

**Introduction à
jGRASP 1.5.2
et
GVD 1.2.5**

GRASP



Auteur: Benoît Guye

Prof. responsable : R. Rentsch

1. Introduction	2
2. Installation	4
3. Démarrage de l'application et présentation de l'interface	5
L'environnement	5
4. Configuration de jGRASP	6
Onglet Languages	7
Onglet CSD	8
Onglet Colors	9
Onglet Font	10
Onglet Font Size	10
Onglet View	10
Onglet Sources	11
5. Configuration de l'impression	11
Onglet Common	11
Onglet CSD/Messages	12
6. Utilisation de jGRASP	14
Créer / sauvegarder / charger un fichier	14
Génération d'une structure CSD	16
Génération d'un graphique CPG	18
Numérotation des lignes	19
Compilation d'un programme	19
Exécution du programme	20
Déverminage du programme à l'aide de GVD	21
Installation	21
Pré-requis à l'utilisation	21
Configuration de jGRASP pour l'utilisation de GVD	22
Utilisation de GVD	23
Insertion des points d'ancrages	23
Démarrage du déverminage	24
Affichage des structures de données	25
Utilisation des templates	28
7. Les projets	29
Créer un projet	29
Ouvrir un projet	29
Ajouter un fichier au projet	30
Suppression d'un fichier d'un projet	30
Fermeture d'un projet	30
Annexe A : Bugs de jGRASP	31

1.

Introduction

jGRASP (java Graphical Representations for Algorithms, Structures, and Processes) est un environnement de développement qui supporte les langages Java, C, C++, et Ada et qui peut être configuré pour travailler avec d'autres langages.

Développé à l'université de Auburn, il est basé sur les outils PC GRASP et UNIX GRASP, écrits en C/C++, et est implémenté en Java. Cet outil est compatible avec toutes les plates-formes qui supportent une machine virtuelle Java.

Simple éditeur à ses débuts, jGRASP est aujourd'hui un environnement de développement complet.

Les principales fonctionnalités mises à disposition par jGRASP sont

- **Un environnement de développement multi-langage:** permet d'avoir un outil convivial permettant l'écriture de programmes dans plusieurs langages. Cette interface (Figure 1) permet, pour des étudiants par exemple, d'utiliser le même programme et donc la même interface, quel que soit le langage utilisé. Par défaut, les langages principaux supportés sont Ada, C, C++ et Java.

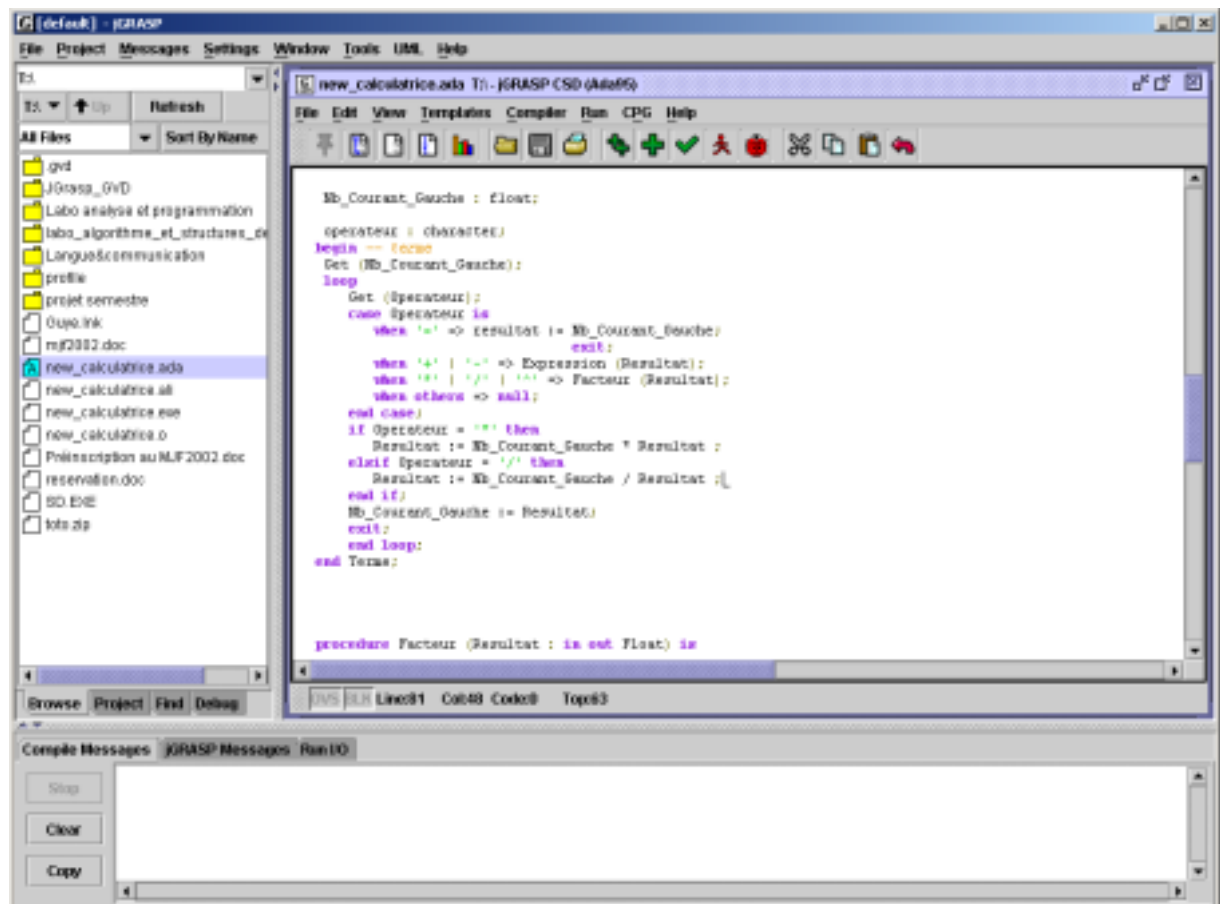


Figure 1 : L'environnement de développement

- **CSD (Control Structure Diagram) :** CSD (Figure 2) est une structure générée automatiquement à partir d'un code source en Ada, C, C++ ou Java permettant

de faciliter la lisibilité du code en insérant différentes « boîtes » en fonction des structures utilisées (cf. page 17).

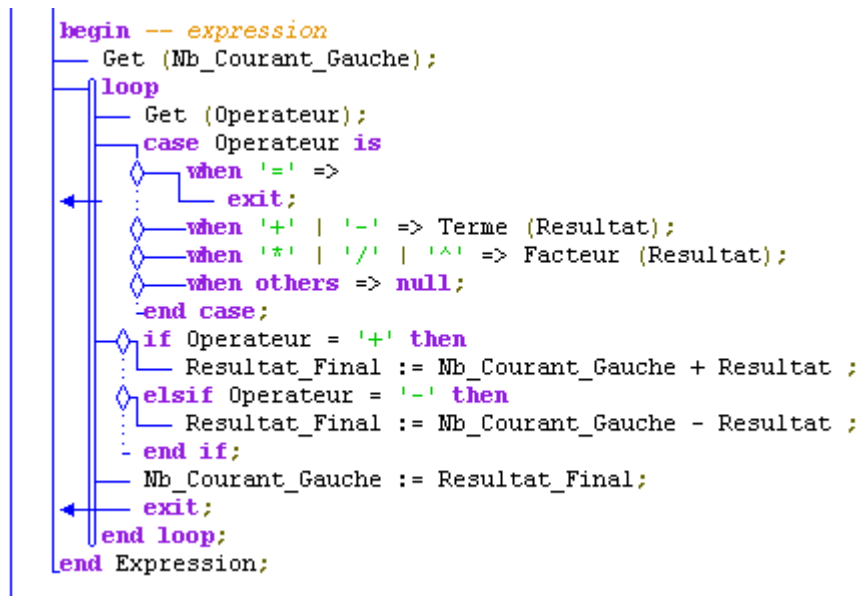


Figure 2 : L'affichage CSD

- **CPG (Complexity Profile Graph)** : Graphique représentant la complexité d'un code source (Figure 3) (cf. page 19).

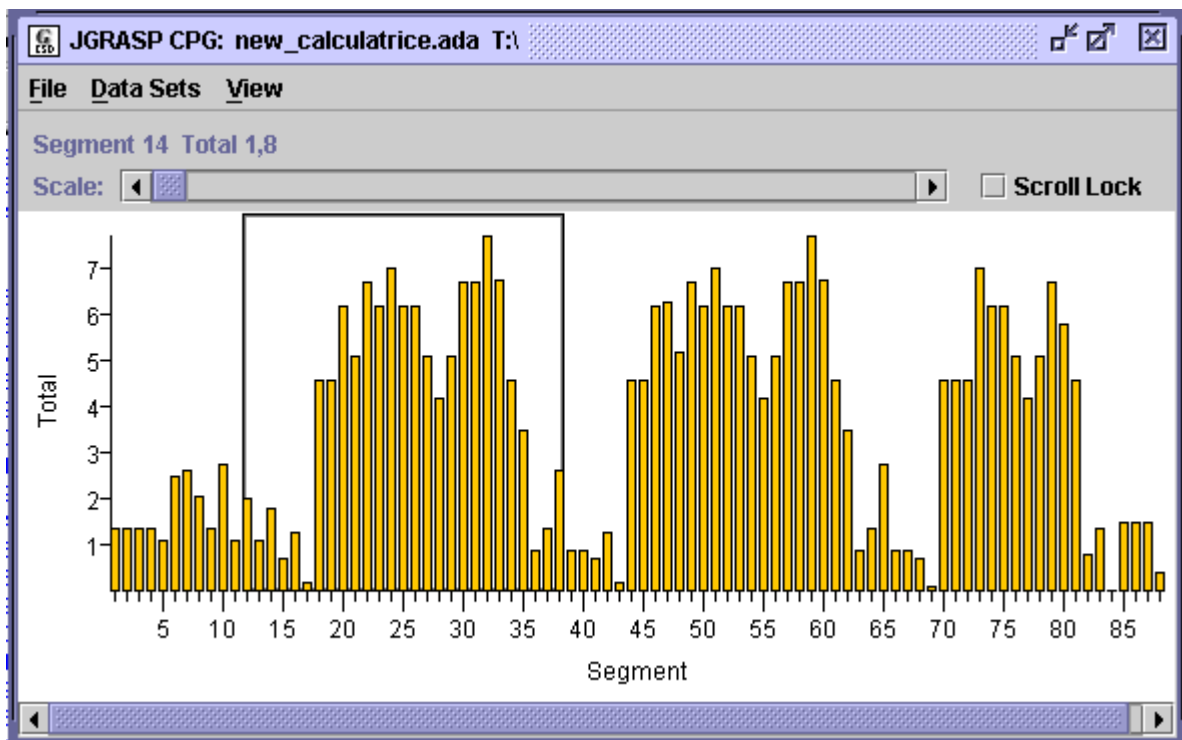


Figure 3 : Un graphique CPG

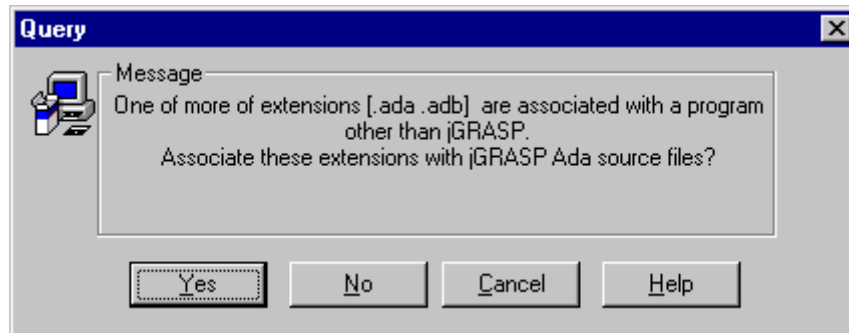
2. Installation

jGRASP peut être téléchargé librement sur le site <http://www.eng.auburn.edu/jgrasp> en 4 versions : deux versions auto-extractables pour Microsoft Windows. L'une avec JRE (Java TM 2 Runtime Environment) et l'autre sans. Celle contenant JRE est à installer si aucune machine virtuelle Java n'est installée sur la machine.

La troisième version est destinée à Mac OS X et la dernière est un fichier zip qui peut être utilisé pour installer ce produit sur une machine Unix ou Linux.

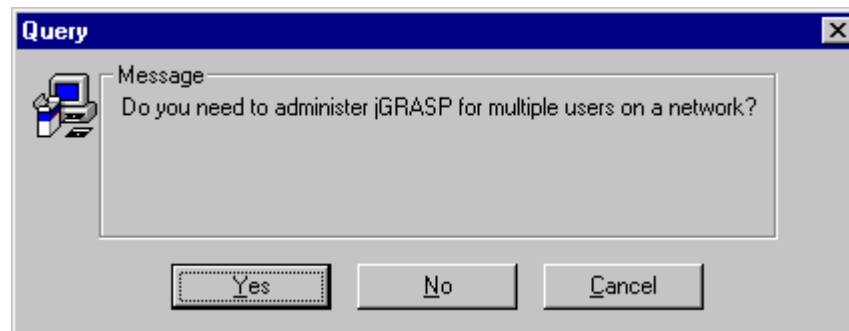
Pour les auto-extractables, l'installation se fait par un simple double-clic sur l'exécutable. Le script d'installation se lance. Il suffit de choisir le répertoire d'installation (par défaut C:\jGRASP), le répertoire d'installation pour les extensions (par défaut C:\jGRASP_extensions) puis de choisir les extensions qui seront reconnues par jGRASP.

La boîte de dialogue ci-dessous permet d'associer les extensions Ada à jGRASP. Cette opération aura simplement pour but d'ouvrir jGRASP en cas de double-clic sur un fichier portant cette extension.



Une fois ces associations effectuées, le script installe le programme à proprement parler.

A la fin de l'installation, le programme demande à l'utilisateur s'il désire administrer jGRASP sur un réseau en mode multi-utilisateur. Cette option permet de partager des fichiers de configuration entre plusieurs utilisateurs. Dans le cas présent, il faut répondre No à cette boîte de dialogue.



L'installation se termine sur cette boîte de dialogue.

3. Démarrage de l'application et présentation de l'interface

Pour ouvrir l'application jGRASP lorsqu'on se trouve dans la fenêtre principale de Windows, il suffit de cliquer sur le bouton **Démarrer**, puis de placer le curseur successivement sur le groupe **Programme**, puis sur le groupe **jGRASP** et finalement de cliquer sur l'icône **jGRASP**. La fenêtre de la figure 4 apparaît (avec d'éventuelles variantes !) :

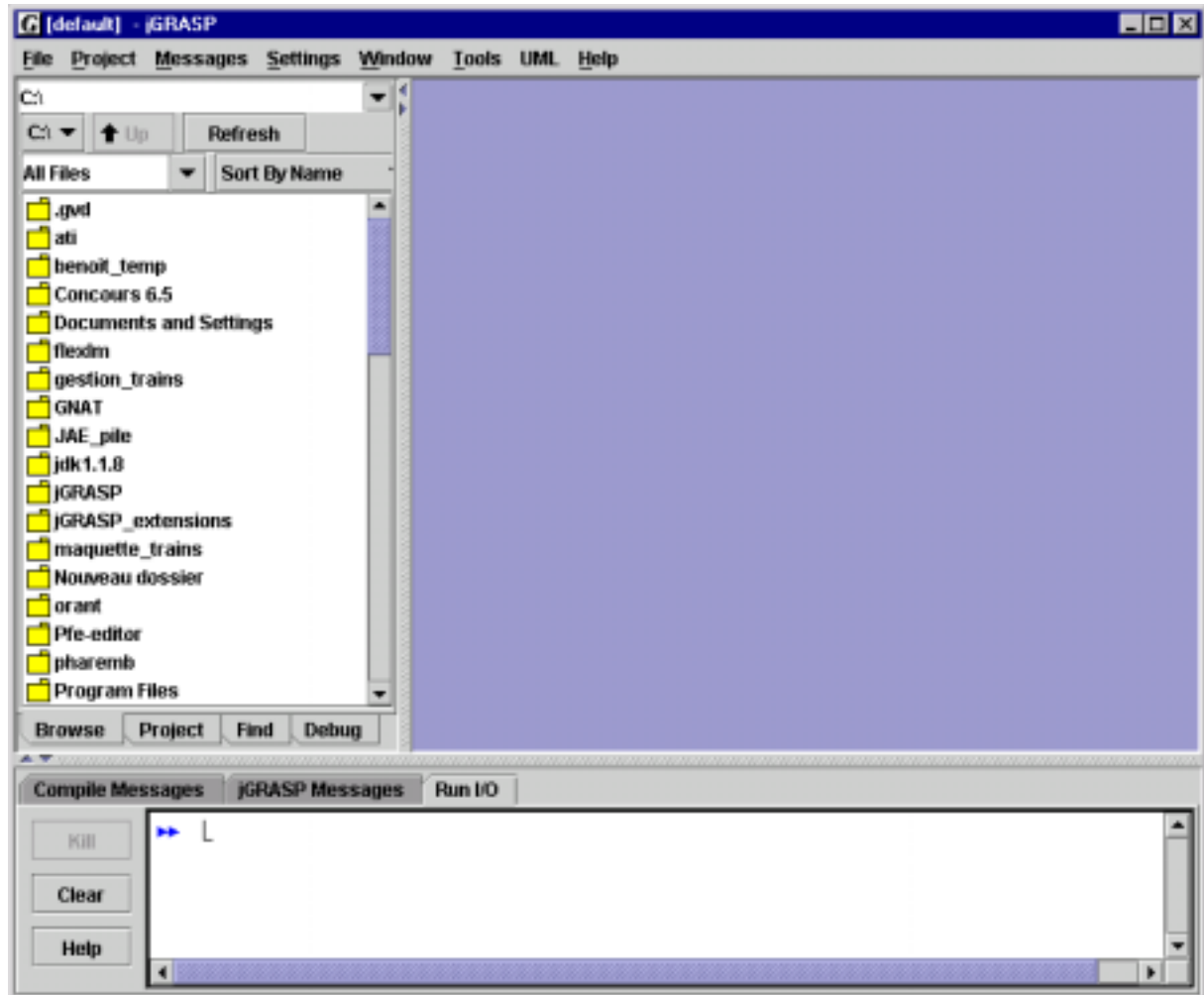


Figure 4 : Fenêtre principale

L'environnement

L'interface de jGRASP est composée de 4 parties distinctes :

- En haut, la barre de menu qui permet d'accéder aux différents menus déroulants. Ces menus seront détaillés plus loin.
- A gauche, le navigateur qui permet de parcourir les différents répertoires du disque.
- A droite, la fenêtre de travail dans laquelle il est possible d'interagir avec le programme. Les différentes fenêtres qui vont être utilisées vont être ouvertes dans cet espace.
- En bas, la fenêtre de dialogue ou de sortie. C'est dans cette partie que le logiciel va inscrire le résultat de son travail.

4. Configuration de jGRASP

La configuration de jGRASP peut se faire sous trois formes différentes :

- Configuration globale : met à jour la configuration pour toute l'application.
- Configuration sur un projet : permet de configurer un projet et tous les fichiers rattachés à ce projet.
- Configuration sur un fichier : permet d'appliquer la configuration à un seul fichier.

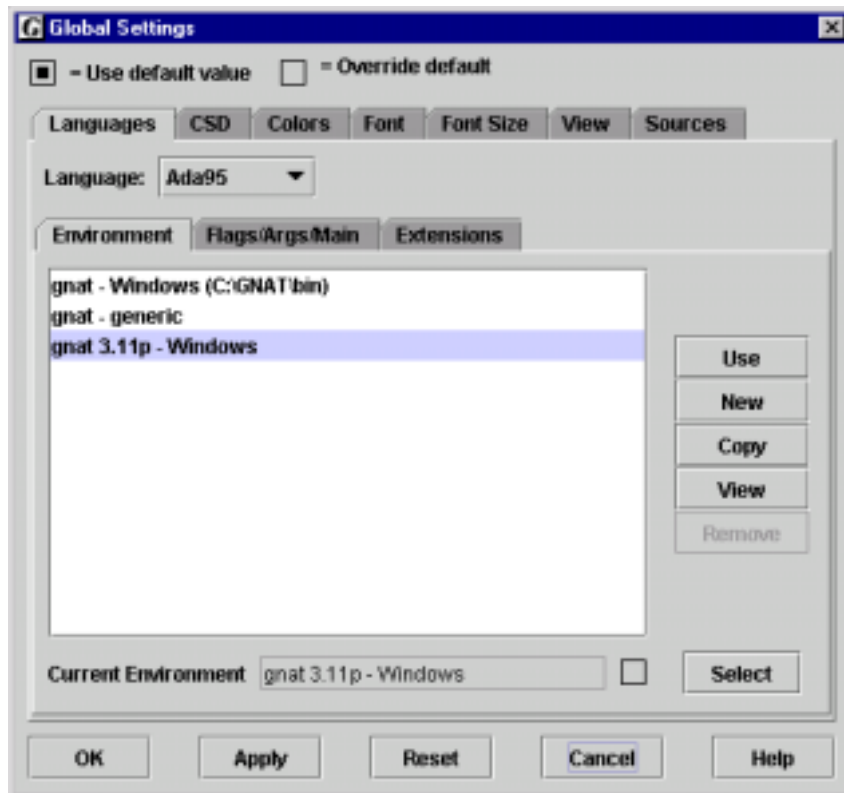


Figure 5 : Fenêtre de configuration globale
Onglet Languages

La configuration sur un fichier est prioritaire par rapport à la configuration sur un projet qui est prioritaire par rapport à la configuration globale.

Lors de l'ouverture de jGRASP, la fenêtre de configuration (figure 5) est accessible par le menu **Settings/CSD Window Settings/Global** ou par la combinaison de touches <Ctrl+G>.

La configuration sur un projet est accessible par le menu **Settings/CSD Window Settings/Project**.

Enfin, la configuration sur un fichier est accessible par le menu **Edit/CSD Window Settings/File** du fichier ouvert dans la fenêtre de travail.

La description des différents onglets et des différentes modifications à apporter pour le bon fonctionnement de l'application sont présentées ci-dessous.

Onglet Languages

L'onglet Languages (Figure 5, ci-dessus) permet de définir l'environnement de programmation en fonction du choix du langage. Pour chaque langage disponible (dans la liste déroulante Language), il existe un certain nombre d'environnements prédéfinis (onglet Environment). Ils spécifient les commandes de compilation, d'édition de liens, de déverminage, d'exécution, les répertoires de travail, les formats d'erreurs et les variables d'environnement. Tous ces environnements prédéfinis ne peuvent être modifiés et sont utilisables pour autant que les outils correspondants soient installés (gnat pour Ada, par exemple).

Bien entendu, il est possible de définir son propre environnement.

Pour choisir son environnement de travail, il suffit de choisir le langage (liste déroulante Language), puis de cliquer sur l'environnement désiré et enfin de cliquer sur le bouton Use. Ainsi, le changement est effectué.

Enfin, l'onglet Extensions permet de définir les extensions se rattachant au langage de programmation.

Cet onglet est paramétrable pour les trois niveaux de configurations.

Onglet CSD

L'onglet CSD (Figure 6) définit les options de génération de la structure CSD.

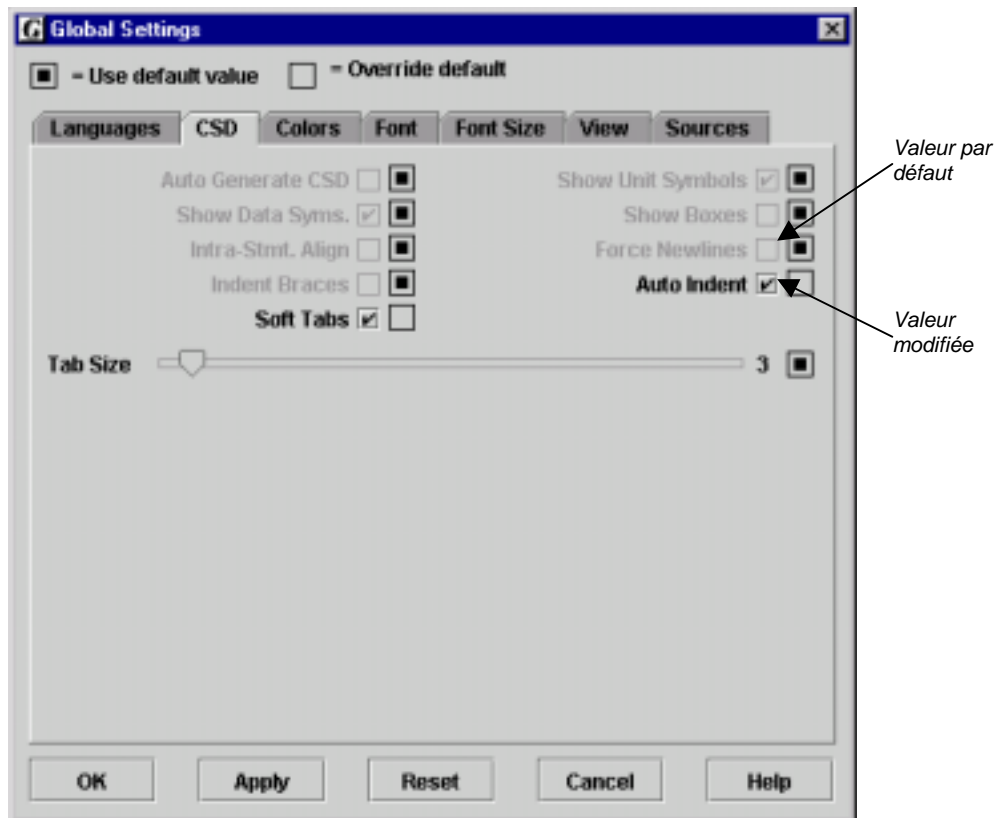


Figure 6 : Fenêtre de configuration globale
Onglet CSD

Les cases noires indiquent que la valeur est celle définie par défaut. Pour modifier une valeur, il faut d'abord cliquer sur la case noire puis ensuite modifier la valeur désirée.

- Auto Generate CSD : Si cochée, génère automatiquement une structure CSD à l'ouverture d'un fichier dans la fenêtre de travail. Laisser la valeur par défaut (non cochée).
- Show Data Syms. : Cette option permet d'afficher les symboles pour les types et les déclarations de variables. Option visible uniquement si l'affichage CSD est généré. Laisser la valeur par défaut.
- Intra-Stmt. Align : permet d'aligner les parenthèses et affectations. Laisser la valeur par défaut.
- Indent Braces : Option spécifique à C, C++ et Java. Laisser la valeur par défaut.
- Soft Tabs : Permet de remplacer les tabulations par des espaces si l'utilisateur utilise la touche de tabulation, indentation de blocs et l'indentation automatique.
Activer l'option.
- Show Unit Symbols : Permet de visualiser les symboles des fonctions, méthodes et paquets. Option visible uniquement si l'affichage CSD est généré. Laisser la valeur par défaut.

- Show Boxes : Si cette option est activée, des boîtes sont dessinées autour des principales structures du code. Option visible uniquement si l'affichage CSD est généré. Laisser la valeur par défaut.
- Force Newlines : Si cette option est activée, les instructions appondues (Skip_Line après un New_Line, par exemple) sont séparées et mises chacune sur une ligne différente. Laisser la valeur par défaut.
- Auto Indent : si cette option est activée, elle indente de la même façon la ligne courante, lors d'un retour à ligne, que la ligne qui précède. **Activer cette option.**
- Tab Size : Permet de définir la taille d'une tabulation en caractères. Par défaut cette option est mise à 3. Laisser la valeur par défaut.

Onglet Colors

Cet onglet (Figure 7) permet de configurer la couleur de l'application (couleur de fond, messages du compilateur...), mais également le rendu visuel (commentaires, mots clés, chaînes de caractères, etc.). L'impression n'est pas affectée par ce choix. Libre à l'utilisateur de configurer cette partie selon ses goûts personnels. Ce paramétrage est uniquement applicable à la configuration globale.



Figure 7 : Fenêtre de configuration globale
Onglet Colors

Onglet Font

Cet onglet (Figure 8) permet de configurer les attributs lexicaux ainsi que la police utilisée par CSD, les différents messages ainsi que l'apparence des fichiers ouverts dans la fenêtre de travail. Ce paramétrage est uniquement applicable à la configuration globale.

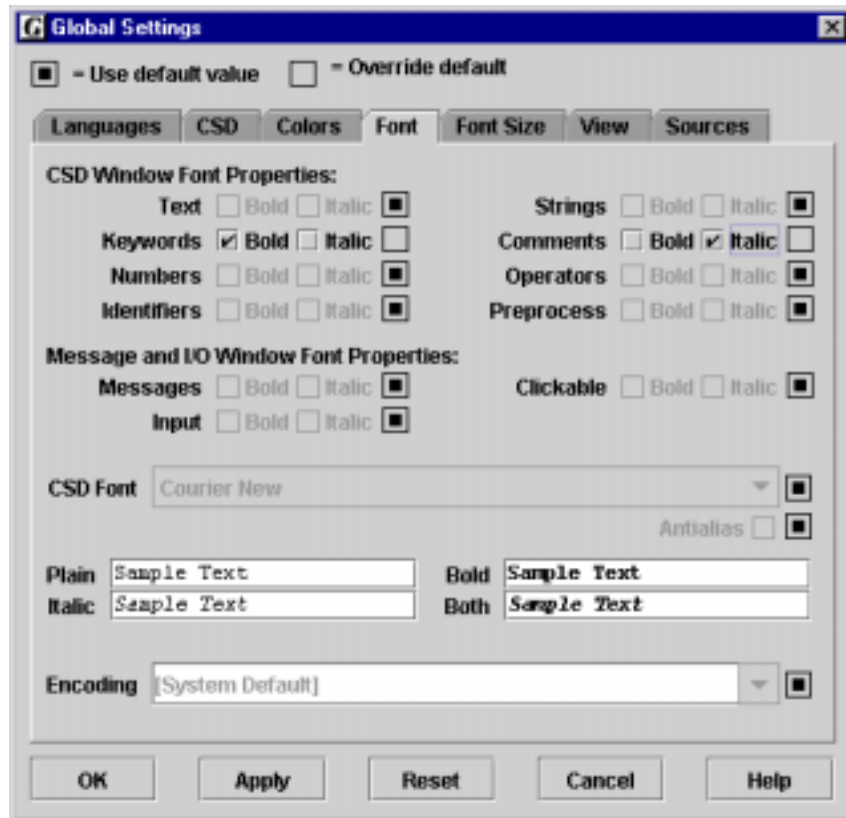


Figure 8 : Fenêtre de configuration globale
Onglet Font

Il est conseillé, pour une meilleure lisibilité du code lors de l'impression, de modifier la configuration de base en mettant les mots clés en gras ainsi que les commentaires en italique (voir Figure 8).

Onglet Font Size

L'onglet Font Size permet de régler la taille de la police dans la zone de travail, pour les messages et pour l'apparence globale. Libre à l'utilisateur de choisir selon ses préférences.

Ce paramétrage est uniquement applicable à la configuration globale.

Onglet View

L'onglet View permet à l'utilisateur de :

- Afficher les numéros de lignes.
- Afficher la barre d'outils.
- Afficher la barre de messages.
- Définir l'apparence de la barre d'outils.

Ce paramétrage est uniquement applicable à la configuration globale.

Onglet Sources

L'onglet Sources permet de spécifier les chemins de recherche de jGRASP. Cet onglet n'est utile que lors de la programmation en Java. Il ne sera donc pas détaillé ici.

5. Configuration de l'impression

La configuration de l'impression peut également se faire de 3 manières différentes : globale, par projet ou par fichier.

Pour accéder à cette fenêtre de configuration, aller sous **Settings/Print Settings/Global**.

La liste déroulante Units permet de choisir l'unité de mesure. Choisir mm.

Onglet Common

Cet onglet (Figure 9) permet de définir les dimensions de la feuille de papier.

Le bouton Standard Paper Sizes permet d'obtenir une liste prédéfinie de formats de feuilles (A4 ou letter). Choisir A4.

Si le format voulu n'existe pas dans le bouton précédemment cité, il y a possibilité de spécifier les dimensions d'impression dans les cases Paper Width et Paper Height



Figure 9 : Fenêtre de configuration de l'impression
Onglet Common

Les cases Horizontal et Vertical DPI permettent de régler la résolution de l'imprimante. Laisser les valeurs par défaut.

Onglet CSD/Messages

Cet onglet (Figure 10) permet de régler les détails de l'impression. Ci-dessous, le détails des options :



Figure 10 : Fenêtre de configuration de l'impression
Onglet CSD/Messages

- Color : Si cette option est activée, l'impression se fait en couleur. Si l'imprimante est une noir/blanc, le listing sera imprimé dans un dégradé de gris.
- Filename Header : Si cette option est activée, le nom du et le chemin complet du fichier est affiché en haut de page. **Activer cette option.**
- Landscape : Permet d'imprimer en format paysage. Cette option n'est pas pris en compte lors d'une impression native Windows. Dans le cas d'une impression native Windows, cette option doit être changée dans la boîte de dialogue d'impression Windows.
- Left Page First : Si cette option est activée, ainsi que l'option Book Format, la première page a des marges inversées.
- Page Numbers : Permet d'insérer les numéros de pages. **Activer cette option.**
- Book Format : Si cette option est activée, les marges de gauches et de droites sont activées.
- Left / Right / Bottom / Top Margins : Permet de régler les marges d'impression.
- Gutter : Permet de régler l'espace entre les colonnes si plusieurs colonnes sont utilisées.

- Line Spacing : Permet de régler l'espace entre les lignes.
- Font Size (pts) : Permet de définir la taille d'impression des caractères.
- Header : Ce champ permet d'insérer un en-tête sur toutes les pages. Attention, une ligne maximum.
- Font : Cette liste déroulante permet de choisir la police d'impression.

Excepté les options Page Numbers et Filename Header, toutes les options par défaut peuvent être conservées. Néanmoins, il peut être conseillé d'augmenter la taille des caractères (10 par défaut) pour une meilleure lisibilité.

L'onglet UML ne sera pas traité dans ce document.

6.

Utilisation de jGRASP

Pour illustrer le fonctionnement de ce programme, la section suivante, explique pas à pas l'utilisation de jGRASP.

Créer / sauvegarder / charger un fichier

Une fois que l'application est démarrée, cliquer sur **File – New File – Ada 95** (Figure 11) pour créer une nouvelle fenêtre qui permettra d'écrire un programme dans le langage spécifié (Ada 95 dans cet exemple).

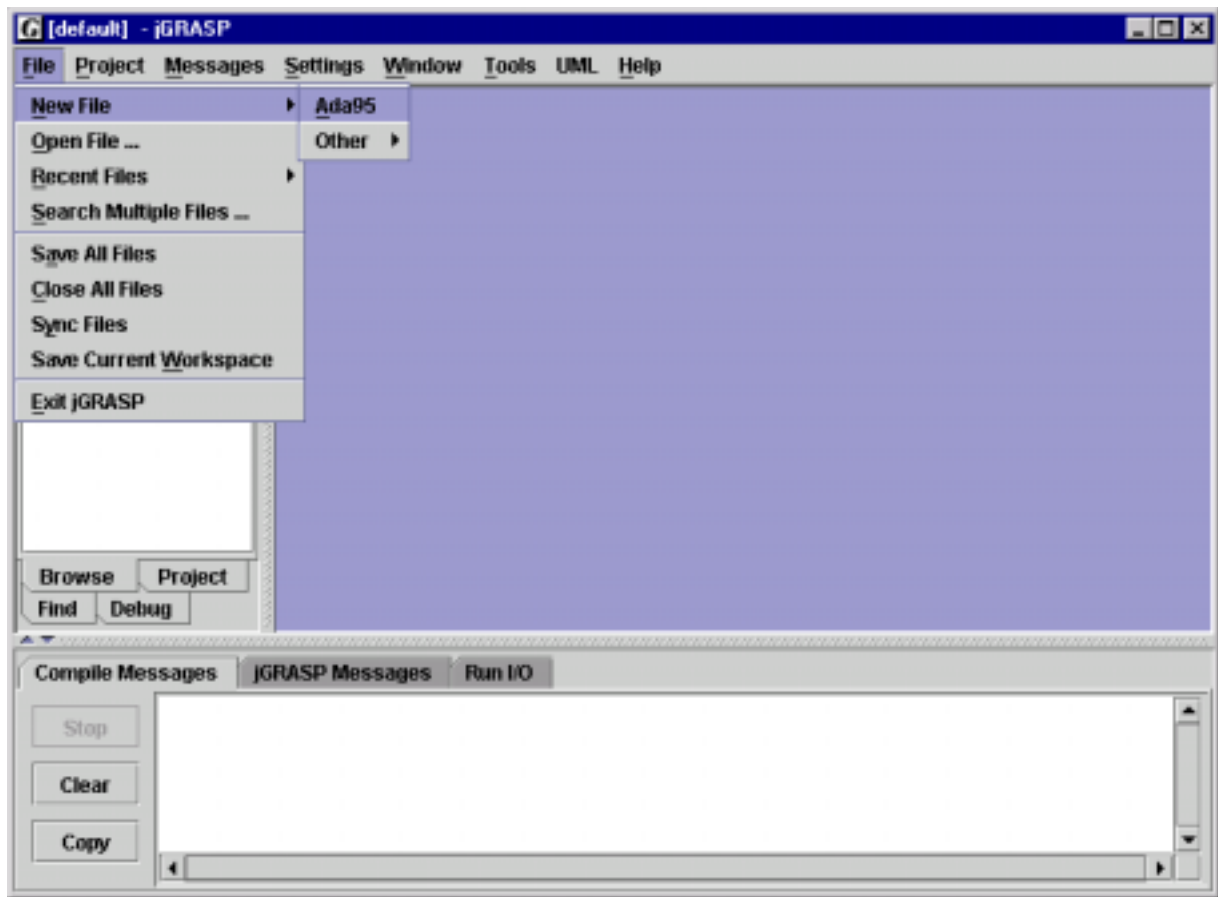


Figure 11 : Création d'un nouveau fichier

Une fois cette opération effectuée, une fenêtre vide portant la mention du langage utilisé est ouverte.

Pour l'exemple, un simple programme, hello.adb, va être créé. Il affiche simplement la phrase « Hello world !!! » sur la console. Les différentes parties du code (mots-clés, chaînes de caractères, commentaires, etc.) sont de couleurs différentes, en fonction de la configuration choisie. La figure 12 illustre la saisie du code.

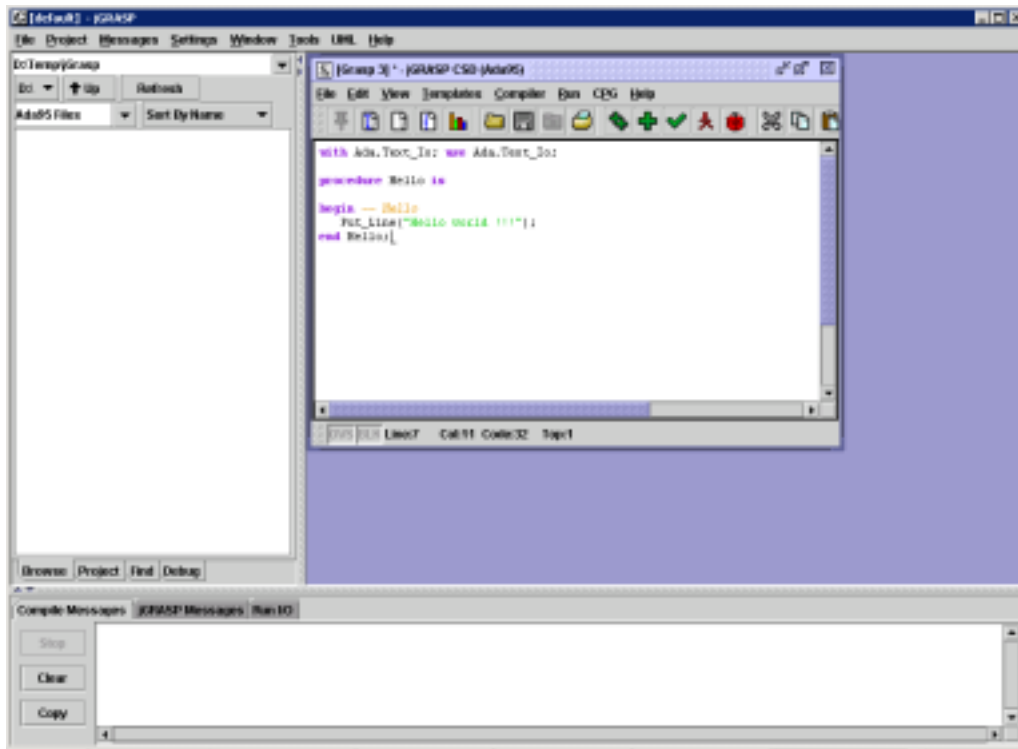


Figure 12 : saisie du code

Attention: La casse du code est entièrement du ressort du programmeur. C'est à ce dernier qu'incombe donc la responsabilité de respecter les conventions Ada 95 concernant la casse utilisée.

La sauvegarde du code se fait comme dans la plupart des applications : par la combinaison de touche **<Ctrl+S>** ou par le menu de la fenêtre de travail **File – Save**.

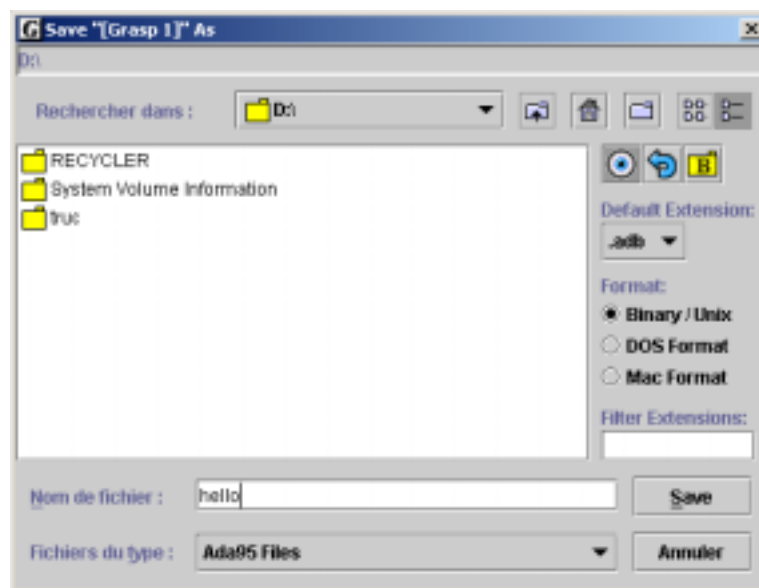



Figure 13 : Sauvegarde d'un fichier

Le menu déroulant Default Extension permet de choisir l'extension dans laquelle le fichier sera sauvegardé. Ces extensions sont définies dans la configuration du langage vue précédemment.

Attention : Pour que la compilation fonctionne correctement le fichier doit porter une extension adb ou ads et non pas ada, ainsi qu'un chemin d'accès sans espaces.

Le chargement d'un fichier peut se faire de deux manières : soit par le menu déroulant **File – Open File ...** ou par le navigateur (onglet Browse, partie gauche de l'interface de jGRASP) via un double-clic sur le fichier à ouvrir. Il est également possible d'ouvrir un fichier depuis la fenêtre de travail (**File – Open ...**). jGRASP gère l'ouverture de plusieurs fichiers dans la fenêtre de travail. Ceux-ci sont accessibles depuis le menu **Window – Nom_du_fichier**. Bien entendu, si les différents fichiers sont visibles, un simple clic sur le fichier désiré permet de le passer au premier plan.

Génération d'une structure CSD

La génération d'une structure CSD peut se faire de trois manières : par le menu **View – Generate CSD**, par le bouton  qui se trouve sur la barre d'outil du fichier ouvert ou par la touche F2.

Si le programme est syntaxiquement correct, l'affichage suivant (Figure 14) est généré.

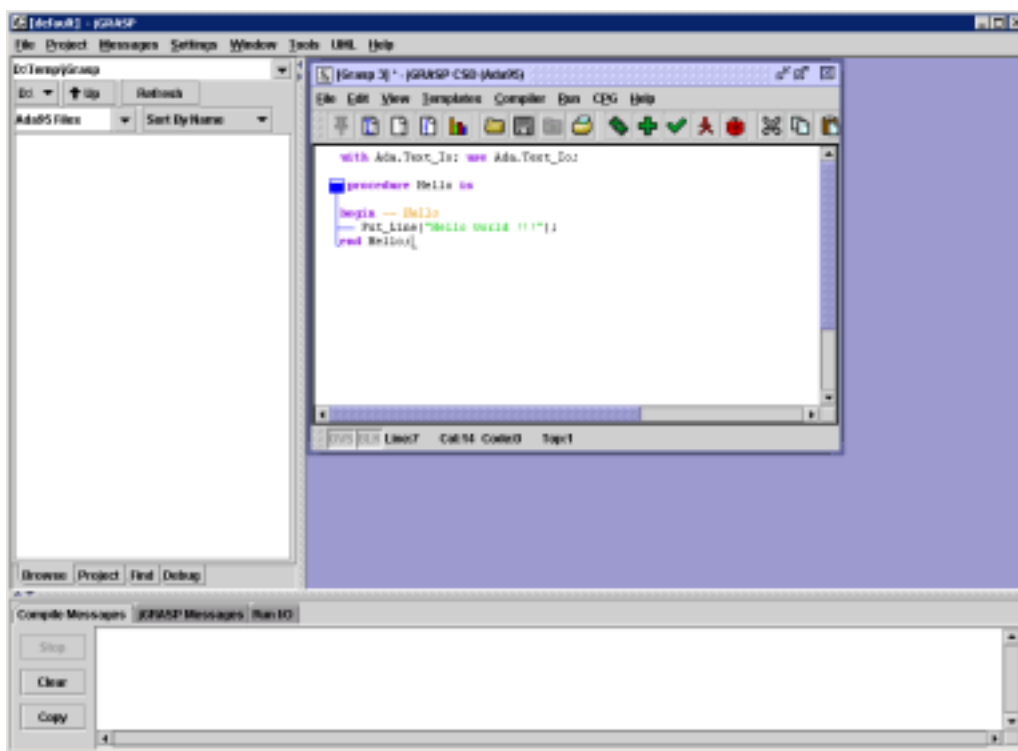


Figure 14 : Affichage CSD

Si la syntaxe n'est pas correcte, un message le signale en indiquant la ligne incriminée (Figure 15).

Il n'est pas possible d'atteindre la ligne incriminée par un double-clic sur le message d'erreur. Utiliser les numéros de ligne pour atteindre l'erreur.

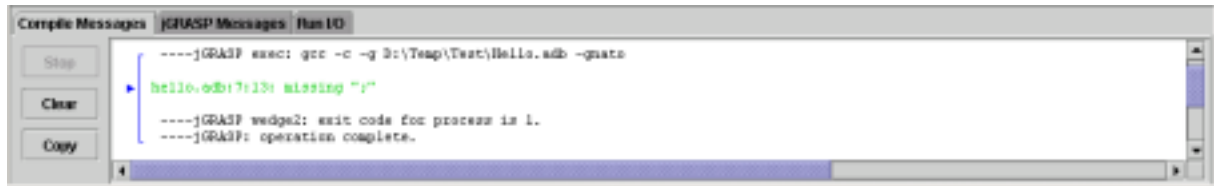










Figure 15 : Erreur de syntaxe

Pour supprimer l'affichage CSD, cliquer sur le menu **View – Remove CSD** ou sur le bouton  ou enfin via la combinaison de touches <Maj+F2>.

NOTE : La suppression de cet affichage peut générer une mauvaise indentation du code.

Dans le tableau ci-dessous se trouve une brève description des principaux symboles.

 <code>package body Outils is</code> <code>end Outils;</code>	Paquetage
 <code>procedure test is</code> <code>begin</code> <code> null;</code> <code>end test;</code>	Procédure ou fonction
 <code>loop</code> <code> null;</code> <code>end loop;</code>	Boucle loop, for ou while
 <code>case test is</code> <code> when 1 => null;</code> <code> when others => Null;</code> <code>end case;</code>	Instruction case
 <code>if test then</code> <code> null;</code> <code>else</code> <code> null;</code> <code>end if;</code>	Instruction if, elsif
 <code>exit;</code>	Instruction exit et return
 <code>raise Liste_Vide;</code>	Exceptions

Une des fonctionnalités intéressantes de CSD est la possibilité de réduire de grosses parties de code sous la forme de « répertoires » (Figure 16) à l'aide d'un double-clic sur les formes qui accompagnent les paquetages, fonctions, procédures, instructions de sélections et instructions d'itérations.

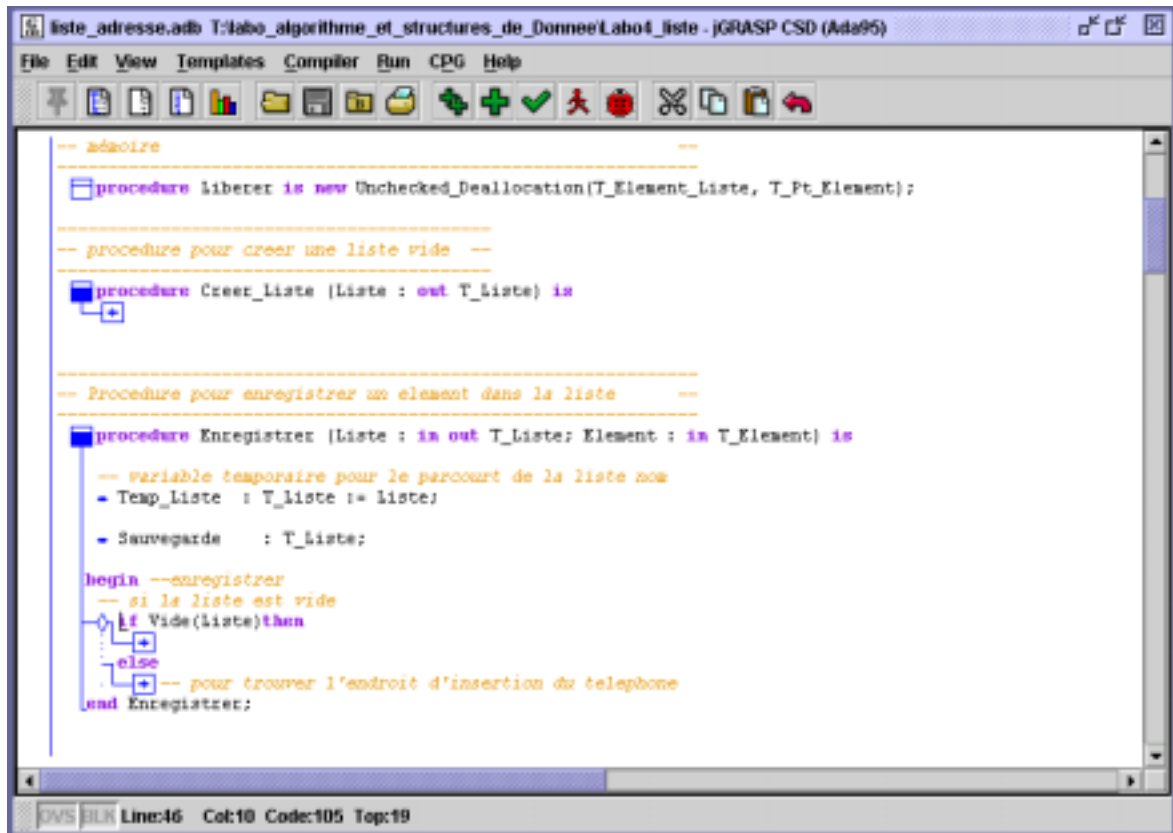



Figure 16 : Réduction du code

Génération d'un graphique CPG

Le graphique CPG permet de visualiser la complexité d'un code source à l'aide d'un graphe. Ce graphe fournit, pour chaque ligne de code, une estimation de sa complexité sous la forme d'une barre verticale (Figure 17). Plus la barre est élevée, plus la ligne qui s'y rapporte est considérée comme complexe. Pour générer ce graphe, cliquer sur le bouton  de la barre d'outil du fichier ou sur le menu **CPG – Generate CPG**.

Cette fonctionnalité est disponible uniquement pour les langages Ada et Java.

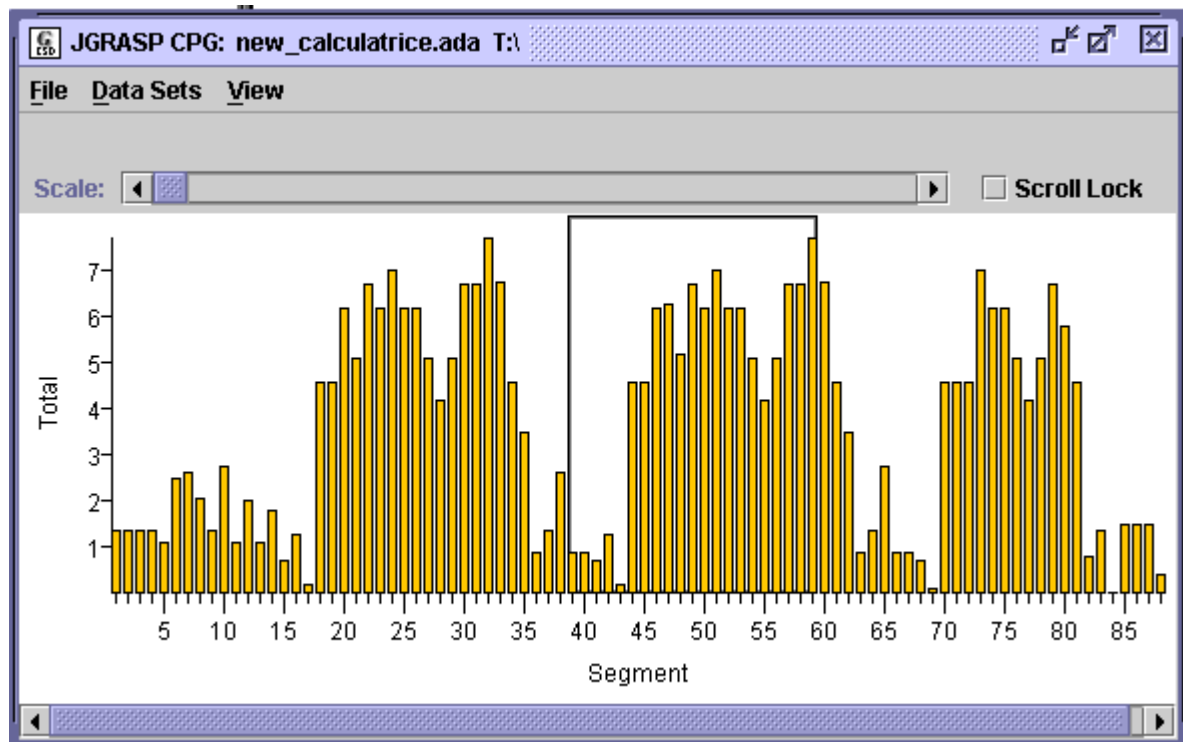



Figure 17 : Graphe CSD


Numérotation des lignes

Il est possible d'afficher le numéro des lignes. Ceux-ci peuvent être pratiques, pour localiser une erreur lors de la compilation par exemple. Les numéros sont générés en cas de clic sur le menu **View – Line Numbers**, par la combinaison de touches

<Ctrl+L> ou par un clic sur le bouton  de la barre d'outils du fichier. Pour enlever ces numéros, réutiliser une de ces trois possibilités.


Les numéros de lignes s'incrémentent automatiquement en cas d'insertion d'une nouvelle ligne.


Il existe une possibilité de bloquer les numéros de lignes, c'est-à-dire qu'en cas de retour à la ligne, les numéros ne sont pas incrémentés. Pour ce faire, trois possibilités : clic sur le menu **View – Freeze Numbers**, La combinaison de touches

<Ctrl + K> ou par un clic sur le bouton  de la barre d'outils du fichier. Pour désactiver cette option, réutiliser une de ces trois possibilités.

Compilation d'un programme

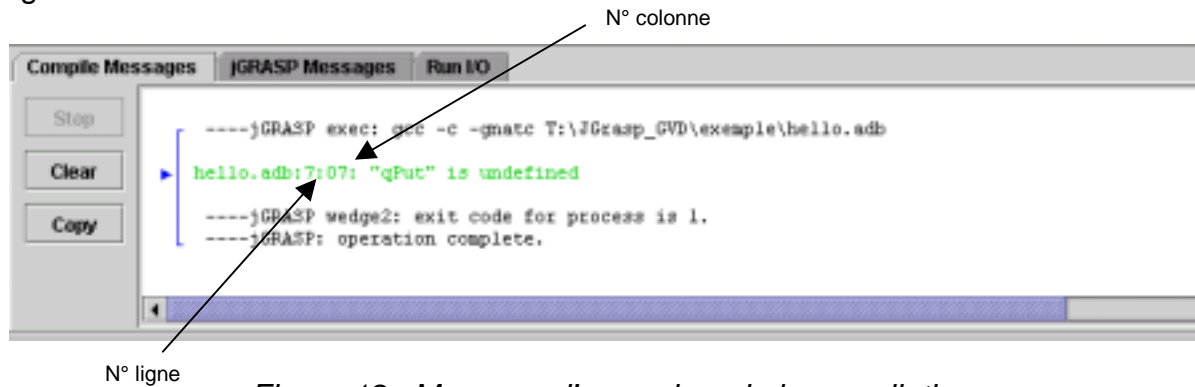
La compilation d'un programme, pour autant que la configuration aie été faite correctement (Voir page 7), se fait assez simplement. La seule compilation peut se faire de trois manières : par un clic sur le menu **Compiler – Compile**, par la pression

de la combinaison de touche <Ctrl+B> ou enfin par un clic sur le bouton  de la barre d'outils du fichier.


L'édition de liens (création de l'exécutable) se fait, quant à elle, par un clic sur le menu **Compile – Compile and Link**, par la combinaison de touche <Ctrl+N> ou par un clic sur le bouton  de la barre d'outils du fichier.

Le résultat de la compilation apparaît dans la fenêtre de dialogue Compile Messages.

En cas de message d'erreur (Figure 18), le fait de cliquer sur le message permet pas de se rendre sur la ligne où se trouve l'erreur. Il fournit uniquement le numéro de ligne et de colonne.




Le fait de compiler un fichier le sauvegarde automatiquement.

Le bouton  de la barre d'outils permet de faire une vérification syntaxique du code.

Le bouton Stop permet d'arrêter un programme en cours de compilation, le bouton Clear d'effacer le contenu de la fenêtre et le bouton Copy de copier le texte sélectionné.

Exécution du programme

Après avoir compilé avec succès le fichier, il ne reste plus qu'à l'exécuter pour vérifier son bon fonctionnement. Un fichier ne peut être exécuté que si le fichier a été compilé et lié avec succès.

Toujours trois manières de procéder : Clic sur le menu **Run – Run** du menu du fichier, combinaison de touches <Ctrl+R> ou clic sur le bouton  de la barre d'outils du fichier.

Quand le programme est lancé, le résultat s'affiche dans fenêtre de dialogue Run I/O (Figure 19).

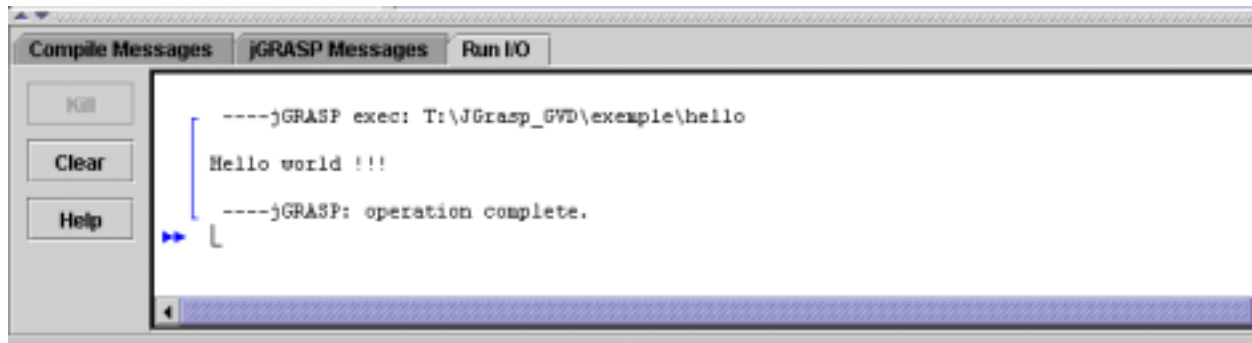


Figure 19 : Entrée/sortie de l'exécution d'un programme

Le bouton Kill permet d'arrêter le programme en cours d'exécution et le bouton Clear d'effacer le contenu de la fenêtre.

Il est également possible de mettre cette fenêtre en plein écran ou de la réduire à l'aide des deux petites flèches situées au-dessus de l'onglet Compile Messages.

Déverminage du programme à l'aide de GVD

GVD est un dévermineur graphique qui a le gros avantage de permettre une visualisation des différentes structures de données qui sont incluses dans le programme. Cette annexe est un survol de GVD et part du principe que l'utilisateur est déjà familiarisé avec l'utilisation d'un dévermineur.

Installation

Le téléchargement de GVD se fait depuis le site <http://libre.act-europe.fr/gvd> et est gratuit.

Une fois le téléchargement effectué, il suffit de lancer l'exécutable `gvd-<num_version>.exe` et de spécifier le répertoire d'installation.

Pré-requis à l'utilisation

Pour utiliser GVD, il faut avoir, d'une part à disposition le fichier exécutable de l'application à déverminer mais également, les fichiers sources contenant les informations de déverminage.

Ces informations sont contenues dans les fichiers `b~<nom_source>.ads` ou `b~<nom_source>.adb` suivant l'extension. Ces fichiers sont générés à la compilation.

Par défaut, ces fichiers ne sont pas générés. Pour ce faire, il suffit de cocher l'option **Compiler – Debug Mode** dans le menu de la fenêtre de travail de jGRASP.

Configuration de jGRASP pour l'utilisation de GVD

1. Ouvrir les options de compilation: **Settings – Compiler Settings – Global**
2. Sélectionner un des environnements proposés dans la liste puis cliquer sur le bouton Copy
3. Saisir dans la première zone de texte, un nom d'environnement (ici gnat 3.14p)
4. Insérer à la ligne Debug, colonne Command, « gvd %MAIN_PATH%SEP%main_base » (sans les «») (Figure 20)
5. Ajouter à la ligne Debug, colonne Directory, « répertoire_d'installation_gvd\bin » (sans les «»).
6. Cliquer sur Save puis OK .

GVD peut maintenant être utilisé avec jGRASP

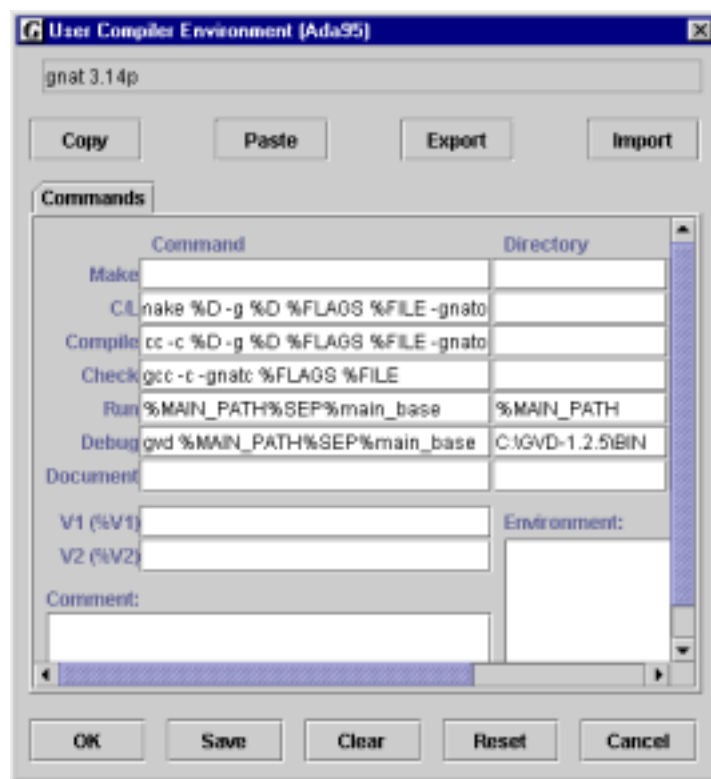



Figure 20 : Configuration pour l'utilisation de GVD

IMPORTANT : A noter ci-dessus (Figure 20) la présence de l'option **gnato** au terme des lignes C/L et Compile, colonne Command. Cette option ne figure pas par défaut et il faut la rajouter pour que le compilateur puisse effectuer correctement les vérifications de dépassement numérique sur les entiers (*numeric overflow*).

Utilisation de GVD

Pour lancer GVD depuis jGRASP, cliquer sur le bouton  de la barre d'outils du fichier. La fenêtre suivante apparaît (Figure 21)

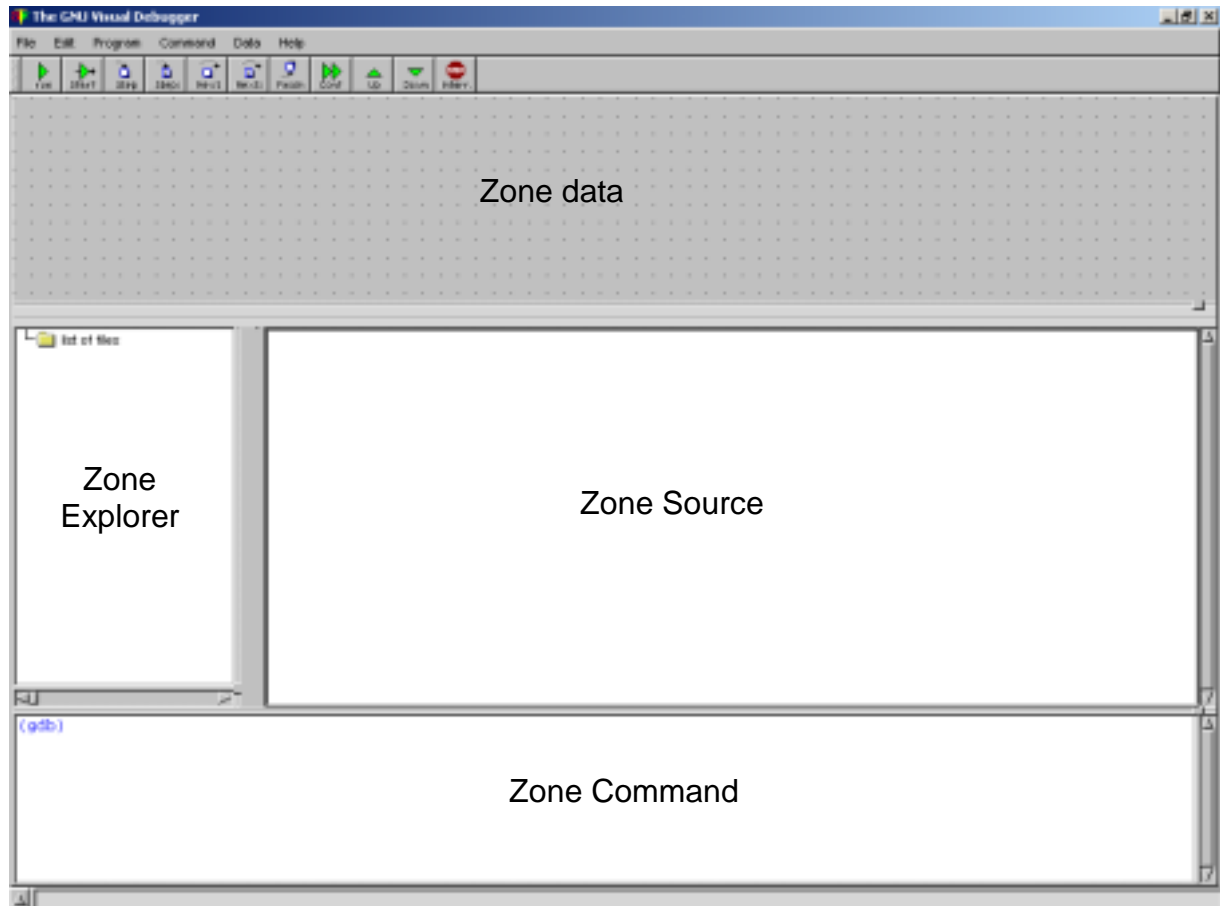


Figure 21 : Interface de GVD

La zone Data contient l'affichage des structures de données.

La zone Explorer permet de naviguer entre les différents éléments du code (paquetages, fonctions, procédures, etc...).

La zone Source contient le code source et permet de suivre le débogage.

La zone Command affiche les résultats (input du programme), les événements relatifs au débogage et permet de saisir des commandes à la main.

Insertion des points d'ancrages


Pour insérer des points d'ancrages, il suffit d'aller à la ligne désirée dans la zone source, puis deux manières de procéder :


- Clic sur le bouton de droite puis Set Breakpoint on Line XX.
- En cliquant sur les points bleus qui se trouvent dans la marge de gauche de la zone source.


Pour enlever le point d'ancrage, il suffit d'aller lui cliquer dessus dans la marge gauche de la zone source.

En cours de déverminage, il est toujours possible d'insérer ou de supprimer un point d'ancrage.

Démarrage du déverminage

Après avoir insérer un (ou plusieurs) point d'ancrage, le démarrage du déverminage de l'application se fait par le bouton  de la barre d'outils, située au-dessus de la zone Data. Dès que le dévermineur arrive à un point d'ancrage, il va s'arrêter.

Le bouton  permet de passer à l'instruction suivante. Si l'instruction courante est un appel de fonction/procédure, le dévermineur passe à l'intérieur de cette fonction/procédure.

Le bouton  permet de passer à l'instruction suivante. Ne passe pas à l'intérieur d'une fonction/procédure.

Le bouton  sort de la fonction/procédure.

Le bouton  permet d'aller au prochain point d'arrêt.

Affichage des structures de données

GVD permet, comme tous les débogueurs, de connaître l'évolution des valeurs des variables en cours de débogage. Néanmoins, son gros avantage est de pouvoir visualiser cette évolution. Cette fonctionnalité est particulièrement utile, notamment pour suivre l'évolution de listes dynamiques.

Pour afficher visuellement les variables (Figure 22), il suffit de cliquer sur celle-ci avec le bouton de droite et de sélectionner **Display – Nom_de_la_variable** après le démarrage du débogage.



Figure 22 : Affichage des structures de données

Les figures suivantes donnent des exemples de structures de données affichées à l'aide de GVD.

Liste chaînée :

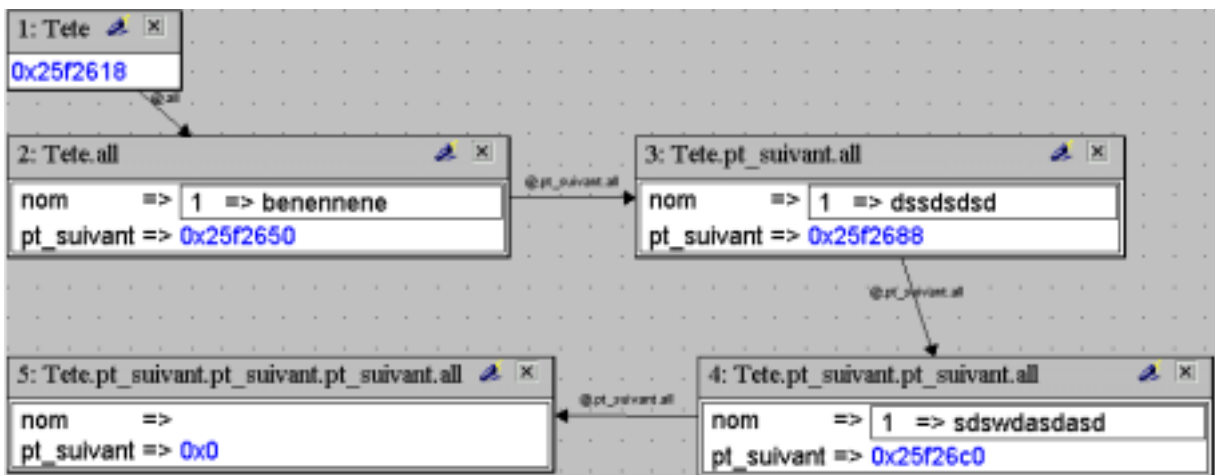


Figure 23: Liste chaînée

Article :

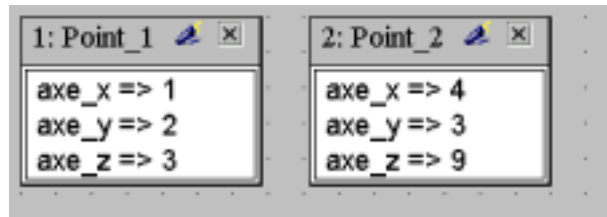
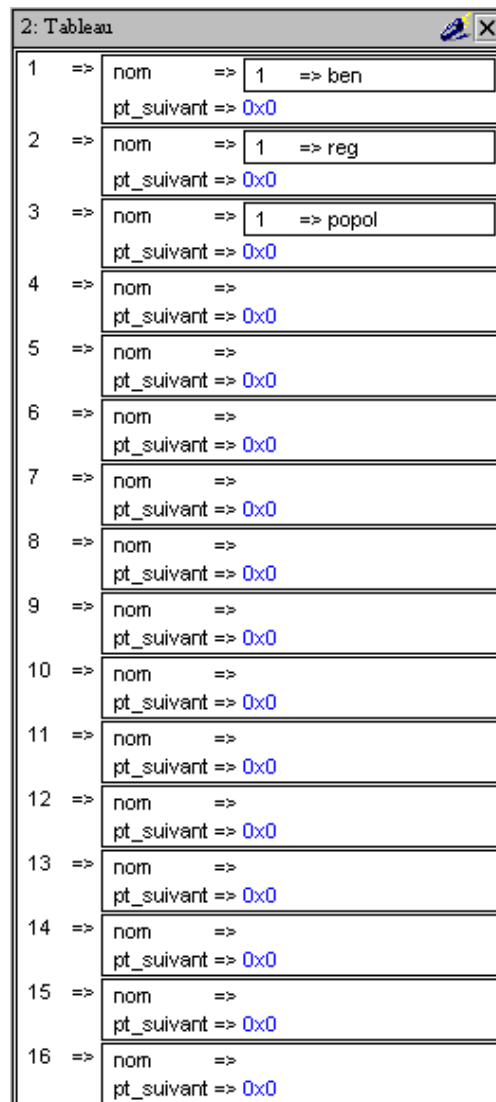


Figure 24: Article

Tableau :

The image shows a window titled '2: Tableau'. It contains a table with 16 rows. Each row has a number from 1 to 16 in the first column, followed by '=>', then a 'nom' field, another '=>', a value in a text box, another '=>', and finally a 'pt_suivant' field with a value in a text box. The values in the 'nom' field are '1 => ben', '1 => reg', and '1 => popol' for rows 1, 2, and 3 respectively. All other 'nom' fields are empty. The values in the 'pt_suivant' field are all '0x0'.

Index	nom	pt_suivant
1	1 => ben	0x0
2	1 => reg	0x0
3	1 => popol	0x0
4		0x0
5		0x0
6		0x0
7		0x0
8		0x0
9		0x0
10		0x0
11		0x0
12		0x0
13		0x0
14		0x0
15		0x0
16		0x0

Figure 25: Tableau

Enfin, Il y a possibilité de séparer la zone Data de la fenêtre principale, pour ce faire : **Edit – Preference**, Onglet data, cocher le bouton Separate Window.

Pour de plus amples informations, se reporter à la documentation fournie avec GVD.

Utilisation des templates

Les templates (Figure 26) fournissent une aide dans la construction syntaxique d'un programme. Ils permettent d'insérer, à l'endroit où se trouve le curseur, une forme prédéfinie utilisée par le langage, telle que des boucles, des tests, des spécifications de fonctions, etc.

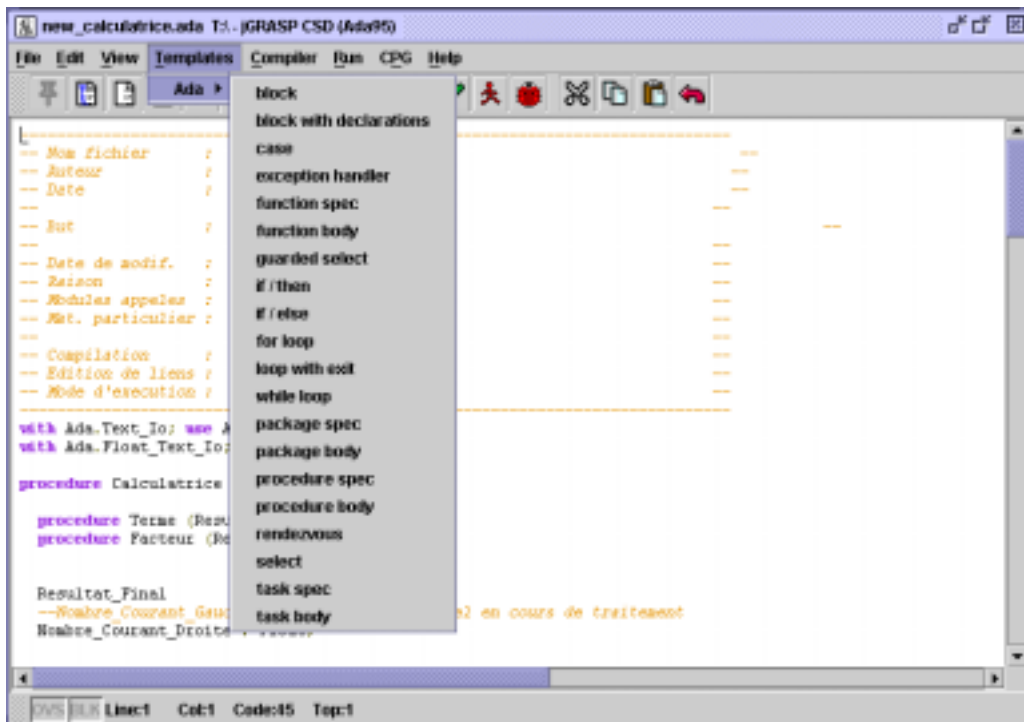


Figure 26 : Utilisation des templates

Les templates sont accessibles par le menu **Template** de la fenêtre de travail.

Ci-dessous, le template de la boucle for inséré sans modification dans un fichier .adb.

```
for INDEX_VARIABLE in VALUE_RANGE loop
    null;
end loop;
```

L'utilisateur peut redéfinir lui-même les templates. Néanmoins il n'existe pas encore d'outils permettant une modification facile de ceux-ci. Cet outil devrait être ajouté dans une prochaine version de jGRASP.

7. Les projets

Les projets, comme dans d'autres outils sont basés sur le regroupement de plusieurs fichiers, qu'ils soient dans le même répertoire ou pas. Lors de la création d'un projet avec jGRASP, toutes les informations relatives à ce projet se trouvent dans un fichier portant l'extension .gpj

Si aucun projet n'est créé, le répertoire qui contient le code source du programme principal fait office de « projet ».

Le menu **Project** de jGRASP contient toutes les opérations relatives aux projets.

Créer un projet

Cliquer sur le menu **Project – New Project...** de la fenêtre de jGRASP pour ouvrir la fenêtre de création de projet (Figure 27).

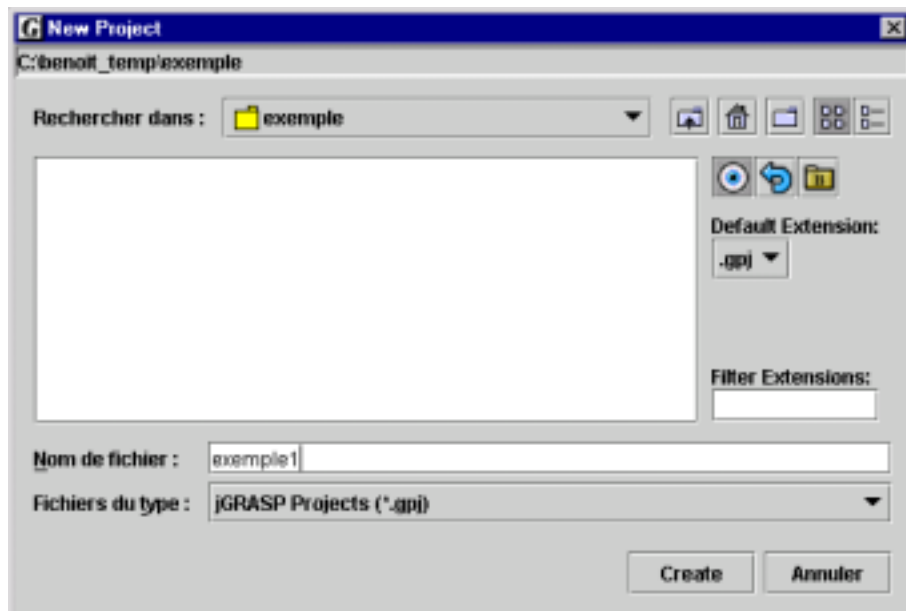


Figure 27 : Création d'un nouveau projet

Aller dans le répertoire voulu et entrer le nom du projet. Il est recommandé de créer le fichier projet dans le même répertoire que celui qui contiendra le programme principal. Appuyer sur Create pour créer le projet.

Ouvrir un projet

Pour ouvrir un projet existant, cliquer sur le menu **Project – Open Project...** Une boîte de dialogue apparaît et permet de sélectionner le projet à ouvrir.

Ajouter un fichier au projet

Le meilleur moyen d'ajouter un fichier à un projet et de passer par le panneau d'exploration. Sélectionner le fichier désiré puis cliquer sur le bouton de droite. Dans le menu qui apparaît, cliquer sur **Add To Project – Relative Path**. Répéter cette action pour tous les fichiers qui doivent être ajoutés au projet.

Pour voir les fichiers contenus dans le projet courant, il suffit de cliquer sur l'onglet **Project** du panneau d'exploration.

Suppression d'un fichier d'un projet

Pour supprimer un fichier d'un projet, aller dans l'onglet Project de l'explorateur puis sélectionner le fichier à supprimer et par un clic sur le bouton de droite cliquer sur **Remove From Project**

Fermeture d'un projet

La fermeture du projet courant se fait par un clic sur le menu **Project – Close Project**.

IMPORTANT: Pour de plus amples informations concernant jGrasp, se reporter à la documentation en ligne fournie avec le logiciel ou au fichier `jgrasp_handbook_6_2_2002.pdf`, téléchargeable sur le même site que jGrasp.

Annexe A : Bugs de jGRASP

- En enlevant l'affichage CSD, certaines parties du code ne sont plus indentées correctement
- En cas de fermeture de jGRASP sans avoir préalablement fermé les fichiers, il arrive que lors de l'ouverture suivante, des caractères de fin de ligne (OD) s'insèrent après chaque ligne de code Pour corriger cela, cliquer sur **File – Close All Files**, puis réouvrir le fichier. Ce caractère aura disparu.