Examen 2016 (POO, L2-S3)

Gaceb

Exercice 1 : Réponse pour questions de cours 7 pts

1.a) différence entre i++ et ++i

i++ : utilise la valeur de i, puis l'incrémente **(0.25 pt)**
++i: incrémente la valeur de i, puis l'utilise **(0.25 pt)**

NB. *Dans la boucle for, la différence n'apparait pas, car cela n'intervient que dans la formule de modification de l'itérateur. En aucun cas la valeur de i n'est utilisée directement, on compare juste sa valeur après modification.*

1.b) différence entre paramètre réel et formel d'une fonction :

A la déclaration de la fonction, on définit une liste de **paramètres formels** entre "(" et ")" séparés par des ",". **(0.25 pt)**

A l'invocation ou l'appel de la fonction, liste d'arguments entre "(" et ")" correspond aux paramètres réels, cela peut être des constantes, des variables ou des expressions, à condition que la concordance des types soit respectée. Chaque valeur réelle est ensuite recopiée dans le paramètre formel correspondant. **(0.25 pt)**

2) int *p, n=5 ; pour récupérer l'adresse de p il faut exécuter l'affectation suivante : **p=&n ;** **(1pt)**

3) Trois modes de transmission de paramètres à une fonction :

- par valeur, par adresse et par référence (0.25pt)

a)Par valeur (0.25pt) : La valeur réelle passée en paramètre est copiée dans la variable formelle. C'est cette variable qui est utilisée pour faire les calculs dans la fonction appelée.

b) Par adresse (0.25pt): Pour modifier le paramètre réel, on passe son adresse plutôt que sa valeur. Paramètre formel est initialisé avec le contenu de l'adresse pointant sur le paramètre réel.

c) Par référence (0.25pt): Au lieu de passer les adresses des variables, il suffit de passer les variables elles-mêmes en utilisant des paramètres sous la forme de références. Le paramètre formel est un alias de l'emplacement mémoire du paramètre réel.

4) Allocation dynamique de la mémoire des tableaux 1d, 2d et 3d :

a) 1D (0.25 pts)

```
int n=50;                          // taille de tableau
int * tab = new int [n];           // allocation dynamique de la mémoire
                                   // utilisation du tablcau tab
delete [] tab;                     // libération de la mémoire
```

b)2D(0.25pts)

```
int L=50;                          // nombre de lignes
int C=20;                          // nombre de colonnes

                                   // allocation dynamique de la mémoire
int ** mat = new int*[L];
for(int i=0; i< L; i++)
 mat[i]= new int [C];
                                   // utilisation de la matrice
for(int i=0; i< L; i++)
  delete [] mat[i];                // libération de la mémoire
delete [] tab;
```

c) 3D(0.5pts)

```
int L=50;                          // nombre de lignes
int C=20;                          // nombre de colonnes
int p=80;                          // profondeur

 // allocation dynamique de la mémoire
int *** mat = new int**[L];
for(int i=0; i< L; i++)
{
  mat[i]= new int* [C];
   for(int j=0; j< C; j++)
    mat[i][j]= new int [P];
}
// Utilisation de la matrice 3d
// désallocation ddc la mémoire
 for(int i=0; i< L; i++)
{
  for(int j=0; j< C; j++)
   delete [] mat[i][j];

   delete [] mat[i];
}
 delete [] mat;                                  // utilisation de la
```

5) Les différents niveaux d'accès en héritage

Dans la classe B : x = non accessible, y = bublic, z= protected **(0.30 pts)**

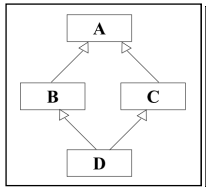
Dans la classe c : x = non accessible, y = protected, z= protected **(0.30pts)**

Dans la classe D : x = non accessible, y = private, z= private **(0.30 pts)**

0.1pts bonus pour réponse juste à 100%

6)Héritage virtuel :

a) (0.5 pts) L'héritage virtuel permet d'éviter la duplication des membres données de la super-classe A dans tous les objets de la classe D qui hérite des deux classes B et C qui héritent de A (voir l'exemple ci-desssous).




```
class A
{ int x, y;
  ....
};

class B : public A {....};
class C : public A {....};

class D : public B, public C
{ ....
};
```

b) **(0.5pts)** La déclaration virtuelle est au niveau des classes B et C (voir l'exemple ci-dessous)

```
// La classe A ne sera introduite qu'une seule fois dans les
// descendants de B aucun effet sur la classe B elle-même

class B : public virtual A {....};

// La classe A ne sera introduite qu'une seule fois dans les
// descendants de C - aucun effet sur la classe C elle-même

class C : public virtual A {....};

// Ici le mot-clé virtual ne doit pas apparaître!!

class D : public B, public C {....};
```

7) Classe abstraite

a) (0.5 pts) Classe abstraite : classe sans instance, destinée uniquement à être dérivée par d'autres classes pour être spécialisée. En C++, une classe est abstraite si elle contient au moins une méthode déclarée virtuelle pure, virtual type methode()=0; // type = void, int…

b) **(0.25 pts) :** Liaison statique→type de l'objet pointé déterminé au moment de la compilation,

NB (ajout) *Polymorphisme statique : le choix de la forme appropriée est déterminé au moment de la compilation à partir de la déclaration de l'objet polymorphe.* ***Mécanisme de surcharge.***

c) **(0.25 pts)** : *Polymorphisme ``dynamique'' : le choix de la forme appropriée se fonde sur le type de l'objet au moment de l'exécution.* ***Mécanisme de liaison ``dynamique'',*** *est un mécanisme distinct (bien qu'étroitement lié) de l'héritage. C'est le fait qu'un nom de fonction membre d'une classe de base peut être associé à une fonction membre d'une classe dérivée.*

Solution Exercice 2 : 9pts

```
// Gaceb, UMBB
#include <iostream>
using namespace std;
class tab
{
 private :
   //----------
     int n;
     int *data;
     int k;

 public :
//Question 1 (0.25 pts)
     tab(const int n=0) {
            this->n=n;
            k=n;
            if(n>0)
            {
              data=new int [n];
              for (int i=0; i<n; i++)
              data[i]=0;
            }
      }
// Question 4 (1.00 pts)
      tab(tab const& t)// Par copie
      {                 n=t.n;
          k=n;
          data=t.data;
            if(n>0)
            {
              data=new int [n];
              for (int i=0; i<n; i++)
              data[i]=t.data[i];
            }
       }
//Question 1(0.25 pts)
      ~tab()      {
        if(n>0)
         delete [] data;
      }
// Qustion 1(0.25 pts)
   void put (int i, int a){
        data[i]=a;
   }
// Question 1 (0.25 pts)
   int isIn ( int a){
     for (int i=0; i<n; i++)
       if(data[i]==a)
        return 1;
      return 0;
   }
// Question 2 (0.50 pts)
   void saisir(){
        cout<<endl<< "saisie de "<< n<<" valeurs";
        for (int i=0; i<n; i++)
        {
        cout<<endl<<"Tab["<< i <<"]="; cin>>data[i];
        }
```

```
    }
// Question 2 (0.50 pts)
   void afficher(){
        cout<<endl<<"Tab[";
        for (int i=0; i<n; i++)
        {
        cout<<data[i] <<"  ";
        }
        cout<<"]";
    }
// Question 3 (1.00 pts)
      void add (tab t)// il y a un problème ici il faut définir un opérateur de recopie
      {
        int max= t.n>n ? t.n : n;
        cout<<endl<<"Somme[";
        if(max==t.n)
        {
          for(int i=0; i<n; i++)
          cout<<t.data[i]+data[i]<<"  ";

          for(int i=n; i<t.n; i++)
          cout<<t.data[i]<<"  ";
      }
      else
        {
          for(int i=0; i<t.n; i++)
          cout<<t.data[i]+data[i]<<"  ";

          for(int i=t.n; i<n; i++)
          cout<<data[i]<<"  ";
      }
       cout<<"]";
      }
// Question 5(1.00 pts)
      tab operator+(tab const& t){
       int max= t.n>n ? t.n : n;
        tab p (max);
        if(max==t.n){
          for(int i=0; i<n; i++) p.data[i]=t.data[i]+data[i];
          for(int i=n; i<t.n; i++) p.data[i]=t.data[i];
        }
      else{

            for(int i=0; i<t.n; i++) p.data[i]=t.data[i]+data[i];
            for(int i=t.n; i<n; i++) p.data[i]=data[i];
      }

       return p;
      }
// Question 6(1.00 pts)
      int &operator[](int i)
      {
        return data[i];
      }




// Question 7(1.00 pts)
void trier() // tri ordre croissant
{
  int tmp=0;
  for(int i = 0; i < n; i++)//On veut remplir la case i du tableau
  {
  for(int j = i+1; j < n; j++)//On vérifie s'il n'y a pas de nombre inférieur
         {                              //Dans les cases suivantes
          if(data[j] < data[i])
            {
              tmp = data[i];    //Si c'est le cas on intervertit les cases
              data[i] = data[j];
              data[j] = tmp;
            }
          }
       }
}
// Question 8 : (1.00 pts)
int somme()
 {
        if (k <= 0)
       return 0;
      else
      return data[(k--)-1] + somme();
  }

//Question 9 (1.00 pts)
// Question ajouter le patron de classe   template <typename T>  class tab
// mettre T à la place de int pour data même pour les type de la fct somme
// appel dans le main tab <type> p(3)  ; type = int, double ou float;

};
// Test
int main(int argc, char** argv) {

      tab t1(3);
      t1.put(0,2);
    t1.put(1,4);
      t1.put(2,5);
      t1.afficher();
      tab t2(4);
    t2.saisir();
      //cout<<t2.isIn(4);
      t2.afficher();
      //t2.add(t1);

      tab t3=t1+t2;
      t3.afficher();

      cout<<endl<<t3[3];
      t3[3]=6;
      t3.afficher();
      t3.trier();
      t3.afficher();
      cout<<t3.somme();

      return 0;
}
```

Exercice 3 : 4 pts

```
// Code Gaceb mars 2019
#include <iostream>
using namespace std;
class Forme   (1pts)
{
  protected:
        double surface ;

  public:
        Forme(const double surface=0)
        {
                this->surface = surface;
        }

        double getSurface()
        {
                return surface;
        }
        virtual void affichetype()=0;
        virtual ~ Forme ()
         {

            cout<<" dest Forme";
         }
} ;

class Carre : public Forme  (0.5 pts)
{

  private:
        double cote;

  public:
        Carre(const double cote=0) : Forme(cote * cote)
      {
                this->cote = cote;
      }

         void affichetype()
        {
         cout<< endl<<"Carre (cote = "<<cote<< "cm)";
        }
        ~Carre ()
        {
            cout << endl<< "dest Carre";
        }

};

class Cercle : public Forme (0.5 pts)
{

  private:
        double rayon;

  public:
  Cercle(const double rayon=0) : Forme(3.14*rayon * rayon)
      {
                this->rayon = rayon;
      }

   void affichetype()
        {
         cout<< endl<<"Cercle (rayon = "<<rayon<< "cm)";
        }

        ~Cercle ()
        {
            cout << endl<< "dest Cercle";
        }

};

class Rectangle : public Forme  (1 pts)
{

  private:
        double h, w;

  public:
        Rectangle(const double h=0, const double w=0 ) : Forme(h*w)
      {
                this->h = h;
                this->w = w;
      }

        void affichetype()
        {
         cout<<endl<<"Rectangle (H="<<h<< "cm et w="<< w << "cm)";
        }

       ~Rectangle ()
        {
            cout << endl<< "dest rectangle";
        }

};

//  (1 pts)  pour l'ajout des destructeur 0.25 pour chaque + reponse affichage

int main()
{

// Tableau de trois figures géométriques
Forme ** figures = new Forme*[3];

//Création d'un carré de 2 cm de coté
figures[0] = new Carre(2);

//Création d'un cercle de 3 cm de rayon
figures [1] = new Cercle(3);

//Création d'un rectangle H=5.2 et L=2 cm
figures [2] = new Rectangle(5.2, 2);

//Affichage des surfaces
for (int i=0 ; i<3; i++)
{
```

```
  figures[i]->affichetype();
  cout <<":surface=)"
      <<figures[i]->getSurface()
      <<"cm2"<<endl;
   figures[i]->~Forme();

}

delete [] figures ;
figures=0;

cout<<"fin";
      return 0;
}
```

```
Carre (cote = 2cm):surface=)4cm2

dest Carre dest Forme
Cercle (rayon = 3cm):surface=)28.26cm2

dest Cercle dest Forme
Rectangle (H=5.2cm et w=2cm):surface=)10.4cm2

dest rectangle dest Formefin
--------------------------------
Process exited after 0.01803 seconds with return value 0
Appuyez sur une touche pour continuer... _
```