

WINTASK

**Développer des scripts
d'automatisation fiables et
efficaces**

Version 3.8



AUTOMATE POUR WINDOWS XP, 2003, 2008, Vista et Windows 7

Publié par : TaskWare
18, allée des Lilas
92150 SURESNES France

© Copyright 1997-2011 TaskWare Octobre 2011

WinTask™ est une marque déposée de TaskWare.

TABLE DES MATIERES

Introduction	5
WinTask, pour quoi faire ?	5
Les modules de WinTask	5
TASKEDIT	5
TASKCOMP	5
TASKEEXEC	6
ESPION	6
TASKWIZ	6
EDITEUR DE BOÎTES DE DIALOGUE	6
Mode Enregistrement	7
Exercice 1	7
Mythe des outils de Capture/Rejoue	8
Exercice 2	8
Un Script simple d'automatisation	9
Exercice 3	11
Editeur WinTask	13
Exercice 4	13
Arrêter l'exécution d'un script	14
Synchronisation	15
Méthodes de Synchronisation	15
Exercice 5	17
Exercice 6	21
Le langage WinTask	29
Instructions du langage	29
Variables	30
Variables directives	31
Entiers	31
Chaînes de caractères	31
Nombres réels	31
Tableaux	32
Opérateurs	32
Instructions de gestion des fenêtres	35
#IgnoreErrors	35
#ActionTimeout	35
#UseExact	36
ExistW()	37
Focus\$()	37
Top\$()	37
Exercice 7	38
Itération	41
Itération	41
Exercice 8	41
Instructions de gestion des fichiers	43
Exist()	43
Kill()	43
Read()	44

Eof()	44
Write()	45
Exercice 9	45
ReadExcel et WriteExcel	46
Exercice 10	47
Fonctions et Sous-Programmes (Sub)	49
Sub...EndSub	49
Convertir un script en sous-programme (Sub)	49
Exercice 11	50
Function...EndFunction.....	51
Exercice 12	53
Assembler les morceaux.....	54
Exercice 13	54
Débugage.....	56
Erreurs de compilation.....	56
Erreurs d'exécution	56
Exercice 14	56
Exercice 15	58
Exécution en mode Débugage	58
Exercice 16	69
Conclusion	72
APPENDICE A	74
Barre d'outils de l'Editeur WinTask.....	74
APPENDICE B	76
Barre d'outils WinTask flottante.....	76
APPENDICE C	78
Solutions des exercices	78
Exercice 7, Script07a.src	78
Exercice 7, Script07b.src	79
Exercice 7, Script07c.src	80
Exercice 7, Script07d.src	81
Exercice 8, Script08a.src	82
Exercice 8, Script08b.src	83
Exercice 9, Script09.src.....	84
Exercice 10, Script10.src	85
Exercice 11, Script11.src	86
Exercice 12, Script12.src	88
Exercice 13, Script13.src	90
Exercice 14, Script14.src	93
Exercice 15, Script15.src	94
Glossaire	96
Index	98

Introduction

Ce manuel a été écrit pour vous aider dans le développement de vos scripts d'automatisation. Nous avons cherché à vous faire part de notre expérience issue de l'écriture de nombreux scripts et des questions les plus usuelles transmises à notre support technique. Le manuel couvre l'interface WinTask et les instructions du langage d'automatisation.

Ce manuel inclut de nombreux exercices pour illustrer le propos et montrer comment éviter les erreurs les plus courantes. Les scripts d'automatisation utilisent les applications Notepad™ et WordPad™ incluses dans Windows. Les techniques présentées s'appliquent ensuite sans difficultés à vos applications Windows spécifiques. Ce manuel ne traite pas l'automatisation de sites Web, sujet traité dans un autre manuel "Livre WinTask pour le Web" que vous pouvez télécharger de www.wintask.fr/manuels-wintask.php.

WinTask, pour quoi faire ?

WinTask est une solution complète d'automatisation pour Windows. Le logiciel peut automatiser n'importe quelle application Windows (même dans une boîte Dos) ou pages Web. WinTask est plus qu'un simple enregistreur de macros. Le script d'automatisation peut être enrichi grâce aux 300 instructions du langage. WinTask inclut un Planificateur de tâches (non disponible sous Vista/Windows 7/Windows 2008 mais vous pouvez utiliser le Planificateur livré avec ces versions de Windows) permettant d'ouvrir un bureau Windows automatiquement, d'exécuter le script et de refermer le bureau une fois la tâche terminée.

Les modules de WinTask

WinTask inclut plusieurs modules. Chacun est discuté plus en détails dans la suite et les exercices montrent comment les utiliser.

TASKEDIT

TaskEdit est l'environnement de développement des scripts WinTask. Grâce à cet Editeur, l'utilisateur peut modifier les instructions générées automatiquement par le mode Enregistrement et enrichir le script à l'aide des nombreuses instructions du langage. L'Editeur comporte plusieurs fenêtres, la syntaxe des mots-clés est en couleurs et l'aide sur chaque instruction est disponible en double-cliquant sur son nom. Les fichiers Script écrits dans cet environnement de développement sont ensuite compilés puis exécutés afin de rejouer les actions décrites dans le script. Les fichiers Script de WinTask sont au format ASCII (ou Unicode) et portent l'extension *.SRC*.

TASKCOMP

Le module Compilateur de WinTask est un *.EXE* qui, via une ligne de commande, compile un script WinTask en un fichier exécutable par le module d'exécution de WinTask. Le Compilateur peut également être lancé à partir de l'Editeur. Le Compilateur vérifie la syntaxe des instructions listées dans le fichier *.SRC* et génère un fichier d'extension *.LST* si des erreurs de syntaxe sont trouvées. Si aucune erreur de compilation n'est trouvée, le Compilateur génère un fichier prêt à être exécuté, fichier

binaire d'extension *.ROB*. Un fichier *.LST* est un fichier ASCII contenant les erreurs et les avertissements renvoyés par le Compilateur.

TASKEEXEC

C'est le module d'exécution. Il peut être appelé via une ligne de commande ou directement via l'Editeur. Il exécute ligne par ligne les instructions rencontrées dans le fichier d'extension *.ROB*, résultat de la compilation du script source.

ESPION

Le module Espion permet de connaître la structure interne des différents objets affichés sur le bureau Windows. La fenêtre ou contrôle sur lequel doit s'effectuer l'action est spécifiée par cette information renvoyée par l'Espion. Cette information, nom de la fenêtre ou nom du contrôle, est utilisée par de nombreuses instructions du langage WinTask.

TASKWIZ

Ce module regroupe les outils de synchronisation et de capture de données. Ils peuvent être utilisés lors de l'enregistrement de vos actions afin d'introduire une attente pour être sûr que l'écran suivant a bien fini de s'afficher avant d'effectuer une nouvelle action. Ces assistants de synchronisation sont décrits en détail dans le chapitre **Synchronisation**.

EDITEUR DE BOÎTES DE DIALOGUE

Dans l'Editeur WinTask, un éditeur de boîtes de dialogue est fourni : le script d'automatisation peut ainsi faire afficher de telles boîtes lors de l'exécution d'un script. Les informations saisies dans la boîte sont alors transmises au script en cours d'exécution. La création et l'appel de boîtes de dialogue sont expliqués au chapitre **Boîtes de dialogue**.


Mode Enregistrement

Pour automatiser rapidement des actions dans une application Windows, le mode Enregistrement de WinTask permet d'enregistrer ses saisies clavier et clics souris dans un script. Le mode Enregistrement est accessible depuis l'Editeur. Une fois le script généré, il est enregistré et son exécution peut être lancée.

Exercice 1

Cet exercice montre comment enregistrer les actions utilisateur. L'application à automatiser est Notepad.


1. Démarrez WinTask en cliquant sur le bouton **Démarrer** de Windows puis en sélectionnant **Tous les programmes/WinTask/WinTask**. Si l'Assistant de Création de Script s'affiche, cliquez sur le bouton **Annuler**.


2. La fenêtre de l'Editeur WinTask s'affiche, cliquez sur l'icône **Enreg**  de la barre d'outils.


3. La boîte de dialogue **Démarrer l'enregistrement** s'affiche, cliquez sur le bouton **OK**.


4. La boîte de dialogue **Lancement d'une application** s'affiche. Tapez **Notepad** dans le champ **Programme** et cliquez sur le bouton **OK**.

5. La fenêtre notepad s'ouvre, la fenêtre de l'Editeur est minimisée et la **barre**

d'outils WinTask  s'affiche.

Une icône clignotante  7:33 PM est également affichée dans la zone de notification de la barre d'états de Windows pour rappeler que le mode Enregistrement est actif. Dans la fenêtre de notepad, saisissez **Bonjour**.

6. Puis dans notepad, sélectionnez le menu **Fichier/Quitter**.
7. Cliquez sur le bouton **Ne pas enregistrer** (ou sur le bouton **Non** sous XP/2003) quand Notepad affiche la boîte de dialogue **Voulez-vous enregistrer les modifications**.
8. Arrêtez le **Mode Enregistrement** en cliquant sur l'icône  première icône de la barre d'outils flottante de WinTask.
9. La fenêtre de l'Editeur revient au premier plan et le script généré par le mode Enregistrement est affiché.

10. Cliquez sur l'icône **Exéc**  de la barre d'outils pour lancer l'exécution du script. Une boîte de dialogue vous demande de donner un nom à ce script, donnez le nom **script01** (l'extension .SRC est ajoutée automatiquement). Cliquez sur le bouton **Enregistrer** après avoir donné un nom. Vous voyez alors toutes vos actions se rejouer.
11. Quittez WinTask en sélectionnant dans la fenêtre de l'Editeur le menu **Fichier/Quitter**.

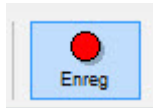
Mythe des outils de Capture/Rejoue

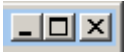
A première vue, le mode Enregistrement apparaît suffisant pour automatiser des tâches sous Windows. Mais le rejoue ne se fait pas aussi facilement dès lors que l'environnement Windows varie.



Exercice 2

Dans cet exercice, WinTask est utilisé en mode Capture/Rejoue sans se poser de questions. Et le script généré ne se rejoue pas correctement !

1. Démarrez WinTask. Si l'Assistant de Création de Script s'affiche, cliquez sur le bouton **Annuler**. La barre de titres de l'Editeur est **WinTask – [SansNom1]**. Sinon sélectionnez le menu **Fichier/Nouveau** ou cliquez sur l'icône **Nouveau** de la barre d'outils.



2. Cliquez sur l'icône **Enreg** de la barre d'outils pour démarrer le mode Enregistrement.
3. La boîte de dialogue **Démarrer l'enregistrement** s'affiche, cliquez sur le bouton **OK**.
4. La boîte de dialogue **Lancement d'une application** s'affiche. Tapez **Notepad** dans le champ **Programme** et cliquez sur le bouton **OK**.
5. La fenêtre notepad s'ouvre, la fenêtre de l'Editeur est minimisée et la **barre d'outils WinTask** s'affiche. Dans la fenêtre de notepad, saisissez **Bonjour** et appuyez sur la touche **Entrée**.
6. Puis dans notepad, sélectionnez le menu **Edition/Heure/Date**.
7. Dans notepad, cliquez sur le bouton Fermeture en haut à droite .
8. Cliquez sur le bouton **Enregistrer** (ou le bouton **Oui** sous Windows XP/2003) quand Notepad affiche la boîte de dialogue **Voulez-vous enregistrer les modifications**.
9. Enregistrez le fichier sous le nom **test02.txt** dans le répertoire courant.

10. Arrêtez le **Mode Enregistrement** en cliquant sur l'icône  première icône de la barre d'outils flottante de WinTask.
11. La fenêtre de l'Editeur revient au premier plan et le script généré par le mode Enregistrement est affiché.
12. Cliquez sur l'icône **Exéc**  de la barre d'outils pour lancer l'exécution du script. Une boîte de dialogue vous demande de donner un nom à ce script. donnez le nom **script02** et cliquez sur le bouton **Enregistrer**. Les résultats de la **Compilation** sont affichées dans la fenêtre Résultats, fenêtre du bas de l'Editeur, puis l'exécution du script compilé démarre.
13. Vous voyez les actions enregistrées dans les étapes 5 à 8 se rejouer. Mais notepad n'enregistre pas le fichier car une boîte de dialogue de confirmation apparaît maintenant.
14. Cliquez sur le bouton **Non** de la boîte de dialogue **Confirmer l'enregistrement**, cliquez sur le bouton **Annuler** de la boîte de dialogue **Enregistrer sous**, et fermez notepad. Cliquez sur le bouton **Ne pas enregistrer** quand Notepad affiche la boîte de dialogue **Voulez-vous enregistrer les modifications**.
15. Questions : pourquoi le script ne s'est pas rejoué correctement ? Comment le modifier pour que l'exécution aille à son terme ?

Quand vous avez enregistré vos actions dans le script, le fichier test02.txt n'existait pas. Lors du rejoue du script, la boîte de dialogue Enregistrer sous a détecté qu'un fichier portant le même nom existait déjà et donc une nouvelle boîte de dialogue s'est affichée pour demander s'il fallait écraser l'ancien. Le script ne gérant pas ce cas, le rejoue ne s'est pas effectué correctement.

Sans des ajustements manuels, un script généré automatiquement par le mode Enregistrement ne se rejouera correctement que si l'environnement est exactement identique à celui lors de l'enregistrement des actions. Dans l'exemple de l'exercice 2, le fichier test02.txt doit être supprimé avant de rejouer le script. Pour ce faire, soit le fichier est supprimé manuellement, soit le script est modifié pour qu'il supprime lui-même le fichier s'il existe.

Un Script simple d'automatisation

Jusqu'à maintenant, nous n'avons pas étudié le contenu du script généré par le mode Enregistrement. Dans ce paragraphe, nous allons examiner la signification des différentes lignes générées dans l'exercice 2, le script script02.src. Les lignes affichées dans le fichier script sont en italiques et sont suivies d'une explication.

L'utilisateur peut visualiser le fichier script dans la fenêtre de l'Editeur WinTask. En mettant le curseur texte sous le nom d'une instruction et en appuyant sur la touche F1, l'aide pour cette instruction s'affiche.

Shell("Notepad", 1)

Shell lance l'application spécifiée. Le deuxième paramètre indique si l'application démarre avec sa taille par défaut, ou iconisée ou maximisée. Cette première ligne du script démarre notepad à sa taille par défaut.

UseWindow("NOTEPAD.EXE/Edit/Sans titre - Bloc-notes/1", 1)

UseWindow spécifie la fenêtre cible vers laquelle les actions suivantes dans le script sont envoyées. Cette instruction attend que la fenêtre soit affichée avant de continuer. Le délai d'attente par défaut est de 30 secondes. Si au bout de 30 secondes, la fenêtre spécifiée n'est pas présente, une erreur d'exécution WinTask est affichée. Cette ligne attend donc que notepad soit complètement lancée et la fenêtre notepad est mise au premier plan.

SendKeys("Bonjour<Enter>")

SendKeys envoie la chaîne de caractères spécifiée à la fenêtre spécifiée par l'instruction *UseWindow* précédente. Les touches spéciales sont désignées par un mnémonique et entourées des signes < >. Cette ligne saisit **Bonjour** dans notepad et appuie sur la touche **Entrée**.

UseWindow("NOTEPAD.EXE/Notepad/Sans titre - Bloc-notes", 1)

Cette instruction *UseWindow* indique que les prochaines actions doivent maintenant être envoyées à la fenêtre de menu de notepad.

ChooseMenu("Normal ", &Edition/He&ure/Date F5")

ChooseMenu sélectionne l'option de menu spécifiée, menu présent dans la fenêtre spécifiée par le *UseWindow* précédent. Cette ligne sélectionne le sous-menu **Heure/Date** sous le menu **Edition**. Notez l'utilisation du symbole & dans *&Edition* et dans *He&ure* pour indiquer que la touche de raccourci clavier est <ALT>+u pour **Heure/Date** et <ALT>+e pour **Edition**.

CloseWindow("NOTEPAD.EXE/Notepad/Sans titre - Bloc-notes", 1)

CloseWindow ferme la fenêtre spécifiée. Cette ligne ferme donc la fenêtre notepad.

UseWindow("NOTEPAD.EXE/CtrlNotifySink/Bloc-notes/7")

Cette ligne change de fenêtre cible pour les prochaines actions. Elles seront désormais envoyées à la boîte de dialogue notepad qui demande à l'utilisateur s'il veut enregistrer ses modifications. Sous XP/2003, le nom de fenêtre est différent : *UseWindow("NOTEPAD.EXE/#32770/Bloc-notes")*

Click(Button, "&Enregistrer")

L'instruction *Click* clique sur le bouton spécifié se trouvant dans la fenêtre spécifiée par l'instruction *UseWindow* précédente. Cette ligne clique sur le bouton **Enregistrer** de la boîte de dialogue Voulez-vous enregistrer les modifications. Notez le & devant Enregistrer pour indiquer que la touche de raccourci est E. Sous XP/2003, le bouton est Oui et donc la ligne est : *Click(Button, "&Oui")*

UseWindow("NOTEPAD.EXE/FloatNotifySink/Enregistrer sous/1",1)

Les prochaines actions seront désormais envoyées à la boîte de dialogue Enregistrer sous. Sous XP/2003, la ligne est
UseWindow("NOTEPAD.EXE/Edit/Enregistrer sous /1",1)

WriteCombo("1","test02.txt")

Cette ligne tape **test02.txt** dans le champ Nom du fichier de la boîte de dialogue Enregistrer sous. Sous XP/2003, la ligne est *SendKeys("test02.txt")*

Click(Button,"&Enregistrer ")

Cette ligne clique sur le bouton **Enregistrer**.

Exercice 3

Cet exercice illustre une autre cause très fréquente de rejoue incorrect d'un script généré par le mode Enregistrement.

1. Supprimez manuellement le fichier **test02.txt** créé à l'exercice 2.
2. Lancez manuellement Notepad (via le menu Démarrer de Windows).
3. Tapez **Bonjour** et appuyez sur la touche **Entrée**.
4. Fermez Notepad en utilisant le menu **Fichier/Quitter**.
5. Cliquez sur le bouton **Enregistrer** (ou le bouton **Oui** sous XP/2003) dans la boîte de dialogue **Voulez-vous enregistrer les modifications ?**.
6. Dans le champ Enregistrer sous, sélectionnez un autre répertoire que celui proposé par défaut et enregistrez le fichier dans ce répertoire sous le nom **test02.txt**. Le nom de fichier est le même que celui dans l'exercice 2.
7. Démarrez WinTask et dans la fenêtre de l'Editeur WinTask, cliquez sur l'icône **Ouvrir** de la barre d'outils (ou sélectionnez le menu **Fichier/Ouvrir**). Dans la boîte de dialogue **Ouvrir**, sélectionnez le script **script02.src** pour l'ouvrir.
8. Cliquez sur l'icône **Exéc** de la barre d'outils WinTask. Vous constatez une nouvelle fois que l'exécution ne va pas à son terme.
9. Question : pourquoi le script ne s'est pas rejoué correctement alors que le fichier **test02.txt** a été supprimé à l'étape 1 ?

Notepad comme beaucoup d'autres applications se souvient du répertoire dans lequel la dernière sauvegarde s'est effectuée. L'étape 6 de l'exercice 3 a modifié le répertoire par défaut de sauvegarde des fichiers notepad. Comme le script enregistre dans le répertoire par défaut, son rejoue a encore échoué.

Enregistrer la navigation du répertoire proposé au répertoire correct ne donne pas non plus des résultats fiables car la hiérarchie des répertoires peut changer d'une

exécution à une autre. Pour un rejoue fiable, le nom complet du fichier incluant son chemin complet doit être spécifié dans le script.

En conclusion, le mode Enregistrement à lui seul n'est pas capable de générer un script fiable et vous devez toujours penser à gérer dans le script les changements éventuels de l'environnement Windows au moment où le script se rejoue.

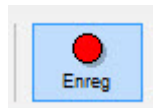
Editeur WinTask

L'Editeur de WinTask fournit un environnement de développement complet pour faciliter la création et la mise au point des scripts d'automatisation. A partir de l'Editeur, l'utilisateur peut enregistrer des actions, modifier les lignes générées, compiler le code et exécuter le script. Une fenêtre de résultats donne les messages de compilation, une autre fenêtre liste les instructions disponibles dans le langage. Une aide contextuelle sur chaque instruction est accessible en appuyant sur la touche F1 quand le curseur est sous l'instruction désirée, ou en double cliquant sur le nom de l'instruction dans la fenêtre du langage affichée à droite (l'appui sur la touche F4 fait apparaître cette fenêtre listant toutes les instructions si elle n'est pas affichée). Si plusieurs scripts sont ouverts simultanément, un clic sur l'onglet du script désiré met en premier plan celui-ci.


Exercice 4

Cet exercice illustre les possibilités de l'environnement de développement, l'Editeur WinTask.

1. Démarrez l'Editeur WinTask. Si l'Assistant de Création de Script s'affiche, cliquez sur le bouton **Annuler**. La barre de titres de l'Editeur est **WinTask – [SansNom1]**. Si WinTask ouvre un script réalisé avant (par exemple **script02.src**), fermez-le en sélectionnant le menu **Fichier/Fermer** puis sélectionnez le menu **Fichier/Nouveau** ou cliquez sur l'icône **Nouveau** de la barre d'outils.



2. Cliquez sur l'icône **Enreg** de la barre d'outils pour démarrer le mode Enregistrement. La boîte de dialogue **Démarrer l'enregistrement** s'affiche, cliquez sur le bouton **OK**.
3. La boîte de dialogue **Lancement d'une application** s'affiche. Tapez **notepad** dans le champ **Programme** et cliquez sur le bouton **OK**.
4. Dans Notepad, saisissez **Bonjour**.
5. A l'aide de la souris, sélectionnez le mot **Bonjour**.
6. Dans notepad, sélectionnez le menu **Format/Police** et changez la **Taille** à **20**. Cliquez sur le bouton **OK** pour fermer la boîte de dialogue **Police**.
7. Cliquez après le mot **Bonjour** et appuyez sur la touche **Entrée**.
8. Sur la deuxième ligne, tapez **WinTask** et appuyez sur la touche **Entrée**.
9. Sélectionnez le mot **WinTask**.

10. Dans Notepad, sélectionnez le menu **Format/Police** et changez le **Style** en **Gras**. Cliquez sur le bouton **OK** pour fermer la boîte de dialogue **Police**.
11. Arrêtez le **Mode Enregistrement** en cliquant sur l'icône  première icône de la barre d'outils flottante de WinTask.
12. Fermez notepad sans enregistrer.
13. La fenêtre de l'Editeur revient au premier plan et le script généré par le mode Enregistrement est affiché.
14. Le script contient deux lignes utilisant l'instruction **Chooseltem**. Mettez le curseur texte sous le mot Chooseltem et appuyez sur la touche **F1** pour appeler l'aide en ligne sur cette instruction. Lisez la rubrique d'aide et fermez la fenêtre d'aide.
15. Mettez le curseur texte au début de la dernière ligne du script.
16. Dans cette ligne blanche, écrivez : **MsgBox(Bonjour WinTask")** et appuyez sur la touche **Entrée** (tapez le texte exactement comme indiqué).
17. Mettez maintenant le curseur texte au début de la première ligne du script. Sélectionnez le menu **Edition/Remplacer**. Utilisez la boîte de dialogue **Remplacer** pour remplacer la taille de police **20** par **24** dans tout le script (donc **Chooseltem(Combo, "3", "20"** devient **Chooseltem(Combo, "3", "24"**).
18. Cliquez sur l'icône **Exéc** de la barre d'outils pour lancer l'exécution du script. Une boîte de dialogue vous demande de donner un nom à ce script. donnez le nom **script04** et cliquez sur le bouton **Enregistrer**.
19. Le Compilateur WinTask détecte une erreur et écrit dans la fenêtre Résultats de l'Editeur des messages. Double-cliquez sur le message d'erreur dans cette fenêtre Résultats, le curseur vient se positionner sur la ligne qui a provoqué l'erreur.
20. Mettez en commentaire cette dernière ligne qui a provoqué l'erreur en insérant au tout début de la ligne le caractère ' (simple quote). Notez que le texte de cette ligne passe en vert pour indiquer que c'est un commentaire. Vous remarquez également que le nom des instructions est en bleu.
21. Cliquez sur l'icône **Exéc** de la barre d'outils pour lancer l'exécution du script. Cette fois le script doit se compiler sans erreur et l'exécution démarre.
22. Dans l'Editeur, sélectionnez le menu **Fichier/Créer raccourci bureau**, mettez en réduction la fenêtre de l'Editeur, vous voyez sur votre bureau Windows un nouveau raccourci de nom **script04**. Si Notepad est ouvert, fermez-le sans enregistrer puis double-cliquez sur le raccourci, l'exécution se lance. Fermez Notepad.

Arrêter l'exécution d'un script

Pour arrêter un script en cours d'exécution, il faut appuyer sur les touches **Ctrl+Shift+Pause**, le script s'arrête une fois la ligne en cours d'exécution terminée.

Synchronisation

Lors de l'exécution d'un script, WinTask doit souvent attendre qu'un événement se produise avant d'exécuter l'action suivante. Dans les exercices précédents, le script devait attendre que notepad soit chargé avant de taper du texte ; quand notepad était fermé, le script devait attendre que la boîte de dialogue Enregistrer soit affichée avant de pouvoir taper le nom du fichier sous lequel enregistrer. Ces deux exemples de synchronisation montrent bien que ces techniques de synchronisation sont au coeur d'un jeu fiable d'un script d'automatisation.

WinTask effectue cette synchronisation via ses propres instructions qui attendent un événement Windows avant de poursuivre. Par exemple l'instruction UseWindow attend que la fenêtre spécifiée soit bien active avant de continuer. Et une erreur d'exécution est renvoyée si la fenêtre spécifiée n'est pas disponible après un temps d'attente (ce timeout est à 30 secondes par défaut).

Parfois, la synchronisation via UseWindow n'est pas suffisante, par exemple quand une action dans une fenêtre rafraîchit le contenu de la fenêtre sans modifier le titre de la fenêtre. Pour ces cas, WinTask inclut des assistants de synchronisation accessibles soit en mode Enregistrement via la barre d'outils flottante, soit directement dans la barre d'outils de l'Editeur (ou le menu **Insérer/Synchronisation**).

Méthodes de Synchronisation

WinTask inclut des méthodes de synchronisation sur six événements système et sur trois événements utilisateurs. Une synchronisation sur un événement système permet d'attendre que Windows ait fini une tâche. Une synchronisation sur un événement utilisateur attend la saisie d'informations par ce dernier. Un assistant est disponible pour chaque méthode de synchronisation. Si cet assistant est appelé de l'Editeur, le code généré est inséré là où le curseur texte se trouve. Si cet assistant est appelé via la barre d'outils flottante en mode Enregistrement, le code est inséré au bon endroit de la séquence des actions enregistrées.

L'aide WinTask fournit des indications supplémentaires sur chaque paramètre disponible dans ces méthodes de synchronisation. Consultez l'appendice A pour la signification des différentes icônes de synchronisation dans la barre d'outils de l'Editeur. Consultez l'appendice B pour les différentes icônes de synchronisation dans la barre d'outils flottante WinTask en mode Enregistrement.

- **Synchronisation sur Texte**

Cette synchronisation attend qu'un certain texte apparaisse quelque part dans une des fenêtres du bureau. Cette méthode de synchronisation est par exemple utilisée dans une fenêtre d'émulation où le texte change après avoir appuyé sur Entrée mais le nom de fenêtre ne change pas. Elle est également utilisée pour attendre un résultat dans une fenêtre de ligne de commande (fenêtre cmd).

L'assistant de synchronisation texte est appelé par le menu **Insérer/Synchronisation/Sur Texte**.

- **Synchronisation sur Texte OCR**

Cette synchronisation attend qu'un certain texte affiché dans un graphique apparaisse quelque part dans une des fenêtres du bureau. Cette méthode de synchronisation est par exemple utilisée pour attendre qu'un bouton représentant une icône et avec un texte dedans soit affiché dans une fenêtre dont le nom n'a pas changé après une action. L'assistant de synchronisation texte OCR est appelé par le menu **Insérer/Synchronisation/Sur Texte OCR**.

- **Synchronisation sur Image**

Cette synchronisation attend qu'une image (un bitmap) soit affichée quelque part dans une des fenêtres du bureau. Cette méthode de synchronisation est par exemple utilisée pour attendre qu'une icône soit affichée dans une fenêtre dont le nom n'a pas changé après une action. L'assistant de synchronisation sur Image est appelé par le menu **Insérer/Synchronisation/Sur Image**.

- **Synchronisation sur Fenêtre**

Cette synchronisation attend qu'une fenêtre devienne active ou qu'une fenêtre disparaisse. L'instruction UseWindow est la plupart du temps suffisante pour attendre qu'une fenêtre devienne active, la méthode de synchronisation sur fenêtre est par exemple utilisée pour attendre qu'une fenêtre de téléchargement de fichiers disparaisse avant de continuer. L'assistant de synchronisation sur Fenêtre est appelé par le menu **Insérer/Synchronisation/Sur Fenêtre**.

- **Synchronisation sur Durée**

C'est la méthode de synchronisation la plus simple, le script attend un intervalle de temps fixe avant de continuer. Cette durée peut s'exprimer en ticks (environ 1/100 sec), en secondes, minutes, heures, jours. Cette méthode de synchronisation est par exemple utilisée quand la validation de la saisie dans un champ fait apparaître quelques centièmes de secondes après un nouveau champ dans lequel le script doit saisir d'autres données. L'assistant de synchronisation sur Durée est appelé par le menu **Insérer/Synchronisation/Sur Durée**.

- **Synchronisation sur Date/Heure**

Cette synchronisation attend qu'il soit la date ou l'heure spécifiée. Cette méthode de synchronisation n'est plus guère utilisée, il est préférable d'utiliser le Planificateur WinTask pour lancer un script par exemple tous les jours à la même heure. L'assistant de synchronisation sur Date/Heure est appelé par le menu **Insérer/Synchronisation/Sur Date/Heure**.

- **Attente action clavier**

Cette synchronisation attend que l'utilisateur appuie sur une touche du clavier. Cette méthode de synchronisation permet par exemple de demander à l'utilisateur d'appuyer sur F1 une fois qu'il a monté un disque amovible, et le script ne reprend son exécution qu'une fois la touche F1 enfoncée. L'assistant d'attente action sur touche est appelé par le menu **Insérer/Attente action/Touche**.

- **Attente action menu**

Cette synchronisation attend que l'utilisateur sélectionne une option dans un menu. Cette méthode de synchronisation n'est plus guère utilisée. L'assistant d'attente action menu est appelé par le menu **Insérer/Attente action/Menu**.

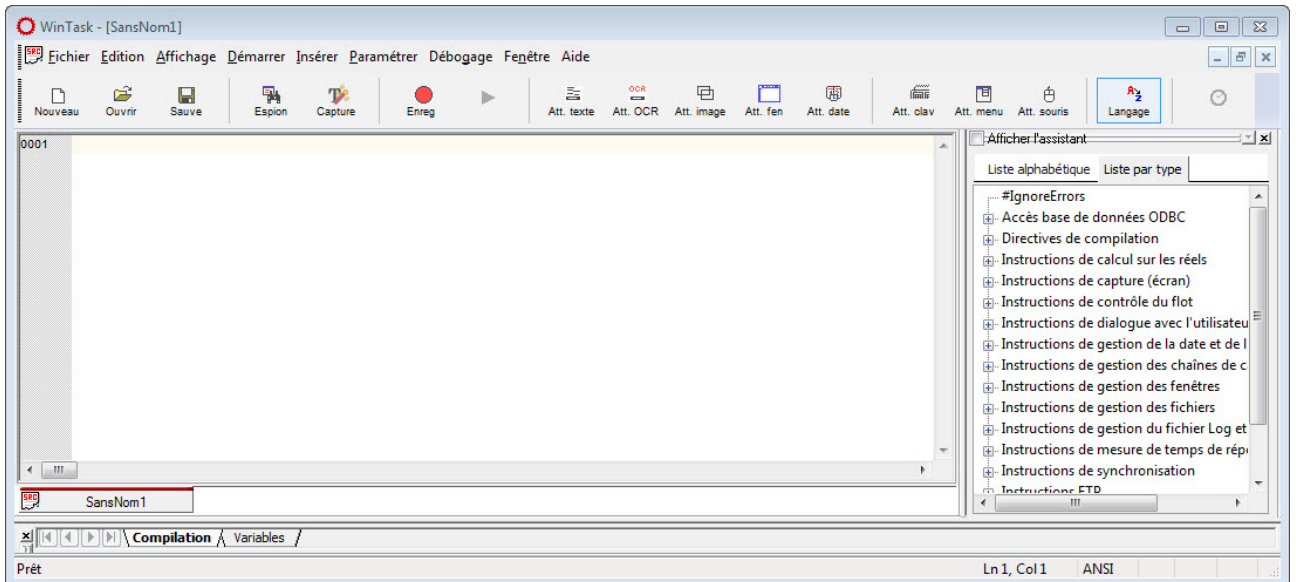
- **Attente action souris**

Cette synchronisation attend que l'utilisateur clique gauche ou droit, ou double-clique. Cette méthode de synchronisation est par exemple utilisée pour faire afficher un message dès que l'utilisateur clique dans une certaine fenêtre. L'assistant d'attente action souris est appelé par le menu **Insérer/Attente action/Souris**.

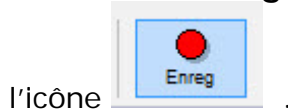
Exercice 5

Cet exercice illustre l'utilisation d'une Synchronisation Texte pour attendre le texte affiché dans la fenêtre Voulez-vous enregistrer les modifications quand vous fermez notepad. Cette synchronisation texte n'est pas nécessaire dans le cas précis de notepad mais vous pourrez utiliser les mêmes étapes pour insérer une telle synchronisation lors de l'automatisation d'une de vos applications quand il faut détecter un nouveau texte apparaissant dans une fenêtre qui ne change pas de nom.

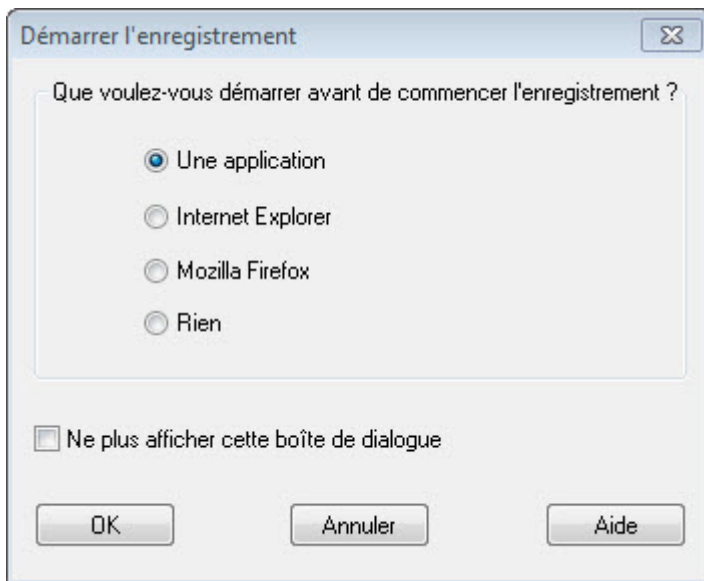
1. Démarrez l'Editeur WinTask. Si l'Assistant de Création de Script s'affiche, cliquez sur le bouton **Annuler**. La barre de titres de l'Editeur est **WinTask – [SansNom1]**. Si WinTask ouvre un script réalisé avant (par exemple **script04.src**), fermez-le en sélectionnant le menu **Fichier/Fermer** puis sélectionnez le menu **Fichier/Nouveau** ou cliquez sur l'icône **Nouveau** de la barre d'outils.
2. La fenêtre de l'éditeur s'affiche, fenêtre de titre SansNom1 comme ci-dessous



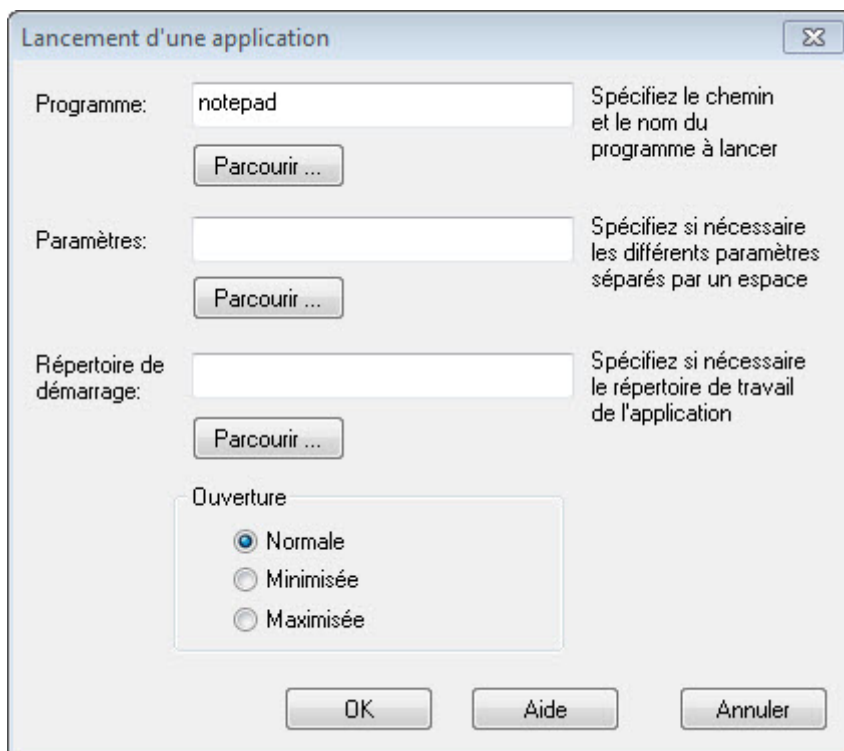
3. Démarrez l'enregistrement en sélectionnant l'option **Démarrer/Enregistrement en lançant une application** ou cliquez sur



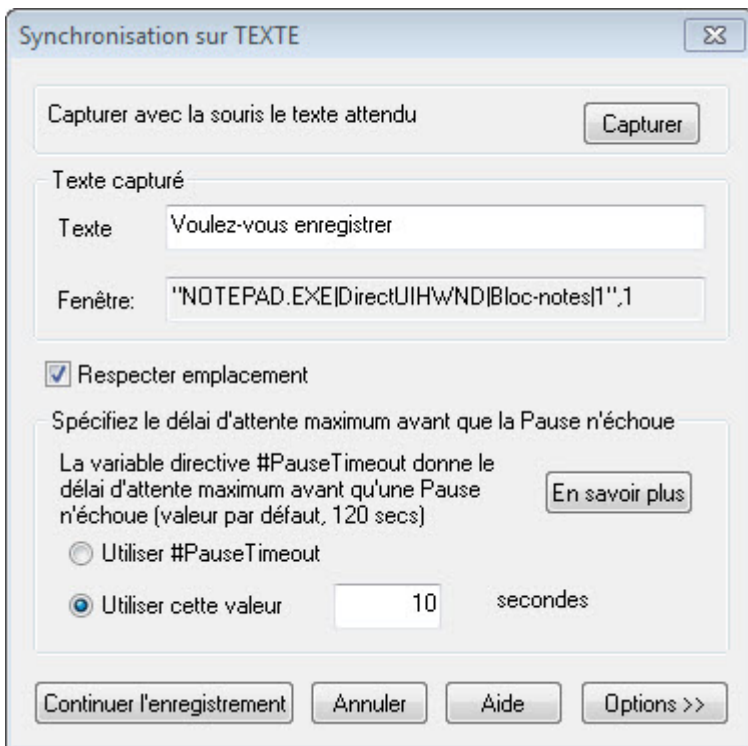
4. Dans la boîte de dialogue suivante, laissez cochée la case **Une application** et cliquez sur le bouton **OK** puisque nous voulons lancer notepad avant de commencer l'enregistrement




5. Dans la boîte de dialogue suivante, spécifiez le nom du programme à lancer, **notepad** et cliquez sur le bouton **OK**.



6. La fenêtre de l'éditeur se met en réduction, la fenêtre Notepad s'ouvre et l'icône représentant une caméra clignotante se retrouve en bas à droite dans la barre des tâches. Et la barre d'outils WinTask est également affichée.
7. Dans la fenêtre Notepad, tapez le texte : **Exemple de synchronisation Texte** puis fermez la fenêtre notepad en cliquant sur la petite croix de fermeture de la fenêtre (en haut à droite de la fenêtre Notepad).



12. Le champ Texte a été rempli par le texte capturé et sur lequel l'attente doit avoir lieu. Décochez la case Respecter emplacement, le texte sera recherché quelle que soit sa position dans la fenêtre. Cliquez sur le bouton **Continuer l'enregistrement**. Cliquez sur le bouton **Ne pas enregistrer** (sous Windows XP, cliquez sur le bouton **Non**).
13. Arrêtez le mode Enregistrement de l'automate en cliquant sur la première icône de la barre d'outils WinTask . La fenêtre de l'éditeur s'affiche et vous remarquez cet ensemble de lignes générées automatiquement par la synchronisation Texte :

Pause 10 secs until

Text("Voulez-vous enregistrer")

InWindow(""NOTEPAD.EXE|DirectUIHWND|Bloc-notes|1",1)


PauseFalse

MsgBox("L'attente à la ligne " + #ErrorLine\$ + " a échoué !")

End

EndPause

Une instruction Pause 10 secs until a été générée ; elle signifie attendre jusqu'à ce que le texte "Voulez-vous enregistrer" apparaisse dans la fenêtre dont le nom est donné par l'instruction InWindow. Si ce texte n'est pas trouvé au bout de 10 secondes, l'instruction suivant le terme PauseFalse est exécutée ; cette instruction affiche un message d'erreur en précisant le numéro de ligne dans le script où l'attente a échoué. Si le texte est trouvé dans l'intervalle des 10 secondes, dès qu'il est trouvé, les instructions suivant EndPause sont exécutées.

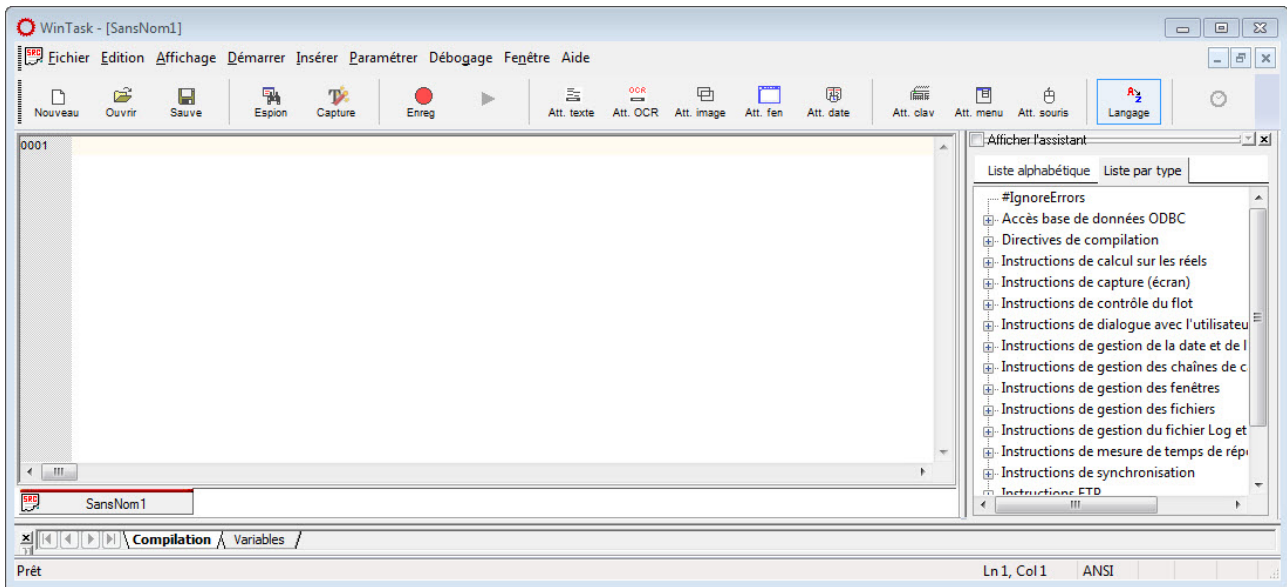
14. Lancez l'exécution du script ainsi généré en sélectionnant dans l'éditeur le menu **Démarrer/Exécution** ou l'icône . Une fenêtre vous demande d'enregistrer le script, donnez-lui le nom **Script05**, la compilation ne doit pas donner d'erreurs et le script rejoue les actions.

Si le texte recherché est intégré dans une icône, la capture du texte n'est pas possible. La synchronisation sur texte OCR doit alors être utilisée, méthode expliquée dans l'exercice suivant.

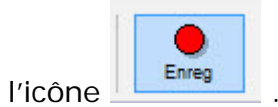
Exercice 6

Cet exercice reprend les mêmes actions à automatiser que l'exercice précédent mais cette fois, une synchronisation sur texte OCR est utilisée pour attendre le texte **Voulez-vous enregistrer**.

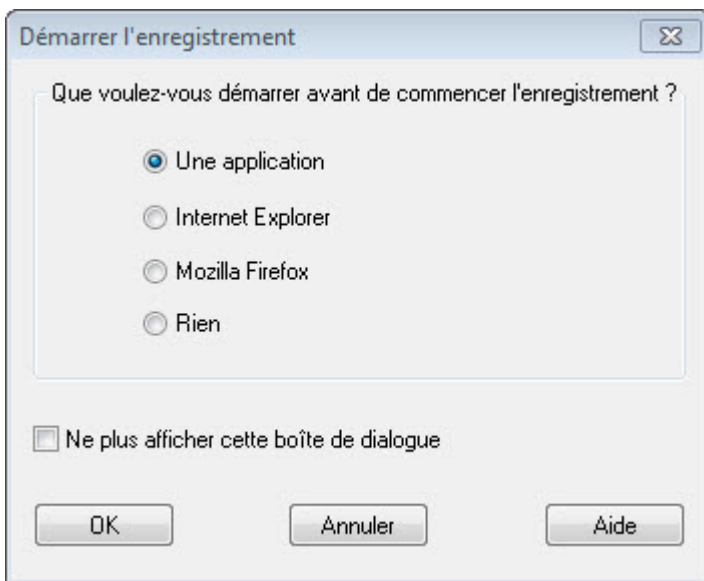
1. Démarrez l'Editeur WinTask. Si l'Assistant de Création de Script s'affiche, cliquez sur le bouton **Annuler**. La barre de titres de l'Editeur est **WinTask – [SansNom1]**. Si WinTask ouvre un script réalisé avant (par exemple **script05.src**), fermez-le en sélectionnant le menu **Fichier/Fermer** puis sélectionnez le menu **Fichier/Nouveau** ou cliquez sur l'icône **Nouveau** de la barre d'outils.
2. La fenêtre de l'éditeur s'affiche, fenêtre de titre SansNom1 comme ci-dessous



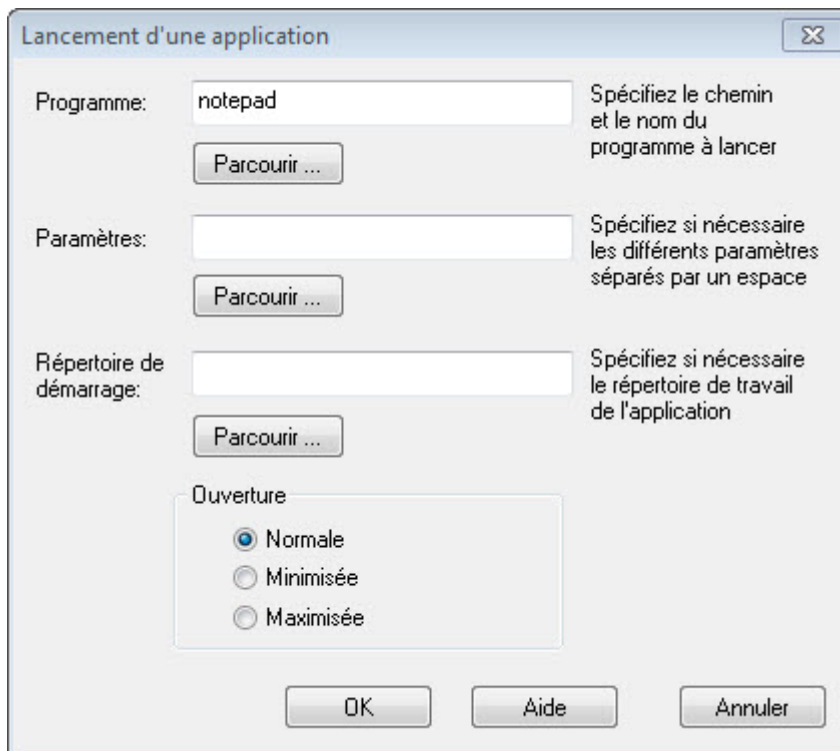
3. Démarrez l'enregistrement en sélectionnant l'option **Démarrer/Enregistrement en lançant une application** ou cliquez sur




4. Dans la boîte de dialogue suivante, laissez cochée la case **Une application** et cliquez sur le bouton **OK** puisque nous voulons lancer notepad avant de commencer l'enregistrement



5. Dans la boîte de dialogue suivante, spécifiez le nom du programme à lancer, **notepad** et cliquez sur le bouton **OK**.



6. La fenêtre de l'éditeur se met en réduction, la fenêtre Notepad s'ouvre et l'icône représentant une caméra clignotante se retrouve en bas à droite dans la barre des tâches. Et la barre d'outils WinTask est également affichée.
7. Dans la fenêtre Notepad, tapez le texte : **Exemple de synchronisation Texte OCR** puis fermez la fenêtre notepad en cliquant sur la petite croix de fermeture de la fenêtre (en haut à droite de la fenêtre Notepad).
8. La fenêtre de titre Bloc-notes contenant le texte "Voulez-vous enregistrer les modifications de Sans titre" s'affiche (sous Windows XP, ce texte est un peu différent et suivez les étapes suivantes en tenant compte de cet autre texte).
9. Appelez l'Assistant de Synchronisation sur Texte OCR en cliquant sur son icône dans la barre d'outils WinTask  (cinquième icône à partir de la gauche) de la barre d'outils WinTask puis sélectionnez **Synchronisation sur texte OCR**.
10. La boîte de dialogue Synchronisation Texte OCR s'affiche. Le champ **Sélectionnez le moteur OCR** permet de choisir le moteur OCR à utiliser ; choisissez le moteur **WinTask** ou si vous avez Office 2003 ou Office 2007 d'installé, sélectionnez **MODI** (moteur OCR livré avec Office, Microsoft Office Document Imaging). Le moteur OCR MODI est plus performant que le moteur OCR WinTask mais vous devez avoir Office 2003/2007.

Synchronisation Texte OCR

Moteur OCR
Sélectionnez le moteur OCR : WinTask

Capture
Cliquez sur le bouton "Capturer" puis à l'aide de la souris, dessinez un rectangle autour du texte sur lequel se synchroniser.

Fenêtre

Seulement dans cette zone (pas dans toute la fenêtre)

Texte analysé par le moteur OCR
Dans le champ "Texte" ci-dessous, le texte tel qu'analysé par le moteur OCR est affiché. Si le texte n'est pas parfait, sélectionnez quelques caractères correctement reconnus et collez-les dans le champ "Copiez du champ Texte".

Texte

Dans le champ "Texte analysé par le moteur OCR", sélectionnez le texte que vous désirez attendre dans cette synchronisation et collez-le dans le champ ci-dessous :

Vérier

Spécifiez le délai d'attente maximum avant que la Pause n'échoue
La variable directive #PauseTimeout donne le délai d'attente maximum avant qu'une Pause n'échoue (valeur par défaut, 120 secs)

Utiliser #PauseTimeout
 Utiliser cette valeur : 10 secondes

En savoir plus

Continuer l'enregistrement Annuler Aide Options >>

11. Cliquez sur le bouton **Capturer**. La forme du pointeur souris change. A l'aide de la souris, entourez d'un rectangle la phrase "Voulez-vous enregistrer". Ce texte tel que compris par le moteur OCR et sur lequel l'automate doit attendre avant de continuer se trouve reporté dans le champ Texte de la fenêtre Synchronisation Texte OCR :

Synchronisation Texte OCR ✕

Moteur OCR
Sélectionnez le moteur OCR : WinTask

Capture
Cliquez sur le bouton "Capturer" puis à l'aide de la souris, dessinez un rectangle autour du texte sur lequel se synchroniser. Capturer

Fenêtre "NOTEPAD.EXE|DirectUIHWND|Bloc-notes|1",1

Seulement dans cette zone (pas dans toute la fenêtre)

Texte analysé par le moteur OCR
Dans le champ "Texte" ci-dessous, le texte tel qu'analysé par le moteur OCR est affiché. Si le texte n'est pas parfait, sélectionnez quelques caractères correctement reconnus et collez-les dans le champ "Copiez du champ Texte".

Texte
Voulez vous en register

Dans le champ "Texte analysé par le moteur OCR", sélectionnez le texte que vous désirez attendre dans cette synchronisation et collez-le dans le champ ci-dessous :

Vérier

Spécifiez le délai d'attente maximum avant que la Pause n'échoue
La variable directive #PauseTimeout donne le délai d'attente maximum avant qu'une Pause n'échoue (valeur par défaut, 120 secs) En savoir plus

Utiliser #PauseTimeout
 Utiliser cette valeur : secondes

12. Comme vous pouvez le constater, la reconnaissance OCR n'est pas parfaite mais elle est tout à fait suffisante pour effectuer la synchronisation. Dans le champ Texte, sélectionnez un mot parfaitement reconnu, le mot **Voulez** par exemple, puis copiez/collez ce texte dans le champ en-dessous :

Synchronisation Texte OCR

Moteur OCR
Sélectionnez le moteur OCR : WinTask

Capture
Cliquez sur le bouton "Capturer" puis à l'aide de la souris, dessinez un rectangle autour du texte sur lequel se synchroniser. Capturer

Fenêtre "NOTEPAD.EXE|DirectUIHWND|Bloc-notes|1",1

Seulement dans cette zone (pas dans toute la fenêtre)

Texte analysé par le moteur OCR
Dans le champ "Texte" ci-dessous, le texte tel qu'analysé par le moteur OCR est affiché. Si le texte n'est pas parfait, sélectionnez quelques caractères correctement reconnus et collez-les dans le champ "Copiez du champ Texte".

Texte
Voulez vous en registrer


Dans le champ "Texte analysé par le moteur OCR", sélectionnez le texte que vous désirez attendre dans cette synchronisation et collez-le dans le champ ci-dessous :

Voulez Vérifier

Spécifiez le délai d'attente maximum avant que la Pause n'échoue
La variable directive #PauseTimeout donne le délai d'attente maximum avant qu'une Pause n'échoue (valeur par défaut, 120 secs) En savoir plus

Utiliser #PauseTimeout
 Utiliser cette valeur : 10 secondes

Coller dans le script Annuler Aide Options >>

13. Cliquez sur le bouton **Vérifier** pour vérifier qu'au rejoue le texte sera vu correctement. Un message vous indique que le texte a été reconnu correctement.
14. Cliquez sur le bouton **Continuer l'enregistrement** ou **Coller dans le script**. Le mode Enregistrement reprend. Cliquez sur le bouton **Ne pas enregistrer** (ou bouton **Non** sous Windows XP/2003).
15. Arrêtez le mode Enregistrement en cliquant sur la première icône  de la barre d'outils WinTask.
16. La fenêtre de l'Editeur revient alors au premier plan et vous remarquez cet ensemble d'instructions dans le script :

ret = UseOCREngine(2)

Pause 10 secs until

TextOCR("Voulez")

InWindow("NOTEPAD.EXE|DirectUIHWND|Bloc-notes|1", 1)


InArea(7,6,31,173)

PauseFalse

MsgBox("L'attente à la ligne " + #ErrorLine\$ + " a échoué !")

End

EndPause

17. Une instruction *Pause 10 secs until* a été générée ; elle signifie attendre jusqu'à ce que le texte "Voulez" apparaisse dans la fenêtre dont le nom est donné par l'instruction *InWindow*. Si ce texte n'est pas trouvé par le moteur OCR au bout d'une valeur par défaut de 10 secondes, l'instruction suivant le terme *PauseFalse* est exécutée ; cette instruction affiche un message d'erreur en précisant le numéro de ligne dans le script où l'attente a échoué.
18. Lancez l'exécution du script ainsi généré en sélectionnant dans l'éditeur le menu **Démarrer/Exécution** ou l'icône . Enregistrez le script sous le nom *script06*.

Les autres assistants de synchronisation sont similaires et leur appel se fait également en cliquant sur leur icône dans la barre d'outils WinTask en mode Enregistrement.

Le langage WinTask

Le langage de programmation de WinTask est voisin de Visual Basic™ de Microsoft. Les scripts d'automatisation doivent respecter une certaine structure et la syntaxe des instructions pour éviter des erreurs de compilation. Les scripts sont composés de trois sections distinctes. La première comporte la déclaration des tableaux de données utilisés dans le script, la deuxième liste les fonctions et sous-routines définies par l'utilisateur et la troisième est le programme principal appelant les tableaux et les fonctions déclarées dans les sections précédentes. Le langage WinTask peut manipuler des entiers, des chaînes de caractères, des tableaux d'entiers ou de chaînes à une seule dimension, et des entiers de type Unsigned (pour des appels à des fonctions internes de Windows).

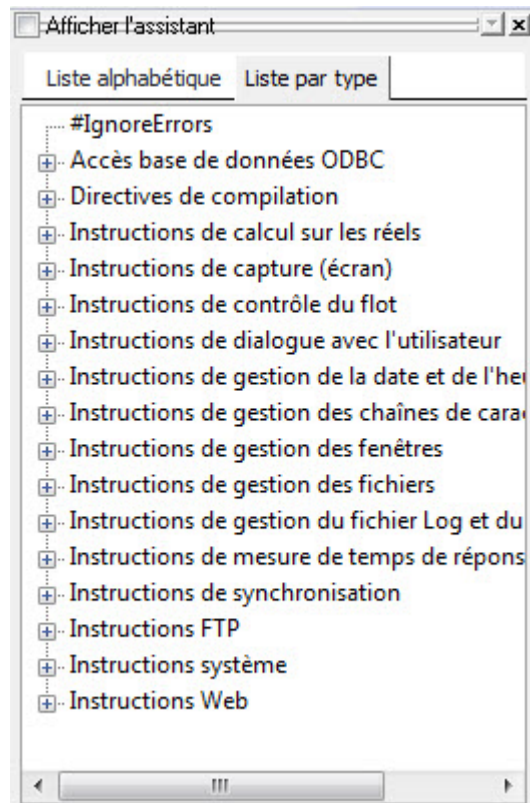
Les scripts d'automatisation créés par le mode Enregistrement génèrent la partie programme principal. Ensuite, comme dans tout langage de programmation, des fonctions ou des sous-routines peuvent être écrites pour traiter les mêmes actions par simple appel d'une fonction, si ces actions à effectuer reviennent plusieurs fois dans le programme principal.

Instructions du langage

Les instructions du langage peuvent avoir des paramètres et peuvent retourner un code retour ou une variable résultat. Si l'instruction a des paramètres, la liste de ceux-ci est mise entre parenthèses juste après le nom de l'instruction et les différents paramètres sont séparés par des virgules. Si une constante de type chaîne de caractères est passée en paramètre à une instruction, la chaîne doit être mise entre double quotes (par exemple "Bonjour"). Les instructions renvoyant une variable résultat de type chaîne de caractères ont leur nom se terminant par le caractère \$. Les instructions renvoyant un entier NE SE TERMINE PAS par le caractère \$. Voici un exemple de syntaxe :

```
Shell("explorer", 1)
```

La fenêtre listant toutes les instructions du langage est dans la partie droite de l'Editeur. Si elle n'est pas affichée, cliquez sur l'icône **Langage** de la barre d'outils de l'Editeur ou appuyez sur la touche F4. Les instructions sont classées par type en arborescence classique. En double cliquant une instruction, l'aide pour cette instruction s'affiche ou un assistant est proposé. Un clic droit sur une instruction affiche également un menu contextuel. Si vous cochez Afficher l'assistant, un double clic sur une instruction fait afficher une boîte de dialogue où les paramètres pour cette instruction sont à saisir.



N'hésitez pas à utiliser cette fenêtre Langage pour accéder aux différentes instructions de WinTask.

Variables

L'utilisation de variables dans un script WinTask permet de stocker en mémoire une donnée pour ensuite la réutiliser plus loin. Une variable n'a pas besoin d'être déclarée en début de script, sauf si c'est une variable de type tableau. Un nom de variable ne peut pas comporter de caractères accentués. Dans cette ligne de script :

```
Read("monfichier.txt", ligne$, crlf)
```

L'instruction *Read* ouvre le fichier *monfichier.txt*, lit la première ligne du fichier et stocke le contenu de cette ligne dans la variable de type chaîne de caractères *ligne\$*. Le script peut ensuite extraire l'information désirée du contenu de la variable *ligne\$*. Comme pour les instructions retournant une chaîne de caractères, les variables de type chaîne de caractères doivent avoir un nom se terminant par le caractère **\$**.

Voici un autre exemple d'utilisation de variables :

```
compta$ = Chr$(34) + "C:\Program Files\visicompt\visicompt.exe" + Chr$(34)
Shell(compta$)
```

```
Shell(Chr$(34) + "C:\Program Files\visicompt\visicompt.exe" + Chr$(34))
```

Les deux premières lignes sont équivalentes à la troisième. Si le script doit souvent faire appel au programme *visicompt.exe*, il est préférable d'utiliser une variable et ensuite d'y faire appel comme dans *Shell(compta\$)*

Variables directives

WinTask dispose d'un ensemble de variables directives afin de paramétrer la manière dont s'exécute un script. Par exemple, si une application met beaucoup de temps à se charger, le timeout par défaut peut être modifié par une variable directive afin que l'application ait le temps de se charger sans que WinTask n'indique une erreur.

Toutes les variables directives commencent par le caractère # . Dans la fenêtre Langage de l'Editeur, double-cliquez sur la tête de chapitre Toutes les instructions, vous voyez en début de liste toutes les variables directives disponibles. Voici un exemple :

```
#IgnoreErrors = 1
```

Entiers

Dans WinTask, les entiers peuvent prendre des valeurs entières de -2 147 483 648 à +2 147 483 647. Une variable de type entier NE DOIT PAS se terminer par le caractère \$.

Chaînes de caractères

La taille maximum autorisée d'une chaîne de caractères est de 32 767 caractères. Une chaîne vide est "", une chaîne contenant le caractère Espace est " ". Une variable de type chaîne de caractères doit se terminer par le caractère \$. Voici un exemple d'assignation d'une chaîne de caractères à une variable de type chaîne :

```
Formulepolitesse$ = "Nous vous prions d'agréer, Madame, Monsieur, l'expression  
de nos sentiments distingués"
```

Nombres réels

Ce type n'est pas supporté par WinTask. Les nombres réels sont stockés sous la forme d'une chaîne de caractères et les instructions **Add\$**, **Divide\$**, **Multiply\$** et **Subtract\$** permettent d'effectuer des opérations sur ces chaînes représentant des nombres réels. Voici un exemple :

```
Reel1$ = "135,46"  
Reel2$ = "-10,46"  
Resultat$=Add$(Reel1$,Reel2$)
```

Msgbox(Resultat\$)

Tableaux

Seuls les tableaux à une dimension sont acceptés dans la syntaxe WinTask. Ils doivent être déclarés en tout début de script en utilisant l'instruction Dim. La taille maximum d'un tableau est de 65535 éléments. Le premier élément d'un tableau est à l'index 0. Voici un exemple de déclaration de tableau de type entier ayant 1000 éléments :

Dim Data(1000)

Voici un exemple de déclaration de tableau de type chaîne de caractères :

Dim Data\$(1000)

Opérateurs

Les opérateurs supportés dans WinTask sont ceux classiques de tout langage, assignation, opérateurs arithmétiques, opérateurs logiques et opérateur de concaténation de chaînes de caractères. Les opérateurs logiques sont utilisés dans les instructions conditionnelles telles **If**, **While** et **Repeat**. Voici la liste avec un exemple pour chacun :

= Assignation : Assigne une valeur à une variable

compteur = 38
nom\$ = "Olivier DUPONT"
ville\$(compteur) = "Cherbourg"

+ Addition sur des entiers

colonne = ligne + 7

- Soustraction sur des entiers

Profit = Revenus - Depenses

* Multiplication

*Doigts = Mains * 5*

/ Division

Heures = Jours / 24

= Egalité logique

If (employes = 15) Then ... Endif

<> Non égal

If (employes <> 15) Then ... Endif

< Inférieur à

While (employes < 15) ... Wend

<= Inférieur ou égal

If (employes <= 15) Then ... Endif

> Plus grand que

Repeat ... Until (employes > 15)

>= Plus grand ou égal

If (employes >= 15) Then ... Endif

AND ET logique

If (employes = 15) AND (bureaux < 10) Then ... Endif

OR OU logique

*If (employes <> bureaux) OR (employes <> ordinateurs) Then ...
Endif*

+ Concaténation de chaînes

Nom\$ = Prenom\$ + " " + Nomfamille\$

Titre\$ = "Le directeur est : " + Nom\$

Instructions de gestion des fenêtres

WinTask inclut différents mécanismes de recherche d'une fenêtre lorsqu'à l'exécution l'automate doit trouver la fenêtre spécifiée dans le script. Les variables directives et instructions les plus importantes gérant l'identification de fenêtres sont décrites dans ce paragraphe.

#IgnoreErrors

La variable directive **#IgnoreErrors** contrôle le comportement de l'automate quand une erreur d'exécution est détectée. Par défaut, cette variable directive est à 0 : en cas d'erreur, un message d'erreur est affiché et l'exécution du script est arrêtée. En mettant cette variable directive à 1, le script continue son exécution en cas d'erreur. L'instruction ayant provoqué l'erreur renvoie un code retour qu'il est vivement recommandé de tester afin de gérer le cas d'erreur.

Voici un exemple où sur une portion du code, la valeur par défaut de **#IgnoreErrors** est modifiée :

```
#IgnoreErrors = 1  
ret = Shell("notepad.exe")  
If (ret = 0) Then  
    MsgBox("Notepad a été lancé avec succès")  
Else  
    MsgBox("Impossible de lancer Notepad !")  
Endif
```

```
#IgnoreErrors = 0
```

Un message est affiché après avoir essayé de lancer **Notepad**. Le message dépend du code retour renvoyé par l'instruction **Shell**.

#ActionTimeout

La variable directive **#ActionTimeout** contrôle le temps d'attente (timeout) maximum avant que WinTask ne retourne une erreur lors de l'exécution d'une ligne dans le script, telle l'identification d'une fenêtre, d'un contrôle, d'une option de menu. La valeur par défaut est de 30 secondes.

Voici un exemple où sur une portion du code, la valeur par défaut de **#ActionTimeout** est modifiée :

```
Shell("notepad.exe")  
#ActionTimeout = 5
```

```
#IgnoreErrors = 1
```

```
ret=UseWindow("NOTEPAD.EXE/Edit/Sans titre - Bloc-notes/1",1)  
If (ret = 0) Then  
MsgBox("Notepad a été lancé en moins de 5 secondes")  
Else  
MsgBox("Achetez un ordinateur plus rapide !")  
Endif
```

```
#ActionTimeout = 30
```

L'instruction UseWindow renvoie une erreur si à l'exécution, la fenêtre spécifiée n'est pas trouvée au bout de 5 secondes. Pour que l'erreur n'arrête pas l'exécution du script, notez #IgnoreErrors=1. Le code retour de UseWindow est ensuite testé et le message affiché dépend de la valeur du code retour.

#UseExact

La variable directive **#UseExact** contrôle l'algorithme d'identification d'une fenêtre lors du rejoue du script. Considérez ces deux lignes :

```
UseWindow("NOTEPAD.EXE/Edit/Sans titre - Bloc-notes/1")  
UseWindow("NOTEPAD.EXE/Edit/test.txt - Bloc-notes/1")
```

Au rejoue, les deux recherchent une fenêtre de **Notepad**. La première ligne recherche une fenêtre notepad lorsqu'un document nouveau est chargé. La deuxième recherche une fenêtre notepad lorsque le document de nom **test.txt** a été ouvert.

Le comportement par défaut de WinTask lors du rejoue est de rechercher d'abord une fenêtre dont le nom est exactement celui spécifié dans l'instruction *UseWindow*. Si au bout de quelques secondes, la fenêtre exacte n'est pas trouvée, une recherche approchée est lancée : les caractères de droite du nom de fenêtre sont tronqués pour essayer de trouver une fenêtre dont le début du nom est le même que celui spécifié dans UseWindow. Le mécanisme s'arrête à *NOTEPAD.EXE/E*. Si malgré les caractères tronqués, aucune fenêtre de même nom n'est trouvée, alors une erreur d'exécution se produit.

Cette méthode d'identification des fenêtres au rejoue peut être modifiée en mettant la variable directive **#UseExact** à la valeur 1 : dans ce cas, seule une fenêtre de nom exactement le même que celui spécifié dans le script sera trouvée. La valeur par défaut de **#UseExact** est 0 et donc dans ce cas, la recherche approchée se fait.

ASTUCE : WinTask essaie de trouver la fenêtre avec exactement le même nom que celui spécifié dans le script pendant environ 3 secondes (1/9 de la valeur de **#ActionTimeout** qui vaut par défaut 30 secondes). Si le titre des fenêtres change au rejoue, cette recherche approchée ralentit l'exécution du script (attente de 3 secondes à chaque fois). Pour éviter ces délais, vous pouvez tronquer manuellement dans le script le nom de fenêtre. Dans l'exemple avec notepad, cette ligne : *UseWindow("NOTEPAD.EXE/Edit/"),* au rejoue, trouvera la fenêtre que ce soit un document nouveau ou le document test.txt chargé.

ASTUCE: Si plusieurs fenêtres d'une même application mais avec différents documents sont présentes sur le bureau, la recherche approchée peut

entraîner une erreur d'identification lors du rejeu. Dans ce cas, forcez **#UseExact** à 1.

ExistW()

L'instruction **ExistW** permet de détecter l'existence d'une fenêtre au rejeu. L'instruction renvoie un code retour qui peut être utilisé dans des instructions conditionnelles. Utilisez cette instruction pour gérer les cas où la fenêtre spécifiée ne s'affiche pas.

Voici un exemple :

```
ret = ExistW("NOTEPAD.EXE|#32770|Enregistrer sous", 1)  
If (ret = 1) Then  
    UseWindow("NOTEPAD.EXE|#32770|Enregistrer sous", 1)  
    Click(Button, "&Enregistrer")  
Else  
    MsgBox("Boîte de dialogue Enregistrer sous inexistante !")  
Endif
```

Focus\$()

L'instruction **Focus\$** permet de connaître dynamiquement le nom de la fenêtre qui est au premier plan lors du rejeu. Pour une application comportant des fenêtres filles, **Focus\$** renvoie le nom de la fenêtre fille qui a le focus.

Voici un exemple :

```
Titre_fenetre$ = Focus$()  
If (Titre_fenetre$ = "NOTEPAD.EXE|Edit|Sans titre - Bloc-notes |1") Then  
    UseWindow("NOTEPAD.EXE|#32770|Enregistrer sous", 1)  
    Click(Button, "&Enregistrer")  
Else  
    MsgBox("La boîte de dialogue Enregistrer sous n'a pas le focus !")  
Endif
```

Top\$()

L'instruction **Top\$** permet de connaître dynamiquement le nom de la fenêtre active lors du rejeu. L'instruction est similaire à **Focus\$** sauf qu'elle renvoie le nom de la fenêtre mère si l'application a des fenêtres filles.

Voici un exemple :

```
Titre_fenetre$ = Top$()
```

```
If (Titre_fenetre$ = "NOTEPAD.EXE/Notepad/ Sans titre - Bloc-notes") Then  
    UseWindow("NOTEPAD.EXE/#32770/Enregistrer sous", 1)  
    Click(Button, "&Enregistrer")  
Else  
    MsgBox("Notepad n'est pas l'application active !")  
Endif
```

Un autre exemple illustrant la gestion de deux documents chargés dans notepad.


```
Shell("notepad.exe c:\file01.txt")  
Shell("notepad.exe c:\file02.txt")
```

```
Titre_fenetre$ = Top$()
```

```
If (Window_title$ = "NOTEPAD.EXE/Notepad/file02.txt – Bloc-notes") Then  
    UseWindow("NOTEPAD.EXE/Edit/file02.txt|1", 1)  
    SendKeys("texte envoyé dans file02")  
Else  
    UseWindow("NOTEPAD.EXE/Edit/file01.txt|1")  
    SendKeys("texte envoyé dans file01")  
Endif
```

Exercice 7

Cet exercice est un exercice de programmation où vous allez mettre en oeuvre les instructions du langage WinTask gérant les fenêtres. La solution de l'exercice est donnée en appendice C. Les fenêtres et boutons correspondent à un environnement Windows 7.

1. Démarrez l'Editeur WinTask. Si l'Assistant de Création de Script s'affiche, cliquez sur le bouton **Annuler**. Le titre de la fenêtre de l'Editeur doit être **WinTask – [SansNom1]**. Si ce n'est pas le cas, cliquez sur l'icône **Nouveau** dans la barre d'outils.
2. Cliquez sur l'icône **Enreg** de la barre d'outils pour démarrer le mode Enregistrement. La boîte de dialogue **Démarrer l'enregistrement** s'affiche, cliquez sur le bouton **OK**.
3. La boîte de dialogue **Lancement d'une application** s'affiche. Tapez **Notepad** dans le champ **Programme** et cliquez sur le bouton **OK**.
4. Tapez **Bonjour** et appuyez sur la touche **Entrée**.
5. Puis dans notepad, sélectionnez le menu **Edition/Heure/Date**. Cliquez après la date insérée et appuyez sur la touche **Entrée**.
6. Dans notepad, cliquez sur le bouton Fermeture en haut à droite .
7. Cliquez sur le bouton **Enregistrer** (ou le bouton **Oui** sous XP/2003) quand Notepad affiche la boîte de dialogue **Voulez-vous enregistrer les modifications**.

8. Enregistrez le fichier sous le nom **test07** dans le répertoire courant. Cliquez sur le bouton **Enregistrer** une fois saisi le nom test07.
9. Arrêtez le **Mode Enregistrement** en cliquant sur la première icône de la barre d'outils WinTask flottante.
10. Dans la fenêtre de l'Editeur WinTask, modifiez le script pour remplacer le nom fichier ("*test07.txt*") par une variable de nom *nomfichier\$*. En début de script, assignez à cette variable la valeur "*test07.txt*".
11. Cliquez sur le bouton **Exéc** de la barre d'outils pour rejouer les actions décrites dans le script. Donnez comme nom au script **script07a** quand la boîte de dialogue Enregistrer sous s'affiche. Corrigez des erreurs de compilation éventuelles si la fenêtre de Résultats de l'Editeur en signale, puis exécutez à nouveau.
12. L'exécution du script ne va pas à son terme puisque le fichier cible existe déjà.
13. Modifiez le script pour automatiser le clic sur le bouton Oui de la boîte de dialogue **Confirmer l'enregistrement**. Pour cela, dans la fenêtre de l'Editeur, mettez le curseur à la dernière ligne du script généré précédemment et cliquez sur l'icône **Enreg**. Cochez **Rien** et cliquez sur le bouton **OK**. En mode Enregistrement, cliquez sur le bouton **Oui** puis arrêtez le mode Enregistrement.
14. Ajoutez le bloc *If...EndIf* pour ne cliquer sur **Oui** que si la fenêtre **Confirmer l'enregistrement** est affichée. Utilisez pour cela l'instruction *ExistW* comme vue plus haut.
15. Enregistrez le script modifié sous le nom **script07b**, et lancez l'exécution en cliquant sur l'icône Exéc de la barre d'outils..
16. Maintenant supprimez la ligne *ExistW*, mettez la variable directive *#IgnoreErrors* à 1, et testez le code retour de l'instruction *UseWindow* qui cherche la fenêtre de nom **Confirmer l'enregistrement**. Modifiez le bloc *If* pour maintenant tester le code retour dans ce *If* et ne cliquer sur **Oui** que si le *UseWindow* a trouvé la fenêtre spécifiée.
17. Enregistrez le script ainsi modifié sous le nom **script07c**. Lancez l'exécution.
18. Dans la première ligne du script où la variable *nomfichier\$* était assigné à "*test07*", modifiez pour que la variable prenne désormais la valeur "*test07d*".
19. Enregistrez le script ainsi modifié sous le nom **script07d**. Lancez l'exécution.
20. Questions ouvertes : la solution mise en oeuvre à l'étape 16 s'exécute en beaucoup plus de temps à l'étape 18, ce qui porterait à croire que la méthode d'avant est mieux. Toutefois, quelle variable directive pourrait être modifiée pour que le *UseWindow* en erreur soit vu plus rapidement ?

Cet exercice illustre qu'il n'y a pas une solution unique à un problème d'automatisation. La puissance du langage WinTask permet différentes approches, au

programmeur de voir si un temps d'exécution le plus rapide possible est très important, ou si au contraire, une programmation plus simple est préférée.

Itération

Le langage WinTask inclut deux instructions pour exécuter répétitivement un bloc de lignes du script. La construction While...Wend exécute les lignes entre ces deux instructions zéro ou n fois tant que la condition du While est vraie. La construction Repeat...Until exécute les lignes entre ces deux instructions une ou n fois tant que la condition du Until est vraie.

Itération

Voici un exemple des deux itérations avec respectivement **While/Wend** et **Repeat/Until**.

```
Debut = 0
Fin = 10
Index = Debut
While (Index < Fin)
    Lignes à exécuter index fois
    Index = Index + 1
Wend
```

```
Start = 0
Fin = 10
Index = Debut
Repeat
    Lignes à exécuter index fois
    Index = Index + 1
Until (Index = Fin)
```

Exercice 8

Cet exercice est un exercice de programmation où vous allez mettre en œuvre une itération. La première étape consiste à utiliser le mode Enregistrement pour générer les lignes de code à itérer, puis vous ajouterez des instructions du langage WinTask gérant les itérations. La solution de l'exercice est donnée en appendice C.

1. Démarrez l'Editeur WinTask. Si l'Assistant de Création de Script s'affiche, cliquez sur le bouton **Annuler**. Le titre de la fenêtre de l'Editeur doit être **WinTask – [SansNom1]**. Si ce n'est pas le cas, cliquez sur l'icône **Nouveau** dans la barre d'outils.
2. Cliquez sur l'icône **Enreg** de la barre d'outils pour démarrer le mode Enregistrement. La boîte de dialogue **Démarrer l'enregistrement** s'affiche, cliquez sur le bouton **OK**.

3. La boîte de dialogue **Lancement d'une application** s'affiche. Tapez **Notepad** dans le champ **Programme** et cliquez sur le bouton **OK**.
4. Tapez **Bonjour 1** et appuyez sur la touche **Entrée**.
5. Tapez **Bonjour 2** sur la ligne suivante et appuyez sur la touche **Entrée**.
6. Arrêtez le **Mode Enregistrement** en cliquant sur la première icône de la barre d'outils WinTask flottante.
7. Fermez Notepad sans enregistrer.
8. Dans la fenêtre de l'Editeur, modifiez le script pour utiliser la construction *While/Wend* afin de saisir 10 lignes de texte sous cette forme :

```
Bonjour 1  
Bonjour 2  
...  
Bonjour 10
```

ASTUCE: Utilisez l'opérateur de concaténation de chaînes (+) pour concaténer le mot "Bonjour" à l'index 1, 2, ... 10. Comme l'index est un entier, utilisez l'instruction Str\$ pour convertir l'entier index en chaîne.

9. Modifiez le script pour que le mot "*Bonjour*" ne soit plus une constante mais une variable dont le contenu est "*Bonjour*".
10. Modifiez le contenu de la variable pour qu'elle contienne maintenant "*Ligne de texte dans Notepad*" au lieu de "*Bonjour*".
11. Enregistrez le script sous le nom **script08a** et cliquez sur l'icône **Exéc** de la barre d'outils pour lancer l'exécution. Si des erreurs de compilation sont affichées, corrigez-les et relancez l'exécution. Vous devez voir notepad s'ouvrir et la première ligne dans notepad est saisie : **Ligne de texte dans notepad 1**, la deuxième ligne, **Ligne de texte dans notepad 2**, etc.....
12. Modifiez le script en remplaçant le bloc *Repeat/Until* par un bloc *While/Wend*.
13. Enregistrez le script modifié sous le nom **script08b** et lancez l'exécution.

Vous savez désormais répéter les mêmes actions dans un script, sans avoir à dupliquer les lignes de code.

Instructions de gestion des fichiers

WinTask inclut des instructions permettant de lire ou d'écrire dans des fichiers texte, des fichiers Excel, des fichiers INI ou des fichiers XML. Ce paragraphe décrit les instructions de gestion des fichiers les plus couramment utilisées. Pour une explication exhaustive, allez dans la fenêtre Langage de l'Editeur et double-cliquez sur la tête de chapitre Instructions de gestion des fichiers.

Exist()

L'instruction **Exist** permet de déterminer si un fichier existe ou pas lors du rejoue du script. Le nom de fichier peut être un nom complet incluant le chemin, un nom UNC, ou un nom de fichier à rechercher dans le répertoire courant. L'instruction renvoie un code retour qui peut ensuite être utilisé dans une instruction conditionnelle. Utilisez cette instruction pour éviter des erreurs d'exécution lors d'un appel à un fichier qui n'existe pas.

Voici un exemple :

```
NomFichier$ = "C:\Program Files\WinTask\Help\WinTask.chm"  
ret = Exist(NomFichier$)  
If ret = 1 Then  
    Shell("hh.exe " + Chr$(34) + NomFichier$ + Chr$(34))  
Else  
    MsgBox("Fichier non trouvé : " + NomFichier$)  
Endif
```

ASTUCE : Le nom de fichier passé à l'instruction **Exist** ne comporte pas de *Chr\$(34)* même si le chemin a des blancs, alors que les *CHR\$(34)* sont obligatoires dans l'instruction Shell car le blanc est un séparateur entre le nom de l'exé et le paramètre pour cet exe. *CHR\$(34)* est le symbole ASCII pour double-quote.

Kill()

L'instruction **Kill** supprime le fichier spécifié. Le nom de fichier peut être un nom complet incluant le chemin, un nom UNC, ou un nom de fichier à rechercher dans le répertoire courant. Si le fichier n'existe pas, l'exécution retourne une erreur, voici un exemple pour supprimer un fichier qu'il existe ou pas :

```

NomFichier$ = "C:\Temp\UnFichierInutile.txt "
#IgnoreErrors=1
ret = Kill(NomFichier$)
If ret = 0 Then
    MsgBox("Fichier : " + NomFichier$ + " a été supprimé. ")
Else
    MsgBox("Le fichier n'existe pas ou il est impossible à supprimer : " +
    NomFichier$)
Endif

#IgnoreErrors=0

```

Read()

L'instruction **Read** permet de lire le contenu d'un fichier de données. Le nom de fichier peut être un nom complet incluant le chemin, un nom UNC, ou un nom de fichier à rechercher dans le répertoire courant. L'instruction renvoie un code retour optionnel qui peut être testé ensuite par un If.

L'instruction **Read** est le plus souvent appelée n fois dans une itération afin de lire le premier enregistrement du fichier, puis le deuxième, le troisième, ... jusqu'à ce que la fin de fichier soit atteinte. A chaque appel de Read, le pointeur de lecture est positionné à l'enregistrement suivant. L'instruction **Eof** (décrite plus loin) permet de vérifier si cette fin de fichier est atteinte.

Voici les deux syntaxes les plus utilisées de l'instruction **Read** :

```
ret = Read(NomFichier$, Buffer$, CRLF)
```

L'instruction lit une ligne de données se trouvant dans le fichier **NomFichier\$** et stocke ces données dans la variable **Buffer\$**. Chaque ligne dans le fichier est séparée par CR-LF (Carriage Return-Line Feed), ce qui correspond en informatique au passage à la ligne suivante.

```
ret = Read(NomFichier$, Buffer$, Separateur$)
```

L'instruction lit le contenu de **NomFichier\$** jusqu'à ce que la chaîne de caractères spécifiée dans **Separateur\$** soit atteinte. Ce contenu est écrite dans **Buffer\$**.

ASTUCE : les utilisateurs avancés peuvent utiliser l'instruction **SetReadPos** pour positionner le pointeur de lecture à un numéro d'enregistrement précis, si la lecture séquentielle des enregistrements ne convient pas.

Eof()

L'instruction **Eof** est utilisée avec l'instruction **Read** pour déterminer si tous les enregistrements du fichier ont été lus. La lecture complète d'un fichier se fait dans une boucle While/Wend ou Repeat/Until jusqu'à ce que la fin de fichier Eof soit atteinte.

Voici un exemple :

```
NomFichier$ = "C:\Temp\MonFichier.txt"  
Read(NomFichier$, Buffer$,CRLF)  
ret = Eof(NomFichier$)  
If ret = 1 Then  
    MsgBox("La fin de fichier est atteinte")  
Else  
    MsgBox("Exécutez une nouvelle fois Read pour lire l'enregistrement suivant !")  
Endif
```

Write()

L'instruction **Write** permet d'écrire des données dans un fichier. Le nom de fichier peut être un nom complet incluant le chemin, un nom UNC, ou un nom de fichier à rechercher dans le répertoire courant. L'instruction renvoie un code retour optionnel qui peut être testé ensuite par un If..

Si le fichier dans lequel écrire n'existe pas, l'instruction **Write** le crée. Si le fichier existe, l'instruction Write écrit les données à la fin du fichier.

Voici les deux syntaxes les plus utilisées de l'instruction **Write** :

```
ret = Write(NomFichier$, Buffer$, CRLF)
```

Une ligne de données (un enregistrement) est écrite dans le fichier **NomFichier\$**. Le contenu de la variable **Buffer\$** est écrit à la fin du fichier et un CR-LF (Carriage Return-Line Feed) est ajouté pour le passage à la ligne suivante.

```
ret = Write(NomFichier$, Buffer$, Separateur$)
```

Le contenu de la variable **Buffer\$** est écrit à la fin du fichier spécifié par **NomFichier\$**. Puis le caractère contenu dans **Separateur\$** est ajouté.

Exercice 9

Cet exercice illustre comment un script peut lire des données dans un fichier externe et les utiliser pour effectuer des traitements dessus. Cet exercice utilise le script réalisé au chapitre intitulé Itération, script de nom **script08a**. La solution de l'exercice 9 est donnée en appendice C.

1. Démarrez l'Editeur WinTask. Si l'Assistant de Création de Script s'affiche, cliquez sur le bouton **Annuler**. Cliquez sur l'icône **Ouvrir** de la barre d'outils pour ouvrir le script **script08a.src**.
2. Cliquez sur l'icône **Exéc** dans la barre d'outils de l'Editeur pour lancer l'exécution du script.

3. Une fois l'exécution terminée, enregistrez le fichier txt qui a été créé par le script sous le nom **test09** et fermez notepad.
4. Cliquez sur l'icône **Nouveau** dans la barre d'outils de l'Editeur pour créer un nouveau script.
5. Cliquez sur l'icône **Enreg** de la barre d'outils pour démarrer le mode Enregistrement La boîte de dialogue **Démarrer l'enregistrement** s'affiche, cliquez sur le bouton **OK**.
6. La boîte de dialogue **Lancement d'une application** s'affiche. Si vous utilisez une version 32 bits de Windows, tapez **wordpad** dans le champ **Programme** et cliquez sur le bouton **OK**. Si vous utilisez une version 64 bits de Windows, cliquez sur le bouton **Parcourir** sous le champ **Programme** et sélectionnez wordpad.exe qui se trouve dans program files (x86)\Windows NT\Accessories.
7. Tapez **Bonjour WordPad** dans la fenêtre WordPad et appuyez sur la touche **Entrée**.
8. Arrêtez le **Mode Enregistrement** en cliquant sur la première icône de la barre d'outils WinTask flottante.
9. Quittez WordPad sans enregistrer.
10. Modifiez le script qui vient d'être généré pour insérer une boucle lisant ligne à ligne le contenu du fichier **test09.txt**. Insérez cette boucle après le lancement de WordPad et avant la saisie du texte **Bonjour WordPad**.

ASTUCE : Utilisez l'instruction *Eof* pour détecter que le fichier **test09.txt** a été entièrement lu.

11. Copiez la ligne qui saisit du texte dans WordPad et collez-la à l'intérieur de la boucle. Remplacez le texte **Bonjour WordPad** par la variable chaîne de caractères contenant la ligne lue à partir du fichier **test09.txt**.
12. Lancez l'exécution en cliquant sur l'icône **Exéc** dans la barre d'outils WinTask. Enregistrez le script sous le nom **script09**. Si la fenêtre de Résultats de la compilation montre des erreurs, corrigez-les puis lancez l'exécution à nouveau.

Vous savez désormais utiliser dans un script des données externes. Une des utilisations les plus courantes de WinTask est la saisie automatique de données dans une application Windows ; ces données sont souvent stockées dans un fichier Excel plutôt qu'un fichier texte, c'est l'objet du paragraphe suivant.

ReadExcel et WriteExcel

Le langage WinTask inclut deux instructions pour lire ou écrire dans des fichiers Excel (fichiers d'extension .XLS ou .XLSX pour Excel 2007/2010). Microsoft Excel doit être installé sur le PC.

L'instruction **ReadExcel** permet de lire une rangée de cellules se trouvant dans une même colonne ou même ligne. Le fichier Excel peut rester ouvert quand cette instruction est utilisée.

L'instruction **WriteExcel** permet d'écrire des données dans une rangée de cellules se trouvant dans une même colonne ou une même ligne. Comme WriteExcel met à jour le fichier Excel directement, il ne faut pas que le fichier Excel soit ouvert quand WriteExcel est utilisé. Le fichier Excel doit avoir été créé auparavant (WinTask inclut une instruction de création de fichier Excel, **CreateExcelFile**).

Les deux instructions ont la syntaxe suivante :

ret = ReadExcel(FichierExcel\$, Rangee_cellules\$, Tableau_donnees\$())

ret = WriteExcel(FichierExcel\$, Rangee_cellules\$, Tableau_donnees\$())

Le premier paramètre ***FichierExcel\$*** contient le nom du fichier Excel à lire ou dans lequel écrire. Si le chemin n'est pas spécifié, le fichier Excel est recherché dans le répertoire par défaut des scripts WinTask.

Le deuxième paramètre ***Rangee_cellules\$*** spécifie dans quelles cellules, les données sont écrites ou lues. La chaîne de caractères est composée de deux parties : le nom de la feuille et la rangée de cellules séparés par un point d'exclamation (par exemple, "*Feuil1!A9:F9*"). Si le nom de la feuille est omis, la rangée de cellules est celle de la première feuille. Par exemple "*Dépenses!B3:B5*" spécifie les cellules B3, B4 et B5 de la feuille Dépenses, "*D8:H8*" spécifie les cellules D8, E8, F8, G8, H8 de la première feuille.

Le paramètre ***Tableau_donnees\$()*** spécifie le tableau contenant les données à écrire, ou le tableau qui va recevoir les données lues. La première donnée lue est stockée dans l'élément 0 de ce tableau. Le tableau doit être défini en début de script en utilisant l'instruction *DIM*.

Exercice 10

Cet exercice de programmation montre comment un script peut mettre à jour un fichier Excel. Cet exercice utilise le fichier texte enregistré à l'exercice 9, fichier de nom **test09.txt**. La solution de l'exercice 10 est donnée en appendice C.

1. Démarrez Excel. Dans Excel, sélectionnez le menu **Fichier/Enregistrer sous**.
2. Donnez ce nom sous un Windows 32 bits : **c:\program files\wintask\scripts\exercice10.xls** (ou **exercice10.xlsx** si vous utilisez Excel 2007/2010) - ou sous un Windows 64 bits : **c:\program files (x86)\wintask\scripts\exercice10.xls** (ou **exercice10.xlsx** si vous utilisez Excel 2007/2010). Quittez Excel.
3. Démarrez l'Editeur WinTask. Si l'Assistant de Création de Script s'affiche, cliquez sur le bouton **Annuler**. Cliquez sur l'icône **Nouveau** de la barre d'outils pour créer un nouveau script.
4. En première ligne, écrivez l'instruction DIM déclarant un tableau de 25 chaînes de caractères.

5. Ajoutez une boucle pour lire le contenu du fichier **test09.txt** enregistrement par enregistrement. Utilisez le tableau déclaré à l'étape 4 pour stocker chacun des enregistrements. Utilisez l'instruction *Eof* pour déterminer la fin de la boucle.
6. Ajoutez une variable de type entier pour compter le nombre d'enregistrements lus. Cette variable va servir pour indexer le tableau, sachant que le premier élément d'un tableau est l'élément 0.
7. Utilisez l'instruction *MsgBox* pour faire afficher le nombre d'éléments dans le tableau. Rappelez-vous que l'instruction *Str\$* permet de convertir un entier en chaîne et que l'opérateur *+* est l'opérateur de concaténation de deux chaînes de caractères.
8. Utilisez l'instruction *WriteExcel* pour écrire le contenu du tableau dans le fichier Excel créé à l'étape 2, de nom **exercice10.xls**. Utilisez une rangée de cellules qui va de A1 à J1 (**test09.txt** contient 10 lignes).
9. Utilisez à nouveau l'instruction *WriteExcel* pour écrire une seconde fois le contenu du tableau dans le même fichier Excel, mais cette fois en colonne C, de C3 à C12.
10. Lancez l'exécution en cliquant sur l'icône **Exéc** dans la barre d'outils WinTask. Enregistrez le script sous le nom **script10**. Si la fenêtre de Résultats de la compilation montre des erreurs, corrigez-les puis lancez l'exécution à nouveau.
11. Ouvrez le fichier **exercice10.xls** dans Excel (ou **exercice10.xlsx** avec Excel 2007/2010). Vérifiez les données écrites automatiquement dans Excel.

Fonctions et Sous-Programmes (Sub)

Le langage WinTask inclut de nombreuses instructions qui rendent l'automatisation de tâches aisée. Un même traitement doit parfois s'effectuer à différents endroits du script, ou un même processus d'automatisation est à mettre en œuvre dans différents scripts. Avec la notion de Sous-Programme incluse dans WinTask, il est possible d'écrire des Fonctions ou des Sub qui peuvent être appelées n'importe où dans un script, permettant ainsi d'écrire des scripts structurés et facilement maintenables.

Sub...EndSub

La syntaxe pour écrire un sous-programme est **Sub... EndSub** :

```
Sub <nom de la sub> ([<paramètre1>[, <paramètre2>] ...])  
    <instructions>  
EndSub
```

Le nom de la Sub est spécifié dans <nom de la sub> et doit être unique dans l'ensemble du script. Une Sub peut avoir des paramètres, la liste est mise entre parenthèses après le nom de la Sub et les paramètres sont séparés par des virgules. Si un paramètre est de type chaîne de caractères, son nom doit se terminer par le caractère \$. Pour sortir abruptement d'une Sub par exemple dans une condition, l'instruction à utiliser est **ExitSub**.

ASTUCE : Comme les variables sont valides pour l'ensemble du script (sauf à utiliser le mot-clé **LOCAL** non traité dans ce manuel), utilisez des noms uniques pour l'ensemble du script.

ASTUCE : Veillez à ce que la Sub ait été déclarée avant de l'utiliser (le bloc Sub...EndSub doit être écrit en début de script, juste après les lignes DIM, et l'appel à la Sub ne peut avoir lieu qu'après cette déclaration).

Convertir un script en sous-programme (Sub)

N'importe quel groupe de lignes d'un script peut constituer un sous-programme, ce qui permet de l'appeler à différents endroits du script. Il faut décider des paramètres à passer.

Voici le script créé à l'exercice 2 (les lignes après la ligne CloseWindow sont différentes si vous utilisez Vista ou Windows 7, mais la transformation en Sub est identique) :

```
Shell("notepad", 1)  
UseWindow("NOTEPAD.EXE/Edit|Sans titre - Bloc-notes|1", 1)  
    SendKeys("Bonjour<Enter>")  
UseWindow("NOTEPAD.EXE/Notepad|Sans titre - Bloc-notes", 1)
```

```

ChooseMenu(Normal ", &Edition/He&ure/Date F5")
CloseWindow("NOTEPAD.EXE/Notepad/Sans titre - Bloc-notes", 1)
UseWindow("NOTEPAD.EXE/#32770/Bloc-notes", 1)
    Click(Button, "&Oui")
UseWindow("NOTEPAD.EXE/Edit/Enregistrer sous/1", 1)
    SendKeys("test02.txt")
UseWindow("NOTEPAD.EXE/#32770/Enregistrer sous", 1)
    Click(Button, "&Enregistrer")

```

Ce script peut être transformé en Sub avec deux paramètres correspondant aux valeurs qui peuvent varier d'une exécution à une autre, le texte tapé "Bonjour<Enter>", et le nom du fichier texte de sauvegarde "test02.txt". Voici le même script mais cette fois en Sub pouvant être appelée n'importe où :

Sub EcritTexte(Texte\$, NomFichier\$)

```

Shell("notepad", 1)
UseWindow("NOTEPAD.EXE/Edit/Sans titre - Bloc-notes/1", 1)
    SendKeys(Texte$ + "<Enter>")
UseWindow("NOTEPAD.EXE/Notepad/Sans titre - Bloc-notes", 1)
ChooseMenu(Normal ", &Edition/He&ure/Date F5")
CloseWindow("NOTEPAD.EXE/Notepad/Sans titre - Bloc-notes", 1)
UseWindow("NOTEPAD.EXE/#32770/Bloc-notes", 1)
    Click(Button, "&Oui")
UseWindow("NOTEPAD.EXE/Edit/Enregistrer sous/1", 1)
    SendKeys(NomFichier$)
UseWindow("NOTEPAD.EXE/#32770/Enregistrer sous", 1)
    Click(Button, "&Enregistrer")

```

EndSub

' Programme principal

Texte1\$ = "Le ciel es bleu sauf quand il pleut !"

NomFichier1\$ = "ciel_bleu.txt"

EcritTexte(Texte1\$, NomFichier1\$)

Texte2\$ = "Des ballons de toutes les formes."

NomFichier2\$ = "ballons.txt"

EcritTexte(Texte2\$, NomFichier2\$)

Le programme principal donne deux appels de la même Sub mais avec des paramètres différents.

Exercice 11

Cet exercice de programmation illustre la transformation d'un script existant en une Sub qui peut être réutilisée. Cet exercice utilise les scripts réalisés lors des exercices 7 et 8. La solution de l'exercice 11 est donnée en appendice C.

1. Démarrez l'Editeur WinTask. Si l'Assistant de Création de Script s'affiche, cliquez sur le bouton **Annuler**. Cliquez sur l'icône **Nouveau** de la barre d'outils pour créer un nouveau script. Cette fenêtre va être utilisée pour créer le nouveau script.

2. Cliquez sur l'icône **Ouvrir** de la barre d'outils WinTask et ouvrez le script **script08a.src** réalisé à l'exercice 8.
3. Sélectionnez toutes les lignes de **script08a.src** et copiez-les dans le Presse-papiers (utilisez **CTRL+C** ou le menu **Edition/Copier**). Fermez la fenêtre où est affichée **script08a.src**. N'enregistrez pas le script.
4. Revenez à la fenêtre de l'Editeur WinTask de nom **SansNom1** et collez le contenu du Presse-papiers dans cette fenêtre (utilisez **CTRL+V** ou le menu **Edition/Coller**).
5. Transformez les actions effectuées dans **script08a.src** en une Sub de nom **Saisie_Notepad** et utilisant trois paramètres, **NomFichier\$**, **Texte\$** et **Compteur**. Le paramètre **NomFichier\$** spécifie le nom du fichier notepad sous lequel enregistrer. Le paramètre **Texte\$** spécifie la ligne de texte à saisir dans notepad. Le paramètre **Compteur** spécifie combien de fois **Texte\$** doit être saisi dans notepad.

ASTUCE : Remplacez les variables dans **script08a.src** par l'appel aux paramètres de la Sub.

6. Cliquez sur l'icône **Ouvrir** de la barre d'outils WinTask et ouvrez le script **script07b.src**.
7. Recherchez l'instruction **CloseWindow**. Sélectionnez toutes les lignes de cette ligne **CloseWindow** à la fin du script, et copiez-les dans le Presse-papiers. Fermez la fenêtre **script07b.src** sans enregistrer.
8. Dans l'Editeur, revenez à la fenêtre **SansNom1** et collez les lignes précédentes à la fin de la Sub (la dernière ligne doit rester **EndSub**).
9. Vérifiez que le nom du fichier sous lequel est enregistré le fichier notepad est bien le paramètre **NomFichier\$**.
10. Ajoutez une partie Programme principal après l'instruction **EndSub**. Appelez la Sub **Saisie_Notepad** avec ces paramètres : "**text11.txt**" pour le nom de fichier, "**Exercice Sub**" pour le texte et **12** lignes de texte.
11. Lancez l'exécution en cliquant sur l'icône **Exéc** dans la barre d'outils WinTask. Enregistrez le script sous le nom **script11**. Si la fenêtre de Résultats de la compilation montre des erreurs, corrigez-les puis lancez l'exécution à nouveau.

L'autre type de sous-programme disponible dans WinTask est Fonction.

Function...EndFunction

La syntaxe pour écrire une fonction est **Function...EndFunction** :

Function <nom de fonction> ([<paramètre1>[, <paramètre2>] ...])

<instructions>

<nom de fonction> = <valeur>

EndFunction

Une fonction est identique à une Sub, sauf que la fonction retourne un résultat. Le nom de la fonction est spécifié dans <nom de fonction> et doit être unique dans l'ensemble du script. Une fonction peut avoir des paramètres, la liste est mise entre parenthèses après le nom de la fonction et les paramètres sont séparés par des virgules. Si un paramètre est de type chaîne de caractères, son nom doit se terminer par le caractère \$. Pour sortir abruptement d'une fonction par exemple dans une condition, l'instruction à utiliser est **ExitFunction**. L'assignation du résultat à la fonction doit se faire avant de sortir de la fonction. Si la valeur retournée par la fonction est une chaîne de caractères, le caractère \$ doit être le dernier caractère du nom de la fonction.

ASTUCE : Comme les variables sont valides pour l'ensemble du script (sauf à utiliser le mot-clé **LOCAL** non traité dans ce manuel), utilisez des noms uniques pour l'ensemble du script.

ASTUCE : Veillez à ce que la fonction ait été déclarée avant de l'utiliser (le bloc Function...EndFunction doit être écrit en début de script, juste après les lignes DIM, et l'appel à la fonction ne peut avoir lieu qu'après cette déclaration).

ASTUCE : Si la fonction n'a pas assigné une valeur à la variable résultat qu'elle doit retourner, le résultat est 0 ou une chaîne vide si le nom de la fonction se termine par le caractère \$.

Voici un exemple de Function et de son appel :

```
Function Absolute_diff(entier1, entier2)
```

```
  If (entier1 = entier2) Then
```

```
    Absolute_diff = 0
```

```
    ExitFunction
```

```
  Endif
```

```
  If (entier1 > entier2) Then
```

```
    Absolute_diff = entier1 - entier2
```

```
  Else
```

```
    Absolute_diff = entier2 - entier1
```

```
  Endif
```

```
EndFunction
```

```
' Programme principal
```

```
Nombre = 89
```

```
Resultat = Absolute_diff(Nombre, 123)
```

```
Msgbox(Resultat)
```

Exercice 12

Cet exercice de programmation illustre la transformation d'un script existant en une Fonction qui peut être réutilisée. Cet exercice utilise le scripts réalisé lors des exercices 9 et 11. La solution de l'exercice 12 est donnée en appendice C.

1. Démarrez l'Editeur WinTask. Si l'Assistant de Création de Script s'affiche, cliquez sur le bouton **Annuler**. Si le script **script11.src** est affiché, passez directement à l'étape 3.
2. Cliquez sur l'icône **Ouvrir** de la barre d'outils WinTask et ouvrez le script **script11.src**.
3. Dans l'Editeur WinTask, sélectionnez le menu **Fichier/Enregistrer sous**. Dans la boîte de dialogue **Enregistrer sous**, enregistrez sous le nom **script12.src**. Cette étape permet de conserver le script **script11.src** tel qu'il était à l'exercice 11.
4. Cliquez sur l'icône **Ouvrir** de la barre d'outils WinTask et ouvrez le script **script09.src**.
5. Sélectionnez toutes les lignes de **script09.src** et copiez-les dans le Presse-papiers. La fenêtre de **script09.src** peut être désormais fermée (sans enregistrer).
6. Dans l'Editeur WinTask, cliquez sur l'onglet **script12.src** pour mettre en premier plan le script 12. Collez les lignes précédemment copiées après la ligne **EndSub** et avant la partie Programme principal.
7. Transformez les actions effectuées par **script09.src** en une Fonction de nom **Saisie_Wordpad** utilisant un seul paramètre, **NomFichier\$**. Ce paramètre spécifie le nom de fichier sous lequel enregistrer le texte saisi dans wordpad. La Fonction doit retourner le nombre de lignes écrites dans WordPad.
8. Dans le programme principal, modifiez les paramètres de **Saisie_Notepad** afin que le premier paramètre contienne désormais **"text12.txt"**, que le texte soit **"Exercice Sub et Function"**, et que **8** lignes soient générées. Appelez la fonction **Saisie_Wordpad** après l'appel à **Saisie_Notepad**. Spécifiez le nom de fichier à passer à **Saisie_Notepad**. Assignez la valeur retournée par **Saisie_Wordpad** à une variable. Utilisez l'instruction *MsgBox* pour faire afficher le nombre de lignes écrites dans le fichier.
9. Lancez l'exécution en cliquant sur l'icône **Exéc** dans la barre d'outils WinTask. Si la fenêtre de Résultats de la compilation montre des erreurs, corrigez-les puis lancez l'exécution à nouveau.
10. Cliquez sur le bouton OK lorsque le message s'affiche et fermez WordPad sans sauver. Le nombre de lignes affiché doit être 9.

Assembler les morceaux

Avec les exercices de ce chapitre, vous savez désormais structurer un script d'automatisation pour le rendre lisible et rendre sa maintenance aisée. Vous pouvez consulter l'aide sur les instructions *Include* et *Run* pour rendre encore plus modulaire un script complexe.

Exercice 13

Cet exercice ajoute l'utilisation d'un tableau, en se servant du code écrit aux exercices 10 et 12. La solution de l'exercice 13 est donnée en appendice C.

1. Démarrez Excel. Dans Excel, sélectionnez le menu **Fichier/Enregistrer sous**.
2. Enregistrez le fichier sous le nom **exercice13.xls** dans le répertoire courant (sous Excel 2007, l'extension est **xlsx**). Quittez Excel.
3. Démarrez l'Editeur WinTask. Si l'Assistant de Création de Script s'affiche, cliquez sur le bouton **Annuler**. Si le script **script12.src** est affiché, passez directement à l'étape 5, sinon sélectionnez l'option **Fichier/Fermer tout**.
4. Cliquez sur l'icône **Ouvrir** de la barre d'outils WinTask et ouvrez le script **script12.src**.
5. Dans l'Editeur WinTask, sélectionnez le menu **Fichier/Enregistrer sous**. Dans la boîte de dialogue **Enregistrer sous**, enregistrez sous le nom **script13.src**. Cette étape permet de conserver le script **script12.src** tel qu'il était à l'exercice 12.
6. Cliquez sur l'icône **Ouvrir** de la barre d'outils WinTask et ouvrez le script **script10.src**.
7. Copiez la déclaration du tableau (**DIM**) située au début du script **script10.src** dans le presse-papiers. Cliquez sur l'onglet **script13.src** pour faire afficher la fenêtre **script13.src** et collez l'instruction Dim au début du script.

ASTUCE : Un tableau ne peut pas être passé en paramètre. Un tableau doit être déclaré globalement pour l'ensemble du script.

8. Basculez vers la fenêtre **script10.src** et copiez le reste du script dans le presse-papiers. Revenez à la fenêtre **script13.src** et collez le contenu du presse-papiers après la ligne **EndFunction** et avant la partie Programme principal.
9. Transformez les actions effectuées par le script **script10.src** en une Sub de nom **Saisie_Excel** avec deux paramètres, **NomFichier\$** et **NomFichierExcel\$**. Le paramètre **NomFichier\$** spécifie le nom du fichier texte à lire. Le paramètre **NomFichierExcel\$** spécifie le nom du fichier Excel à mettre à jour.

ASTUCE : L'appel à l'instruction **Read** dans la Sub **Saisie_Wordpad** met le pointeur de lecture en fin de fichier. Comme la Sub **Saisie_Excel** lit le même fichier, le pointeur de lecture se trouvera en fin de fichier et donc rien ne sera lu. Utilisez l'instruction **SetReadPos** avant de commencer à lire le

fichier dans **Saisie_Excel** pour positionner le pointeur de lecture sur le premier enregistrement.

10. Supprimez la ligne **MsgBox** dans la Sub **Saisie_Excel**.
11. Le tableau a été déclaré pour un maximum de 25 éléments. Ajoutez une condition dans la ligne **While** pour tester que la boucle ne va pas au-delà de cette limite.
12. Dans cet exercice, le fichier texte généré n'aura que 5 lignes. Remplacez "**A1:J1**" et "**C3:C12**" par respectivement "**A1:E1**" et "**C3:C7**".
13. Optionnel et plus complexe : Remplacez "**A1:E1**" et "**C3:C7**" par des rangées de cellules construites dynamiquement pour prendre en compte combien d'éléments ont été lus dans la boucle précédente. La rangée ne sera plus C3 à C7 mais C3 à Cn, et la ligne écrite commencera en A1 mais ira jusqu'à une lettre à déterminer par le script.

ASTUCE : Utilisez les instructions **Asc**, **Chr\$**, **Str\$** et **Val** pour déterminer la lettre de la dernière cellule à écrire et le numéro de ligne de la dernière cellule à écrire dans la colonne C.

14. Modifiez les paramètres d'appel de **Saisie_Notepad** dans le programme principal pour utiliser comme nom de fichier "**text13.txt**", comme texte à saisir "**Exercice Sub, Func et Excel**", et **5** comme nombre de lignes à générer. Appelez **Saisie_Excel** avec les paramètres corrects, après l'appel à **Saisie_Wordpad**. Spécifiez le même nom de fichier pour le premier paramètre que celui utilisé comme premier paramètre de **Saisie_Notepad** et **Saisie_Wordpad**. Spécifiez "**exercice13.xls**" pour le deuxième paramètre.
15. Lancez l'exécution en cliquant sur l'icône **Exéc** dans la barre d'outils WinTask. Si la fenêtre de Résultats de la compilation montre des erreurs, corrigez-les puis lancez l'exécution à nouveau.
16. Cliquez sur le message donnant le nombre de lignes et fermez WordPad sans enregistrer. Ouvrez le fichier Excel pour vérifier que les données ont été écrites aux deux endroits désirés.

Cet exercice a illustré comment un tableau peut être utilisé dans un script structuré. Vous savez maintenant écrire des scripts complexes, partant du mode Enregistrement pour générer un canevas et utilisant le langage pour transformer les actions à répéter en sous-programmes.

Débogage

Erreurs de compilation

Quand vous lancez l'exécution d'un script (fichier SRC), ce dernier est d'abord compilé. Les erreurs de compilation éventuelles sont affichées dans la fenêtre Résultats de l'Editeur (si la fenêtre Résultats n'est pas affichée, sélectionnez **Affichage/Fenêtre Résultats** pour la faire afficher).

En double-cliquant sur une ligne rapportant une erreur, le curseur texte se met dans la fenêtre principale de l'Editeur, sur la ligne où l'erreur se produit.

Souvenez-vous de la structure d'un script :

Les instructions Dim déclarant des tableaux en premier, les déclarations des Sub et Fonctions en deuxième, et l'appel à une fonction à l'intérieur d'une autre nécessite que la fonction appelée ait été déclarée avant, le programme principal en dernier.

Un script comportant des scripts inclus doit respecter cet ordre une fois toutes les inclusions effectuées à la compilation

Un exemple d'instruction Include : Include "ma_boite_outils.src"

Erreurs d'exécution

Trace en utilisant MsgBox ou MsgBoxFrame

Exercice 14

1. Démarrez l'Editeur WinTask. Si l'Assistant de Création de Script s'affiche, cliquez sur le bouton **Annuler**. La barre de titre doit être **WinTask – [SansNom1]**. Si WinTask ouvre un script écrit précédemment, sélectionnez **Fichier/Fermer** puis cliquez sur l'icône **Nouveau** de la barre d'outils.
2. Cliquez sur l'icône **Enreg** de la barre d'outils pour démarrer le mode Enregistrement La boîte de dialogue **Démarrer l'enregistrement** s'affiche, cliquez sur le bouton **OK**.
3. La boîte de dialogue **Lancement d'une application** s'affiche. Tapez **Notepad** dans le champ **Programme** et cliquez sur le bouton **OK**.
4. Tapez **Bonjour** dans notepad et appuyez sur la touche **Entrée**.
5. Fermez notepad sans sauver.

6. Arrêtez le **Mode Enregistrement** en cliquant sur la première icône de la barre d'outils WinTask flottante.
7. Modifiez le script ainsi généré en ajoutant une boucle sur deux index, i et j, afin de taper Bonjour i et Bonjour j dans notepad. Démarrez l'index i à 0 en le faisant incrémenter dans la boucle. Démarrez l'index j à 10 en le faisant décrémenter. Répétez la boucle jusqu'à ce que i=8.
8. Lancez l'exécution en cliquant sur l'icône **Exéc** dans la barre d'outils WinTask. Enregistrez le script sous le nom **script14**. Si la fenêtre de Résultats de la compilation montre des erreurs, corrigez-les puis lancez l'exécution à nouveau.

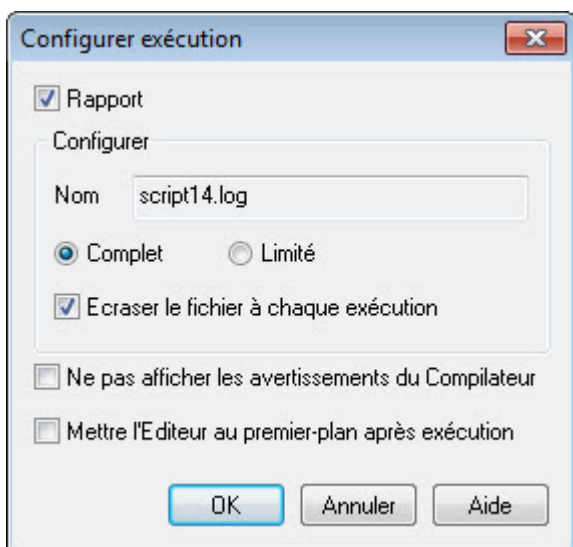
En imaginant que l'index j est à surveiller car une erreur se produit, insérez dans le script une instruction *MsgBox* pour faire afficher la valeur de l'index à chaque itération. Lancez l'exécution. Remplacez la ligne *MsgBox* par *MsgFrame* et rejouez.

ASTUCE : Notez la grande différence entre *MsgBox* et *MsgFrame*. *MsgBox* prend le focus et l'utilisateur doit cliquer sur OK après l'affichage pour que le script continue son exécution. *MsgFrame* par contre ne modifie aucunement le comportement applicatif.

Fichier Rapport

Il est possible de paramétrer l'exécution pour que chaque ligne soit écrite dans un fichier rapport (fichier d'extension .LOG).

Sélectionnez le menu **Paramétrer/Exécution** :



Si la case **Complet** est cochée, toutes les lignes exécutées sont écrites dans le fichier rapport. Si **Limité** est coché, seules les instructions *Comment* sont écrites dans le rapport.

Exercice 15

1. Démarrez l'Editeur WinTask. Si l'Assistant de Création de Script s'affiche, cliquez sur le bouton **Annuler**. Si le script **script14.src** n'est pas le script chargé dans la fenêtre de l'Editeur, cliquez sur l'icône **Ouvrir** de la barre d'outils de l'Editeur et ouvrez le script **script14.src**.
2. Paramétrez l'exécution pour générer un rapport complet (sélectionnez le menu **Paramétrer/Exécution**) et lancez l'exécution (icône **Exéc** de la barre d'outils).
3. Ouvrez le fichier rapport généré (sélectionnez le menu **Fichier/Ouvrir Rapport**) – étudiez les lignes générées. Fermez le fichier log.
4. Remplacez la ligne msgbox ou msgframe par la ligne Comment ("l'index i vaut :"+str\$(i)).
5. Ajoutez une autre ligne Comment pour écrire dans le fichier rapport la valeur de l'index j.
6. Enregistrez le script modifié sous le nom **script15** (menu Fichier/Enregistrer sous).
7. Paramétrez maintenant l'exécution pour générer un rapport limité.
8. Cliquez sur l'icône **Exéc** de la barre d'outils de l'Editeur pour lancer l'exécution. Si la fenêtre de Résultats de la compilation montre des erreurs, corrigez-les puis lancez l'exécution à nouveau.
9. Ouvrez le fichier rapport (menu **Fichier/Ouvrir Rapport**) et étudiez les lignes générées.

Création de votre propre fichier log :

Vous pouvez écrire vous-même une Sub écrivant les informations que vous désirez dans un fichier, voici un exemple :

```
sub log(msg_log$)
  local buffer$
  buffer$=Date$()+", "+hour$()+": "+min$()+": "+sec$()+ " --> "+msg_log$
  write(fichier_log$,buffer$,CRLF)
endsub
```

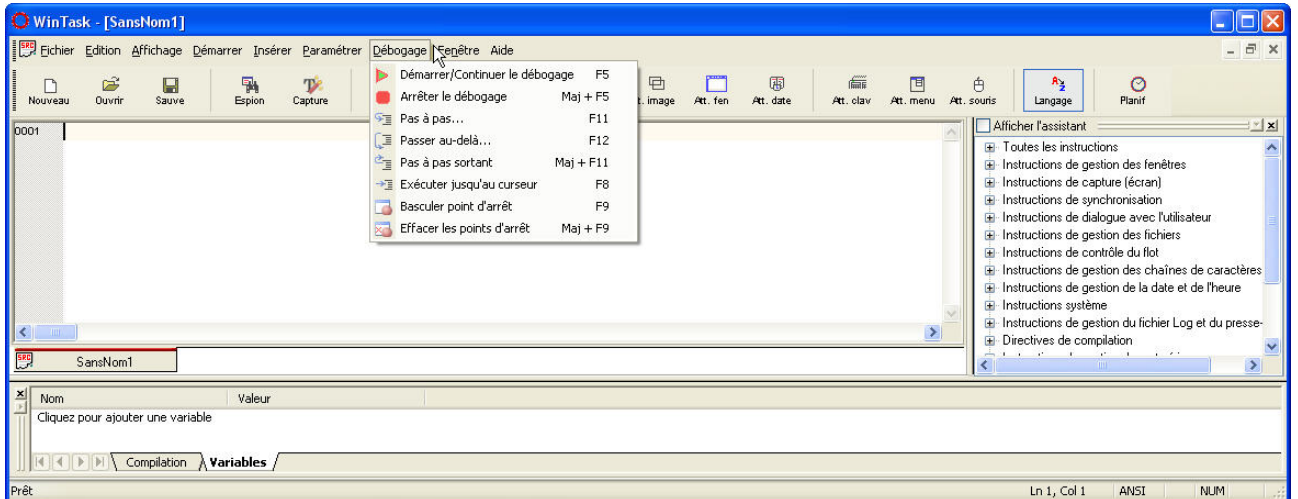
En reprenant le script écrit dans l'exercice 15, intégrez cette Sub dans le script (en début de script) et remplacez les deux lignes COMMENT par deux lignes appelant cette Sub. Ajoutez une ligne pour assigner la variable fichier_log\$ à un nom de fichier d'extension .txt et enregistrez le script ainsi modifié sous le nom script15b. Lancez l'exécution et ouvrez le fichier généré.

Exécution en mode Débogage

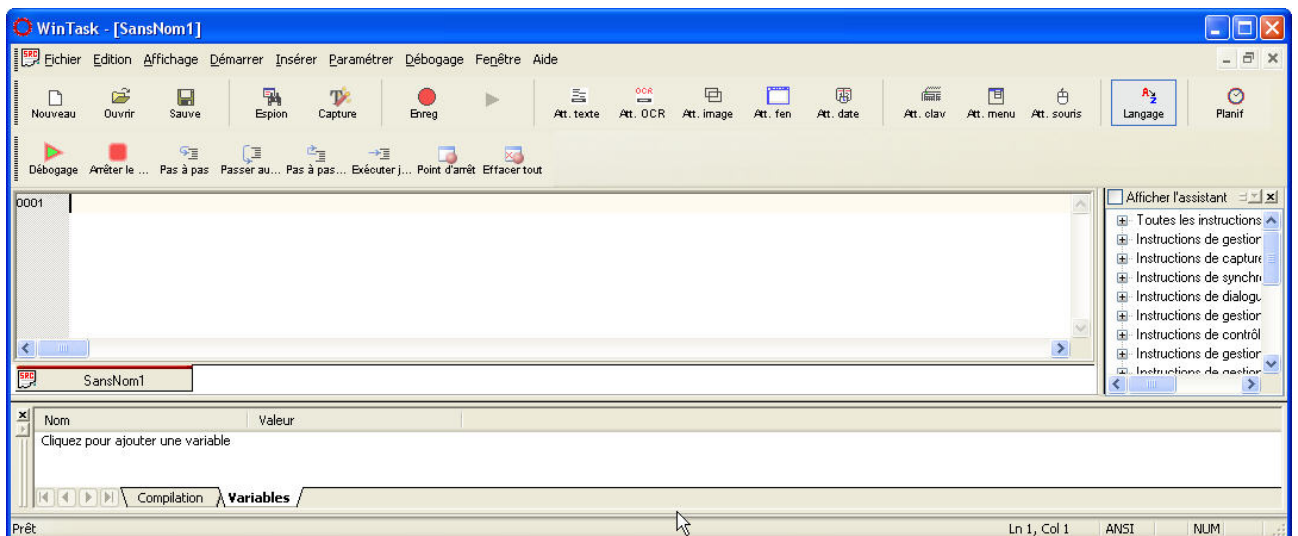
WinTask inclut un débogueur de scripts. En exécutant un script en mode débogage,

vous pouvez arrêter l'exécution du script à une certaine ligne, afficher le contenu des variables à ce point d'arrêt, puis reprendre l'exécution jusqu'au point d'arrêt suivant que vous avez spécifié.

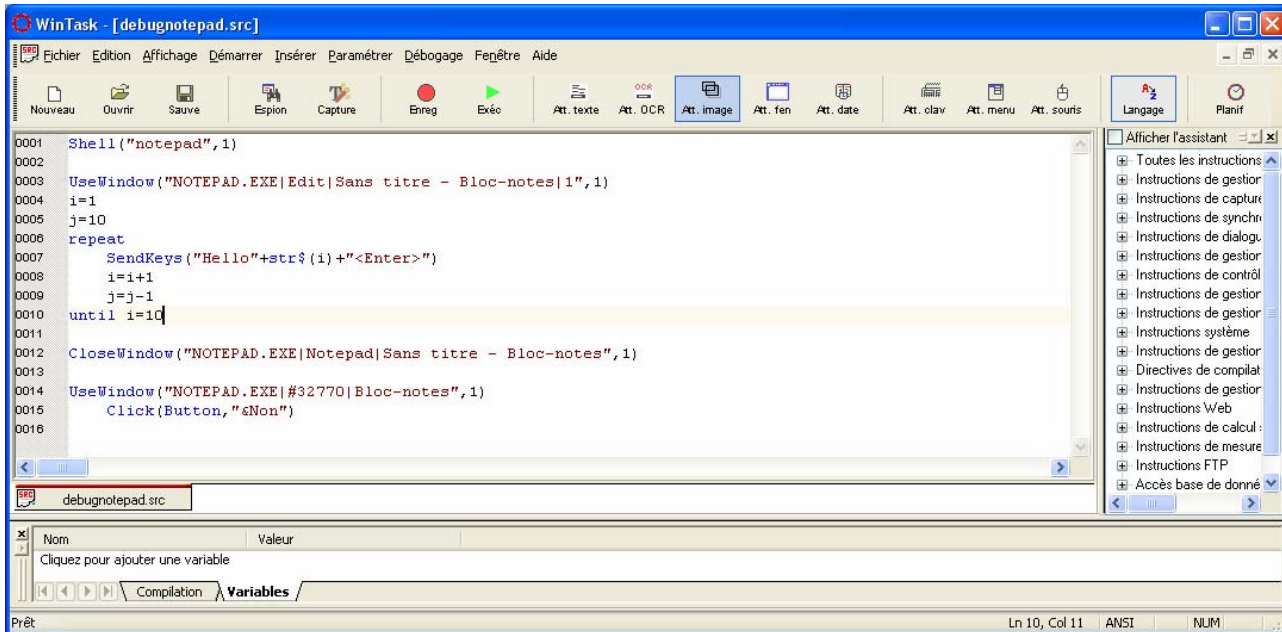
Toutes les fonctionnalités du mode Débogage sont accessibles depuis le menu **Débogage** dans la fenêtre de l'Editeur WinTask. Vous pouvez également faire afficher la barre d'outils Débogage en sélectionnant le menu **Affichage/Barre d'outils Débogage**.



La fenêtre de l'Editeur avec la barre d'outils Débogage :



Pour utiliser sur un exemple le mode Débogage, créez le petit script tel qu'affiché ci-dessous :

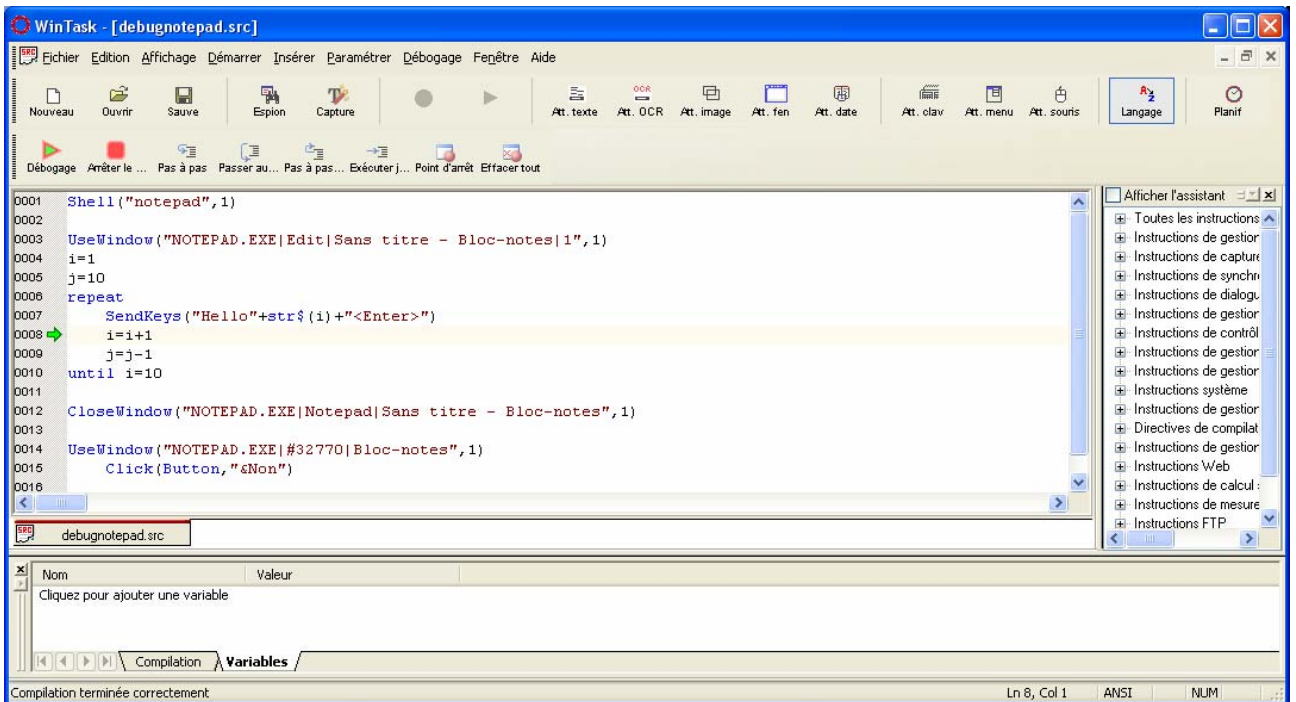


Ce script enregistré sous le nom "debugnotepad" lance notepad. Puis dans une boucle, il saisit dans la fenêtre notepad Hello1, puis Hello2, etc... jusqu'à ce que l'index i vaille 10. L'index j n'est pas utilisé, il est là juste pour illustrer le fonctionnement du mode Débogage. Notez que si vous utilisez Vista ou Windows 7, la fermeture de notepad ligne 14 utilise un nom de fenêtre différent et le bouton est Ne pas enregistrer au lieu de Non.

La manière la plus simple d'utiliser le mode Débogage est de mettre le curseur devant la ligne où l'exécution doit être suspendue, puis de visualiser le contenu des variables désirées à ce point d'arrêt. Donc mettez le curseur en début de ligne 8 et sélectionnez le menu **Débogage/Exécuter jusqu'au curseur**, ou cliquez l'icône **Exécuter**

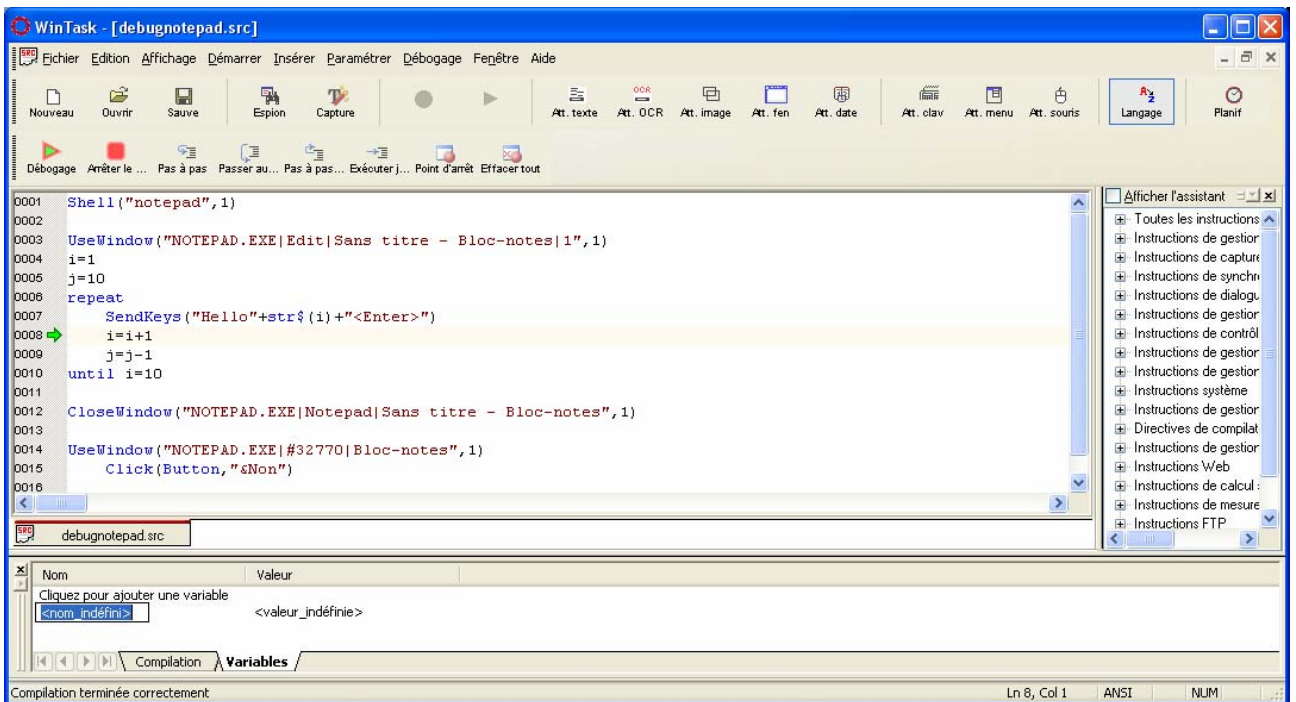
jusqu'au curseur dans la barre d'outils Débogage.

Dés que vous avez sélectionné l'exécution jusqu'au curseur, la fenêtre de l'Editeur est mise en réduction et notepad se lance. Vous voyez Hello1 tapé dans notepad, puis le script arrête son exécution et la fenêtre de l'Editeur revient au premier-plan avec une flèche verte sur la ligne où le script s'est arrêté :

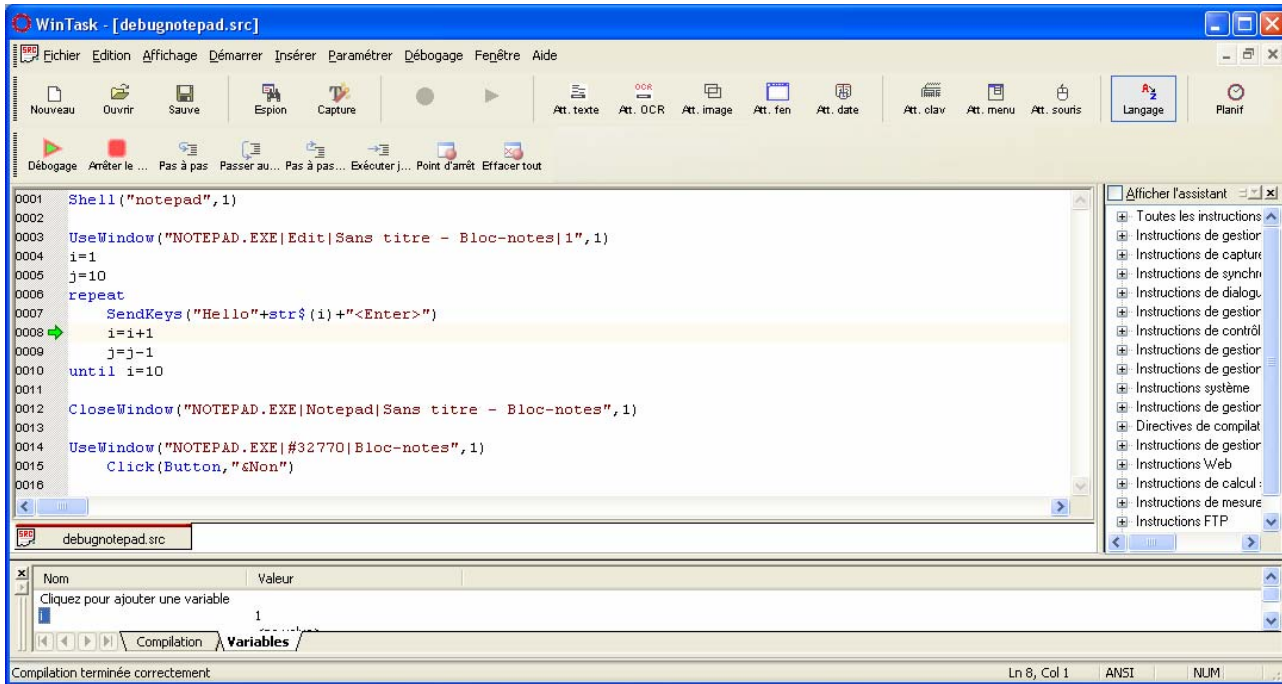


Avant de continuer l'exécution, vous voulez connaître la valeur de la variable i.

Dans la fenêtre Résultats de l'Editeur (la fenêtre en bas), cliquez sur **Cliquez pour ajouter une variable** :



Tapez **i** dans le champ <nom indéfini> et appuyez sur la touche **Entrée**, la valeur de i s'affiche :



Pour reprendre l'exécution du script, sélectionnez le menu **Débogage/Démarrer/Continuer le débogage** ou cliquez sur l'icône **Débogage**



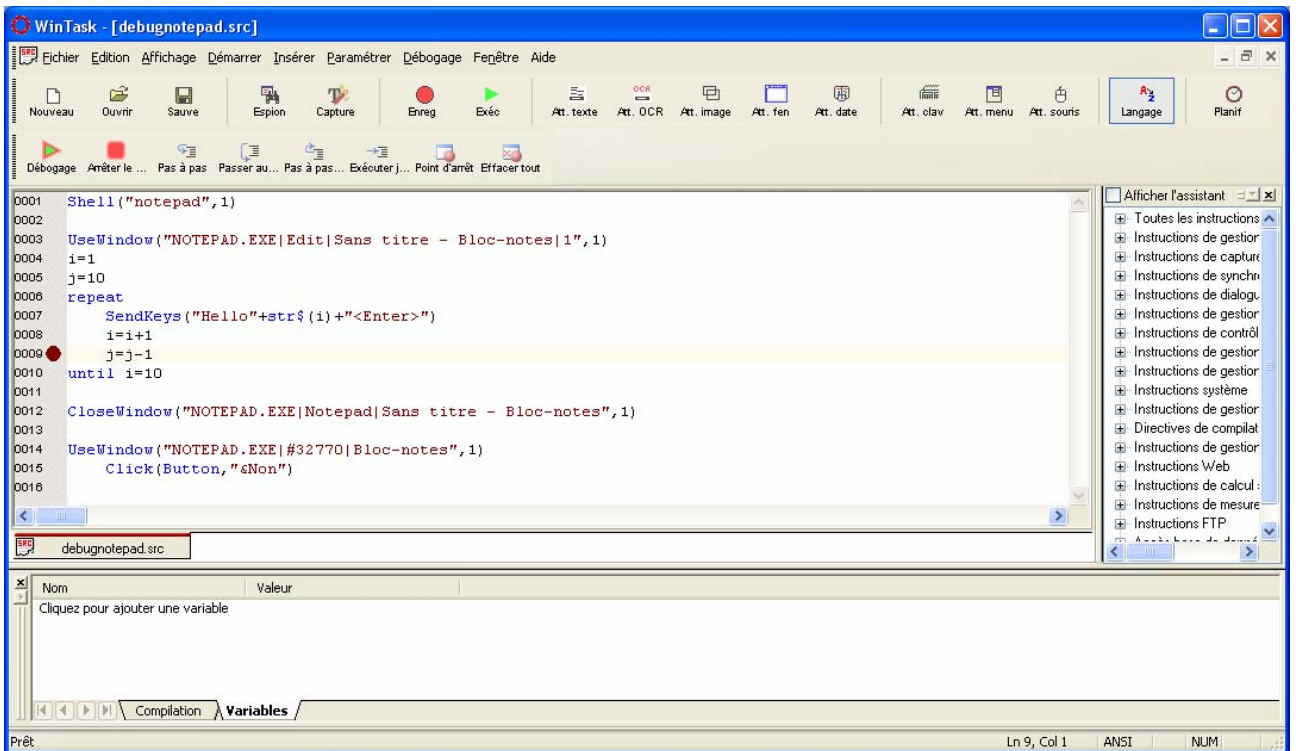
dans la barre d'outils Débogage.

Pour suspendre l'exécution plusieurs fois dans le script, utilisez les points d'arrêt. Par exemple, insérez un point d'arrêt à la ligne 9 : mettez le curseur en début de ligne 9 puis sélectionnez le menu **Débogage/Basculer point d'arrêt** ou cliquez sur l'icône



Point d'arrêt dans la barre d'outils Débogage.

La fenêtre de l'Editeur affiche un point rouge pour le point d'arrêt :

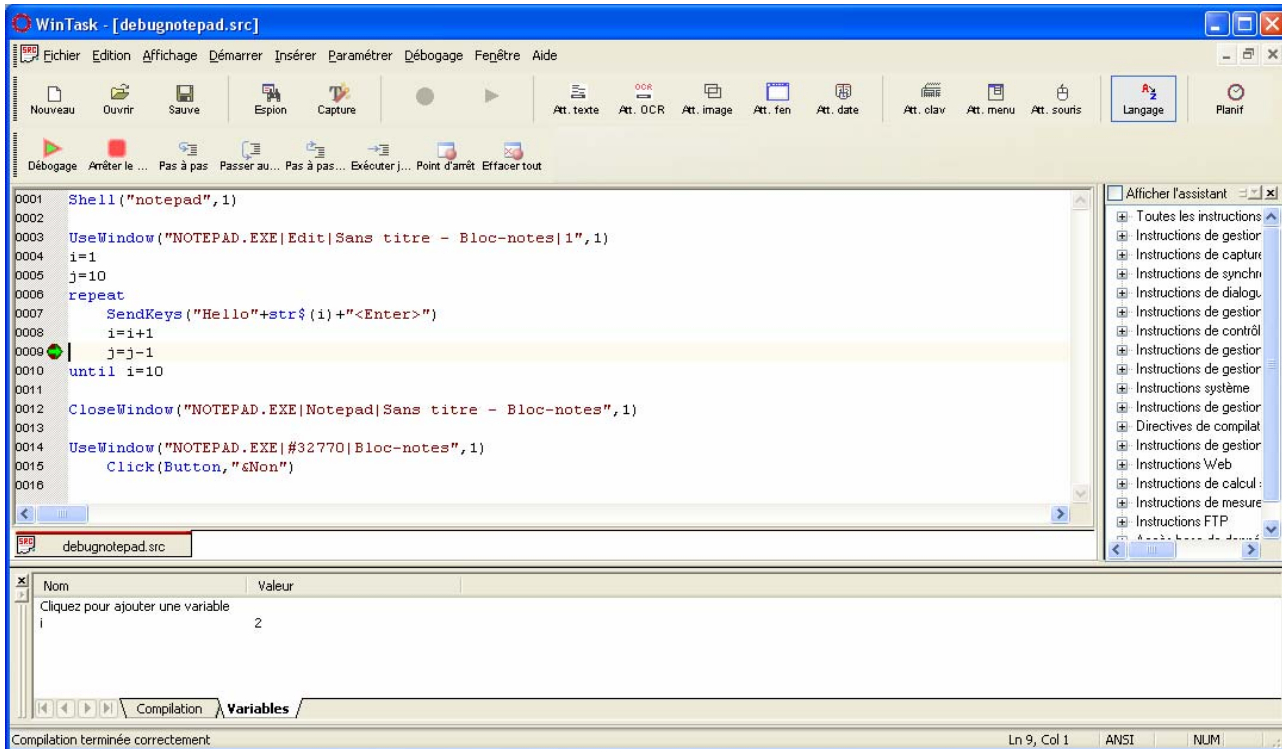


Démarrez alors l'exécution en mode Débogage : sélectionnez le menu **Débogage/Démarrer/Continuer le débogage** ou cliquez sur l'icône **Débogage**

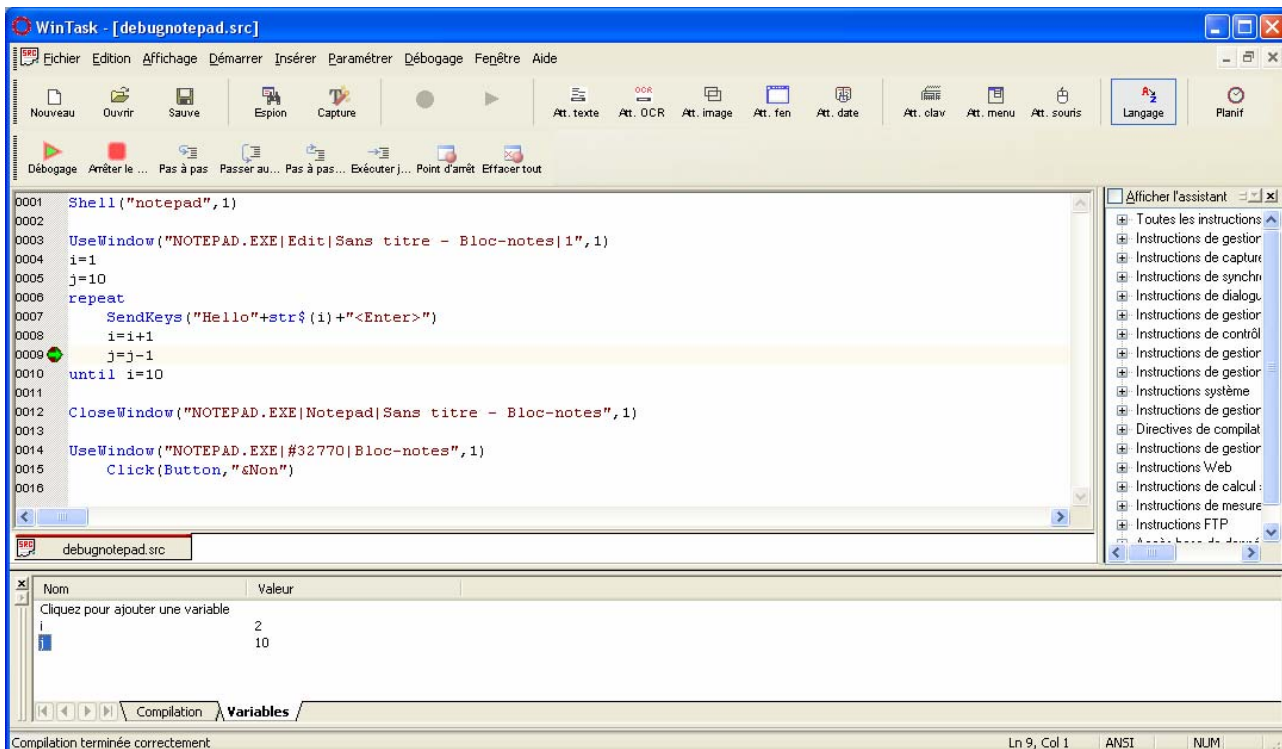


dans la barre d'outils de Débogage. Le script s'exécute jusqu'au point d'arrêt sans exécuter la ligne où est située le point d'arrêt.

Au point d'arrêt, la fenêtre de l'Editeur revient au premier-plan, la flèche verte est à la ligne 9, vous voyez la nouvelle valeur de la variable i qui est maintenant à 2 dans la fenêtre Résultats en bas.



Vous pouvez ajouter une nouvelle variable à visualiser à ce point d'arrêt en cliquant sur Cliquez pour ajouter une variable, ajoutez la variable j :



Poursuivez l'exécution avec le point d'arrêt toujours présent : cliquez sur l'icône



Débogage de la barre d'outils Débogage. A chaque itération, vous voyez la fenêtre de l'Editeur revenir au premier plan juste avant l'exécution de la ligne 9 et le contenu des variables i et j s'affiche.

Pour supprimer le point d'arrêt, mettez le curseur sur la ligne où se situe le point d'arrêt et sélectionnez le menu **Débogage/Basculer point d'arrêt** ou cliquez sur

l'icône **Point d'arrêt**  dans la barre d'outils Débogage.

Pour arrêter l'exécution du script en mode débogage, sélectionnez le menu

Débugage/Arrêter le débogage ou cliquez sur l'icône Arrêter le débogage



Objet non trouvé (fenêtre, bouton, option de menu, ...)

C'est l'erreur d'exécution la plus courante.

Pour comprendre d'où vient l'erreur, notez le numéro de ligne indiqué par le message d'erreur et cliquez sur le bouton OK pour faire disparaître le message. Vous êtes alors sur l'écran de l'application.

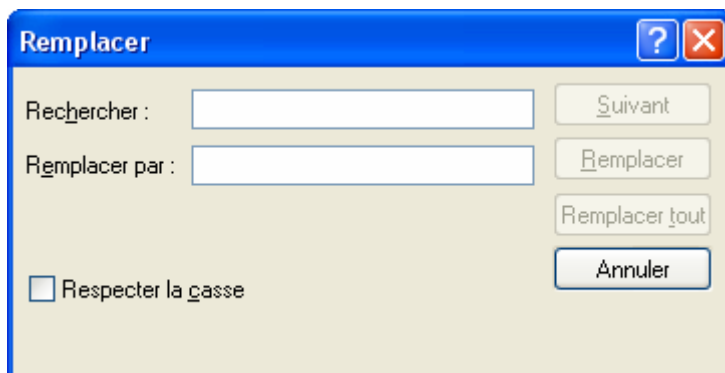
- L'objet est-il vraiment là ? Si ce n'est pas le cas, un problème de synchronisation en est sans doute la cause. Pour vérifier, ajoutez une ligne Pause x secs juste au-dessus de la ligne qui a provoqué l'erreur. Fermez toutes les applications ouvertes par le script et rejouez. Si l'erreur ne se produit plus, vous pouvez alors remplacer la pause absolue par une synchronisation plus élaborée.
- L'objet est là. Utilisez alors l'Espion pour vérifier que le nom de l'objet lors du rejoue est bien le même que celui spécifié dans le script.

Le script est long et vous ne voyez pas pourquoi cette erreur intervient. Restez sur la fenêtre de l'application où l'erreur se produit, extrayez dans un nouveau script les quelques lignes en cause dans le long script. Lancez l'exécution de ce petit script, le débogage est beaucoup plus facile à réaliser.

Saisie incorrecte dans un champ (WriteEdit, WriteCombo versus SendKeys)

WinTask identifie les champs d'un formulaire en utilisant un index. Si au rejoue, cet index a change, les champs se remplissent incorrectement.

Par exemple dans Notepad, menu Edition/Remplacer, la boîte de dialogue Remplacer :



Le mode Enregistrement avec lequel vous enregistrez la saisie de **Bonjour** dans le

champ Rechercher, et la saisie de **Bonjour 1** dans Remplacer par génère :

```
UseWindow("NOTEPAD.EXE/#32770/Remplacer", 1)
WriteEdit("1", "Bonjour")
WriteEdit("2", "Bonjour 1")
```

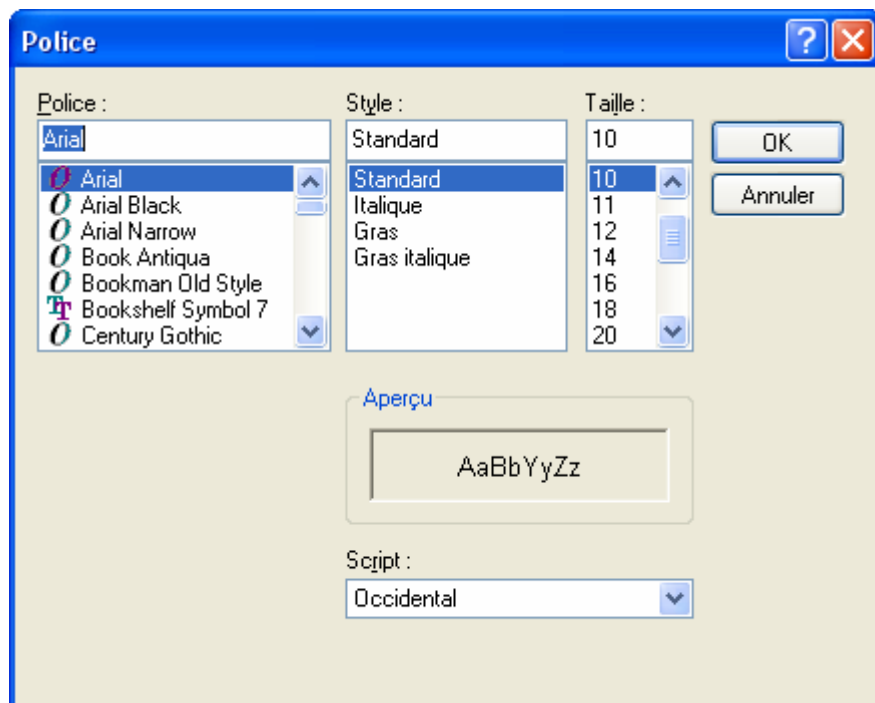
Le paramètre "1" de l'instruction WriteEdit indique qu'il faut écrire dans le champ 1 de la boîte de dialogue. Donc dans une application réelle, si le numéro de champ change, l'instruction n'écrira pas correctement. Dans ce cas, utilisez l'instruction SendKeys et le caractère de Tabulation pour passer d'un champ à un autre. Dans notre petit exemple, voici le résultat :

```
UseWindow("NOTEPAD.EXE/#32770/Remplacer", 1)
SendKeys("Bonjour", NoActivate)
pause 5 ticks
SendKeys("<Tab>", Noactivate)
pause 5 ticks
SendKeys("Bonjour 1", NoActivate)
```

Le mot-clé NoActivate indique que Sendkeys doit envoyer les caractères spécifiés à l'endroit où est le curseur, sans se préoccuper du nom de fenêtre du champ. Notez l'utilisation des lignes Pause 5 ticks pour ajouter un tout petit délai pour laisser le temps à l'application de passer au champ suivant.

Un élément d'une liste n'est pas correctement sélectionné

Prenons un exemple à nouveau dans Notepad, la boîte de dialogue Police (menu **Format/Police**) :



Le mode Enregistrement avec lequel vous enregistrez la sélection de **Gras** dans la liste Style génère :

```
UseWindow("NOTEPAD.EXE/#32770/Police", 1)
ChooseItem(Combo, "2", "Gras", single, left)
```

Le code généré se rejoue correctement car notepad utilise les listes et combo standard de Windows. Mais si l'application à automatiser a été développée avec des outils non standard Windows, la sélection dans une liste peut ne pas se rejouer correctement. Utilisez alors la touche Down autant de fois que nécessaire pour sélectionner l'élément désiré. Dans notre petit exemple :

```
UseWindow("NOTEPAD.EXE/#32770|Font", 1)
SendKeys("<Tab>")
Pause 5 ticks
SendKeys("<Down> <Down>", Noactivate)
```

Option de Menu non sélectionnée

Au rejoue, un ChooseMenu devant sélectionner une option dans un menu, ne sélectionne pas l'option désirée. Vous pouvez alors utiliser les touches raccourci :

Par exemple dans notepad, le menu Fichier **Fichier/Enregistrer** :

```
UseWindow("NOTEPAD.EXE/Notepad/Sans titre - Bloc-notes")
SendKeys("<Alt f>", Noactivate)
pause 5 ticks
SendKeys("e", NoActivate)
```

Le nom de fenêtre spécifié par l'instruction UseWindow doit être le nom de la fenêtre menu (utilisez comme d'habitude l'Espion pour trouver le nom), et notez à nouveau la ligne Pause pour que le menu Fichier ait le temps de s'ouvrir avant d'envoyer la deuxième touche de raccourci.

Un clic ne s'exécute pas

Un cas typique est le clic sur une icône incluant du texte. Le plus souvent, WinTask n'est pas capable de reconnaître un tel objet.

Trois instructions sont disponibles pour cliquer sur un objet si l'instruction Click ne fonctionne pas :

ClickOnText Si le texte est inclus dans une icône, le texte ne sera pas reconnu.
ClickOnBitmap Si aucun texte n'est dans l'icône, utilisez cette instruction
ClickOnTextOCR Le clic le plus "intelligent", le seul inconvénient est que la reconnaissance OCR prend quelques dixièmes de secondes.

Un assistant est disponible pour chacune de ces instructions. Pour y accéder, dans la fenêtre de droite de l'Editeur (fenêtre Langage), double-cliquez sur le nom de l'instruction désirée (sélectionnez le menu **Insérer/Instruction** ou appuyez sur la touche F4 si la fenêtre Langage n'est pas affichée).

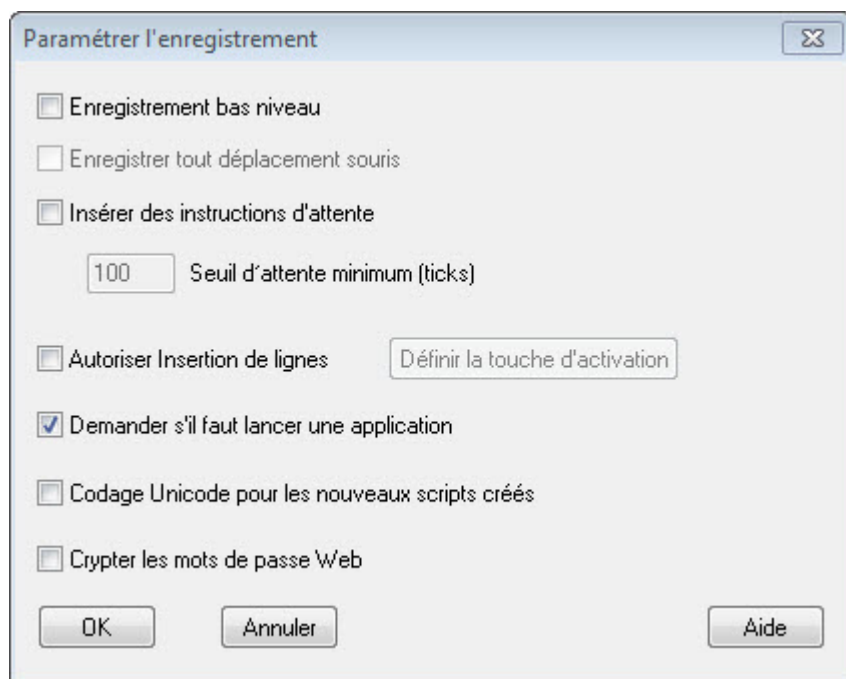
Enregistrement Bas Niveau quand rien d'autre ne marche

Avec cet autre mode d'Enregistrement, les actions sont enregistrés par des clics souris

et donc l'enregistrement n'est plus orienté objet.

N'utilisez ce mode que pour enregistrer les quelques actions qui ne sont pas rejouées correctement si elles ont été enregistrées en mode objet.

Configurez l'Enregistrement en bas niveau en sélectionnant le menu **Paramétrer/Enregistrement**.



Par exemple, voici ce que génère l'enregistrement du menu Fichier/Ouvrir de notepad, d'abord en normal, puis en bas niveau :

```
Shell("notepad", 1)
```

```
UseWindow("NOTEPAD.EXE/Notepad/Sans titre - Bloc-notes", 1)  
ChooseMenu(Normal, "&Fichier|&Ouvrir... Ctrl+O")
```

```
UseWindow("NOTEPAD.EXE/#32770/Ouvrir", 1)  
Click(Button, "Annuler")
```

```
CloseWindow("NOTEPAD.EXE/Notepad/Sans titre - Bloc-notes", 1)
```

En bas niveau :

```
Shell("notepad", 1)
```

```
UseWindow("NOTEPAD.EXE/Notepad/Sans titre - Bloc-notes", 1, NoActivate)  
ClickMouse(Left, Down, 5, 38)  
ClickMouse(Left, Up, 5, 38)  
ClickMouse(Left, Down, 33, 81)  
ClickMouse(Left, Up, 33, 81)
```

```
UseWindow("NOTEPAD.EXE/Button/Annuler!", 1, NoActivate)  
ClickMouse(Left, Down, 50, 14)  
ClickMouse(Left, Up, 50, 14)
```

```
UseWindow("NOTEPAD.EXE/Notepad/Sans titre - Bloc-notes", 1)
ClickMouse(Left, Down, 1082, 6)
ClickMouse(Left, Up, 1082, 6)
```

Les coordonnées souris générées sont différentes d'un PC à un autre.

Le rejoue est lent

Exercice 16

1. Démarrez notepad et tapez Bonjour dans notepad.
2. Démarrez l'Editeur WinTask. Si l'Assistant de Création de Script s'affiche, cliquez sur le bouton **Annuler**. La barre de titre doit être **WinTask – [SansNom1]**. Si WinTask ouvre un script écrit précédemment, sélectionnez **Fichier/Fermer** puis cliquez sur l'icône **Nouveau** de la barre d'outils. Cette fenêtre va être utilisée pour créer un nouveau script.
3. Vérifiez que le mode Enregistrement est en mode normal : sélectionnez le menu **Paramétrer/Enregistrement** et décochez la case **Bas niveau** si elle est cochée. Cliquez sur le bouton **OK**.
4. Cliquez sur l'icône **Enreg** de la barre d'outils pour démarrer le mode Enregistrement. La boîte de dialogue **Démarrer l'enregistrement** s'affiche, cochez le bouton **Rien** et cliquez sur le bouton **OK**.
5. Enregistrez ces deux actions : sélection du menu **Fichier/Imprimer** et clic sur le bouton **Annuler**.
6. Arrêtez le **Mode Enregistrement** en cliquant sur la première icône de la barre d'outils WinTask flottante.
7. Fermez notepad et enregistrez le document dans le répertoire courant sous le nom **Document1**.
8. Ouvrez manuellement le document de nom Document1.
9. Lancez l'exécution du script que vous venez de créer, enregistrez-le sous le nom **script16**.

Pourquoi le script attend un peu avant de rejouer la sélection Fichier/Ouvrir ?
Comment éviter ce délai ?

Ce délai est dû à cette ligne :

```
UseWindow("NOTEPAD.EXE/Notepad/Sans titre - Bloc-notes", 1)
```

Quand le script se rejoue après que vous ayez ouvert le fichier texte Document1, le nom de fenêtre n'est plus "NOTEPAD.EXE/Notepad/Sans titre - Bloc-notes" mais "NOTEPAD.EXE/Notepad/Document1 – Bloc-notes".

Au rejoue, WinTask recherché d'abord une fenêtre de nom exactement celui spécifié

dans UseWindow, puis au bout d'environ 3 secondes (valeur par défaut issue de la valeur par défaut de la variable directive #ActionTimeout), commence à tronquer le nom de fenêtre en commençant par la droite.

Si vous tronquez manuellement dans le script le nom de fenêtre pour prendre en compte n'importe quel nom de document texte ouvert dans notepad, la fenêtre sera trouvée immédiatement :

"NOTEPAD.EXE/Notepad/", ce nom de fenêtre est trouvé immédiatement au rejeu, que le titre de la fenêtre soit Sans titre ou Document1.

Deux autres variables directives influent sur la vitesse d'exécution :

#ExecuteDelay

#SendKeysDelay

Consultez l'aide sur ces deux variables.

Conclusion

Avec ces quelques exemples, vous avez pu constater l'extraordinaire flexibilité de WinTask pour automatiser tout type de tâches sur des logiciels Windows.

Nous avons également indiqué nos astuces les plus courantes pour développer des scripts qui se rejouent de manière fiable. Le point le plus important est de rendre le script le plus indépendant possible de l'environnement, si le fichier existe ou pas, si le répertoire courant a changé, ...

Nous n'avons pas traité les mêmes aspects pour l'automatisation de sites Web, vous pouvez consulter des Tutoriaux dédiés Web à l'adresse :

<http://www.wintask.fr/manuels-wintask.php>

Nous vous remercions de nous envoyer vos remarques sur ce manuel à

info@wintask.fr

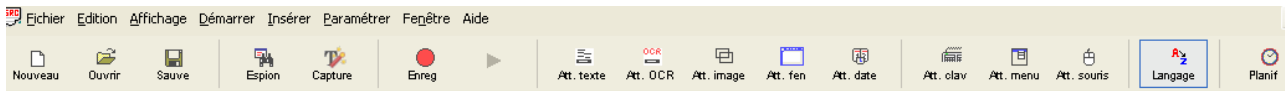
|










|






APPENDICE A

Barre d'outils de l'Editeur WinTask

La barre d'outils de l'Editeur WinTask permet d'accéder aux fonctionnalités listées ci-dessous. Les icônes présentées sont les grandes icônes (menu **Affichage/Grandes icônes**).



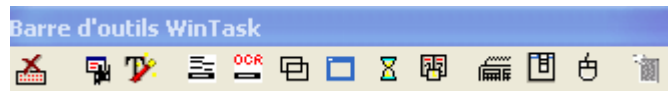
	Crée un nouveau script. Une nouvelle fenêtre s'ouvre.
	Ouvre un script existant. Le script est ouvert dans une nouvelle fenêtre de l'Editeur.
	Enregistre le script. Le script affiché dans la fenêtre courante est sauvé.
	Démarre l'outil Espion. L'Espion permet de trouver les noms des fenêtres se trouvant sur le bureau Windows.
	Démarre l'Assistant de Capture. Il permet de capturer le texte affiché dans une fenêtre.
	Démarre le mode Enregistrement. Toutes les actions utilisateurs seront désormais enregistrées afin de générer un script WinTask dans la fenêtre courante de l'Editeur.
	Compile et lance l'exécution. Le script affiché dans la fenêtre courante est sauvegardé, puis compilé et si aucune erreur de compilation n'est trouvée, l'exécution est lancée.
	Synchronisation Texte. Appelle l'assistant de synchronisation Texte. Les lignes générées par cet assistant sont insérées dans le script courant.
	Synchronisation Texte OCR. Appelle l'assistant de synchronisation Texte OCR. Les lignes générées par cet assistant sont insérées dans le script courant.
	Synchronisation Image. Appelle l'assistant de synchronisation Image. Les lignes générées par cet assistant sont insérées dans le script courant.
	Synchronisation Fenêtre. Appelle l'assistant de synchronisation Fenêtre. Les lignes générées par cet assistant sont insérées dans le script courant.
	Synchronisation Date/Heure. Appelle l'assistant de synchronisation Date/Heure. Les lignes générées par cet assistant sont insérées dans le script courant.

 Att. clav	Attente action Touche. Appelle l'assistant de l'attente clavier. Les lignes générées par cet assistant sont insérées dans le script courant.
 Att. menu	Attente action Menu. Appelle l'assistant de l'attente menu. Les lignes générées par cet assistant sont insérées dans le script courant.
 Att. souris	Attente action Souris. Appelle l'assistant de l'attente souris. Les lignes générées par cet assistant sont insérées dans le script courant.
 Language	Affiche la liste des instructions du langage. La liste est affichée dans une fenêtre à droite dans l'Editeur.
 Planif	Démarre le Planificateur WinTask. Le Planificateur permet de lancer l'exécution de scripts à une certaine date et heure. Le Planificateur est capable d'ouvrir un bureau Windows, de lancer l'exécution et de fermer le bureau (le Planificateur n'est pas disponible sous Vista/Windows 7/2008).



APPENDICE B

Barre d'outils WinTask flottante

En mode Enregistrement, l'Editeur WinTask est mis en réduction et une barre d'outils flottante s'affiche. La barre d'outils WinTask flottante permet d'accéder aux fonctionnalités listées ci-dessous :



	Arrête le mode Enregistrement. Et la fenêtre de l'Editeur est restaurée.
	Démarre l'outil Espion. Lance l'outil Espion sans quitter le mode Enregistrement.
	Démarre l'Assistant de Capture. Les lignes générées par cet assistant sont collées dans le script en cours.
	Synchronisation Texte. Appelle l'assistant de synchronisation Texte. Les lignes générées par cet assistant sont collées dans le script en cours.
	Synchronisation Texte OCR. Appelle les outils OCR inclus dans WinTask.
	Synchronisation Image. Appelle l'assistant de synchronisation Image. Les lignes générées par cet assistant sont collées dans le script en cours.
	Synchronisation Fenêtre. Appelle l'assistant de synchronisation Fenêtre. Les lignes générées par cet assistant sont collées dans le script en cours.
	Synchronisation sur Durée. Insère une pause d'une certaine durée. Cette instruction Pause est collées dans le script en cours.
	Synchronisation Date/Heure. Appelle l'assistant de synchronisation sur Date/Heure. Les lignes générées par cet assistant sont collées dans le script en cours.
	Attente clavier. Appelle l'assistant d'attente clavier. Les lignes générées par cet assistant sont collées dans le script en cours.
	Attente menu. Appelle l'assistant d'attente menu. Les lignes générées par cet assistant sont collées dans le script en cours.

	Attente souris. Appelle l'assistant d'attente souris. Les lignes générées par cet assistant sont collées dans le script en cours.
	Insérer des lignes. Appelle la boîte de dialogue Insérer des lignes dans le script qui permet d'insérer pendant l'Enregistrement des instructions autres que celles générées automatiquement.

APPENDICE C

Solutions des exercices

Les solutions aux exercices de programmation sont fournies ci-dessous.

Exercice 7, Script07a.src

```
=====
' Script07a
'
' Manuel WinTask Version 3.8
' © Copyright 1997-2011 TaskWare   Octobre 2011
'
' Exercice 7, partie 1
'
' Ce script utilise une variable pour saisir un nom de fichier.
=====

' Cette variable de type chaîne de caractères est utilisée ensuite
' comme nom de fichier .txt dans la boîte de dialogue Enregistrer sous
nomfichier$="test07"

Shell("notepad",1)

UseWindow("NOTEPAD.EXE|Edit|Sans titre - Bloc-notes|1",1)
    SendKeys("Bonjour<Enter>")

UseWindow("NOTEPAD.EXE|Notepad|Sans titre - Bloc-notes",1)
    ChooseMenu(Normal,"&Edition|He&ure/Date   F5")

UseWindow("NOTEPAD.EXE|Edit|Sans titre - Bloc-notes|1",1)
    SendKeys("<Enter>")

CloseWindow("NOTEPAD.EXE|Notepad|Sans titre - Bloc-notes",1)

UseWindow("NOTEPAD.EXE|CtrlNotifySink|Bloc-notes|7",1)
    Click(Button,"&Enregistrer")

' La variable nomfichier$ remplace "test07"
UseWindow("NOTEPAD.EXE|FloatNotifySink|Enregistrer sous|1",1)
    WriteCombo("1",nomfichier$)

UseWindow("NOTEPAD.EXE|#32770|Enregister sous",1)
    Click(Button,"&Enregistrer")
```

Exercice 7, Script07b.src

```
=====
' Script07b
'
' WinTask Version 3.8
' © Copyright 1997-2011 TaskWare   Octobre 2011
'
' Exercice 7, partie 2
'
' Ce script introduit une condition et utilise l'instruction ExistW pour
' tester l'existence d'une fenêtre et ne cliquer dedans que si la fenêtre existe.
=====

' Cette variable de type chaîne de caractères est utilisée ensuite
' comme nom de fichier .txt dans la boîte de dialogue Enregistrer sous
nomfichier$="test07"

Shell("notepad",1)

UseWindow("NOTEPAD.EXE|Edit|Sans titre - Bloc-notes|1",1)
    SendKeys("Bonjour<Enter>")

UseWindow("NOTEPAD.EXE|Notepad|Sans titre - Bloc-notes",1)
    ChooseMenu(Normal,"&Edition|He&ure/Date  F5")

UseWindow("NOTEPAD.EXE|Edit|Sans titre - Bloc-notes|1",1)
    SendKeys("<Enter>")

CloseWindow("NOTEPAD.EXE|Notepad|Sans titre - Bloc-notes",1)

UseWindow("NOTEPAD.EXE|CtrlNotifySink|Bloc-notes|7",1)
    Click(Button,"&Enregistrer")

' La variable nomfichier$ remplace "test07"
UseWindow("NOTEPAD.EXE|FloatNotifySink|Enregistrer sous|1",1)
    WriteCombo("1",nomfichier$)

UseWindow("NOTEPAD.EXE|#32770|Enregister sous",1)
    Click(Button,"&Enregistrer")

' Test de l'existence de la fenêtre Confirmer l'enregistrement
' qui demande si vous voulez remplacer le fichier existant
If ExistW("NOTEPAD.EXE|CtrlNotifySink|Confirmer l'enregistrement|7")=1 Then
    UseWindow("NOTEPAD.EXE|CtrlNotifySink|Confirmer l'enregistrement|7",1)
    Click(Button,"&Oui")
Endif
```

Exercice 7, Script07c.src

```
=====
' Script07c
'
' Manuel WinTask Version 3.8
' © Copyright 1997-2011 TaskWare   Octobre 2011
'
' Exercice 7, partie 3
'
' Ce script introduit une condition sur le code retour d'une instruction UseWindow,
' quand la gestion des erreurs est effectuée à l'intérieur du script lorsque #IgnoreErrors=1.
=====
' Cette variable de type chaîne de caractères est utilisée ensuite
' comme nom de fichier .txt dans la boîte de dialogue Enregistrer sous
nomfichier$="test07"

Shell("notepad",1)

UseWindow("NOTEPAD.EXE|Edit|Sans titre - Bloc-notes|1",1)
    SendKeys("Bonjour<Enter>")

UseWindow("NOTEPAD.EXE|Notepad|Sans titre - Bloc-notes",1)
    ChooseMenu(Normal,"&Edition|He&ure/Date   F5")

UseWindow("NOTEPAD.EXE|Edit|Sans titre - Bloc-notes|1",1)
    SendKeys("<Enter>")

CloseWindow("NOTEPAD.EXE|Notepad|Sans titre - Bloc-notes",1)

UseWindow("NOTEPAD.EXE|CtrlNotifySink|Bloc-notes|7",1)
    Click(Button,"&Enregistrer")

' La variable nomfichier$ remplace "test07"
UseWindow("NOTEPAD.EXE|FloatNotifySink|Enregistrer sous|1",1)
    WriteCombo("1",nomfichier$)

UseWindow("NOTEPAD.EXE|#32770|Enregister sous",1)
    Click(Button,"&Enregistrer")

' #IgnoreErrors est mis à 1 de sorte que si la fenêtre spécifiée dans le UseWindow
' n'est pas trouvée, le script continue quand même son exécution.
' Ensuite le code retour de UseWindow est testé pour ne cliquer sur le bouton Oui
' que si la fenêtre a été trouvée
#IgnoreErrors = 1
ret = UseWindow("NOTEPAD.EXE|CtrlNotifySink|Confirmer l'enregistrement|7",1)
If ret = 0 Then
    Click(Button,"&Oui")
Endif
```

Exercice 7, Script07d.src

```
'=====
' Script07d
'
' Manuel WinTask Version 3.8
' © Copyright 1997-2011 TaskWare   Octobre 2011
'
' Exercice 7, partie 4
'
' Ce script introduit une condition sur le code retour d'une instruction UseWindow,
' quand la gestion des erreurs est effectuée à l'intérieur du script lorsque #IgnoreErrors=1.
' Le contenu de la variable où est stocké le nom de fichier sous lequel enregistré est modifié
' afin de tester la condition écrite dans script07c.
'=====

' Cette variable de type chaîne de caractères est utilisée ensuite
' comme nom de fichier .txt dans la boîte de dialogue Enregistrer sous
nomfichier$="test07d"

Shell("notepad",1)

UseWindow("NOTEPAD.EXE|Edit|Sans titre - Bloc-notes|1",1)
    SendKeys("Bonjour<Enter>")

UseWindow("NOTEPAD.EXE|Notepad|Sans titre - Bloc-notes",1)
    ChooseMenu(Normal,"&Edition|He&ure/Date   F5")

UseWindow("NOTEPAD.EXE|Edit|Sans titre - Bloc-notes|1",1)
    SendKeys("<Enter>")

CloseWindow("NOTEPAD.EXE|Notepad|Sans titre - Bloc-notes",1)

UseWindow("NOTEPAD.EXE|CtrlNotifySink|Bloc-notes|7",1)
    Click(Button,"&Enregistrer")

UseWindow("NOTEPAD.EXE|FloatNotifySink|Enregistrer sous|1",1)
    WriteCombo("1",nomfichier$)

UseWindow("NOTEPAD.EXE|#32770|Enregister sous",1)
    Click(Button,"&Enregistrer")

' #IgnoreErrors est mis à 1 de sorte que si la fenêtre spécifiée dans le UseWindow
' n'est pas trouvée, le script continue quand même son exécution.
' Ensuite le code retour de UseWindow est testé pour ne cliquer sur le bouton Oui
' que si la fenêtre a été trouvée
#IgnoreErrors = 1
ret = UseWindow("NOTEPAD.EXE|CtrlNotifySink|Confirmer l'enregistrement|7",1)
If ret = 0 Then
    Click(Button,"&Oui")
Endif
```

Exercice 8, Script08a.src

```
=====
' Script08a
'
' Manuel WinTask Version 3.8
' © Copyright 1997-2011 TaskWare   Octobre 2011
'
' Exercice 8, partie 1
'
' Ce script répète une saisie 10 fois dans notepad.
=====
```

```
Shell("notepad",1)
```

```
Texte$ = "Ligne de texte dans notepad"
```

```
NumeroLigne = 1
```

```
UseWindow("NOTEPAD.EXE|Edit|Sans titre - Bloc-notes|1",1)
```

```
  ' Ce bloc s'exécute 10 fois
```

```
  While numeroligne <= 10
```

```
    SendKeys(texte$ + " " + Str$(numeroligne) + "<Enter>")
```

```
    NumeroLigne = NumeroLigne + 1
```

```
  Wend
```

Exercice 8, Script08b.src

```
=====
' Script08b
'
' Manuel WinTask Version 3.8
' © Copyright 1997-2011 TaskWare   Octobre 2011
'
' Exercice 8, partie 2
'
' Ce script répète une saisie 10 fois dans notepad, cette fois en utilisant repeat/until.
=====

Shell("notepad",1)

Texte$ = "Ligne de texte dans notepad"
NumeroLigne = 1

UseWindow("NOTEPAD.EXE|Edit|Sans titre - Bloc-notes|1",1)
' Ce bloc s'exécute 10 fois
Repeat
    SendKeys(texte$ + " " + Str$(numeroligne) + "<Enter>")
    NumeroLigne = NumeroLigne + 1
Until numeroligne = 11
```

Exercice 9, Script09.src

```
=====
' Script09
'
' Manuel WinTask Version 3.8
' © Copyright 1997-2011 TaskWare   Octobre 2011
'
' Exercice 9
'
' Ce script lit le contenu d'un fichier texte et saisit chaque enregistrement
' dans WordPad.
=====

'Sous un Windows 64 bits
Shell(Chr$(34)+"C:\Program Files (x86)\Windows NT\Accessories\wordpad.exe"+Chr$(34),1)
'Sous un Windows 32 bits
Shell("wordpad",1)

' Si le fichier test09.txt n'a pas été sauvegardé dans le repertoire courant des scripts, insérez
' le chemin complet, par exemple nomfichier$="c:\docs\test\test09.txt"
NomFichier$ = "test09.txt"

UseWindow("WORDPAD.EXE|RICHEDIT50W|Document - WordPad|1",1)
' Itération jusqu'à ce que la fin de fichier soit atteinte
While Eof(NomFichier$) = 0
    ' Lecture d'un enregistrement, les données lues sont stockées dans la variable buffer$
    ret = Read(NomFichier$, Buffer$, CRLF)
    ' Saisie de la ligne lue dans WordPad
    SendKeys(Buffer$ + "<Enter>")
Wend
SendKeys("Bonjour Wordpad<Enter>")
```

Exercice 10, Script10.src

```
=====
' Script10
'
' Manuel WinTask Version 3.8
' © Copyright 1997-2011 TaskWare   Octobre 2011
'
' Exercice 10
'
' Ce script met à jour un fichier Excel à partir de données lues dans un fichier texte.
=====

' Déclaration d'un tableau pouvant contenir 26 chaînes (de 0 à 25)
Dim data$(25)

' Les noms de fichiers n'ont pas de chemin complet, ils sont donc supposés être dans le même
'répertoire que celui où est le fichier script, .SRC
fichier$="test09.txt"
fichierexcel$ = "exercice10.xls"

' Boucle pour lire le fichier test09.txt et stocker chaque enregistrement dans le i ème
'élément du tableau data$
i=0
repeat
Read(fichier$,data$(i), CRLF)
i=i+1
until eof(fichier$)

msgbox("nombre d'éléments dans le tableau : "+str$(i))

' Ecriture du tableau dans les cellules A1 à J1
WriteExcel(fichierexcel$, "A1:J1", data$())

' Ecriture du tableau dans les cellules C3 à C12
WriteExcel(fichierexcel$, "C3:C12", data$())
```

Exercice 11, Script11.src

```
=====
' Script11
'
' Manuel WinTask Version 3.8
' © Copyright 1997-2011 TaskWare   Octobre 2011
'
' Exercice 11
'
' Ce script ré-écrit le script de l'exercice 8 en sous-programme.
=====

=====
' Paramètres de la Sub Saisie_notepad :
'   nomfichier$   - Nom de fichier sous lequel enregistrer le fichier texte généré
'   Texte$       - Texte à écrire dans notepad
'   Compteur      - Nombre de fois à écrire Texte$
=====

Sub Saisie_Notepad(Nomfichier$, Texte$, Compteur)

    Shell("notepad",1)

    UseWindow("NOTEPAD.EXE|Edit|Sans titre - Bloc-notes|1",1)
    i=0

    while i<compteur

        SendKeys(texte$+" "+str$(i)+"<Enter>")

        i=i+1

    wend

CloseWindow("NOTEPAD.EXE|Notepad|Sans titre - Bloc-notes",1)

UseWindow("NOTEPAD.EXE|CtrlNotifySink|Bloc-notes|7",1)

    Click(Button,"&Enregistrer")

UseWindow("NOTEPAD.EXE|FloatNotifySink|Enregistrer sous|1",1)

    WriteCombo("1",nomfichier$)

UseWindow("NOTEPAD.EXE|#32770|Enregistrer sous",1)

    Click(Button,"&Enregistrer")

If ExistW("NOTEPAD.EXE|CtrlNotifySink|Confirmer l'enregistrement|7")=1 then

    UseWindow("NOTEPAD.EXE|CtrlNotifySink|Confirmer l'enregistrement|7",1)

    Click(Button,"&Oui")

EndIf

EndSub

=====
' Programme principal
=====
```

NotepadTexte\$ = "Exercice Sub"

' Appel de la Sub pour écrire 12 lignes dans notepad
Saisie_Notepad("test11.txt",NotepadTexte\$, 12)

Exercice 12, Script12.src

```
=====
' Script12
'
' Manuel WinTask Version 3.8
' © Copyright 1997-2011 TaskWare   Octobre 2011
'
' Exercice 12
'
' Ce script ré-écrit le script des exercices 9 et 11
' en Sub et Function.
=====

'
' Cette Sub écrit le texte spécifié n fois dans notepad.
' n est spécifié par le paramètre Compteur.
' Le fichier généré est enregistré sous le nom spécifié par nomfichier$
'
' Paramètres de la Sub Saisie_notepad :
'   nomfichier$   - Nom de fichier sous lequel enregistrer le fichier texte généré
'   Texte$       - Texte à écrire dans notepad
'   Compteur     - Nombre de fois à écrire Texte$
=====
Sub Saisie_Notepad(NomFichier$, Texte$, Compteur)

    Shell("notepad",1)

    UseWindow("NOTEPAD.EXE|Edit|Sans titre – Bloc-notes|1",1)
        i=0
        while i<compteur
            SendKeys(texte$+" "+str$(i)+"<Enter>")
            i = i + 1
        Wend

    CloseWindow("NOTEPAD.EXE|Notepad|Sans titre - Bloc-notes",1)

    UseWindow("NOTEPAD.EXE|CtrlNotifySink|Bloc-notes|7",1)

    Click(Button,"&Enregistrer")

    UseWindow("NOTEPAD.EXE|FloatNotifySink|Enregistrer sous|1",1)

    WriteCombo("1",nomfichier$)

    UseWindow("NOTEPAD.EXE|#32770|Enregistrer sous",1)

    Click(Button,"&Enregistrer")

    If ExistW("NOTEPAD.EXE|CtrlNotifySink|Confirmer l'enregistrement|7")=1 then
        UseWindow("NOTEPAD.EXE|CtrlNotifySink|Confirmer l'enregistrement|7",1)

        Click(Button,"&Oui")
    Endif

EndSub

EndSub
```

```

=====
' Cette fonction lit le texte à partir du fichier spécifié
' et écrit le texte dans Wordpad.
'
' NOTE: WordPad n'est pas fermé et le contenu n'est pas enregistré.
'
' Paramètres de la fonction saisie_wordpad :
'   nomfichier$   - Nom du fichier à lire
'
' Renvoi :
'   Nombre de lignes écrites dans :
=====
Function Saisie_Wordpad(NomFichier$)

    'Lancement de wordpad sous un Windows 32 bits
    Shell("wordpad",1)
    'Lancement de wordpad sous un Windows 64 bits
    Shell(Chr$(34)+"C:\Program Files (x86)\Windows NT\Accessories\wordpad.exe"+Chr$(34),1)

    LignesEcrites = 0

    UseWindow("WORDPAD.EXE|RichEdit50W|Document - WordPad|1",1)
        repeat
            ret = Read(NomFichier$, Buffer$, CRLF)
            SendKeys(Buffer$ + "<Enter>")
            LignesEcrit = LignesEcrites + 1
        Until Eof(NomFichier$)
        SendKeys("Bonjour Wordpad<Enter>")
        LinesEcrites = LignesEcrites + 1

    Saisie_Wordpad = LignesEcrites

EndFunction

=====
' Programme principal
=====

' Le fichier text12.txt est supposé être dans le même répertoire que le script. Si ce n'est pas le cas,
' incluez le chemin complet dans le nom.
fichiernotepad$ = "text12.txt"
textenotepad$ = "Exercice Sub et Fonction"

' Appel à la Sub pour écrire 8 lignes dans notepad et enregistrer sous le nom "text12.txt"
Saisie_Notepad(fichiernotepad$, Textenotepad$, 8)

' Appel à la fonction pour écrire le contenu de "text12.txt" dans wordpad
' et compter le nombre de lignes écrites
Resultat = Saisie_Wordpad(fichiernotepad$)

MsgBox("Nombre de lignes écrites dans WordPad = " + Str$(Resultat))

```

Exercice 13, Script13.src

```
=====
' Script13
'
' Manuel WinTask Version 3.8
' © Copyright 1997-2011 TaskWare   Octobre 2011
'
' Exercice 13
'
' Cet exercice utilise les bouts de script réalisés dans les
' exercices 10 et 12 et ajoute l'utilisation d'un tableau
=====

' Déclaration d'un tableau pouvant contenir jusqu'à 26 chaînes de caractères (de 0 à 25)
Dim data$(25)

=====
' Cette Sub écrit le texte spécifié n fois dans notepad.
' n est spécifié par le paramètre Compteur.
' Le fichier généré est enregistré sous le nom spécifié par nomfichier$.
'
' Paramètres de la Sub Saisie_notepad :
'   nomfichier$   - Nom de fichier sous lequel enregistrer le fichier texte généré
'   Texte$       - Texte à écrire dans notepad
'   Compteur     - Nombre de fois à écrire Texte$
=====
Sub Saisie_Notepad(NomFichier$, Texte$, Compteur)

    Shell("notepad",1)

    i = 0

    UseWindow("NOTEPAD.EXE|Edit|Sans titre - Bloc-notes|1",1)
        While i < Compteur
            SendKeys(texte$+" "+str$(i)+"<Enter>")
            i = i + 1
        Wend

    CloseWindow("NOTEPAD.EXE|Notepad|Sans titre - Bloc-notes",1)

    UseWindow("NOTEPAD.EXE|CtrlNotifySink|Bloc-notes|7",1)

    Click(Button,"&Enregistrer")

    UseWindow("NOTEPAD.EXE|FloatNotifySink|Enregistrer sous|1",1)

    WriteCombo("1",nomfichier$)

    UseWindow("NOTEPAD.EXE|#32770|Enregistrer sous",1)

    Click(Button,"&Enregistrer")

    If ExistW("NOTEPAD.EXE|CtrlNotifySink|Confirmer l'enregistrement|7")=1 then
        UseWindow("NOTEPAD.EXE|CtrlNotifySink|Confirmer l'enregistrement|7",1)
        Click(Button,"&Oui")
    Endif

Endif
```

EndSub

```
=====
' Cette fonction lit le texte à partir du fichier spécifié
' et écrit le texte dans Wordpad.
'
' NOTE: WordPad n'est pas fermé et le contenu n'est pas enregistré.
'
' Paramètres de la fonction saisie_wordpad ::
'   nomfichier$   - Nom du fichier à lire
'
' Renvoi :
'   Nombre de lignes écrites dans wordpad
=====
Function Saisie_Wordpad(NomFichier$)
```

```
    Shell("wordpad",1)

    LignesEcrites = 1

    UseWindow("WORDPAD.EXE|RICHEDIT50W|Document - WordPad|1",1)
        repeat
            Read(NomFichier$, Buffer$, CRLF)
            SendKeys(Buffer$ + "<Enter>")
            LignesEcrites = LignesEcrites + 1
        Until Eof(NomFichier$)
        LignesEcrites = LignesEcrites + 1

    Saisie_Wordpad = LignesEcrites
```

EndFunction

```
=====
' Cette Sub lit le texte contenu dans le fichier spécifié
' et l'écrit dans un tableau Excel.
' Le contenu du fichier est écrit d'abord sur une même ligne en commençant
' en cellule A1 puis sur une même colonne à partir de C3.
'
' Paramètres de la Sub :
'   NomFichier$   - Nom du fichier texte à lire
'   NomFichierExcel$ - Nom du fichier Excel à metre à jour
=====
Sub Saisie_Excel(NomFichier$,NomFichierExcel$)
```

```
    ' Remettre le pointeur de lecture en début de fichier.
    ' Car il est actuellement en fin suite à tous les Read() de Saisie_Wordpad
    SetReadPos(NomFichier$,0)

    ' Remplir un tableau avec chacune des lignes lues
    i = 0
    While (Eof(NomFichier$) = 0) AND (i <= 25)
        Read(nomfichier$,data$(i), CRLF)
        i = i + 1
    Wend

    ' L'index issu de la boucle ci-dessus indique combien d'éléments ont été lus.
```

' i-1 contient le nombre d'éléments lus.

' L'écriture doit se faire en colonne C à partir de la ligne 3 et il faut écrire i-1 éléments

PremCelluleLigne\$ = "3"

CelluleLigne = Val(PremCelluleLigne\$) + i - 1

DerniereCelluleLigne\$ = Str\$(CelluleLigne)

Rangee_colonne\$ = "C" + PremCelluleLigne\$ + ":C" + DerniereCelluleLigne\$

' Plus complexe : calcul de la lettre de la colonne où écrire le dernier élément.

' La première colonne est A.

PremCelluleCol\$ = "A"

CelluleCol = Asc(PremCelluleCol\$) + i - 1

DerniereCelluleCol\$ = Chr\$(CelluleCol)

Rangee_ligne\$ = PremCelluleCol\$ + "1:" + DerniereCelluleCol\$ + "1"

WriteExcel(nomfichierexcel\$,rangee_ligne\$,data\$())

WriteExcel(nomfichierexcel\$,Rangee_colonne\$,data\$())

EndSub

=====
' Programme principal
=====

' Vérifiez ici que les chemins indiqués sont ceux où se trouvent vos fichiers.

fichiernotepad\$ = "c:\text13.txt"

textenotepad\$ = "Exercice Sub, Function et Excel"

fichierexcel\$="c:\program files\wintask\scripts\exercice13.xls"

' Appel à la Sub pour écrire 5 lignes dans notepad et enregistrer sous le nom "text13.txt"

Saisie_Notepad(fichiernotepad\$, Textenotepad\$, 5)

' Appel à la fonction pour écrire le contenu de "text13.txt" dans wordpad

' et compter le nombre de lignes écrites.

Resultat = Saisie_Wordpad(fichiernotepad\$)

' Appel à la Sub pour écrire dans Excel

Saisie_Excel(fichiernotepad\$,fichierexcel\$)

MsgBox("Nombre de lignes écrites dans WordPad = " + Str\$(Resultat))

Exercice 14, Script14.src

```
'=====
' Script14
'
' Manuel WinTask Version 3.8
' © Copyright 1997-2011 TaskWare   Octobre 2011
'
' Ce script illustre la différence entre msgbox et msgframe pour faire afficher une variable.
'=====
```

```
Shell("notepad",1)

i = 0
j = 10

UseWindow("NOTEPAD.EXE|Edit|Sans titre - Bloc-notes|1",1)
    SendKeys("Bonjour<Enter>")

Repeat
    SendKeys("Bonjour"+str$(i)+"<Enter>")
    SendKeys("Bonjour"+str$(j)+"<Enter>")
    MsgBox("l'index j vaut : "+str$(j))
    i=i+1
    j=j-1
until i = 8

CloseWindow("NOTEPAD.EXE|Notepad|Sans titre - Bloc-notes")

UseWindow("NOTEPAD.EXE|CtrlNotifySink|Bloc-notes|8",1)
    Click(Button,"Ne pas en&registrer")
```

```
'=====
' Le même script mais avec msgframe.
'=====
```

```
Shell("notepad",1)

i = 0
j = 10

UseWindow("NOTEPAD.EXE|Edit|Sans titre - Bloc-notes|1",1)
    SendKeys("Bonjour<Enter>")

Repeat
    SendKeys("Bonjour"+str$(i)+"<Enter>")
    SendKeys("Bonjour"+str$(j)+"<Enter>")
    MsgBox("l'index j vaut : "+str$(j),1)
    i=i+1
    j=j-1
until i = 8

CloseWindow("NOTEPAD.EXE|Notepad|Sans titre - Bloc-notes")

UseWindow("NOTEPAD.EXE|CtrlNotifySink|Bloc-notes|8",1)
    Click(Button,"Ne pas en&registrer")
```

Exercice 15, Script15.src

```
=====
' Script15
'
' Manuel WinTask Version 3.8
' © Copyright 1997-2011 TaskWare   Octobre 2011
'
' Ce script illustre la génération d'un fichier rapport de type limité où
' seules les lignes Comment sont écrites dans le fichier rapport lors de l'exécution.
=====
```

```
Shell("notepad",1)
```

```
i = 0
j = 10
```

```
UseWindow("NOTEPAD.EXE|Edit|Sans titre - Bloc-notes|1",1)
    SendKeys("Bonjour<Enter>")
```

```
Repeat
```

```
    SendKeys("Bonjour"+str$(i)+"<Enter>")
    SendKeys("Bonjour"+str$(j)+"<Enter>")
    Comment("l'index i vaut : "+str$(i))
    Comment("l'index j vaut : "+str$(j))
    i=i+1
    j=j-1
```

```
until i = 8
```

```
CloseWindow("NOTEPAD.EXE|Notepad|Sans titre - Bloc-notes")
```

```
UseWindow("NOTEPAD.EXE|CtrlNotifySink|Bloc-notes|8",1)
    Click(Button,"Ne pas en&registrer")
```

```
=====
' Script15b
'
' Manuel WinTask Version 3.8
' © Copyright 1997-2011 TaskWare   Octobre 2011
'
' Ce script génère un fichier log en appelant une Sub définie par le programmeur.
=====
```

```
sub log(msg_log$)
local buffer$
buffer$=Date$()+", "+hour$()+":"+min$()+":"+sec$()+ " --> "+msg_log$
write(fichier_log$,buffer$,CRLF)
endsub
```

```
fichier_log$="C:\script15blog.txt"
```

```
Shell("notepad",1)
```

```
i = 0
j = 10
```

```
UseWindow("NOTEPAD.EXE|Edit|Sans titre - Bloc-notes|1",1)
    SendKeys("Bonjour<Enter>")
```

```
Repeat
    SendKeys("Bonjour"+str$(i)+"<Enter>")
    SendKeys("Bonjour"+str$(j)+"<Enter>")
    log("!index i vaut : "+str$(i))
    log("!index j vaut : "+str$(j))
    i=i+1
    j=j-1
until i = 8

CloseWindow("NOTEPAD.EXE|Notepad|Sans titre - Bloc-notes")

UseWindow("NOTEPAD.EXE|CtrlNotifySink|Bloc-notes|8",1)
    Click(Button,"Ne pas en&registrer")
```

GLOSSAIRE

Compilateur : Programme WinTask qui lit un fichier script source et le transforme en fichier exécutable. Les scripts WinTask peuvent être compilés en sélectionnant le menu **Démarrer/Compilation seule** (ou touche Ctrl+F7), en sélectionnant le menu **Démarrer/Exécution** ou en cliquant sur l'icône **Exéc** de la barre d'outils. Dans les deux derniers cas, si aucune erreur de compilation n'est détectée, l'exécution est lancée une fois la compilation terminée. Notez que le lancement de la compilation enregistre d'abord le fichier script source.

Boîte de dialogue : Fenêtre affichée par un programme où l'utilisateur saisit des informations utilisées dans la suite du programme. Les boîtes de dialogue peuvent également afficher des données.

Editeur : Programme WinTask permettant la réalisation des scripts d'automatisation. De la fenêtre de l'Editeur sont accessibles toutes les fonctions nécessaires à l'écriture de scripts : mode Enregistrement, Compilation, Exécution, Assistants de synchronisation, Assistant de capture, Insertion d'instructions.

TaskExec : Programme WinTask qui exécute toutes les instructions auparavant compilées dans un fichier d'extension .ROB. Un double-clic sur un .ROB lance l'exécution de ce .ROB.

Log ou fichier rapport : Fichier contenant les instructions exécutées lors du rejoue. Un fichier rapport sert à comprendre pourquoi le rejoue n'a pas donné les résultats escomptés.

Macro : Voir Script.

Mode Enregistrement : Fonctionnalité de l'Editeur WinTask permettant de transformer les actions effectuées par l'utilisateur en instructions WinTask et ainsi créer le script d'automatisation.

Runtime : Logiciel WinTask permettant seulement d'exécuter des scripts compilés réalisés avec le logiciel WinTask complet, appelé WinTask-pack de développement. Le runtime WinTask est un sous-ensemble du pack de développement, il ne contient pas l'Editeur, le Compilateur ni le Planificateur de tâches.

.ROB : Fichier issu de la compilation d'un script source. TaskExec lit les instructions contenues dans un .ROB pour exécuter les différentes lignes et ainsi rejouer les actions enregistrées.

Script : Ensemble d'instructions du langage WinTask organisées en fichier de type texte et décrivant les différentes actions à automatiser. Les fichiers Script de WinTask portent l'extension .SRC.

Espion : Outil WinTask permettant de trouver le nom Windows d'un objet (d'une fenêtre, d'un bouton, ...).

.SRC : Fichiers de type texte contenant les instructions du langage WinTask décrivant les différentes actions à automatiser. L'Editeur WinTask permet de créer et de modifier les fichiers .SRC. Le Compilateur WinTask lit les fichiers .SRC et les transforme en fichiers .ROB, fichiers exécutable par le programme TaskExec de WinTask.

Nom de fenêtre: Spécifie de manière unique le nom d'une fenêtre présente sur le bureau Windows. Ce nom de fenêtre est généré automatiquement par le mode Enregistrement ou peut être trouvé en utilisant l'outil Espion. Au rejoue, TaskExec utilise le nom de fenêtre pour envoyer les actions suivantes à la fenêtre spécifiée.

Barre d'outils WinTask flottante : Barre d'outils affichée pendant le mode Enregistrement. Ses icônes permettent d'accéder aux principaux outils de WinTask sans sortir du mode Enregistrement (l'Espion, les assistants de synchronisation, l'assistant de capture et le bouton Stop).

Index

#ActionTimeout	34	Focus\$()	36
#IgnoreErrors.....	34	Function/ExitFunction/EndFunction.....	50
#UseExact.....	35	Include	55
Arrêter l'exécution d'un script.....	14	Instructions du langage	29
Assistants de Synchronisation.....	6	Itération (Boucles).....	39
Attente action clavier	16	Kill()	41
Attente action menu	17	Msgbox	55
Attente action souris	17	MsgFrame	55
Chaînes de caractères	31	Nombres réels.....	31
ChooseItem	65	Opérateurs	32
ChooseMenu	65	Read()	42
ClickOnBitmap	66	ReadExcel	44
ClickOnText	66	Rejoue trop lent.....	68
Compilateur		Repeat/Until	39
TaskComp.....	5	Sub/ExitSub/EndSub.....	47
Editeur		Synchronisation Bitmap.....	16
TaskEdit.....	5	Synchronisation Date/Heure	16
Editeur de Boîtes de Dialogue	6	Synchronisation Durée.....	16
Enregistrement Bas Niveau	66	Synchronisation Fenêtre.....	16
Entiers	31	Synchronisation OCR Texte	16
Eof()	42	Synchronisation Texte	15
Erreur objet not trouvé.....	63	Tableaux	32
Erreurs de compilation.....	55	Top\$()	36
Erreurs d'exécution	55	Variables	30
Espion	6	Variables directives.....	31
Exécution		While/Wend	39
TaskExec	6	Write()	43
Exécution en mode Débogage.....	58	WriteCombo.....	64
Exist()	41	WriteEdit.....	64
ExistW()	36	WriteExcel	44
Fichier rapport	56		