Université Paris-Nord L1
Institut Galilée Année 2008-2009
Licence 1 - 2ième semestre

Travaux dirigés 2 : pointeurs et tableaux

Correction.
- Le premier exercice fait reference au cours. L'arithmétique des pointeurs n'a été abordée que rapidement à la fin du cours, donc il est nécessaire de faire un bref rappel avant d'attaquer l'exo de rappel de cours.
- Pour le dernier exercice, il peut être bon de leur rappeler la procédure à suivre :
  - on se donne des exemples
  - on trouve un algorithme en francais
  - on traduit l'algorithme en C en s'aidant de commentaires issus de l'algo en francais
  - on test sur les exemples qu'on s'est donnes

  C'est un exercice qu'ils ont déjà vu au semestre 1 (première version).

Dans l'ensemble du TD, on considère que l'on travaille sur une architecture 32 bits.

1 Rappel de cours

On considère que les déclarations suivantes ont été faites :

```
int a;
char tab[10];
```

Une expression avec pointeurs (resp. sans pointeurs) vous est donnée, vous devez la ré-écrire sans (resp. avec) l'usage explicite des pointeurs.

1. *(&a)
2. *tab
3. *(tab + 0)
4. (*tab) + 1
5. &(tab[0])
6. &(tab[i])
7. ++tab[i]

Correction.

1. *(&a) : a
2. *tab : tab[0]
3. *(tab + 0) : tab[0]
4. (*tab) + 1 : tab[0] + 1
5. &(tab[0]) : tab

6. &(tab[i]) : (tab + i)
7. ++tab[i] : ++(*(tab + i))

Si l'on écrit la suite d'instructions suivante :

```
char *p;

p = tab + 1;
```

1. Quelle sera la différence en octets entre les deux adresses `tab` et `p` ?

   Correction. sizeof(char) = 1 octet en C (standard, dépend pas de l'archi)
2. Même question si `tab` et `p` sont déclarés pointer sur des `int` ?

   Correction. sizeof(int) = 4 octet sur PC 32 bits, Leur dire que l'on peut faire la difference entre pointeurs. Pas dit en cours. L'exo s'y prete bien.

2 Occupation mémoire

Il est demandé dans cet exercice de représenter en mémoire les données déclarées dans un programme, ainsi que leurs différentes valeurs, à un moment donné de l'exécution. Pour cela, vous représenterez l'occupation des données en mémoire dans un tableau à 3 colonnes comme montré ci-dessous :

identificateur	adresse	valeur
a	?	?
⋮	⋮	⋮

Pour déterminer les adresses, on fera les approximations suivantes :
– les données sont réservées en mémoire dans l'ordre de leur déclaration
– la première adresse démarre à 1000[1]
– l'architecture est 32 bits

Le programme est donné ci-dessous. Notez que le programme fait une utilisation abusive des pointeurs, l'objectif étant simplement de vous familiariser avec la syntaxe et la sémantique des instructions manipulant des pointeurs.

```
#include <stdlib.h>
#include <stdio.h>

int main()
{
  int a = 10;
  int b = 5;
  int tab[3] = {1,2,3};
  int *p_int;

  /* représenter l'occupation mémoire */
```

[1]L'adresse est donnée en notation décimale bien que la notation hexadécimale soit la plus utilisée pour des raisons de concision et de facilité de conversion en base 2

Correction.

identificateur	adresse	valeur
a	1000	10
b	1004	5
tab[0]	1008	1
tab[1]	1012	2
tab[2]	1016	3
p_tab	1020	?

```
  tab[0] = a;
  *(tab + 1) = b;
  p_int = tab + 2;

  /* représenter l'occupation mémoire */
```

Correction.

identificateur	adresse	valeur
a	1000	10
b	1004	5
tab[0]	1008	10
tab[1]	1012	5
tab[2]	1016	3
p_tab	1020	1016

```
  *p_int = *(p_int - 1);
  --p_int;
  *p_int = *(p_int - 1);
  --p_int;
  *p_int = *(p_int + 2);

  /* représenter l'occupation mémoire */
```

Correction.

identificateur	adresse	valeur
a	1000	10
b	1004	5
tab[0]	1008	5
tab[1]	1012	10
tab[2]	1016	5
p_tab	1020	1008

```
  printf("%d\t%d\t%d\t%d\t%d\n",a,b,tab[0],tab[1],tab[2]);

  /* donner l'affichage */
```

Correction. /* 10 5 5 10 5 */

```
  return EXIT_SUCCESS;
}
```

3 Compter le nombre d'occurences d'un entier dans un tableau d'entiers

Nous voulons écrire un programme qui, étant donné un tableau d'entiers déjà initialisé, demande à l'utilisateur quel entier chercher et affiche ensuite le nombre d'occurrences de cet entier dans le tableau.

1. Écrire le programme en utilisant l'opérateur `[]`.

Correction.

```
#include <stdlib.h> /* EXIT_SUCCESS */
#include <stdio.h> /* printf */

#define TAILLE 4 /* taille du tableau utilisateur */

/* declaration de fonctions utilisateurs */

int main()
{
    int tab[TAILLE] = {-1,3,3,2}; /* tableau a initialiser par l'utilisateur */
    int elt; /* l'elt a chercher */
    int nb_occ = 0; /* le nombre d'occurrences de elt dans tab */
    int i; /* var. de boucle */

    /* saisie de l'entier à chercher */
    printf("Compte le nombre d'occurrences de quel entier ?\n");
    scanf("%d",&elt);

    /* compte le nombre d'occurrences */
    for(i = 0;i < TAILLE;i = i + 1) /* chaque case du tableau */
    {
        if(tab[i] == elt) /* trouvé */
        {
            /* un de plus */
            nb_occ = nb_occ + 1;
        }
    }
    /* i >= TAILLE */

    /* affiche résultats */
    printf("Il y a %d occurrences de %d dans le tableau.\n",nb_occ,elt);

    return EXIT_SUCCESS;
}

/* implantation de fonctions utilisateurs */
```

2. Écrire le programme en utilisant explicitement les pointeurs pour accéder aux éléments du tableau, c'est-à-dire sans utiliser une variable d'indice.

Correction.

```
1   #include <stdlib.h> /* EXIT_SUCCESS */
2   #include <stdio.h> /* printf */
3
4   #define TAILLE 4 /* taille du tableau utilisateur */
5
6   int main()
7   {
8       int tab[TAILLE] = {-1,3,3,2}; /* tableau a initialiser par l'utilisateur *
9       int elt; /* l'elt a chercher */
10      int nb_occ = 0; /* le nombre d'occurrences de elt dans tab */
11      int *p_int; /* var. de boucle */
12
13      /* représentation de l'occupation mémoire */
14
15      /* saisie de l'entier à chercher */
16      printf("Compte le nombre d'occurrences de quel entier ?\n");
17      scanf("%d",&elt);
18
19      /* compte le nombre d'occurrences */
20      for(p_int = tab;p_int < (tab + TAILLE);p_int = p_int + 1) /* chaque case c
21      {
22          if(*p_int == elt) /* trouvé */
23          {
24              /* un de plus */
25              nb_occ = nb_occ + 1;
26          }
27      }
28      /* p_int >= tab + TAILLE */
29
30      /* affiche résultats */
31      printf("Il y a %d occurrences de %d dans le tableau.\n",nb_occ,elt);
32
33      return EXIT_SUCCESS;
34  }
```

3. Faire la trace de cette deuxième version. Une ligne sera ajoutée au tableau de trace pour noter l'adresse des variables. Les adresses seront choisies en respectant la même convention que pour l'exercice précédent.

Correction. Leur donner le format avant.

numéro de ligne	tab[0]	tab[1]	tab[2]	tab[3]	elt	nb_occ	p_int	Affichage (sortie écran)
adresse	1000	1004	1008	1012	1016	1020	1024	
initialisation	-1	3	3	2	?	0	?	
16								Compte le nombre d'occurrences de quel entier ?\n
17					3			
20							1000	
27							1004	
25						1		
27							1008	
25						2		
27							1012	
27							1016	
31								Il y a 2 occurrences de 3 dans le tableau.\n

N.B. : La notation `[]` a été introduite pour rendre plus lisible et plus explicite l'accès au tableau, donc la première version est considérée comme la version correcte.