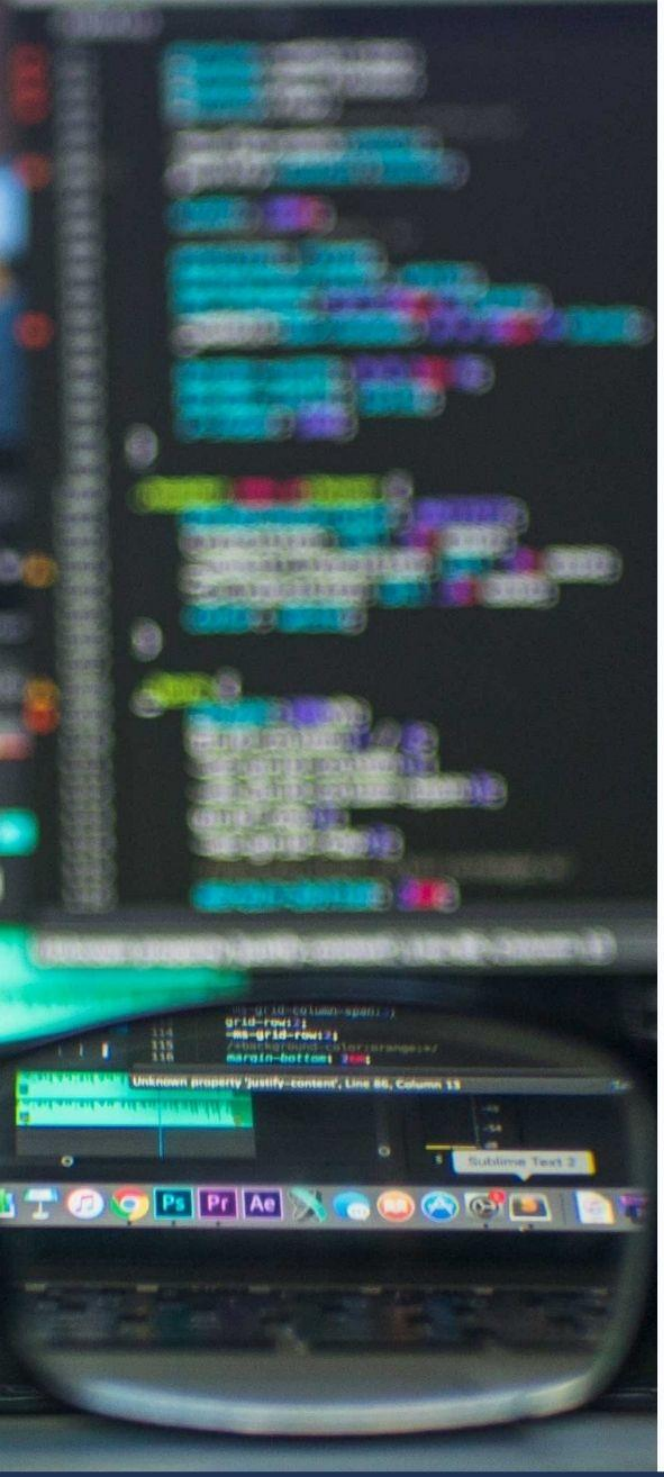


ECOLE NORMALE SUPÉRIEURE DE
L'ENSEIGNEMENT TECHNIQUE MOHAMMEDIA
DÉPARTEMENT MATHÉMATIQUES ET
INFORMATIQUE

24/04/2022

Compte rendu

Des travaux pratique de la chapitre 8

L'heritage multiple

PRÉPARÉE POUR
TAFFAH ACHRAF

ENCADRÉ PAR
PR. KHALIFA MANSOURI

Filière d'ingénieur : Génie du Logiciel et des Systèmes Informatiques

TABLE DES MATIERES

INTRODUCTION

LE C++ FOURNIT LA NOTION D'HERITAGE MULTIPLE, C'EST-A-DIRE LA POSSIBILITE POUR UNE CLASSE D'AVOIR NON PAS UNE, MAIS PLUSIEURS CLASSES DE BASE. CETTE FONCTIONNALITE EST ASSEZ CONTROVERSEE, SOUVENT CONSIDEREE COMME TROP COMPLEXE ET PEU UTILE. BEAUCOUP DE LANGAGES AYANT LEURS ORIGINES DANS LE C++ ONT DECIDE DE N'EN PERMETTRE QU'UNE VERSION LIMITEE : L'HERITAGE MULTIPLE D'INTERFACES.
AYANT DÛ TRAVAILLER AVEC UN DE CES LANGAGES, LE C#, J'AI PU CONSTATER EN PRATIQUE COMBIEN L'HERITAGE MULTIPLE ME MANQUAIT ET EN QUOI SON ABSENCE ENTRAINAIT LA DUPLICATION DE CODE.
C'EST CETTE EXPERIENCE QUE JE SOUHAITE PARTAGER ICI, AINSI QUE QUELQUES INFORMATIONS SUR LA MANIERE DONT ON PEUT METTRE EN ŒUVRE L'HERITAGE MULTIPLE EN C++.

PARTIE PRATIQUE

1. Exemples de cour :

REMARQUE :

Dans le cas de l'héritage simple, le constructeur devant pouvoir retransmettre des informations de la classe de base. Il en va de même ici, avec cette différence qu'il y a deux classes de base. L'en-tête du constructeur se présentera ainsi.

Déclaration et affichage de pointcool

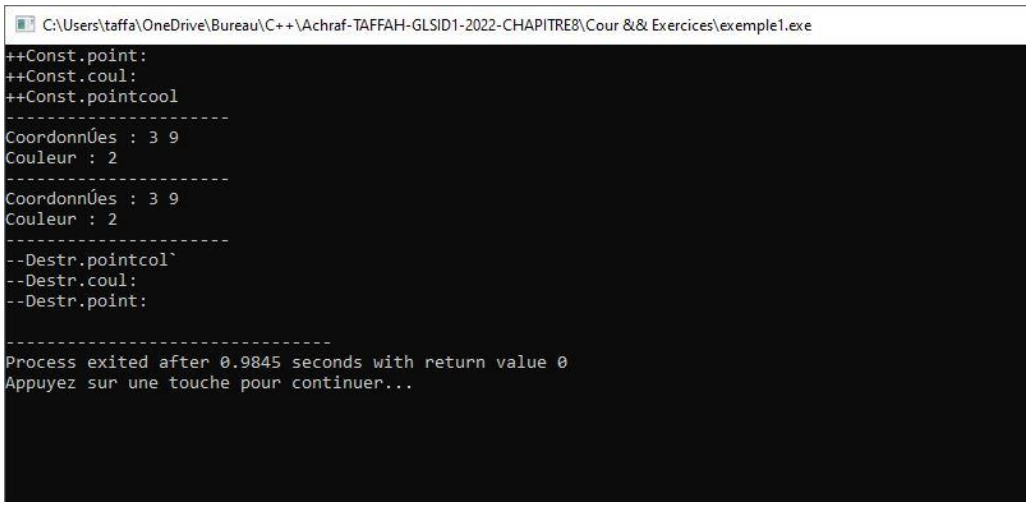
EXEMPLE :

[*] exemple1.cpp

```
#include<iostream>
#include<conio.h>
using namespace std;
class point{
    int x,y;
    public:
        point(int abs,int ord){
            cout<<"++Const.point:" << "\n";
            x=abs;
            y=ord;
        }
        ~point(){
            cout<<"--Destr.point:" << "\n";
        }
        void affiche(){
            cout<<"Coordonnées : "<<x<< " "<< y << "\n";
        }
};
class coul{
    short couleur;
    public:
        coul(short cl){
            cout<<"++Const.coul:"<< "\n";
            couleur=cl;
        }
        ~coul(){
            cout<<"--Destr.coul:"<<"\n";
        }
        void affiche(){
            cout<<"Couleur : "<< couleur << "\n";
        }
};
```

```
pointcol ::pointcol(int abs,int ord,short cl) :point(abs,ord),coul(cl){
    cout<<"++Const.pointcool"<<"\n";
}
int main(){
    pointcol p(3,9,2);
    cout << "----------------------\n";
    p.affiche();
    cout << "----------------------\n";
    p.pointcol::affiche();
    cout << "----------------------\n";
    return 0;
}
```

RESULTAT :



CONCLUSION

DANS LES CHAPITRES PRECEDENTS, J'AI ETE CONVAINCU DE L'INTERET DE L'HERITAGE MULTIPLES. CE CHAPITRE REVIENT MAINTENANT SUR CERTAINS POINTS DE DETAIL A MAITRISER POUR BIEN UTILISER CE CONCEPT. BIEN QUE LA MISE EN ŒUVRE PUISSE ETRE ASSEZ COMPLIQUEE, L'UTILISATION DE CETTE FONCTIONNALITE N'EST, DANS LA PLUPART DES CAS, PAS TRES COMPLIQUEE.