TD1 : Vie et mort des variables en mémoire (1/2)

Exercice 1 : Procédure récursive sur un tableau

Considérons le programme C++ suivant :

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

void mystere(double t[], int a, int b) {
  double aux;
  if (a < b) {
    aux = t[a];
    t[a] = t[b];
    t[b] = aux;
    mystere(t, a+1, b-1);
  }
  else {
    /* dessiner l'état de la mémoire */
  }
}

int main() {
  double monTab[4] = {9.0, 10.0, 11.0, 12.0};
  int i;

  mystere(monTab, 0, 3);

  cout << "Tableau apres traitement :\n";
  for (i = 0; i < 4; i++)
    cout << "monTab["<< i <<"] = "<< monTab[i] << endl;

  return 0;
}
```

a. Que fait la procédure `mystere` ? Autrement dit, si vous deviez lui donner un nom plus explicite, lequel choisiriez-vous ?

b. Dessinez l'état de la pile au moment indiqué en commentaire. Vous utiliserez le modèle théorique de pile vu en cours. Vous supposerez que la valeur de retour du main est stockée à l'adresse 3 987 546 988.

Exercice 2 : Tableaux de structures et fonction récursive

Soit le programme C++ suivant :

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

struct Point {
  double x;
  double y;
};
```

```
double dist (Point p1, Point p2) {
  return sqrt((p1.x - p2.x)*(p1.x - p2.x) + (p1.y - p2.y)*(p1.y - p2.y));
}

double longueurchemin (const Point * chemin, int nb) {
/* Préconditions : chemin est un tableau contenant au moins nb points,
                   avec nb superieur ou égal à 2
   Résultat : retourne la longueur du chemin de points
*/
  double res;
  double d1, d2;

  if (nb==2) res = dist(chemin[1],chemin[0]);
  else {
    d1 = dist(chemin[nb-1],chemin[nb-2]);
    d2 = longueurchemin(chemin,nb-1);
    res = d1 + d2;
  }
  return res;
}

int main() {
  double perimetre;
  Point cheminTriangle[4];
  cheminTriangle[0].x = 0.0 ;
  cheminTriangle[0].y = 0.0 ;
  cheminTriangle[1].x = 3.0 ;
  cheminTriangle[1].y = 0.0 ;
  cheminTriangle[2].x = 3.0 ;
  cheminTriangle[2].y = 1.0 ;
  cheminTriangle[3].x = cheminTriangle[0].x ;
  cheminTriangle[3].y = cheminTriangle[0].y ;

  perimetre = longueurchemin(cheminTriangle, 4);
  cout << "Le perimetre du triangle vaut " << perimetre << endl;

  return 0;
}
```

a. Combien d'octets une variable de type Point occupe-t-elle en mémoire (sur une machine où un int occupe 4 octets et un double 8) ?

b. Combien d'octets occupe le tableau cheminTriangle ?

c. La fonction longueurchemin est-elle récursive ? Même question pour la fonction dist.

d. A quoi sert le mot-clé const dans l'entête de la fonction longueurchemin ?

e. Dessinez l'évolution de la pile lors de l'exécution de ce programme, en utilisant le modèle théorique de pile vu en cours. Vous supposerez que la valeur de retour du main est stockée à l'adresse 3 987 546 998. Vous ferez un dessin à chaque fois que vous êtes sur le point de sortir d'un sous-programme (i.e. après l'affectation de la valeur de retour et avant le retour au programme appelant).

f. Combien de fois la fonction longueurchemin est-elle appelée ? Même question pour la fonction dist.

g. Que pensez-vous de l'implémentation récursive de longueurchemin, en termes de ressources mémoire ?

h. Quelle valeur obtenez-vous pour la variable perimetre ? Indication : $\sqrt{10} \approx 3.162$