

## arit

### Quelques fonctions supplémentaires d'arithmétique

**Calculatrices :** 89 92+ V200

**Niveaux :** 1ere S Tale Tale S

**Descriptif :** Groupe de programmes - Apporte des fonctions d'arithmétique supplémentaires présentes dans Derive : implémentation des fonctions next\_prime, nth\_prime, des algorithmes d'Euclide, Bezout, d'Eratosthène, fonctions d'inversion modulo n, de résolution d'équation diophantienne

**Auteur :** Alain Pomirol

**Mots-clefs :** Euclide, Bezout, congruences, entiers, premiers

**Date de dernière révision :** Septembre 2002

**Annexe :**

Fonctions complémentaires nécessaires : iter.zip

#### Présentation :

Il s'agit d'un ensemble de fonctions qui sont des implémentations de fonctions présentes dans le logiciel de calcul formel Derive mais absentes de la calculatrice.

#### **Premsuiv(x) :**

La fonction premsuiv(x) donne le plus petit nombre premier supérieur à x. Elle fonctionne aussi bien si m est positif ou négatif (et même s'il n'est pas entier). Elle correspond à la fonction next\_prime de Derive.

#### **Premnum(x) :**

La fonction premnum(n) donne le n<sup>ième</sup> nombre premier, positif si n est positif, négatif si n est négatif.

Elle correspond à la fonction nth\_prime du fichier number.mth de Derive.

#### **Eratosth(n) :**

Ce programme du crible d'Ératosthène est l'adaptation du programme Maple donné page 160 du livre de la collection Terracher, Hachette, TS spécialité.

#### **Bezout(a,b) :**

Cette fonction utilise trois listes locales u, v et w et une boucle while. Elle donne une liste contenant dans cet ordre le PGCD de a et b, l'entier x et l'entier y tels que  $a.x + b.y = \text{PGCD}(a,b)$ .

#### **Bezout2(a,b) :**

Cette fonction utilise la fonction iterate présente dans le fichier joint iter.zip. Elle donne une liste contenant 6 réels dont les trois premiers sont le PGCD de a et b, l'entier x et l'entier y tels que  $a.x + b.y = \text{PGCD}(a,b)$ .

**Euclide(a,b) :**

Cette fonction calcule le PGCD de a et b par l'algorithme d'Euclide, en utilisant une boucle while (programmation classique) et trois entiers u, v et r.

**Euclide2(a,b) :**

Cette fonction calcule le PGCD de a et b par l'algorithme d'Euclide, en utilisant la fonction iterate présente dans le fichier joint [iter.zip](#).

**Invmod(a,m) :**

Cette fonction donne l'entier naturel x inférieur à m tel que  $a \equiv x \pmod{m}$ .

L'algorithme utilisé est dérivé de l'algorithme de Bezout.

**Powermod(a,n,m) :**

Cette fonction donne :

$$a^n \pmod{m} \text{ si } n > 0$$

$$x \text{ tel que } x * a^{|n|} \equiv 1 \pmod{m} \text{ si } n < 0$$

**Solvemod(équation,var,mod) :**

Cette fonction permet de résoudre les équations du type  $ax + b \equiv cx + d \pmod{m}$ .

Elle part du principe suivant :

Soit l'équation  $ax \equiv b \pmod{m}$ .

- On sait que si a et m sont premiers entre eux, il existe u et v tels que  $au + mv = 1$ .

Donc on en déduit que  $au \equiv 1 \pmod{m}$  puis que bu est solution. La fonction *invmod* décrite précédemment donne u.

- Si a et m ne sont pas premiers entre eux, il existe u et v tels que  $au + mv = d$  où d est le PGCD de a et m. Donc  $au \equiv d \pmod{m}$ . Si  $b \equiv 0 \pmod{m}$  il y a des solutions, sinon il n'y en a pas.

Pour trouver une solution on résout alors l'équation  $\frac{a}{d} x \equiv \frac{b}{d} \pmod{\frac{m}{d}}$  qui est équivalente

mais qui est du premier type car  $\frac{a}{d}$  et  $\frac{m}{d}$  sont premiers entre eux.

Par la suite, on connaît la périodicité et le nombre des solutions inférieures à m : il y a d solutions et la période est  $\frac{m}{d}$ . On en prend d consécutives puis leur modulo m.

**Mode d'emploi :**

L'utilisation des fonctions est décrite dans la présentation, mais vous devez avant tout ne pas oublier de copier les fonctions complémentaires nécessaires et de vous placer dans le répertoire contenant les fonctions. En voici quelques exemples :

premsuiv(2^100) donne 1267650600228229401496703205653 (environ une minute)

premsuiv(-111) donne -113

premsuiv(-113.001) donne -127

premnum(100) donne 541

premmum(-1000) donne -7919 (temps de l'ordre de 1'30")

solvemod(3x=30, x, 9) , solvemod(3x-30=0, x, 9) et solvemod(4x-20=x+10, x, 9) donnent le même résultat : {1, 4, 7}.

### Sources :

**Bezout** (a,b)

Func

Local r,q,u,v,w

remain(a,b)→r

{a,1,0}→u

{b,0,1}→v

While r≠0

  remain(u[1],v[1])→r

  (u[1]-r)/(v[1])→q

  u-q\*v→w

  v→u

  w→v

EndWhile

string(a)&"\*"&string(u[2])&"+"&string(b)&"\*"&string(u[3])&"="&string(u[1])

EndFunc

**Bezout2** (a,m)

iterate(when(mod(v[4],v[1])=0,v,{mod(v[4],v[1]),v[5]-

v[2]\*floor(v[4]/(v[1])),v[6]-

v[3]\*floor(v[4]/(v[1])),v[1],v[2],v[3]),v,{m,0,1,a,1,0},-1)

**eratosth** (n)

Prgm

Local i,j,l,m,s

ClrIO

seq(i,i,1,n)→l

seq(0,i,1,n)→s

0→l[1]

2→i

While i\*i<n

  If l[i]≠0 Then

    For j,i\*i,n,i

      0→l[j]

    EndFor

  EndIf

  i+1→i

EndWhile

0→m

For i,1,n

  If l[i]≠0 Then

    m+1→m

    l[i]→s[m]

  EndIf

EndFor

```

For i,1,m,5
  Disp mid(s,i,5)
EndFor
EndPrgm

Euclide (a,b)
Func
Local u,v,r
remain(a,b)→r
a→u:b→v
While r>0
  v→u:r→v:remain(u,v)→r
EndWhile
v
EndFunc

Euclide2 (a,b)
iterate(when(v[2]=0,v,{v[2],remain(v[1],v[2])}),v,{a,b},-1)[1]

invmod (a,m)
when(gcd(a,m)=1,mod(iterate(when(mod(v[1],v[2])=0,v,{v[2],mod(v[1],v[2])},v[4],v[3]-floor(v[1]/(v[2]))*v[4])),v,{a,m,1,0},-1)[4],m),false,undef)

powermod (a,n,m)
mod((when(n>0,a,invmod(a,m)))^(abs(n)),m)

premnum (n)
Func
Local x,s,i
sign(n)→s
floor(abs(n))→n
If n≤1 Then
  2→x
Else
  3→x:2→i
  While i<n
    x+2→x
    If IsPrime(x):i+1→i
  EndWhile
EndIf
s*x
EndFunc

Premsui v (x)
Func
Local s
sign(x)→s
floor(abs(x))→x
If x<2 Then
  2→x
ElseIf mod(x,2)=0 Then
  x+1→x
Else
  x+2→x
EndIf

```

```
While not IsPrime(x)
```

```
  x+2→x
```

```
EndWhile
```

```
s*x
```

```
EndFunc
```

```
SolveMod (u,x,m)
```

```
Func
```

```
Local a,b,d,a1,b1,m1
```

```
If part(u,0)="":left(u)-right(u)→u
```

```
d(u,x)→a
```

```
-limit(u,x,0)→b
```

```
gcd(a,m)→d
```

```
a/d→a1
```

```
b/d→b1
```

```
m/d→m1
```

```
when(mod(b,d)=0,mod(iterates(s+m1,s,invmod(a1,m1)*b1,d-1),m),{ })
```

```
EndFunc
```