Module boolean

Instruction d'affectation autorisee

Operateurs disponibles:

boolean (boolean and boolean)

boolean (boolean or boolean)

boolean (boolean == boolean)

boolean (boolean != boolean)

Module int

Instruction d'affectation autorisee

Operateurs disponibles:

boolean (int < int)

boolean (int <= int)

boolean (int > int)

boolean (int >= int)

boolean (int == int)

boolean (int != int)

int (int + int)

int (int - int)

int (int * int)

int (int / int)

int (int % int)

int (- int)

proc inc(int op)

Ajout 1 au parametre op

correspond a l'operateur classique ++

func int abs(int op)

Retourne la valeur absolue

func int intRandom(int mini,int maxi)

Retourne un nombre entier au hasard compris dans l'intervalle

[mini,maxi]

func int shl(int op,int decal)

Retourne la valeur de op decalee a gauche de la valeur de decal

correspond a l'operateur classique <<

func int orbin(int op1,int op2)

```
Effectue un "ou" bit a bit des deux operandes

Correspond a l'operateur classique |
```

func int andbin(int op1,int op2)

```
Effectue un "et" bit a bit des deux operandes

Correspond a l'operateur classique &
```

Module string

```
   Instruction d'affectation autorisee

   Operateurs disponibles:

      boolean ( string <  string )

      boolean ( string <= string )

      boolean ( string >  string )

      boolean ( string >= string )

      boolean ( string ==  string )

      boolean ( string != string )

      string ( string +  string )
```

func string trim(string op)

```
   Retourne une chaine apres avoir elimine les espaces a droite et a gauche

   Exemple:

   cbstring d

   d="        ABC              "

   println(trim(d))
```

func boolean startsWith(string op,string critere)

```
   Retourne vrai si la chaine op commence par le critere fourni

   Exemple:

   cbstring d

   d="Hello"

   if startsWith(d,"He")

      println("Vrai pour He")

   end if

   if startsWith(d,"Hell")==true

      println("Vrai pour Hell")

   end if

   if startsWith(d,"Bonsoir")==false

      println("Faux pour Bonsoir")
```

```
end if
```

func boolean endsWith(string op,string critere)

```
Retourne vrai si la chaine op se termine par le critere fourni
Exemple:
cbstring d
d="Hello"
if endsWith(d,"lo")
   println("Vrai pour lo")
end if
if endsWith(d,"ello")==true
   println("Vrai pour ello")
end if
if endsWith(d,"soir")==false
   println("Faux pour soir")
end if
```

func string substring(string op,int debut,int fin)

```
Retourne la sous-chaine de op commencant a l'indice debut inclusif et
se terminant a l'indice fin exclusif
Attention :
utiliser un schema try...catch...end catch si un indice est hors limite.
Exemple:
cbstring d,r
d="Hello"
r=subcbstring(d,1,3)
println(r)
```

func string charAt(string op,int indice)

```
Retourne la sous-chaine de longueur 1 de op commencant a l'indice debut
Attention :
utiliser un schema try...catch...end catch si l'indice est hors limite.
Exemple:
cbstring d,r
d="Hello"
r=charAt(d,1)
println(r)
```

func string right(string op,int debut)

```
Retourne la sous-chaine de op commencant a l'indice debut
Attention :
```

utiliser un schema try...catch...end catch si l'indice est hors limite.

Exemple:

```
cbstring d,r
d="Hello"
r=right(d,1)
println(r)
```

func string left(string op,int fin)

Retourne la sous-chaine de op commencant a l'indice 0 et se terminant

a 'indice exclusif fin

Attention :

utiliser un schema try...catch...end catch si l'indice est hors limite.

Exemple:

```
string d,r
d="Hello"
r=left(d,1)
println(r)
```

func int size(string op)

Retourne le nombre de caracteres de op commencant a l'indice 0

Exemple:

```
string d
int n
d="Hello"
n=size(d)
println(itos(n))
```

func int indexOf(string op,string critere)

Retourne l'indice dans op ou se trouve la premiere occurence de critere

Retourne -1 si critere ne se trouve pas dans op

Exemple:

```
cbstring d
int n
d="Hello"
n=indexOf(d,"lo")
println(itos(n))
n=indexOf(d,"zut")
println(itos(n))
```

func int lastIndexOf(string op,string critere)

```
Retourne l'indice dans op ou se trouve la derniere occurence de critere
Retourne -1 si critere ne se trouve pas dans op
Exemple:
string d
int n
d="Hello"
n=lastIndexOf(d,"l")
println(itos(n))
n=lastIndexOf(d,"zut")
println(itos(n))
```

func string replace(string op,string marqueur,string remplacement)

```
Retourne une chaine a partir de op
ou toutes les occurences du parametre marqueur ont ete remplacees
par le parametre remplacement
Exemple:
string d,r
d="HelloHello"
r=replace(d,"lo","AZERTY")
println(r)
```

func string toLowerCase(string op)

```
Retourne une chaine a partir de op
ou toutes les lettres majuscules ont ete remplacees par
leur equivalent minuscule
Exemple:
string d,r
d="HelloHello"
r=toLowerCase(d)
println(r)
```

func string toUpperCase(string op)

```
Retourne une chaine a partir de op
ou toutes les lettres minuscules ont ete remplacees par
leur equivalent majuscule
Exemple:
string d,r
d="HelloHello"
r=toUpperCase(d)
println(r)
```

func boolean isDigit(string op)

```
Retourne vrai si et seulement si op ne contient que des chiffres
Exemple:
string d
d="2012"
if isDigit(d)
   println("vrai")
else
   println("faux")
end if
```

func boolean isAscii(string op)

```
Retourne vrai si et seulement si op ne contient que des
caracteres >= a l'espace
```

func boolean isMinusLetter(string op)

```
Retourne vrai si et seulement si op ne contient que des lettres minuscules
Exemple:
cbstring d
d="abCd"
if isMinusLetter(d)
   println("vrai")
else
   println("faux")
end if
```

func boolean isMajusLetter(string op)

```
Retourne vrai si et seulement si op ne contient que des lettres majuscules
Exemple:
string d
d="abCd"
if isMajusLetter(d)
   println("vrai")
else
   println("faux")
end if
```

func boolean isLetter(string op)

```
Retourne vrai si et seulement si op ne contient que des lettres
Exemple:
```

```
string d
d="abCd"
if isLetter(d)
   println("vrai")
else
   println("faux")
end if
```

func string deleteLastChar(string op)

Module double

```
Instruction d'affectation autorisee
Operateurs disponibles:
   boolean ( double <  double )
   boolean ( double <= double )
   boolean ( double >  double )
   boolean ( double >= double )
   boolean ( double ==  double )
   boolean ( double != double )
   double ( double +  double )
   double ( double - double )
   double ( double /  double )
   double ( double *  double )
   double ( - double )
Le type double definit des nombres reels sur 8 octets.
Type             Nombre     Nombre            Intervalle
d'octets   chiffres
significatifs
byte    Entier   1          3                 -128 a 127
short   Entier   2          5                 -32768 a 32767
int     Entier   4          10                -2 147 483 648 a 2 147 483 647
long    Entier   8          19                -9 223 372 036 854 775 808 a 9223372036854775807
float   flottant 4          7
double  flottant 8          15
```

func double abs(double op)

```
retourne la valeur absolue
```

func double sin(double op)

```
Retourne la valeur du sinus de op (op exprime en radians)
```

```
Exemple:

double x,y

x=3.1415926

y=sin(x/2.0)

println("Sin PI/2 "+dtos(y))
```

func double cos(double op)

```
Retourne la valeur du cosinus de op (op exprime en radians)
```

func double tan(double op)

```
Retourne la valeur de la tangente de op (op exprime en radians)
```

func double asin(double op)

```
Retourne la valeur de l'arc sinus de op (op exprime en radians)
```

func double acos(double op)

```
Retourne la valeur de l'arc cosinus de op (op exprime en radians)
```

func double atan(double op)

```
Retourne la valeur de l'arc tangente de op (op exprime en radians)
```

func double atan2(double y,double x)

```
Returns the angle theta from the conversion of rectangular coordinates (x, y) to polar coordinates (r, theta).
```

func double sinh(double op)

```
Retourne la valeur du sinus hyperboliquede op (op exprime en radians)
```

func double cosh(double op)

```
Retourne la valeur du cosinus hyperboliquede op (op exprime en radians)
```

func double tanh(double op)

```
Retourne la valeur de la tangente hyperbolique de op (op exprime en radians)
```

func double sqrt(double op)

```
Retourne la valeur de la racine carree de op (op exprime en radians)
```

func double exp(double op)

```
Retourne la valeur de l'exponentielle a base e de op (op exprime en radians)
```

func double log(double op)

```
Retourne la valeur du logarithme a base e de op (op exprime en radians)
```

func double pow(double op,double exposant)

Retourne la valeur de op a la puissance exposant

Module long

Le type long definit des entiers signes sur 8 octets.

Type		Nombre d'octets	Nombre chiffres significatifs	Intervalle
byte	Entier	1	3	-128 a 127
short	Entier	2	5	-32768 a 32767
int	Entier	4	10	-2 147 483 648 a 2 147 483 647
long	Entier	8	19	-9 223 372 036 854 775 808 a 9223372036854775807
float	flottant	4	7	
double	flottant	8	15	

Instruction d'affectation autorisee

Operateurs disponibles:

- boolean (long < long)
- boolean (long <= long)
- boolean (long > long)
- boolean (long >= long)
- boolean (long == long)
- boolean (long != long)
- long (long + long)
- long (long - long)
- long (long * long)
- long (long / long)
- long (long % long)
- long (- long)

Module byte

Le type byte definit des entiers signes sur un octet.

Type		Nombre d'octets	Nombre chiffres significatifs	Intervalle
byte	Entier	1	3	-128 a 127
short	Entier	2	5	-32768 a 32767
int	Entier	4	10	-2 147 483 648 a 2 147 483 647

```
long    Entier   8          19               -9 223 372 036 854 775 808 a 9223372036854775807
float   flottant 4          7
double  flottant 8          15
   Instruction d'affectation autorisee
   Operateurs disponibles:
      boolean ( byte <  byte )
      boolean ( byte <= byte )
      boolean ( byte >  byte )
      boolean ( byte >= byte )
      boolean ( byte ==  byte )
      boolean ( byte != byte )
      byte ( byte +  byte )
      byte ( byte - byte )
      byte ( byte *  byte )
      byte ( byte /  byte )
      byte ( byte % byte )
      byte ( - byte )
```

Module system

func int argCount()

```
   Retourne le nombre d'arguments de la ligne de commande
```

func string argGet(int numero)

```
   Retourne l'argument de la ligne de commande dont le numero est specifie.
   Le premier argument a 0 pour numero
```

proc exit()

```
   Arrete l'execution de l'application
```

proc throwex(string msg)exception

```
   Genere une exception
```

func string exceptionToString()

```
   Retourne la chaine de caracteres correspondant a l'exception
   A utiliser entre catch et end catch
   Exemple:
   int n
   try
      n=1/0
```

```
catch
   println(exceptionToString())
end catch
```

func string exceptionToMessage()

```
Retourne la chaine de caractères correspondant à l'exception
A utiliser entre catch et end catch
Exemple:
int n
try
   n=1/0
catch
   println(exceptionToMessage())
end catch
```

proc execute(string nomClasse,string methodname)exception

```
Execute une methode "static" ou "public" de la classe specifiee
La methode ne doit pas avoir d'arguments
```

proc execMethod(string nomMethod)exception

```
Execute une methode
Reserve au systeme
Ne doit pas etre execute directement
```

func string[] systemGetPropertyNames()exception

```
Retourne les noms des proprietes systeme disponibles
```

func string systemGetProperty(string nomPropriete)

```
Exemple :
string[]t
int i,n
try
   t=systemGetPropertyNames()
   n=length(t)
   for i while i
```

proc executerCmd(string commande)exception

```
Execute une commande de type MSDOS ou SHELL.
L'application attend que la commande soit terminee.
Deux fichiers de type texte stderr.txt et stdout.txt contiennent
les traces de l'execution.
```

Exemple:

```
try
   executerCmd("explorer toto.txt")
catch
end catch
```

proc startCmd(string commande)exception

Execute une commande de type MSDOS ou SHELL.

L'application n'attend pas que la commande soit terminee.

Exemple:

```
try
   startCmd("explorer toto.txt")
catch
end catch
```

Module console

proc print(string d)

Affiche dans la console la chaine d sans passer a la ligne

proc println(string d)

Affiche dans la console la chaine d et passe a la ligne

func string readLine()

Lit une ligne a partir de la console

func string readPassword()

Lit une ligne a partir de la console sans afficher les caracteres tapes

---

Module stringvector

Le type stringvector permet de constituer une suite de string dont on

ne connait pas a priori le nombre d'elements

proc add(stringvector v,string d)

Ajoute une chaine au vecteur

func int size(stringvector v)

Retourne le nombre d'elements du vecteur

proc removeAll(stringvector v)

Supprime toutes les chaines du vecteur

func string elementAt(stringvector v,int n)

Retourne l'element situe a l'indice n

proc removeElementAt(stringvector v,int n)

Supprime l'element situe a l'indice n

func boolean contains(stringvector v,string d)

retourne vrai si et seulement si v contient d

proc update(stringvector v,int n,string d)

Modifie le i-ieme element

Module conversions

func string btos(boolean op)

Convertit un booleen en chaine de caracteres true ou false

func boolean stob(string op)

func string stringToHex(string input)

Retourne une chaine ou chaque caractere de la chaine Ascii input

a ete converti en deux caracteres hexadecimaux

func string hexToString(string txtInHex)

Retourne une chaine ou chaque doublet de caracteres hexadecimaux

de la chaine txtInHex a ete converti en un caractere Ascii

func string itos(int op)

Convertit un entier en chaine de caracteres

func double ltod(long d)

Convertit un long en double

func int htoi(string op)

Convertit un chaine de caracteres hexa en entier

func string itoh(int op)

Convertit un entier en chaine de caracteres hexa

func string dtos(double op)

Convertit un nombre reel en chaine de caracteres

func int bytoi(byte op)

Convertit un byte en int

le byte est considere comme non signe

func string ltos(long op)

Convertit un long en chaine de caracteres

func long stol(string op)

Convertit une chaine en long

func long itol(int op)

Convertit un int en long

func double itod(int op)

Convertit un int en double

func int stoi(string op)

Convertit un chaine de caracteres supposee contenir une

notation d'entier en entier

Si la chaine ne contient pas une notation d'entier,

la valeur 0 est retournee

func double stod(string d)

Convertit un chaine de caracteres en double

func string padIntLeft(int val,int max,string car)

retourne une chaine comportant max caracteres cadre a gauche

par la chaine car

Exemple:

println(padIntLeft(2012,20,"*"))

func string padIntZeroLeft(int val,int max)

Retourne une chaine ou la valeur val a ete completee a gauche par des zeros

func string[] getWords(string donnee,string separateurs)

Retourne les mots d'une chaine donnee en utilisant LES separateurs specifies

func string[] vtos(stringvector v)

Retourne un tableau de string correspondant au vecteur

func int dtoi(double d)

Retourne la partie entiere d'un nombre reel

func double arrondi(double d)

Retourne le nombre reel arrondi en ajoutant 0.5

func string arrondiString(string d)

Si la chaine d contient un nombre reel

retourne l'arrondi par ajout de 0.5

Sinon

retourne la chaine initiale d

Fin si

func string fdtos(string pattern,double value)

Pattern:

[A][Z]000000[.000]

Le symbole A est facultatif (si utilise, fait un arrondi)

Le symbole Z est facultatif (si utilise, remplace les zeros a gauche par des espaces)

Le symbole 0 represente une position a afficher

Exemple:

```
double x
x=3.141
println(fdtos("Z0000000000",x))
println(fdtos("0000000000",x))
println(fdtos("Z00000000.00",x))
println(fdtos("00000000.00",x))
println(fdtos("AZ0000000000",x))
println(fdtos("A0000000000",x))
println(fdtos("AZ00000000.00",x))
println(fdtos("A00000000.00",x))
x=3.145
println(fdtos("Z0000000000",x))
println(fdtos("0000000000",x))
println(fdtos("Z00000000.00",x))
println(fdtos("00000000.00",x))
println(fdtos("AZ0000000000",x))
println(fdtos("A0000000000",x))
println(fdtos("AZ00000000.00",x))
println(fdtos("A00000000.00",x))
```

func string normeDouble(double d,int max)

Conserve 2 chifres apres la virgule et retourne une chaine contenant max caracteres

func string formatDouble(double d,int maxi)

fait un arrondi puis appel normeDouble

func int[] initIntArraySep(string valeurs,string separateur)

Permet d'initialiser un tableau d'entiers

Les valeurs doivent etre separees par le separateur

Exemple:

```
int[]t
t=initIntArraySep("72;74;45;200;800",";")
```

func int[] initIntArray(string valeurs)

Les valeurs doivent etre separees par des virgules

Module lexical

Types lexicaux:

IDENTIFICATEUR "I"

ENTIER "E"

REEL "R"

LITTERAL "L"

SYMBOLE "S"

COMMENTAIRE "/"

BOOLEEN "B"

Les commentaires sont elimines

proc start(lexical lex,string data,boolean noComment)

si noComment vaut true, les commentaires sont elimines

Caracteres speciaux

\n newline 0A

\t tabulation 09

\b backspace 08

\r carriage return 0D

\f form feed 0C

\\ anti slash 5C

\' simple quote 27

\" double quote 22

func string us(lexical lex)exception

Retourne le mot suivant

Retourne "" si on est arrive a la fin

func string getType(lexical lex)

Retourne le type du mot courant

func int getNum(lexical lex)

Retourne le numero de ligne du mot courant

func int getPos(lexical lex)

Retourne l'indice courant dans la chaine

Module datetime

func int[] getDate()

Retourne la date et heure courante dans le tableau a 7 elements

indice definition

0 annee

1 mois

2 jour

3 heures

4 minutes

5 secondes

6 milli-secondes

func string stamp()

retourne un stamp sous la forme

aaaammjjHHMMSSTTT

ou

aaaa annee

mm mois

jj jour

HH Heures

MM Minutes

SS Secondes

TTT Milli-secondes

proc sleep(int milliSecondes)

Suspend l'execution pendant le nombre de millisecondes specifie

Exemple :

```
println("Bonjour")
sleep(1000)
println("Bonsoir")
```

proc startChrono()

Demarre le chronometre

Exemple:

```
startChrono()
```

func int stopChrono()

Arrete le chronometre, on recupere la duree en milli-secondes

Exemple:

```
int duree
startChrono()
sleep(1000)
```

```
duree=stopChrono()
println("Duree "+itos(duree))
```

func string durationToString(int duration)

```
* Retourne une chaine resultant de la conversion d'une duree
* exprimee en milli-secondes sous la forme:
* 99 h 99 m 99 s 999 ms
```

func string topChrono()

```
 * Arrete le chronometre,
 * retourne durationToString,
 * redemarre le chronometre
```

func string getDateJour()

```
retourne la date du jour sous le format jj/mm/aaaa
```

func string stringToDate(string d)

```
donnee verifie que d est sous la forme jj/mm/aaaa
retourne "null" si erreur
aaaammjj
```

func int getNbJoursMois(string d)

```
Donnee date d sous la forme jj/mm/aaaa
retourne le nombre de jours du mois
retourne -1 si la date est incorrecte
```

func int getNumJour(string date)

```
Date donnee jj/mm/aaaa
retourne
1 Dimanche
2 Lundi
3 Mardi
4 Mercredi
5 Jeudi
6 Vendredi
7 Samedi
-1 si date incorrecte
```

func string getNomJour(string date)

```
Date donnee jj/mm/aaaa
retourne
Dimanche
Lundi
```

Mardi

Mercredi

Jeudi

Vendredi

Samedi

null si date incorrecte

func string stampToString(string stamp)

Donnee : aaaammjjHHMMSSTTT

Retourne : 3 premiers caracteres du libelle du jour puis jj/mm/aaaa HH:MM:SS

Exemple : Lun 12/08/2011 10:56:04

func string getNomMois(string date)

Date donnee jj/mm/aaaa

retourne

Janvier

Fevrier

...

Decembre

null si date incorrecte

func string compareDate(string d1,string d2)

d1 et d2 doivent etre sous la forme jj/mm/aaaa

retourne

< si d1 < d2 ou

> si d1 > d2 ou

= si d1 == d2 ou

null si d1 ou d2 ne sont pas des dates correctes

func int diffDate(string date1,string date2)

date1 et date2 de la forme jj/mm/aaa

func int diffDateMois(string date1,string date2)

Retourne le nombre de mois entre deux dates

Les dates doivent etre au format jj/mm/aaaa, en cas d'erreur on retourne -1

func string addDate(string date,int nbJours)

Date donnee jj/mm/aaaa

func string addDateOuvrable(string date,int nbJours)

Donnee date sous la forme jj/mm/aaaa

Ajoute le nombre de jours "ouvrables", c'est a dire ne compte pas les

samedi et les dimanche

func string dernierJourMoisPrecedent(string date)

retourne le nombre de jours du dernier jour du mois precedent

La date doit etre sous la forme jj/mm/aaaa

Module filedirectory

proc mkdir(string chemin)exception

Cree un repertoire

proc createDirectories(string chemin)exception

Analyse un chemin et cree TOUS les repertoires necessaires

proc systemOpen(string chemin)exception

Ouvre un fichier externe avec le programme par default associe au type de

fichier.

Exemple:

systemOpen("book.pdf")

Remarque:

Le chemin ne doit pas debuter par . ou ..

proc renameFile(string ancienNom,string nouveauNom)exception

Permet de renommer un fichier

func boolean existsFile(string nomFichier)

Retourne vrai si et seulement le fichier ou le repertoire existe.

func long sizeFile(string nomFichier)

retourne la taille d'un fichier en nombre d'octets

proc deleteFile(string chemin)exception

Detruit le fichier ou le repertoire specifie

deleteFile("toto")

Remarque:

Si le chemin est un repertoire, celui-ci doit etre vide

proc copyFile(string cheminDepart,string cheminArrivee)exception

Permet de copier un fichier

proc createFile(string chemin)exception

Cree un fichier vide

proc createTmpFile(string nomFichier)exception

Cree un fichier temporaire vide, cree automatiquement tous les sous-repertoires necessaires

func long lastModified(string chemin)

Retourne la date de derniere modification d'un fichier

exprimee en nombre de milli-secondes ecoulees depuis le 1 janvier 1970 a 0 h GMT

func string lastModifiedDateFile(string chemin)

Retourne la date de derniere modification d'un fichier sous la forme

jj/mm/aaaa hh:mm:ttt

ou ttt sont les milliemes de secondes

func int superCopyFile(string cheminDepart,string cheminArrivee,boolean bEcrase)exception

si bEcrase vaut vrai, le fichier est toujours recopie

sinon

le fichier est copie si la date de derniere modification du fichier de depart

est plus recente que la date de derniere modification du fichier d'arrivee.

Dans ce cas, le chemin d'arrivee est analyse, les sous-repertoires necessaires

sont crees automatiquement

fin si

retourne 1 si le fichier a ete copie, 0 sinon

func string getCurrentDirectory()

Retourne le repertoire de courant user.dir

func string[] getDirectory(string nomDirectory)exception

Retourne le premier niveau

func string[] getSuffixeDirectory(string nomDirectory,string suffixe)exception

Retourne le premier niveau en ne conservant que les noms de fichiers se

terminant par le suffixe precise

Insensible a la casse

func string[] getRecurseDirectory(string rep,string filtre)

Le parametre 'rep' peut se terminer eventuellement par le symbole / ou

le symbole \

Retourne la liste de tous les noms absolus de fichiers.

ATTENTION : Chaque nom absolu est prefixe par la lettre D ou F (Directory ou File)

Le parcours d'arbre est en profondeur.

Le parametre 'filtre' permet de definir les suffixes des fichiers a conserver.

Chaque suffixe doit etre separe par le symbole ;

Le paramere suffixe peut etre ""

Exemple .java;jpg;gif

func boolean cutCopyFile(string cheminDepart,string cheminArrivee)exception

Le chemin d'arrivee ne doit pas exister

Equivalent de la commande Windows Cut puis Copy

func boolean isDirectory(string chemin)

Retourne vrai si et seulement si le chemin est un repertoire

proc deleteDirectory(string chemin)exception

ATTENTION,

detruit le repertoire et TOUS ses sous-repertoires

func int superCopyDirectory(string cheminDepart,string cheminArrivee,boolean bEcrase,boolean bTrace)exception

Le chemin d'arrivee peut se terminer ou non par un slash.

si bEcrase vaut vrai, le repertoire est toujours recopie

sinon

chaque fichier du repertoire est copie si la date de derniere modification du fichier de depart

est plus recente que la date de derniere modification du fichier d'arrivee.

Dans ce cas, le chemin d'arrivee est analyse, les sous-repertoires necessaires

sont crees automatiquement

fin si

si bTrace vaut vrai affichage du detail des fichiers copies

fin si

retourne le nombre de fichiers copies

Module textfile

proc open(textfile fichier,string nom,int mode)exception

Ouvre un fichier.

Mode

0 INPUT	Lecture
1 OUTPUT	Creation
2 EXTEND	Ajout de lignes a la fin du fichier
3 OUTPUT TEMPORARY FILE	Creation d'un fichier temporaire detruit a la fin de l'execution de l'application

proc setTmp(string nomRepTmp)exception

Initialise le nom du repertoire repTmp contenant les fichiers temporaires,

par defaut, ce nom est egal a "tmptmp"

func string openTmp(textfile fichier,string nom,int mode)exception

cree un fichier temporaire avec generation automatique du nom:

repTmp+"/"+nomUtilisateur+"/"+nom+stamp()

le parametre mode ne sert a rien, il est conserve pour compatibilite

avec l'open classique

func boolean readLine(textfile fichier,string ligne)exception

```
Lit une ligne
Retourne vrai si OK
Retourne faux si on est arrive en fin de fichier
Exemple:
textfile fich
string ligne
boolean b
try
   open(fich,"toto.txt",0)
   b=readLine(fich,ligne)
   while b
      println(ligne)
      b=readLine(fich,ligne)
   end while
   close(fich)
catch
   println(exceptionToString())
end catch
```

proc close(textfile fichier)exception

```
Fermeture du fichier
```

proc print(textfile fichier,string d)exception

```
Ecrit dans le fichier sans passer a la ligne
```

proc println(textfile fichier,string d)exception

```
Ecrit dans le fichier en passant a la ligne
```

func int getLineNumber(string nomFichier)exception

```
Retourne le nombre de lignes du fichier
```

func string[] getFileArray(string nomFichier)exception

```
Charger un tableau directement
```

proc putFileArray(string nomFichier,string[] t)exception

```
Ecrit le contenu d'un tableau
```

proc putFileString(string nomFichier,string data)exception

```
Ecrit un fichier contenant la chaine data
```

func string getFileString(string nomFichier)exception

Retourne le contenu d'un fichier texte

Module inifile

Structure d'un fichier ini

- une ligne commencant par // est un commentaire

- un fichier ini peut comporter plusieurs sections:

[nom de section]

- une section contient des definitions:

motcle=valeur

- Une section peut comporter plusieurs fois le meme mot-cle.

proc open(inifile fichier,string nom)exception

Ouvre et charge en memoire un fichier ini.

Ensuite, on peut utiliser toutes les fonctions decrites dans ce module.

Il n'y a pas de close a faire

Exemple:

```
inifile fich
string w
try
   open(fich,"toto.ini")
   w=getWord(fich,"general","nom")
catch
   println(exceptionToString())
end catch
```

func string[] getSections(inifile fichier)

Retourne les noms de sections

func string[] getWords(inifile fichier,string nomSection,string motCle)

Si motCle == "" la fonction retourne tous les mots de la section

func string getFirstWord(inifile fichier,string nomSection,string motCle)exception

Provoque une exception si non trouve

proc setFirstWord(inifile fichier,string nomSection,string motCle,string valeur)exception

Modifie un fichier ini

en remplacant le premier mot d'une section par la valeur specifiee

Pas d'erreur si le mot-cle n'existe pas

func string getWord(inifile fichier,string nomSection,string motCle)

Retourne "null" si non trouve

Module arraytools

proc sort(string[] t)

Tri le tableau par ordre croissant par la methode des bulles (Bubble Sort)

proc print(string[] t)

Affiche dans la fenetre d'execution un tableau a de chaines a 1 dimension

proc printArray(string[][] t)

Affiche dans la fenetre d'execution un tableau de chaines a 2 dimensions

proc printIntArray(int[][] t)

Affiche dans la fenetre d'execution un tableau d'entiers a 2 dimensions

---

Module stringhashmap

Le type **stringhashmap** permet de constituer une hash de couples (cle,valeur)

de string dont on ne connait pas a priori le nombre d'elements

func int size(stringhashmap v)

Retourne le nombre d'elements de la hashmap

proc removeAll(stringhashmap v)

Supprime tous les elements de la hashmap

func string get(stringhashmap v,string cle)

Retourne l'element situe de valeur de cle

Retourne ??? si non trouve

proc remove(stringhashmap v,string cle)

Supprime l'element de valeur de cle

proc put(stringhashmap v,string cle,string valeur)

Modifie le i-ieme élement

proc putArray(stringhashmap v,string[] tCle,string[] tVal)

func string[] getKeys(stringhashmap v)

func string[][] getHashmap(stringhashmap v)

Module stringstack

proc raz(stringstack pile)

Supprime tous les elements de la pile

proc push(stringstack pile,string valeur)

Empile une valeur

func string pop(stringstack pile)

Depile une valeur

func boolean isEmpty(stringstack pile)

Retourne vrai ssi la pile est vide

Module zip

proc filesToZip(string nomArchive,string[] listeNoms)exception

* Cree un fichier zip contenant tous les fichiers definis par listeNoms

---