# TP4 : sous-programmes

Tous les sous-programmes de ce TP seront internes.

**Exercice 1.** Recopiez les exemples du cours de procédure et de fonction. Compilez et testez.

**Exercice 2.**

1. Ecrire une fonction qui renvoie le $n^{ieme}$ terme de la suite de réels définie par :

$$u_1 = 1 \quad ; \quad u_2 = 3 \quad ; \quad u_n = \frac{1}{u_{n-2}} + u_{n-1}$$

2. Ecrire un programme qui demande à l'utilisateur un entier $n$ et qui imprime à l'écran le $n^{ieme}$ terme de la suite.

**Exercice 3.**

1. Ecrire une procédure
   - qui prend comme arguments d'entrées un entier `nmax` et un réel `seuil` et comme arguments de sorties un réel et un booléen,
   - cherche parmi les `nmax` premiers termes de la suite définie ci-dessous, un terme qui est supérieur strictement à `seuil`,

$$u_1 = 1 \quad ; \quad u_2 = 3 \quad ; \quad u_n = \frac{1}{u_{n-2}} + u_{n-1}$$

   - si ce terme existe, les arguments de sorties auront pour valeur ce terme et `true`, sinon la valeur du booléen sera `false`.
2. Ecrire un programme
   - demande à l'utilisateur un réel et un entier
   - appelle la procédure précédente avec ces valeurs
   - selon la valeur du booléen renvoyé, écrit à l'écran : soit la valeur du terme de la suite, soit un message indiquant que la valeur n'est pas atteinte.

**Exercice 4. Passage des tableaux en arguments.** Dans cet exercice, on représentera une fonction polynomiale par un vecteur de taille égale au degré du polynôme plus un et dont la case $i$ contiendra le coefficient du monôme de degré $i$. Par exemple, le polynome $P(x) = 3x^3 + 5x^2 + 3$ sera représenté par le tableau suivant :

| 3 | 5 | 0 | 3 |
|---|---|---|---|

1. Ecrire une procédure qui prend comme argument d'entrée un polynôme (*i.e.* un vecteur) et qui modifie ce polynôme de la façon suivante : tous les coefficients dont le degré du monôme est pair seront multipliés par 2. Par exemple $P$ deviendra $3x^3 + 10x^2 + 6$.
2. Ecrire une fonction qui calcule la valeur d'un polynôme en un point. Cette fonction prendra pour arguments un vecteur et un réel et renverra un réel.
3. Ecrire un programme qui appelle les deux sous-programmes précédents.

**Exercice 5.**

Dans cet exercice, le polynôme $P(x) = \sum_{i=0}^{n} a_i X^i$ sera représenté par un type dérivé `polynome` composé :

— d'un entier `degre` qui représente le degré du polynôme
— d'un tableau de réels `coefficients` à une dimension contenant 101 cases numérotées de 0 à 100, représentant les coefficients du polynôme (on suppose que le degré du polynôme est au maximum 100).

Exemple : Pour $P$, l'entier `degre` vaut $n$ et le tableau `coefficients` est

$$(\backslash a_0, a_1, ..., a_n, 0, ..., 0\backslash)$$

Pour chaque question, tester la validité du sous-programme que vous avez écrit avant de passer à la suivante.

Dans un fichier `poly.f90`, créer le type dérivé `polynome`.

1. Ecrire une fonction `initialize_pol` qui prend comme paramètre un entier $n$ et qui retourne un polynôme de degré $n$ dont tous les coefficients seront initialisés à 0.
2. Ecrire une procédure `print_pol` qui prend comme arguments :
   — un polynôme,
   — une chaine de caractère `ch`,
   et qui affiche à l'écran sur une même ligne `ch` le degré du polynôme et ses coefficients (attention à ne pas écrire les 101 réels du tableau `coeff`).
3. Ecrire une procédure `cree_monome`, qui modifie la valeur d'un coefficient d'un polynôme donné. Vous prendrez garde à initialiser le polynôme si celui-ci a un degré 0. Cette procédure prend comme paramètres :
   — un polynôme `Poly`,
   — le degré $n$ du monome à modifier dans `Poly`,
   — la valeur du coefficent $a_n$ associé au monome de degré $n$.
   Remarque : si le degré du monome à modifier est supérieur au degré de `Poly`, on augmente la valeur du degré de `Poly`.
4. Ecrire une fonction `copie_pol` qui prend comme paramètres un polynôme $A$ et qui retourne un polynôme $B$ tel que $B = A$.
5. Ecrire une fonction `add_pol` qui prend comme paramètres deux polynômes $A$ et $B$, et qui retourne un polynôme $C = A + B$. Attention à gérer les cas où les termes de plus au degré s'annulent l'un l'autre.
6. Ecrire sur papier l'algorithme permettant de multiplier deux polynômes.
7. Ecrire de la même façon une procédure `mult_pol` qui effectue le produit de 2 polynômes.
8. Ecrire une fonction `derivpol` qui dérive un polynôme. Elle prendra en paramètre le polynôme et retournera sa dérivée.