Programmation « orientée système »

APPROFONDISSEMENTS SEMAINE 6

Jean-Cédric Chappelier

Laboratoire d'Intelligence Artificielle
Faculté I&C

©EPFL 2022
Jean-Cédric Chappelier
EPFL

Rappels des points clés

1) ref
2) généric
3) alloc. dyn.

- (encore et toujours) Les 3 cas d'utilisation des pointeurs
- Stack/Heap

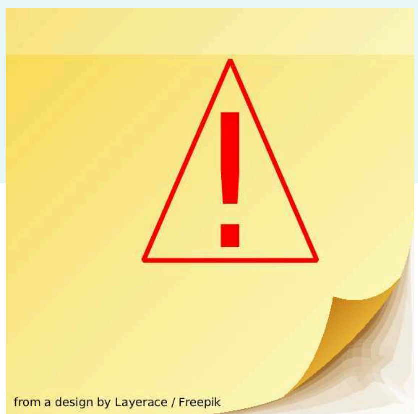
int tab[var];

- On n'utilise l'allocation statique (stack) que si ces **trois** conditions sont remplies :
  - on connait la taille (ou un majorant) *a priori*
  - cette taille (ou majorant) est « petite » (max. quelques Mo)
  - cette taille (ou majorant) ne varie pas

  Dans **tous** les autres cas : allocation *dynamique* (heap)
- Tout `malloc()`/`calloc()` doit avoir son `free()` (mais pas dans la même portée !)
- Préférez `calloc()` à `malloc()`
- Conseil : mettre `p = NULL;` après un `free()`
- ...et tous les autres conseils du slide 14 !

©EPFL 2022
Jean-Cédric Chappelier
EPFL

Récapitulons : règles de bon usage



Si vous suivez ces règles, vous vous faciliterez la vie de programmeur avec pointeurs :

- Toute zone mémoire allouée dynamiquement (`malloc`) doit **impérativement** être libérée par un `free` correspondant !

  et c'est celui qui alloue qui doit libérer

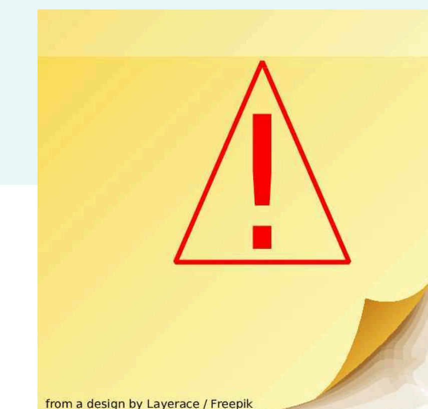
  Corollaire : si vous fournissez une fonction qui alloue de la mémoire vous **devez** fournir une fonction « réciproque » qui libère la mémoire, de sorte que celui qui appelle votre première fonction puisse respecter la règle ci-dessus (en appelant la seconde fonction)

- Testez systématiquement vos `malloc`/`calloc` :

```
pointeur = calloc(nombre, sizeof(type));
if (pointeur == NULL) {
  /* ... gestion de l'erreur ... */
  /* ... et sortie (return code d'erreur) */
}
/* suite normale */
```

©EPFL 2023
Jean-Cédric Chappelier
EPFL

Récapitulons : règles de bon usage



Si vous suivez ces règles, vous vous faciliterez la vie de programmeur avec pointeurs :

- Pour les allocations multiples, utilisez `calloc` et non pas `malloc`
- Initialisez toujours vos pointeurs.
  Utilisez `NULL` si vous ne connaissez pas encore la mémoire pointée au moment de l'initialisation
- Initialisez toujours **toute** la mémoire utilisée (`memset`).
- ajoutez un `ptr = NULL;` après chaque `free`
- utilisez toujours `const` dans vos « faux » passages par référence (optimisation)
- utilisez les outils supplémentaires de votre environnement de développement : options du compilateur, debugger, programmes de surveillance de la mémoire (e.g. `valgrind`, `Address Sanitizer`, ...), programmes de recherche de bugs (`scan-build`, `splint`, `flawfinder`, ...)

©EPFL 2023
Jean-Cédric Chappelier
EPFL

Etudes de cas

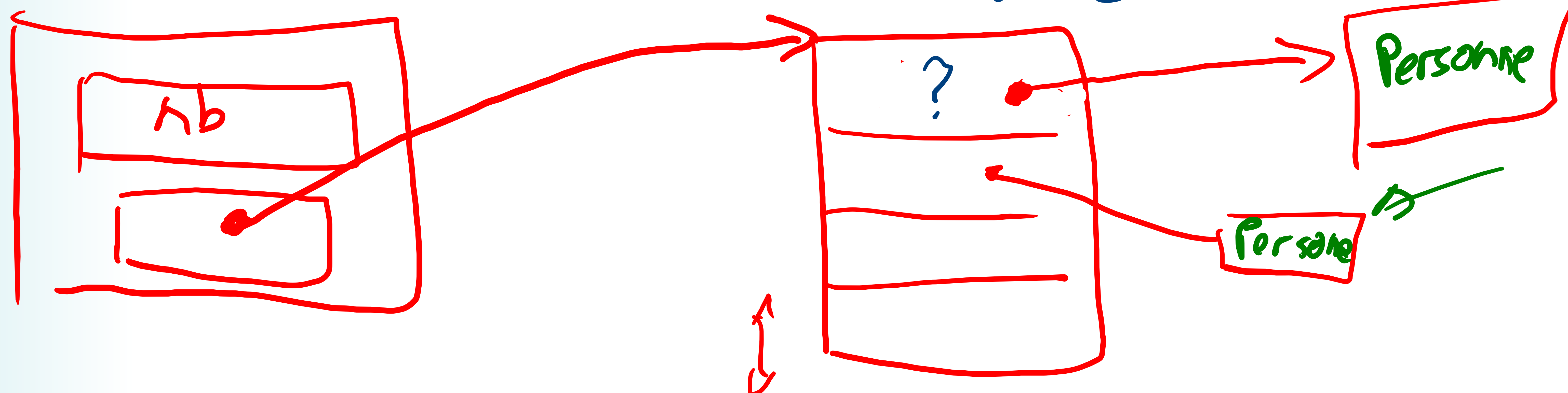
- reprendre en détail l'un ou l'autre exemple du cours ?

  (travaillez bien le « pattern » « tableau dynamique » ; c'est archi-classique et très formateur à mon avis)

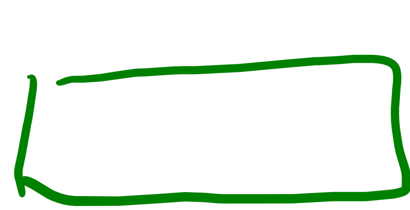
- voiture avec chauffeur : version dynamique :
  - générateur (« *factory* ») : retourner un pointeur
  - passage de pointeurs par référence : pointeur de pointeur

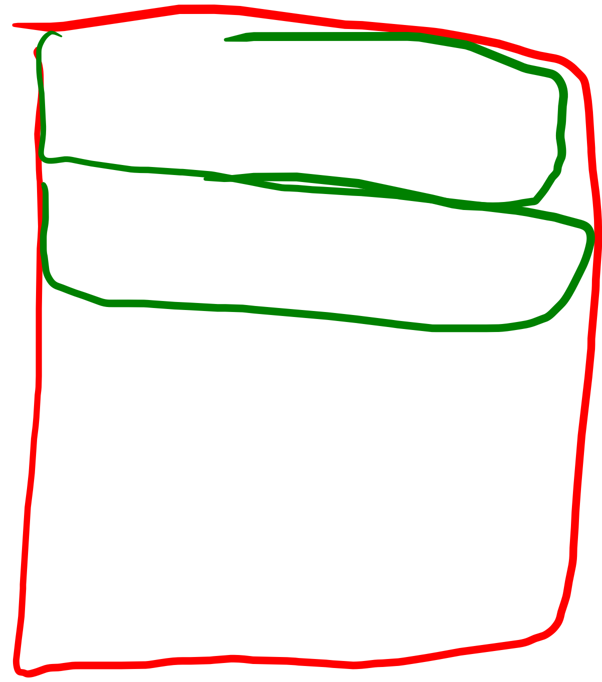
cas #3

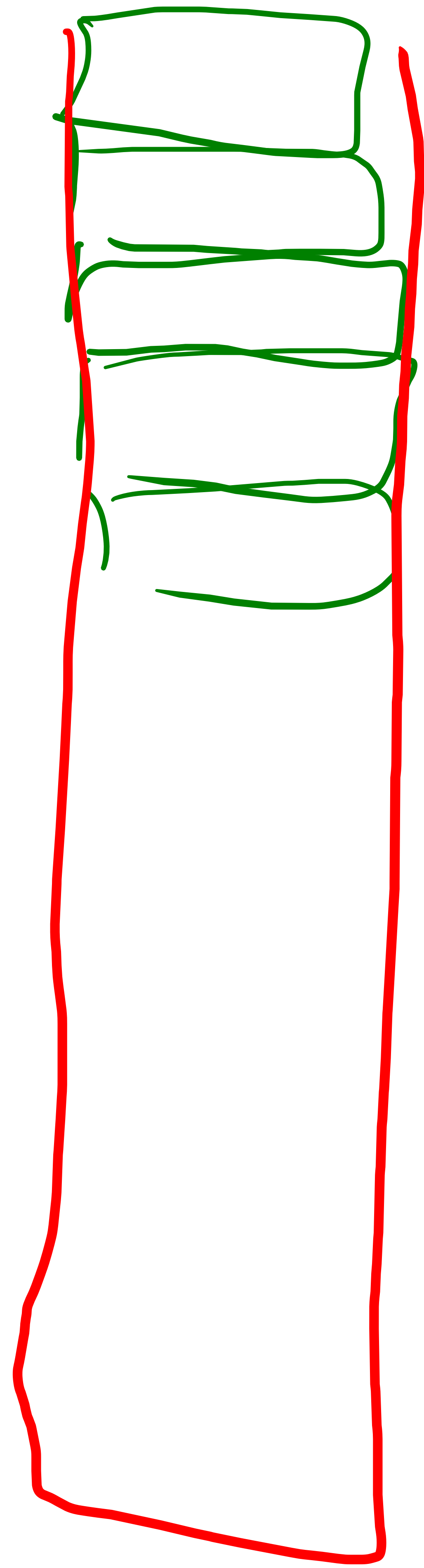
Peuple : ens. de personnes



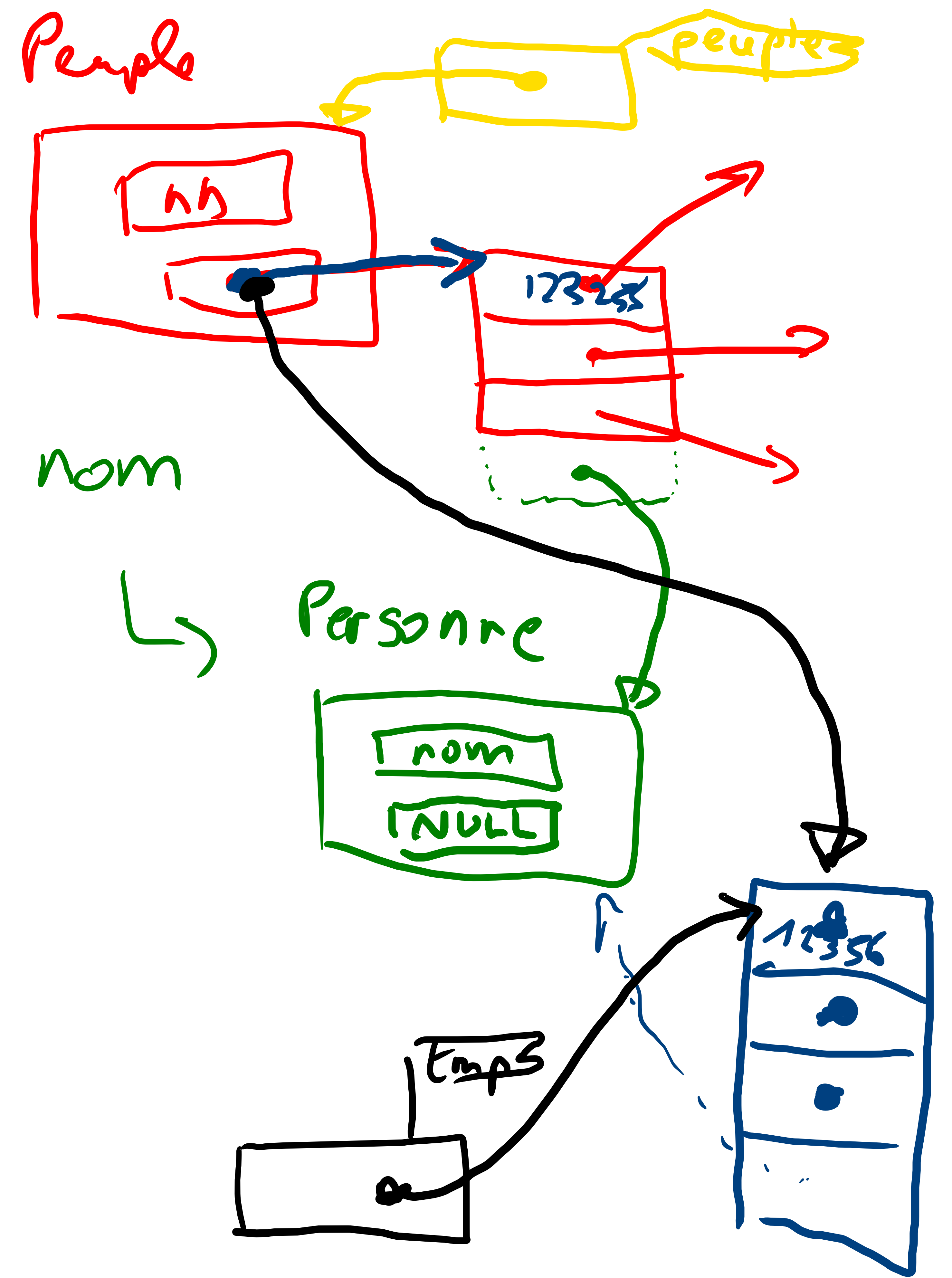
©EPFL 2022
Jean-Cédric Chappelier
EPFL







X2



$a * b$ n'overflow pas

$$a * b > MAX$$

$$a > MAX / b$$