

SNOUSSI ANIS
MED LAMINE BARGHOUDA

TP3 C++

GL 2-2

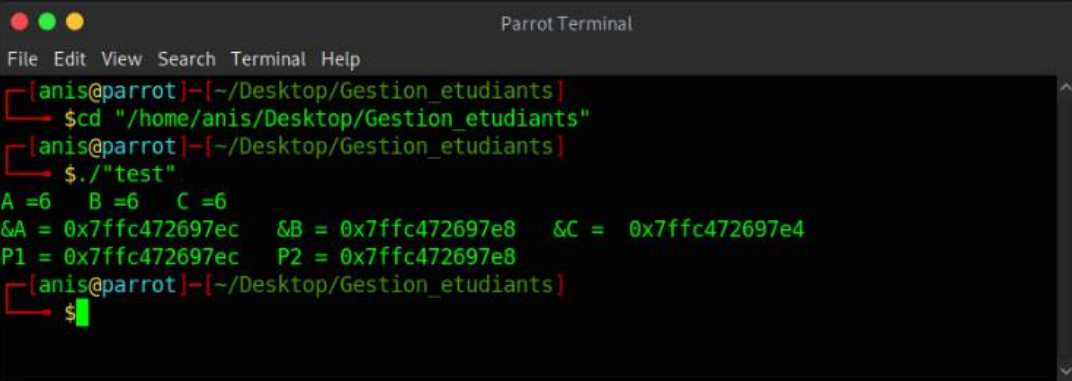
Compte rendue

Question 1

```
#include <iostream>

main() {

    int A =1, B=2, C=3;
    int *P1, *P2;
    P1=&A;
    P2=&C;
    *P1=(*P2)++;
    P1=P2;
    P2=&B;
    *P1-=*P2;
    ++*P2;
    *P1*=*P2;
    A=++*P2**P1;
    P1=&A;
    *P2=*P1/=*P2;
    std::cout << "A =" << A << " " ;
    std::cout << "B =" << B << " " ;
    std::cout << "C =" << C << '\n' ;
    std::cout << "&A = " << &A << "    &B = " << &B << "    &C = " << &C << "\n" ;
    std::cout << "P1 = " << P1 << "    P2 = " << P2 << "\n" ;
}
```



```
Parrot Terminal
File Edit View Search Terminal Help
[anis@parrot]~/Desktop/Gestion_etudiants
$ cd "/home/anis/Desktop/Gestion_etudiants"
[anis@parrot]~/Desktop/Gestion_etudiants
$ ./"test"
A =6   B =6   C =6
&A = 0x7ffc472697ec   &B = 0x7ffc472697e8   &C = 0x7ffc472697e4
P1 = 0x7ffc472697ec   P2 = 0x7ffc472697e8
[anis@parrot]~/Desktop/Gestion_etudiants
$
```

Après l'exécution on obtient:

A = B = C = 6

P1 = &A

P2 = &B

Question 2

Version 1: en utilisant (point)++;

```
#include <iostream>


class tableau{
private:

public:
    void afficheA(int A[]){
        for(int i=0 ;i<10 ; i++){
            std::cout << "A[" << i << "] = " << A[i] << "\n" ;
        }
    };
    void rempliA(int A[]){
        for(int i=0 ;i<10; i++){
            std::cout << "donner A[" << i << "] : " ;
            std::cin >> A[i] ;
        }
    };
};

main() {
    int A[10];
    tableau t;
    t.rempliA(A);
    int & ref=A[5];
    int * point;
    point=A+7;
    (point)++;
    t.afficheA(A);
}
```

```
donner A[0] : 12
donner A[1] : 14
donner A[2] : 115
donner A[3] : 121
donner A[4] : 2
donner A[5] : 33
donner A[6] : 2
donner A[7] : 51
donner A[8] : 44
donner A[9] : 5
A[0] = 12
A[1] = 14
A[2] = 115
A[3] = 121
A[4] = 2
A[5] = 33
A[6] = 2
A[7] = 51
A[8] = 44
A[9] = 5
```

A[7] ne change
pas



Version2: en utilisant (*point)++ ;

```
#include <iostream>

class tableau{
private:

public:
    void afficheA(int A[]){
        for(int i=0 ; i<10 ; i++){
            std::cout << "A[" << i << "] = " << A[i] << "\n" ;
        }
    };
    void rempliA(int A[]){
        for(int i=0 ; i<10; i++){
            std::cout << "donner A[" << i << "] : " ;
            std::cin >> A[i] ;
        }
    };
};

main() {
    int A[10];
    tableau t;
    t.rempliA(A);
    int & ref=A[5];
    int * point;
    point=A+7;
    (*point)++ ;
    t.afficheA(A);
}
```

```
donner A[0] : 12
donner A[1] : 114
donner A[2] : 152
donner A[3] : 2
donner A[4] : 1
donner A[5] : 44
donner A[6] : 12
donner A[7] : 5
donner A[8] : 33
donner A[9] : 2
A[0] = 12
A[1] = 114
A[2] = 152
A[3] = 2
A[4] = 1
A[5] = 44
A[6] = 12
A[7] = 6
A[8] = 33
A[9] = 2
```

A[7] incrémente



Après l'exécution on constate que :

- **L'incrémentation de la valeur de la référence entraine l'incrémentation de la valeur correspondant dans le tableau.**
- **Lors de l'incrémentation de pointeur, le pointeur va pointer vers la case suivante du tableau.**
- **L'incrémentation de la valeur de la référence entraine l'incrémentation de la valeur correspondant dans le tableau.**

Question 3

```
Int main() {  
    int n=50 ;  
    myClass s ;  
    s.meth(n) ;  
    cout<<"n^2"<<n ;  
    return(0) ;  
}
```

Version1

```
Int main(){  
    int n =50 ;  
    myClass s ;  
    s.square(&n) ;  
    cout<<"n^2="<<*n ;  
    return(0) ;  
}
```

Version2

```
Int main() {  
    int n =50 ;  
    myClass s ;  
    s.square(n) ;  
    cout<<"n^2="<<n ;  
    return(0) ;  
}
```

Version3

- La valeur du nombre passé en paramètre ne change pas lors du premier passage par valeur.
- La valeur du nombre passé en paramètre ne change que dans un passage par adresse ou par référence.



Question3: Gestion des étudiants

MAIN.CPP

```
#include <iostream>
#include <cstdlib>
#include "Etudiant.h"
#include "Matiere.h"
#include "Filiere.h"
using namespace std;

int main () {
    // On a appliqué la consigne en testant avec 4 étudiants, 3 matières et 2 filières
    Filiere GL, IIA;

    cout << "\n ** Initialisation de la filiere GL ** \n";
    GL.setIdFil();
    GL.setTabEtudiant();

    cout << "\n ** Initialisation de la filiere IIA ** \n";
    IIA.getIdFil();
    IIA.setTabEtudiant();

    cout << "Affichage de La filiere GL" << endl;
    system ("pause");
    GL.Afficher();
    system ("pause");

    cout << "Affichage de La filiere IIA" << endl;
    system ("pause");
    IIA.Afficher();
    system ("pause");

    return 0;
};
```


ETUDIANT.H

```
#ifndef ETUDIANT_H
#define ETUDIANT_H
#include "Matiere.h"

class Etudiant{
private:
    int NumCarte;
    int Telephone;
    double Moyenne;
    const int NbMatiere = 3;
    Matiere Matieres[3];

public:
    // Constructeur
    Etudiant();
    // Destructeur
    ~Etudiant();

    void Afficher();
    double CalculMoyenne();
    bool Reussi();
    // getters
    int getNumCarte();
    int getTelephone();
    double getMoyenne();
    Matiere* getMatiere();
    int getNbMatiere();
    // setters
    void setNumCarte();
    void setTelephone();
    void setMatiere();
    // "setMoyenne" est automatique avec setMatiere car la Moyenne dépend de
    notes de Matieres
    void setAll();
};

#endif /* ETUDIANT_H */
```

FILIERE.H

```
#ifndef FILIERE_H
#define FILIERE_H
#include "Etudiant.h"

class Filiere{
private:
    int IdFil;
    const int NbEtudiant=4;
    Etudiant TabEtudiant[4];
public:
    // Constructeur
    Filiere();
    // Destructeur
    ~Filiere();

    void Afficher();
    // setters
    void setIdFil();
    void setTabEtudiant();
    // getters
    int getIdFil();
    Etudiant* getTabEtudiant();
    int getNbEtudiant();
};

#endif /* FILIERE_H */
```

MATIERE.H

```
#ifndef MATIERE_H
#define MATIERE_H

class Matiere{
private:
    char Intitule[20];
    double Coefficient;
    double Note;
public:
    // Constructeur
    Matiere();
    // Destructeur
    ~Matiere();
    // getters
    double getCoefficient();
    double getNote();
    char* getIntitule();
    // setters
    void setCoefficient();
    void setNote();
    void setIntitule();
    void setAll();
};

#endif /* MATIERE_H */
```

ETUDIANT.CPP

```
#include <iostream>
#include "Etudiant.h"
#include "Matiere.h"
using namespace std;

Etudiant::Etudiant(){
    NumCarte=0;
    Telephone=0;
    Moyenne=-1;
    for (int i=0;i<NbMatiere;i++)
        Matieres[i];
}

Etudiant::~~Etudiant(){
    cout << "Object Etudiant Deleted !" << endl;
}

void Etudiant::Afficher(){
    cout << "Num Carte : " << NumCarte << endl;
    cout << "Telephone : " << Telephone << endl;
    cout << "Moyenne : " << Moyenne << endl;
    cout << "Liste des Matieres : " << endl;
    for(int i=0;i<2;i++){
        cout << Matieres[i].getIntitule() << "\t";
        cout << "Note: " << Matieres[i].getNote() << endl;
    }
    if(Reussi())
        cout << "Etudiant " << NumCarte << " a réussi !" << endl;
    else
        cout << "Etudiant " << NumCarte << " n'a pas réussi !" << endl;
}

double Etudiant::CalculMoyenne(){
    double SN=0,SC=0;
    for(int i=0;i<2;i++){
        SN+= Matieres[i].getNote()*Matieres[i].getCoefficient();
        SC+=Matieres[i].getCoefficient();
    }
    Moyenne=SN/SC;
    return(Moyenne);
}
```

```

bool Etudiant::Reussi(){
    if(Moyenne>=10)
        return(true);
    return(false);
}
int Etudiant::getNumCarte(){
    return NumCarte;
}
int Etudiant::getTelephone(){
    return Telephone;
}
double Etudiant::getMoyenne(){
    return Moyenne;
}
Matiere* Etudiant::getMatiere(){
    return(Matiere);
}
int Etudiant::getNumMatiere() {
    return NbMatiere;
}
void Etudiant::setNumCarte(){
    cout << "Saisir Num Carte" << endl;
    cin >> NumCarte;
}
void Etudiant::setTelephone(){
    cout << "Saisir Telephone" << endl;
    cin >> Telephone;
}
void Etudiant::setMatiere(){
    cout << "\nSetMatiere activated ! " << endl;
    for(int i=0;i<NbMatiere;i++)
        Matieres[i].setAll();
    // On doit mettre à jour la Moyenne en fonction de table de Matieres
    Moyenne = CalculMoyenne();
}
void Etudiant::setAll(){
    setNumCarte();
    setTelephone();
    setMatiere();
}

```

FILIERE.CPP

```
#include <iostream>
#include "Etudiant.h"
#include "Filiere.h"
using namespace std;

Filiere::Filiere(){
    IdFil=0;
    for(int i=0;i<NbEtudiant;i++)
        TabEtudiant[i];
}
Filiere::~~Filiere(){
    cout << "Object Filiere Deleted !" << endl;
}
void Filiere::Afficher(){
    cout << "IdFiliere : " << IdFil << endl;
    cout << "Liste d'etudiants : " << endl;
    for(int i=0;i<NbEtudiant;i++){
        cout << "\n** Etudiant NÂ° " << i+1 << " **" << endl;
        TabEtudiant[i].Afficher();
    }
    cout << endl;
}
void Filiere::setIdFil() {
    cout << "Saisir l'ID de Filiere" << endl;
    cin >> IdFil;
}
int Filiere::getIdFil() {
    return(IdFil);
}
Etudiant* Filiere::getTabEtudiant() {
    return (TabEtudiant);
}
void Filiere::setTabEtudiant() {
    cout << "\nSetTabEtudiant activated !" << endl;
    for(int i=0;i<NbEtudiant;i++){
        cout << "\nEtudiant " << i+1 << endl;
        TabEtudiant[i].setAll();
    }
};
int Filiere::getNbEtudiant() {
    return NbEtudiant;
}
```

MATIERE.CPP

```
#include <iostream>
#include "Matiere.h"
#include <cstring>
using namespace std;

Matiere::Matiere(){
    Coefficient=0;
    Note=-1;
}
Matiere::~~Matiere(){
    cout << "Object Matiere deleted !" << endl;
}
double Matiere::getCoefficient(){
    return Coefficient;
}
double Matiere::getNote(){
    return Note;
}
char* Matiere::getIntitule(){
    return Intitule;
}
void Matiere::setCoefficient(){
    cout << "Saisir Coefficient" << endl;
    cin >> Coefficient;
}
void Matiere::setNote(){
    cout << "Saisir Note" << endl;
    cin >> Note;
}
void Matiere::setIntitule(){
    cout << "Saisir Intitule" << endl;
    cin >> Intitule;
}
void Matiere::setAll() {
    setIntitule();
    setCoefficient();
    setNote();
}
```

SRCHARGE DU CONSTRUCTEUR DE MATIERE

```
Matiere::Matiere(char I[],double C,double N){  
    strcpy(Intitule,I);  
    Coefficient= C;  
    Note= N;  
}
```

LA METHODE : APPARTENANCE (paramètre)

Le méthode intitulée Boolean appartenance(paramètre) doit être définie au sein de la class filière pour vérifier directement l'appartenance de l'étudiant dans le tableau d'étudiants correspondant.

MODES DE PASSAGE D'UN OBJET

1. Boolean appartenance (Etudiant e) : passage par valeur

Les variables qui stockent des objets possèdent en fait la référence de l'objet. Le fait de passer un objet à une méthode équivaut à passer la référence de l'objet en paramètre.

(La définition simultanée des deux fonctions 1 et 3 peut provoquer une ambiguïté lors de l'exécution)

2. Boolean appartenance (Etudiant *e) : passage par adresse

L'adresse de e est copiée dans une variable de type Etudiant* (un pointeur sur un objet Etudiant) et puis utilisé pour accéder à cette variable et la modifier.

3. Boolean appartenance (Etudiant &e) : passage par référence

Le passage par référence fonctionnellement comporte comme un passage par pointeur mais syntaxiquement c'est plus facile (comme dans un passage par valeur)