

Pour apprendre à programmer, il est CRUCIAL d'indenter : de faire apparaître les sous-parties de chaque programme en le décalant vers la gauche. La combinaison de touche majuscule+entree devrait vous aider.

I) Procédure

La syntaxe de définition d'une procédure est :

```
nomdeprocedure:=proc(var1,..,varn)
```

```
(facultatif) local lvar1,..,lvarp; N.B. : variables internes temporaires, servant au travail de la procédure et disparaissant à son retour
```

```
(facultatif) global gvar1,..,gvarp; N.B. : variables renvoyées avec leurs valeurs modifiées à l'issue de l'exécution de la procédure
```

```
instruction 1; N.B. : on peut terminer une instruction par : ou point virgule;
```

```
...
```

```
instruction r;
```

```
(facultatif) return resultat;
```

```
end proc;
```

La procédure `nomdeprocedure` peut ensuite être appelée avec des variables ou des valeurs qui remplaceront `var1,..,varn`, éventuellement avec des conditions, par exemple `n : : integer`

1. Construire une procédure calcul prenant pour argument deux entiers p et q et renvoyant $p + q$.

II) Structures de contrôle

Boucle If

La syntaxe de définition d'une boucle if est :

```
if condition1
```

```
then instructions1
```

```
(facultatif) elif condition2
```

```
(facultatif) then instructions2;
```

```
else instructionsn;
```

```
end if;
```

1. Définir une procédure `maxi(p,q)` qui renvoie le maximum de p et q .

Boucle For :

La syntaxe de définition d'une boucle for est :

```
for 'compteur' from 'indicedebut' to 'indicefin' (facultatif) by 'pas'
```

```
do instructions
```

```
end do;
```

1. Définir une procédure `geom(a,N)` qui calcule successivement les sommes $(S_k)_{0 \leq k \leq N} = \left(\sum_{j=0}^k a^j \right)_{0 \leq k \leq N}$ et renvoie S_N .
2. Comparer avec la fonction `sum`

Boucle While :

La syntaxe de définition d'une boucle while est :

```
while 'expressionbooléenne'
```

```
do instructions
```

```
end do;
```

attention, on ne peut sortir d'une telle boucle que si l'on est sûr d'avoir au cours de l'exécution l'expression booléenne `False`.

1. Définir une procédure `nmin(x,a)` qui à tout réel $x \in]-1, 1[$ et tout réel $a > 0$ renvoie le plus petit entier n tel que $x^n \leq a$.

2. Executer `nmin(1/√3, 0.01)`;
3. Vous aller devoir stopper la procédure!
Executer `nmin(2, 0.01)`;

III) Algorithme d'exponentiation rapide.

Lorsque a un élément de $\mathbb{R}, \mathbb{C}, \mathfrak{M}_n(\mathbb{K}), \dots$ est fixé, pour calculer a^8 , il est astucieux de calculer $a \times a = a^2$, puis $a^2 \times a^2 = a^4$, puis $a^4 \times a^4 = a^8$.

On va généraliser cette méthode à l'aide de l'algorithme d'exponentiation rapide.

1. A l'aide d'une boucle `for`, écrire une procédure `exponaive(a,n)` qui calcule a^n en $n - 1$ multiplications.
2. Lorsque $n = 2^p$, pour $p \in \mathbb{N}$
Définir une procédure `expopuiss2(a,p)` qui calcule $a^{(2^p)}$ en $p - 1$ multiplications.
3. Si a^k est connu, montrer que deux multiplications suffisent pour calculer a^{2k+1} .

```

4. expor :=proc(a,n)
    local aa,res,nn;
    res := 1; nn := n; aa :=a;
    while(nn>0) do
        if type(nn,odd)
            then res :=res*aa;nn :=nn-1;
            else aa :=aa*aa;nn :=nn/2;
            end if;
        od;
    res;
end proc;

```

5. Comparer la rapidité des deux algorithmes :

```

depart :=time() :exponaive(5,1000);arrivee :=time()-depart;
depart :=time() :expor(5,1000);arrivee :=time()-depart;

```

6. Une procédure récursive est une procédure qui "s'appelle elle-même". Comme pour les boucles `while`, il est prudent d'assurer une condition d'arrêt...
Définir la procédure récursive suivante :

```

exporec :=proc(a,n) local p,y;
    if n=0 then 1
    else
        p :=n mod 2;
        y :=a*a;
        if p=0 then exporec(y,n/2)
            else a*exporec(y,(n-1)/2)
            end if;
        end if;
    end proc;

```

7. Définir la matrice $A := \begin{pmatrix} 2 & 1 \\ 1 & 1 \end{pmatrix}$, et calculer `exporec(A,100)` ;

8. En utilisant la relation $\text{pgcd}(a, b) = \text{pgcd}(a \bmod b, b)$, définir une procédure récursive `pgcd(a, b)` qui connaissant deux entiers a et b calcule leur `pgcd`.

(on pourra utiliser les fonctions `iquo`, `irem`, `mod`),

9. Définir une procédure récursive `euclide(a,b)` qui connaissant deux entiers a et b calcule leur `pgcd`, en effectuant les divisions euclidiennes successives de l'algorithme du cours.
10. Programmer l'algorithme de division euclidienne des polynômes dans $\mathbb{R}[X]$.