eBook Gratuit

APPRENEZ progress-4gl

eBook gratuit non affilié créé à partir des **contributeurs de Stack Overflow.**

#progress-4gl

Table des matières

À propos 1

Chapitre 1: Démarrer avec progress-4gl 2

Remarques 2

Versions 2

Examples 3

Installation ou configuration 3

Bonjour le monde! 10

FizzBuzz 11

Mise en place de l'environnement 11

Création de la base de données de démonstration "sports2000" à partir de la ligne de comma 12

Code de commentaire 14

Fichiers de programme 14

Faire du sport2000 en tant que service 15

Chapitre 2: Compiler 18

Introduction 18

Syntaxe 18

Examples 18

Compilateur d'application 18

Déclaration COMPILE 22

Poignée du système COMPILER 23

Chapitre 3: Cordes 28

Introduction 28

Remarques 28

Examples 28

Définir, associer et afficher une chaîne 28

Chaînes concaténantes 28

Manipulation de cordes 28

Cordes SENSIBLES 33

COMMENCE et MATCHS 33

Conversion des majuscules et minuscules 34

Des listes ... 35
Caractères spéciaux (et échapper) ... 36
Chapitre 4: Expressions conditionnelles ... 38
Introduction ... 38
Examples ... 38
IF ... THEN ... ELSE-statement ... 38
CAS ... 39
IF ... THEN ... ELSE-function ... 40
Chapitre 5: Itérer ... 42
Introduction ... 42
Examples ... 42
FAIRE PENDANT ... 42
DO var = commencer pour finir [BY step] ... 42
RÉPÉTER ... 44
Chapitre 6: Les fonctions ... 46
Introduction ... 46
Remarques ... 46
Examples ... 46
Fonction simple ... 46
Fonctions de déclaration anticipée ... 46
Paramètres d'entrée multiples ... 47
Instructions de retour multiples (mais une seule valeur de retour) ... 47
Paramètres de sortie et d'entrée-sortie ... 48
Récursivité ... 49
Appel dynamique d'une fonction ... 49
Chapitre 7: Les variables ... 53
Introduction ... 53
Syntaxe ... 53
Examples ... 53
Déclarations de variables de base ... 53
Tableaux - définition et accès ... 54
Utilisation du mot clé LIKE ... 56

Chapitre 8: Procédures 57
Introduction 57
Syntaxe 57
Examples 57
Une procédure interne de base 57
Paramètres INPUT et OUTPUT 57
Récursivité - voir récursivité 58
Portée 59
Chapitre 9: Requêtes 61
Introduction 61
Syntaxe 61
Examples 61
Requête de base 61
Requête multi-tables 62
Déplacement d'une empreinte avec une requête utilisant next, first, prev et last 63
Chapitre 10: TEMP-TABLE 65
Introduction 65
Examples 65
Définir une table temporaire simple 65
Une table temporaire avec un index 65
Plus d'index - index 65
Saisie et sortie de tables temporaires 67
Chapitre 11: Travailler avec des nombres 71
Introduction 71
Examples 71
Les opérateurs 71
Plus de fonctions mathématiques 71
Comparaison des nombres 73
Générateur de nombres aléatoires 73
Chapitre 12: TROUVER une déclaration 74
Introduction 74
Examples 74

Trouver des exemples de base....74
Disponibilité et étendue....74
TROUVER et verrouiller....76
Chapitre 13: Utilitaires OS....78
Introduction....78
Examples....78
OS-COMMAND....78
OPSYS....79
OS-ERROR....79
Fonction OS-GETENV....80
OS-COPY....81
OS-DELETE....81
OS-CREATE-DIR....82
OS-APPEND....82
OS-RENAME....82
OS-DRIVES (Windows uniquement)....82
Crédits....84

À propos

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: progress-4gl

It is an unofficial and free progress-4gl ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official progress-4gl.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Chapitre 1: Démarrer avec progress-4gl

Remarques

ABL (Advanced Business Language). Plus tôt connu sous le nom de Progress 4GL.

Progress ABL est un langage de programmation lié à l'environnement Progress OpenEdge, à sa base de données et aux utilitaires environnants. Cela en fait un langage de programmation " quatrième génération ".

Progress ABL est un langage de programmation de type anglais, fortement typé et tardif, qui prend de plus en plus en charge l'orientation des objets. Le code compilé est exécuté par "AVM" (machine virtuelle ABL).

Le langage est développé et maintenu par Progress Corporation (anciennement Progress Software).

Versions

Version	Retraité	Remarque	Date de sortie
11.7	à déterminer		2017-04-04
11.6	à déterminer	Dernier: 11.6.3	2015-10-01
11,5	2017-déc		2014-12-01
11.4	2017-août		2014-08-01
11.3	2016-août		2013-07-01
11.2	2016-février		2013-02-01
11.1	2014-février		2012-06-01
11.0	2013-juin		2011-12-01
10.2B	à déterminer	Renommé OpenEdge	2009-12-01
10.1C	2014-juillet		2008-02-01
10.0B	2006-mars		2004-08-01
9.1E	2015-oct		2004-11-01
8.3E	2010-février		2001-12-01

Examples

Installation ou configuration

Installation du progrès

Téléchargez votre distribution depuis Progress. Si vous voulez une licence de démonstration, vous devez les contacter. Assurez-vous de télécharger un fichier tar 64 bits et non 32 bits (sauf si vous utilisez un ordinateur 32 bits).

les fenêtres

Le téléchargement sera une archive zip. Décompressez-le et lancez simplement setup.exe. L'installation sera graphique mais exactement identique à celle décrite ci-dessous.

Linux / Unix / HP-UX etc.

Placez le fichier tar sur votre système Progress. Disons que vous l'avez dans votre répertoire personnel:

```
/home/user/PROGRESSFILENAME.tar
```

Extraire le:

```
cd /home/user
tar xvf PROGRESSFILENAME.tar
```

Il va créer un répertoire nommé

```
proinst
```

Changez de répertoire vers une autre destination et créez un répertoire temporaire à cet endroit. Par exemple:

```
cd /tmp
mkdir proinst116
cd proinst116
```

Une fois l'installation terminée, ce répertoire contiendra des informations sur l'installation ainsi que les fichiers que vous pouvez enregistrer et utiliser pour les futures répétitions automatiques de la même installation.

Maintenant, lancez le script d'installation (nommé "proinst" dans le répertoire "proinst"):

```
/home/user/proinst/proinst
```

Cela va démarrer l'installation:

```
+--------------------------------------------------------------------+
|                             Welcome                                |
+--------------------------------------------------------------------+
|                                                                    |
|         WELCOME TO THE OPENEDGE INSTALLATION UTILITY               |
|                                                                    |
| Ensure that you have your completed "Preinstallation Checklist     |
| for Unix" handy to consult. This information will facilitate your  |
| installation and ensure your choices are accurately recorded.      |
|                                                                    |
|     Copyright (c) 1984-2015 Progress Software Corporation          |
|         and/or one of its subsidiaries or affiliates.              |
|                     All Rights Reserved.                           |
|                                                                    |
|                                                                    |
|                            [Enter=OK]                              |
+--------------------------------------------------------------------+
```

Vous devez maintenant insérer les clés de licence, le nom de la société, etc. Il est recommandé de télécharger un "fichier additif". Vous pouvez simplement appuyer sur `Ctrl + A` et l'utiliser.

```
+--------------------------------------------------------------------+
|                     Product Configuration Data                     |
+--------------------------------------------------------------------+
|                                                [Enter=Additional]  |
| Company Name: ____________________________     [Ctrl-E=Done]       |
| Serial Number: __________                      [CTRL-T=Quit]       |
| Control Number: ______  ______  ______         [CTRL-N=Release Notes]|
|                                                [CTRL-V=View]       |
|                                                [TAB=Next Field]    |
|                                                [CTRL-P=Help]       |
|                                                [CTRL-A=Addendum File]|
|                                                                    |
+--------------------------------------------------------------------+
```

Ajouter un fichier addenda:

```
  +-------------------------------------------------------------------------+
  |                           License Addendum File                         |
  +-------------------------------------------------------------------------+
  |                                                                         |
  | Enter Path: /home/myuser/myfile.txt_____________________________________ |
  |                                                                         |
  |                                                                         |
  |                                                                         |
  |      [Enter=OK]  [CTRL-N=Cancel]                                        |
  +-------------------------------------------------------------------------+
    |                                                 [TAB=Next Field]      |
    |                                                 [CTRL-P=Help]         |
    |                                                 [CTRL-A=Addendum File]|
    |                                                                       |
    +-----------------------------------------------------------------------+
```

Une fois les licences ajoutées manuellement ou chargées via un fichier, vous pouvez appuyer sur `Ctrl` + `V` pour afficher les produits à installer:

```
      +-----------------------+
```

```
      |Entered Product List    |
      +------------------------+
      | 4GL Development System |
      | OE Application Svr Ent |
+-----| OE Enterprise RDBMS    |--------------------------------------+
|     | OpenEdge Replication   |nfiguration Data                      |
+-----+------------------------+--------------------------------------+
|                                              [Enter=Additional]    |
| Company Name: _____________________________  [Ctrl-E=Done]         |
| Serial Number: __________                    [CTRL-T=Quit]         |
| Control Number: ______  ______  ______       [CTRL-N=Release Notes]|
|                                              [CTRL-V=View]         |
|                                              [TAB=Next Field]      |
|                                              [CTRL-P=Help]         |
|                                              [CTRL-A=Addendum File]|
|                                                                    |
+--------------------------------------------------------------------+
```

Une fois que vous êtes satisfait, appuyez sur `Ctrl + E` pour continuer l'installation ou sur `Ctrl` + `Q` pour quitter.

Si vous passez à autre chose, il vous faudra encore une chose:

```
   +--------------------------------------------------------------------+
   |                 Done Configuration Data Confirmation               |
   +--------------------------------------------------------------------+
   |                                                                    |
   |Are you sure that you are done entering all the control numbers for the|
   |OpenEdge products that will be installed?                           |
   |                                                                    |
   |                           [Y=YES] [N=NO]                           |
   +--------------------------------------------------------------------+
    |                                              [CTRL-P=Help]         |
    |                                              [CTRL-A=Addendum File]|
    |                                                                    |
    +--------------------------------------------------------------------+
```

Appuyez sur `Y` pour continuer ou sur `N` pour revenir en arrière.

Selon ce que vous installez, vous devrez peut-être configurer différents produits lors de l'installation.

La prochaine étape consiste à décider si vous souhaitez activer "OpenEdge Explorer". `Y` ou `N`. Cela peut être changé plus tard.

```
                  +--------------------------------+
                  | Install Type and Destination   |
                  +--------------------------------+
                  | Select Type of Installation    |
                  | Select Destination Pathname    |
                  | Select Management Pathname     |
                  | Continue with Installation     |
                  | View Release Notes             |
                  | Cancel                         |
                  | Quit Installation              |
                  | Help                           |
                  +--------------------------------+
```

```
+------------------------------------------------------------------------------+
|Type: Complete Install                                                        |
|Destination pathname: /usr/dlc                                                |
|Working Dir pathname: /usr/wrk                                                |
|Management pathname: /usr/oemgmt                                              |
|Management Working Dir pathname: /usr/wrk_oemgmt                              |
|                                                                              |
+------------------------------------------------------------------------------+
```

Vous devez maintenant décider des répertoires dans lesquels vous souhaitez installer Progress ainsi que du répertoire de travail principal (où vous souhaitez stocker votre code). Modifiez-les ou continuez avec les valeurs par défaut. Historiquement, `/usr/dlc` a toujours été la valeur par défaut. Vous voudrez peut-être la remplacer par un élément unique pour cette version spécifique de Progress - qui pourrait vous aider lors de la mise à niveau. Choisissez une `Complete Install` (valeur par défaut).

Une fois terminé: choisissez `Continue with Installation` utilisant les touches fléchées et appuyez sur `Entrée` pour continuer.

```
   +--------------------------------------------------+
   |                 Select Server Engine             |
   +--------------------------------------------------+
   |*SQL   -Provides SQL access to OpenEdge data files |
   | Continue with Install                            |
   | Cancel                                           |
   | Help                                             |
   +--------------------------------------------------+
```

Si vous ne prévoyez aucun accès SQL, vous pouvez appuyer sur Entrée une fois et supprimer le `*` avant SQL, sinon `Continue with Install` .

```
+------------------------------------------------------------------------------+
|                                ATTENTION                                     |
+------------------------------------------------------------------------------+
|                                                                              |
|The OpenEdge Adapter for Sonic ESB is a recommended component of this         |
|installation and requires a Sonic ESB installation somewhere on your         |
|network.                                                                      |
|                                                                              |
|Choose YES if you plan on using OpenEdge Adapter for Sonic ESB, else choose   |
|NO.                                                                           |
|                                                                              |
|                          [Y=YES] [N=NO] [H=Help]                             |
+------------------------------------------------------------------------------+
```

Très probablement, vous n'avez pas besoin de l'adaptateur OpenEdge pour Sonic ESB, alors appuyez sur `N` - sinon vous savez quoi faire.

```
 +--------------------------------------------------------------+
 |                            ATTENTION                         |
```

```
+-------------------------------------------------------------+
|                                                             |
|WebSpeed is a recommended component of this installation and |
|requires a Web Server installed somewhere on your network.   |
|                                                             |
|Choose YES if you plan on using WebSpeed and you are installing  |
|on the system where your Web Server is installed, else choose NO.|
|                                                             |
|                    [Y=YES] [N=NO] [H=Help]                  |
+-------------------------------------------------------------+
```

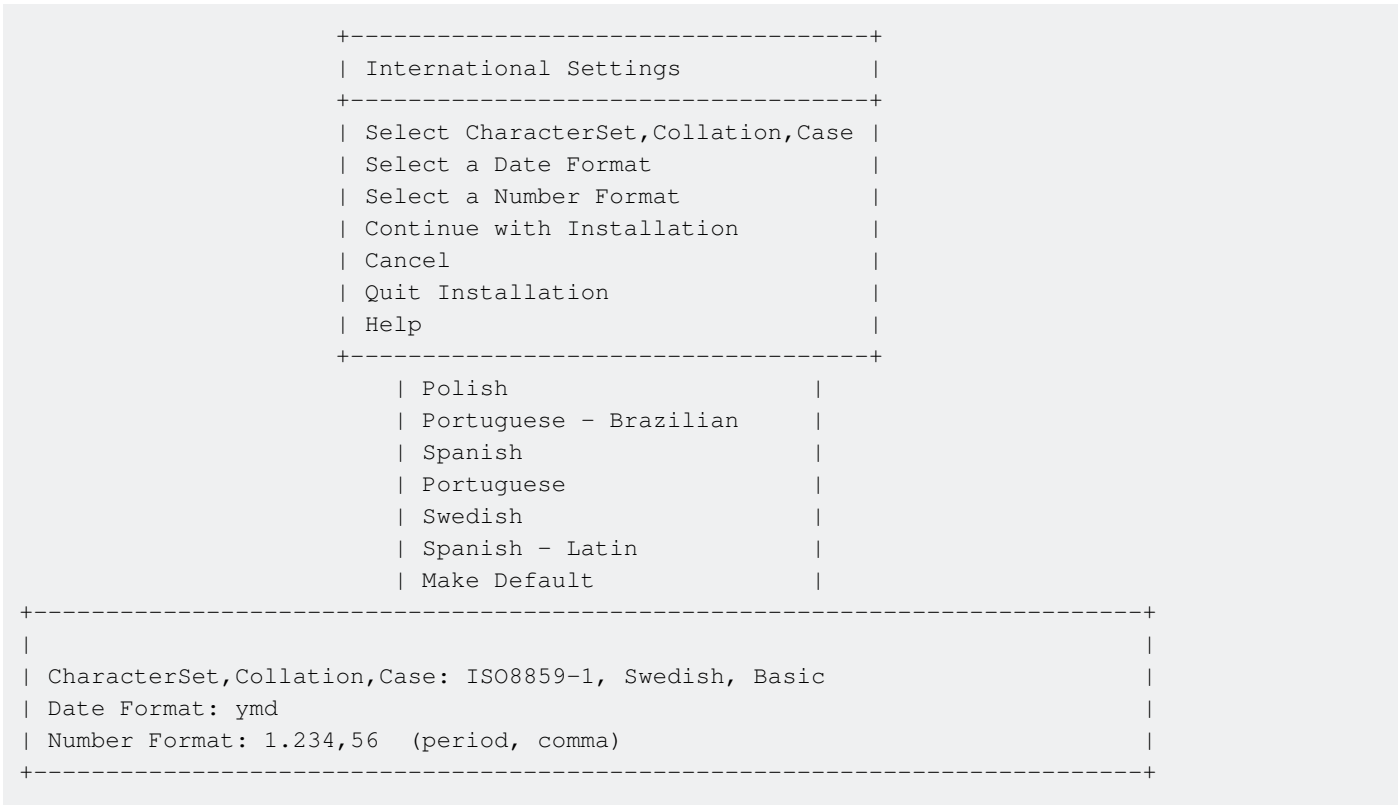
Si vous prévoyez d'utiliser WebSpeed pour produire du HTML dynamique, appuyez sur `Y` , sinon `N`.

```
                  +-----------------------------------+
                  | Web Server Type                   |
                  +-----------------------------------+
                  | Select Web Server Type            |
                  | Select Web Server Script directory |
                  | Copy the static HTML to docroot   |
                  | Continue with Installation        |
                  | Cancel                            |
                  | Quit Installation                 |
                  | Help                              |
                  +-----------------------------------+
```

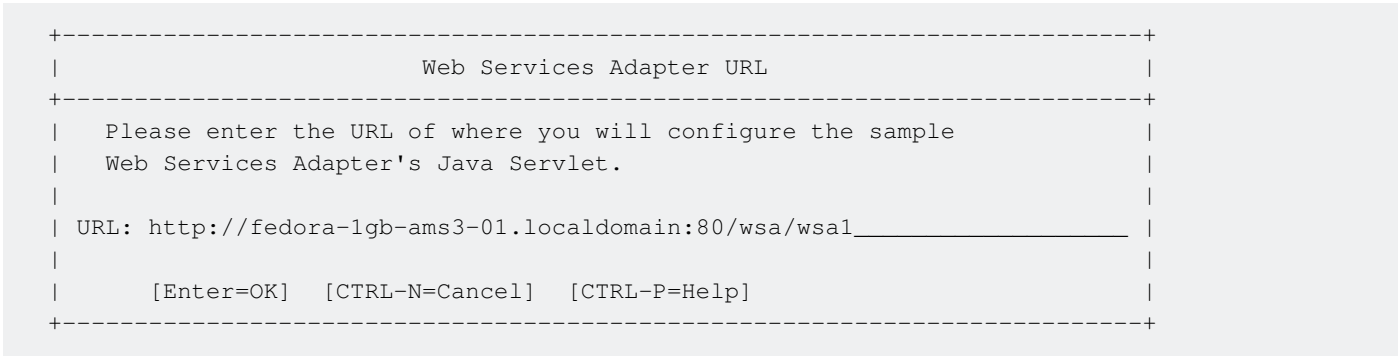
Configuration de WebSpeed: Choisissez `Select Web Server Type` et définissez-le sur `cgi` (probablement de toute façon). Le répertoire de script du serveur Web peut être défini sur le répertoire `cgi-bin` de votre serveur ou sur quelque chose comme `/tmp` . Ne copiez pas le code HTML statique - c'est vraiment obsolète. Continuer!

```
                    +----------------------------+
                    |Language Selection          |
                    +----------------------------+
                    | Chinese (Simplified)       |
                    | Czech                      |
                    | Dutch                      |
                    | English - American         |
                    | English - International    |
                    | French                     |
                    | German                     |
                    | Italian                    |
                    | Polish                     |
                    | Portuguese - Brazilian     |
                    | Spanish                    |
                    | Portuguese                 |
                    | Swedish                    |
                    | Spanish - Latin            |
                    | Make Default               |
                    | Continue with Installation |
                    | Cancel                     |
                    | Help                       |
                    +----------------------------+
```

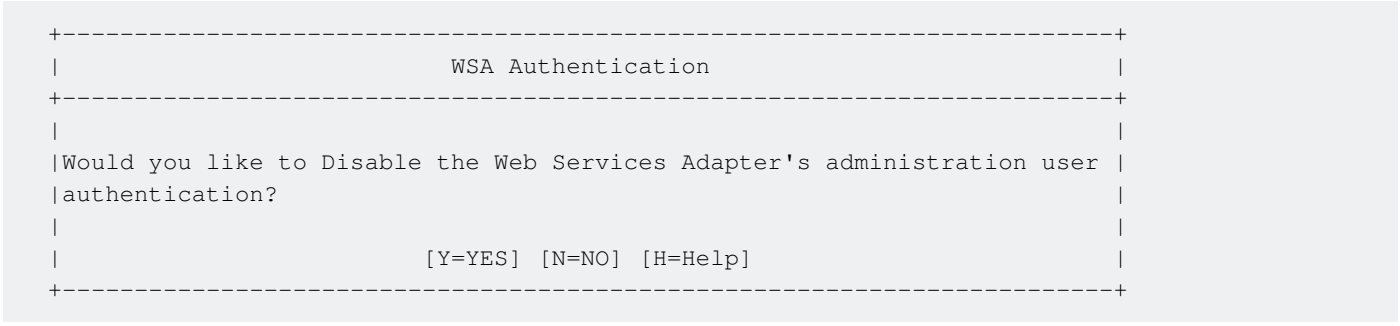
Choisissez l' `English` moins que vous ayez vraiment besoin d'autre chose, vous pouvez en sélectionner plus d'un - faites un défaut dans ce cas. Continuer!



Pour les paramètres internationaux, vous devez essayer de faire correspondre toutes les installations précédentes pour vous aider à l'avenir. Sinon, vous pouvez le configurer selon vos besoins. Cela peut être changé dans le futur. Utilisez `UTF-8` si vous le souhaitez.



Conservez les valeurs par défaut pour l'URL de l'adaptateur de services Web, sauf si vous avez une bonne raison.



Désactiver l'authentification utilisateur? Très probablement `N` est ce que vous voulez.

```
|                          Complete Installation                          |
+-------------------------------------------------------------------------+
|                                                                         |
|The following products will be installed:                                |
|'4GL Development System (x USERS)', 'OE Application Svr Ent (y USERS)', |
|'OE Enterprise RDBMS (z USERS)', 'OpenEdge Replication (u USERS)'      |
|                                                                         |
|Disk Space Required for Products: 1,138,163,712 bytes                    |
|Disk Space Required for Installation: 1,139,343,360 bytes                |
|Disk Space Remaining After Installation: 26,534,129,664 bytes            |
|                                                                         |
|Selected Destination Path: /usr/dlc                                      |
|                                                                         |
|Do you want to install the above listed product(s)?                      |
|                                                                         |
|                         [Y=YES] [N=NO] [H=Help]                         |
+-------------------------------------------------------------------------+
```

Ceci est l'écran final (mais un) avant que l'installation commence.

```
                  +-----------------------------+
                  |        Copy Scripts?        |
                  +-----------------------------+
                  |                             |
                  |Copy the scripts to /usr/bin?|
                  |                             |
                  |   [Y=YES] [N=NO] [H=Help]   |
                  +-----------------------------+
```

Si vous choisissez de le faire, vous voudrez peut-être vous assurer qu'une installation précédente n'est pas remplacée.

```
+-------------------------------------------------------------------------+
|                            Installing Files                             |
+-------------------------------------------------------------------------+
|                                                                         |
|                 Installing subcomponent: Common Files (m)               |
|                        Installing file: libjvm.so                       |
|                                    17%                                  |
| +---------------------------------------------------------------------+ |
| |                                                                     | |
| +---------------------------------------------------------------------+ |
|                                                                         |
|                              [CTRL-T=Quit]                              |
+-------------------------------------------------------------------------+
```

Installation en cours. Prend une minute ou deux.

```
 +-------------------------------------------------------------------------+
 |                          Configuring WebSpeed                           |
 +-------------------------------------------------------------------------+
 |                                                                         |
 | a. Set up and start your Web server                                     |
 |    - If you did not select to "Copy static HTML files to               |
 |      Document Root directory", then manually copy the files             |
 |      or set a link.                                                     |
 |    - For NSAPI Messenger, edit the "obj.conf" and "start" files         |
```

```
|      on the Web server.                                                  |
| b. Set up the Broker machine.                                            |
|    - Set environment variables if necessary.                             |
|    - Edit the properties file (ubroker.properties), then start Broker.   |
| c. To validate your configuration through the Messenger                  |
|    Administration Page, enter ?WSMAdmin after the Messenger name         |
|    in a URL.                                                             |
|    (For example, for CGI, http://hostname/cgi-bin/wspd_cgi.sh?WSMAdmin)  |
|    (For example, for NSAPI, http://hostname/wsnsa.dll?WSMADmin)          |
|                                                                          |
|See the "OpenEdge Application Server: Administration" guide for details.  |
|                                                                          |
|                           [Enter=OK] [H=Help]                            |
+--------------------------------------------------------------------------+
```

Quelques informations sur WebSpeed.

```
   +-----------------------------------------------------------+
   |Installation of selected OpenEdge products is complete.    |
   |Refer to the installation notes for more information.      |
   +-----------------------------------------------------------+
   | End the OpenEdge Installation                             |
   | View Release Notes                                        |
   | Help                                                      |
   +-----------------------------------------------------------+
```

Écran final - Termine l'installation ou affiche les notes de publication.

Vous avez terminé!

Installation silencieuse

L'installation a stocké un fichier nommé `/usr/dlc/install/response.ini` (ou votre répertoire d'installation). Ce fichier peut être utilisé pour répéter exactement la même installation dans une installation "silencieuse" qui peut être scriptée et exécutée sans aucune interaction.

Pour exécuter une installation silencieuse, faites simplement:

```
/path-to-proinst/proinst -b /path-to-response-file/response.ini -l /path-to-store-
log/silent.log
```

Bonjour le monde!

Une fois que vous avez démarré votre éditeur de progrès (il y a plusieurs options), écrivez simplement:

```
DISPLAY "Hello, World!".
```

Et exécutez en appuyant sur la touche ou l'élément de menu correspondant:

Sous Windows dans AppBuilder: `F1` (Compile -> Run)

Sous Linux / Unix dans l'éditeur 4GL: `F2` (ou `Ctrl` + `X`) (Compile -> Run)

Sous Windows dans Developer Studio: `alt` + `shift` + `X` , suivi de `G` (Exécuter -> Application Exécuter en tant que OpenEdge)

FizzBuzz

Un autre exemple de programme de style "Hello World" est FizzBuzz .

```
DEFINE VARIABLE i AS INTEGER     NO-UNDO.
DEFINE VARIABLE cOut AS CHARACTER   NO-UNDO.

DO i = 1 TO 100:

    /* Dividable by 3: fizz */
    IF i MODULO 3 = 0 THEN
        cOut = "Fizz".
    /* Dividable by 5: buzz */
    ELSE IF i MODULO 5 = 0 THEN
        cOut = "Buzz".
    /* Otherwise just the number */
    ELSE
        cOut = STRING(i).

    /* Display the output */
    DISPLAY cOut WITH FRAME x1 20 DOWN.
    /* Move the display position in the frame down 1 */
    DOWN WITH FRAME x1.
END.
```

Mise en place de l'environnement

Linux / Unix

Une fois Progress installé, il est très facile à exécuter.

Vous avez seulement besoin de quelques variables d'environnement. Le répertoire où Progress a été installé (par défaut `/usr/dlc` mais peut être autre chose) doit figurer dans la variable DLC

```
DLC=/usr/dlc
```

Et vous pourriez aussi vouloir le sous-répertoire `"bin"` de `DLC` dans votre `PATH` :

```
PATH=$PATH:$DLC/bin
```

Maintenant vous êtes prêt!

Theres également un script installé appelé `proenv` qui le fera (et un peu plus) pour vous. Son emplacement par défaut est `/usr/dlc/bin/proenv` .

Quelques utilitaires:

```
showcfg
```

Cela listera tous vos produits installés.

```
pro
```

Cela va lancer l'éditeur de procédure où vous pouvez éditer et exécuter vos programmes.

```
pro program.p
```

Ouvrira program.p pour le modifier s'il existe. Sinon, une erreur sera affichée.

```
pro -p program.p
```

Cela lancera "program.p". Si un fichier compilé (program.r) est présent, il sera exécuté, sinon il sera temporairement compilé et exécuté. Le fichier compilé ne sera pas enregistré.

Création de la base de données de démonstration "sports2000" à partir de la ligne de commande

Cela montre comment créer la base de données de démonstration utilisée dans la documentation de Progress: sports2000.

Cela suppose que vous avez installé les produits Progress avec au moins un type de licence de base de données.

Exécutez le script `proenv` / bat-file qui vous donnera une invite avec toutes les variables d'environnement définies.

Créez un répertoire

Cet exemple concerne Windows. La gestion des répertoires, etc. peut être différente dans un autre système d'exploitation.

```
proenv> cd \
proenv> mkdir db
proenv> cd db
proenv> mkdir sports2000
proenv> cd sports2000
```

Créer une base de données sports2000 en utilisant "prodb"

```
proenv> prodb mySportsDb sports2000
```

Syntaxe de prodb:

nom prodb nom-de-nouvelle-base nom-et-chemin-de-source-base

Cela créera une base de données appelée "mySportsDb" dans le répertoire en cours. Cette base de données est une copie exacte de la base de données sports2000 fournie avec l'installation Progress. Comme la base de données sports2000 source se trouve dans le répertoire

d'installation Progress, vous n'avez pas besoin de spécifier de chemin.

Si vous regardez le contenu du répertoire, vous verrez certains fichiers:

```
proenv> dir
2017-01-12  20:24         2 228 224 mySportsDb.b1
2017-01-12  20:24         1 703 936 mySportsDb.d1
2017-01-12  20:24            32 768 mySportsDb.db
2017-01-12  20:24             2 951 mySportsDb.lg
2017-01-12  20:07               368 mySportsDb.st
2017-01-12  20:24           327 680 mySportsDb_10.d1
2017-01-12  20:24            65 536 mySportsDb_10.d2
2017-01-12  20:24         1 310 720 mySportsDb_11.d1
2017-01-12  20:24         1 376 256 mySportsDb_11.d2
2017-01-12  20:24           327 680 mySportsDb_12.d1
2017-01-12  20:24            65 536 mySportsDb_12.d2
2017-01-12  20:24           327 680 mySportsDb_7.d1
2017-01-12  20:24            65 536 mySportsDb_7.d2
2017-01-12  20:24           655 360 mySportsDb_8.d1
2017-01-12  20:24           655 360 mySportsDb_8.d2
2017-01-12  20:24           327 680 mySportsDb_9.d1
2017-01-12  20:24            65 536 mySportsDb_9.d2
```

Nom de fichier	Contient
.db	Le fichier de base de données principal. Contient le schéma de base de données
.lg	Le fichier journal de la base de données. Contient des informations de journalisation au format texte
.st	Le fichier de structure de la base de données. Décrire la disposition du stockage dans un format texte
.ré?	Les données réelles. Différents fichiers stockent des données de différents formats. Le fichier .st peut indiquer quel format
.b?	Fichiers avant image. Contient des informations sur les transactions en cours.

Vous pouvez maintenant accéder directement à la base de données en tapant simplement `pro mySportsDb` . Cela démarrera un éditeur de progression connecté à la base de données. Ce sera une connexion mono-utilisateur afin que personne d'autre ne puisse accéder à la base de données en même temps.

Dans l'éditeur, vous pouvez simplement taper:

```
FOR EACH bill NO-LOCK:
  DISPLAY bill.
END.
```

Pour accéder à la base de données. Appuyez sur `Ctrl` + `X` pour exécuter. Cela affichera tout le

contenu de la table "facture". Si vous souhaitez annuler, vous pouvez appuyer sur `Ctrl` + `C`.

Code de commentaire

```
/*
In all versions of
Progress ABL you can write
multi line comments
*/

/* They can also span a single line */

//Starting with version 11.6 you can also write single line comments

//Can you nest single line comments? //Yes you can

string = "HELLO". //A single line comment can be written after some code

string2 = "Goodbye". /* And the same thing
goes for multi line comments. A difference is
that a multi line comment also can preceed some code */ i = 1.

/* Is it possible to mix comments?
//Yes, but multi line comments always needs to be terminated! */

/* You can nest multi line comments as well
/* but then all nested comments must be terminated */ or the compiler
will generate an error */
```

Formellement, le commentaire sur une seule ligne commence par la double barre oblique `//` et se termine par une nouvelle ligne, un retour chariot ou une fin de fichier.

Fichiers de programme

Le code de progression ABL est normalement stocké dans des fichiers avec une fin différente selon ce qu'ils contiennent. Les fins sont facultatives mais plutôt une norme de facto:

Extension de nom de fichier	Contient
.p	Un programme de progrès. Peut contenir plusieurs procédures internes, fonctions, etc.
.je	Inclure le fichier à inclure dans d'autres fichiers
.w	Un fichier contenant une représentation graphique d'une fenêtre ou d'un dialogue, basée sur WinForm.
.r	Le résultat compilé de tout fichier contenant Progress 4GL. Appelé r-code.
.cls	Une classe orientée objet de progression

Extension de nom de fichier	Contient
.wrx	Un conteneur pour les données ActiveX si nécessaire (généré en compilant dans "AppBuilder").

Pour exécuter un fichier programme dans Progress 4GL, l'instruction `RUN` est utilisée:

```
RUN program.p. //Will run program.p without parameters.
RUN program.w (INPUT true). //Will run program.w with input parameter set to true.

RUN program. //Will run program.r if present otherwise internal procedure "program".
```

Pour inclure un autre fichier dans un programme de progression, la directive `{}` est utilisée:

```
{program.i} //Includes program.i in the current program
```

Faire du sport2000 en tant que service

Une fois la base de données sports2000 installée, il est temps de l'exécuter en tant que serveur autonome (et de ne pas s'y connecter en tant que fichier).

Démarrez proenv (`proenv` dans le startmeny sous Windows ou `/usr/install-directory/bin/proenv` sous Linux / Unix).

Cet exemple provient de Windows. Linux est le même, mais vous devez modifier les chemins, etc. pour correspondre à votre installation.

```
proenv> cd \db\sports2000
proenv> proserve mySportsDb -H localhost -S 9999
OpenEdge Release 11.6 as of Fri Oct 16 19:01:51 EDT 2015
20:09:54 BROKER     This broker will terminate when session ends. (5405)
20:09:54 BROKER     The startup of this database requires 17Mb of shared memory.  Maximum
segment size is 128Mb.
20:09:54 BROKER  0: Multi-user session begin. (333)
20:09:55 BROKER  0: Begin Physical Redo Phase at 0 . (5326)
20:17:36 BROKER  0: Before Image Log Initialization at block 1  offset 5300. (15321)
20:09:55 BROKER  0: Login by xyz on CON:. (452)
20:09:55 BROKER  0: Started for 9999 using TCP IPV4 address 127.0.0.1, pid 2892. (5644)
proenv>
```

(Vous pourriez ne pas obtenir exactement cette sortie).

Cela démarrera mySportsDb sur `localhost` et utilisera le port 9999 comme port principal pour l'accès à la base de données. Si vous souhaitez vous connecter à cette base de données à partir d'un autre client sur le même réseau ou ailleurs, localhost ne fonctionnera pas. Utilisez plutôt votre adresse IP ou votre nom d'hôte:

```
proenv> proserve mySportsDb -H 192.168.1.10 -S 9999.
```

Connexion et déconnexion

Une fois que votre base de données est opérationnelle, vous pouvez vous y connecter dans votre éditeur de progression:

```
CONNECT mySportsDb -H localhost -S 9999.
```

ou

```
CONNECT "-db mySportsDb -H localhost -S 9999".
```

Si vous obtenez un message d'erreur, vous avez soit des informations erronées dans la commande, soit la base de données n'est pas opérationnelle. Vous pourriez également avoir un pare-feu logiciel ou une interférence similaire.

Vous pouvez vérifier le fichier journal de la base de données (`mySportsDb.lg` dans cet exemple) pour tous les indices.

La déconnexion est tout aussi simple:

```
DISCONNECT mySportDb.
```

ou

```
DISCONNECT "mySportsDb".
```

Arrêter la base de données (ou déconnecter les utilisateurs)

Pour fermer la base de données , vous pouvez exécuter le bas `proshut` commande à partir proenv:

```
proenv> proshut mySportsDb
OpenEdge Release 11.6 as of Fri Oct 16 19:01:51 EDT 2015
usr    pid    time of login           user id     Type  tty                  Limbo?
 24   7044 Wed Feb 01 20:22:57 2017   xyz         REMC  XYZ-PC               no
            1  Disconnect a User
            2  Unconditional Shutdown
            3  Emergency Shutdown (Kill All)
            x  Exit
```

1. Utilisez `1` pour déconnecter des utilisateurs spécifiques.
2. Utilisez `2` pour arrêter la base de données. **Remarque:** pas de questions posées, l'arrêt démarre directement!
3. Utilisez `3` uniquement si vous ne pouvez pas supprimer la base de données autrement. Cela pourrait corrompre vos données.
4. Utilisez `x` pour quitter l'utilitaire proshut.

Vous pouvez également arrêter la base de données directement depuis la ligne de commande:

```
proenv>proshut mySportsDb -by
```

Ou déconnectez un utilisateur de la ligne de commande (en supposant que vous connaissez son

numéro d'utilisateur, usr dans la liste ci-dessus):

```
proenv>proshut mySportsDb -C disconnect 24
OpenEdge Release 11.6 as of Fri Oct 16 19:01:51 EDT 2015
User 24 disconnect initiated. (6796)
```

Lire Démarrer avec progress-4gl en ligne: https://riptutorial.com/fr/progress-4gl/topic/8124/demarrer-avec-progress-4gl

Chapitre 2: Compiler

Introduction

Compiler le code de progression appelé "code-r" et est normalement enregistré dans un fichier avec l'extension .r. Il existe différentes méthodes de compilation: à l'aide de l'instruction `COMPILE` ou sous Linux ou AppBuilder: le compilateur d'application intégré. Developer Studio (l'environnement Eclipse) a été compilé dans son processus de construction.

Vous devez avoir 4GL Development ou OpenEdge Studio installé pour compiler les programmes 4GL qui mettent à jour la base de données.

Syntaxe

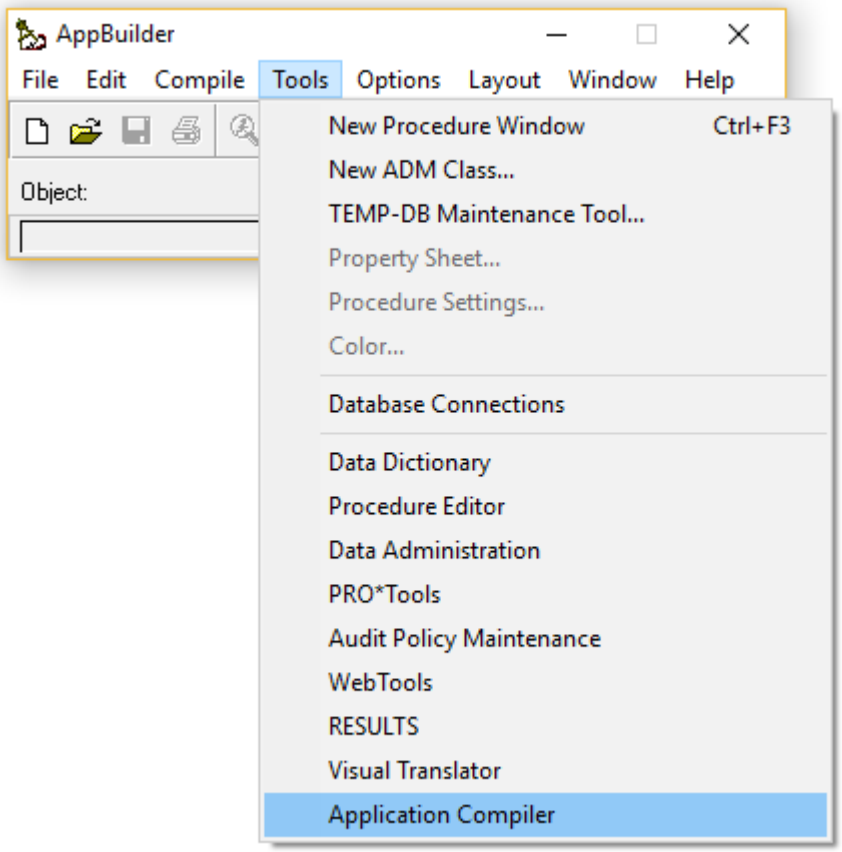
- COMPILE program.p SAVE. // Compile program.p et sauvegarde son code r
- Valeur de compilation (var) SAVE. // Compile le nom enregistré dans la variable "var" et enregistre son code-r
- COMPILE prog.p XREF prog.xref LISTING prog.list. // Compile prog.p et crée des fichiers xref et listing-files. Ne sauvegardez pas le r-code.
- COMPILE program.p SAVE NO-ERROR. // Compilez program.p, enregistrez le code r et supprimez les erreurs pour arrêter l'exécution.

Examples

Compilateur d'application

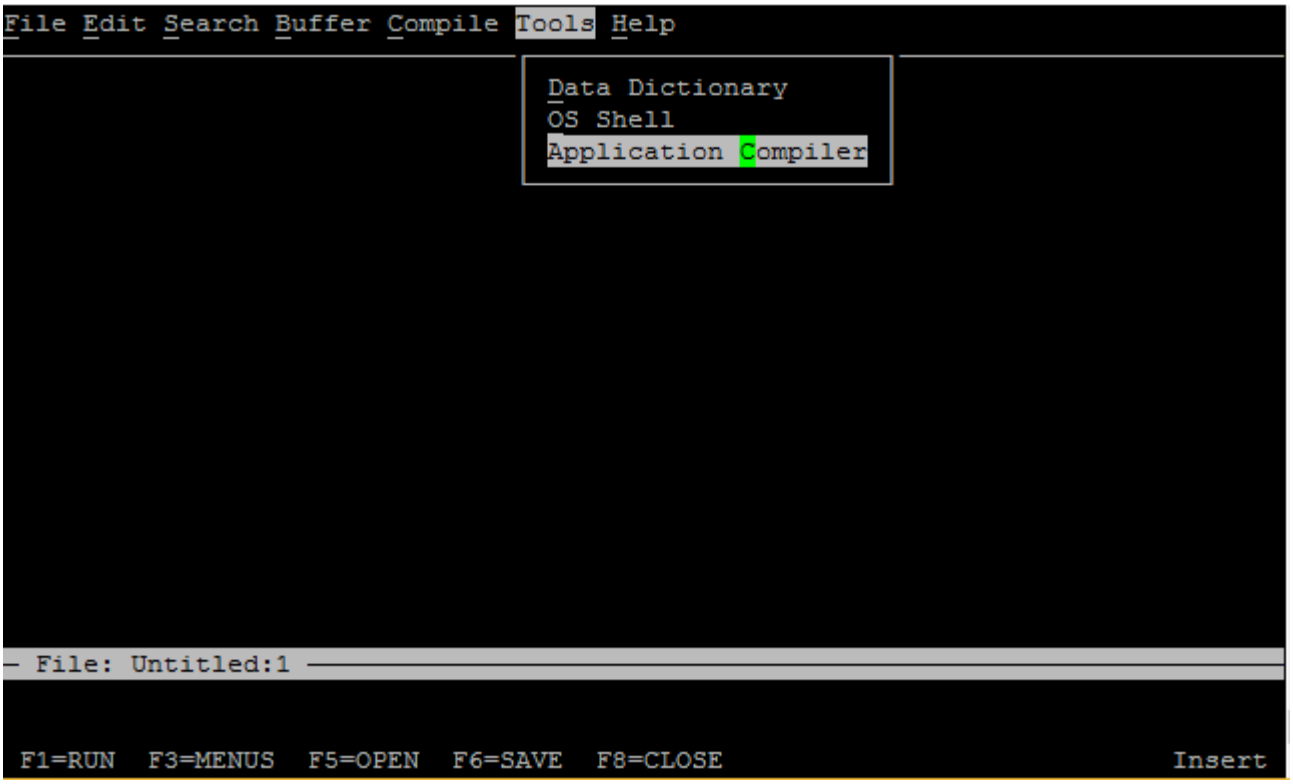
Windows AppBuilder

Dans Windows Appbuilder, le compilateur d'application se trouve dans le menu Outils.

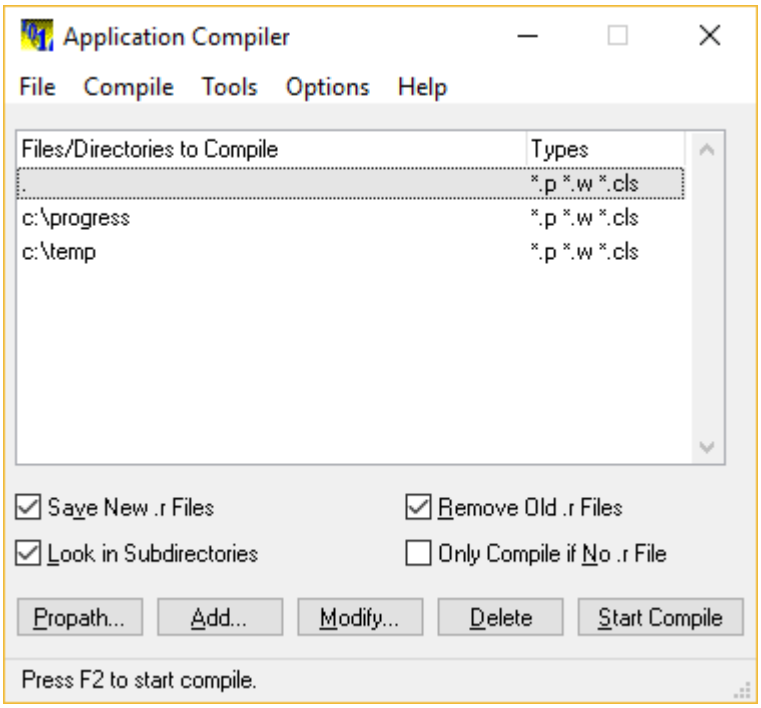


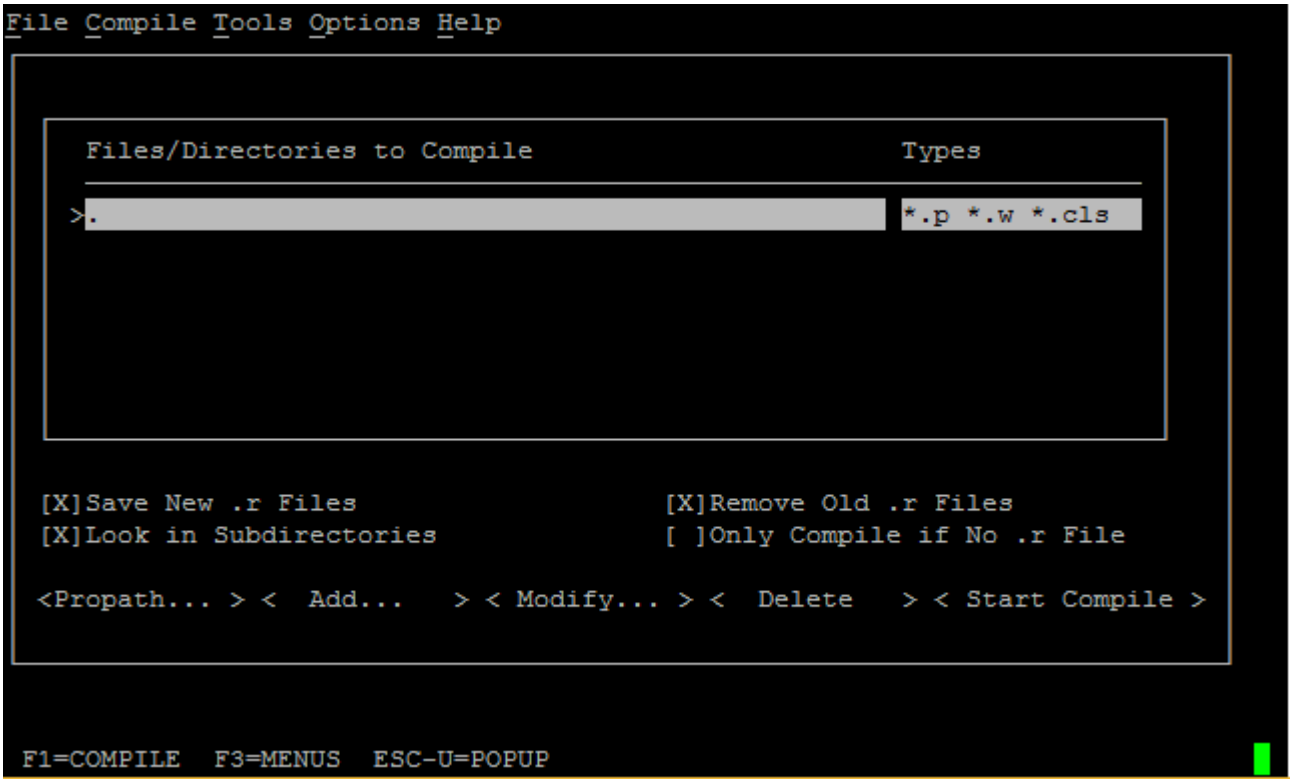
Éditeur de procédure (Linux - `pro` ou Windows `pro.exe`

Dans l'éditeur de procédures (Linux et Windows), le compilateur se trouve dans le menu Outils.



Compilateur d'application





Indépendamment du système d'exploitation, les fonctionnalités du compilateur sont les mêmes. Vous pouvez ajouter des répertoires et / ou des fichiers et les compiler.

Paramètres principaux (plus ci-dessous):

- Enregistrez le nouveau fichier .r. Si ce n'est pas coché, les fichiers seront simplement compilés mais pas enregistrés. Utile pour le suivi des erreurs, par exemple.
- Regardez dans les sous-répertoires. Sinon, les sous-répertoires devront être ajoutés.
- Supprimez les anciens fichiers .r. Ecraser l'ancien fichier .r
- Onlu Compiler si No .r File. Ne compile que les fichiers non compilés.

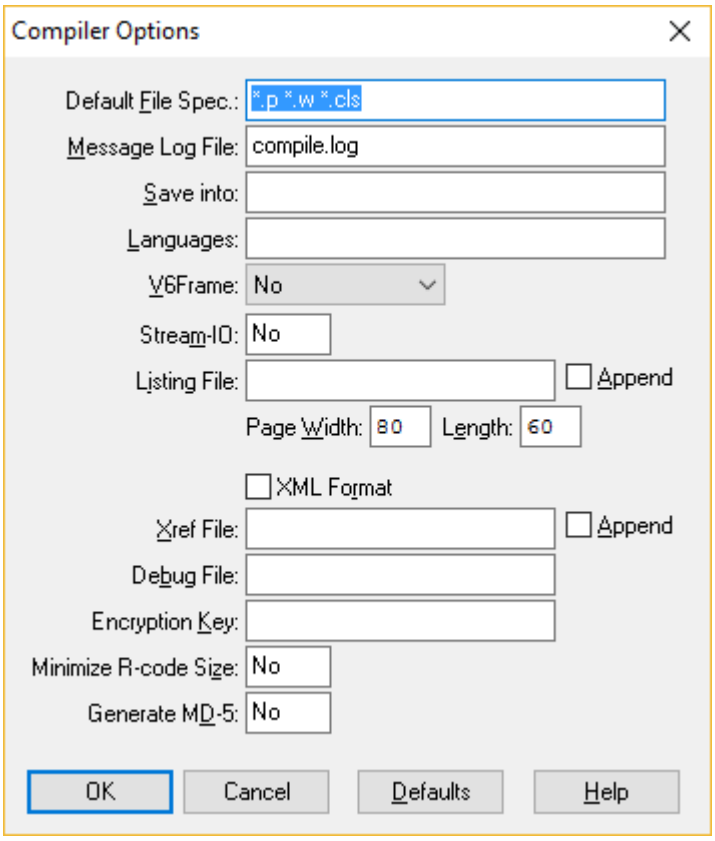
Options:

- Propath - vous montre le propath et vous permet de sélectionner les répertoires à compiler.
- Ajouter - vous permet d'entrer un répertoire ou un fichier.
- Modifier - permet de modifier une entrée existante.
- Supprimer - Supprime une entrée.
- Démarrer la compilation - Démarre le compilateur. Raccourci: `F2`

Les choix de menu principaux:

- Fichier -> Quitter: quitte le compilateur
- Compiler -> Start Compile: lance le compilateur. Raccourci: `F2`
- Outils -> Accès à d'autres outils
- Option -> Compilateur ...: Paramètres, voir ci-dessous.
- Aide -> Aide OpenEdge (Windows uniquement). Aide en ligne. Raccourci: `F1`

Paramètres



- Spécification de fichier par défaut: extensions de nom de fichier à compiler
- Message Log File: Fichier pour enregistrer les messages, les avertissements et les erreurs dans
- Enregistrer dans: Où stocker le fichier .r. Si vide le même répertoire que le code.
- Langues: pour les traductions. Non couvert ici.
- V6Frame: Ancien et inutile ...
- Steam-IO: Si vous souhaitez imprimer le résultat du compilateur. Préférablement pas.
- Liste de fichiers: Si vous souhaitez que le compilateur crée un fichier de liste. Utile pour le débogage

- Ajouter: ajouter au fichier de liste existant. Sinon écraser.
- Largeur de page + Longueur: Format du fichier de listage.
- Fichier Xref: Si vous souhaitez que le compilateur crée un `XREF` . Utile pour le débogage, la vérification de l'utilisation des index, etc.
- Format XML: Si le compilateur xref doit être un XML. Sinon, texte "clair".
- Ajouter: ajoute au fichier xref existant. Sinon écraser.
- Fichier de débogage: fichier de liste du débogueur.
- Clé de cryptage: si le fichier source est crypté à l'aide de `xcode` insérez la clé ici.
- Réduire la taille du code R: Supprimez certaines informations de débogage pour garder le code r plus petit.
- Générer MD-5: Principalement pour la compilation WebClient.

Utilisation de base

1. Démarrer le compilateur
2. Ajouter un chemin (s'il n'est pas déjà enregistré à la dernière session)
3. Appuyez sur `F2` pour compiler.
4. Observez les erreurs.
5. Sortie

Déclaration COMPILE

L'instruction de compilation vous permet de compiler des programmes en Progress ABL:

Utilisation de base:

```
COMPILE hello-world.p SAVE.
```

Avec une variable:

```
DEFINE VARIABLE prog AS CHARACTER   NO-UNDO.

prog = "hello.p".

COMPILE VALUE(prog) SAVE.
```

Il y a plusieurs options à l' `COMPILE` COMPILE:

`SAVE` indique que le code .r doit être enregistré pour une utilisation ultérieure.

```
COMPILE hello-world.p SAVE.
```

`SAVE INTO dir OR SAVE INTO VALUE(dir-variable)` enregistre le r-code dans le répertoire spécifié:

```
COMPILE hello-world.p SAVE INTO /usr/sources.
```

`LISTING file` . Crée un fichier de liste contenant des informations de débogage concernant les blocs, les inclusions, etc.

```
COMPILE program.p SAVE LISTING c:\temp\listing.txt.
```

Le listing a quelques options pour ajouter des fichiers, taille de page et largeur de page:

```
APPEND PAGE-SIZE num PAGE-WIDTH num
```

`XREF xreffile` enregistre un fichier xref de compilateur contenant des informations sur l'utilisation de la chaîne et de l'index, etc. Vous pouvez également `APPEND` celui-ci.

```
COMPILE checkFile.p SAVE XREF c:\directory\xref-file.txt.
```

`XREF-XML xreffile-or-dir` fera la même chose que `XREF` mais enregistre le fichier au format xml à la place. Si vous utilisez un répertoire, le fichier xref s'appellera `programname.xref.xml` .

```
COMPILE file.p SAVE XREF c:\temp\.
```

`NO-ERROR` supprimera toute erreur d'arrêt de votre programme.

```
COMPILE program SAVE NO-ERROR.
```

`DEBUG-LIST file` génère un fichier de débogage avec des numéros de ligne.

```
COMPILE checkFile.p SAVE DEBUG-LIST c:\temp\debug.txt.
```

`PREPROCESS file` traduira d'abord tous les préprocesseurs, puis créera un nouveau `PREPROCESS file` avec le code avant la compilation.

```
 COMPILE checkFile.p SAVE PREPROCESS c:\temp\PREPROC.txt.
```

`XCODE key` compilera un code source chiffré avec la `key` comme clé. Vous ne pouvez pas utiliser XCODE avec les options XREF, XREF-XML, STRING-XREF ou LISTING.

```
COMPILE program.p SAVE XCODE myKey.
```

Vous pouvez combiner plusieurs options:

```
COMPILE prog.p SAVE INTO /usr/r-code XREF /usr/xrefs/xref.txt APPEND LISTING /usr/listings.txt
APPEND NO-ERROR.
```

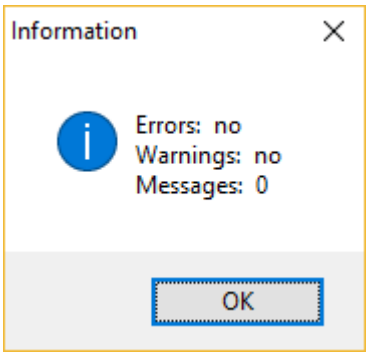
Poignée du système COMPILER

Le `COMPILER` système `COMPILER` vous permet d’examiner des informations concernant une compilation récente.

En supposant que `ok-program.p` est un programme sans erreur ni avertissement:

```
COMPILE ok-program.p SAVE NO-ERROR.

DEFINE VARIABLE iError AS INTEGER     NO-UNDO.

MESSAGE
    "Errors: "   COMPILER:ERROR SKIP
    "Warnings: " COMPILER:WARNING SKIP
    "Messages: " COMPILER:NUM-MESSAGES
    VIEW-AS ALERT-BOX INFORMATION.
```

Cela va aboutir:



Compiler un programme avec un avertissement:

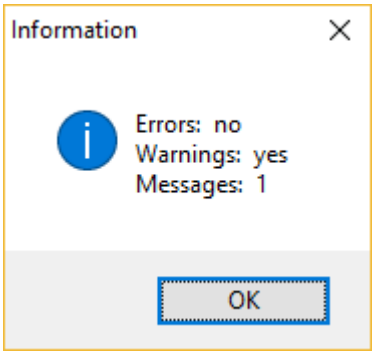
```
/* program-with-warning.p */
DEFINE VARIABLE c AS CHARACTER   NO-UNDO.
DEFINE VARIABLE i AS INTEGER     NO-UNDO.

c = "hello".
DISPLAY c.
//This RETURN makes the program exit here and the code below unreachable.
RETURN.

IF TRUE THEN DO:
    i = 10.
END.
```

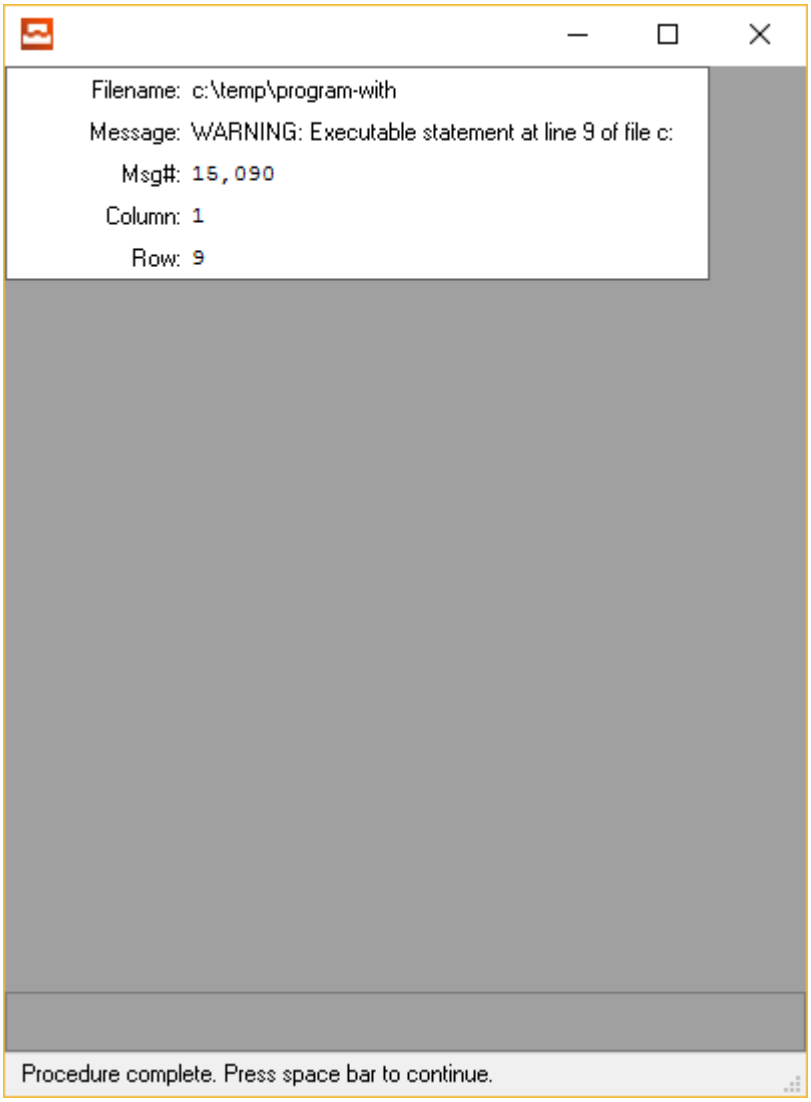
Compiler le programme:

```
COMPILE program-with-warning.p SAVE.

DEFINE VARIABLE iError AS INTEGER     NO-UNDO.

MESSAGE
    "Errors: "   COMPILER:ERROR SKIP
    "Warnings: " COMPILER:WARNING SKIP
    "Messages: " COMPILER:NUM-MESSAGES
    VIEW-AS ALERT-BOX INFORMATION.

DO iError = 1 TO COMPILER:NUM-MESSAGES:
    DISPLAY
        COMPILER:GET-FILE-NAME(iError)    LABEL "Filename" FORMAT "x(20)"
        COMPILER:GET-MESSAGE(iError)      LABEL "Message"  FORMAT "x(50)"
        COMPILER:GET-NUMBER(iError)       LABEL "Msg#"
        COMPILER:GET-ERROR-COLUMN(iError) LABEL "Column"
        COMPILER:GET-ERROR-ROW(iError)    LABEL "Row"
```

```
            WITH FRAME fr1 SIDE-LABELS 1 COLUMNS.
END.
```

Résultat:





Compiler un programme avec une erreur

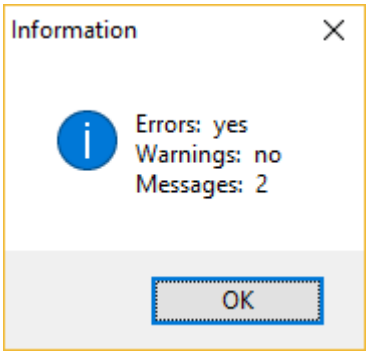
```
DEFINE VARIABLE c AS CHARACTER   NO-UNDO.
DEFINE VARIABLE i AS INTEGER     NO-UNDO.

c = "hello".
DISPLAY c.
//Casting should be required below...
```

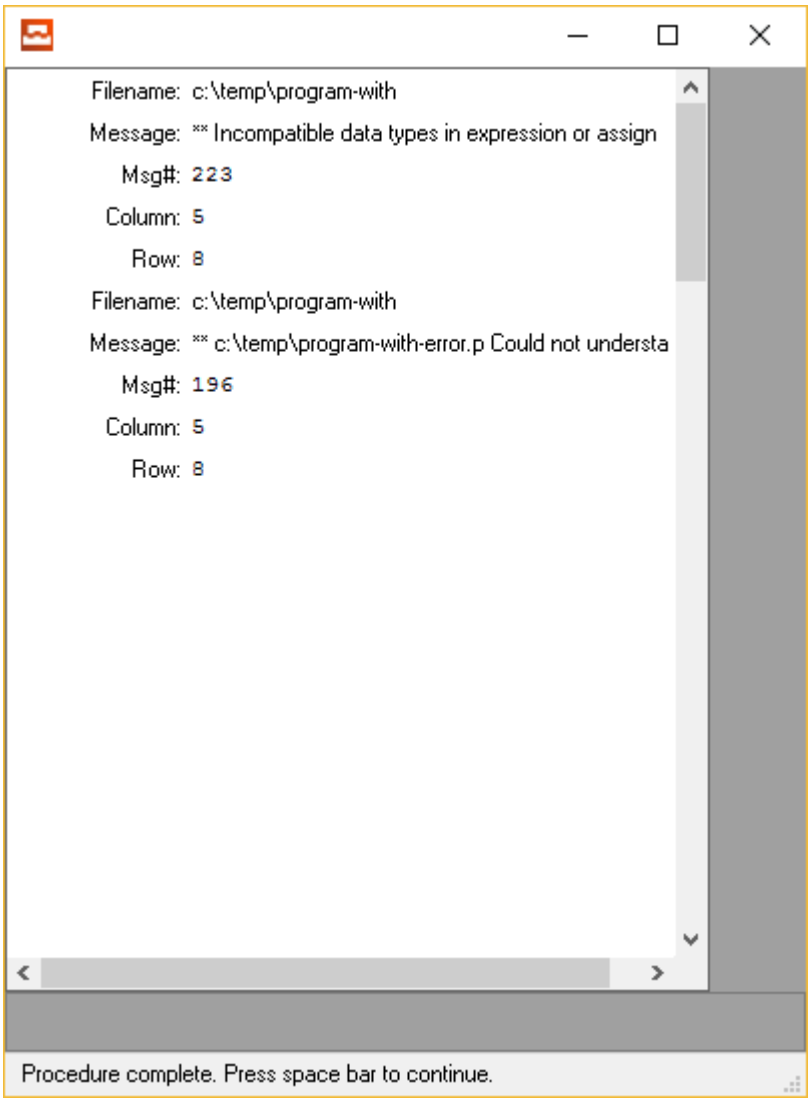
```
IF TRUE THEN DO:
    i = c.
END.
```

Compiler le programme:

```
//Use no-errors to supress any error messages from interrupting us.
COMPILE c:\temp\program-with-error.p SAVE NO-ERROR.

DEFINE VARIABLE iError AS INTEGER     NO-UNDO.

MESSAGE
    "Errors: "   COMPILER:ERROR SKIP
    "Warnings: " COMPILER:WARNING SKIP
    "Messages: " COMPILER:NUM-MESSAGES
    VIEW-AS ALERT-BOX INFORMATION.

DO iError = 1 TO COMPILER:NUM-MESSAGES:
    DISPLAY
        COMPILER:GET-FILE-NAME(iError)    LABEL "Filename" FORMAT "x(20)"
        COMPILER:GET-MESSAGE(iError)      LABEL "Message"  FORMAT "x(50)"
        COMPILER:GET-NUMBER(iError)       LABEL "Msg#"
        COMPILER:GET-ERROR-COLUMN(iError) LABEL "Column"
        COMPILER:GET-ERROR-ROW(iError)    LABEL "Row"
            WITH FRAME fr1 SIDE-LABELS 1 COLUMNS 20 DOWN.

    DOWN WITH FRAME fr1.
END.
```

Résultat, il y a presque toujours deux erreurs par erreur. "Impossible de comprendre" est suivi de l'erreur réelle:





Lire Compiler en ligne: https://riptutorial.com/fr/progress-4gl/topic/9029/compiler

Chapitre 3: Cordes

Introduction

Dans Progress ABL, il existe deux types de chaînes, celles définies comme `CHARACTER` et celles définies comme `LONGCHAR` . Un fichier de plus de 32K de longueur est un `LONGCHAR` . La plupart des chaînes sont à moins d'indication contraire de la casse.

Remarques

Rappelez-vous - toutes les positions commencent par la position 1!

Examples

Définir, associer et afficher une chaîne

En général, vous devez toujours définir toutes les variables et tous les paramètres comme `NO-UNDO` sauf si vous en avez vraiment besoin.

```
DEFINE VARIABLE cString AS CHARACTER   NO-UNDO.

cString = "HELLO".

DISPLAY cString.
```

Chaînes concaténantes

En utilisant l'opérateur `+` , vous pouvez facilement concaténer deux chaînes ou plus.

```
DEFINE VARIABLE cString AS CHARACTER   NO-UNDO.

cString = "HELLO".

cString = cString + " " + "GOODBYE".

DISPLAY cString FORMAT "X(20)".
```

Manipulation de cordes

Il existe deux fonctions intégrées utiles pour travailler avec des chaînes. Toutes les fonctions fonctionnant avec la position des caractères commencent par l'index 1 en tant que premier caractère, et non 0, ce qui est courant dans de nombreuses langues.

STRING - convertit toute valeur en chaîne

Cet exemple convertit le nombre entier 2000 en chaîne "2000".

```
DEFINE VARIABLE i AS INTEGER     NO-UNDO.
DEFINE VARIABLE c AS CHARACTER   NO-UNDO.

i = 2000.

c = STRING(i).

DISPLAY c.
```

CHR et **ASC** - convertit les caractères uniques en ascii.

CHR (entier)

Renvoie la représentation des caractères pour un entier de code ascii

ASC (caractère)

Renvoie la valeur entière ascii du caractère

```
DEFINE VARIABLE ix     AS INTEGER   NO-UNDO.
DEFINE VARIABLE letter AS CHARACTER NO-UNDO FORMAT "X(1)" EXTENT 26.

DO ix = 1 TO 26:
  letter[ix] = CHR((ASC("A")) - 1 + ix).
END.

DISPLAY SKIP(1) letter WITH 2 COLUMNS NO-LABELS
  TITLE "T H E  A L P H A B E T".
```

LONGUEUR - renvoie la longueur d'une chaîne

LONGUEUR (chaîne). // Retourne un entier avec la longueur de la chaîne.

```
DEFINE VARIABLE cString AS CHARACTER   NO-UNDO.

cString = "HELLO".

MESSAGE "The string " cString " is " LENGTH(cString) " characters long" VIEW-AS ALERT-BOX.
```

SUBSTRING - retourne ou assigne une partie d'une chaîne

- SUBSTRING (chaîne, position de départ, longueur).

Renvoie les caractères "length" à partir de "string" à partir de la position "start-position".

- SUBSTRING (chaîne, position de départ).

Renvoie le reste de "string", en commençant à la position "start-position"

```
DEFINE VARIABLE cString AS CHARACTER   NO-UNDO.

cString = "ABCDEFGH".
```

```
DISPLAY SUBSTRING(cString, 4, 2). //Displays "DE"
DISPLAY SUBSTRING(cString, 4). //Displays "DEFGH"
```

La sous-chaîne peut également être utilisée pour écraser une partie d'une chaîne. Utilisez la même syntaxe, mais affectez cette sous-chaîne à la place:

```
DEFINE VARIABLE cString AS CHARACTER   NO-UNDO.

cString = "ABCDEFGH".

SUBSTRING(cString, 4, 2) = "XY". //Replaces position 4 and 5 with "XY"

DISPLAY cString.
```

Il existe également une fonction similaire appelée `OVERLAY` Cet exemple de la documentation Progress couvre les différences entre `OVERLAY` et `SUBSTRING` :

```
/* This procedure illustrates the differences between the SUBSTRING and
   OVERLAY statements. */
DEFINE VARIABLE cOriginal  AS CHARACTER NO-UNDO INITIAL "OpenEdge".
DEFINE VARIABLE cSubstring AS CHARACTER NO-UNDO.
DEFINE VARIABLE cOverlay   AS CHARACTER NO-UNDO.
DEFINE VARIABLE cResults   AS CHARACTER NO-UNDO.

/* Default behavior without optional LENGTH. */
ASSIGN
 cSubstring              = cOriginal
 SUBSTRING(cSubstring,2) = "***"
 cOverlay                = cOriginal
 OVERLAY(cOverlay,2)     = "***"
 cResults                = "target = ~"OpenEdge~". ~n~n"
  + "If you do not supply a length, SUBSTRING and OVERLAY default as follows:
  ~n~n" + "SUBSTRING(target,2) = ~"***~"  yields:  " + cSubstring + ". ~n"
  + "OVERLAY(target,2)     = ~"***~"  yields:  " + cOverlay + ".".

/* Behavior with zero LENGTH. */
ASSIGN
 cSubstring                = cOriginal
 SUBSTRING(cSubstring,2,0) = "***"
 cOverlay                  = cOriginal
 OVERLAY(cOverlay,2,0)     = "***"
 cResults                  = cResults + "~n~n"
  + "For a zero length, SUBSTRING and OVERLAY behave as follows:  ~n~n"
  + "SUBSTRING(target,2,0) = ~"***~"  yields:  " + cSubstring + ". ~n"
  + "OVERLAY(target,2,0)     = ~"***~"  yields:  " + cOverlay + ".".

/* Behavior with LENGTH < replacement. */
ASSIGN
 cSubstring                = cOriginal
 SUBSTRING(cSubstring,2,1) = "***"
 cOverlay                  = cOriginal
 OVERLAY(cOverlay,2,1)     = "***"
 cResults                  = cResults + "~n~n"
  + "For a length shorter than the replacement, SUBSTRING and OVERLAY behave
  as follows: ~n~n" + "SUBSTRING(target,2,1) = ~"***~"  yields:  "
  + cSubstring + ". ~n" + "OVERLAY(target,2,1)     = ~"***~"  yields:  "
  + cOverlay + ".".
```

```
/* Behavior with LENGTH = replacement. */
ASSIGN
 cSubstring                = cOriginal
 SUBSTRING(cSubstring,2,3) = "***"
 cOverlay                  = cOriginal
 OVERLAY(cOverlay,2,3)     = "***"
 cResults                  = cResults + "~n~n"
  + "For a length equal to the replacement, SUBSTRING and OVERLAY behave as
  follows:  ~n~n" + "SUBSTRING(target,2,3) = ~"***~"  yields:  "
  + cSubstring + ". ~n" + "OVERLAY(target,2,3)     = ~"***~"  yields:  "
  + cOverlay + ".".

/* Behavior with LENGTH > replacement. */
ASSIGN
 cSubstring                = cOriginal
 SUBSTRING(cSubstring,2,6) = "***"
 cOverlay                  = cOriginal
 OVERLAY(cOverlay,2,6)     = "***"
 cResults                  = cResults + "~n~n"
  + "For a length greater than the replacement, SUBSTRING and OVERLAY behave
  as follows:  ~n~n" + "SUBSTRING(target,2,6) = ~"***~"  yields:  "
  + cSubstring + ". ~n" + "OVERLAY(target,2,6)     = ~"***~"  yields:  "
  + cOverlay + ".".

MESSAGE cResults VIEW-AS ALERT-BOX.
```

INDEX - renvoie la position d'une chaîne dans une chaîne.

`R-INDEX` va à la même chose mais chercher de droite à gauche.

> INDEX (source, cible)
>
> Recherchez la cible dans la source (de gauche à droite) et retournez sa position. S'il manque le résultat 0.
>
> INDEX (source, cible, position de départ).
>
> Comme ci-dessus, mais commencez à chercher au départ

```
DEFINE VARIABLE str AS CHARACTER   NO-UNDO.

str = "ABCDEFGH".

DISPLAY INDEX(str, "cd") INDEX(str, "cd", 4). //Will display 3 and 0
```

REPLACE - remplace une chaîne dans une chaîne.

> REPLACE (chaîne, from-string, to-string)
>
> Remplace de string par to-string dans string. From-string et to-string n'ont pas besoin d'avoir la même longueur, to-string peut aussi ne rien avoir ("") pour supprimer un caractère.

```
DEFINE VARIABLE c AS CHARACTER   NO-UNDO.
```

```
c = "ELLO".

DISPLAY REPLACE(c, "E", "HE"). // Displays "HELLO"

c = "ABABABA".

DISPLAY REPLACE(c, "B", ""). // Remove all Bs
```

TRIM - supprime les espaces de début et de fin (ou autres caractères).

Cela peut être utile pour nettoyer l'indata.

> TRIM (chaîne)
>
> Supprime tous les espaces de début et de fin, les tabulations, les sauts de ligne et les retours chariot.
>
> TRIM (chaîne, caractère).
>
> Supprime tous les "caractères" de début et de fin.

`LEFT-TRIM` et `RIGHT-TRIM` font la même chose mais ne font que conduire ou suivre.

```
DEFINE VARIABLE c AS CHARACTER   NO-UNDO.

c = "__HELLO_WORLD_____".

DISPLAY TRIM(c, "_").
/*Displays HELLO_WORLD without all the leading and
trailing underscores but leaves the one in the middle.
REPLACE would have removed that one as well */
```

SUBSTITUTE - remplace les paramètres dans une chaîne.

```
SUBSTITUTE is a limited function for replacing up to nine preformatted parameters in a string.
```

> SUBSTITUTE (chaîne, param1, param2, ..., param9).

Les paramètres doivent être au format `&1` à `&9` .

Si vous voulez utiliser une esperluette dans la chaîne (et ne pas l'utiliser comme paramètre), écartez-la avec une autre perluète: `&&` .

```
DEFINE VARIABLE str AS CHARACTER   NO-UNDO.

str = "&1 made &2 goals in &3 games playing for &4".

MESSAGE SUBSTITUTE(str, "Zlatan Ibrahimovic", 113, 122, "Paris Saint-Germain") VIEW-AS ALERT-
BOX.
MESSAGE SUBSTITUTE(str, "Mats Sundin", 555, 1305, "Toronto Maple Leafs") VIEW-AS ALERT-BOX.
```

Un paramètre peut apparaître plus d'une fois dans une chaîne, tous seront remplacés:

```
MESSAGE SUBSTITUTE("&1 &2 or not &1 &2", "To", "Be") VIEW-AS ALERT-BOX.
```

Cordes SENSIBLES

Toutes les chaînes dans Progress ABL sont sensibles à la casse, sauf indication contraire.

Cet exemple affiche une boîte de message indiquant que les chaînes sont identiques.

```
DEFINE VARIABLE str1 AS CHARACTER   NO-UNDO.
DEFINE VARIABLE str2 AS CHARACTER   NO-UNDO.

str1 = "abc".
str2 = "ABC".

IF str1 = str2 THEN
    MESSAGE "The strings are identical" VIEW-AS ALERT-BOX.
```

Pour déclarer une chaîne sensible à la casse, ajoutez simplement l'attribut `CASE-SENSITIVE`

```
DEFINE VARIABLE str1 AS CHARACTER   NO-UNDO CASE-SENSITIVE.
DEFINE VARIABLE str2 AS CHARACTER   NO-UNDO.

str1 = "abc".
str2 = "ABC".

IF str1 = str2 THEN
    MESSAGE "The strings are identical" VIEW-AS ALERT-BOX.
ELSE
    MESSAGE "There's a difference" VIEW-AS ALERT-BOX.
```

(Il suffit que l'une des chaînes l'a dans ce cas).

COMMENCE et MATCHS

BEGINS - renvoie TRUE si une chaîne *commence* par une autre chaîne.

> string1 BEGINS string2
>
> Si string1 COMMENCE avec (ou est égal à) string2, cela retournera true. Sinon, il retournera faux. Si la chaîne deux est vide (""), elle retournera toujours true.

BEGINS est très utile dans les requêtes où vous souhaitez rechercher le début de quelque chose, par exemple un nom. Mais c'est essentiellement une fonction travaillant sur des chaînes.

```
DEFINE VARIABLE str AS CHARACTER   NO-UNDO.
DEFINE VARIABLE beg AS CHARACTER   NO-UNDO.

str = "HELLO".
beg = "HELLO".
DISPLAY str BEGINS beg. // yes

str = "HELLO".
beg = "H".
```

```
DISPLAY str BEGINS beg. // yes

str = "HELLO".
beg = "".
DISPLAY str BEGINS beg. // yes


str = "HELLO".
beg = "HELLO WORLD".
DISPLAY str BEGINS beg. // no
```

MATCHES renvoie true si certains critères de caractères génériques sont remplis dans une chaîne.

> string1 MATCHES expression
>
> Renvoie true si string1 correspond à l'expression générique:
>
> * (astérisque) = 0 à n caractères (essentiellement n'importe quelle chaîne de n'importe quelle longueur)
>
> . (point) = caractère générique pour tout caractère (sauf null)

```
DEFINE VARIABLE str AS CHARACTER   NO-UNDO.
DEFINE VARIABLE beg AS CHARACTER   NO-UNDO.

str = "HELLO".
beg = "HELLO".
DISPLAY str MATCHES beg. // yes

str = "HELLO".
beg = "H*".
DISPLAY str MATCHES beg. // yes

str = "HELLO".
beg = "*O".
DISPLAY str MATCHES beg. // yes

str = "HELLO WORLD".
beg = "HELLO.WORLD".
DISPLAY str MATCHES beg. // yes

str = "HELLO WORLD".
beg = "*WORL..".
DISPLAY str MATCHES beg. // no

str = "*HELLO WORLD".
beg = "WOR*LD".
DISPLAY str MATCHES beg. // no
```

Conversion des majuscules et minuscules

Comme mentionné précédemment, les chaînes sont normalement insensibles à la casse mais cela ne concerne que la comparaison des chaînes. Il y a des fonctions intégrées pour changer de boîtier.

CAPS (chaîne)

Rend la chaîne supérieure

LC (chaîne)

Rend les cordes minuscules

```
DEFINE VARIABLE c AS CHARACTER   NO-UNDO.
DEFINE VARIABLE d AS CHARACTER   NO-UNDO.

c = "Hello".
d = "World".

DISPLAY CAPS(c) LC(d). // HELLO world
```

Rappelez-vous que les chaînes sont normalement insensibles à la casse

```
DEFINE VARIABLE c AS CHARACTER   NO-UNDO.
DEFINE VARIABLE d AS CHARACTER   NO-UNDO.

c = "hello".
d = "hello".

DISPLAY CAPS(c) = LC(d). // yes
```

Sauf spécification comme `CASE-SENSITIVE`

```
DEFINE VARIABLE c AS CHARACTER   NO-UNDO CASE-SENSITIVE.
DEFINE VARIABLE d AS CHARACTER   NO-UNDO.

c = "hello".
d = "hello".

DISPLAY CAPS(c) = LC(d). // no
```

Des listes

Il existe un certain nombre de fonctions et de méthodes permettant de travailler avec des listes séparées par des virgules (ou d'autres caractères) dans Progress 4GL.

NUM-ENTRIES Renvoie le nombre d'entrées dans une liste. Vous pouvez éventuellement spécifier le délimiteur, la virgule est la valeur par défaut

NUM-ENTRIES (chaîne [, délimiteur])

En utilisant la virgule, le délimiteur par défaut:

```
DEFINE VARIABLE cList AS CHARACTER   NO-UNDO.

cList = "Goodbye,cruel,world!".

DISPLAY NUM-ENTRIES(cList). //3
```

En utilisant un autre délimiteur, semilcolon:

```
DEFINE VARIABLE cList AS CHARACTER   NO-UNDO.

cList = "Goodbye;cruel;world!".

DISPLAY NUM-ENTRIES(cList, ";"). //3
```

ENTRY - function - retourne une entrée spécifiée dans une liste

Comme d'habitude la position de départ est 1, pas 0!

ENTRÉE (entrée, liste [, délimiteur]).

```
DEFINE VARIABLE cList AS CHARACTER   NO-UNDO.

cList = "Goodbye,cruel,world!".

DISPLAY ENTRY(2, cList). //cruel
```

ENTRY - méthode - attribuer la valeur d'une entrée spécifiée dans une liste

ENTRY (entrée, liste [, délimiteur]) = valeur

```
DEFINE VARIABLE cList AS CHARACTER   NO-UNDO.

cList = "Goodbye,cruel,world!".

ENTRY(1, cList) = "Hello".
ENTRY(2, cList) = "nice".

MESSAGE REPLACE(cList, ",", " ") VIEW-AS ALERT-BOX. //Hello nice world!
```

LOOKUP - vérifie une liste pour une entrée spécifique. Renvoie son entrée.

Si la chaîne n'est pas présente dans la liste, la recherche renvoie 0

LOOKUP (chaîne, liste [, délimiteur])

```
DEFINE VARIABLE cList AS CHARACTER   NO-UNDO.

cList = "Hello,nice,world!".

MESSAGE LOOKUP("nice", cList) VIEW-AS ALERT-BOX. //2
MESSAGE LOOKUP("cruel", cList) VIEW-AS ALERT-BOX. //0
```

Caractères spéciaux (et échapper)

Dans Progress 4GL, la manière normale d'écrire un caractère spécial est de le précéder d'un caractère tilde (~).

Ce sont les caractères spéciaux par défaut

Séquence	Interprété comme	Commentaire
~ "	"	Utilisé pour écrire "à l'intérieur des chaînes définies en utilisant" string ".
~ '	'	Utilisé pour écrire 'les chaînes internes définies à l'aide de' chaîne '.
~~	~	Par exemple, si vous souhaitez imprimer la séquence et non comment elle est interprétée.
~ \	\	
~ {	{	{est utilisé dans les préprocesseurs et parfois il est nécessaire de s'échapper.
~ nnn	Un seul personnage	nnn est un nombre octal représentant la valeur ascii du caractère.
~ t	languette	
~ n	Nouvelle ligne / saut de ligne	
~ r	Retour de voiture	
~ E	Échapper	
~ b	Retour arrière	
~ f	Flux de formulaire	

Si vous voulez afficher le tilde du tout, il doit être échappé!

```
MESSAGE "A single tilde: ~~" VIEW-AS ALERT-BOX.

MESSAGE "At sign: ~100" SKIP
        "Tab~tseparated~twords!" SKIP
        "A linefeed:~n"
        "Escaping a quote sign: ~"This is a quote!~"" SKIP VIEW-AS ALERT-BOX.
```

Lire Cordes en ligne: https://riptutorial.com/fr/progress-4gl/topic/8872/cordes

Chapitre 4: Expressions conditionnelles

Introduction

Progress ABL prend en charge deux instructions conditionnelles: `IF/THEN/ELSE` et `CASE` .

Examples

IF ... THEN ... ELSE-statement

Dans l'instruction `IF THEN ELSE` , le résultat peut être soit une seule instruction:

```
DEFINE VARIABLE i AS INTEGER     NO-UNDO.

IF i = 0 THEN
    MESSAGE "Zero".
ELSE
    MESSAGE "Something else".
```

Ou un bloc, par exemple en ajoutant un bloc `DO` :

```
DEFINE VARIABLE i AS INTEGER     NO-UNDO.

IF i = 0 THEN DO:
    RUN procedure1.
    RUN procedure2.
END.
ELSE DO:
    RUN procedure3.
    RUN procedure4.
END.
```

Plusieurs états `IF` peuvent être imbriqués avec `ELSE IF` -syntax:

```
DEFINE VARIABLE i AS INTEGER     NO-UNDO.

IF i = 0 THEN DO:
    RUN procedure1.
    RUN procedure2.
END.
ELSE IF i = 1 THEN DO:
    RUN procedure3.
    RUN procedure4.

END.
ELSE DO:
    RUN procedure5.
    RUN procedure6.
END.
```

La partie `ELSE` n'est pas obligatoire:

```
DEFINE VARIABLE l AS LOGICAL     NO-UNDO.

l = TRUE.

IF l = TRUE THEN DO:
    MESSAGE "The l variable has the value TRUE" VIEW-AS ALERT-BOX.
END.
```

Le `IF` / `ELSE IF` peut comparer plusieurs conditions, avec ou sans connexions internes. Cela vous laisse libre de gâcher votre code de plusieurs manières:

```
DEFINE VARIABLE i AS INTEGER     NO-UNDO.
DEFINE VARIABLE l AS LOGICAL     NO-UNDO.

IF i < 30 OR l = TRUE THEN DO:

END.
ELSE IF i > 30 AND l = FALSE OR TODAY = DATE("2017-08-20") THEN DO:

END.
ELSE DO:
    MESSAGE "I dont really know what happened here".
END.
```

CAS

L'instruction `CASE` est beaucoup plus stricte que la condition `IF/ELSE` . Il ne peut comparer qu'une seule variable et seulement une égalité, pas une plus petite / plus grande que etc.

DEFINE VARIABLE c AS CARACTERE NON-UNDO.

```
CASE c:
    WHEN "A" THEN DO:
        RUN procedureA.
    END.
    WHEN "B" THEN DO:
        RUN procedureB.
    END.
    OTHERWISE DO:
        RUN procedureX.
    END.
END CASE.
```

En utilisant un `OR` chaque `WHEN` peut comparer différentes valeurs:

```
DEFINE VARIABLE c AS CHARACTER   NO-UNDO.

CASE c:
    WHEN "A" THEN DO:
        RUN procedureA.
    END.
    WHEN "B" OR WHEN "C" THEN DO:
        RUN procedureB-C.
    END.
    OTHERWISE DO:
```

```
        RUN procedureX.
    END.
END CASE.
```

Tout comme avec l'information `IF` chaque branche peut être une seule instruction ou un bloc. Tout comme avec l' `ELSE` -Déclaration, `OTHERWISE` est pas obligatoire.

```
DEFINE VARIABLE c AS CHARACTER   NO-UNDO.

CASE c:
    WHEN "A" THEN
        RUN procedureA.
    WHEN "B" OR WHEN "C" THEN
        RUN procedureB-C.
END CASE.
```

Contrairement à un `switch` style c, il n'est pas nécessaire d'échapper à l'instruction `CASE` - une seule branche sera exécutée. Si plusieurs correspond à `WHEN` , seul le premier se déclenchera. `OTHERWISE` doit être dernier et ne se déclenchera que si aucune des branches ci-dessus ne correspond.

```
DEFINE VARIABLE c AS CHARACTER   NO-UNDO.

c = "A".

CASE c:
    WHEN "A" THEN
        MESSAGE "A" VIEW-AS ALERT-BOX. //Only "A" will be messaged
    WHEN "A" OR WHEN "C" THEN
        MESSAGE "A or C" VIEW-AS ALERT-BOX.
END CASE.
```

IF ... THEN ... ELSE-function

`IF THEN ELSE` peut également être utilisé comme une fonction pour renvoyer une seule valeur. C'est beaucoup comme le ternaire `?` -opérateur de C.

```
DEFINE VARIABLE i AS INTEGER     NO-UNDO.
DEFINE VARIABLE c AS CHARACTER   NO-UNDO.

/* Set c to "low" if i is less than 5 otherwise set it to "high"
c = IF i < 5 THEN "low" ELSE "high".
```

L'utilisation de parenthèses peut faciliter la lisibilité du code comme celui-ci.

```
DEFINE VARIABLE i AS INTEGER     NO-UNDO.
DEFINE VARIABLE c AS CHARACTER   NO-UNDO.

c = (IF i < 5 THEN "low" ELSE "high").
```

La valeur de la partie `IF` et la valeur de la partie `ELSE` doivent être du même type. Il n'est pas possible d'utiliser `ELSE IF` dans ce cas.

```
DEFINE VARIABLE dat                AS DATE        NO-UNDO.
DEFINE VARIABLE beforeTheFifth     AS LOGICAL   NO-UNDO.

dat = TODAY.

beforeTheFifth = (IF DAY(dat) < 5 THEN TRUE ELSE FALSE).
```

Plusieurs comparaisons peuvent être effectuées dans le `IF` :

```
DEFINE VARIABLE between5and10 AS LOGICAL     NO-UNDO.
DEFINE VARIABLE i             AS INTEGER     NO-UNDO INIT 7.

between5and10 = (IF i >= 5 AND i <= 10 THEN TRUE ELSE FALSE).

MESSAGE between5and10 VIEW-AS ALERT-BOX.
```

Lire Expressions conditionnelles en ligne: https://riptutorial.com/fr/progress-4gl/topic/8904/expressions-conditionnelles

Chapitre 5: Itérer

Introduction

Il existe plusieurs manières d'itérer (boucler) dans Progress ABL.

Examples

FAIRE PENDANT

Une boucle `DO WHILE` continuera à boucler sauf si la partie `WHILE` est satisfaite. Cela le rend facile à exécuter pour toujours et à manger à tout moment à partir d'un cœur de processeur.

> FAIRE TOUTE *expression* :
>
> FIN.
>
> *expression* est une combinaison de logique booléenne, de comparaisons, de variables, de champs, etc., évaluée à une valeur vraie.

```
/* This is a well defined DO WHILE loop that will run as long as i is lower than 10*/
DEFINE VARIABLE i AS INTEGER     NO-UNDO.
DO WHILE i < 10:
    i = i + 1.
END.

DISPLAY i. // Will display 10
```

Vous pouvez utiliser n'importe quel nombre de contrôles dans la partie `WHILE` :

```
DEFINE VARIABLE i AS INTEGER     NO-UNDO.
DO WHILE TODAY = DATE("2017-02-06") AND RANDOM(1,100) < 99:
    i = i + 1.
END.

MESSAGE i "iterations done" VIEW-AS ALERT-BOX.
```

Cependant, le compilateur ne vous aidera pas à vérifier que la partie `WHILE` est finalement remplie:

```
/* Oops. Didnt increase i. This will run forever... */
DEFINE VARIABLE i AS INTEGER     NO-UNDO.
DO WHILE i < 10:
    i = 1.
END.
```

DO var = commencer pour finir [BY step]

Cette itération modifie une valeur d'un point de départ à une fin, éventuellement par une valeur

spécifiée pour chaque étape. Le changement par défaut est 1.

```
DEFINE VARIABLE i AS INTEGER     NO-UNDO.

DO i = 10 TO 15:
    DISPLAY i WITH FRAME x1 6 DOWN .
    DOWN WITH FRAME x1.
END.
```

Résultat:

```
---------------i

        10
        11
        12
        13
        14
        15
```

Vous pouvez également parcourir des dates:

```
DEFINE VARIABLE dat AS INTEGER        NO-UNDO.

DO dat = TODAY TO TODAY + 3:

END.
```

Et sur les décimales. Mais alors vous voudrez probablement utiliser `BY` - sinon un `INTEGER` aurait fait aussi bien ...

```
DEFINE VARIABLE de AS DECIMAL     NO-UNDO.

DO de = 1.8 TO 2.6 BY 0.2:
    DISPLAY "Value" de.
END.
```

En utilisant `BY` un nombre négatif , vous pouvez également passer d'un plus à une valeur inférieure:

```
DEFINE VARIABLE i AS INTEGER     NO-UNDO.

DO i = 5 TO 1 BY -1:

END.
```

L'expression sera testée jusqu'à ce qu'elle ne soit plus remplie. Cela rend le compteur plus haut (si vous vous déplacez vers le haut) ou plus bas (si vous vous déplacez vers le bas) une fois la boucle terminée:

```
DEFINE VARIABLE i AS INTEGER     NO-UNDO.

DO i = 5 TO 1 BY -1:
```

```
END.

MESSAGE i. // Will message 0
```

Un autre exemple:

```
DEFINE VARIABLE da AS DATE     NO-UNDO.

DISPLAY TODAY. //17/02/06
DO da = TODAY TO TODAY + 10:

END.
DISPLAY da. //17/02/17 (TODAY + 11)
```

RÉPÉTER

REPEAT, se répète pour toujours sauf si vous quittez explicitement la boucle:

```
//Runs forever
REPEAT:
    // Do stuff
END.
```

Pour quitter la boucle, vous pouvez utiliser le mot-clé `LEAVE` . Avec ou sans étiquette. Sans étiquette, `LEAVE` affectera toujours la boucle en cours. Avec un nom, vous pouvez spécifier la boucle à `LEAVE` .

```
/* Without a label */
REPEAT:
  //Do stuff
  IF TRUE THEN LEAVE.
END.

/* With a label */
loopLabel:
REPEAT:
  //Do stuff
  IF <somecondition> THEN LEAVE loopLabel.
END.

/* Two nested REPEATS */
DEFINE VARIABLE i AS INTEGER     NO-UNDO.
loopLabelOne:
REPEAT:
    loopLabelTwo:
    REPEAT:
        i = i + 1.
        IF RANDOM(1,100) = 1  THEN LEAVE loopLabelTwo.
        IF RANDOM(1,100) = 1  THEN LEAVE loopLabelOne.
    END.
    IF RANDOM(1,100) = 1  THEN LEAVE loopLabelOne.
END.
DISPLAY i.
```

Lire Itérer en ligne: https://riptutorial.com/fr/progress-4gl/topic/9009/iterer

Chapitre 6: Les fonctions

Introduction

Une fonction définie par l'utilisateur dans Progress ABL est un module de programme réutilisable.

Remarques

- Une fonction doit être déclarée dans la procédure "principale". Il ne peut pas être déclaré dans une procédure ou dans une autre fonction.
- Une fonction en cours ABL n'est pas un "citoyen de première classe" contrairement aux langages de programmation comme Haskell ou Javascript. Vous ne pouvez pas transmettre une fonction en tant que paramètre d'entrée ou de sortie. Vous pouvez cependant les appeler dynamiquement en utilisant `DYNAMIC-FUNCTION` ou l'objet `CALL` .
- L'appel de fonctions dans vos requêtes peut entraîner de mauvaises performances, car la correspondance avec les index est préjudiciable. Essayez d'affecter la valeur de la fonction à une variable et utilisez cette variable dans la `WHERE` WHERE à la place.

Examples

Fonction simple

```
/* This function returns TRUE if input is the letter "b" and false otherwise */
FUNCTION isTheLetterB RETURNS LOGICAL (INPUT pcString AS CHARACTER):
  IF pcString = "B" THEN
    RETURN TRUE.
  ELSE
    RETURN FALSE.
END FUNCTION.

/* Calling the function with "b" as input - TRUE expected */
DISPLAY isTheLetterB("b").

/* Calling the function with "r" as input - FALSE expected */
DISPLAY isTheLetterB("r").
```

Certaines parties de la syntaxe ne sont en fait pas nécessaires:

```
/* RETURNS isn't required, INPUT isn't required on INPUT-parameters */
FUNCTION isTheLetterB LOGICAL (pcString AS CHARACTER):
  IF pcString = "B" THEN
    RETURN TRUE.
  ELSE
    RETURN FALSE.
/* END FUNCTION can be replaced with END */
END.
```

Fonctions de déclaration anticipée

Une fonction peut être déclarée en avant, ceci est similaire aux spécifications d'un fichier d'en-tête C. De cette manière, le compilateur sait qu'une fonction sera disponible ultérieurement.

Sans déclarations directes, la fonction DOIT être déclarée avant d'être appelée dans le code. La déclaration forward comprend la spécification `FUNCTION` (nom de la fonction, type de retour et types de données et ordre des paramètres). Si la déclaration forward ne correspond pas à la fonction réelle, le compilateur produira des erreurs et le code ne pourra pas s'exécuter.

```
FUNCTION dividableByThree LOGICAL (piNumber AS INTEGER) FORWARD.

DISPLAY dividableByThree(9).

FUNCTION dividableByThree LOGICAL (piNumber AS INTEGER):

    IF piNumber MODULO 3 = 0 THEN
        RETURN TRUE.
    ELSE
        RETURN FALSE.
END.
```

Paramètres d'entrée multiples

/ * Cela va faire apparaître une boîte de message disant "HELLO WORLD" * /

```
FUNCTION cat RETURNS CHARACTER ( c AS CHARACTER, d AS CHARACTER):

    RETURN c + " " + d.

END.

MESSAGE cat("HELLO", "WORLD") VIEW-AS ALERT-BOX.
```

Instructions de retour multiples (mais une seule valeur de retour)

Une fonction peut avoir plusieurs déclarations de retour et elles peuvent être placées dans différentes parties de la fonction réelle. Cependant, ils doivent tous retourner le même type de données.

```
FUNCTION returning DATE ( dat AS DATE):
    IF dat < TODAY THEN DO:
        DISPLAY "<".
        RETURN dat - 200.
    END.
    ELSE DO:
        DISPLAY ">".
        RETURN TODAY.
    END.
END.

MESSAGE returning(TODAY + RANDOM(-50, 50)) VIEW-AS ALERT-BOX.
```

Une fonction n'a pas besoin de retourner quoi que ce soit. Alors, la valeur de retour sera? (inconnu). Le compilateur n'attrapera pas cela (mais vos collègues l'éviteront donc).

```
/* This function will only return TRUE or ?, never FALSE, so it might lead to troubles */
FUNCTION inTheFuture LOGICAL ( dat AS DATE):
    IF dat > TODAY THEN DO:
        RETURN TRUE.
    END.
END.
MESSAGE inTheFuture(TODAY + RANDOM(-50, 50)) VIEW-AS ALERT-BOX.
```

Paramètres de sortie et d'entrée-sortie

Une fonction ne peut renvoyer qu'une seule valeur mais il y a un moyen de contourner cela: les paramètres ne sont pas limités aux paramètres d'entrée. Vous pouvez déclarer les paramètres `INPUT` , `OUTPUT` et `INPUT-OUTPUT` .

Contrairement aux paramètres `INPUT` , vous devez spécifier `OUTPUT` ou `INPUT-OUTPUT` avant les paramètres.

Certaines conventions de codage peuvent ne pas aimer cela, mais cela peut être fait.

```
/* Function will add numbers and return a sum (AddSomSumbers(6) = 6 + 5 + 4 + 3 + 2 + 1 = 21
*/
/* It will also have a 1% per iteration of failing
*/
/* To handle that possibility we will have a status output parameter
*/
FUNCTION AddSomeNumbers INTEGER ( INPUT number AS INTEGER, OUTPUT procstatus AS CHARACTER):

    procStatus = "processing".

    DEFINE VARIABLE i AS INTEGER     NO-UNDO.
    DEFINE VARIABLE n AS INTEGER     NO-UNDO.
    /* Iterate number times */
    DO i = 1 TO number:
        /* Do something */

        n = n + i.

        /* Fake a 1% chance for an error that breaks the function */
        IF RANDOM(1,100) = 1 THEN
            RETURN 0.
    END.

    procStatus = "done".
    RETURN n.
END.


DEFINE VARIABLE ret   AS INTEGER     NO-UNDO.
DEFINE VARIABLE stat  AS CHARACTER   NO-UNDO.

/* Call the function */
ret = AddSomeNumbers(30, OUTPUT stat).

/* If "stat" is done we made it! */
IF stat = "done" THEN DO:
    MESSAGE "We did it! Sum:" ret VIEW-AS ALERT-BOX.
END.
```

```
ELSE DO:
    MESSAGE "An error occured" VIEW-AS ALERT-BOX ERROR.
END.
```

Voici un exemple de paramètre `INPUT-OUTPUT` :

```
/* Function doubles a string and returns the length of the new string */
FUNCTION doubleString RETURN INTEGER (INPUT-OUTPUT str AS CHARACTER).

    str = str + str.

    RETURN LENGTH(str).

END.

DEFINE VARIABLE str AS CHARACTER   NO-UNDO.
DEFINE VARIABLE len AS INTEGER     NO-UNDO.

str = "HELLO".

len = doubleString(INPUT-OUTPUT str).

MESSAGE
    "New string: " str SKIP
    "Length: " len VIEW-AS ALERT-BOX.
```

Récursivité

Voir récursivité

Une fonction peut s'appeler elle-même et, par conséquent, se rétablir.

```
FUNCTION factorial INTEGER (num AS INTEGER).

    IF num = 1 THEN
        RETURN 1.
    ELSE
        RETURN num * factorial(num - 1).

END FUNCTION.

DISPLAY factorial(5).
```

Avec les paramètres standard (paramètre de démarrage), la session Progress ne sera pas capable de gérer des nombres très importants dans cet exemple. `factorial(200)` remplira la pile et générera une erreur.

Appel dynamique d'une fonction

En utilisant `DYNAMIC-FUNCTION` ou `CALL` -object, vous pouvez appeler dynamiquement des fonctions.

```
DEFINE VARIABLE posY      AS INTEGER     NO-UNDO.
DEFINE VARIABLE posX      AS INTEGER     NO-UNDO.
DEFINE VARIABLE OKkeys    AS CHARACTER   NO-UNDO INIT "QLDRUS".
```

```
DEFINE VARIABLE Step      AS INTEGER     NO-UNDO INIT 1.
DEFINE VARIABLE moved     AS LOGICAL     NO-UNDO.
/* Set original position */
posY = 10.
posX = 10.

/* Move up (y coordinates - steps ) */
FUNCTION moveU LOGICAL (INPUT steps AS INTEGER):

    IF posY = 0 THEN
        RETURN FALSE.

    posY = posY - steps.

    IF posY < 0 THEN
        posY = 0.

    RETURN TRUE.
END FUNCTION.

/* Move down (y coordinates + steps ) */
FUNCTION moveD LOGICAL (INPUT steps AS INTEGER):

    IF posY = 20 THEN
        RETURN FALSE.

    posY = posY + steps.

    IF posY > 20 THEN
        posY = 20.

END FUNCTION.

/* Move left (x coordinates - steps ) */
FUNCTION moveL LOGICAL (INPUT steps AS INTEGER):

    IF posX = 0 THEN
        RETURN FALSE.

    posX = posX - steps.

    IF posX < 0 THEN
        posX = 0.

    RETURN TRUE.
END FUNCTION.

/* Move down (x coordinates + steps ) */
FUNCTION moveR LOGICAL (INPUT steps AS INTEGER):

    IF posX = 20 THEN
        RETURN FALSE.

    posX = posX + steps.

    IF posX > 20 THEN
        posX = 20.

END FUNCTION.
```

```
REPEAT:

    DISPLAY posX posY step WITH FRAME x1 1 DOWN.
    READKEY.

    IF INDEX(OKKeys, CHR(LASTKEY)) <> 0 THEN DO:
        IF CHR(LASTKEY) = "q"  THEN LEAVE.
        IF CAPS(CHR(LASTKEY)) = "s" THEN UPDATE step WITH FRAME x1.
        ELSE DO:
            moved = DYNAMIC-FUNCTION("move" + CAPS(CHR(LASTKEY)), INPUT step).
            IF moved = FALSE THEN
                MESSAGE "Out of bounds".
        END.
    END.
END.
```

L'objet `CALL` n'est pas aussi léger que la DYNAMIC-FUNCTION. Il peut être utilisé pour appeler différentes choses: fonctions, procédures, programme externe, fonctions DLL Windows. Il peut également invoquer des méthodes sur des objets et accéder à des getters / setters.

```
DEFINE VARIABLE functionHandle AS HANDLE      NO-UNDO.
DEFINE VARIABLE returnvalue    AS CHARACTER   NO-UNDO.

FUNCTION isPalindrome LOGICAL (INPUT txt AS CHARACTER, OUTPUT txtBackwards AS CHARACTER):
    DEFINE VARIABLE i AS INTEGER     NO-UNDO.

    DO i = LENGTH(txt) TO 1 BY -1:
        txtBackwards = txtBackwards + SUBSTRING(txt, i, 1).
    END.

    IF txt = txtBackwards THEN
        RETURN TRUE.
    ELSE
        RETURN FALSE.

END FUNCTION.

CREATE CALL functionHandle.
functionHandle:CALL-NAME      = "isPalindrome".
/* Sets CALL-TYPE to the default */
functionHandle:CALL-TYPE  = FUNCTION-CALL-TYPE.
functionHandle:NUM-PARAMETERS = 2.
functionHandle:SET-PARAMETER(1, "CHARACTER", "INPUT", "HELLO WORLD").
functionHandle:SET-PARAMETER(2, "CHARACTER", "OUTPUT", returnvalue).
functionHandle:INVOKE.

MESSAGE "Text backwards: " returnvalue "Is it a palindrome? " functionHandle:RETURN-VALUE
VIEW-AS ALERT-BOX.

DELETE OBJECT functionHandle.


CREATE CALL functionHandle.
functionHandle:CALL-NAME      = "isPalindrome".
/* Sets CALL-TYPE to the default */
functionHandle:CALL-TYPE  = FUNCTION-CALL-TYPE.
functionHandle:NUM-PARAMETERS = 2.
functionHandle:SET-PARAMETER(1, "CHARACTER", "INPUT", "ANNA").
functionHandle:SET-PARAMETER(2, "CHARACTER", "OUTPUT", returnvalue).
```

```
functionHandle:INVOKE.

MESSAGE "Text backwards: " returnvalue "Is it a palindrome? " functionHandle:RETURN-VALUE
VIEW-AS ALERT-BOX.

DELETE OBJECT functionHandle.
```

Lire Les fonctions en ligne: https://riptutorial.com/fr/progress-4gl/topic/8857/les-fonctions

Chapitre 7: Les variables

Introduction

Progress ABL est typé statiquement. Les variables doivent être déclarées et le type de données ne peut pas être modifié pendant l'exécution.

Syntaxe

- DÉFINIR VARIABLE COMME INT64 INITIAL -200 NO-UNDO. // Un entier de 64 bits initialisé à -200
- DÉFINIR VARIABLE L AS NON LOGIQUE. // Une variable logique nommée l
- DEFINIR VARIABLE c COMME CARACTERE SANS AUCUN CAS. // Une variable sensible à la casse ('a' <> 'A').
- DÉFINIR VARIABLE dt DATE INTIALE AUJOURD'HUI NON-UNDO. // Une variable de date définie sur la date du jour.
- DÉFINIR VARIABLE ET COMME CARACTÈRE EXTENT 5 NO-UNDO. // Un tableau de caractères avec une longueur = 5
- DÉFINIR VARIABLE j COMME INTÉRIEUR EXTENT NO-UNDO. // une étendue sans longueur définie
- DEFINE VARIABLE b AS DATETIME LABEL "Heure de départ". // Une variable avec une étiquette

Examples

Déclarations de variables de base

```
/*

These variables are declared with `NO-UNDO`.
That states that no undo handling is wanted for this specific variable
in case of a transactional roll-back.

This should always be the default unless transactional control over
this variable is a requirement.
*/

/* Strings. A character longer than 32K should be a longchar */
DEFINE VARIABLE c   AS CHARACTER   NO-UNDO.
DEFINE VARIABLE cl  AS LONGCHAR    NO-UNDO.

/* Integers and decimals. INTEGER = 32 bit. INT64 = 64 bits */
DEFINE VARIABLE i   AS INTEGER     NO-UNDO.
```

```
DEFINE VARIABLE j   AS INT64       NO-UNDO.
DEFINE VARIABLE k   AS DECIMAL     NO-UNDO.

/* Date and datetimez. Unset variables have the unknown value ? */
DEFINE VARIABLE d   AS DATE        NO-UNDO.
DEFINE VARIABLE dt  AS DATETIME    NO-UNDO.
DEFINE VARIABLE dtz AS DATETIME-TZ NO-UNDO.

/* LOGICAL = Boolean data. True or false (or ?) */
DEFINE VARIABLE l   AS LOGICAL     NO-UNDO.

/* Rowids and recids are internal identifiers to database records */
DEFINE VARIABLE rid AS ROWID       NO-UNDO.
DEFINE VARIABLE rec AS RECID       NO-UNDO.

/* A handle is a handle to anything: a session, an on screen widget etc */
/* A Com-handle is used for ActiveX Com-automation */
DEFINE VARIABLE h   AS HANDLE      NO-UNDO.
DEFINE VARIABLE hc  AS COM-HANDLE  NO-UNDO.

/* A raw variable can contain any data. Binary, strings etc */
DEFINE VARIABLE rw  AS RAW         NO-UNDO.

/* A mempointer contains a sequence of bytes in memory. */
DEFINE VARIABLE m   AS MEMPTR      NO-UNDO.
```

Tableaux - définition et accès

Progress prend en charge les tableaux à une dimension, mais ils sont appelés `EXTENTS` .

```
/* Define a character array with the length 5, and display it's length */
DEFINE VARIABLE a AS CHARACTER EXTENT 5 NO-UNDO.
DISPLAY EXTENT(a).
```

Positions individuelles i le tableau est accessible en utilisant des crochets de style c "standard". Mais l'index commence à 1. La taille maximale est 28000.

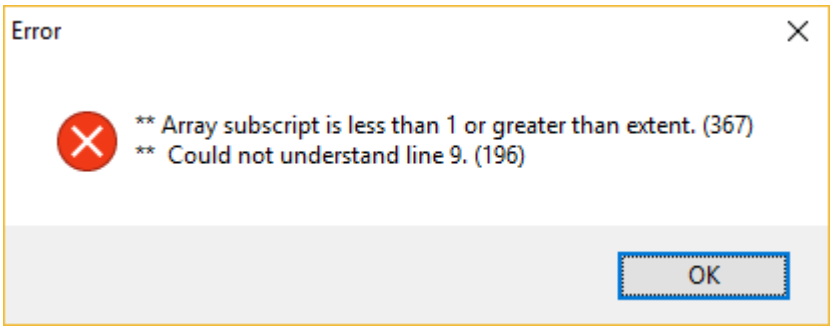
```
a[1] = "A".
a[2] = "B".
a[3] = "C".
a[4] = "D".
a[5] = "E".

DISPLAY a[5].
```

Résultat:

L'index 0 va générer une erreur:

```
DISPLAY a[0].
```

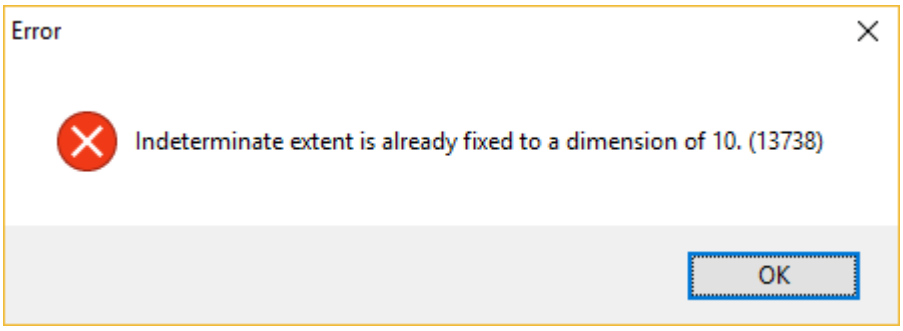
Résultat:



Vous pouvez également définir un tableau indéterminé sans longueur définie. La longueur (étendue) peut être définie au moment de l'exécution. Mais seulement une fois!

```
DEFINE VARIABLE a AS CHARACTER EXTENT NO-UNDO.
EXTENT(a) = 10.
EXTENT(a) = 1.
```

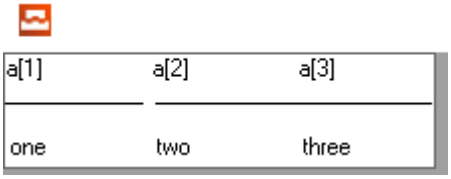
La troisième ligne comportera l'erreur suivante:



Vous pouvez utiliser l'option `INITIAL` de l'instruction `DEFINE VARIABLE` pour définir les valeurs initiales.

```
DEFINE VARIABLE a AS CHARACTER EXTENT 3 INITIAL ["one","two","three"] NO-UNDO.
/* Some statements (like DISPLAY) can handle a whole array: */
DISPLAY a.
```

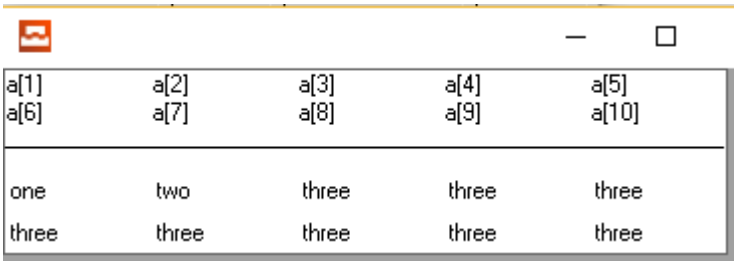
Résultat:



Si vous ne définissez pas toutes les extensions, la dernière valeur définie sera la suivante:

```
DEFINE VARIABLE a AS CHARACTER EXTENT 10 INITIAL ["one","two","three"] NO-UNDO.
DISPLAY a.
```

Résultat:



Utilisation du mot clé LIKE

En utilisant `LIKE` vous pouvez baser la définition de votre variable sur une autre variable ou un champ dans une base de données ou une table temporaire.

Définir une variable `LIKE` un champ de base de données nécessite que la base de données soit toujours connectée. Ce n'est peut-être pas toujours ce que vous voulez.

```
DEFINE VARIABLE i AS INTEGER NO-UNDO LABEL "Nr" FORMAT "99999".
/* Define a variable with the same properties as "i" */
DEFINE VARIABLE j LIKE i.

/* Define a variable based on Customer.Custnum from the sports2000 database but
override the label-definition */
DEFINE VARIABLE k LIKE Customer.Custnum LABEL "Client".
```

Lire Les variables en ligne: https://riptutorial.com/fr/progress-4gl/topic/8800/les-variables

Chapitre 8: Procédures

Introduction

Il existe deux types de procédures dans Progress ABL: les procédures internes et les prototypes de procédures qui sont des façades pour les DLL Windows ou les procédures de bibliothèque partagée Unix / Linux.

Tout comme avec les fonctions, les procédures ne peuvent pas être imbriquées. Vous ne pouvez pas imbriquer des fonctions dans les procédures et inversement.

Une procédure est appelée avec l'instruction `RUN` .

Syntaxe

- Nom de procédure RUN. // Exécute une procédure appelée nom-procédure.
- RUN proc1 (INPUT "HELLO"). // Saisit la chaîne HELLO pour proc1
- RUN proc2 (INPUT var1, sortie var2). // Entrée var1 et sorties var2 vers / depuis proc2
- RUN proc3 (entrée "name = 'joe'", OUTPUT TABLE ttResult). // Entrée nom = joe et produit des enregistrements dans une table
- PROCEDURE proc: // Déclare une procédure nommée proc
- PROCÉDURE DE FIN. // Termine la procédure en cours

Examples

Une procédure interne de base

Contrairement aux fonctions, il n'est pas nécessaire de transmettre une procédure de déclaration. Il peut être placé n'importe où dans votre code, avant ou après l'appel via `RUN` .

```
RUN proc.

//Procedure starts here
PROCEDURE proc:

//Procedure ends here
END PROCEDURE.
```

Le nom de la procédure est suivi d'un signe deux-points nous indiquant qu'il s'agit du début d'un bloc. Le bloc se termine par `END PROCEDURE.` (mais cela peut être remplacé par simplement `END.`).

Paramètres INPUT et OUTPUT

Une procédure peut avoir des paramètres de différents types: entrée, sortie, entrée-sortie (bidirectionnelle) et certains types spéciaux tels que les tables de temp-temp et les jeux de données).

Dans l'instruction run, il est facultatif de déclarer `INPUT` (il est considéré comme par défaut) - toutes les autres directions doivent être déclarées spécifiquement.

Une procédure prenant deux entiers en entrée et produisant une décimale.

```
PROCEDURE divideAbyB:
    DEFINE INPUT  PARAMETER piA       AS INTEGER     NO-UNDO.
    DEFINE INPUT  PARAMETER piB       AS INTEGER     NO-UNDO.
    DEFINE OUTPUT PARAMETER pdeResult AS DECIMAL     NO-UNDO.

    pdeResult = piA / piB.

END PROCEDURE.

DEFINE VARIABLE de AS DECIMAL     NO-UNDO.

RUN divideAbyB(10, 2, OUTPUT de).

DISPLAY de. //5.00
```

Les paramètres sont totalement optionnels. Vous pouvez mélanger et assortir comme vous le souhaitez. L'ordre des paramètres dépend de vous, mais il est utile de commencer par une entrée et de terminer par une sortie - vous devez les placer dans le bon ordre dans l'instruction d'exécution et les directions de mixage peuvent être gênantes.

Récursivité - voir récursivité

La récursivité est facile - `RUN` la procédure elle-même depuis l'intérieur de la procédure. Cependant, si vous recentrez trop loin, la pile manquera d'espace.

Un calcul de procédure de la factorielle.

```
PROCEDURE factorial:
    DEFINE INPUT  PARAMETER piNum AS INTEGER     NO-UNDO.
    DEFINE OUTPUT PARAMETER piFac AS INTEGER     NO-UNDO.

    DEFINE VARIABLE iFac AS INTEGER     NO-UNDO.

    IF piNum = 1 THEN DO:
        pifac = 1.
    END.
    ELSE DO:
        RUN factorial(piNum - 1, OUTPUT iFac).
        piFac = piNum * iFac.
    END.

END PROCEDURE.

DEFINE VARIABLE f AS INTEGER     NO-UNDO.

RUN factorial(7, OUTPUT f).
```

```
DISPLAY f.
```

Portée

La procédure a sa propre portée. La portée extérieure "saigne" dans la procédure mais pas dans l'autre sens.

```
DEFINE VARIABLE i AS INTEGER     NO-UNDO INIT 1.
DEFINE VARIABLE j AS INTEGER     NO-UNDO.

PROCEDURE p:

    MESSAGE i VIEW-AS ALERT-BOX. // 1
    MESSAGE j VIEW-AS ALERT-BOX. // 0

    j = 2.

END PROCEDURE.

RUN p.

MESSAGE i VIEW-AS ALERT-BOX. // 1
MESSAGE j VIEW-AS ALERT-BOX. // 2
```

Déclarer une variable dans une procédure qui porte le même nom qu'un paramètre à l'extérieur ne créera qu'une variable locale.

```
DEFINE VARIABLE i AS INTEGER     NO-UNDO INIT 1.
DEFINE VARIABLE j AS INTEGER     NO-UNDO.

PROCEDURE p:

    DEFINE VARIABLE i AS INTEGER     NO-UNDO INIT 5.

    MESSAGE i VIEW-AS ALERT-BOX. // 5
    MESSAGE j VIEW-AS ALERT-BOX. // 0

    j = 2.

END PROCEDURE.

RUN p.

MESSAGE i VIEW-AS ALERT-BOX. // 1
MESSAGE j VIEW-AS ALERT-BOX. // 2
```

Toute variable créée à l'intérieur d'une procédure est accessible à cette procédure uniquement.

Cela générera une erreur de compilation:

```
PROCEDURE p:

    DEFINE VARIABLE i AS INTEGER     NO-UNDO INIT 5.
```

```
END PROCEDURE.

RUN p.

MESSAGE i VIEW-AS ALERT-BOX. // Unknown Field or Variable name i - error 201
```

Lire Procédures en ligne: https://riptutorial.com/fr/progress-4gl/topic/8914/procedures

Chapitre 9: Requêtes

Introduction

Les exemples seront basés sur une copie de la base de données de démonstration `Sports 2000` fournie avec la configuration de Progress.

Lorsque vous travaillez avec des requêtes en cours, vous devez:

`DEFINE` la requête et définir les tampons (tables) et les champs sur lesquels elle fonctionne.

`OPEN` la requête avec une `WHERE` spécifique qui définit comment récupérer les enregistrements. Peut-être aussi trier (`BY` / `BREAK BY`)

`GET` les données réelles - qui peuvent être `FIRST` , `NEXT` , `PREV` (pour précédent) ou `LAST` correspondant.

Syntaxe

- DEFINE QUERY nom-requête FOR nom-tampon. // Définition générale de la requête pour un tampon
- DEFINE QUERY nom-requête FOR nom-tampon1, nom-tampon2. // Rejoindre deux tampons
- DEFINE QUERY nom-requête FOR nom-tampon FIELDS (champ1 champ2). // Ne récupère que field1 et field2
- DEFINE QUERY nom-requête FOR nom-tampon EXCEPT (field3). // Récupère tous les champs sauf field3.

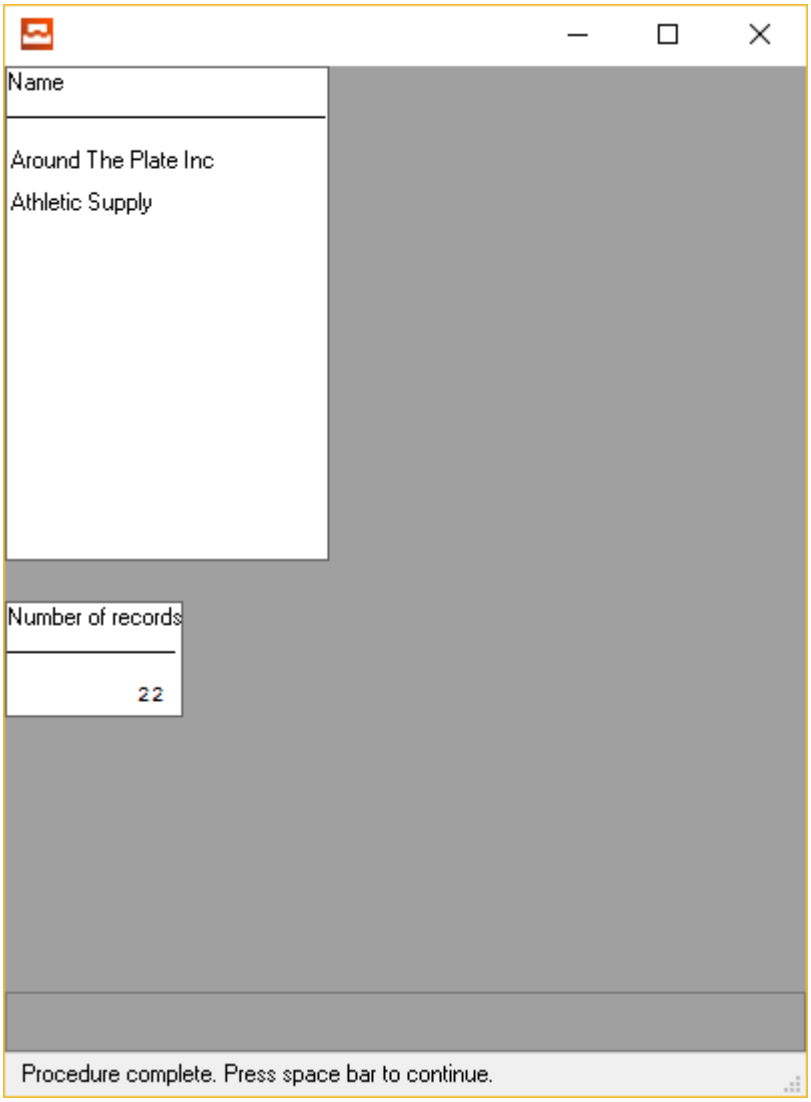
Examples

Requête de base

```
/* Define a query named q1 for the Customer table */
DEFINE QUERY q1 FOR Customer.
/* Open the query for all Customer records where the state is "tx" */
OPEN QUERY q1 FOR EACH Customer WHERE Customer.state ='TX'.

/* Get the first result of query q1 */
GET FIRST q1.


/* Repeat as long as query q1 has a result */
DO WHILE NOT QUERY-OFF-END('q1'):
    /* Display Customer.Name in a frame called frame1 with 10 rows */
    DISPLAY Customer.Name WITH FRAME frame1 10 DOWN.
    /* Move down the target line where to display the next record */
    DOWN WITH FRAME frame1.
    /* Get the next result of query q1 */
```

```
    GET NEXT q1.
END.
/* Display how many results query q1 had. */
DISPLAY NUM-RESULTS('q1') LABEL "Number of records".

/* Close the query */
CLOSE QUERY q1.
```

Sortie (troisième écran sous Windows gui):



Requête multi-tables

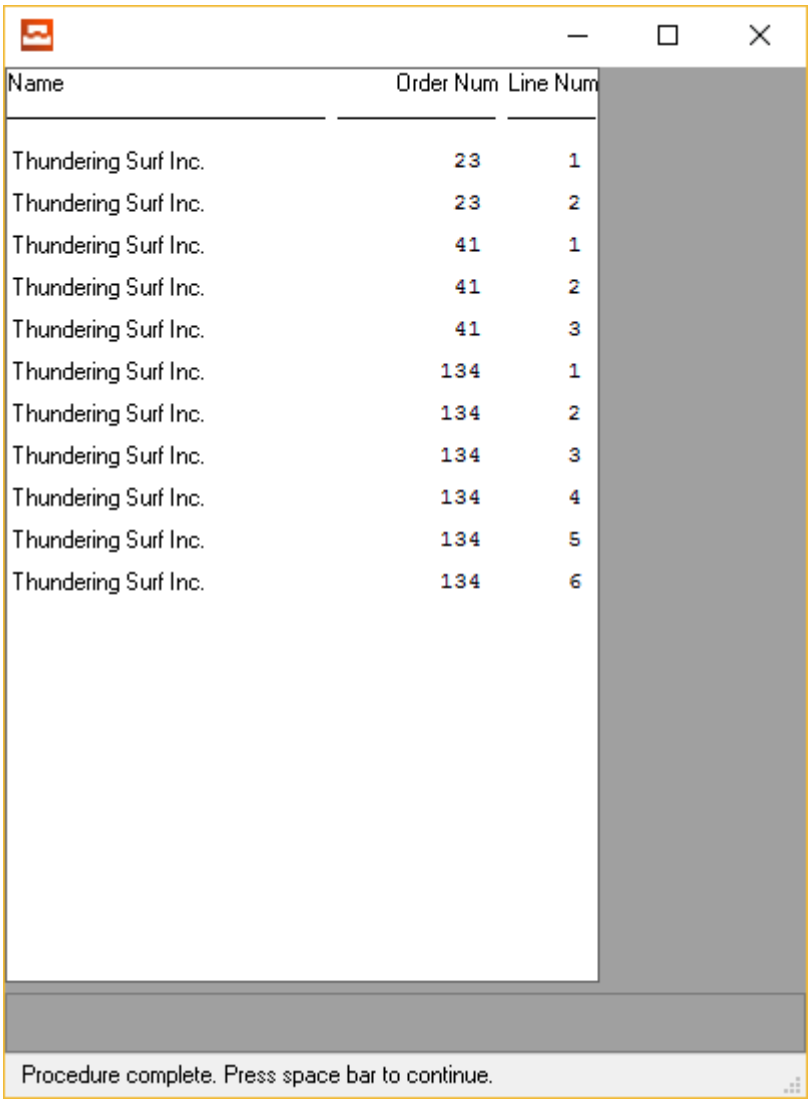
Cette requête joint trois tables: Client, Commande et Ligne de commande.

L'utilisation de l'instruction `OF` comme dans `childtable OF parenttable` suppose que les index sont construits de manière spécifique. C'est le cas dans la base de données sports2000.

```
DEFINE QUERY q1 FOR Customer, Order, Orderline.

OPEN QUERY q1 FOR EACH Customer WHERE Customer.state = 'TX'
    , EACH Order OF customer WHERE order.custnum < 1000
    , EACH orderline OF order.
```

```
GET FIRST q1.
DO WHILE NOT QUERY-OFF-END('q1'):
    DISPLAY Customer.Name Order.OrderNum OrderLine.LineNum
        WITH FRAME frameA 20 DOWN.
    DOWN WITH FRAME frameA.
    GET NEXT q1.
END.

CLOSE QUERY q1.
```

Résultat: dans l'interface graphique de Windows:



Déplacement d'une empreinte avec une requête utilisant next, first, prev et last

```
DEFINE QUERY q1 FOR Customer.

OPEN QUERY q1 FOR EACH Customer.

GET FIRST q1.

loop:
REPEAT:
```

```
    IF AVAILABLE Customer THEN DO:
        DISPLAY Customer.NAME CustNum WITH FRAME frClient TITLE "Client data".

        DISPLAY
            "(P)revious" SKIP
            "(N)ext" SKIP
            "(F)irst" SKIP
            "(L)ast" SKIP
            "(Q)uit" SKIP
            WITH FRAME frInstr
                TITLE "Instructions".
    END.

    READKEY.

    IF LASTKEY = ASC("q") THEN LEAVE loop.
    ELSE IF LASTKEY = ASC("n") THEN
        GET NEXT q1.
    ELSE IF LASTKEY = ASC("p") THEN
        GET PREV q1.
    ELSE IF LASTKEY = ASC("l") THEN
        GET LAST q1.
    ELSE IF LASTKEY = ASC("f") THEN
        GET FIRST q1.

END.

MESSAGE "Bye" VIEW-AS ALERT-BOX.
```

Lire Requêtes en ligne: https://riptutorial.com/fr/progress-4gl/topic/8694/requetes

Chapitre 10: TEMP-TABLE

Introduction

Le `TEMP-TABLE` est une fonctionnalité très puissante de Progress ABL. C'est une table temporaire en mémoire (surtout du moins) qui peut être utilisée pour écrire une logique complexe. Il peut être utilisé comme paramètre d'entrée / sortie pour des procédures, des fonctions et d'autres programmes. Une ou plusieurs tables temporaires peuvent constituer la base d'un `DATASET` (souvent appelé ProDataset).

Presque tout ce qui peut être fait avec une table de base de données Progress native peut être fait avec une table temporaire.

Examples

Définir une table temporaire simple

C'est la définition d'un `TEMP-TABLE` nommé ttTempTable avec trois champs. `NO-UNDO` indique qu'aucune gestion d'annulation n'est nécessaire (c'est généralement ce que vous voulez faire, sauf si vous avez vraiment besoin du contraire).

```
DEFINE TEMP-TABLE ttTempTable NO-UNDO
    FIELD field1 AS INTEGER
    FIELD field2 AS CHARACTER
    FIELD field3 AS LOGICAL.
```

Une table temporaire avec un index

Les tables temporaires peuvent (et doivent) être créées avec des index si vous prévoyez d'exécuter des requêtes sur ces dernières.

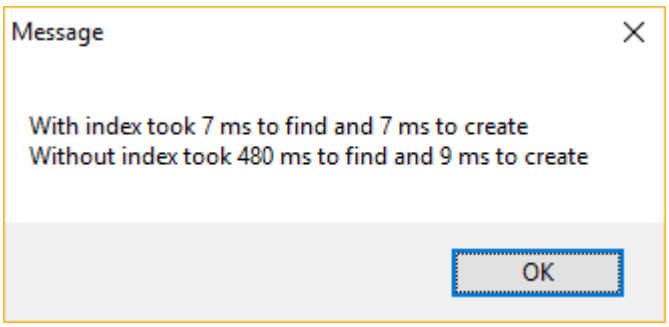
Cette table a un index (index1) contenant un champ (field1). Cet index est primaire et unique (ce qui signifie que deux enregistrements ne peuvent pas avoir le même contenu que field1).

```
DEFINE TEMP-TABLE ttTempTable NO-UNDO
    FIELD field1 AS INTEGER
    FIELD field2 AS CHARACTER
    FIELD field3 AS LOGICAL
    INDEX index1 IS PRIMARY UNIQUE field1 .
```

Plus d'index - index ...

Vous pouvez définir plusieurs index pour chaque table temporaire. Si vous en avez besoin, définissez-les. Fondamentalement, un index correspondant à votre requête et / ou à votre ordre de tri aidera à la performance!

```
DEFINE TEMP-TABLE ttWithIndex NO-UNDO
    FIELD field1 AS INTEGER
    FIELD field2 AS CHARACTER
    FIELD field3 AS LOGICAL
    INDEX field1 field1.

DEFINE TEMP-TABLE ttWithoutIndex NO-UNDO
    FIELD field1 AS INTEGER
    FIELD field2 AS CHARACTER
    FIELD field3 AS LOGICAL.

DEFINE VARIABLE i              AS INTEGER     NO-UNDO.
DEFINE VARIABLE iWithCreate    AS INTEGER     NO-UNDO.
DEFINE VARIABLE iWithFind      AS INTEGER     NO-UNDO.
DEFINE VARIABLE iWithoutCreate AS INTEGER     NO-UNDO.
DEFINE VARIABLE iWithoutFind   AS INTEGER     NO-UNDO.

ETIME(TRUE).
DO i = 1 TO 1000:
    CREATE ttWithIndex.
    ttWithIndex.field1 = i.
END.
iWithCreate = ETIME.

ETIME(TRUE).
DO i = 1 TO 1000:
    CREATE ttWithoutIndex.
    ttWithoutIndex.field1 = i.
END.
iWithoutCreate = ETIME.

RELEASE ttWithIndex.
RELEASE ttWithoutIndex.

ETIME(TRUE).
DO i = 1 TO 1000:
    FIND FIRST ttWithIndex WHERE ttWithIndex.field1 = i NO-ERROR.
END.
iWithFind = ETIME.

ETIME(TRUE).
DO i = 1 TO 1000:
    FIND FIRST ttWithoutIndex WHERE ttWithoutIndex.field1 = i NO-ERROR.
END.
iWithoutFind = ETIME.

MESSAGE
    "With index took" iWithFind "ms to find and" iWithCreate "ms to create" SKIP
    "Without index took" iWithoutFind "ms to find and" iWithoutCreate "ms to create"
    VIEW-AS ALERT-BOX.
```



La recherche avec index était environ 70 fois plus rapide que celle sans index! Il ne s'agit là que d'un essai, et non d'une preuve scientifique, mais la configuration de votre index aura un impact.

Saisie et sortie de tables temporaires

Il est très simple de faire passer des tables temporaires dans et hors des programmes, des procédures et des fonctions.

Cela peut être pratique si vous souhaitez qu'une procédure traite un plus grand nombre de données que vous ne pouvez facilement stocker dans une chaîne ou un objet similaire. Vous pouvez passer des tables temporaires sous forme de données `INPUT` , `OUTPUT` et `INPUT-OUTPUT` .

Entrer une table temporaire et en générer une autre:

```
DEFINE TEMP-TABLE ttRequest NO-UNDO
    FIELD fieldA AS CHARACTER
    FIELD fieldB AS CHARACTER.

/* Define a temp-table with the same fields and indices */
DEFINE TEMP-TABLE ttResponse NO-UNDO LIKE ttRequest.

/* A procedure that simply swap the values of fieldA and fieldB */
PROCEDURE swapFields:
    DEFINE INPUT  PARAMETER TABLE FOR ttRequest.
    DEFINE OUTPUT PARAMETER TABLE FOR ttResponse.

    FOR EACH ttRequest:
        CREATE ttResponse.
        ASSIGN
            ttResponse.fieldA = ttRequest.fieldB
            ttResponse.fieldB = ttRequest.fieldA.
    END.
END PROCEDURE.

CREATE ttRequest.
ASSIGN ttRequest.fieldA = "A"
       ttRequest.fieldB = "B".

CREATE ttRequest.
ASSIGN ttRequest.fieldA = "B"
       ttRequest.fieldB = "C".

CREATE ttRequest.
ASSIGN ttRequest.fieldA = "C"
       ttRequest.fieldB = "D".

/* Call the procedure */
RUN swapFields ( INPUT  TABLE ttRequest
               , OUTPUT TABLE ttResponse).

FOR EACH ttResponse:
    DISPLAY ttResponse.
END.
```

Résultat:

```
fieldA--------fieldB--------

B             A
C             B
D             C
```

Entrée-sortie d'une table temporaire:

```
DEFINE TEMP-TABLE ttCalculate NO-UNDO
    FIELD num1     AS INTEGER
    FIELD num2     AS INTEGER
    FIELD response AS DECIMAL.

PROCEDURE pythagoras:
    DEFINE INPUT-OUTPUT PARAMETER TABLE FOR ttCalculate.

    FOR EACH ttCalculate:
        ttCalculate.response = SQRT( EXP(num1, 2) + EXP(num2, 2)).
    END.

END PROCEDURE.

CREATE ttCalculate.
ASSIGN ttCalculate.num1 = 3
       ttCalculate.num2 = 4.

CREATE ttCalculate.
ASSIGN ttCalculate.num1 = 6
       ttCalculate.num2 = 8.

CREATE ttCalculate.
ASSIGN ttCalculate.num1 = 12
       ttCalculate.num2 = 16.

/* Call the procedure */
RUN pythagoras ( INPUT-OUTPUT  TABLE ttCalculate ).

FOR EACH ttCalculate:
    DISPLAY ttCalculate.
END.
```

Résultat:

```
----------num1-- ----------num2-- -------response-

         3               4             5.00
         6               8            10.00
        12              16            20.00
```

Passage aux fonctions

```
DEFINE TEMP-TABLE ttNumbers NO-UNDO
    FIELD num1     AS INTEGER
    FIELD num2     AS INTEGER
    INDEX index1 num1 num2.

DEFINE VARIABLE iNum AS INTEGER     NO-UNDO.
```

```
/* Forward declare the function */
FUNCTION hasAPair RETURNS LOGICAL (INPUT TABLE ttNumbers) FORWARD.

DO iNum = 1 TO 100:
    CREATE ttNumbers.
    ASSIGN ttNumbers.num1 = RANDOM(1,100)
           ttNumbers.num2 = RANDOM(1,100).
END.

MESSAGE hasAPair(INPUT TABLE ttNumbers) VIEW-AS ALERT-BOX.


/* Function to check if two records has the same value in num1 and num2 */
FUNCTION hasAPair RETURNS LOGICAL (INPUT TABLE ttNumbers):

    FIND FIRST ttNumbers WHERE ttNumbers.num1 = ttNumbers.num2 NO-ERROR.
    IF AVAILABLE ttNumbers THEN
        RETURN TRUE.
    ELSE
        RETURN FALSE.

END FUNCTION.
```

Passer aux fichiers de programme

Vous transmettez les tables temporaires vers et depuis d'autres programmes .p de la même manière que vous les transmettez à d'autres procédures. La seule différence est que l'appelant et le programme appelé doivent avoir la même déclaration de table temporaire. Un moyen simple consiste à stocker le programme de la table temporaire dans un troisième fichier - une inclusion utilisée dans les deux programmes.

Fichier d'inclusion contenant la définition de la table temporaire: / * ttFile.i * / DEFINE TEMP-TABLE ttFichier NON-UNDO FIELD fName AS FORMAT DE CARACTERE "x (20)" FIELD isAdirectory AS LOGICAL.

Programme vérifiant tous les fichiers d'une table temporaire. Sont-ils des répertoires?

```
/* checkFiles.p */
{ttFile.i}

DEFINE INPUT-OUTPUT PARAMETER TABLE FOR ttFile.

FOR EACH ttFile:
    FILE-INFO:FILE-NAME = ttFile.fName.

    IF FILE-INFO:FILE-TYPE BEGINS "D" THEN
        ttFile.isADirectory = TRUE.
END.
```

Programme principal:

```
{ttFile.i}

CREATE ttFile.
ASSIGN ttFile.fname = "c:\temp\".
```

```
CREATE ttFile.
ASSIGN ttFile.fname = "c:\Windows\".

CREATE ttFile.
ASSIGN ttFile.fname = "c:\Windoose\".

RUN checkFiles.p(INPUT-OUTPUT TABLE ttFile).

FOR EACH ttFile:
    DISPLAY ttFile.
END.
```

Résultat:

```
fName----------------- isADirector

c:\temp\               yes
c:\Windows\            yes
c:\Windoose\           no
```

Lire TEMP-TABLE en ligne: https://riptutorial.com/fr/progress-4gl/topic/8957/temp-table

Chapitre 11: Travailler avec des nombres

Introduction

Progress ABL prend en charge trois formats de nombres: les entiers et les flottants 32 et 64 bits.

Examples

Les opérateurs

Progress prend en charge les opérateurs + / - *. Ils ne peuvent pas être surchargés. Division renvoie toujours une décimale. Si l'un des nombres dans un calcul est une décimale, une décimale sera renvoyée. Sinon un `INTEGER` ou `INT64` .

Il n'y a pas d'opérateur `+=` ou `++` . Pour augmenter ou diminuer une variable, vous devez lui attribuer plus ou moins quelque chose. Donc, pour ajouter 1 à une variable que vous faites: `i = i + 1.`

```
DEFINE VARIABLE i AS INTEGER     NO-UNDO.
DEFINE VARIABLE j AS INTEGER     NO-UNDO.

i = 3.
j = 2.

DISPLAY i + j. // 3 + 2 = 5

DISPLAY i - j. // 3 - 2 = 1

DISPLAY i / j. // 3 / 2 = 1.5

DISPLAY INTEGER(i / j). //Integer(3/2) = 2.

DISPLAY i * j. //3 x 2 = 6
```

Plus de fonctions mathématiques

EXP - Renvoie le résultat de l'augmentation d'un nombre à une puissance.

EXP (base, exposant)

```
MESSAGE EXP(10, 2) VIEW-AS ALERT-BOX. // Messages 100
```

SQRT - Retourne la racine carrée d'un nombre.

SQRT (nombre)

```
MESSAGE "The square root of 256 is " SQRT(256) VIEW-AS ALERT-BOX. // Messages 16
```

MODULO - Détermine le reste après la division.

expression base MODULO

```
DISPLAY 52 MODULO 12. //Displays 4
```

ROUND - Arrondit une expression décimale à un nombre spécifié de lieux après le point décimal.

ROUND (nombre, précision)

```
DISPLAY ROUND(67.12345, 6) FORMAT "99.99999". // 67.12345
DISPLAY ROUND(67.12345, 5) FORMAT "99.99999". // 67.12345
DISPLAY ROUND(67.12345, 4) FORMAT "99.99999". // 67.12350
DISPLAY ROUND(67.12345, 3) FORMAT "99.99999". // 67.12300
DISPLAY ROUND(67.12345, 2) FORMAT "99.99999". // 67.12000
DISPLAY ROUND(67.12345, 1) FORMAT "99.99999". // 67.10000
DISPLAY ROUND(67.12345, 0) FORMAT "99.99999". // 67.00000
```

TRUNCATE Truncate une expression décimale à un nombre spécifié de décimales, renvoyant une valeur décimale.

TRUNCATE (nombre, lieux)

```
DISPLAY TRUNCATE(67.12345, 6) FORMAT "99.99999". // 67.12345
DISPLAY TRUNCATE(67.12345, 5) FORMAT "99.99999". // 67.12345
DISPLAY TRUNCATE(67.12345, 4) FORMAT "99.99999". // 67.12340
DISPLAY TRUNCATE(67.12345, 3) FORMAT "99.99999". // 67.12300
DISPLAY TRUNCATE(67.12345, 2) FORMAT "99.99999". // 67.12000
DISPLAY TRUNCATE(67.12345, 1) FORMAT "99.99999". // 67.10000
DISPLAY TRUNCATE(67.12345, 0) FORMAT "99.99999". // 67.00000
```

ABSOLUTE - Retourne la valeur absolue d'un nombre

```
DISPLAY ABS(10 - 12). //Displays 2
DISPLAY ABS(-2) = ABS(2). //Displays yes
```

MINIMUM et **MAXIMUM** - retourne le plus petit et le plus grand nombre

MINIMUM (number1, number2, ... numbern)

MAXIMUM (number1, number2, ... numbern)

```
DEFINE VARIABLE i AS INTEGER     NO-UNDO.
DEFINE VARIABLE j AS INTEGER     NO-UNDO.
DEFINE VARIABLE k AS INTEGER     NO-UNDO.

i = 40.
j = 45.
k = 56.

DISPLAY MINIMUM(i, j, k) MAXIMUM(i, j, k). // Displays 40 and 56
```

Comparaison des nombres

Il y a des fonctions standard intégrées pour comparer l'égalité, l'inégalité, etc.

prénom	symbole	Alternative	Exemple
Égal	=	EQ	i = j
Inégal	<>	NE	i <> j
Moins que	<	LT	i <j
inférieur ou égal	<=	LE	i <= j
Plus grand que	> =	GT	i> j
Meilleur que ou égal	≥ =	GE	i> = j

Le symbole peut être échangé avec l'alternative et vice versa. Donc `var1 <> var2` est la même chose que `var1 NE var2` .

Vous pouvez comparer un flottant à un entier, mais vous ne pouvez pas comparer, par exemple, une date avec un entier.

Générateur de nombres aléatoires

RANDOM - génère un nombre aléatoire

> RANDOM (bas, haut)
>
> Génère un entier pseudo-aléatoire entre bas et haut

```
// Example that generates 20 random numbers between 1 and 20 (1 and 20 included)
DEFINE VARIABLE i AS INTEGER     NO-UNDO.

DO i = 1 TO 20.
    DISPLAY i RANDOM(1, 20).
    PAUSE.
END.
```

Lire Travailler avec des nombres en ligne: https://riptutorial.com/fr/progress-4gl/topic/8878/travailler-avec-des-nombres

Chapitre 12: TROUVER une déclaration

Introduction

L'instruction `FIND` est utilisée pour extraire un enregistrement unique d'une seule table. Il a quelques limitations par rapport à `FOR` ou `QUERY` , mais c'est une déclaration simple et pratique pour un accès rapide aux enregistrements.

Examples

Trouver des exemples de base

Un exemple simple de sports2000:

```
FIND FIRST Customer NO-LOCK WHERE CustNum = 1 NO-ERROR.
IF AVAILABLE Customer THEN DO:
    DISPLAY Customer.NAME.
END.
ELSE DO:
    MESSAGE "No record available".
END.
```

FIRST - trouver le premier enregistrement qui correspond à la requête

NO-LOCK - ne verrouille pas l'enregistrement - ce qui signifie que nous ne ferons que lire et non modifier l'enregistrement.

OERE - c'est la requête

NO-ERROR - n'échoue pas s'il n'y a aucun enregistrement disponible.

(IF) DISPONIBLE - vérifiez si nous avons trouvé un enregistrement ou non

```
findLoop:
REPEAT :
    FIND NEXT Customer NO-LOCK WHERE NAME BEGINS "N" NO-ERROR.

    IF AVAILABLE customer THEN DO:
        DISPLAY Customer.NAME.
    END.
    ELSE DO:
        LEAVE findLoop.
    END.
END.
```

Disponibilité et étendue

La dernière trouvaille est toujours celui de la vérification de la disponibilité fonctionnera contre - une découverte unsuccesful fera `AVAILABLE` faux retour:

```
DEFINE TEMP-TABLE tt NO-UNDO
    FIELD nr AS INTEGER.

CREATE tt.
tt.nr = 1.

CREATE tt.
tt.nr = 2.

CREATE tt.
tt.nr = 3.

DISPLAY AVAILABL tt. // yes (tt with nr = 3 is still available)

FIND FIRST tt WHERE tt.nr = 1 NO-ERROR.
DISPLAY AVAILABLE tt. //yes

FIND FIRST tt WHERE tt.nr = 2 NO-ERROR.
DISPLAY AVAILABLE tt. //yes

FIND FIRST tt WHERE tt.nr = 3 NO-ERROR.
DISPLAY AVAILABLE tt. //yes

FIND FIRST tt WHERE tt.nr = 4 NO-ERROR.
DISPLAY AVAILABLE tt. //no
```

Un enregistrement trouvé dans "main" sera disponible dans toutes les procédures.

```
DEFINE TEMP-TABLE tt NO-UNDO
    FIELD nr AS INTEGER.

PROCEDURE av:
    DISPLAY AVAILABLE tt.

    IF AVAILABLE tt THEN DO:
        DISPLAY tt.nr.
    END.
END PROCEDURE.

CREATE tt.
tt.nr = 1.

RUN av. // yes. tt.nr = 1

CREATE tt.
tt.nr = 2.

RUN av. // yes. tt.nr = 2

FIND FIRST tt WHERE tt.nr = 4 NO-ERROR.

RUN av. // no (and no tt.nr displayed)
```

En outre, un enregistrement trouvé dans une procédure sera toujours disponible après la fin de cette procédure.

```
DEFINE TEMP-TABLE tt NO-UNDO
    FIELD nr AS INTEGER.
```

```
PROCEDURE av:
    FIND FIRST tt WHERE tt.nr = 1.
END PROCEDURE.

CREATE tt.
tt.nr = 1.

CREATE tt.
tt.nr = 2.

DISPLAY AVAILABLE tt WITH FRAME x1. // yes.

IF AVAILABLE tt THEN DO:
    DISPLAY tt.nr WITH FRAME x1. //tt.nr = 2
END.

PAUSE.

RUN av.

DISPLAY AVAILABLE tt WITH FRAME x2. // yes.

IF AVAILABLE tt THEN DO:
    DISPLAY tt.nr WITH FRAME x2. //tt.nr = 1
END.
```

TROUVER et verrouiller

Chaque fois que vous `FIND` un enregistrement que vous pouvez acheter un verrou de celui - ci.

NO-LOCK: Utilisé pour les opérations en lecture seule. Si vous faites un `FIND <record> NO-LOCK` vous ne pouvez en aucun cas modifier l'enregistrement.

```
FIND FIRST Customer NO-LOCK NO-ERROR.
```

EXCLUSIVE-LOCK: Utilisé pour les mises à jour et les suppressions. Si vous faites cela, vous "posséderez" l'enregistrement et personne d'autre ne pourra le modifier ou le supprimer tant que vous n'en aurez pas terminé. Ils peuvent le lire (sans verrou) tant que vous ne l'avez pas supprimé.

```
FIND FIRST Customer EXCLUSIVE-LOCK NO-ERROR.
```

SHOCK-LOCK: Évitez à tout prix. Cela va causer des maux de tête.

```
FIND FIRST Customer EXCLUSIVE-LOCK NO-ERROR. //Do this instead.
```

MISE À NIVEAU de votre verrou de NO-LOCK à EXCLUSIVE-LOCK

Vous pouvez facilement passer d'un `NO-LOCK` à un `EXCLUSIVE-LOCK` si vous avez besoin de modifier un enregistrement:

```
FIND FIRST Customer NO-LOCK NO-ERROR.
```

```
// Some code goes here
// Now we shall modify
FIND CURRENT Customer EXCLUSIVE-LOCK NO-ERROR.
```

Vous pouvez également passer de EXCLUSIVE-LOCK à NO-LOCK.

REGISTRES VERROUILLÉS

Chaque fois que d'autres utilisateurs peuvent acquérir une serrure de disque, il vaut mieux prendre en compte cette possibilité. Des collisions se produiront!

Il est préférable de gérer cela par programmation en utilisant l'instruction `NO-WAIT` . Cela dit à l'AVM de simplement passer le FIND si l'enregistrement est verrouillé par quelqu'un d'autre et de vous laisser gérer ce problème.

```
FIND FIRST Customer EXCLUSIVE-LOCK NO-ERROR NO-WAIT.

/* Check for availability */
IF AVAILABLE Customer THEN DO:

    /* Check that no lock (from somebody else) is present */
    IF NOT LOCKED Customer THEN DO:
        /* Do your stuff here */
    END.
    ELSE DO:
        MESSAGE "I'm afraid somebody else has locked this record!" VIEW-AS ALERT-BOX ERROR.
    END.
END.
```

Lire TROUVER une déclaration en ligne: https://riptutorial.com/fr/progress-4gl/topic/8941/trouver-une-declaration

Chapitre 13: Utilitaires OS

Introduction

Il existe plusieurs fonctions et instructions intégrées pour accéder au système d'exploitation.

Examples

OS-COMMAND

Exécute une commande OS.

OS-COMMAND sans aucune option démarrera un nouveau shell et ne le quittera pas. Ainsi, sous OS graphique: vous laisserez une fenêtre "en suspens".

```
DEFINE VARIABLE cmd AS CHARACTER   NO-UNDO.

cmd = "dir".

OS-COMMAND VALUE(cmd).
```

Il existe trois options: `SILENT` , `NO-WAIT` et `NO-CONSOLE` .

SILENCIEUX

> Après avoir traité une commande du système d'exploitation, le shell AVM s'interrompt. Pour quitter la fenêtre dans les plates-formes d'interface graphique Windows, vous devez taper exit. Pour quitter la fenêtre dans les plates-formes de caractères Windows, vous devez taper exit et appuyer sur RETOUR ou sur ESPACE. Vous pouvez utiliser l'option SILENT pour éliminer cette pause. Utilisez cette option uniquement si vous êtes certain que le programme, la commande ou le fichier de commandes ne génère aucune sortie à l'écran. Ne peut pas être utilisé avec NO-WAIT.

```
OS-COMMAND SILENT VALUE("runprogram.exe").
```

NON ATTENDS

> Dans un environnement multitâche, l'AVM passe immédiatement le contrôle à l'instruction suivante après l'OS-COMMAND sans attendre la fin de la commande du système d'exploitation. Ne peut pas être utilisé avec SILENT. Cette option est prise en charge uniquement sous Windows.

```
OS-COMMAND NO-WAIT VALUE("DIR > dirfile.txt").
```

Sur Linux / Unix, vous devrez le faire en précédant la commande par un signe `&` -sign:

```
OS-COMMAND VALUE("ls >> file.txt &").
```

NO-CONSOLE

> Lors du traitement d'une commande du système d'exploitation, l'AVM crée une fenêtre de console. La fenêtre de la console ne peut pas être nettoyée après l'exécution de la commande. Vous pouvez utiliser l'option NO-CONSOLE pour empêcher cette fenêtre d'être créée en premier lieu.

```
OS-COMMAND NO-CONSOLE VALUE("startbach.bat").
```

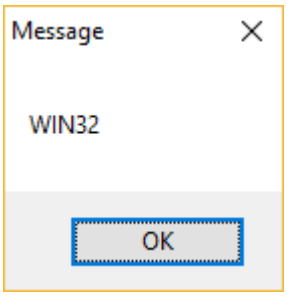
Aucune erreur n'est jamais renvoyée de `OS-COMMAND` vers Progress ABL, vous devez donc rechercher les erreurs d'une autre manière, en les écrivant éventuellement dans un fichier dans un script shell ou similaire.

OPSYS

La `OPSYS` OPSYS renvoie le système d'exploitation sur lequel le programme est exécuté:

```
MESSAGE OPSYS VIEW-AS ALERT-BOX.
```

Résultat:



Il peut être utilisé pour sélectionner l'utilitaire OS à appeler:

```
IF OPSYS = "LINUX" THEN
    OS-COMMAND VALUE("ls -l").
ELSE
    OS-COMMAND VALUE("dir").
```

OS-ERROR

Retourne une erreur d'un précédent appel `OS-*` représenté par un entier. Les appels pouvant renvoyer une erreur OS sont les suivants:

- OS-APPEND
- OS-COPY
- OS-CREATE-DIR
- OS-DELETE
- OS-RENAME

- SAUVEZ CACHE

Notez que `OS-COMMAND` est manquant. Vous devez gérer vous-même les erreurs dans OS-COMMAND.

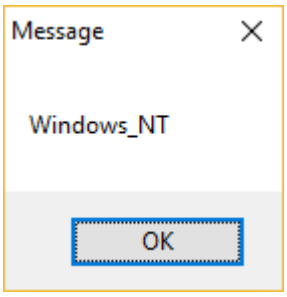
Numéro d'erreur	La description
0	Pas d'erreur
1	Pas propriétaire
2	Aucun fichier ou répertoire de ce nom
3	Appel système interrompu
4	Erreur d'E / S
5	Numéro de dossier incorrect
6	Plus de processus
7	Pas assez de mémoire de base
8	Permission refusée
9	Mauvaise adresse
dix	le fichier existe
11	Aucun tel appareil
12	Pas un annuaire
13	Est un directeur
14	Débordement de table de fichiers
15	Trop de fichiers ouverts
16	Fichier trop large
17	Pas d'espace disponible sur le périphérique
18	Répertoire non vide
999	Erreur non mappée (par défaut ABL)

Fonction OS-GETENV

Renvoie la valeur de toute variable d'environnement du système d'exploitation.

```
MESSAGE OS-GETENV ("OS") VIEW-AS ALERT-BOX.
```

Sur une machine Windows:




```
MESSAGE OS-GETENV ("SHELL") VIEW-AS ALERT-BOX.
```

Résultat sur une machine Linux avec Bash en tant que shell actuel:

```
┌──────── Message ────────┐
│       /bin/bash        │
│ ─────────────────────── │
│          <OK>           │
└─────────────────────────┘
```

OS-COPY

Copier un fichier

> Fichier cible du fichier source COPY

Copiez `c:\temp\source-file.txt` dans `c:\temp\target-file.txt` . Vous devez vérifier `OS-ERROR` pour la réussite ou l'absence de celle-ci.

```
OS-COPY VALUE("c:\temp\source-file.txt")  VALUE("c:\temp\target-file.txt").
IF OS-ERROR <> 0 THEN DO:
    MESSAGE "An error occured" VIEW-AS ALERT-BOX ERROR.
END.
```

OS-DELETE

Supprime un fichier ou une arborescence de fichiers.

Comme avec beaucoup d'autres utilitaires OS- *, vous devez vérifier l'état dans `OS-ERROR` .

> OS-DELETE fichier-ou-dir-to-delete [RECURSIVE]

Supprimez l'arbre entier `/tmp/dir` :

```
OS-DELETE VALUE("/tmp/dir") RECURSIVE.
```

Supprimez le fichier appelé `c:\dir\file.txt`

```
OS-DELETE VALUE("c:\dir\file.txt").
```

OS-CREATE-DIR

Crée un répertoire, le statut est dans `OS-ERROR`

> Répertoire OS-CREATE-DIR

Créez un répertoire appelé `/usr/local/appData`

```
OS-CREATE-DIR VALUE("/usr/local/appData").
```

OS-APPEND

Ajouter un fichier à un autre. L'état est vérifié dans `OS-ERROR`

> Cible source OS-APPEND

Ajoute `targetfile.txt` avec `sourcefile.txt` :

```
OS-APPEND VALUE("sourcefile.txt") VALUE("targetfile.txt").
```

OS-RENAME

Renommez un fichier ou un répertoire. L'état est dans `OS-ERROR` . Peut également être utilisé pour déplacer des fichiers (ou les déplacer et les renommer).

> OS-RENAME oldname newname

Renommez `/tmp/old-name` en `/tmp/new-name` :

```
OS-RENAME VALUE("/tmp/old-name") VALUE("/tmp/new-name").
```

Déplacez le fichier `c:\temp\old.txt` vers `c:\new-dir\old.txt` :

```
OS-RENAME VALUE("c:\temp\old.txt") VALUE("c:\new-dir\old.txt").
```

OS-DRIVES (Windows uniquement)

Renvoie une liste de tous les lecteurs d'un système.

```
MESSAGE OS-DRIVES VIEW-AS ALERT-BOX.
```

Résultat avec quatre lecteurs, C à F:

Sous Linux, la liste sera simplement vide car, par définition, aucun "lecteur" n'est connecté. Lister les répertoires se fait d'une autre manière (`INPUT FROM OS-DIR`)

Lire Utilitaires OS en ligne: https://riptutorial.com/fr/progress-4gl/topic/9056/utilitaires-os

Crédits

S. No	Chapitres	Contributeurs
1	Démarrer avec progress-4gl	Community, Jensd, Stephen Leppik
2	Compiler	Jensd
3	Cordes	Jensd
4	Expressions conditionnelles	Jensd
5	Itérer	Jensd
6	Les fonctions	Jensd
7	Les variables	Jensd
8	Procédures	Jensd
9	Requêtes	Jensd, R3uK
10	TEMP-TABLE	Jensd
11	Travailler avec des nombres	Jensd
12	TROUVER une déclaration	Jensd
13	Utilitaires OS	Jensd