

## PROGRAMMER LA SUITE DES NOMBRES DE FIBONACCI

- DISCIPLINE : Mathématiques
- COURS : Calcul différentiel (201-NYA-05)
- LOGICIEL UTILISÉ : Maple
- NATURE : Programmation
- THÈME PRINCIPAL ABORDÉ : La suite des nombres de Fibonacci
- OBJECTIF(S) PÉDAGOGIQUE(S) : Maîtriser la démarche algorithmique. Maîtriser la commande répétitive et la récursion
- DURÉE APPROXIMATIVE : 20 minutes
- AUTEUR : Philippe Etchecopar
- COURRIEL/TÉLÉPHONE : [etchecop@globetrotter.qc.ca](mailto:etchecop@globetrotter.qc.ca)
- COLLÈGE OU UNIVERSITÉ D'ORIGINE : Cégep de Rimouski

### ÉNONCÉ DU THÈME À PROGRAMMER

*Déterminez un programme permettant de calculer le terme de rang  $n$  de la suite de Fibonacci.*

## GUIDE DU TUTEUR

### 1. Analyse

Comme le terme général est la somme des deux termes précédents, nous avons besoin de connaître les deux premiers termes. Si nous prenons comme deux premiers termes 0 et 1, les premiers éléments de la suite de Fibonacci s'écrivent :

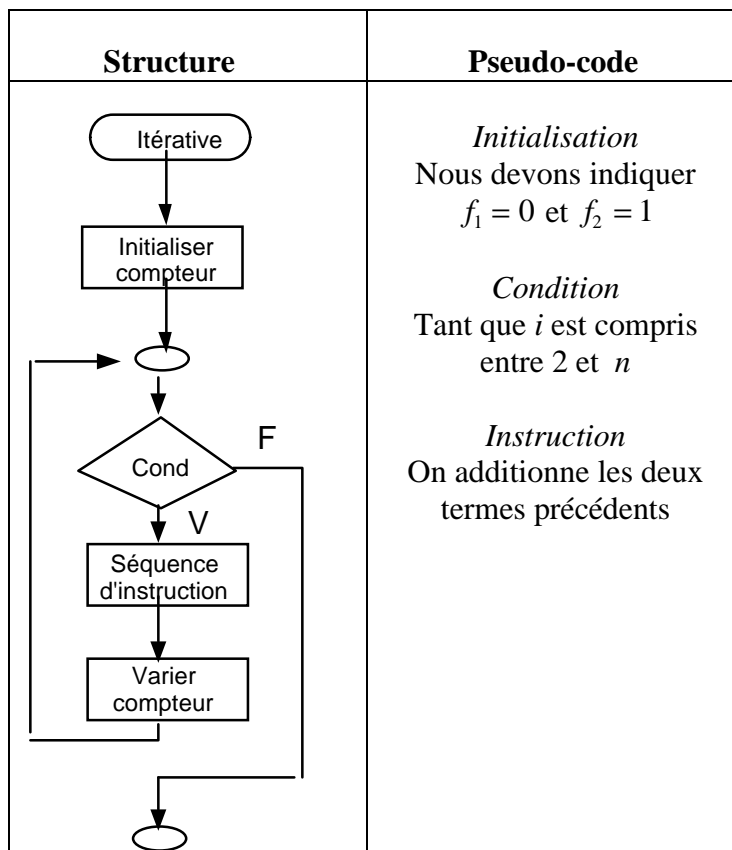
0, 1, 1, 2, 3, 5, 8, 13, 21, 34, ...

Le paramètre est le rang du terme recherché et nous devons obtenir ce terme. Comme il faut calculer les termes jusqu'à celui qui est demandé, la structure est itérative. Le rang  $i$  est l'itérateur.

### 2. Schéma (organigramme)

Reprenons ci-dessous le schéma d'une structure itérative. La séquence d'instructions consiste à prendre pour le terme de rang  $i$  la somme des deux termes précédents. Nous devons donner les deux premiers termes. Cette structure se termine lorsque cette différence est inférieure à la valeur donnée initialement.

Cela se traduit sur le schéma ci-dessous :



### 3. Traduction en langage *Maple* et validation

Ce schéma se traduit facilement en code *Maple*. Vous remarquez qu'à chaque boucle, le terme le plus antérieur,  $f_1$ , est le terme le moins antérieur de la boucle précédente,  $f_2$ , tandis que le terme le moins antérieur de cette boucle,  $f_2$ , est le terme précédent  $f$ . Pour valider le programme, nous calculerons sa valeur en deux points et nous représenterons la fonction

```
fibonacci:=proc(n)  
local i, f1, f2, f;  
f1:=0;  
f2:=1;  
for i from 2 to n do  
    f:=f1+f2;  
    f1:=f2;  
    f2:=f;  
od;  
RETURN(f2);  
end;  
  
fibonacci(40);
```

La validation nous donne :

<pre><b>"Validation";</b> <b>fibonacci(10);</b></pre>	<pre><b>"Validation"</b> <b>55</b></pre>
---	--

### 4. Évaluation

Il serait possible de compléter le programme pour obtenir la suite des termes.

Remarque

Nous pouvons appeler la procédure elle-même: Pour obtenir le terme de rang  $n$  de la suite de Fibonacci, nous aurions pu écrire :

```
fibonacci:=proc(n)  
option remember;  
if n<2 then  
    n;  
else  
    fibonacci(n-1)+fibonacci(n-2);  
fi;  
end;
```

- \* Si  $n \leq 2$ , le terme de la suite est  $n$  lui-même;
- \* Autrement il vaut la somme des deux termes précédents;
- \* L'option **remember** permet à *Maple* de garder en mémoire les calculs effectués et lui évite de recommencer à calculer pour chaque terme de la suite de Fibonacci tous les termes précédents. C'est un gain de temps important.

La validation nous donne :

<code>fib(20);</code>	6765
-----------------------	------

```
restart;
"Variables données";
f:=piecewise(x<-2, x, x<2, x^2-6, x);
a:=2;
"Graphique";
plot(f, x=-5..5, discont=true, color=red);
"Limites";
limit(f, x=2);
```

## Objectifs (notions et concepts visés)

Primaires : *Maîtrise de la démarche algorithmique*

Secondaires : *Maîtrise de la commande répétitive*

## Synthèse et question(s) de relance

*Déterminer d'autres variantes du programme.*

*Utiliser la commande `time()` pour comparer la performance de ces programmes.*