E3FR 3R-IN1A
Initiation à la programmation C
2024-2025

ESIEE PARIS

TP 1 : Les fonctions et les boucles

1 Prise en main de linux et du terminal

Créez un répertoire dans lequel vous placerez tous vos TP de C. Pour chaque séance, vous créerez un sous répertoire dans lequel vous placerez tous vos exercices.
N'utilisez pas de navigateur de fichier ! Nous allons voir comment utiliser le terminal.

- Utilisez la commande `ls`. Elle vous permet de lister le contenu du répertoire courant. Si vous voulez plus de détails sur les fichiers listés, vous pouvez utiliser la commande `ls -l`.
- La commande `mkdir` vous permet de créer un répertoire. Créez un répertoire `TP_3R-IN1A` avec la commande `mkdir TP_3R-IN1A`. Avec la commande `ls`, vérifiez que le répertoire a bien été créé.
- La commande `cd` vous permet de changer de répertoire. Entrez dans le répertoire `TP_3R-IN1A` avec `cd TP_3R-IN1A` et vérifiez qu'il est vide. Créer dedans un répertoire `TP1`.
- le répertoire parent du répertoire courant est le répertoire `..`. Utilisez `cd ..` pour retourner dans votre répertoire principal.
- La commande `mv` vous permet de déplacer des fichiers ou des répertoires.
- Vous pouvez enfin tester la commande de suppression `rm`, qui vous permet de supprimer un fichier. Pour supprimer un répertoire et tout ce qu'il contient, vous pouvez utiliser `rm -r` (pour récursivement).

2 Compilation (rappel)

- Pour compiler un fichier *`exercice1.c`*, il faut utiliser la commande suivante :
  `gcc -ansi -pedantic -Wall exercice1.c -o exo1`
  Si vous ne mettez pas `-o exo1`, le fichier produit s'appellera *`a.out`*.
- Pour lancez votre programme, il suffit de taper `./exo1` dans le terminal.

3 Les fonctions simples

Exercice 1.
Écrivez une fonction `add` qui prend en paramètre deux entiers et retourne leur somme.

Exercice 2.
Écrivez une procédure `add_and_print` qui prend en paramètre deux entiers et affiche leur somme

4 Les fonctions récursives

Exercice 3.
Écrivez une fonction `fact` qui prend en paramètre un entier et qui retourne la factorielle [1] de cet entier. Pour cela vous utilisez le concept d'appel de fonction récursive.

1. La factorielle de n est égale à $n! = n(n-1) \times ... \times 3 \times 2 \times 1$.

Exercice 4.
Écrivez une fonction `expo` qui prend en paramètre deux entiers a et b et retourne a^b. Pour cela vous utilisez encore le concept d'appel de fonction récursive. (essayer sans regarder le td2 !).

Exercice 5.

1) Écrivez une procédure récursive `print_n_char()` qui affiche n fois un caractère. Cette procédure prend en paramètre un caractère (`char c`) et un entier non signé (`unsigned int n`). Par exemple `print_n_char('*',5)` affiche `*****`

2) Écrivez la procédure récursive `pyra()` qui prend en paramètre un entier et qui affiche une demi pyramide renversée dont le nombre de niveau est cet entier.
Par exemple `pyra(4)` ; affiche :
```
****
***
**
*
```
Utilisez votre procédure `print_n_char` pour afficher une ligne. Pour passer à la ligne suivante, utilisez l'instruction `printf("\n");`

3) Modifiez la procédure `pyra()` pour afficher la pyramide à l'endroit.
C'est à dire `pyra(4)` ; affiche :
```
*
**
***
****
```

4) Modifiez la fonction `pyra()` pour afficher une pyramide complète.
Par exemple `pyra(4);` affiche :
```
   *
  ***
 *****
*******
```
Pour cela, demandez vous d'abord combien il y a d'étoiles sur la k-ième ligne, puis posez vous la même question pour le nombre d'espaces. Puis chercher une formule qui donnerait le nombre d'espace et d'étoile à afficher sur la ligne numéro k en fonction de la taille n de la pyramide.
Vous pouvez rajouter un deuxième paramètre à la procédure `pyra()`. Renommer votre procédure qui affiche la pyramide `pyra_rec()`, et ajouter une procédure `pyra()` qui ne prend qu'un seul argument et qui appelle `pyra_rec()`. Par exemple
```
void pyra (int n){
  pyra_rec(n,n);
}
```

Pour afficher les espaces, vous pouvez utiliser votre routine `print_n_char()`.
Par exemple `print_n_char(' ',2);` affiche 2 espaces.

5 Les boucles

Tout ce qui peut s'écrire en programmation récursive peut s'écrire en programmation itérative, et vice versa. Il y a équivalence stricte entre ce qui est programmable en récursif, et ce qui est programmable en itératif.

Exercice 6.
Après avoir écouter les explications du chargé de TD sur la boucle "for", refaites les exercices 3 à 5 mais cette fois sans utiliser la récursivité.