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 ?

Elle inverse l'ordre des éléments du tableau qu'on lui passe en paramètre. On pourrait la nommer « inverserOrdre » par exemple.

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.

PILE

VR main		3 987 546 988
monTab[3]	9.0	3 987 546 980
monTab[2]	10.0	3 987 546 972
monTab[1]	11.0	3 987 546 964
monTab[0]	12.0	3 987 546 956 = monTab
i		3 987 546 952
appel à mystere(monTab, 0, 3)		
t	3 987 546 956	3 987 546 944
a	0	3 987 546 940
b	3	3 987 546 936
aux	9.0	3 987 546 928
appel à mystere(t, a+1,b-1)		
t	3 987 546 956	3 987 546 920
a	1	3 987 546 916
b	2	3 987 546 912
aux	10.0	3 987 546 904
appel à mystere(t, a+1,b-1)		
t	3 987 546 956	3 987 546 896
a	2	3 987 546 892
b	1	3 987 546 888
aux		3 987 546 880

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) ?

2 fois 8 octets, soit 16 octets.

b. Combien d'octets occupe le tableau cheminTriangle ?

4 fois sizeof(Point), soit 4*16 = 64 octets.

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

longueurchemin est récursive, mais pas dist.

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

Un paramètre déclaré avec le qualifieur const ne doit pas être modifié à l'intérieur de la fonction. Ainsi, une instruction comme `chemin[2].x = 0.0;` provoquerait une erreur de compilation. Cela permet d'indiquer aux utilisateurs de cette fonction qu'elle n'a pas d'effet de bord : le tableau de points n'est pas modifié, bien qu'il soit passé par adresse. En termes algorithmiques, il est passé en mode « donnée ».

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).

Pour une meilleure visibilité, seulement les trois derniers chiffres des adresses sont indiqués.

Dessin #1 (sortie de dist)

PILE

VR main		998
perimetre		990
cheminTriangle[3].y	0.0	982
cheminTriangle[3].x	0.0	974
cheminTriangle[2].y	1.0	966
cheminTriangle[2].x	3.0	958
cheminTriangle[1].y	0.0	950
cheminTriangle[1].x	3.0	942
cheminTriangle[0].y	0.0	934
cheminTriangle[0].x	0.0	926
appel à longueurChemin(cheminTriangle,4)		
VR		918
chemin	926	910
nb	4	906
res		898
d1		890
d2		882
appel à dist(chemin[3], chemin[2])		
VR	3.162	874
p1.y	0.0	866
p1.x	0.0	858
p2.y	1.0	850
p2.x	3.0	842

Dessin #2 (sortie de dist)

PILE

VR main		998
perimetre		990
cheminTriangle[3].y	0.0	982
cheminTriangle[3].x	0.0	974
cheminTriangle[2].y	1.0	966
cheminTriangle[2].x	3.0	958
cheminTriangle[1].y	0.0	950
cheminTriangle[1].x	3.0	942
cheminTriangle[0].y	0.0	934
cheminTriangle[0].x	0.0	926
appel à longueurChemin(cheminTriangle,4)		
VR		918
chemin	926	910
nb	4	906
res		898
d1	3.162	890
d2		882
appel à longueurChemin(chemin,3)		
VR		874
chemin	926	866
nb	3	862
res		854
d1		846
d2		838
appel à dist(chemin[2], chemin[1])		
VR	1.0	830

p1.y	1.0	822
p1.x	3.0	814
p2.y	0.0	806
p2.x	3.0	798

Dessin #3 (sortie de dist)

PILE

VR main		998
perimetre		990
cheminTriangle[3].y	0.0	982
cheminTriangle[3].x	0.0	974
cheminTriangle[2].y	1.0	966
cheminTriangle[2].x	3.0	958
cheminTriangle[1].y	0.0	950
cheminTriangle[1].x	3.0	942
cheminTriangle[0].y	0.0	934
cheminTriangle[0].x	0.0	926

appel à longueurChemin(cheminTriangle,4)

VR		918
chemin	926	910
nb	4	906
res		898
d1	3.162	890
d2		882

appel à longueurChemin(chemin,3)

VR		874
chemin	926	866
nb	3	862
res		854
d1	1.0	846
d2		838

appel à longueurChemin(chemin,2)

VR		830

Dessin #4 (sortie de longueurchemin)

PILE

VR main		998
perimetre		990
cheminTriangle[3].y	0.0	982
cheminTriangle[3].x	0.0	974
cheminTriangle[2].y	1.0	966
cheminTriangle[2].x	3.0	958
cheminTriangle[1].y	0.0	950
cheminTriangle[1].x	3.0	942
cheminTriangle[0].y	0.0	934
cheminTriangle[0].x	0.0	926

appel à longueurChemin(cheminTriangle,4)

VR		918
chemin	926	910
nb	4	906
res		898
d1	3.162	890
d2		882

appel à longueurChemin(chemin,3)

VR		874
chemin	926	866
nb	3	862
res		854
d1	1.0	846
d2		838

appel à longueurChemin(chemin,2)

VR	3.0	830

chemin	926	822
nb	2	818
res		810
d1		802
d2		794

appel à dist(chemin[1], chemin[0])

VR	3.0	786
p1.y	0.0	778
p1.x	3.0	770
p2.y	0.0	762
p2.x	0.0	754

chemin	926	822
nb	2	818
res	3.0	810
d1		802
d2		794

Dessin #5 (sortie de longeurchemin)

PILE

VR main		998
perimetre		990
cheminTriangle[3].y	0.0	982
cheminTriangle[3].x	0.0	974
cheminTriangle[2].y	1.0	966
cheminTriangle[2].x	3.0	958
cheminTriangle[1].y	0.0	950
cheminTriangle[1].x	3.0	942
cheminTriangle[0].y	0.0	934
cheminTriangle[0].x	0.0	926

appel à longueurChemin(cheminTriangle,4)

VR		918
chemin	926	910
nb	4	906
res		898
d1	3.162	890
d2		882

appel à longueurChemin(chemin,3)

VR	4.0	874

Dessin #6 (sortie de longueurchemin)

PILE

VR main		998
perimetre		990
cheminTriangle[3].y	0.0	982
cheminTriangle[3].x	0.0	974
cheminTriangle[2].y	1.0	966
cheminTriangle[2].x	3.0	958
cheminTriangle[1].y	0.0	950
cheminTriangle[1].x	3.0	942
cheminTriangle[0].y	0.0	934
cheminTriangle[0].x	0.0	926

appel à longueurChemin(cheminTriangle,4)

VR	7.162	918
chemin	926	910
nb	4	906
res	7.162	898
d1	3.162	890
d2	4.0	882

chemin	926	866
nb	3	862
res	4.0	854
d1	1.0	846
d2	3.0	838

Dessin #7 (sortie de main)

PILE

VR main	0	998
perimetre	7.162	990
cheminTriangle[3].y	0.0	982
cheminTriangle[3].x	0.0	974
cheminTriangle[2].y	1.0	966
cheminTriangle[2].x	3.0	958
cheminTriangle[1].y	0.0	950
cheminTriangle[1].x	3.0	942
cheminTriangle[0].y	0.0	934
cheminTriangle[0].x	0.0	926

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

3 fois chacune.

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

L'implémentation récursive est beaucoup plus gourmande en ressources mémoire que le serait l'implémentation itérative. L'implémentation récursive ne doit pas être utilisée pour des chemins très longs.

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

$\sqrt{10} + 3 + 1 \approx 7.162$