INF155 — RAPPELS: LES SOUS PROGRAMMES

Anis Boubaker, Ph.D.
Maître d'enseignement
École de Technologie Supérieure

SYNTAXE

Fonction:

```
type_retour identifiant_fonction( [type_p1 p1, type_p2 p2, … ] )
{

        //Instructions

}
```

Prototype

- Le prototype comprend trois informations:
  - L'identifiant de la fonction
  - Le type de données retourné par la fonction
  - La liste de paramètres

QUAND DEVRAIT-ON CRÉER UN FUNCTION?

- Lorsqu'une des situations suivantes se présente:
  - On a besoin d'effectuer un traitement plusieurs fois dans notre programme → Éviter la redondance de code
  - On identifie un sous problème qu'on tente de résoudre:
    - On écrit l'algorithme pour résoudre le problème global, en supposant que les sous problemes sont résolus
    - On implémente **(et on teste)** les fonctions pour résoudre chacun des sous-problèmes
    - On implémente notre programme principal selon notre algorithme
  - On identifie un traitement que l'on fait, dont on pourrait probablement se servir dans un autre contexte/programm → Réutilisabilité.

COMMENTER UNE FONCTION

- En plus des commentaires du programme, les fonctions doivent être commentées (ce sont des sous-programmes!)
- Le commentaire d'une fonction doit inclure les informations suivantes:
  - Nom et description de la fonction
  - Liste de paramètres et utilité de chaque paramètre
  - Valeur retournée par la fonction
  - Les effets de bord: spécifier quels paramètres changent de valeur suite à l'appel de la fonction (nous verrons ça au prochain cours!)

EXEMPLE DE FONCTION

```
/*
Fonction: factorielle
Description: calcule la valeur de la factorielle d'un nombre n
Arguments:
    - n (entier): nombre dont ont veut calculer la factorielle
Retour: Valeur de la factorielle (entier stocké dans un double)
Paramètres modifiés:
    - Aucun
*/
double factorielle(int n)
{
    //déclaration de variables
    int i; //Compteur de boucle
    double resultat=1; //Stocke le résultat du calcul

    for (i = 1; i <= n; i++)
    {
        resultat *= i;
    }
    return resultat;
}
```

Valeur retournée par la fonction

PASSAGE DE PARAMÈTRES PAR VALEUR

- Lors d'un appel de fonction, les arguments sont copiés et fournis à la fonction.
- Les paramètres formels auront donc comme valeur **la valeur des arguments**.
- Par exemple,

```
int k = 10;
int j;
j = factorielle(k);
```

La valeur de **k** est passée en paramètre (donc 10), et non la variable **k**.

DÉCLARATION DU PROTOTYPE

```
/********** DÉCLARATIONS DES FONCTION ***************/
/*
Fonction: factorielle
Description: calcule la valeur de la factorielle d'un nombre n
Arguments:
    - n (entier): nombre dont ont veut calculer la factorielle
Retour: Valeur de la factorielle (entier stocké dans un double)
Paramètres modifiés:
    - Aucun
*/
double factorielle(int n);
```

VISIBILITÉ

- On appelle **visibilité d'une variable**, la portion de code où la variable est accessible (i.e. utilisable)

- **Règles de visibilité en C:**
  - Une variable locale (i.e. non globale) est visible dans la fonction où elle a été déclarée.
  - Une *variable globale* est visible dans tout le programme

- **L'utilisation des variables globales EST INTERDITE.**