TD4 : Classe et objet (2/2)

Exercice 1 : Constructeur et destructeur en mémoire

Soit le programme C++ suivant :

```
#include <iostream>
using namespace std;

class Point2D {
  public:
    float x,y;
    Point2D () {x = 0.0; y = 0.0; cout << "Point2D nul créé\n";}
    Point2D (float _x, float _y) {
      x = _x; y = _y;
      cout << "Point2D créé\n";
    }
    Point2D (const Point2D & p) {
      x = p.x; y = p.y;
      cout << "Point2D copié\n";
    }
    ~Point2D () {cout << "Point2D détruit\n";}
};

int main() {
  Point2D pt1;
  Point2D pt2 (1.0,2.5);
  Point2D * ppt3 = new Point2D (pt2);
  delete ppt3;
  return 0;
}
```

a. Dessiner l'état de la mémoire après chaque construction d'objet en supposant que la valeur de retour du main est stockée à l'adresse 3 987 546 988 et qu'il n'y a pas de problème d'allocation mémoire.

b. Donner la trace écran de l'exécution de ce programme.

c. Au lieu de définir un constructeur sans paramètre et un autre avec comme paramètres les données membres, vous pouvez utiliser des valeurs par défaut. Ecrivez un constructeur avec paramètres par défaut pouvant remplacer les deux autres.

Exercice 2 : Appel à des fonctions membres

Soit le programme C++ suivant :

```
#include <iostream>
using namespace std;

class Point2D {
  public:
    float x,y;

    Point2D (float _x, float _y) {x = _x; y = _y;}
    ~Point2D () {}
```

```
    float distanceOrigine() const {return sqrt(x*x+y*y);}
    float distancePoint (const Point2D & p) const {
       return sqrt((p.x-x)*(p.x-x)+(p.y-y)*(p.y-y));
    }
}

int main() {
  Point2D pt1 (2.6,7.5);
  Point2D pt2 (1.4,3.8);
  float distpt1O;
  float distpt2pt1;
  distpt1O = pt1.distanceOrigine();
  distpt2pt1 = pt2.distancePoint(pt1);
  return 0;
}
```

a. Dessiner l'état de la mémoire avant les suppressions de frame (sauf constructeurs) en supposant que la valeur de retour du main est stockée à l'adresse 3 987 546 988.

b. La fonction distanceOrigine n'est finalement qu'un cas particulier de distancePoint. Réécrivez le code de distanceOrigine afin d'utiliser le code de distancePoint.