Chapitre 9 : Pointeurs

Informatique de base
2013-2014

Sup Galilée

Qu'est-ce qu'un pointeur ?

Un pointeur est l'adresse d'une donnée.

En C, les pointeurs sont des valeurs au même titre que les nombres ou encore les structures. Il existe donc des constantes et des variables de type pointeur, ou plus exactement de type "pointeur vers une donnée de type T". Il existe aussi des opérateurs spécifiques pour manipuler des pointeurs.

Définition d'un pointeur

La définition d'un pointeur est réalisée comme suit :
`T *ident;`

Par exemple :

```
int *p;
char *p;
struct personne
  {
  char[20] nom;
  int age;
  } * ptr;
```

pour définir un pointeur sur un entier, un pointeur sur un caractère, et un pointeur sur une structure.

On peut déclarer simultanément des variables de type T et des variables vers une donnée de type T. Par exemple : `int n, *p;`

Manipulation d'un pointeur : adresse d'une donnée modifiable

L'adresse d'une donnée est obtenue par l'opérateur &

Si exp est le nom d'une donnée modifiable (une valeur gauche), alors `&exp` désigne une expression telle que

- type(`&exp`) = pointeur vers une donnée de type type(`exp`),
- val(`&exp`) = adresse de la donnée désignée par `exp`.

Par exemple, l'instruction suivante affecte l'adresse de la variable `v` à la variable `p` :

```
int *p, v;
p = &v;
```

Attention ! `&exp` n'est pas une valeur gauche. Par exemple, si `x` est une variable de type `int`, alors on ne peut pas écrire `&x = 12;`

Manipulation d'un pointeur : indirection

L'opérateur `*` permet d'accéder à la valeur d'une donnée à partir d'un pointeur vers cette donnée, c'est-à-dire à partir de son adresse. On dit que l'on réalise une indirection.

Si exp est un expression de type pointeur vers une donnée de type T, alors `*exp`
est une expression telle que

- type(`*exp`)=T,
- val(`*exp`)=valeur enregistrée dans la case désignée par `exp`.

L'expression `*exp` est une valeur gauche : elle est le nom de la donnée modifiable dont val(`exp`) est l'adresse.

Manipulation d'un pointeur : exemple

Considérons l'exemple suivant :

```
(1) int *p, v;
(2) v=0;
(3) p=&v;
(4) *p=15;
```

La ligne (3) permet d'affecter à `p` l'adresse de la variable `v`, et la ligne (4) permet de changer la valeur de `v` (par une indirection).

Pointeurs et tableaux

Soit un tableau `tab` de `n` cases de type T déclaré par `T[n] tab`. Alors le nom `tab` est en fait une constante dont la valeur est l'adresse du premier élément du tableau désigné par `tab`, autrement dit `tab` est un synonyme de `&tab[0]`.

De plus, si `i` est un entier compris entre 0 et `n-1`, alors `tab[i]` correspond à l'adresse de la `i+1`-ème case du tableau `tab` que l'on peut aussi noter `*(tab+i)`.

Par exemple, si `tab` est un tableau déclaré par `int tab[3]={ 15, 8, 23 };`, alors

- val(`tab[0]`)=val(`*tab`)=15,
- val(`tab[1]`)=val(`*(tab+1)`)=8,
- val(`tab[2]`)=val(`*(tab+2)`)=23.

Pointeurs et chaînes de caractères

En C, une chaîne de caractères est représentée par un tableau dont les éléments sont de type `char` et dont le dernier élément est `'\0'` (symbole de fin de chaîne). Une chaîne de caractères est donc manipulée comme un tableau.

Une constante littérale "chaîne de caractères" est un tableau qui a n+1 éléments où n est la longueur de la chaîne (le nombre de caractères), puisque le dernier élément de la chaîne est '\0'.

Par exemple, `char *t ="Bonjour";` est un tableau de 8 caractères : 'B', 'o', 'n', 'j', 'o', 'u', 'r', '\0'.

Passage d'arguments par adresse

En C les arguments d'une fonction sont transmis par valeur ce qui signifie que la valeur d'un argument est recopiée dans l'environnement dans lequel est évaluée la fonction.

Considérons par exemple le programme suivant :

```
int somme (int x, int y)
  {
  return (x+y);
  }
main ()
  {
  int a,b,c;
  a=5;
  b=6;
  c=somme(a,b);
  }
```

Passage d'arguments par adresse

L'état du programme immédiatement après l'appel de la fonction `somme` par la fonction `main` est le suivant

```
variable    y    6
variable    x    5
------------------------
variable    c    (-)
variable    b    (6)
variable    a    (5)
------------------------
fonction    main()
fonction    somme()
```

On voit que les valeurs de `a` et de `b` ont été recopiées au sommet de la pile dans les cases mémoires liées à `x` et à `y`.

Passage d'arguments par adresse

Voici un autre exemple :

```
void init (int p, int v)
  {
  p=v;
  }
main ()
  {
  int x=0;
  init (x,5);
  printf("La valeur de x est %d.\n",x);
  }
```

Passage d'arguments par adresse

L'état du programme immédiatement après l'appel de la fonction `init` par la fonction `main` est le suivant

```
variable    v    5
variable    p    0
------------------------
variable    x    (0)
------------------------
fonction    main()
fonction    init()
```

Puis à la fin de l'exécution de `init` on se retrouve avec

```
variable    x    0
------------------------
fonction    main()
fonction    init()
```

De sorte que l'affichage final sera `La valeur de x est 0.` et non `La valeur de x est 5.` comme on pourrait le croire.

Passage d'arguments par adresse

Le passage d'un argument par valeur souffre de deux défauts :

- la fonction ne peut pas modifier la valeur de cet argument puisqu'elle n'a accès qu'à sa copie,
- si l'argument est une structure ou un tableau, sa copie peut être coûteuse.

Pour résoudre cela, il faut utiliser un pointeur vers l'argument plutôt que l'argument lui-même. On dit que l'argument est passé par adresse.

Illustrons cela sur l'exemple précédent :

```
void init(int *p, int v)
  {
  *p=v;
  }
main ()
  {
  int x=0;
  init(&x, 5);
  printf("La valeur de x est %d.\n",x);
  }
```

Passage d'arguments par adresse

L'état de ce programme immédiatement avant l'appel de la fonction `init` est :

```
variable    x    0
---------------------------
fonction    main()
fonction    init()
```

Passage d'arguments par adresse

L'état de ce programme immédiatement après l'appel de la fonction `init` :

```
variable    v    5
variable    p    &x
---------------------------
variable    x    0
---------------------------
fonction    main()
fonction    init()
```

Passage d'arguments par adresse

L'état de ce programme immédiatement après le retour à la fonction principale :

```
variable    x    5
-----------------------------
fonction    main()
fonction    init()
```

Avec ce passage par adresse, l'affichage final est `La valeur de x est 5.` comme souhaité.