I2010 - Langage C

Chap. 7:
Les fonctions

Déclarer une fonction

- Déclarer le **prototype**

```
type fct (type1, type2, … );
```

- Exemple

```
int search (int*, int, int);
```

Définir une fonction

- Type + paramètres + code

```
int search (int* t, int sz, int e) {
  int i = 0;
  while (i < sz && t[i] != e) {
    i++;
  }
  return i;
}
```

Type

- Une fonction retourne une **valeur** ou rien (**void**)

```
double sum (double a, double b) {
    return (a+b);
}

void itoa (int a) {
    printf("%d", a);
    return;   // (optionnel)
}
```

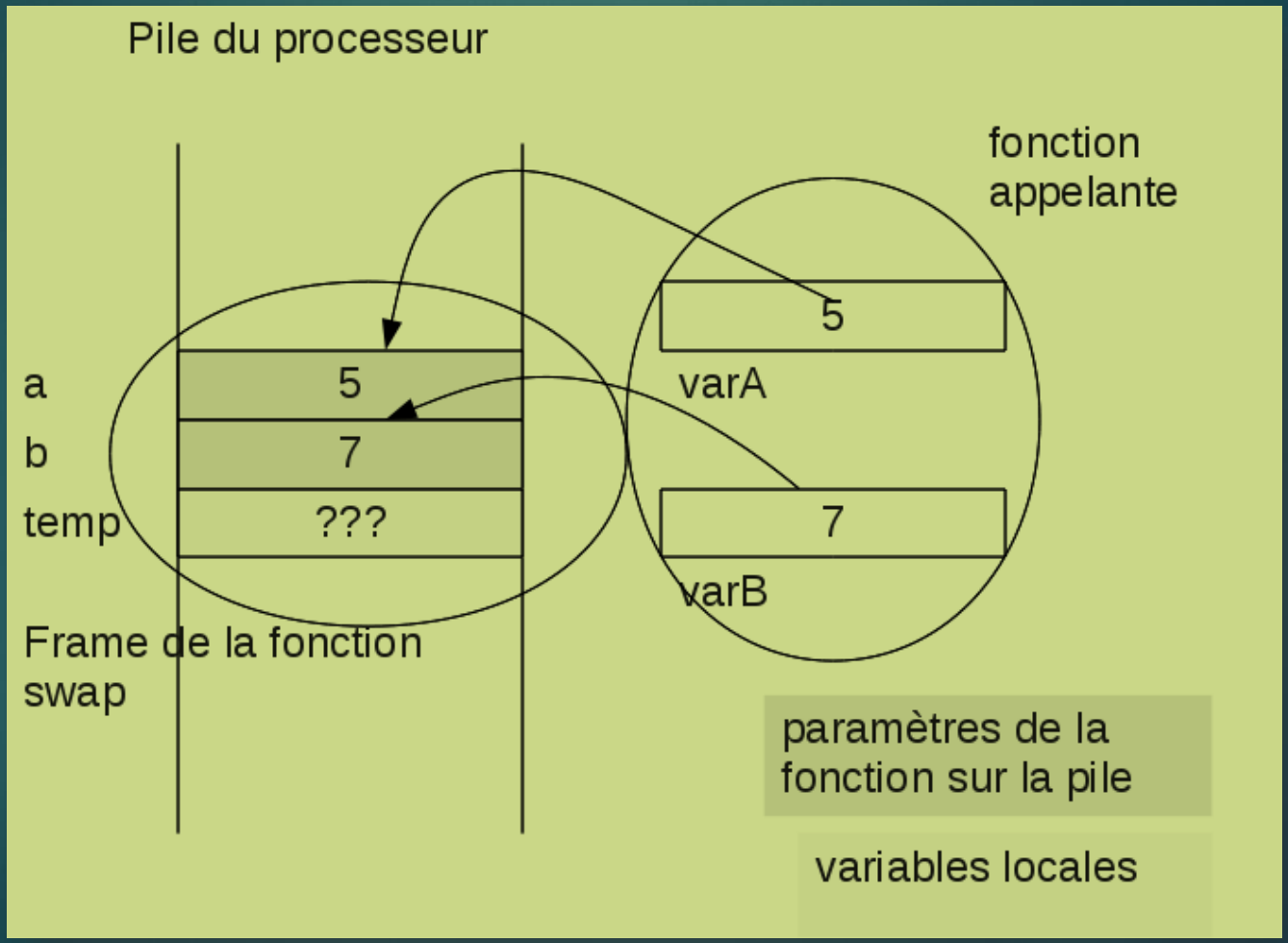
Type

- C n'exige pas de **`return`** mais le résultat peut être **indéterminé** !

```
int divint (int a, int b) {
    if (b != 0) {
        return (a/b);
    }
}
```

Paramètres

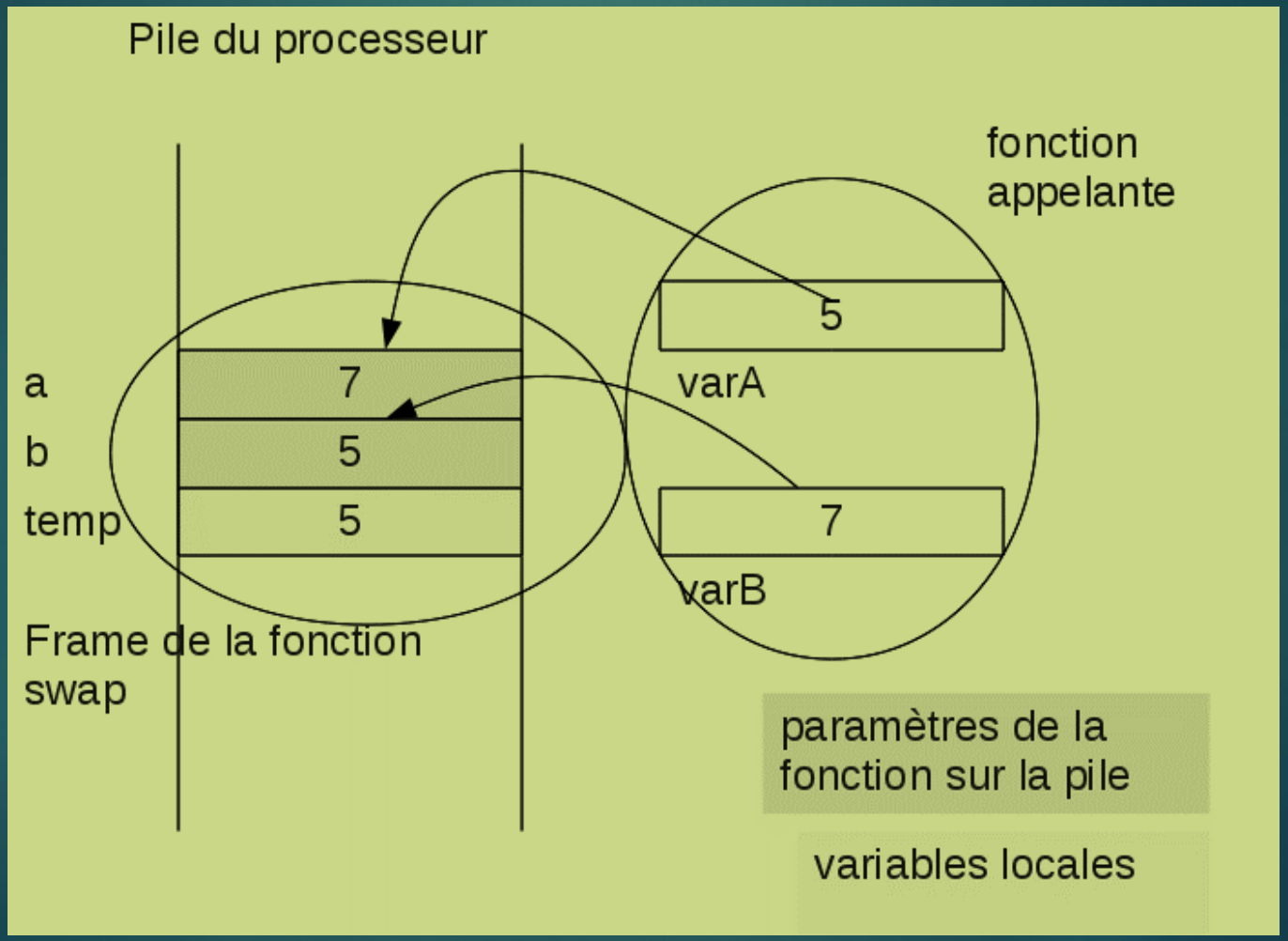
Passage par **valeur**



Permuter deux valeurs

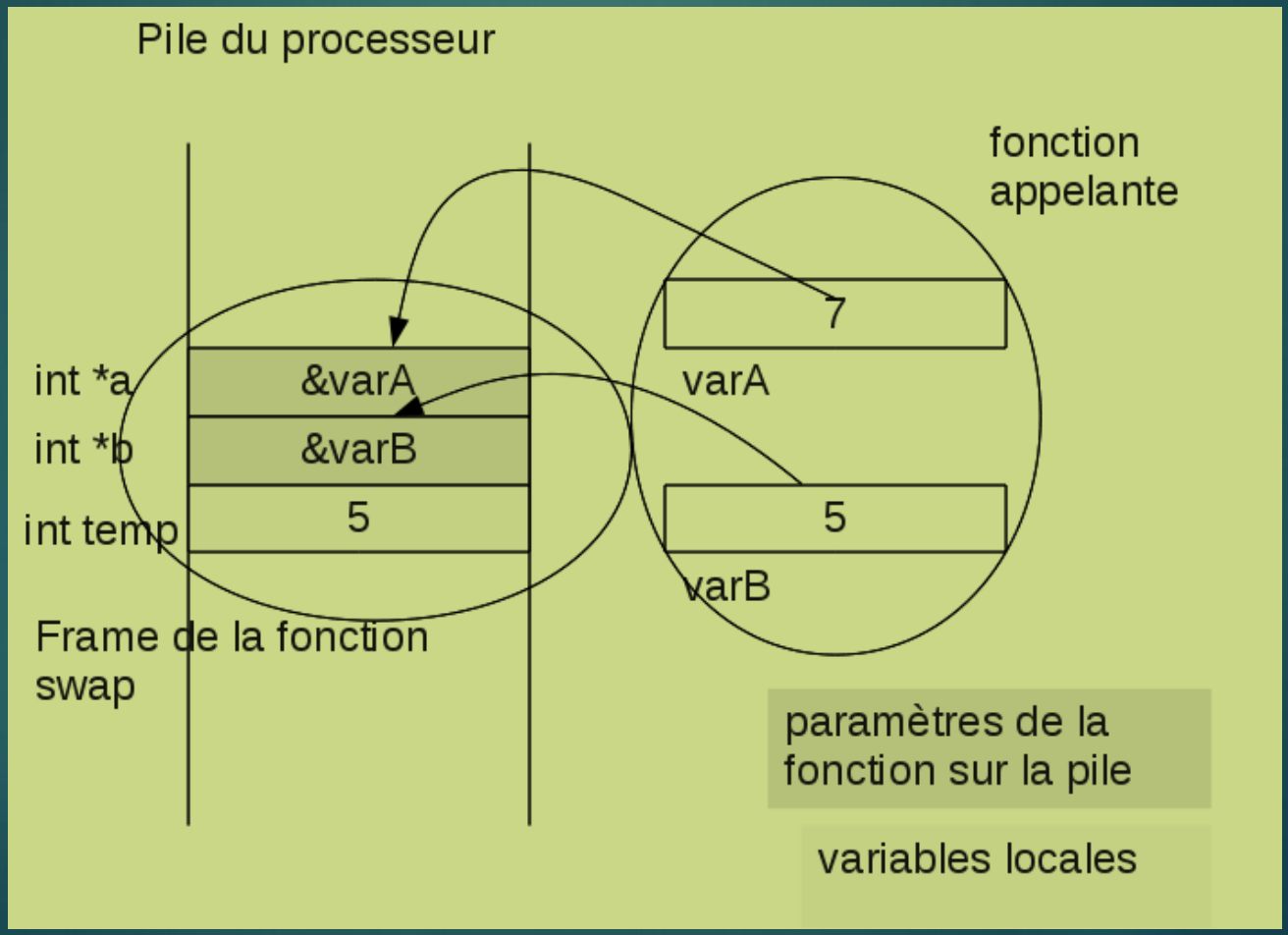

```
void swap1 (int a, int b) {
    int temp = a;
    a = b;
    b = temp;
}
```

swap1 est **incorrect**



Paramètres et pointeurs

Par **valeur**, mais on peut passer des **pointeurs**



Permuter deux valeurs (correct)


```
void swap2 (int *a, int *b) {
    int temp = *a;
    *a = *b;
    *b = temp;
}
...
swap2(&varA, &varB);
```

Fonctions et tableaux

- C convertit systématiquement les **tableaux** passés en paramètres à une fonction en **pointeurs**

```
void initTable (int *tab, int taille){
  for (int i=0; i != taille; i++) {
    t[i] = i;
  }
}
```

Fonctions et tableaux

- Une fonction peut renvoyer un **tableau** mais il doit être alloué **dynamiquement**

```
int* creerTable (int taille) {
   int* tab = (int*) malloc(taille*sizeof(int));
   for (int i=0; i<taille; i++)
      tab[i]=i;
   return tab;
}
...
int* t = creerTable(10);
```

Spécifications

- Les spécifications de fonctions sont des règles qui déterminent quels sont les **conditions** d'utilisation et les **résultats** de cette fonction.
- Ces règles forment un contrat qui précise les **responsabilités** entre le client (le programme appelant) et le fournisseur d'un morceau de code logiciel (la fonction).

Spécifications

- **Précondition:** phrase logique qui décrit les hypothèses sur l'état du programme au moment d'appeler la fonction.

  Si la précondition n'est pas vérifiée, le résultat de la fonction est indéterminé.
- **Postcondition:** phrase logique qui décrit comment la fonction a modifié l'état du programme.
- **Résultat:** le résultat renvoyé par la fonction.

Spécifications

```
int search (int* t, int sz, int e);
```

- **PRE:** `t` : tableau non-null de longueur `sz`
- **POST:** `t` n'est pas modifié
- **RES:** Si `e` est un élément de `t`, alors un entier `idx` est renvoyé de telle sorte que `t[idx] == e` ; sinon `sz` est renvoyé.

Spécifications

```
void print (int* t, int sz);
```

- **PRE:** `t` : tableau non-null de longueur `sz`
- **POST:** `t` n'est pas modifié et une ligne représentant le contenu de `t` est affichée sur stdout

Spécifications

Par défaut, on supposera que:

- **tous les paramètres pointeurs sont non-nuls**
- **les paramètres ne sont pas modifiés**

```
void print (int* t, int sz);
```

- **PRE:** `t` : tableau ~~non-null~~ de longueur `sz`
- **POST:** ~~`t` n'est pas modifié~~ et une ligne représentant le contenu de `t` est affichée sur stdout

Spécifications

Par défaut, on supposera que:

- **tous les paramètres pointeurs sont non-nuls**
- **les paramètres ne sont pas modifiés**

```
void print (int* t, int sz);
```

- **PRE:** `t` : tableau de longueur `sz`
- **POST:** une ligne représentant le contenu de `t` est affichée sur stdout