Fiche de TP2 : Notion de classe en C++

Pour gagner du temps dans la rédaction des bouts de programmes fournis dans les exercices, nous rappelons que les fichiers sources de ces programmes sont téléchargeables à l'adresse habituelle.
On utilisera un fichier **Makefile** *pour la compilation.*

Thème - 1 *Énumération - Tableaux*

Exercice-1 **:** On considère le programme suivant suivant

fichier: enumeration.cpp

```
#include <iostream>

using namespace std;

enum E{a,b,c=0,d,e};

int main(int argc, char** argv)
{
  cout<< " a= " << a <<endl;
  cout<< " b= " << b <<endl;
  cout<< " c= " << c <<endl;
  cout<< " d= " << d <<endl;
  cout<< " e= " << e <<endl;

return 0;
}
```

Q-1-1 **:** Qu'affiche ce programme ?

Q-1-2 **:** Modifier le programme en mettant cette fois

fichier: enumeration.cpp

```
enum E{a,b,c,d,e};
```

Qu'observez-vous ?

Exercice-2 **:**

Q-2-1 **:** On considère le programme suivant

fichier: tableau_ivalue.cpp

```
#include <iostream>
```

```cpp
using namespace std;

int main(int argc, char** argv)
{
  int tab[3]={1,2,3};
  *(tab + 1) = -1;
  for(int i=0; i<3; i++){tab++;}

return 0;
}
```

Ce programme compile-t-il ? justifier votre réponse.

Q-2-2 : On considère le programme suivant

fichier: tableau_argument.cpp

```cpp
#include <iostream>

using namespace std;

//FONCTION D'AFFICHAGE
void afficheTableau(int t[],int size)
{
  for(int i=0; i< size; i++) cout<<t[i]<<",\t";
  cout<<endl;
}

//FONCTION D'INCRÉMENTATION DU CONTENU
void incrementeTableau(int t[]){
   int size = sizeof(t)/sizeof(*t);
  for(int i=0; i< size; i++) { t[i]++;}
}

// EVALUATION
int main(int argc, char** argv)
{
  int tab[3]={1,2,3};

  cout<<" tab avant : \n";
  afficheTableau(tab,3);

  incrementeTableau(tab);

  cout<<" t apres : \n";
  afficheTableau(tab,3);

return 0;
}
```

1. La fonction **incrementeTableau** réalise-t-elle le but visé ?
2. Que se passerait-il si on remplaçait dans la fonction **main**, la fonction **incrementeTableau** par son contenu ?
3. Comment peut-on corriger la fonction **incrementeTableau** pour qu'elle réalise le but visé ?

Q-2-3 : On considère le programme suivant, dire s'il compile. Justifier la réponse.

fichier: tableau_const.cpp

```cpp
#include <iostream>
using namespace std;
int main(int argc, char** argv)
{
  int tab[3]={1,2,3};
  const int *ptab1 = tab;
  const int *  const ptab2 = tab;
  const int ctab[3] = {4,5,6};
  *(ptab1 + 1) = -1;
  *(ptab2 + 1) = -1;
  ctab[1] = -1;
  for(int i=0; i< 3; i++){ptab1++ ; ptab2++;}
  return 0;
}
```

Thème - 2 *Notion de classes*

Exercice-1 **:**

- Créer une classe qui contient un **`int`** (un entier).
- Implémenter l' **`operator<<`** pour l'afficher.
- Tester la solution.

Exercice-2 **:**

- Créer une classe contenant un **`int`**.
- Ajouter une méthode pour l'afficher.
- Implémenter les opérateurs **++** préfixe et suffixe incrémentant cet entier.
- Prouver que la solution fonctionne.

Exercice-3 **:** On considère le programme suivant :

fichier: methodeprivees.cpp

```cpp
#include <iostream>
//=========================================================
// CLASS
//=========================================================
class ObjC
{
 int* ptr;
 public:
   ObjC(int& a):ptr(&a){;}
   int& refData()const {return *ptr;}
   int*& refPtrData(){   return ptr;}
   int* ptrData()const{return ptr;}
};
//=========================================================
// EVALUATION
//=========================================================
using std::cout;
int main()
```

```
{
  int a =2;
  int b =3;
  ObjC o(a);
  cout<<" avant a "<< a<<"\n";
  o.refData() = b;                // Expliquer pourquoi ceci marche
  cout<<" apres a "<< a <<"\n";
  o.refPtrData()= &b;             // Expliquer pourquoi ceci marche
  cout<<" apres a "<< o.refData() <<"\n";

  const ObjC oo(b);
  //oo = o;                        //Expliquer pourquoi ceci ne marchera ↩
     pas
  cout<<" avant b "<< b<<"\n";
  * oo.ptrData() = 2;             //Expliquer pourquoi ceci marche
  // oo.ptrData() = &b;            //Expliquer pourquoi ceci ne marchera ↩
     pas
  cout<<" apres b "<< b <<"\n";
   return 0;
}
```

Répondre aux commentaires et remarques données dans le code.

Thème - 3

Exercice-1 : On donne le programme suivant

fichier: methodeprivees.cpp

```
#include <iostream>
using namespace std;
//-------------------------------------------------------
class Foo{
 private:
  Foo(){}
  ~Foo(){}
};
//-------------------------------------------------------
int main(int argc, char** argv)
{
  Foo C;
return 0;
}
```

- Expliquer pourquoi ce programme ne compile pas.
- Donner deux façons de créer un objet de la classe **`Foo`**, sans changer de modificateur.
  Une approche consistera à ajouter dans la classe **`Foo`** deux méthodes **`statiques`** créant et détruisant les objets de type **`Foo*`** ; une autre approche consistera à utiliser une **`classe amie`**, pour le faire.

Exercice-2 : On considère le programme suivant :

fichier: temporaryobjects.cpp

```cpp
#include <iostream>
using namespace std;
struct C{};
C f(){return C();}
void g(C& c){}
int main(int argc, char** argv)
{
  g(f());
return 0;
}
```

- Ce programme compile-t-il ? Justifier.

Exercice-3 : On considère le programme suivant :

fichier: assignation.cpp

```cpp
#include <iostream>
using namespace std;
class C{
  int id;
public:
  C(int i): id(i) {cout<<" C::C(int) : "<<id <<endl;}
  C& operator=(const C& c){
    id =c.id;
    cout<<"C::operator =(const C&) : "<<id<<endl;
  return *this;
  }
};

int main(int argc, char** argv)
{
  C c1(1);
  C c2 = c1;
return 0;
}
```

- Qu'affiche-t-il ? Justifier.
- Il y a une erreur grave. Corriger.

Exercice-4 : On considère le programme suivant :

fichier: objectsplicing.cpp

```cpp
#include <iostream>
using namespace std;
//-------------------------------------------------------------
class C{
public:
  virtual void f()const{cout <<"C::f()" << endl;}
};
//-------------------------------------------------------------
class D: public C{
public:
  virtual void f()const{cout <<"D::f()" << endl;}
};
//-------------------------------------------------------------
void f(const C& c){
  c.f();
  D* d = reinterpret_cast<D *> (&const_cast<C&>(c)); //cast le plus ↩
```

```
      violent; à éviter autant que possible
  (*d).f();
}
//-------------------------------------------------------------
int main(int argc, char** argv)
{
  D d; f(d);
return 0;
}
```

- Qu'affiche-t-il ? Expliquer.

Thème - 4 *Applications : une classe pour la gestion d'éléments de* $\mathbb{R}^2$

Exercice-1 **:** Écrire la classe **`R2`**.

▲ Utiliser un tableau de 2 **`doubles`** pour stocker x et y

▲ Coder les fonctions d'accès à x et y de deux manières :

- créer les fonctions

  `double& x()`
  `double& y()`
  `const double& x() const`
  `const double& y() const`

- ainsi qu'un accès par numéro de composante en surchargeant l'opérateur [].

▲ Programmer les opérations standards (somme, différence, ajout, retranchement, produit scalaire, produit vectoriel, produit par un scalaire, ...).

Démontrer le fonctionnement des fonctions écrites dans le programme principal.