

Chapitre 12 : Type générique et pointeur de fonction

Construction et maintenance de logiciels

Guy Francoeur

basé sur du matériel pédagogique d'Alexandre Blondin Massé, professeur

UQÀM | **Département d'informatique**

1. Type générique

2. Pointeur de fonction

1. Type générique
2. Pointeur de fonction

La généricité en C, comme dans d'autres langages, permet d'effectuer des actions, sur un ensemble de données de manière générique, c'est-à-dire quelque soit son type.

Définition - type générique

On peut donc réaliser des traitements qui pourront s'appliquer sur des **types entiers** comme des **types réels**, pourvu que le **type** qui sera utilisé lors de l'appel de fonction soit supporté par la **fonction générique** et ses instructions (le code).

Le pointeur générique

- ▶ Le langage C dispose d'un pointeur particulier appelé pointeur générique qui est un pointeur **compatible** avec tous les autres pointeurs;
- ▶ Le pointeur est à même de pointer vers n'importe quel **type d'objet**;
- ▶ Le premier exemple que nous connaissons :

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

Le pointeur générique - Résumé

- ▶ Il n'est pas possible de dé-référencer un pointeur générique;
- ▶ Ceci s'explique aussi par le fait que le pointeur générique ne sait pas vers quel type de variable son contenu pointe;
- ▶ En conséquent, le dérérérencer voudrait dire que l'on aurait une variable dont l'espace de stockage ne serait pas défini.
- ▶ Qu'est ce que détermine la grosseur en octet du premier élément pointé?

Table des matières

1. Type générique
2. Pointeur de fonction

- ▶ Jusqu'à maintenant, nous avons manipulé des pointeurs sur objet, c'est-à-dire des adresses vers des zones mémoires contenant des données **typées** (des entiers, des flottants, des structures, tableaux, ...);
- ▶ Mais aussi typées génériquement **type générique** (il y a quelques minutes de cela);
- ▶ Toutefois, il est également possible de référencer des instructions et ceci est réalisé en C à l'aide des pointeurs de fonction.

Déclaration

```
1  int (*pf)(char) // avec parametre  
2  int (*pf)() //sans argument
```

- ▶ elle retour un **int**;
- ▶ elle accepte un paramètre **char**;
- ▶ lorsque nous n'avons pas de paramètre, il faut garder la parenthèse comme dans la déclaration des fonctions;

- Les deux syntaxes suivantes sont acceptées.

```
1   pf = &fonction; // avec l'éperluette  
2   pf = fonction;  // sans éperluette
```

Nous allons voir deux façons d'usage du pointeur de fonction :

- ▶ par une variable qui pointe vers la fonction;
- ▶ par le passage du pointeur de la fonction vers une variable local;

Exemple

```
1 //pointeur_fonction1.c
2 #include <stdio.h>
3
4 static int ascii(char c) {
5     return c;
6 }
7
8 int main(int argc, char *argv[]) {
9     int (*pt)(char) = &ascii;
10    char c='0';
11
12    if (argc > 1) c = argv[1][0];
13
14    printf("%d\n", (*pt)(c));
15    return 0;
16 }
```

Exemple

```
1 //pointeur_fonction1.c
2 #include <stdio.h>
3
4 static int ascii(char c) {
5     return c;
6 }
7
8 static void affiche(char c, int (*pf)(char)){
9     printf("%d\n", (*pf)(c));
10 }
11
12 int main(int argc, char *argv[]) {
13     //int (*pt)(char) = &ascii;
14     char c='0';
15
16     if (argc > 1) c = argv[1][0];
17
18     affiche(c, &ascii);
19     return 0;
20 }
```

type générique et pointeur de fonction

- ▶ La fonction `qsort(...)` de la bibliothèque standard du C;
- ▶ Pour rappel sa signature est la suivante :

```
1 void qsort(void *base, //pointeur du premier élément
2           size_t nmemb, //nombre d'éléments
3           size_t size,  //grosseur d'UN élément
4           // et le pointeur de fonction de comparaison
5           int (*compar)(const void *, const void *));
```

<http://www.cplusplus.com/reference/cstdlib/qsort/>

Exemple - type générique et pointeur de fonction

```
1 // qsort.c
2 #include <stdio.h>      // printf
3 #include <stdlib.h>     // qsort
4
5 int values[] = { 40, 10, 100, 90, 20, 25 };
6
7 int compare (const void * a, const void * b){
8     return ( *(int*)a - *(int*)b );
9 }
10
11 int main () {
12     int n;
13     qsort (values, 6, sizeof(int), compare);
14     for (n=0; n<6; n++)
15         printf ("%d ", values[n]);
16
17     return 0;
18 }
```

<http://www.cplusplus.com/reference/cstdlib/qsort/>

- ▶ Un pointeur de fonction permet de stocker une référence vers une fonction;
- ▶ Il n'est pas nécessaire d'employer l'opérateur & pour obtenir l'adresse d'une fonction;
- ▶ Le dé-référencement n'est pas obligatoire lors de l'utilisation d'un pointeur de fonction;
- ▶ Il est possible d'utiliser un pointeur **générique** de fonction en ne fournissant aucune information quant aux arguments lors de sa définition;
- ▶ Une fonction employant un pointeur vers une autre fonction reçu en argument est appelée une fonction de rappel (*callback* function en anglais) .