Programmation Impérative 2 - L2 Informatique

TP1

1 Mise en route

Programmer proprement l'exercise sur le triangle isocèle à l'exercice IV du TD1. La longueur du triangle - son nombre de lignes - sera indiqué en argument sur la ligne de commande. Proposez une solution itérative et une solution récursive.

2 Fonction de Ackermann

La fonction de Ackermann est une fonction à deux arguments définie récursivement de la manière suivante :

$$A(m,n) = \begin{cases} n+1 & \text{si } m = 0 \\ A(m-1,1) & \text{si } m > 0 \text{ et } n = 0 \\ A(m-1, A(m,n-1)) & \text{si } m > 0 \text{ et } n > 0 \end{cases}$$

Programmez cette fonction récursive. Jusqu'à quelle valeur de m et n pouvez-vous aller ?

3 Types

Ecrire les instructions suivantes dans un programme, puis exécutez-le.

```
int i=2;
float x;
x = 5/i;
printf("Valeur de x: %f\n",x);
```

Que se passe-t-il maintenant si on remplace la ligne `x=5/i;` par:

- `x=5.0/i;`
- `x=5./i;`
- `x=5/(float)i;`
- `x=(float)5./i;`

Que se passe-t-il si on remplace le dernier `printf` par: `printf("Valeur de x: %d\n",x);`. Si nous sommes sûrs de nous, comment éviter un warning à la compilation ?

4 Calculatrice

4.1 Une version simple

Nous cherchons à programmer une calculatrice simple qui s'utilisera en ligne de commande et qui permettra deux types d'opérations:

- des opérations sur deux réels, du type:
  `calculer 3 op 4`
  `op` pouvant désigner une addition, soustraction, multiplication, division ou une fonction puissance (pour cette dernière vous pourrez programmer la fonction vue en TD). `op` pourra être défini à l'aide d'un caractère adéquat (`'+', '-', '*', '/', '^'`).

- le calcul d'une fonction sur un réel:
  `calculer fonction 5`
  `fonction` pouvant désigner des fonctions trigonométriques (sinus, cosinus), ou des fonctions classiques telles que `exp, ln, log, sqrt, ....`

Vous devrez écrire votre programme de façon modulaire avec les modules suivants:

- `main.c` qui contient la fonction principale lançant les appels aux différentes fonctions;

- `utilitaire.c` qui contiendra une fonction d'usage, avec nombre variable de paramètres, affichant l'aide du programme avec son usage correct et à la fin la ligne rentrée par l'utilisateur. Ce module devra également contenir 3 fonctions pour le cas d'une opération: une renvoyant le premier opérande, une autre le second et la dernière retournant l'opération (ou son code); ainsi que deux fonctions pour le second cas d'utilisation: une renvoyant la fonction (ou son code) à appliquer et une autre renvoyant la valeur du réel placé en argument;

- `fonctions_trigo.c` qui implémentera les fonctions trigonométriques;

- `fonctions_std.c` qui implémentera les fonctions standard `exp, ln, log, sqrt, ....` (vous pourrez utiliser la fonction `assert` pour vérifier les valeurs de certains paramètres)

Pour cette réalisation, vous penserez à bien respecter les remarques suivantes :

- la réalisation de fichiers d'en-tête propres,

- l'ajout de commentaires dans chaque fichier source/en-tête, de sorte qu'au début il y ait un commentaire indiquant ce que le fichier contient, le nom de l'auteur et la date de création, de plus on ajoutera un commentaire avant fonction informant du but de la fonction.

- la mise en place d'un Makefile en réfléchissant bien aux dépendances (avec éventuellement les en-têtes), attention à bien placer les tabulations aux bons endroits !

- la création de cibles avec des petits programmes de test visant uniquement à tester des modules (indépendamment des autres)

- donner des noms de variables et de fonction intelligibles et intelligents,

- vous servir de la librairie `math` pour l'implantation des fonctions.

4.2 Tests

Valider chacun des modules à l'aide des petits programme de tests, puis tester votre calculatrice.
Observer le comportement de l'utilitaire `make` lorsque vous modifiez certains fichiers, en fonction des dépendances indiquées. Vous pouvez également modifier/ajouter/supprimer des options pour voir les différents comportements.
Jetez également un coup d'œil à la réaction du compilateur si vous oubliez d'insérer des fichiers en-têtes.

4.3 Extensions (bonus)

- Maintenant, vous devez ajouter un nouveau modules avec des fonctions non standard permettant de calculer la factorielle d'un entier, la suite de Fibonacci jusqu'à un entier n. On souhaite également ajouter de nouveaux opérateurs comme `modulo` et `pgcd`. Que faut-il faire?
  Mettez tout ceci en place et testez votre nouvelle calculatrice.

- On souhaite maintenant permettre à l'utilisateur de fournir les arguments au clavier, plutôt que sur la ligne de commande. Ajoutez cette nouvelle fonctionnalité, de sorte qu'elle s'intègre bien au découpage modulaire déjà effectué. Vous penserez à gérer les erreurs.
  Une fois que tout ceci sera fonctionnel, l'utilisateur pourra utiliser la calculatrice jusqu'à ce qu'il rentre la valeur -1.

- Complétez maintenant votre calculatrice en proposant une interface graphique en utilisant la librairie MLV vue au premier semestre. Cette interface devra permettre d'écrire les mêmes opérations que la ligne du dessus.
  `http://www-igm.univ-mlv.fr/~boussica/mlv/`

- Vous pouvez enregistrer les opérations faites dans un fichier, et à chaque lancement de la calculatrice vous pouvez proposer dans un menu la liste des k dernières opérations effectuées pour permettre à l'utilisateur de relancer le même clacul.

- Si le temps le permet améliorez votre calculatrice pour gérer plus d'opérations, des parenthèses, etc.