputerName);

CDevice::operator=(obj);

}

return \*this;

}

void CComputer::PlayGame(const char\* game)

{

cout << "Playing the game " << game << endl;

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

}

void CComputer::UpdateOS(double version)

{

cout << "OS updated to version " << version << endl;

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

}

**CComputer.h:**

#include "CDevice.h"

#include <cstring>

#ifndef MotoRacer

#define MotoRacer

class CComputer : virtual public CDevice

{

protected:

char\* m\_ComputerName;

public:

CComputer();

CComputer(char\* name, char\* computerName);

CComputer(const CComputer& obj);

virtual ~CComputer();

void ShowProperty();

virtual void ShowFullInformation();

void PlayGame(const char\*);

void UpdateOS(double version);

CComputer& operator=(CComputer&);

};

#endif

**CDevice.cpp:**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include "CDevice.h"

CDevice::CDevice() : m\_CreatorName(new char[20]), m\_Weight(0)

{

cout << "CDevice Default Constructor" << endl;

strcpy(m\_CreatorName, "No name");

}

CDevice::CDevice(char\* creatorName)

{

cout << "CDevice Constructor" << endl;

m\_CreatorName = new char[strlen(creatorName) + 1];

strcpy(m\_CreatorName, creatorName);

}

CDevice::CDevice(const CDevice& obj)

{

cout << "CDevice Copy Constructor" << endl;

m\_CreatorName = new char[strlen(obj.m\_CreatorName) + 1];

strcpy(m\_CreatorName, obj.m\_CreatorName);

m\_Weight = obj.m\_Weight;

}

CDevice::~CDevice()

{

cout << "CDevice Destructor" << endl;

delete[] m\_CreatorName;

}

void CDevice::ShowProperty()

{

cout << "I have no properties" << endl;

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

}

CDevice& CDevice::operator=(const CDevice& obj)

{

if (this != &obj)

{

cout << "CDevice operator=" << endl;

delete[] m\_CreatorName;

m\_CreatorName = new char[strlen(obj.m\_CreatorName) + 1];

strcpy(m\_CreatorName, obj.m\_CreatorName);

}

return \*this;

}

**CDevice.h:**

#pragma once

#ifndef DEVICE

#define DEVICE

#include <stdlib.h>

#include <iostream>

#include <cstring>

using namespace std;

class CDevice

{

protected:

char\* m\_CreatorName;

double m\_Weight;

public:

CDevice();

CDevice(char\* creatorName);

CDevice(const CDevice& obj);

virtual ~CDevice();

inline void SetWeight(double w) { m\_Weight = w; }

double GetWeight() { return m\_Weight; }

void ShowProperty();

virtual void ShowFullInformation() = 0;

CDevice& operator=(const CDevice&);

};

#endif // !DEVICE

**Висновок:** Виконуючи дану лабораторну роботу я ознайомився із множинним спадкуванням класів та поліморфізмом.