Programmation C
TP n° 8
Pointeurs

Florent Devin

1 Rappels

1.1 Adressage direct

En programmation, on utilise des variables pour stocker des informations. La valeur d'une variable se trouve à un endroit spécifique dans la mémoire interne de l'ordinateur (adresse). Le nom de la variable nous permet alors d'accéder *directement* à cette valeur.

Adressage direct : Accès au contenu d'une variable par le nom de la variable. Exemple : `main(){ int a ; double b ; a=10 ; b=22.2 ; }`

1.2 Adressage indirect

Si on ne veut ou ne peut pas utiliser le nom d'une variable, il est possible de copier l'adresse de cette variable dans une variable spéciale, appelée pointeur. Ensuite, il est possible de retrouver l'information contenu dans la variable en passant par le pointeur.

Adressage indirect : Accès au contenu d'une variable, en passant par un pointeur qui contient l'adresse de la variable. Exemple : soit a une variable contenant la valeur 10 et pa un pointeur qui contient l'adresse de a. `main(){ int a ; int *pa ; a = 10 ; pa = &a ; }`

Définition : Un pointeur est une variable spéciale qui peut contenir l'adresse d'une autre variable. En langage C, chaque pointeur est limité à un type de données. Il peut contenir l'adresse d'une variable simple de ce type ou l'adresse d'une composante d'un tableau de ce type. Si un pointeur pa contient l'adresse d'une variable a, on dit que "pa pointe sur a". Lors du travail avec des pointeurs, on utilise le plus souvent :

- l'opérateur "adresse de " : `&variable` pour obtenir l'adresse d'une variable,
- l'opérateur " contenu de " : `*pointeur` pour accéder au contenu d'une adresse,
- une syntaxe de déclaration pour pouvoir déclarer un pointeur : `type* pointeur`

Lorsqu'il vous est demandé de réfléchir, puis de vérifier, faîtes le de le sens demandé. Ces exercices ne sont la que dans le but de vérifier votre compréhension des pointeurs.

2 Pointeurs

2.1 Exercice inutile

Déclarer un entier i et un pointeur p vers un entier. Initialiser l'entier i à une valeur arbitraire et faire pointer p vers i. Imprimer la valeur de i. Modifier l'entier pointé par p (en utilisant p, pas i). Imprimer la valeur de i.

2.2 Pointeur de pointeur

En s'aidant d'une représentation graphique, donner, à la fin de la séquence suivante, les valeurs de a, *p, **pp. Vérifier vos résultats à l'aide d'un programme.

```
int main (int argc, char** argv) {
  int a=0, b=1, *p, **pp;
  p = &a;
  a += b;
  pp = &p;
}
```

2.3 Un peu moins facile

Déclarer et initialiser statiquement un tableau d'entiers t avec des valeurs dont certaines seront nulles. Écrire une procédure qui parcoure le tableau t et qui imprime les index des éléments nuls du tableau, sans utiliser aucune variable de type entier.

2.4 Chaîne

On va coder un algorithme de cryptage très simple : on choisit un décalage (par exemple 5), et un a sera remplacé par un f, un b par un g, un c par un h, etc. On ne cryptera que les lettres majuscules et minuscules sans toucher ni à la ponctuation ni à la mise en page. On supposera que les codes des lettres se suivent de a à z et de A à Z. On demande de :

- Déclarer un tableau de caractères `mess` initialisé avec le message en clair ;
- Écrire une procédure `crypt` de cryptage d'un caractère qui sera passé par adresse ;
- Écrire le `main` qui activera `crypt` sur l'ensemble du message et imprimera le résultat.