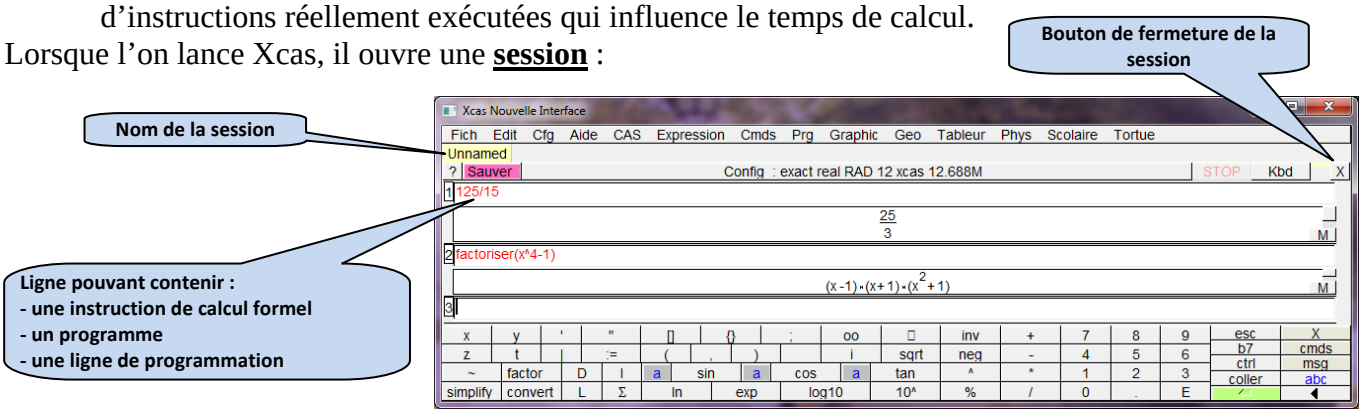
Programmer avec Xcas : version 0.8.6 et 0.9

I. L'environnement de travail de Xcas

Xcas permet d'écrire des programmes, comme n'importe quel langage de programmation.

- C'est un langage fonctionnel. L'argument d'une fonction peut être une autre fonction.
- Il n'y a pas de distinction entre programme et fonction : une fonction renvoie la valeur de la dernière instruction évaluée ou ce qui suit le mot réservé **return**.
- Le langage est non typé. On distingue seulement les variables globales, qui ne sont pas déclarées, et les variables locales, déclarées en début de fonction.
- Xcas est interprété et non compilé. Plus que le nombre de lignes du programme, c'est le nombre d'instructions réellement exécutées qui influence le temps de calcul.

Lorsque l'on lance Xcas, il ouvre une **session** :



Pour supprimer une ligne : cliquer sur son numéro (il passe en vidéo inversée), puis appuyer sur la touche RETOUR ARRIÈRE du clavier.

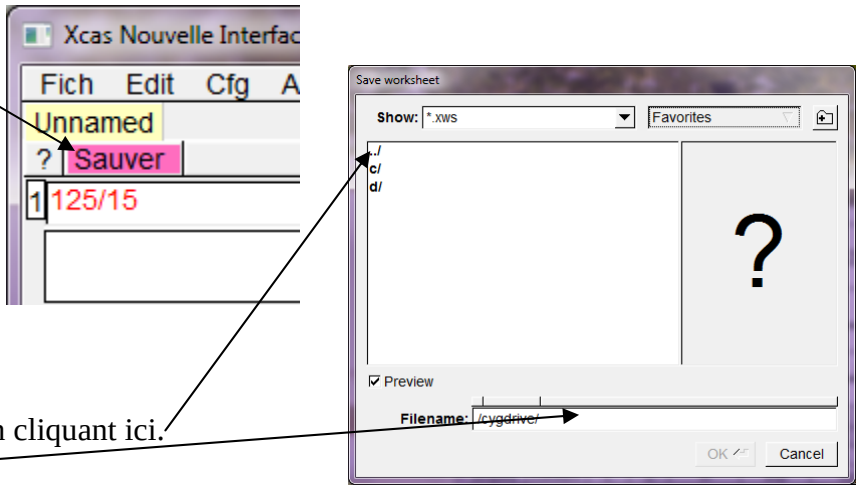
II. Sauvegarder une session

a) Cliquer sur le bouton SAUVER : il est rouge si la session est non vide et non sauvegardée.

b) Dans la fenêtre qui s'ouvre on choisit l'emplacement en définissant le chemin d'accès.
Attention : la longueur du chemin ne doit pas être trop importante sous peine de ne pas pouvoir récupérer les données avec Xcas.

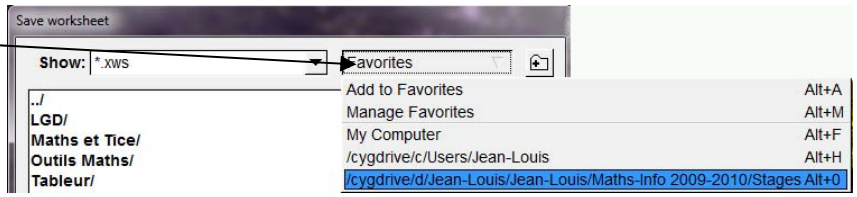
c) On navigue dans l'arborescence en cliquant ici.

d) On tape le nom du fichier ici.



Remarque :

Le bouton FAVORITES permet de stocker le ou les chemins des emplacements de stockage les plus utilisés afin de les retrouver immédiatement.



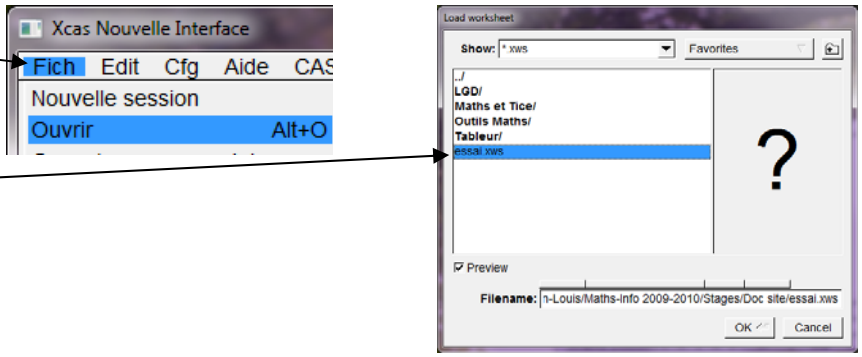
III. Ouvrir une session déjà enregistrée

Fich → Ouvrir

a) On se place dans le dossier contenant la session à ouvrir.

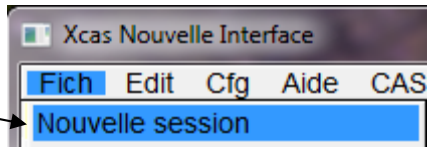
b) On clique sur le nom de la session à ouvrir (fichier d'extension **.xws**).

c) On clique sur **OK.**



IV. Créer une session

Fich → Nouvelle session



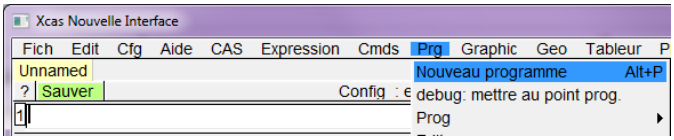
V. Créer un programme

On va illustrer ici la création d'un programme permettant de répondre au problème suivant :

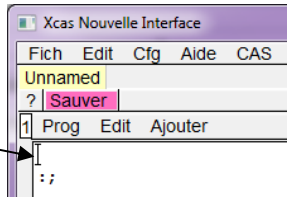
> Un robot est posé au centre d'une table carrée de 90 cm de côté. Toutes les secondes, il effectue un pas de 10 cm dans une des quatre directions.
> En moyenne, combien de temps reste-t-il sur la table ?

- **Pour créer un nouveau programme :**

a) On se place sur une ligne vide de la session.
Prg → Nouveau programme



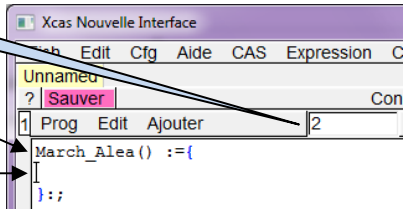
b) Le nouveau programme va être édité sur cette ligne de la session.



Numéro de la ligne sur laquelle se trouve le point d'insertion

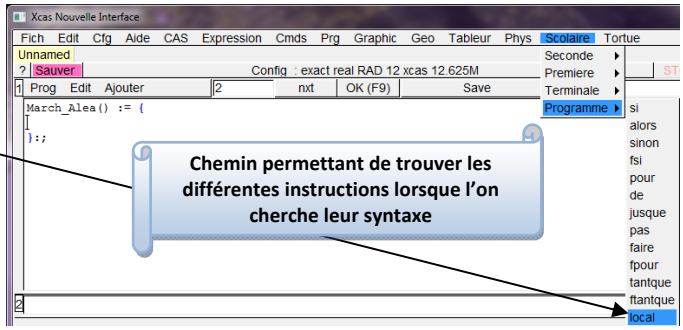
c) On tape le nom du programme (sans espaces) suivi d'une paire de parenthèses (c'est une fonction) et de **:=**.

d) Le programme sera un bloc contenu entre **deux accolades** qui doivent être tapées.



- **Pour définir les variables utilisées :**

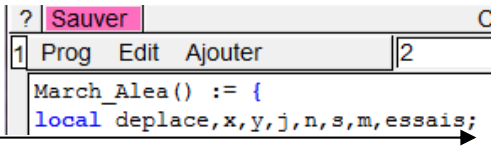
Scolaire → Programme → local ou taper directement **local**



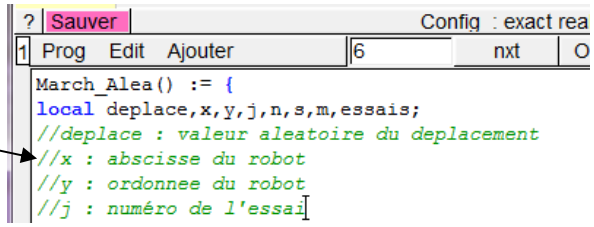
a) A la suite du mot **local** on donne la liste des variables qui seront utilisées séparées par **une virgule** (*si les variables ne sont pas définies comme locales à la procédure, on court le risque que devenant globales, elles récupèrent une valeur définie par une autre instance de Xcas*).

b) On termine la ligne par **un point-virgule**.

c) On appuie sur ENTRÉE pour passer à la ligne suivante.

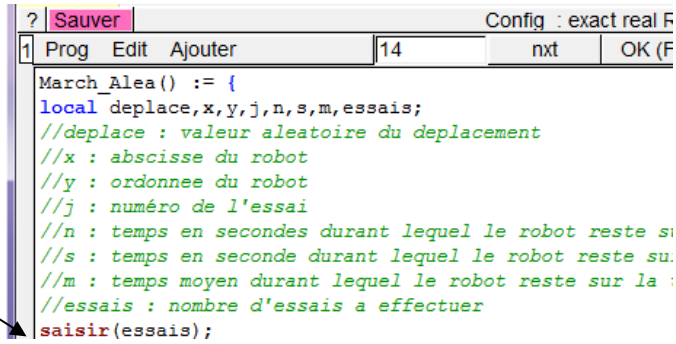


d) On peut rajouter **un commentaire**, permettant de préciser par exemple ici le rôle des différentes variables, en le faisant précéder de deux barres obliques (*un commentaire est ignoré lors de l'exécution du programme*).



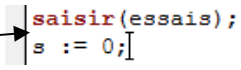
- **Pour demander l'entrée d'une valeur numérique :**

On tape **saisir(essais)** pour que le programme demande à l'utilisateur d'entrer la valeur de la variable **essais**.



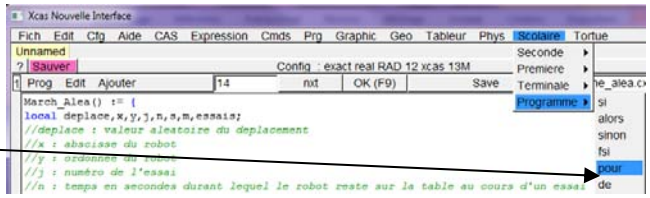
- **Pour initialiser une variable :**

  On lui affecte la valeur voulue (le symbole d'affectation est **:=**).

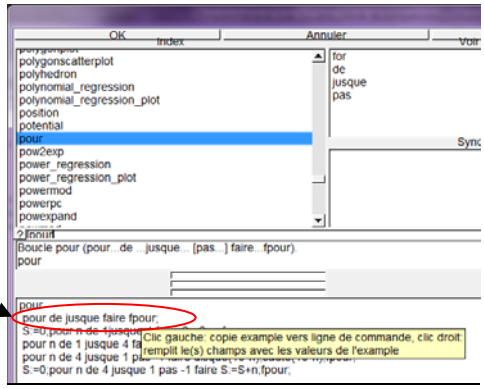


- **Pour créer une boucle « Pour … » :**

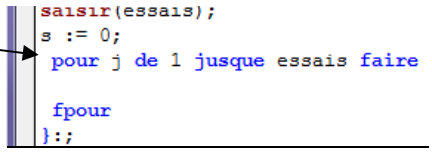
a) On va chercher **l'instruction** (*méthode utile pour débuter, mais il est aussi possible de la taper directement au clavier*).



b) Une fenêtre d'aide s'ouvre alors, un clic sur **l'instruction voulue** permet de l'insérer dans le code du programme.

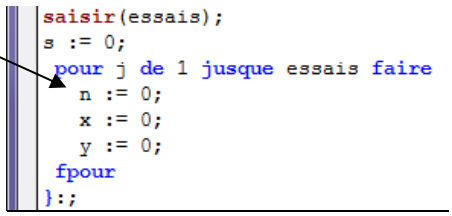


c) On complète l'instruction en passant **fpour** à la ligne afin d'insérer le code dans le bloc.



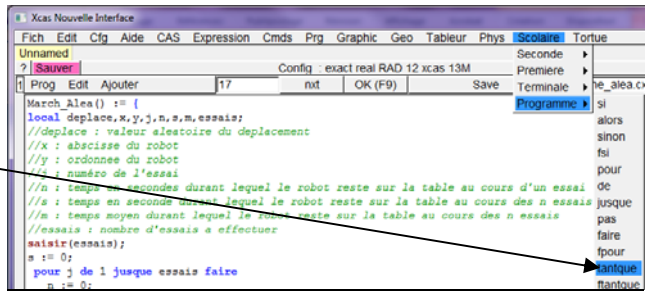
Chaque ligne doit être terminée par un point-virgule.

Pour améliorer la lisibilité du code, les instructions à répéter peuvent être placées entre les deux lignes avec une indentation.

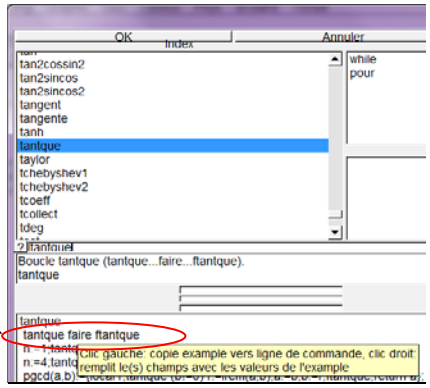


- **Pour créer une boucle « Tant que … »**

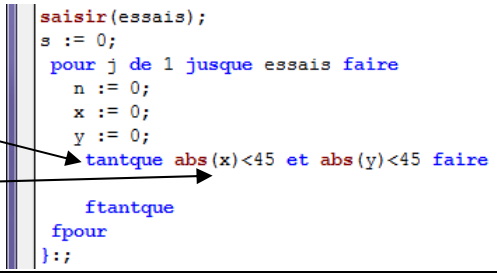
a) On va chercher **l'instruction** (*méthode utile pour débuter, mais il est aussi possible de la taper directement au clavier*).



b) Une fenêtre d'aide s'ouvre alors, un clic sur **l'instruction voulue** permet de l'insérer dans le code du programme.



c) On complète l'instruction en passant **ftantque** à la ligne afin d'insérer le code dans le bloc.



Test d'arrêt de la boucle :
abs renvoie la valeur absolue d'un nombre
et permet de relier deux conditions

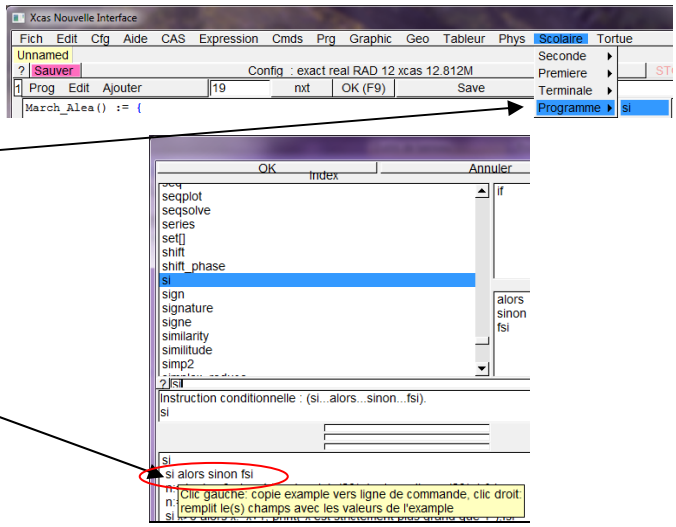
Simulation
alea(0,1) renvoie un nombre au hasard uniformément distribué dans l'intervalle [0 ;1[
floor renvoie la partie entière d'un nombre réel

```
saisir(essais);
s := 0;
 pour j de 1 jusque essais faire
   n := 0;
   x := 0;
   y := 0;
    tantque abs(x)<45 et abs(y)<45 faire
     deplace := floor(4*alea(0,1)+1);
    ftantque
 fpour
};
```

Pour améliorer la lisibilité du code, les instructions à répéter peuvent être placées entre les deux lignes avec une indentation.

- **Pour créer une instruction conditionnelle « Si … Alors … Sinon … » :**

a) On va chercher **l'instruction** (*méthode utile pour débuter, mais il est aussi possible de la taper directement au clavier*).



b) Une fenêtre d'aide s'ouvre alors, un clic sur **l'instruction voulue** permet de l'insérer dans le code du programme.

c) On complète l'instruction en passant **fsi** à la ligne afin d'insérer le code dans le bloc et en supprimant éventuellement le **sinon**.

```
saisir(essais);
s := 0;
 pour j de 1 jusque essais faire
   n := 0;
   x := 0;
   y := 0;
    tantque abs(x)<45 et abs(y)<45 faire
     deplace := floor(4*alea(0,1)+1);
      si alors
      fsi
    ftantque
 fpour
};
```

Tests
On utilise : **== pour** est égal à
!= pour est différent de
< **<=** **>** **>=**

Les instructions à exécuter lorsque la condition est vraie sont placées après **alors** (s'il y en a plusieurs elles sont sur des lignes différentes terminées par un point-virgule). S'il y a des instructions à effectuer lorsque la condition est fausse, elles seront placées après un **sinon**.

```
saisir(essais);
s := 0;
 pour j de 1 jusque essais faire
   n := 0;
   x := 0;
   y := 0;
    tantque abs(x)<45 et abs(y)<45 faire
     deplace := floor(4*alea(0,1)+1);
      si deplace==1 alors x := x+10;
      fsi
    ftantque
 fpour
};
```

- **Pour afficher un résultat :**

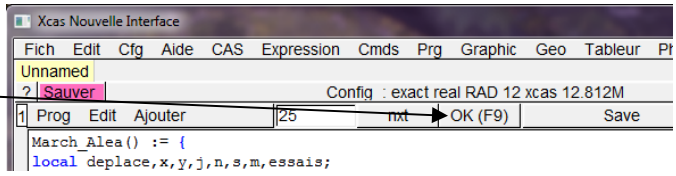
On utilise l'instruction **afficher** qui permet de combiner l'affichage d'une chaîne de caractères (entre guillemets) avec l'affichage de la valeur d'une ou de plusieurs variables.

L'instruction **round** permet d'arrondir la valeur de **m** (à 2 décimales ici). Xcas étant un logiciel de calcul formel renverra autrement la valeur exacte de m.

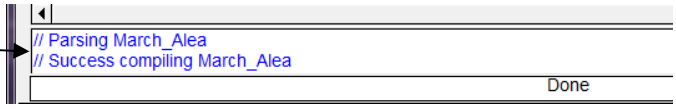
```
saisir(essais);
s := 0;
 pour j de 1 jusque essais faire
   n := 0;
   x := 0;
   y := 0;
   tantque abs(x)<45 et abs(y)<45 faire
     deplace := floor(4*alea(0,1)+1);
     si deplace==1 alors x := x+10;
     fsi
     si deplace==2 alors x := x-10;
     fsi
     si deplace==3 alors y := y+10;
     fsi
     si deplace==4 alors y := y-10;
     fsi
     n := n+1;
   ftantque
   s := s+n;
 fpour
m := round(s/essais,2);
afficher("Au cours de "+essais+" essais, le temps moyen
durant lequel le robot est resté sur la table est de
 "+m+" secondes.");
};
```

- **Exécuter le programme :**

a) Cliquer sur **OK** afin d'interpréter le programme saisi et de le rendre exécutable.



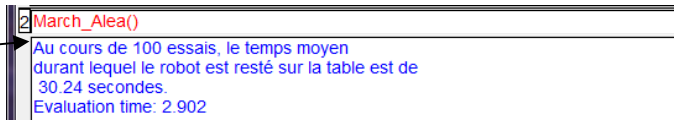
b) Vérifier la présence d'éventuelles erreurs de syntaxe.



c) Taper sur une ligne de commande le nom du programme à exécuter, puis appuyer sur **Entrée**.

d) Une boîte de dialogue s'ouvre demandant de saisir la valeur voulue qui sera ensuite affectée, lors de l'appui sur **OK,** à la variable **essais**.

e) Le ou les affichages demandés apparaissent sous la ligne de commande.

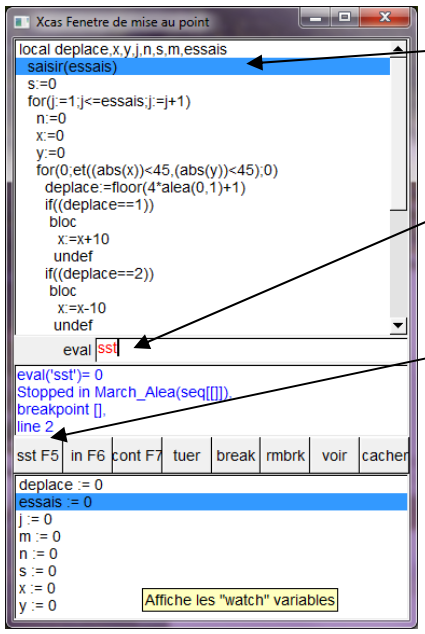


VI. Débugger un programme

Pour utiliser le débuggeur, il faut que le programme soit syntaxiquement correct. Si le programme est syntaxiquement correct mais ne fait pas ce qu'il devrait faire il faut analyser son fonctionnement pour le corriger.
Avec le **débuggeur**, on a la possibilité d'exécuter le programme pas à pas (sst) afin de visualiser les valeurs prises par les différentes variables à chacune des étapes.

On tape sur une ligne de commande : debug(nom _du_programme()).

L'écran du **débuggeur** s'ouvre : il est formé par trois écrans séparés par une ligne **eval** et une barre de boutons sst, in, cont, ...



- dans l'écran du haut, le programme source est écrit.
- dans la ligne **eval**, apparaît l'action en cours par exemple sst.
- pour exécuter le programme pas à pas on utilise le bouton sst : chaque appui sur le bouton sst exécute la ligne courante (celle qui est en surbrillance), met en surbrillance l'instruction suivante et affiche dans l'écran du bas la valeur des différentes variables.
- à la différence du bouton sst, le bouton dans ou in exécute la ligne courante en entrant dans les fonctions définies à l'extérieur du programme et utilisées par celui-ci (cf. document : pour aller plus loin avec XCas).

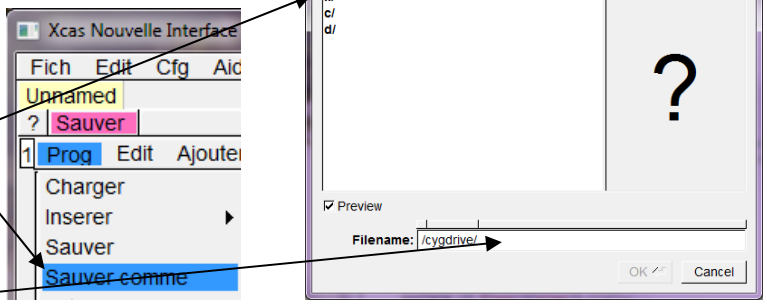
VII. Sauvegarder un programme

a) **Prog → Sauver comme**

b) Dans la fenêtre qui s'ouvre on choisit l'emplacement en définissant le chemin d'accès.
Attention : la longueur du chemin ne doit pas être trop importante sous peine de ne pas pouvoir récupérer les données avec Xcas.

c) On navigue dans l'arborescence en cliquant ici.

d) On tape le nom du fichier ici.



VIII. Exécuter un programme déjà enregistré

- **Pour insérer un programme enregistré préalablement :**

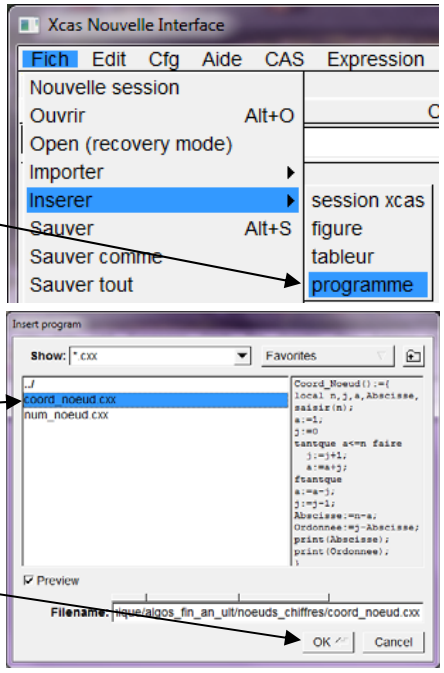
a) Se placer sur une ligne de commande vide de la session.

b) **Fich → Inserer → programme**.

c) On se place dans le dossier contenant le programme à insérer.

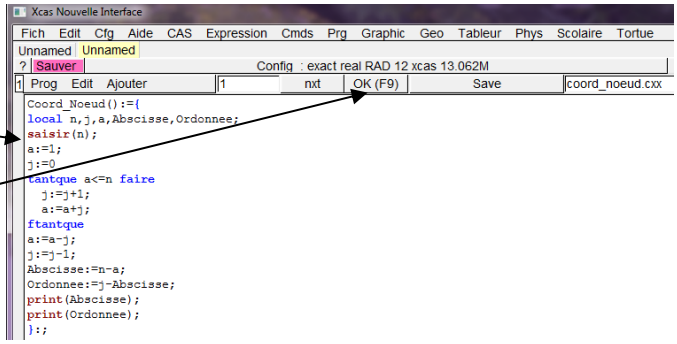
d) On clique sur le nom du programme.

e) On clique sur **OK**.

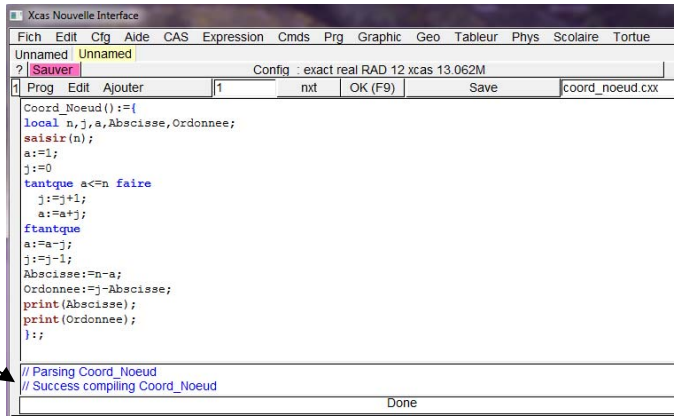


f) Le code du programme apparaît dans la session Xcas.

g) On clique sur **OK** pour interpréter le programme afin de pouvoir l'exécuter.



h) On vérifie ici qu'il n'y a aucun problème de syntaxe.



i) On tape sur une nouvelle ligne de la session le nom du programme.

j) On le lance par appui sur la touche **Entrée**.