fen_calcul_formel

Informations sur les champs

Fenêtre : fen_calcul_formel

Champ de saisie

	Position	Position	Largeur	Hauteur	Plan	Visib	Etat initial	Touc	Type saisie	Multi	Form	Mot	Saisi	Mise	NUL	Saisi	Effac	Fin	Défil	Défil	Asc.	Asc.
sai_formule	50	114	701	33	0	O	Actif	O	Texte								O		O			

Bouton

	Position	Position	Largeur	Hauteur	Plan	Visib	Etat initial	Touc	Type bouton
bt_go	671	188	80	24	0	O	Actif	O	Normal

Interrupteur

	Position	Position	Largeur	Hauteur	Plan	Visib	Etat initial	Touc	Nb. colonnes	Col.	Trois
int_afficher_commentaires	6	175	309	34	0	O	Actif	O	1	O	

fen_calcul_formel

Code

Déclarations globales de fen_calcul_formel

```
PROCEDURE MaFenêtre()

CONSTANTE                                   // 2 constantes purement arbitraires.
    code_erreur ="$_ERR_$"
    num_erreur = -999
FIN

s_compte_rendu_traitement est une chaîne    // On a tracer toutes les étapes du traitement dans cette chaine, et on pourra l'afficher à la fin.
s_solution est une chaîne                       // Chaine dans laquelle on va écrire la ou les solutions.
ch_droite, ch_gauche    est une chaîne          // Respectivement partie droite et gauche de la formule en cours , le séparateur étant le signe =
ch_formule    est une chaîne                // Redondant avec ch_droite et ch_gauche.
ch_a_isoler   est une chaîne                // La variable qu'on veut isoler.
```

fen_calcul_formel

Code des champs

Clic sur bt_go

```
ch  est une chaîne
i, lg_gauche est un entier

s_compte_rendu_traitement = ""
s_solution = ""
ch_formule = sai_formule
ff_00_nettoie_formule()
ch = ff_01_verifie_syntaxe(ch_formule)
SI ch = code_erreur ALORS
    Erreur ( s_compte_rendu_traitement)
    RENVOYER Faux
FIN

POUR i = 1 A 26
    ch_formule = sai_formule
    ch_a_isoler = Caract(96+i)
    SI ChaîneOccurrence(ch_formule,ch_a_isoler)= 1 ALORS
        s_compte_rendu_traitement += [RC+RC]+ " traitement pour isoler " + ch_a_isoler
        s_compte_rendu_traitement += RC + ch_formule

        // Ma chaine a une syntaxe valide , donc il y a un seul signe =  ; je peux initialiser chaine_droite et chaine_gauche
        ch_gauche = ExtraitChaîne(ch_formule,  1, "=" )
        ch_droite = ExtraitChaîne(ch_formule,  2, "=" )
        SI Position( ch_formule,ch_a_isoler) > Position (ch_formule, "=") ALORS ff_02_permute_droite_gauche()

        lg_gauche = Taille(ch_gauche)
        TANTQUE lg_gauche > 1
            ch = ff_03_passe_a_droite()
            SI ch = code_erreur ALORS
                Erreur ( s_compte_rendu_traitement)
                RENVOYER Faux
```

```
            FIN
            ch = ff_04_parentheses_en_trop()
            SI ch = code_erreur ALORS
                Erreur ( s_compte_rendu_traitement)
                RENVOYER Faux
            FIN
            lg_gauche = Taille(ch_gauche)
            s_compte_rendu_traitement +=RC + ch_gauche +"=" + ch_droite
        FIN
        s_solution += [RC] + ch_gauche + "="+ ch_droite
    FIN

FIN

SI int_afficher_commentaires[1] ALORS
     zz_avertissement ( s_compte_rendu_traitement)
FIN
Info ( s_solution)
```

fen_calcul_formel

Procédures

Procédure locale ff_00_nettoie_formule

```
// Résumé : <nettoyage de ce qui a été saisi ; suppression des espaces.>
// Syntaxe :
// ff_00_nettoie_formule ()
//
// Paramètres :
//  Aucun
// Valeur de retour :
//  Aucune
//
// Exemple :
// Indiquez ici un exemple d'utilisation.
//
PROCEDURE ff_00_nettoie_formule()

// Je supprime les espaces.. et c'est tout pour cette version.
ch_formule = Remplace(ch_formule, " ", "")

// Rien à renvoyer, ch_formule est déclarée comme GLOBALE.
```

Procédure locale ff_01_verifie_syntaxe

```
// Résumé : <Je vérifie que la chaine proposée respecte les contraintes.>
// Syntaxe :
//[ <Résultat> = ] ff_01_verifie_syntaxe (<ch00>)
//
// Paramètres :
//  ch00 :<indiquez ici le rôle de ch_formule>
// Valeur de retour :
//  chaîne ANSI : // Chaine vide si Ok, Code d'erreur sinon.
//
// Exemple :
```

```
// Indiquez ici un exemple d'utilisation.
//
PROCEDURE ff_01_verifie_syntaxe(ch00)
idem  est un booléen
ch,  ch2 , c0  est une chaîne
ch = ch00
ch = Remplace ( ch, " " , "")  // Redondant ...

POUR i = 1 A 26
    c0= Caract(96+i)
    ch = Remplace(ch, c0 , "X")
FIN

// si j'ai des nombres, je les remplace par un X **********************
POUR i = 48 A 57
    c0= Caract(i)
    ch = Remplace(ch, c0 , "0")
FIN
ch = Remplace(ch, ".","0")
POUR i = 1 A 10
    ch = Remplace(ch, "00","0")
FIN
ch = Remplace(ch, "0","X")
// fin du traitement des nombres. ***********************************

TANTQUE idem = Faux
    ch2 = Remplace( ch, "(X+X)", "X")
    ch2 = Remplace( ch2, "(X-X)", "X")
    ch2 = Remplace( ch2, "(X/X)", "X")
    ch2 = Remplace( ch2, "(X*X)", "X")
    ch2 = Remplace( ch2, "(X)", "X")
    SI ch2 = ch ALORS idem = Vrai
    ch = ch2
FIN
SI ch = "X=X+X" ALORS RENVOYER ""
SI ch = "X=X-X" ALORS RENVOYER ""
SI ch = "X=X*X" ALORS RENVOYER ""
SI ch = "X=X/X" ALORS RENVOYER ""
SI ch = "X=X" ALORS RENVOYER ""

s_compte_rendu_traitement = "Erreur , la chaine fournie n'est pas valide. On ne sait pas traiter les chaines de type :  " + ch
RENVOYER code_erreur
```

```
// Résumé : <permute les chaines ch_droite et ch_gauche>
// Syntaxe :
// ff_02_permute_droite_gauche ()
//
// Paramètres :
//  Aucun
// Valeur de retour :
//  Aucune
//
// Exemple :
// Indiquez ici un exemple d'utilisation.
//
PROCEDURE ff_02_permute_droite_gauche()

ch_droite <=> ch_gauche
ch_formule = ch_gauche + "=" + ch_droite

s_compte_rendu_traitement += RC +  Complète(ch_formule,50) + " // permutation droite-gauche "
```

Procédure locale ff_03_passe_a_droite

```
// Résumé : <exemple : = partir de x+a=b, je génère x=b-a>
// Syntaxe :
//[ <Résultat> = ] ff_03_passe_a_droite ()
//
// Paramètres :
//  Aucun
// Valeur de retour :
//  Type indéterminé : //     Aucune
//
// Exemple :
// Indiquez ici un exemple d'utilisation.
//
PROCEDURE ff_03_passe_a_droite()

// J'ai ch_droite et ch_gauche
// ch_gauche est de type    chaine1+operateur+chaine2 ;  et soit chaine1, soit chaine2 contient ch_a_isoler
// Et donc, je vais faire passer à doire, soit chaine1, soit chaine2.
// Du coup, partant par exemple de chaine1+chaine2=chaine3,
// je vais passer à chaine1=(chaine3)-chaine2
```

```
// Etape1 , repérer l'opérateur central dans ch_gauche.

po_central, po_a_isoler  est un entier
caract_central , ch1 , ch2 est une chaîne

po_central = gg_rang_operateur_central(ch_gauche)
SI po_central = num_erreur ALORS RENVOYER num_erreur
caract_central = ch_gauche[[po_central]]
ch1 = ch_gauche[[ A po_central-1]]
ch2 = ch_gauche[[po_central+1 A ]]

//Je tiens le caractère central.
// Où se situe la variable à isoler par rapport à ce caractère central ?
po_a_isoler = Position ( ch_gauche, ch_a_isoler)

SI po_a_isoler > po_central ALORS          // ch2 contient la variable à isoler, on va passer ch1 à droite.
    SELON caract_central
        CAS "+"
            ch_gauche = ch2
            ch_droite = ff_03b_prth(ch_droite) + "-" + ch1
        CAS "-"
            ch_gauche = ch2
            ch_droite = ch1 + "-" + ff_03b_prth(ch_droite)
        CAS "*"
            ch_gauche = ch2
            ch_droite = ff_03b_prth(ch_droite) + "/" + ch1
        CAS "/"
            ch_gauche = ch2
            ch_droite = ch1 + "/" +ff_03b_prth(ch_droite)
        AUTRE CAS
            s_compte_rendu_traitement+=RC+ "ERREUR : Le séparateur n'est pas une des 4 opérations, bug programme" + RC + "séparateur :[" +
            caract_central + "]"
            RENVOYER code_erreur

    FIN
SINON
    SELON caract_central
        CAS "+"
            ch_gauche = ch1
            ch_droite = ff_03b_prth(ch_droite)+ "-" + ch2
        CAS "-"
            ch_gauche = ch1
```

```
            ch_droite = ff_03b_prth(ch_droite) + "+" + ch2
        CAS "*"
            ch_gauche = ch1
            ch_droite = ff_03b_prth(ch_droite)+ "/" + ch2
        CAS "/"
            ch_gauche = ch1
            ch_droite = ff_03b_prth(ch_droite)+ "*" + ch2
        AUTRE CAS
            s_compte_rendu_traitement+=RC+ "ERREUR : Le séparateur n'est pas une des 4 opérations, bug programme"+ RC + "séparateur :[" +
            caract_central + "]"
            RENVOYER code_erreur

    FIN
FIN


RENVOYER ""
```

Procédure locale ff_03b_prth

```
// Résumé : <indiquez ici ce que fait la procédure>
// Syntaxe :
//[ <Résultat> = ] ff_03b_prth (<ch0>)
//
// Paramètres :
//  ch0 :<indiquez ici le rôle de ch_droite>
// Valeur de retour :
//  Type indéterminé : //     Aucune
//
// Exemple :
// Indiquez ici un exemple d'utilisation.
//
PROCEDURE ff_03b_prth(ch0)

//  si ch0 est 'composite', j'ajoute des parenthèses ALORS
SI Position( ch0, "+") ALORS RENVOYER "("+ch0 + ")"
SI Position( ch0, "-") ALORS RENVOYER "("+ch0 + ")"
SI Position( ch0, "/") ALORS RENVOYER "("+ch0 + ")"
SI Position( ch0, "*") ALORS RENVOYER "("+ch0 + ")"
```

```
// sinon je renvoie la chaine originale
RENVOYER ch0
```

Procédure locale ff_04_parentheses_en_trop

```
// Résumé : <indiquez ici ce que fait la procédure>
// Syntaxe :
//[ <Résultat> = ] ff_04_parentheses_en_trop ()
//
// Paramètres :
//  Aucun
// Valeur de retour :
//  chaîne ANSI : // Aucune
//
// Exemple :
// Indiquez ici un exemple d'utilisation.
//
PROCEDURE ff_04_parentheses_en_trop()
// Suppression des parenthèses en trop.
ch_gauche = ff_04a_prth(ch_gauche)
ch_droite = ff_04a_prth(ch_droite)  // ne devrait rien changer.
RENVOYER ""
```

Procédure locale ff_04a_prth

```
// Résumé : <Suppression des parenthèses en trop.>
// Syntaxe :
//[ <Résultat> = ] ff_04a_prth (<chg>)
//
// Paramètres :
//  chg :<chg est la chaine à traiter >
// Valeur de retour :
//  Type chaine : // On renvoie généralement la chaine initiale, ou bine une chaine sans les 1er et dernier caractère
// Exemple :
// Indiquez ici un exemple d'utilisation.
//ff_04a_prth ("(a+b)")  renvoie "a+b"
//ff_04a_prth ("(a+b)*(c+d)")  renvoie la chaine inchangée.
PROCEDURE ff_04a_prth(chg)
```

```
// Suppression des parenthèses en trop.
// On renvoie une chaine
t ,i, k  est un entier
t = Taille(chg)
prth_fermantes est un tableau de t entiers
prth_ouvrantes est un tableau de t entiers
tb_en_attente  est un tableau de 0 entiers

// A partir d'une chaine comme ((a+1)*c)*(b+2), j'aurai prth_fermante[1]=9  ; prth_fermante[2]=6 ; prth_fermante[11]=15   et prth_fermante[*]=0 pour
les autres.

// Et bien sur,prth_ouvrante donne l'inverse : prth_fermante[9]=1  ; prth_fermante[6]=2 ; prth_fermante[15]=11

// Si la chaine commence par une parenthèse, et finit par une parenthèse, et que ces 2 parenthèses sont ''soeurs
'' l'une de l'autre, alors, je les supprime. Et éventuellement, je boucle.

POUR i = 1 A t
    SI chg[[i]]= "(" ALORS
        TableauAjouteLigne(tb_en_attente, i)
    FIN
    SI chg[[i]]= ")" ALORS
        k = TableauOccurrence(tb_en_attente)
        SI k < 1 ALORS
            s_compte_rendu_traitement += [RC] + " Problème, ')' sans '(' correspondante !" + "[[[" + chg + "]]]"
            RENVOYER num_erreur
        FIN
        prth_ouvrantes[i] = tb_en_attente[k]
        prth_fermantes[ tb_en_attente[k] ] = i
        TableauSupprime(tb_en_attente,k)
    FIN
FIN
SI TableauOccurrence(tb_en_attente) > 0 ALORS
    s_compte_rendu_traitement += [RC] + " Problème, '(' sans ')' correspondante !" + "[[[" + chg + "]]]"
    RENVOYER num_erreur
FIN

SI prth_fermantes[1] = t ALORS
    RENVOYER ff_04a_prth(chg[[ 2 A t-1]]   )  // Suppression du 1er et du dernier caractère, et je rappelle la même fonction, au cas où.
SINON
    RENVOYER chg
FIN
```

Procédure locale gg_rang_operateur_central

```
// Résumé : <A partir d'une chaine de type (a+x)*(b+y), je détermine où se trouve l'opérateur central (* dans cet exemple) >
// Syntaxe :
//[ <Résultat> = ] gg_rang_operateur_central (<chg>)
//
// Paramètres :
//  chg :<La chaine à analyser>
// Valeur de retour :
//  Type entier : // la position du caractère 'central'.
//
// Exemple :
// Indiquez ici un exemple d'utilisation.
//
PROCEDURE gg_rang_operateur_central(chg )

// chg est une chaine de type (a+1)*2  ou b*((c+2)+3) ou encore (a+2)*(2+b)
// Et je cherche l'emplacement de l'opérateur central (celui qui n'est pas encadré de parenthèses)
// C'est * dans ces 3 exemples.
//
// On va donc devoir repérer les couples de parenthèses ouvrante/fermante
// Et ensuite , 2 cas.
// - Soit le premier caractère de la chaine est une parenthèse ouvrante, et dans ce cas, le caractère central est celui qui suit la parenthèse
fermante correspondante.

// - Soit le premier caractère de la chaine n'est pas une parenthèse ouvrante, et dans ce cas, le caractère centrale est le caractère n°2.
//   - Correctif, la chaine peut être de type 13+a ... et dans ce cas, l'opérande est en position 3 et non 2.
//
// avant d'entrer dans cette procédure, j'ai fait en sorte de ne pas avoir de parenthèse surnuméraires, du type ((a+b)*c) ... que j'aurai
préalablement remplacé par (a+b)*c

//
// Et surtout, grace à la toute première procédure de controle, j'ai l'assurance d'avoir une chaine valide.

t ,i, k , prem0, prem1   est un entier
t = Taille(chg)
prth_fermantes est un tableau de t entiers
prth_ouvrantes est un tableau de t entiers
tb_en_attente  est un tableau de 0 entiers

// A partir d'une chaine comme ((a+1)*c)*(b+2), j'aurai prth_fermante[1]=9  ; prth_fermante[2]=6 ; prth_fermante[11]=15   et prth_fermante[*]=0 pour
les autres.

// Et bien sur,prth_ouvrante donne l'inverse : prth_fermante[9]=1  ; prth_fermante[6]=2 ; prth_fermante[15]=11
```

```
// Et après, il suffit de renvoyer  prth_fermante[1]+1    , ou bien 2.

POUR i = 1 A t
    SI chg[[i]]= "(" ALORS
        TableauAjouteLigne(tb_en_attente, i)
    FIN
    SI chg[[i]]= ")" ALORS
        k = TableauOccurrence(tb_en_attente)
        SI k < 1 ALORS
            s_compte_rendu_traitement += [RC] + " Problème, ')' sans '(' correspondante !" + "[[[" + chg + "]]]"
            RENVOYER num_erreur
        FIN
        prth_ouvrantes[i] = tb_en_attente[k]
        prth_fermantes[ tb_en_attente[k] ] = i
        TableauSupprime(tb_en_attente,k)
    FIN
FIN
SI TableauOccurrence(tb_en_attente) > 0 ALORS
    s_compte_rendu_traitement += [RC] + " Problème, '(' sans ')' correspondante !" + "[[[" + chg + "]]]"
    RENVOYER num_erreur
FIN

SI prth_fermantes[1] > 0 ALORS RENVOYER prth_fermantes[1] +1

// en principe je renvoie 2, sauf si la chaine est de type 123+a  , du coup, j'ajoute cette ligne de code :
prem1 = 9999
prem0  = Position(chg ,"+") ; SI prem0 > 0 ALORS prem1 = Min ( prem1, prem0)
prem0  = Position(chg ,"-") ; SI prem0 > 0 ALORS prem1 = Min ( prem1, prem0)
prem0  = Position(chg ,"*") ; SI prem0 > 0 ALORS prem1 = Min ( prem1, prem0)
prem0  = Position(chg ,"/") ; SI prem0 > 0 ALORS prem1 = Min ( prem1, prem0)
SI prem1 = 9999 ALORS
    s_compte_rendu_traitement += [RC] + " Problème, pas d'opérateur +/* ni - dans la chaine :" + "[[[" + chg + "]]]"
    RENVOYER num_erreur
FIN

RENVOYER prem1
```

Procédure locale zz_avertissement

```
// Résumé : <indiquez ici ce que fait la procédure>
// Syntaxe :
```

```
//zz_avertissement (<xy>)
//
// Paramètres :
//  xy :<indiquez ici le rôle de Param1>
// Valeur de retour :
//  Aucune
//
// Exemple :
// Indiquez ici un exemple d'utilisation.
//
//  s_compte_rendu_traitement : <indiquez ici le rôle de s_compte_rendu_traitement>
PROCEDURE zz_avertissement(xy)
Trace ( Répète( " -=- ",10))
Trace (" Inversion de " + sai_formule)
Trace ( " ")

POUR TOUTE CHAÎNE xx DE xy SEPAREE PAR RC
    Trace (xx)
FIN
Trace ( Répète( " -=- ",10))
```