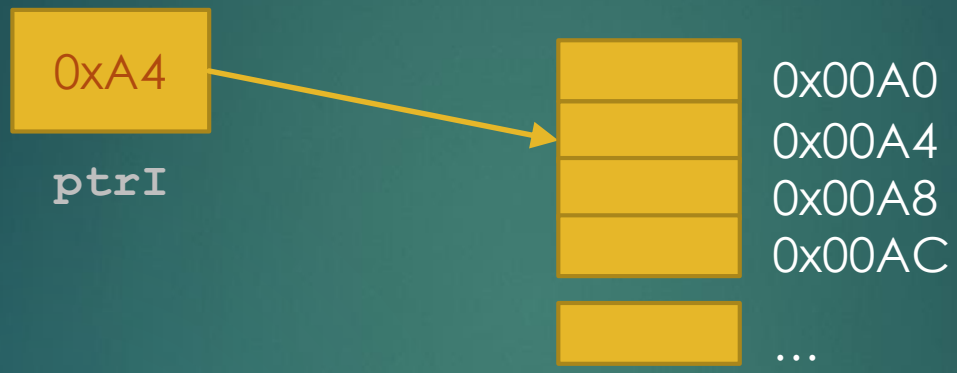
# I2010 - Langage C

# Chap. 4:
# Les pointeurs

# Les pointeurs



- Les pointeurs contiennent des adresses mémoires




- Les pointeurs sont typés

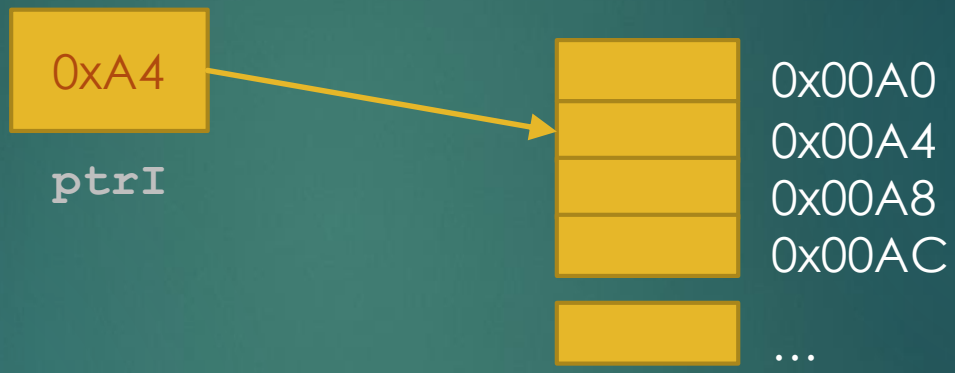
```
int *ptrI;   // pointeur vers un entier
double *ptrI; // pointeur vers un double
Noeud *ptrN;    // pointeur vers un Noeud
```

- Un peu comme une référence Java mais…

# Ça sert à quoi?



- Accéder une adresse précise (driver, mémoire vidéo…)




- Allouer dynamiquement de la mémoire
- Créer des structures de données chaînées
- Passer des paramètres par référence

# Initialisation



- Pas initialisé par défaut ⇒ Danger!
- Prendre l'adresse d'une variable

```
int a = 38;

int *ptrI = &a;  // reçoit l'adresse de a
```

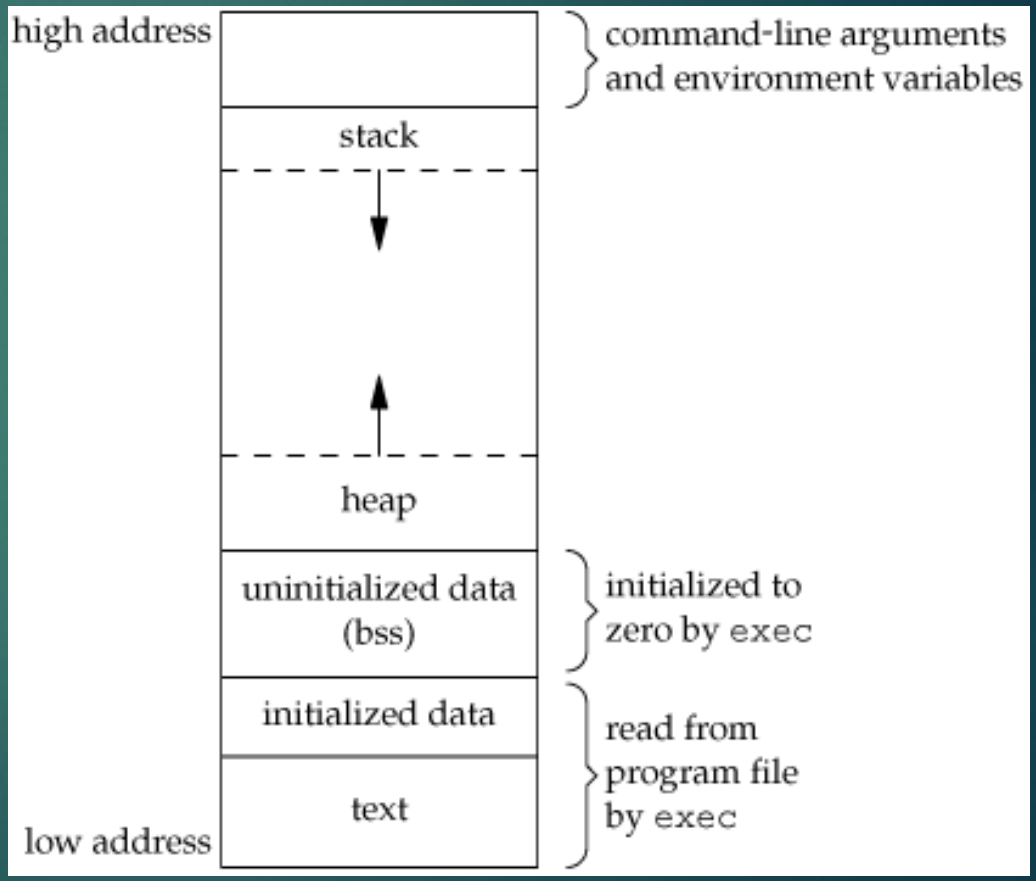

- Vous avez déjà utilisé le '&'. Où ça?

# Initialisation



```
int truc(){
    int a = 38;
    int *ptrI = &a;
}
```

- *a* est sur le stack



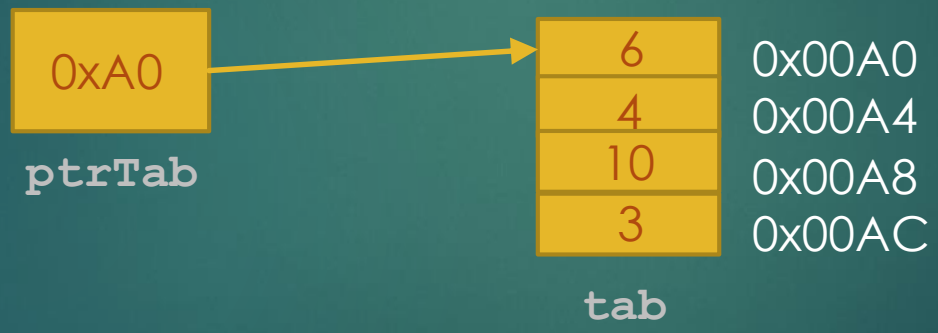
# Initialisation



- Prendre l'adresse d'un tableau

```
int tab[4] = {6, 4, 10, 3};

int *ptrTab = tab;  // adresse de l'élément 0
```

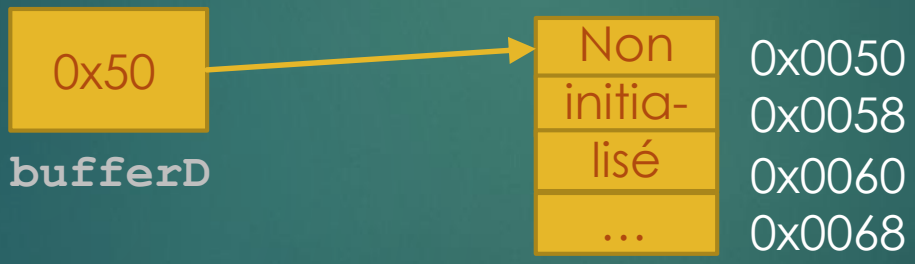


# Mémoire dynamique



- Allouer une zone mémoire de taille quelconque

```
double *bufferD =
      (double*)malloc(4*sizeof(double));
```




- Remarquez les +8 dans les adresses. Pourquoi?

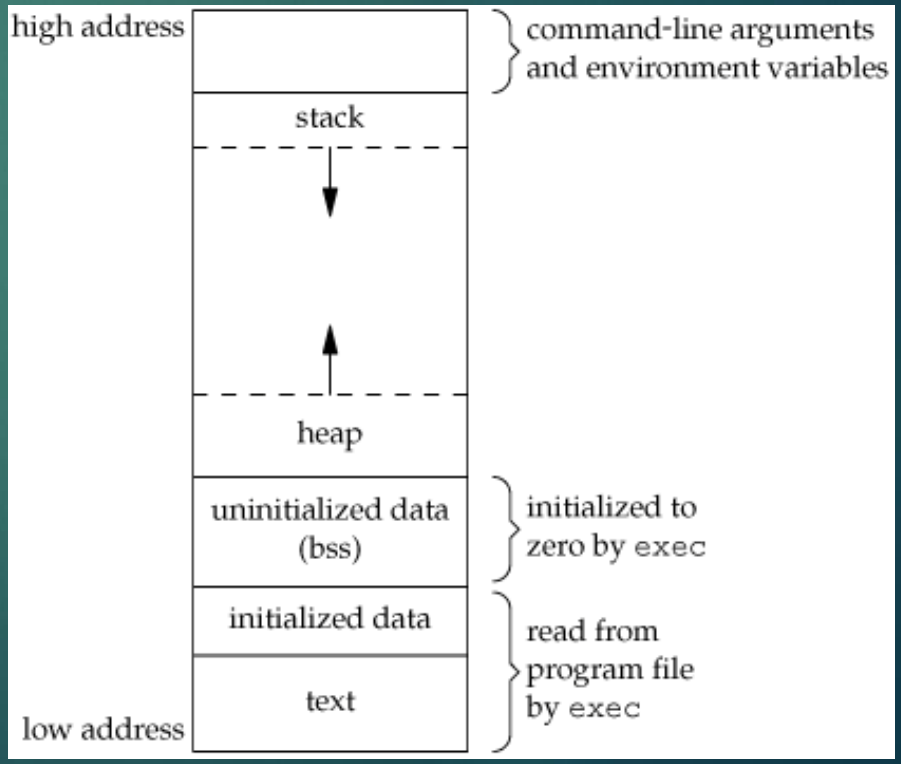
# Mémoire dynamique



- Allouer une zone mémoire de taille quelconque

```
double *bufferD =
        (double*)malloc(4*sizeof(double));
```

- *malloc* travaille sur le heap



# Mémoire dynamique



- Si à court de mémoire

```
double *bufferD =
   (double*)malloc(4*sizeof(double));

if (bufferD == NULL) ...
```

# Mémoire dynamique



- Pas de garbage collector
- Libérez la mémoire!

```
double *bufferD =
     (double*)malloc(4*sizeof(double));

...

free(bufferD);
```

# Déréférencement

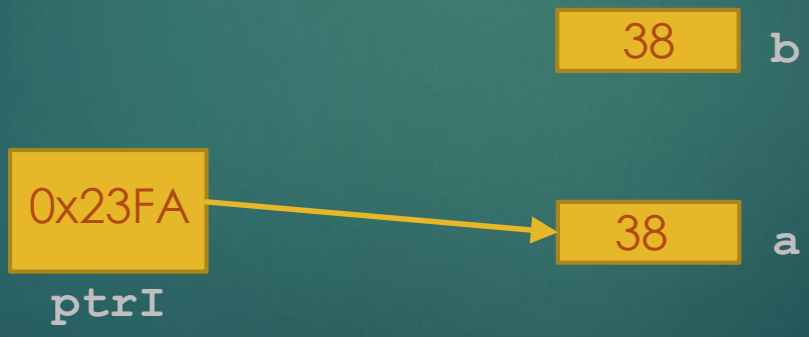


- Accéder au contenu de l'élément pointé

```
int a = 38;

int *ptrI = &a;  // reçoit l'adresse de a

int b = *ptrI;  // déréférencement: prend le
                // contenu de l'élément pointé
```



# Déréférencement



- Modifier le contenu de l'élément pointé

```
int a = 38;

int *ptrI = &a;  // reçoit l'adresse de a

*ptrI = 27;  // modifie le contenu de a!
```

# Déréférencement



```
double *buffer =
     (double*)malloc(4*sizeof(double));

...

double db = buffer[1]; // déréférencement: accès
                       // au contenu de l'élément 1
```

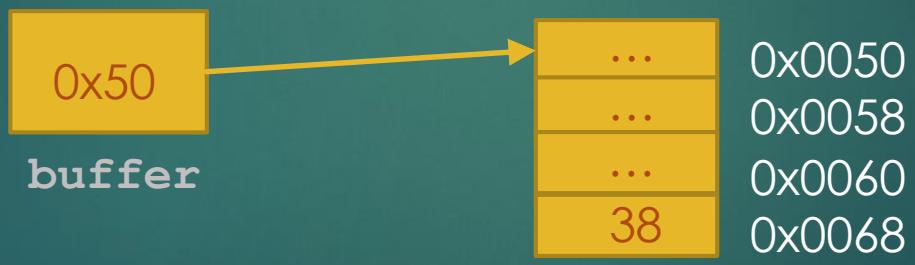
# Déréférencement



```
double *buffer =
    (double*)malloc(4*sizeof(double));

...

buffer[3] = 38;  // modifie l'élément 3
```

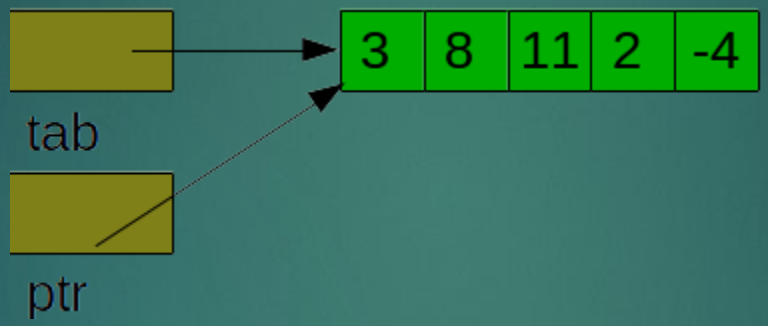


# Arithmétique des pointeurs



```
int tab[5] = {3, 8, 11, 2, -4};

int *ptr = tab;
```




```
ptr++;
```