Memento de Programmation

- Le symbole `#` sert à commenter le programme, pour savoir ce que représentent les différentes variables, quelles sont les différentes étapes de l'algorithme... En pratique, tout ce qui est à droite d'un `#` sur une ligne du programme n'est pas lu par Maple et n'est utile que pour l'utilisateur (voir corrigé de l'exercice "renverser les chiffres").

- N'oubliez pas que les lignes où vous déclarez les variables locales et globales prennent un point-virgule mais pas la première ligne du programme *nom_prog*`:=proc(`*arguments*`)` ...

- On peut imposer le type des arguments dans une procédure avec la syntaxe *prog*`:=proc(`*arg*1`::`*type*1`,`*arg*2`::`*type*2`)` ...

- Les boucles `for` et `while` fonctionnent avec `do` et `end do;`

- Utilisation de `if`
  On peut utiliser différentes profondeurs de la modalité `if`

  – "*Si* condition *alors* conséquence" (`if ... then ...`)
  Exemple: cette procédure affiche le carré de l'argument, et en plus "hourrah" si l'argument est supérieur à 100, et en plus "cool" si il est supérieur à 1000.

```
ola1:=proc(n)
   if n>=100 then print(‘hourrah‘) end if;
   if n>=1000 then print(‘cool‘) end if;
n^2
end proc;
```

  On a deux `if... then...` à la suite.

– "*Si* condition *alors* conséquence1 *sinon* conséquence2"
(`if ... then ... else ...`)
Exemple: cette procédure fait la même chose que `ola1` avec une utilisation différente de `if`.

```
ola2:=proc(n)
if n>=100 then
     print(‘hourrah‘);
      if n>=1000 then print(‘cool‘); n^2
      else n^2
      end if;
else n^2
end if;
end proc;
```

On a deux `if... then... else...` imbriqués.

– "*Si* condition1 *alors* conséquence1 *sinon*, *si* condition2 *alors* conséquence2 ... *sinon* conséquence n"
(`if ... then ... elif ... then ... else ...`)
Exemple: encore le même résultat, programmé différemment...

```
ola3:=proc(n)
   if n>=1000 then print(‘hourrah‘); print(‘cool‘); n^2
   elif n>=100 then print(‘hourrah‘); n^2
   else n^2
   end if;
end proc;
```

Un seul `if... then... elif... then... else...`

- N'oubliez pas que l'on ne peut pas affecter une nouvelle valeur aux arguments de la procédure. Il faut déclarer une nouvelle variable (locale) à laquelle on donne la valeur de l'argument et qu'on peut manipuler librement.

- En programmation on se sert souvent de compteurs. Typiquement c'est une variable locale (par exemple `i`) qu'on initialise à 0 (`i:=0;`) et qui augmente d'une unité à chaque fois qu'on passe dans une boucle `for` ou `while` (`i:=i+1;`).