CORRIGE du TP 1

B. Landreau

Solution des exercices proposes en cours et en fin de feuille TP1

Euclide

Algorithme d'Euclide

Les appels a la fonction printf (impression formatee) peuvent-etre occultes, il suffit de mettre un # en debut de ligne

```
> euclide:=proc(n,m)
  local a,b,r;
  a:=n;b:=m;
  while b>0 do
    r:=irem(a,b);
    printf("%d = %d * %d + %d\n",a,b,iquo(a,b),r);
    a:=b;b:=r;
  od;
  printf("\n le PGCD est %d",a);
  RETURN(a);
  end:
> euclide(145225,1524174);
145225 = 1524174 * 0 + 145225
1524174 = 145225 * 10 + 71924
145225 = 71924 * 2 + 1377
71924 = 1377 * 52 + 320
1377 = 320 * 4 + 97
320 = 97 * 3 + 29
97 = 29 * 3 + 10
29 = 10 * 2 + 9
10 = 9 * 1 + 1
9 = 1 * 9 + 0

 le PGCD est 1
```



```
>
>
```

Exponentiation rapide

Algorithme d'exponentiation rapide

```
> exp_rapide:=proc(a,n)
  local y,m,z;
  y:=1;z:=a;m:=n;
  while (m>0) do
    if m mod 2=1 then y:=y*z;
    fi;
    z:=z*z;
    m:=iquo(m,2);
  od;
  RETURN(y);
  end:
```

```
> exp_rapide(a,61);
>
```

$$a^{61}$$

```
>
>
```

Nombres premiers

Voici une premiere procedure utilisant la fonction **isprime**

```
- > premiers:=proc(n)
    local m,L;
    L:=NULL;#liste initiale vide
    m:=2;
    while m<=n do
      if isprime(m) then
        L:=L,m;
      fi;
      m:=m+1;
    od;
    RETURN([L]);
    end:
> P:=premiers(100);P[13];
```

$$P := [2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97]$$

$$41$$

On peut ameliorer la procedure precedente en ne commencant les tests qu'a 3 et allant de 2 en 2.

Une autre methode qui est tres efficace est le **crible d'Eratostene.**
Le principe est le suivant :
on dispose d'un tableau ligne de bits (0 ou 1) indexe de 2 jusqu'a N;
Au depart, tous les bits sont a 1.
On parcourt le tableau en commencant a 2.
On met le premier entier qui a un bit a 1 dans la liste des premiers, c'est a dire 2,
et ensuite on met a 0 tous les multiples de 2 jusqu'au bout du tableau.
Ensuite on avance jusqu'a temps que l'on rencontre a nouveau un 1.
En l'occurence c'est 3, on le met dans la liste des premiers, et on met tous
ses multiples a 0.
On itere le processus jusqu'a $\sqrt{N}$
Ensuite, il suffit de lire le tableau de
$\sqrt{N}$ jusqu'a N. Les 1 donnent les nombres premiers restants.
En effet, il est facile de voir que si un nombre n n'est pas premier, il a forcement un diviseur
inferieur ou egal a $\sqrt{n}$

```
>

> eratosthene:=proc(n)
  local a,i,k,tab,premier;
  tab:=vector(n,i->1);
  premier:=NULL; # liste des nombres premiers
```

```
  a:=floor(sqrt(n));# partie entiere de n^(1/2)
  i:=2;
  while (i<=a) do
    # on avance jusqu'a rencontrer un 1
    while tab[i]=0 do
      i:=i+1;
    od;
    premier:=premier,i; # on stocke
    # on barre tous les multiples de i
    k:=2*i;
    while k<=n do
      tab[k]:=0;
      k:=k+i;
    od;
    # on avance d'un cran
    i:=i+1;
  od;
  # il reste a noter les premiers de n^(1/2) a n
  while i<=n do
    if tab[i]=1 then
       premier:=premier,i;
    fi;
    i:=i+1;
  od;
  RETURN([premier]);
  end:
```

```
> P:=eratosthene(100);P[13];
```

$$P := [2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97]$$


```
>
```

Pascal

Fabrication du triangle de Pascal a l'aide de la formule
$\text{binomial}(n, p) = \text{binomial}(n-1, p-1) + \text{binomial}(n-1, p)$

On peut n'utiliser qu'un seul vecteur ligne a condition d'effectuer le calcul
de la nouvelle ligne a partir de l'ancienne en allant de la droite vers la gauche.
Attention, un vecteur est indexe a partir de 1 et non 0. Il faut donc tout decaler d'un cran.

```
> restart;
```

```
> pascal:=proc(n)
  local i,j,tab;
  tab:=vector(n+1,i->0);
  tab[1]:=1;
  print(1);
```

```
  for i from 1 to n do
    # calcul de la nouvelle ligne
    for j from i+1 to 2 by -1 do
      tab[j]:=tab[j-1]+tab[j];
    od;
      print(seq(tab[j],j=1..i+1));
  od;
  end:
> pascal(10);
```

$$1$$

$$1, 1$$

$$1, 2, 1$$

$$1, 3, 3, 1$$

$$1, 4, 6, 4, 1$$

$$1, 5, 10, 10, 5, 1$$

$$1, 6, 15, 20, 15, 6, 1$$

$$1, 7, 21, 35, 35, 21, 7, 1$$

$$1, 8, 28, 56, 70, 56, 28, 8, 1$$

$$1, 9, 36, 84, 126, 126, 84, 36, 9, 1$$

$$1, 10, 45, 120, 210, 252, 210, 120, 45, 10, 1$$

Fibonacci

La suite de Fibonacci est definie par
F_n=F_n-1+F_n-2

Une premiere version.

```
> fibo:=proc(n)
  local a,b,c,i;
  if n=0 then RETURN(0);
  fi;
  if n=1 then RETURN(1);
  fi;
  a:=0;b:=1;
  for i from 2 to n  do
    c:=a+b;
    a:=b;
    b:=c;
  od;
  RETURN(c);
  end:
```

```
> fibo(4);
```

$$3$$

```
> seq(fibo(i),i=0..10);
```

$$0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55$$

Une version **recursive**.
L'option remember indique a Maple qu'il doit memoriser les valeurs calculees pour eviter de les calculer plusieurs fois.

```
> fibo_rec:=proc(n)
  option remember;
  if n=0
    then RETURN(0);
  fi;
  if n=1
    then RETURN(1);
  fi;
  RETURN(fibo_rec(n-1)+fibo_rec(n-2));
  end:
> fibo_rec(4);
                                  3
> seq(fibo_rec(i),i=0..10);
                      0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55
>
>
```

Magic

Cette procedure calcule le pgcd de m et de n par l'algorithme d'Euclide de facon recursive.

```
> magic:=proc(n,m)
  if m=0 then
    RETURN (n);
  else
    magic(m,irem(n,m));
  fi;
  end;
```

magic := **proc**(*n*, *m*) **if** *m* = 0 **then** RETURN(*n*) **else** magic(*m*, irem(*n*, *m*)) **end if end proc**

```
> magic(12,16);
                                  4
> magic(14321437237,465464654323);
                                  1
>
>
```

FIN