Polytech Nice Sophia
SI3 / 2021-2022
Erick Gallesio

Programmation Procédurale

Feuille 5

Fonctions à arité variable

1 Moyenne

Écrire la fonction moyenne

```
float moyenne(int count, ...);
```

s qui renvoie la moyenne des `count` nombres flottants qui lui sont passés en paramètres. Un exemple d'utilisation de cette fonction est donné ci-dessous:

```
printf("Moyenne1 = %f\n", moyenne(2, 10.0, 15.0));                // ⇒ 12.5
printf("Moyenne2 = %f\n", moyenne(5, 10.0, 15.0, 18.5, 0.0, 3.5)); // ⇒ 9
printf("Moyenne3 = %f\n", moyenne(0));                            // ⇒ 0
```

2 La fonction cat_strings

Coder la fonction

```
void cat_strings(char str1[], ...);
```

qui affiche à la suite tous ses paramètres (qui sont des chaînes de caractères) jusqu'à trouver le pointeur nul. Par exemple pour afficher `essai` on veut pouvoir écrire dans un programme:

```
{
...
cat_strings("es", "sai", NULL);
...
}
```

Note:

Pour récupérer une chaîne de caractères avec `va_arg` , il faudra spécifier `char *` pour le deuxième paramètre (on verra pourquoi dès que l'on aura vu les pointeurs en cours).

3 Calculatrice

On désire réaliser une petite calculatrice en C. Pour cela, on a besoin de la fonction à nombre variable de paramètres `evaluer` dont le prototype est

```
int evaluer(char operateur, int operande, ...);
```

Cette fonction permet d'appliquer `operateur` à sa liste d'opérandes. On supposera ici que cette fonction ne travaille que sur des nombres positifs et que la fin de sa liste d'opérandes sera dénotée par un nombre négatif. D'autre part, cette fonction n'implémente que les quatre opérations arithmétiques classiques (c'est-à-dire '+', '-', '*' et '/'). Ainsi,

```
evaluer('+', 1, 2, 3, -1)                          ⟶ 6
evaluer('-', 10, evaluer('*', 2, 2, 2, -1), 2, -1)  ⟶ 0
```

4 Une version simplifiée de printf (facultatif)

Coder la fonction

```
void Printf(char format[], ...);
```

qui se comporte comme la fonction `printf` standard et reconnaît dans son format les séquences suivantes:

- `%d` : affichage d'un entier en décimal;
- `%x` : affichage d'un entier en hexadécimal;
- `%f` : affichage d'un nombre flottant
- `%c` : affichage d'un caractère;
- `%s` : affichage d'une chaîne de caractères;
- `%%` : affichage du caractère `'%'` .

Remarque:

Afficher un nombre flottant est assez difficile, on trichera don un peu ici en utilisant la fonction `snprintf` qui permet de faire un affichage dans une chaîne de caractères. Par exemple,

```
char buffer[40]; // On suppose ici que 40 caractères suffisent pour un float
snprintf(buffer, 40, "%f", 3.1415926);
printf("%s\n", buffer);        // affiche 3.141593
```

Note:

Votre fonction ne pourra utiliser que la fonction `putchar` pour réaliser ses affichages.

Pour tester votre fonction, vous pourrez utiliser la fonction `main` suivante:

```
int main() {
  Printf("DEBUT\n%s%c c'est moi.\nTest nombres: 0x%x et %d et un négatif %d\n",
         "salut", ',', 161, 123, -12);
  Printf("Trois nombres sur la même ligne: %d %f %d\n", 1, 2.0, 3);
  Printf("Affichage d'un '%c' et encore un d'une autre façon '%%'\n", '%');
  Printf("Affichage d'un %% non suivi d'un caractère spécial ==> %z...\n");
  Printf("Attention au %% en fin de la chaîne format ==> %");
  Printf("\nFIN\n");

  return 0;
}
```

L'affichage attendu est le suivant:

```
DEBUT
salut, c'est moi.
Test nombres: 0xA1 et 123 et un négatif -12
Trois nombres sur la même ligne: 1 2.000000 3
Affichage d'un '%' et encore un d'une autre façon '%'
Affichage d'un % non suivi d'un caractère spécial ==> %z...
Attention au % en fin de la chaîne format ==> %
FIN
```

5 Une fonction de debug

On veut écrire la fonction `void Debug(char *format, ...)` pour la mise au point de nos programmes. Cette fonction, se comporte comme la fonction `printf`, si ce n'est qu'elle affiche toujours le message `***DEBUG:` en début de ligne et rajoute un saut de ligne à la fin de l'impression.

Pour écrire cette fonction, on écrit tout d'abord la fonction

```
void Vprintf(char format[], va_list ap)
```

en partant de la fonction écrite à l'exercice précédent. Une fois que cela est fait, on peut en profiter pour réécrire la fonction `Printf` précédente.

Ainsi,

```
int main() {
  Debug("Utilisation de la fonction de debug");
  Debug("On peut utiliser les caractères %% de Printf");
  Debug("comme on peut le voir ici: 0x%x et %d et un négatif %d",
        161, 123, -12);
  Debug("");
  Debug("et pour finir une %s", "chaîne");
  return 0;
}
```

produira:

```
***DEBUG: Utilisation de la fonction de debug
***DEBUG: On peut utiliser les caractère % de Printf
***DEBUG: comme on peut le voir ici: 0xA1 et 123 et un négatif -12
***DEBUG:
***DEBUG: et pour finir une chaîne
```