# Programmation Impérative
# Premiers programmes en Ada

Notions acquises à l'issue du TP

- Savoir compiler un programme ADA.
- Savoir choisir la 'bonne' structure de contrôle dans un algorithme.
- Savoir spécifier et raffiner les fonctions.

## 1 Compiler et exécuter un programme ADA

1. Spécifier et raffiner en langage algorithmique une un programme 'somme' calculant la somme des entiers $i$ compris entre deux entiers naturels $m$ et $n$ donnés, tels que $m \leq i \leq n$.
2. Ecrire en ADA un programme affichant (si cela est possible) la somme des entiers $i$ compris entre deux entiers naturels $m$ et $n$ donnés, tels que $m \leq i \leq n$.

Mise en œuvre sur machine.

- Compilation et exécution ADA. Le nom du fichier contenant le programme ADA doit être 'somme_ada.adb'. Pour compiler 'somme_ada.adb' taper :

```
> gnatmake -gnatwa somme_ada.adb
```

Le programme exécutable s'appelle 'somme_ada'. L'exécuter en tapant :

```
> ./somme_ada
```

La commande 'gnatclean' efface les fichiers engendrés.

```
> gnatclean somme_ada
```

**Attention :** L'option `-gnatwa` demande au compilateur de signaler davantage de messages d'avertissement. Même s'ils n'empêchent pas le compilateur d'engendrer un exécutable, les avertissements correspondent généralement à des erreurs ou des maladresses qu'il faut corriger.

## 2 Calcul de la racine carrée d'un nombre (Méthode de Newton)

La $k^{ieme}$ approximation de la racine carrée de $x$ est donnée par

$$a_{k+1} = 0.5 * (a_k + x/a_k)$$

sachant que $a_0 = 1.0$.

On arrête le calcul quand la différence entre $a_{k+1}$ et $a_k$ est inférieure en valeur absolue à $\epsilon$ fixé.

1. Spécifier et raffiner en langage algorithmique un programme 'newton' calculant la racine carrée d'un nombre réel.
2. Ecrire le code ADA du programme 'newton'. La valeur et la précision seront lues au clavier.

## 3 Puissance

Afficher la puissance entière d'un réel en utilisant somme et multiplication. On pourra commencer par traiter le cas où l'exposant est un entier naturel puis généraliser aux entiers relatifs.

## 4 Suite de Fibonacci

Les termes de la suite de Fibonacci sont définis par la relation de récurrence suivante :
$fib(0) = 0$
$fib(1) = 1$
$fib(n) = fib(n-1) + fib(n-2) si\ n \geq 2$

1. Afficher le $n - ime$ terme de la suite de fibonacci, $n$ entier positif lu au clavier.
2. Afficher le $1er$ terme de la suite de Fibonnacci supérieur à $M$, un entier naturel lu au clavier tel que $M > 1$

## 5 Nombres parfaits

Écrire un programme qui affiche tous les nombres parfaits entre 2 et $N$ , $N$ étant lu au clavier. Un nombre parfait est un entier égal à la somme de ses diviseurs, lui exclu. Par exemple, 28 est un nombre parfait ($28 = 1 + 2 + 4 + 7 + 14$).

## 6 Nombres amis

Deux nombres $N$ et $M$ sont amis si la somme des diviseurs de $M$ (en excluant $M$ lui-même) est égale à $N$ et la somme des diviseurs de $N$ (en excluant $N$ lui-même) est égale à $M$ .

Par exemple, 220 et 284 sont amis. En effet, la somme des diviseurs de 220 hors 220 est $1 + 2 + 4 + 5 + 10 + 11 + 20 + 22 + 44 + 55 + 110 = 284$ et la somme des diviseurs de 284 hors 284 est $1 + 2 + 4 + 71 + 142 = 220$.

Écrire un programme qui affiche tous les couples $(N, M)$ de nombres amis tels que $1 < N < M \leq MAX, MAX$ étant lu au clavier.

**Indication :** Les nombres amis compris entre 2 et 100000 sont (220, 284), (1184, 1210), (2620, 2924), (5020, 5564), (6232, 6368), (10744, 10856), (12285, 14595), (17296, 18416), (66928, 66992), (67095, 71145), (63020, 76084), (69615, 87633) et (79750, 88730).