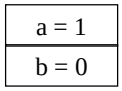
Les pointeurs

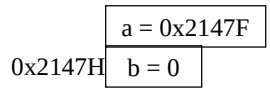
Le passage d'arguments par adresse

```
void passageParValeur( int a ){
      a = 1 ;
}
void main(){
      int b = 0 ;
      f( b ) ;
      /* b = 0 */
}
```



- a est une copie de b.

```
void passageParAdresse( int* a ){
      *a = 1 ;
}
void main(){
      int b = 0 ;
      f( &b ) ;
      /* b = 1 */
}
```



- a est un pointeur qui contient l'adresse de b.

Passer un tableau à une fonction

- Passage d un tableau en lecture écriture :

```
void fonction( int* tab, int taille ){     /* <=> void fonction( int tab[], int taille ){ */
    tab[ 1 ] = 0 ;                         /* <=> *(tab+1) = 0 ; */
}
void main(){
    int tab[ 10 ] ;
    fonction( tab ) ;                      /* <=> fonction( &tab[0] ) ; */
}
```

- Passage d'un tableau en lecture :

```
void fonction( const int* tab, int taille ){
    tab[ 1 ] = 0 ;                         /* erreur à la compilation !! */
}
void main(){
    int tab[ 10 ] ;
    fonction( tab ) ;
}
```


Retourner un tableau d'une fonction

```
int* fonction(){
    int tab[ 10 ] ;
    return tab ;
}          /* tab est détruit ici ! */
void main(){
    int* t = fonction() ;
}
```

- **Règle** : ne jamais retourner l'adresse d'une variable locale.



- t pointe sur un tableau qui n'est plus réservé !

- Allocation dynamique :

```
void *malloc( size_t size ) ;
```



```
int* fonction( int taille ){
    int* tab = (int*)malloc( taille * sizeof( int ) ) ;
    return tab ;
}
void main(){
    int* t = fonction() ;
    if( t != NULL ){
        t[ 2 ] = -1 ;
    }
}
```

Les pointeurs de fonctions

- Les pointeurs de fonctions permettent de réaliser des appels de fonctions par valeur :

```
void fonction( int i, int j ){
    /* ... */
}
void main(){
    void (*ptrFonction)( int, int ) ; /* déclaration d'un pointeur de fonction */
    ptrFonction = fonction ;          /* initialisation du pointeur */
    ptrFonction( 3, 2 ) ;             /* appel par pointeur */
}
```


Les tableaux de pointeurs de fonctions 1/2

```
int plus( int x, int y ){
    return x + y ;
}
int moins( int x, int y ){
    return x - y ;
}
void main(){
    int (*ptrFonction[])( int, int ) = { plus, moins } ;    /* tableau de pointeurs */
    int i, j ;
    char op ;
    scanf( "%d%c%d", &i, &op, &j ) ;
    switch( op ){
        case ' + ' :    ptrFonction[ 0 ]( i, j ) ;    /* appel de la fonction plus*/
                        break ;
        case ' - ' :    ptrFonction[ 1 ]( i, j ) ;    /* appel de la fonction moins*/
                        break ;
        default :       break ;
    }
}
```

Utilisation d'une fonction générique 2/2

Fonction pouvant appeler n'importe quelle fonction recevant 2 entiers et en retournant un.

```
/* ...* /
void fonctionGenerique( int (*ptrFonction)( int, int ), int i, int j ){
    ptrFonction( i, j ) ;      /* appel de plus ou moins */
}
void main(){
    int (*ptrFonction[])( int, int ) = { plus, moins } ;
    int i, j ;
    char c ;
    scanf( "%d%c%d", &i, &op, &j ) ;
    switch( op ){
        case ' + ' :        fonctionGenerique( ptrFonction[ 0 ], i, j ) ;
                            break ;
        case ' - ' :        fonctionGenerique( ptrFonction[ 1 ], i, j ) ;
                            break ;
        default :           break ;
    }
}
```