Département de Mathématiques **Année Universitaire 2017-2018**

Faculté des Sciences

Université de Tlemcen

2ème année licence Mathématiques (Module : Outils Mathématiques)

Durée : 01h30mn

Epreuve Finale

Exercice N 01 : (12.00 pts)

Construire une classe qu'on appellera **Personne,** ou on trouvera :

- Trois données membres qui sont :

mon_Age (entier), mon_Nom (type énumération notée Nom), mon_ Prenom (Type énumération notée Prenom) .

- Ainsi que les méthodes suivantes :

1) Constructeurs par défaut, à 1 paramètre (mon_Age) , 2 Constructeurs à 2 paramètres ((mon_Age, mon_Nom), (mon_Nom,mon_Prenom)) , à 3 Paramètres ,Constructeur de copies et un Destructeur.

2) les Fonctions d'**Accès**

3) **Ajout_Note, Enleve_Note, Affiche _Info , Note_Min, Note_Max** qui vont affichées respectivement les commentaires suivants :

« Note_Personne«, « Enleve_Note_Personne« , « Info_Personne«

« Note_Min_Personne« et « Note_Max_Personne«

On construira aussi une autre classe qu'on appellera **Eleve** qui hérite (dérive) de la classe de base **Personne**, ou on trouvera :

- Une seule donnée membre notée **ma_Note (type entier)**

Ainsi que les Fonctions Membres suivantes :

1) Constructeur par défaut, Constructeur à un paramètre (ma_Note) ,Constructeurs à 4 paramètres (mon_Age, mon_Nom , mon_Prenom,ma_Note) .
2) Constructeur de Copies et un Destructeur.
3) Fonction d'Acces.
4) **Ajout_Note, Enleve_Note, Affiche _Info , Note_Min, Note_Max** qui vont affichées respectivements les commentaires suivants :

« Note_eleve«, « Enleve_Note_Eleve« , « Info_Eleve«

« Note_Min_Eleve« et « Note_Max_Eleve«.

5) **Calcul_Moyenne,Calcul_Nbre_Note** qui afficherons :

« Moyenne_Eleve est :«, « Nbre_Note_Eleve est : « .

Ecrire le programme Principal qui nous permettra :

- D'utiliser tous les constructeurs définis ci-dessus (même les constructeurs de copies), avec des instances (objets) des deux classes (une instance pour chaque constructeur).
- Faire appel aux méthodes (des deux classes) :
  Ajout_Note , Enleve_Note , Affiche _Info , Note_Min , Note_Max.

Avec une instance (objet) de la classe de base (**Personne**) de 3 manières différentes.

Exercice N 02 : (08.00 pts)

Ecrire un programme en C++ qui nous permettra de calculer la **dérivée d'ordre 1** d'une fonction **f** au point **x0** de **3 manières** (**f** : fonction à valeurs réelles)**.**

Les données sont :

- **La fonction f (on prendra f(x) = x+1) (placer la dans une function) , x0 et h .**

  N.B :

  - Utiliser **3 functions** pour chaque formule qu'on appellera **(Formule_1, Formule_2 et Formule_3) ,** chaque fonction est constituée de **2 arguments x0 et h.**

  - Utiliser le mot clé **typedef (** pour construire une fonction **modèle** qui **ressemblera aux 3** functions définies ci-dessus **),** ainsi que l'instruction **switch** pour que l'utilisateur ait le **choix** entre **l'usage** des Trois Formules**.**

Bon Courage

```cpp
#include <iostream>
#include<math.h>
#include<conio.h>
using namespace std;

enum Nom { Nom_1, Nom_2, Nom_3, Nom_4, Nom_5, Nom_6 };
enum Prenom { Prenom_1, Prenom_2, Prenom_3, Prenom_4, Prenom_5, Prenom_6 };

class Personne
{
public:
    // constructors
    Personne();
    Personne(int );
    Personne(int, Nom );
    Personne(Nom,Prenom );
    Personne(int, Nom , Prenom );

    // Constructeur de copies
    Personne( const Personne & );
     // Destructeur
    ~ Personne();

    // Fonctions d Acces
    int get_Age() const ;//{ return mon_Age; }
    Nom get_Nom() const;// { return mon_Nom; }
    Prenom get_Prenom() const;// { return mon_Prenom; }

    // Autres Methodes
    virtual void Ajout_Note();  // « Note_Personne«,
    virtual void Enleve_Note( ) ;  //  « Enleve_Note_Personne« ,
    virtual void Affiche_Info( ) ;  // « Info_Personne«
    virtual void  Note_Min ( );  // « Note_Min_Personne«
    virtual void  Note_Max(); //  et « Note_Max_Personne«



protected:
    int mon_Age;
    Nom mon_Nom;
    Prenom mon_Prenom;
};


class Eleve : public Personne
{
public:

     // constructeurs
    Eleve();
    Eleve(int );
    Eleve(int, Nom , Prenom ,int);

    // Constructeur de copies
    Eleve( const Eleve & );
     // Destructeur
    ~Eleve();

    // Fonctions d Acces
    int get_ma_Note() const ;//{ return ma_Note; }


    // Autres Methodes
    void Ajout_Note();  // « Note_Eleve«,
    void Enleve_Note( ) ;  //  « Enleve_Note_Eleve« ,
    void Affiche_Info( ) ;  // « Info_Eleve«
    void  Note_Min ( );  // « Note_Min_Eleve«
    void  Note_Max(); //  et « Note_Max_Eleve«


private:
    int ma_Note;
};
```

```
// Partie Implementation  de la classe Personne

 // mon_Age; mon_Nom;mon_Prenom
Personne::Personne():mon_Age(10),mon_Nom(Nom_1),mon_Prenom(Prenom_1)
{
    cout << " Personne constructeur par defaut ...\n"<<endl;
}

Personne::Personne(int age):mon_Age(age),mon_Nom(Nom_1),mon_Prenom(Prenom_2)
{
    cout << "Personne (int) constructeur ...\n"<<endl;
}

  Personne::Personne(int age,Nom nomm):mon_Age(age),mon_Nom(nomm),mon_Prenom(Prenom_3)
{
    cout << "Personne (int,Nom) constructeur ...\n"<<endl;
}

Personne::Personne(Nom nomm,Prenom prenomm):mon_Age(10),mon_Nom(nomm),mon_Prenom(prenomm)
{
    cout << "Personne (Nom,Prenom) constructeur ...\n"<<endl;
}


Personne::Personne(int age,Nom nomm, Prenom
prenomm):mon_Age(age),mon_Nom(nomm),mon_Prenom(prenomm)
{
    cout << "Personne (int,Nom,Prenom) constructeur ...\n"<<endl;
}

Personne::~Personne()
{
    cout << "Personne destructeur ...\n"<<endl;
}

  Personne::Personne( const Personne & rhs ) :
  mon_Age(rhs.mon_Age),mon_Nom(rhs.mon_Nom),mon_Prenom(rhs.mon_Prenom)
    {

        cout<<"Constructeur de copies de la Classe Personne ...:\n"<<endl;
    }

int Personne :: get_Age() const
            {
             return mon_Age;
             }

Nom Personne::get_Nom() const
        {
            return mon_Nom;
         }
Prenom Personne:: get_Prenom() const
         {
            return mon_Prenom;
             }

void Personne:: Ajout_Note(){
                                cout<<"Note_Personne ...\n"<<endl;
                              }
void Personne:: Enleve_Note( ) {
                                    cout<<" Enleve_Note_Personne ...\n"<<endl;
                                 }
void Personne:: Affiche_Info( ) {
                                     cout<<" Info_Personne ...\n"<<endl;
                                  }
void  Personne:: Note_Min ( )    {
                                     cout<<" Note_Min_Personne ...\n"<<endl;
                                 }

 void  Personne:: Note_Max()    {
                                cout<<" Note_Max_Personne ...\n"<<endl;
                                }
```

```
// Partie Implementation de la classe Eleve

Eleve::Eleve():Personne(),ma_Note(14)
  {
    cout << "Eleve defaut  constructeur ...\n";
   }
Eleve::Eleve(int notee):Personne(),ma_Note(notee)
  {
    cout << "Eleve  constructeur a 1 parametre...\n";
   }

Eleve::Eleve(int age,Nom nomm,Prenom prenomm,int
notee):Personne(age,nomm,prenomm),ma_Note(notee)
  {
    cout << "Eleve constructeur a 4 parametres...\n";
   }

Eleve::Eleve( const Eleve & rhs ) :Personne(rhs),ma_Note(rhs.ma_Note)
    {

        cout<<"Constructeur de copies de la Classe Eleve ...:\n"<<endl;
    }

Eleve::~Eleve()
{
    cout << "Eleve destructeur ...\n";
}

int Eleve::get_ma_Note() const{
                                  return ma_Note;
                               }

void Eleve:: Ajout_Note(){
                                   cout<<"Note_Eleve ...\n"<<endl;
                                }
void Eleve:: Enleve_Note( ) {
                                     cout<<" Enleve_Note_Eleve ...\n"<<endl;
                                   }
void Eleve:: Affiche_Info( ) {
                                      cout<<" Info_Eleve ...\n"<<endl;
                                    }
void  Eleve:: Note_Min ( )   {
                                      cout<<" Note_Min_Eleve ...\n"<<endl;
                                   }

 void  Eleve:: Note_Max()   {
                                  cout<<" Note_Max_Eleve ...\n"<<endl;
                                  }


int main()
{
    // Invoquer les diferents Constructeurs ( Classe Personne et Eleve

    Personne *PPers, Pers_00;
    Personne Pers_0,
    Pers_1(10),Pers_2(10,Nom_1),Pers_3(Nom_2,Prenom_3),Pers_4(10,Nom_4,Prenom_2);
    Eleve Elev_00,Elev_0,Elev_1(5),Elev_2(12,Nom_5,Prenom_1,15);

    // Invoquer les Fonctions d'Acces

       cout<<"Age de Pers_0 est :\n"<<Pers_0.get_Age()<<endl;

        cout<<"Nom de Pers_0 est :\n"<<Pers_0.get_Nom()<<endl;

    cout<<"Prenom  de Pers_0 est :\n"<<Pers_0.get_Prenom()<<endl;

    cout<<"Age de Elev_0 est :\n"<<Elev_0.get_Age()<<endl;
    cout<<"Nom de Elev_0 est :\n"<<Elev_0.get_Nom()<<endl;
    cout<<"Prenom de Elev_0 est :\n"<<Elev_0.get_Prenom()<<endl;
    cout<<"Note de Elev_0 est :\n"<<Elev_0.get_ma_Note()<<endl;
```

```
    // Invoques les autres Méthodes
      // Affiche _Info , Note_Min, Note_Max

         Pers_1.Ajout_Note();
         Elev_1.Personne::Ajout_Note();
         PPers=& Pers_00;
        // Affiche_Info
         PPers->Affiche_Info();
         PPers=&Elev_00;
          (*PPers).Enleve_Note();
          PPers=new Personne();
          PPers->Note_Min();
          delete PPers;
          PPers=new Eleve(10);
          (*PPers).Note_Max();


    getch();
    return 0;
}
```

```
#include <iostream>
float f(float);

void printresult(float);

typedef float (*VPF_1)(float );
typedef float  (*VPF)(float , float );
float Formule_1(float, float);
float Formule_2(float, float);
float Formule_3(float, float);
//void getVals(VPF_1,int&, int&);
using namespace std;
int main()
{
   // void (*pFunc)(int&, int&);
   VPF pFunc; VPF_1 pfunc_1;
    bool fQuit = false;

    float x0,h;
    cout<<"lire x0 ...\n"<<endl;
    cin>>x0;
    cout<<"lire h ...\n"<<endl;
    cin>>h;
    int choix;
    while (fQuit == false)
    {
        cout << "(0) Quit (1) Formule_1...,(2) Formule_2 (3) Formule_3   \n "<<endl;
        cin >> choix;
        switch (choix)
        {
           case 1:{

                     pFunc = Formule_1;
                     break;
                  }
          case 2: {
                     pFunc = Formule_2;
                     break;
                  }
          case 3: {
                     pFunc = Formule_3;
                     break;
                  }
        default : {

                     fQuit = true;
                     break;
                  }
        }

        if (fQuit)
            break;

        ;

        //Pfunc_1=f;
        float result= pFunc(x0, h);
         printresult(result);
    }
    cout<<"Terminer... \n"<<endl;
    return 0;
}

void printresult(float x)
{
    cout << " la derivee premiere de f est.... =\n " << x  << "\n";
}

float  Formule_1(float X, float Y)
{
    cout<<"je suis dans Formule_1\n"<<endl;
     float x1=X+Y;
    float yy=f(x1);
```

```
      float yy1=f(X);
      float yy2=yy-yy1;
      yy2/=Y;
    //cout<<"yy2=\n"<<yy2<<endl;
    return yy2;
}

float  Formule_2(float X, float Y)
{
    cout<<"je suis dans Formule_2\n"<<endl;
     float x1=X-Y;
    float yy=f(x1);
      float yy1=f(X);
      float yy2=yy1-yy;
      yy2/=Y;
    //cout<<"yy2=\n"<<yy2<<endl;
    return yy2;
}
float  Formule_3(float X, float Y)
{
    cout<<"je suis dans Formule_3\n"<<endl;
     float yy3=2*Y;
     float x1=X+Y;
    float yy1=f(x1);
      float x2=X-Y;
      float yy2=f(x2);
      float yy4=yy1-yy2;
      yy4/=yy3;
    //cout<<"yy2=\n"<<yy2<<endl;
    return yy4;
}


float f( float x1){
    float y=x1+1.;
    return y;
}
```