Programmation système et réseaux

Corrigé du TP 5: Pointeurs

Arnaud Giersch Guillaume Latu

Consigne. Tous les exercices devront se faire en utilisant l'arithmétique des pointeurs. En particulier, vous ne devrez pas utiliser d'indices pour accéder aux éléments des tableaux.

1. Écrire la fonction sommeEtMultiplie calculant la somme et le produit de deux réels passés en paramètres, puis un programme qui utilise cette fonction. Pour cela vous écrirez votre code dans un fichier *sommemult.c* et utiliserez le fichier *sommemult.h* qui suit :

   ```
   void sommeEtMultiplie (float a, float b, float *somme, float *mult);
   ```

   Correction :

   ```
   #include <stdio.h>
   #include "sommemult.h"

   void sommeEtMultiplie (float a, float b, float *somme, float *mult)
   {
       *somme = a + b;
       *mult = a * b;
   }

   int main ()
   {
       float x = 2.0, y = 9.0;
       float res1, res2;
       sommeEtMultiplie (x, y, &res1, &res2);
       printf ("a = %g; b = %g; somme = %g; mult = %g\n", x, y, res1, res2);
       return 0;
   }
   ```

2. Écrire une fonction qui calcule la somme et le produit terme à terme de deux vecteurs de réels de taille *n*, puis un programme qui utilise cette fonction. Vous écrirez aussi une fonction vAffiche vous permettant d'afficher le contenu d'un vecteur de réels de taille *n*. Vous utiliserez le fichier *vsommemult.h* suivant :

   ```
   void vSommeEtMultiplie (int n, float *a, float *b, float *somme, float *mult);
   void vAffiche (int n, float *a);
   ```

   Correction :

   ```
   #include <stdio.h>
   #include "vsommemult.h"

   void vSommeEtMultiplie (int n, float *a, float *b, float *somme, float *mult)
   {
       float *fina;
       for (fina = a + n; a < fina; a++, b++) {
           *somme++ = *a + *b;
           *mult++ = *a * *b;
   ```

```
    }
}

void vAffiche (int n, float *a)
{
    printf ("[");
    while (n-- > 0)
        printf ("%g%s", *a++, (n > 0)? ", ": "]\n");
}

#define TAILLE 4

int main()
{
    float x[TAILLE ]= {1.0, 2.0, 3.0, 4.0};
    float y[TAILLE ] = {2.5, 2.1, 1.2, 0.5};
    float res1[TAILLE], res2[TAILLE];
    vSommeEtMultiplie (TAILLE, x, y, res1 , res2 );
    printf ("x = "); vAffiche (TAILLE, x);
    printf ("y = "); vAffiche (TAILLE, y);
    printf ("somme = "); vAffiche (TAILLE, res1);
    printf ("mult = "); vAffiche (TAILLE, res2);
    return 0;
}
```

3. Coder une fonction qui inverse lettre à lettre une chaîne de caractères (sans utiliser de fonctions de la bibliothèque standard, y compris strlen). La chaîne « étira » sera par exemple inversée en « arité ». Pour information, une chaîne de caractères se termine par le caractère '\0'. Le prototype de la fonction est le suivant :

```
void inverseChaine (char *in , char *out );
```

Correction :

```
#include < stdio .h>
#include "inversechaine.h"

void inverseChaine (char *in , char *out)
{
    char *cin;
    /* fait pointer out à la fin du résultat */
    for (cin = in; *cin != '\0'; cin++)
        /*nop*/ ;
    out += cin - in;
    /* copie la chaîne à l'envers */
    *out-- = '\0';
    for (cin = in; *cin != '\0'; cin++)
        *out-- = *cin;
}

int main()
{
    char *chaine = "arité";
    char resultat [20];
    inverseChaine (chaine , resultat );
    printf ("\"%s\" donne \"%s\"\n", chaine, resultat );
    return 0;
}
```

4. Écrire une fonction compareChaine qui permet de comparer deux chaînes de caractères. Cette fonction renverra 1 si les chaînes sont identiques et 0 sinon. Le prototype de la fonction à implémenter est suivant :

```
int compareChaine (char *ch1, char *ch2);
```

Correction :

```
#include <stdio.h>
#include "comparechaine.h"

int compareChaine (char *c1, char *c2)
{
  for (; *c1 == *c2 && (*c1 != '\0'); c1++, c2++)
      /*nop*/;
  return (*c1 == *c2);
}

int main()
{
    char *c1 = "charité";
    char *c2 = "arité";
    char *c3 = "charité";
    int res;
    res = compareChaine (c1, c2);
    printf ("compare (\"%s\", \"%s\") -> %d\n", c1, c2, res);
    res = compareChaine (c1, c3);
    printf ("compare (\"%s\", \"%s\") -> %d\n", c1, c3, res);
    return 0;
}
```

5. Coder une routine sousChaine qui permet de savoir si une chaîne « patron » apparaît au sein d'une chaîne « cible ». Cette routine renverra 1 si c'est le cas, et 0 sinon. Le prototype de la fonction à implémenter est suivant :

```
int sousChaine (char *patron, char *cible);
```

Correction :

```
#include <stdio.h>
#include "souschaine.h"

/* retourne 1 si patron est un préfixe de cible, 0 sinon */
int estPrefixe (char *patron, char *cible)
{
  for (; *patron == *cible && (*patron != '\0'); patron++, cible++)
      /*nop*/;
  return (*patron == '\0');
}

int sousChaine (char *patron, char *cible)
{
    for (; *cible != '\0'; cible++) {
        if (estPrefixe (patron, cible))
            return 1;
    }
    return 0;
}

int main()
```

```
{
    char *c1 = "départ";
    char *patron = "part";
    char *c2 = "partir en mer";
    int res;
    res = estPrefixe (patron, c1);
    printf ("compare (\"%s\", \"%s\") -> %d\n", patron, c1, res);
    res = estPrefixe (patron, c2);
    printf ("compare (\"%s\", \"%s\") -> %d\n", patron, c2, res);
    c1 = "il serait bon de partir en vacances";
    res = sousChaine (patron, c1);
    printf ("compare (\"%s\", \"%s\") -> %d\n", patron, c1, res);
    c2 = "manges ta soupe";
    res = sousChaine (patron, c2);
    printf ("compare (\"%s\", \"%s\") -> %d\n", patron, c2, res);
    return 0;
}
```

6. Le but de la fonction triSelection que vous allez écrire est de trier un vecteur de nombres réels. Pour cela, vous utiliserez l'algorithme du *tri par sélection* dont le principe est le suivant (pour un vecteur dont les éléments sont numérotés de 1 à n) :
   – pour $d = 1$ à $n - 1$, faire
     – rechercher le minimum parmi les élément d à n du vecteur ;
     – échanger ce minimum avec le d^{e} élément du vecteur.

   Voici le prototype de la fonction :

```
void triSelection (int n, float *vect);
```

Correction :

```
#include <stdio.h>
#include "triselection.h"

void swap (float *a, float *b)
{
    float tmp = *a;
    *a = *b;
    *b = tmp;
}

void triSelection (int n, float *vect)
{
    float *pdeb;
    float *pfin = vect + n - 1;
    for (pdeb = vect; pdeb < pfin; pdeb++) {
        float *pmin = pdeb;
        float *pelt;
        for (pelt = pdeb + 1; pelt <= pfin; pelt++) {
            if (*pelt < *pmin)
                pmin = pelt;
        }
        swap (pdeb, pmin);
    }
}

void vAffiche (int n, float *a)
{
    printf ("[");
```

```
    while (n-- > 0)
        printf ("%g%s", *a++, (n > 0)? ", ": "]\n");
}

int main()
{
    float x[] = {1, 5, 22, 7, 4, 2, 8, 9, 0, 10, 24, 11};
    int taille = sizeof x / sizeof x[0];
    printf ("taille = %d\n", taille);
    printf ("x = ");
    vAffiche (taille, x);
    triSelection (taille, x);
    printf ("x trié = ");
    vAffiche (taille, x);
    return 0;
}
```