Questions Hangout 2 – Socle en informatique Session 3

S5-2

Bonjour.

Dans la vidéo S5-2 (5:20), Mme Delacroix dit que le passage des arguments aux fonctions se fait *par valeurs*. Dans ce cas, je ne comprends pas comment fonctionne l'exemple suivant, présenté par Mr Graffion dans la démo Codecast S6-4 intitulée "Copie de chaîne de caractères", dont voici le code :

```
#include <stdio.h>
void strCopie(char dest[], char source[]);
int main(){
    char s1[10];
    strCopie(s1, "ABCDEF");
    printf("s1 = %s\n", s1);

    return 0;
}
void strCopie(char dest[], char source[]) {
    int i = 0;
    while (source[i] != '\0') {
        dest[i] = source[i];
        i++;
    }
    dest[i] = '\0';
}
```

La fonction `strCopie` copie la chaîne `"ABCDEF"`, passée en second paramètre, dans la chaîne `s1[]` passée en premier paramètre. Je comprends que le second paramètre puisse être recopié dans la variable locale `source[]`, mais comment cela fonctionne-t-il entre `s1[]` et `dest[]` ?

Pierre

Liste chaînée de nombres

Bonjour.

REMARQUES : Mme Delacroix nous présente, en exemple d'utilisation des pointeurs, la conception d'une *liste chaînée de nombres*. Elle indique que l'avantage d'une telle liste, par rapport à un tableau classique de nombres, est de ne pas avoir une taille maximum fixée à l'avance. C'est à dire, d'après ce que je comprends, qu'on pourra toujours rajouter un *élément* à la chaîne (on aurait pu dire un *maillon*) pour stocker un nouveau nombre.

Je crois qu'il faut ajouter deux contraintes pour que cet avantage soit réel :

1. Il faut mémoriser dans des variables pointeur nommées par exemple `prem` et `dern` les adresses du premier et du dernier élément de la chaîne :
   - `dern` est nécessaire pour chaîner un nouvel élément si besoin.
   - `prem` et `dern` sont nécessaires lorsque l'on souhaite accéder au contenu de la chaîne.
2. Si on souhaite accéder au contenu d'un élément particulier de la chaîne, il faut aussi connaître son rang, comme dans un tableau classique.

Compte tenu de la deuxième contrainte, le concept de liste chaînée semble réellement intéressant si on manipule la chaîne comme un tout, c'est à dire si on ne désire pas accéder aux éléments individuellement. Cela arrive peut-être plus souvent avec les chaînes de caractères.

Break

Bonjour,
Est-ce que, comme il est dit dans la vidéo, l'oubli d'un "`break`" entraîne bien l'exécution du bloc d'instructions suivant ? , ou conduit-il juste à tester les autres cas ?
Par exemple, si on veut imbriquer des cas, peut-on écrire :

```
…
case 0 :
case 1 :
case 2 :
    { instr. réalisées pour les valeurs 0, 1, 2}
case 2 :
    { instr. supplémentaires pour la seule valeur 2}
    break;
```

Juste une petite remarque pour le "`if/else`" présenté dans la vidéo :
N'aurait-on pu mieux utiliser le "`else`" , comme on l'a fait pour le dernier ?

```
...
if (score < 2)
    ...
else
    if (score  < 5)
        ...
```

Boucle répéter

Dans la session S3-2, que calcule exactement l'algo de la boucle répéter ?? Ce n'est pas dit et ce n'est pas limpide... à mon niveau

Syntaxe printf() et scanf()

Ma deuxième question concerne la syntaxe des primitives printf() et scanf(). J'ai remarqué que dans l'exemple sur la chaîne de caractères, il suffisait d'écrire le nom de la variable après la virgule :

scanf("%s", line);

Cependant, dans l'exemple des valeurs numériques, on observe l'ajout de l'opérateur & avant le nom de la variable :

scanf("%d", &ntl);

scanf("%f", &ftl);

Ma question est la suivante : quel est le rôle de cet opérateur & et dans quelles circonstances doit il apparaître ?

Entrée dans boucle FOR

Bonjour à tou-te-s,

Voilà, une question me taraude à propos des boucles FOR. On le voit dans l'exemple du programme devant dessiner un rectangle mais également dans d'autres exemples au cours des vidéos. Comment se fait-il qu'une boucle du type for (i = 1; i < x; i = i +1) ... s'execute alors que i a été initialisé en début de programme à 0? Et pourquoi initialiser la valeur dans la boucle for alors qu'elle existe déjà ? Ou si c'est obligatoire, dans ce cas, pourquoi avoir donné une valeur à la variable i en initialisation de début de programme et ne pas s'être contenté de la déclarer ?

Je ne sais pas si ma question est claire.