TD9 – Nombre variable d'arguments et qualificatifs *static* et *extern*

Langage C (LC4)

semaine du 29 mars

1 Fonctions à nombre variable d'arguments

La page « `man stdarg` » vous est donnée en document attaché.

Question 1. Écrire une fonction `moyenne` à nombre variable d'arguments prenant un premier argument fixe de type entier indiquant le nombre d'arguments qui suivent, supposés de type **`double`**, et renvoyant leur moyenne arithmétique. On rappelle que la moyenne arithmétique d'un ensemble de nombres $(x_i)_{1\leq i\leq n}$ est donnée par :

$$\frac{1}{n}\sum_{i=1}^{n} x_i$$

Question 2. Écrire une fonction `concatenation` qui prend en argument fixe un caractère `sep` suivi de chaînes de caractères puis d'un pointeur `NULL` et affiche les chaînes dans l'ordre, séparées par le caractère `sep`.

Pour la question suivante on suppose que **`void`** `print_objet_bizarre(objet_bizarre o)` sait afficher des `objet_bizarre`.

Question 3. Écrire une fonction `monprintf` prenant en argument une chaîne de caractères pouvant contenir un certain nombre de fois le motif `%O` et affichant sur la sortie standard la chaîne où les différentes occurrences de `%O` sont remplacées par les arguments suivants de l'appel à `monprintf`, qui seront du type `objet_bizarre`.

2 Qualificatifs *static* et *extern*

2.1 Utilisation de `static` sur les fonctions

Question 4. Définissez une fonction **`void`** `echange(`**`int`** `*c1,` **`int`** `*c2)` qui prend en argument deux pointeurs vers des entiers et échange les valeurs vers lesquelles ils pointent (Vous commencez à connaître cette fonction par coeur). Définissez une fonction **`void`** `tri3int(`**`int`** `*c1,` **`int`** `*c2,` **`int`** `*c3)` qui prend en argument trois pointeurs vers des entiers et qui les trie dans l'ordre croissant, en ce sens que le pointeur `c1` pointera vers le plus petit entier et `c3` vers le plus grand. (Il s'agit d'utiliser la fonction `echange`.)

On suppose que ces fonctions appartiennent à un fichier `tri3int.c`.

Question 5. De manière analogue définissez une fonction **void** `echange(`**char** `*c1,` **char** `*c2)` et une fonction **void** `tri3car(`**char** `*c1,` **char** `*c2,` **char** `*c3)`. On considère que l'ordre des caractères est l'ordre de leur code *ASCII*.

On suppose que ces fonctions appartiennent à un fichier `tri3car.c`.

Question 6. Que se passe-t-il lorsque l'on tente de compiler le code suivant (un fichier `main.c` à l'aide de la commande `gcc main.c tri3car.c tri3int.c`. Justifiez votre réponse et surtout expliquez comment il serait possible de remédier à ce problème.

```
#include <stdio.h>

void tri3int(int *, int *, int *);
void tri3car(char *, char *, char *);

int main()
{
  int i1 = 3, i2 = 2, i3 = 1;
  char c1 = 'x', c2 = 'z', c3 = 'y';

  tri3int(&i1, &i2, &i3);
  tri3car(&c1, &c2, &c3);

  printf("les entiers : %d, %d, %d\n", i1, i2, i3);
  printf("les caracteres : %c, %c, %c\n", c1, c2, c3);
}
```

2.2 Le qualificatif `extern`

Maintenant on va se donner pour but de compter le nombre d'appels aux fonctions `echange`, `tri3car` et `tri3int`. Pour cela on va déclarer trois entiers dans le fichier `main.c` qui seront initialisés à 0.

Voici une première modification de `main.c`.

```
#include <stdio.h>

void tri3int(int *, int *, int *);
void tri3car(char *, char *, char *);

int c_echange = 0;
int c_tri3car = 0;
int c_tri3int = 0;

int main()
{
  int i1 = 3, i2 = 2, i3 = 1;
  char c1 = 'x', c2 = 'z', c3 = 'y';

  tri3int(&i1, &i2, &i3);
  tri3car(&c1, &c2, &c3);

  printf("les entiers : %d, %d, %d\n", i1, i2, i3);
```

```
  printf("les carateres : %c, %c, %c\n", c1, c2, c3);

  printf("nombre d'echanges : %d\n", c_echange);
  printf("nombre de tri3car : %d\n", c_tri3car);
  printf("nombre de tri3int : %d\n", c_tri3int);
}
```

Question 7. Comment accéder aux entiers `c_echange`, `c_tri3car` et `c_tri3car` depuis les autres fichiers ?

Question 8. Modifiez `tri3int.c` et `tri3car.c` afin que `c_echange`, `c_tri3car` et `c_tri3car` comptent bien ce que l'on attend qu'ils comptent.

Question 9. Le qualificatif **`extern`** peut-il être utilisé sur toutes déclarations d'une même variable (d'un même entier par exemple) ?

Question 10. Cela aurait-il du sens de déclarer une variable de la manière suivante ?

```
extern int ma_var = 0;
```

Pourquoi ?

2.3 Utilisation de `static` à l'intérieur des fonctions

Maintenant on change d'avis et on décide de ne compter que les appels aux fonctions `tri3car` et `tri3int`. Pour ce faire on souhaite déclarer des compteurs à l'intérieur même de ces fonctions. En plus on en modifie les types, elles deviennent désormais :
– `int tri3car(char *c1,char *c2, char *c3)` et
– `int tri3int(int *c1, int *c2, int *c3),`
et l'entier renvoyé est le nombre d'appels déjà effectués.

Question 11. Comment feriez vous ?

2.4 Si on a le temps ...

Question 12. Qu'affiche le programme suivant ?

```
#include <stdio.h>

int f()
{
  static int cpt = 0;
  cpt++;
  return cpt;
}

int g()
{
  static int cpt = 1;
  cpt *= 2;
  return cpt;
}
```

```
int main()
{
  int i;
  for (i = 0; i < 3; i++)
    printf("%d\n", f());
  printf("--------\n");
  for (i = 0; i < 3; i++)
    printf("%d, %d\n", f(), g());
  return 0;
}
```