TD & TP 2: Pointeurs, tableaux et fonctions

IUT CAEN – Département Informatique

Module M1103

Dans ce TD, vous aurez peut être besoin d'afficher la valeur d'une adresse mémoire. Pour ce faire, vous pouvez utiliser printf avec le format %u.

Seuls les exerices 5, 6 et 7 sont à faire en TP.

Exercice 1

Compléter le tableau ci-dessous.

programme	a	b	c	p_1	$*p_1$	p_2	$*p_2$
int a, b, c, $*p_1$, $*p_2$;	?	?	?	-	?	-	?
$a = 1$, $b = 2$, $c = 3$;	?	?	?	-	?	-	?
$p_1 = \&a$, $p_2 = \&c$;	?	?	?	?	?	?	?
$*p_1 = (*p_2) + +$;	?	?	?	?	?	?	?
$p_1 = p_2$;	?	?	?	?	?	?	?
$p2 = \&b$;	?	?	?	?	?	?	?
$*p1- = *p2$;	?	?	?	?	?	?	?
$+ + *p2$;	?	?	?	?	?	?	?
$*p1* = *p2$;	?	?	?	?	?	?	?
$a = + + *p2 * *p1$;	?	?	?	?	?	?	?
$p_1 = \&a$;	?	?	?	?	?	?	?
$*p_2 = *p_1/ = *p_2$;	?	?	?	?	?	?	?

Exercice 2

Soit `iptr` un pointeur qui "pointe" sur un tableau `tabA`:

```
int tabA[] = {12, 23, 34, 45, 56, 67, 78, 89, 90};
int *iptr=NULL;
iptr = tabA;
```

Quelles valeurs ou adresses fournissent ces expressions :

1. *iptr+2 ,
2. *(iptr+2),
3. &iptr+1 ,
4. &tabA[4]-3 ,
5. tabA+3 ,
6. &tabA[7]-iptr
7. iptr+(*iptr-10) ,
8. *(iptr+*(iptr+8)-tabA[7])

Exercice 3 (Échanges)

1. Soit la fonction suivante :

```
void echange1 (int x, int y) {
     int z;
     z=x; x=y; y=z;
}
```

   - Pourquoi ne fonctionne-t-elle pas lorsqu'on l'appelle avec par exemple `a=2;b=3; echange1(a,b)` ?
   - Représentez la mémoire lors de l'exécution de ce morceau de programme.

2. Soit la fonction suivante :

```
void echange2 (int *x, int *y) {
     int *z;
     *z=*x; *x=*y; *y=*z;
}
```

   - Pourquoi risque-t-elle ne pas fonctionner lorsqu'on l'appelle avec par exemple `a=2;b=3; echange2(&a,&b)` ?
   - Représentez la mémoire lors de l'exécution de ce morceau de programme.

3. Soit la fonction suivante :

```
void echange3 (int *x, int *y) {
     int z;
     z=*x; *x=*y; *y=z;
}
```

   - Fonctionne-t-elle lorsqu'on l'appelle avec par exemple `a=2;b=3;echange2(&a,&b)` ?
   - Représentez la mémoire lors de l'exécution de ce morceau de programme.

Exercice 4 (Pointeurs vs. tableau)

Soit les deux déclarations suivantes :

```
char tabString[]="toto";
char *ptrString="Titi";
```

Considérons les instructions suivantes :

```
tabString[2]='e';
ptrString[2]='e';
tabString=ptrString;
ptrString = tabString;
```

Lesquelles provoquent une erreur ?

Exercice 5 (Jeu des cailloux: À faire en TP)

Le jeu des cailloux : un tas de N cailloux se trouve entre deux joueurs ; à tour de rôle chacun prend 1, 2 ou 3 cailloux. Celui qui est obligé de prendre le dernier caillou a perdu.
Ce jeu possède une stratégie gagnante : le joueur qui réussit, à laisser un nombre de cailloux égal à un multiple de 4 plus 1, à chaque fois qu'il doit jouer, gagne à coup sûr.
Le but est d'écrire un programme qui simule le jeu des cailloux (entre vous et la machine) :

- Ecrire une fonction `initialiser` qui demande le nombre de cailloux et quel joueur commence le premier.
- Ecrire une fonction `utilisateur_joue` qui demande à l'utilisateur le nombre de cailloux à prendre et qui met à jour le nombre de cailloux restants.
- Ecrire une fonction `machine_joue` qui permet à la machine de prendre un nombre de cailloux et qui met à jour le nombre de cailloux restants.
- Ecrire le programme principal.

Les prototypes :

```
void initialiser(int *nbCa, int *jo);

void utilisateur_joue(int *nbCa) ;

void machine_joue(int *nbCa) ;
```

`nbCa` : le nombre de cailloux

`jo` : joueur (1 : lutilisateur et 2 : la machine)

Exercice 6 (Insertion dans un tableau trié: À faire en TP)

Dans cet exercice vous devez utiliser des pointeurs à chaque fois que cela est possible. L'objectif est d'écrire un programme qui insère des éléments dans un tableau tout en les triant. Nous distinguerons la taille `taille` du tableau (c'est-à-dire le nombre maximum d'éléments qu'il peut contenir) du nombre `nbEl` d'éléments déjà insérés dans le tableau. Au début, le tableau est vide et à chaque nouvelle valeur, nous allons rechercher où l'insérer dans le tableau puis nous allons décaler les éléments suivants pour faire de la place. Il n'est plus possible d'ajouter de nouvel élément dès que `nbEl=taille`.

1. Écrivez une fonction `indiceInsert` qui, étant donné un tableau d'entiers `tab` de taille `taille` contenant `nbEl` éléments triés par ordre croissant et un entier `val`, retourne l'indice auquel `val` doit être inséré pour que le tableau reste trié. Si le tableau est plein ou si l'élément est déjà présent, la fonction retournera la valeur -1.
2. Écrivez une fonction `insertElt` qui réalise l'insertion de `val` dans `tab` si l'élément n'est pas déjà présent et si le tableau n'est pas plein. Cette fonction retournera 1 si l'insertion a été réalisée, 0 sinon.
3. Écrivez un programme complet qui permet de tester les fonctions précédentes.