Algorithmique et Structures de données
Pointeurs en C

Abdelkamel, Ben Ali

Université Echahid Hamma Lakhdar - El Oued

Décembre 2020

Licence 1 MI

Notion de pointeur

- **Rappel**. Une variable désigne un "tiroir" dans la mémoire, dans lequel on peut ranger une valeur.
- Un **pointeur** est aussi un tiroir, mais il contient une référence à un autre tiroir.
- Le numéro d'un tiroir est appelé l'adresse –de la variable. Si la variable a correspond à tel tiroir, et ce tiroir est le tiroir n° 3200, un pointeur p qui fait référence à a sera un tiroir qui contiendra le numéro 3200.

- Le pointeur p contient la valeur 3200, qui est l'adresse de la variable a.
- On dit que p pointe sur a

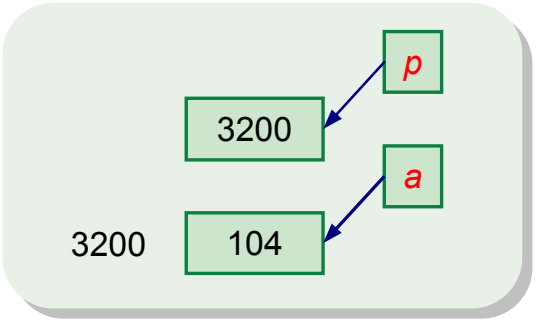
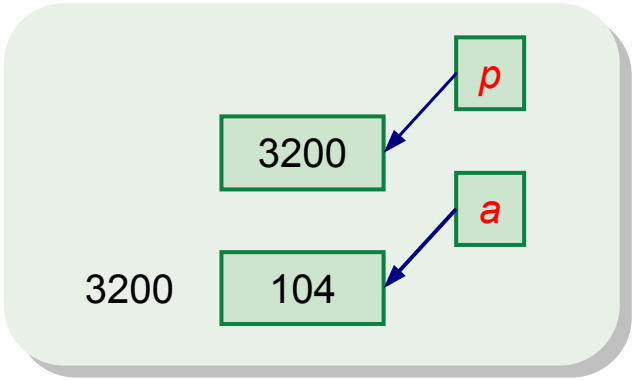


Figure: Une variable et un pointeur

Notion de pointeur



- On utilise les notations suivantes :
  - `a` pour désigner le contenu de (du tiroir) `a` (104)
  - `p` pour désigner le contenu de (du tiroir) `p` (3200)
  - `&a` pour désigner l'adresse du tiroir `a` (3200)
  - `*p` pour désigner le contenu du tiroir sur lequel pointe `p` (104)

Passage par adresse

- Utilisation des pointeurs pour le passage des paramètres par adresse.

  Considérons les procédures données en Alg. 1, et l'exécution de la procédure principal `main` (Fig.).

Alg. 1 – Passage d'un paramètre par valeur

```
void ajoute_un(int a)
{
    a = a + 1;
}

main()
{
    int b;
    b = 3;
    ajoute_un(b);
    printf("%d", b);
}
```

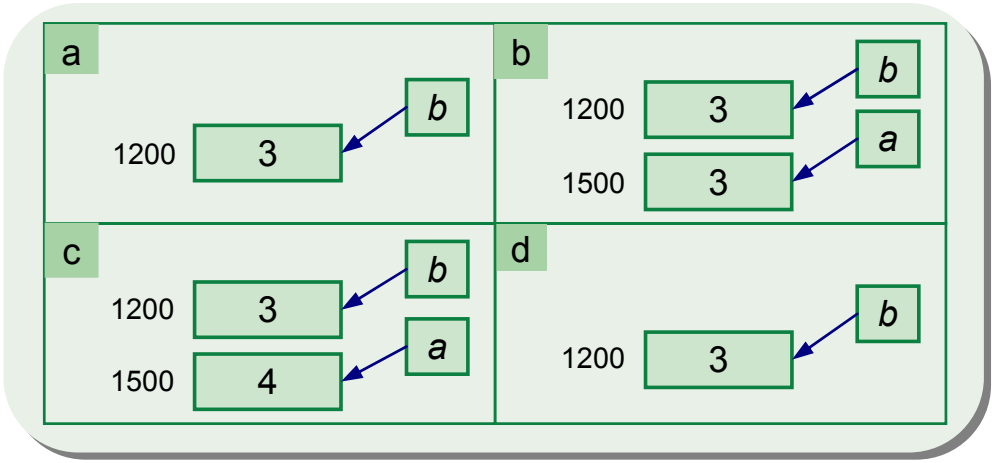


Figure: Exécution de la procédure principal `main` de Alg. 1

Passage par adresse

- Dans certains cas, nous souhaitons que les paramètres soient effectivement modifiés lors d'un appel de sous programme.
- Pour rendre ceci possible, la solution serait, plutôt que de communiquer la valeur, de communiquer l'adresse.
- Voici les nouvelles procédures . . .

Alg. 2 – Passage d'un paramètre par adresse

```
void ajoute_un_bis(int *p)
{
    *p = *p + 1;
}

main()
{
    int b;
    b = 3;
    ajoute_un_bis(&b);
    printf("%d", b);
}
```

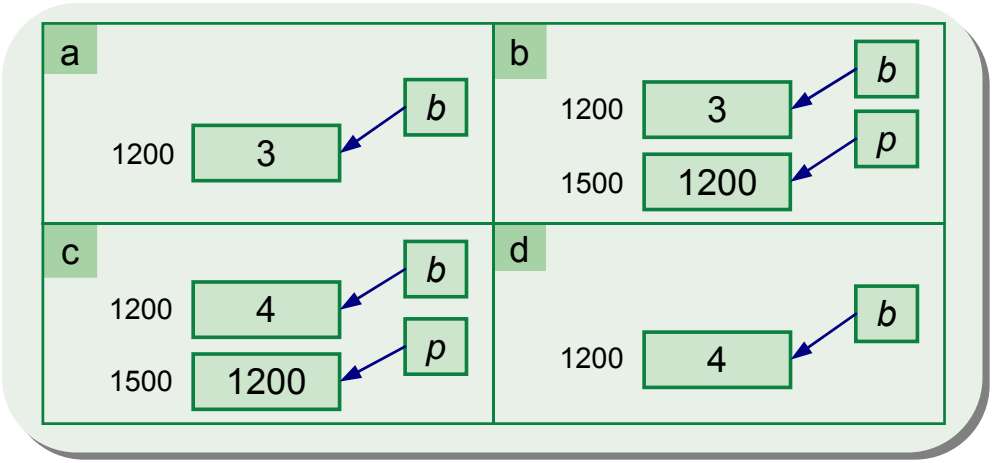


Figure: Exécution de la procédure principal `main` de Alg. 2

Passage par adresse

- le passage par adresse permet de modifier les variables du (sous) programme
  - . . .c'est l'adresse de la variable, plutôt que sa valeur qui est transmise au sous programme.
- pour le passage par adresse, il y a deux changements à opérer par rapport au passage par copie :
  - lors de l'appel, on met le signe **&** devant le paramètre
  - dans la fonction/procédure, on met le signe * devant le nom du paramètre.

A retenir Les tableaux sont toujours passés par adresse, et ceci sans notation particulière (pas de & et pas de *).

Exercice 1 – Fonctions

1. Qu'affiche le programme suivant ? Expliquer.

```
void ma_procedure(int x)
{
    int y;
    x = 0;
    y = -1;
}

void main()
{
    int x = 22;
    int y = 23;
    ma_procedure(x);
    printf("x=%d␣␣y=%d\n", x, y);
}
```

Exercice 1 – Fonctions

2. Donner le prototype d'une fonction nommée `EstPremier` testant si un nombre n est ou non premier.
3. Donner le prototype d'une fonction `DemandePoint` qui demande à l'utilisateur les coordonnées (x, y) d'un point (on ne demande pas d'écrire le code de la fonction).
4. Que fait la fonction C suivante ?

```
void Mystere(int *dp, int *sp)
{
        if (*sp > 0) {
                *dp = *sp;
        }
        else {
                *dp = 0;
        }
}
```

Exercice 1 – Fonctions

5. Dans le programme suivant, rayez les lignes illégales (erreurs). Qu'est ce qui s'affiche ?

```
void main(void)
{
    int i = 0;
    int *p;
    float x = 3.14;
    float *f;

    p = &i;
    *f = 666;
    f = &x;
    *f = *p;
    *p = 34;
    p = f
    *p = *p + 1;
    printf("%d␣␣%f\n", i, *f);
}
```

Exercice 2 – Passage par adresse

1. Ecrire une fonction qui échange le contenu de deux variables entières. Exemple d'utilisation :

```
...
int a = 10, b = 22;
Echanger(&a, &b);
/* ici a == 22 et b == 10 */
```

2. En utilisant la fonction `Echanger`, écrire une fonction `Permute3` qui effectue une permutation circulaire de trois variables, comme dans l'exemple ci-dessous :

```
int a = 10, b = 22, c = 33;
Permute3(&a, &b, &c);
/* ici a == 33, b == 10, c == 22 */
```

3. Ecrire une fonction `DivisionEuclidienne` qui donne le quotient et le reste de la division de deux nombres entiers. On utilisera les opérateurs / et % du langage C.

Exercice 3 – Tableaux

On désire écrire une fonction nommée `ElimineDupliques` qui modifie un tableau d'entiers pour en éliminer les éléments identiques consécutifs. Cette fonction s'utilisera comme dans l'exemple :

```
...
int T[] = {1, 2, 2, 3, 3, 3, 4, 6, 6, 9};
int N = 10; /* Nombre d'éléments de T */
int U[10];  /* pour le résultat */
int Nu;     /* Nombre d'éléments placés dans U */

ElimineDupliques(T, N, U, &Nu);

/* ici U contient { 1, 2, 3, 4, 6, 9 }, et Nu == 6 */
```

1. Pourquoi passe-t-on N par valeur, mais Nu par adresse ? En déduire le prototype de la fonction `ElimineDupliques`.
2. Ecrire le corps de `ElimineDupliques`.
3. Pouvez-vous changer l'algorithme de `ElimineDupliques` de façon à ce que le tableau T soit modifié en place (pour se passer du tableau U) ?