EXEMPLE 1

```
n)
```

issant ?

PROBLÈME

```
void tri_decroissant(int *tab, int
{
    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;
            }
        }
    }
}
```

EXEMPLE 2

```

```

POINTEUR

- Permet de référencer dir
- Permet de passer une fon
- Permet d'associer une for

nd à un pointeur statique

a fonction conduit à son
onel, mais recommandé

RÉCUPÉRATION DE L'A

- Soucis pointeur générique
  - bien que gérée par certa
    portable
  - il n'est pas possible de de
    générique car sans type
    mémoire à récupérer
  - on ne peut donc pas non
    sur ce type de pointeur (

UR DE FONCTION

ype 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
```

EXE

- Déclaration de pointeurs

```
void foo_1(void) {...}
void foo_2(int i){...}
void foo_3(int i, char * s, size_t

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

URS DE FONCTIONS

on est une adresse, il est
dans un tableau

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

```
LLE])(l_types_fct);
```

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

TABLEAU DE POINT

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

TEUR DE FONCTION

ourner un pointeur de

envoyant le pointeur (disons
ointées s'entremêlent ainsi :

```
oo(l_args_foo))
_fct);
```

ramètre une liste
) et retourne un pointeur
otype est `type_ret_fct`

EXE

```
void foo_1(void){ printf(", World
int foo_2(char * s){ return printf
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

ILITÉ: EXEMPLE

mplémentées comme suit et pointeurs de fonctions

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

ILITÉ: EXEMPLE

tions manquent pour que le

```
;
;
```

RETOUR SUR L'

- Le choix de l'opér

```
double (* selectOperation(char cho
    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];
}
```

CONC

- Les pointeurs de fonctions certaine flexibilité dans le c
- La déclaration de type, vari utilisant un pointeur de fon type de fonction correspon
- Les erreurs sont faciles et c nécessite une grande vigila