L3 - Méthodes numériques
TP 1 - Représentation numérique et rappels de C

Support de TP

Les TPs visent à appliquer de manière concrète les notions vues en cours. Il est donc conseillé de venir avec vos supports de cours ou de les télécharger sur le web.

Un programme C type ainsi que le fichier Makefile permettant de le compiler est disponible sur le site du cours. Pour chacune des parties du TP, vous devez créer un programme "tp1_i.c" où i est le numéro de la section.

Les TPs notés doivent être soumis sur la page Jalon du cours dans les deux semaines suivant le TP. Vous devez soumettre un fichier zip contenant le code **compilable** avec un simple appel à `make` ainsi qu'un court rapport de 2 pages maximum en **PDF** (sans page de titre). Le rapport ne doit contenir aucune sortie terminal ou ligne de code. Vous devez par contre pour chaque section du TP rédiger une réflexion sur les notions utilisées et ce que vous avez appris.

1 Taille mémoire des différents types

La taille mémoire des types `char, int, long, double, float,` dépend de chaque système. Pour utiliser ces différents types au maximum de leur potentiel, il est nécessaire de connaître leur taille mémoire ainsi que leurs limites numériques.

1. Coder un programme qui imprime à l'écran la taille mémoire de chacun des types cités précédemment. Vous pouvez pour celà faire appel à la fonction `sizeof`.
2. Pour chaque type entier (signé et non signé avec `unsigned`) imprimer la valeur maximum et minimum qu'ils peuvent contenir. Ces valeurs sont défines sur chaque système dans le fichier d'entête `<limits.h>`.

2 Fonction factorielle

1. Coder une fonction `factorielle` qui calcule la factorielle d'un nombre entier. Le type de la valeur retournée doit être un type `entier` qui sera définit au préalable à l'aide de l'instruction `typedef`. Faire attention à bien retourner la valeur 1 pour une entrée 0 de la fonction.
2. Coder dans le programme une boucle permettant d'afficher à l'écran la valeur de la factorielle pour une entrée allant de 0 à 22.
3. Faire varier le type de l'entier retourné par la fonction `factorielle` (`char,int,long,long long`) et déterminer jusqu'à quelle valeur d'entrée le résultat est exact pour chaque type (valeurs exactes sur wikipedia). Commenter.
4. Quelle est la complexité de la fonction `factorielle` ? Quelle est la complexité totale du programme ? Serait-il possible de diminuer la complexité asymptotique ? Si oui, comment ?

3 Série entière

Soit la série entière suivante :

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

Cette série peut être implémentée à l'aide d'une boucle `for` qui commence soit à `i=1`, soit à `i=n`.

1. Coder une première fonction `somme1` qui retourne un `float` et qui commence à `i=1`. Imprimer son résultat pour `n=100000`.
2. Coder une deuxième fonction `sommen` qui retourne un `float` et qui commence à `i=n` Imprimer son résultat pour `n=100000`.
3. Calculer la différence entre les deux résultats et l'imprimer à l'écran.
4. Coder une fonction `somme` plus précise qui utilise le type `double` pour déterminer une valeur plus exacte.
5. D'où vient la différence entre `somme1` et `sommen` ? Quelle implémentation est la plus précise ?

4 Plus grand commun diviseur (PGCD)

Dans cette section, on vous demande de coder une fonction retournant le PGCD de deux nombres entier.

1. Chercher sur le web des informations concernant l'algorithme d'Euclide.
2. Implémenter cet algorithme dans une fonction `pgcd`.
3. Afficher à l'écran des résultats pour différents entiers `a` et `b` et vérifier le fonctionnement correct de votre fonction.