

EXEMPLE 1

n)

issant ?

EXEMPLE 2

PROBLÈME

```
void tri_decroissant(int *tab, int length)
{
    int i, j;
    for (i = length - 1; i > 0; i--)
        for (j = 0; j < i - 1; j++)
            if (tab[j] < tab[j + 1])
            {
                int temp = tab[j];
                tab[j] = tab[j + 1];
                tab[j + 1] = temp;
            }
}
```

POINTEUR

- Permet de référencer directement une variable
- Permet de passer une fonction en argument
- Permet d'associer une fonction à une variable

nd à un pointeur statique

a fonction conduit à son
onnel, mais recommandé

UR DE FONCTION

type précis dépendant du
ée

pe de retour et au nombre
nts de la fonction

e fonction s'effectue ainsi:

```
ct);  
des paramètres de la  
ter
```

RÉCUPÉRATION DE L'A

- Soucis pointeur générique
 - bien que gérée par certains compilateurs, elle n'est pas portable
 - il n'est pas possible de déclarer un pointeur générique car sans type de données en mémoire à récupérer
 - on ne peut donc pas non plus déclarer un pointeur sur ce type de pointeur (pointeur de pointeur)

EXE

- Déclaration de pointeurs

```
void foo_1(void) {...}  
void foo_2(int i){...}  
void foo_3(int i, char * s, size_t n){...}  
  
int main(void) {  
    void (*ptr_1)(void);  
    void (*ptr_2)(int);  
    void (*ptr_3)(int, char *, size_t);  
    ptr_1=&(foo_1);  
    ptr_2=&(foo_2);  
    ptr_3=&(foo_3);  
    printf("%p - %p - %p\n", ptr_1, ptr_2, ptr_3);  
    return 0;  
}
```

COURS DE FONCTIONS

On est une adresse, il est
dans un tableau

un tableau d'un type "simple"
déclaration d'un tableau de
généralement sous la forme

```
...[LLE])(l_types_fct);
```

déclaration d'un pointeur de
la taille qui fait toute la

TEUR DE FONCTION

ourner un pointeur de

envoyant le pointeur (disons
pointées s'entremêlent ainsi :

```
...o(l_args_foo))  
_fct);
```

paramètre une liste

) et retourne un pointeur
otype est type_ret_fct

TABLEAU DE POINT

- Ces tableaux sont soumis à
que les tableaux "standard"
 - indice commence à 0
 - un tableau ne connaît pas
 - Tous les éléments conten
être de même type – les
adresses devront avoir u
type de retour et mêmes

EXE

```
void foo_1(void){ printf(", World");  
int foo_2(char * s){ return printf("%s", s);  
void (* foo_3(int i))(void){  
    printf("%d\n", i);  
    return &(foo_1);  
}  
int (* foo_4(int i))(char *){  
    printf("%d\n", i);  
    return &(foo_2);  
}  
int main(void){  
    void (*ptr_1)(void)=foo_3(1);  
    void (*ptr_2)(char *)=foo_4(2);  
    (*ptr_2)("Hello");  
    (*ptr_1)();  
    return 0;  
}
```

Que ce co

UTILITÉ: EXEMPLE

complémentées comme suit et
pointeurs de fonctions

```
= {&(add), &(subtract),
```

RETOUR SUR L'

- Le choix de l'opérateur

```
double (* selectOperation(char choice)
{
    int i=-1;
    switch(choice){
        case '+': i=0; break;
        case '-': i=1; break;
        case 'x': i=2; break;
        case '/': i=3; break;
        default: return NULL;
    }
    return operations[i];
}
```

UTILITÉ: EXEMPLE

tions manquent pour que le

```
;
```

CONC

- Les pointeurs de fonctions offrent une certaine flexibilité dans le code
- La déclaration de type, variable, utilisant un pointeur de fonction, doit être du même type de fonction correspondant
- Les erreurs sont faciles et courantes, il est donc nécessaire une grande vigilance