TD #7&8 – DEV

14 octobre 2024
Erwan Kerrien
Abdel Bourjij

Rappel : si vous avez des questions sur ce TD ou sur le cours, n'hésitez pas à m'envoyer un mail à Erwan.Kerrien@inria.fr (je consulte plus rarement mon mail Erwan.Kerrien@univ-lorraine.fr).

1 Exercices

1. Écrire une fonction `ValeurAbsolue` qui prend en entrée un réel et en renvoie la valeur absolue.

   Écrire ensuite une fonction `MoitiéRésidu` qui divise un nombre réel par deux jusqu'à ce que sa valeur absolue tombe strictement sous 1, et renvoie cette valeur.
2. Dans l'exercice 1 du TD1, il était demandé d'entrer une valeur entre 50 et 100. Pour cela, nous avons écrit une boucle qui demande d'entrer une telle valeur et ne s'arrête que lors que l'utilisateur le fait. Mais avec un utilisateur bête (ou un programme malveillant), cela pourrait mener à une boucle infinie. Écrivez la fonction `LireDansIntervalle` qui demande de lire une nombre qui doit être compris entre la valeur minimale *imin* et la valeur maximale *imax* (incluse). Afin d'éviter la boucle infinie, on indiquera aussi un nombre maximal de tests à réaliser (*ntests*). La fonction renvoie *imin*, en affichant un message d'erreur si le nombre maximal est atteint sans qu'aucune valeur entrée soit valable. Si $ntests \leq 0$ (par exemple 0), la fonction s'exécute comme précédemment et la boucle se poursuit jusqu'à ce qu'une valeur correcte soit entrée.
3. Écrivez une fonction `AfficherLigne` qui prend en paramètre un entier n et affiche une ligne comportant n caractères *, et termine par un retour à la ligne.

   Reprenez les exercices 1.1 et 1.2 du TD2 et simplifiez-les en utilisant cette fonction. Pour mémoire, le premier demandait d'afficher un rectangle de largeur et hauteur données. Et le deuxième demandait d'afficher un triangle pointant vers la droite, de hauteur maximale donnée.
4. Reprenez l'exercice 2.2 du TD2 et réécrivez-le de telle manière que $f(t)$ soit calculée par un appel à une fonction appelée `Cubique`. Je vous rappelle le texte de cet exercice :

   Soit la fonction $f(t) = 2t^3 - t^2 - 37t + 36$. *Écrire un algorithme qui affiche la valeur minimale et la valeur maximale prise par cette fonction sur l'intervalle* $[-5, 5]$. *Pour ce faire, on calculera toutes les valeurs prises par la fonction pour chaque valeur de t allant de -5 à 5, tous les 0.25, et on en déterminera le minimum et le maximum.*

   Vous écrirez donc les fonctions suivantes :
   (a) Écrivez une fonction `Cubique` qui prend en entrée un réel t et renvoie la valeur de $f(t)$.
   (b) Écrivez une fonction `ValeursIntervalle` qui prend deux bornes d'un intervalle *imin* et *imax*, ainsi qu'un pas *step*, et renvoie toutes les valeurs de `Cubique` pour t allant de *imin* à *imax* tous les *step*.

   On pourra utiliser la fonction `AjouterFin(`T`,`E`)` qui ajoute un élément E en fin d'un tableau T, et renvoie le tableau augmenté, et/ou la fonction `Allouer(`n`)` qui réserve de l'espace mémoire pour un tableau à n éléments.
   (c) Écrivez une fonction qui prend en entrée un tableau de valeurs et renvoie la valeur minimale ainsi que la valeur maximale qu'il contient. On supposera donnée la fonction `Longueur` qui indique le nombre d'éléments dans un tableau. On suivra également la convention utilisée en Python qui accepte qu'une fonction renvoie plusieurs valeurs (*tuple*)
5. Écrivez une fonction `RésoudreQuad` qui résout une équation du second degré de la forme $ax^2 + bx + c = 0$: elle prendra en paramètre les valeurs de a, b et c et renverra le résultat sous la forme d'un tableau qui contiendra les solutions (2 en général, 1 si la solution est double et 0 s'il n'y a pas de solution), ainsi que le nombre de solutions. On supposera que la fonction `RacineCarrée` est disponible et renvoie la racine carrée d'un nombre (revoir la méthode de Héron pour avoir une idée de comment ça peut se calculer). Pour les plus rapides, implantez la fonction `RacineCarrée` en employant la méthode de Héron.

2 Exercices récursivité

1. Réécrivez la fonction `MoitiéRésidu` du premier exercice ci-dessus, qui prend en entrée un réel et le divise par deux successivement jusqu'à obtenir un nombre dont la valeur absolue est strictement inférieure à 1 qu'elle renvoie alors.

2. Écrivez, sous forme récursive, une fonction `Exposant` qui prend entrée un réel x et un entier n et renvoie x^n si $n > 0$. On remarquera que $x^n = x \times x^{n-1}$. Cet exemple est très proche de l'exemple classique de la factorielle (pour mémoire, la factorielle d'un nombre entier positif n est notée $n!$ et vaut $n \times (n-1)...2 \times 1$, soit le produit de tous les nombres inférieurs ou égaux à n). Ce n'est pas une récursion terminale car lors de la phase de dépilement, on récupère la valeur calculée par la fonction **puis** on fait le produit avec n avant de renvoyer la nouvelle valeur calculée.
3. La suite de Syracuse pour un entier n est définie de la manière suivante :

$$u_0 = n \tag{1}$$

$$\forall k \geq 0 \quad u_{k+1} = \begin{cases} \dfrac{u_k}{2} & \text{si } u_k \text{ est pair} \\ 3u_k + 1 & \text{si } u_k \text{ est impair} \end{cases} \tag{2}$$

   Écrivez une fonction récursive qui renvoie le terme d'indice N de la suite pour une valeur donnée de n.
4. La suite de Fibonacci (F_n) est définie pour tout entier n positif par :

$$F_0 = 0 \tag{3}$$

$$F_1 = 1 \tag{4}$$

$$\forall k \geq 2 \quad F_k = F_{k-1} + F_{k-2} \tag{5}$$

   Écrivez une fonction qui renvoie le terme d'indice $n \geq 0$ de la suite de Fibonacci. On pourra suivre les étapes suivantes :
   (a) En écrire une version itérative (avec boucle). Quelle en est la complexité ?
   (b) Écrire la version récursive (sans se compliquer la vie). Quelle en est la complexité ?
   (c) Proposer une version récursive efficace.