

ALGORITHMIQUE

Le but de ces TP est de s'initier à l'algorithmique au travers de l'outil CompAlgo v2. Les exercices mettent en œuvre les différents aspects de l'algorithmique vus en cours.

Note : Ces supports contiennent un grand nombre d'exercices. Les exercices présents dans la section « Pour aller plus loin... » vous permettent d'approfondir les notions, de réviser...

Ordonnancement des TPs :

Les TP 0 – 1 – 2 doivent être terminés en 2 séances,

Les TP 3 – 4 doivent être terminés en 2 séances,

Le TP 5 doit être terminé en 1 séance,

Le TP 6 doit être terminé en 1 séance,

Le TP 7 est un mini-projet destiné à la révision.

0 - Initiation à CompAlgo

Répertoire : INITIATION°

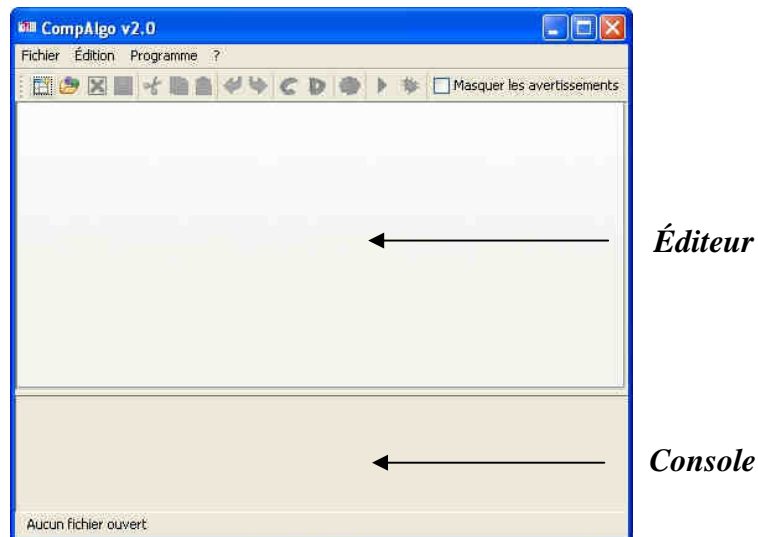
CompAlgo est un outil permettant de vérifier la syntaxe et d'exécuter des algorithmes. Cet outil va vous servir pour tous les exercices qui vous sont proposés.

De tout autre endroit, suivez le lien suivant dans un navigateur (Java est obligatoire sur votre ordinateur) :

<http://www.iut-tlse3.fr/moodle/course/view.php?id=1528>

Il s'agit d'un cours public nommé « Compilateur Algorithmique » du département Informatique de Toulouse. Pour exécuter CompAlgo suivre les explications fournies.

La fenêtre principale de l'éditeur est divisée en deux parties : en haut l'éditeur où vous saisirez votre algorithme et la console en bas qui récapitule les opérations en cours (message de fin de compilation par exemple) ou exécution d'un programme. La saisie des valeurs se fera également au travers de la console.



Les options du menu sont réparties en trois grandes catégories :

- **Fichier** : Rassemble les items liés à la création, à l'ouverture, à la fermeture et à la sauvegarde des fichiers (l'extension des algorithmes sera par convention `.algo`). Dans ce menu est également présente l'item permettant de quitter l'application.
- **Édition** : Rassemble les items liés à l'édition de l'algorithme dans l'éditeur.
- **Programme** : Contient l'item lié à la compilation de l'algorithme dans la fenêtre active (F11).
- **?** : Quelques informations liées au compilateur.

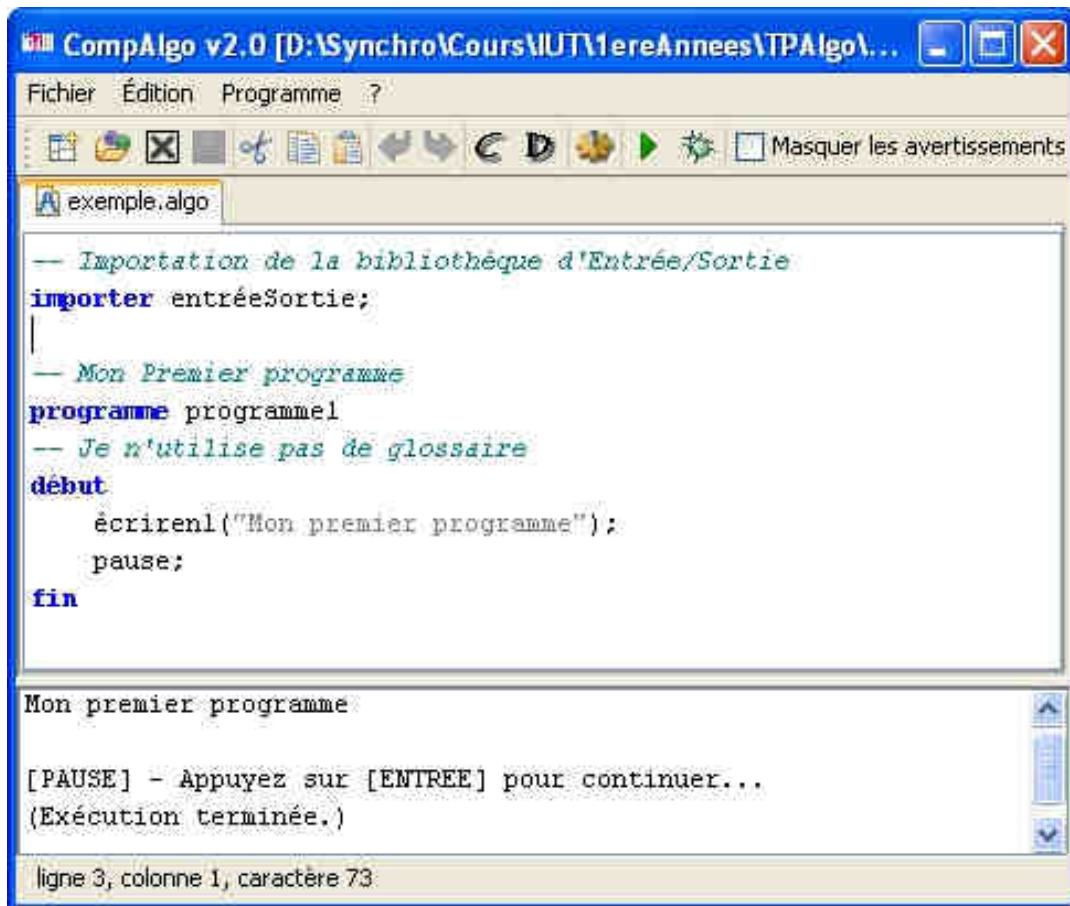
Dans la suite, nous allons utiliser l'éditeur pour compiler et exécuter un premier algorithme simple.




Avant de pouvoir utiliser le compilateur vous devez obtenir les bibliothèques que vous allez utiliser dans vos programmes. Pour cela, copier tous les fichiers « `.algo` » et « `.spec` » que vous aurez téléchargés depuis la page du cours du compilateur algorithmique (décompresser l'archive téléchargée). **Il vous faudra copier les fichiers (`.algo` ou `.spec`) nécessaires dans les différents sous-répertoires où se situent vos sources.**

Tapez ensuite dans l'éditeur le programme de la figure ci-dessous. Pour cela, créez tout d'abord un nouveau fichier algorithme (Fichier/Nouveau ou Ctrl+N). Remarquez que les mots-clés du langage algorithmique sont signalés en gras et colorés en bleu.

ATTENTION à la lecture du programme :

- « **Programme1** » (Un)
- « **écrire**n1 » (L minuscule)



- Sauvegardez votre programme (Fichier/Enregistrer ou Ctrl+S) en indiquant un nom (respectez les conventions d'écriture des programmes en langage algorithmique \Rightarrow l'extension du fichier doit être .algo. Les fichiers .spec concernent les fichiers d'entêtes des bibliothèques).
- Compilez le programme grâce à l'option Programme/Compiler (F11) ou en cliquant sur l'icône correspondant . La console indique si votre algorithme est erroné c'est-à-dire s'il possède des erreurs. Si tel est le cas corrigez les erreurs indiquées dans la console et recompilez le programme.
- Dès que toutes les erreurs sont corrigées et que vous obtenez le message « Compilation réussie ! », exécutez le programme par le menu ou par l'icône correspondant  par exemple.
- Pour pouvoir suivre l'exécution du programme algorithmique un débogueur a été introduit dans la version 2 de Compalgo. Pour y accéder, cliquez notamment sur le bouton  ou appuyez sur F7.

1 - Dessinez c'est gagné !

Répertoire : TRACE

Le but de ce TP est d'utiliser les notions algorithmiques appliquées à une machine à tracer.

Une machine à tracer est définie par :

- un stylet,
- une orientation (ou points cardinaux),
- une unité de longueur (u),
- une surface de travail carrée de 100 unités de côté,
- une position centrale (origine) noté O,
- cinq compteurs nommés c1, c2, c3, c4 et c5.

À cette machine à tracer est associée un ensemble de sous-programmes qui se trouvent dans les bibliothèques présentes dans les fichiers `entréeSortie.spec`, `graphique.spec`, `traceur.spec` et `traceur.algo` (corps de la bibliothèque). Ces fichiers doivent être présents dans le répertoire contenant vos sources. Ces deux bibliothèques doivent être importées dans le programme grâce à l'instruction `importer` (ex : `importer traceur ;`). Ces bibliothèques contiennent les sous-programmes suivants :

```
Fichier graphique.spec : contient les sous-programmes permettant de
manipuler une table traçante pour réaliser des dessins.
-- Rompt le contact entre le stylet et la surface de travail
-- Si le stylet est déjà levé, l'opération est sans effet
procédure leverStylet ;

-- Établit le contact entre le stylet et la surface de travail
-- Si le stylet est déjà baissé, l'opération est sans effet
procédure baisserStylet ;

-- Déplace le stylet à la position centrale de la surface de
-- travail sans modification de sa hauteur
procédure centrerStylet ;

-- Déplace le stylet de 1 unité dans la direction cardinale
-- courante sans modification de sa hauteur
procédure déplacerStylet ;

-- Modifie la direction cardinale courante de 90° vers la
-- droite
procédure pivoterDroite ;

-- Modifie la direction cardinale courante de 90° vers la
-- gauche
procédure pivoterGauche ;

-- Oriente le stylet vers le nord (haut de la surface)
procédure orienterNord ;
```

```
-- Initialise la surface de travail et l'affiche à l'écran.
-- L'appel à cette procédure est obligatoire avant l'appel aux procédures
-- citées ci-dessus
procédure ouvrirGraph ;

-- Libère la surface de travail et ferme la fenêtre correspondante.
procédure fermerGraph ;
```

Fichier traceur.spec : Contient les sous-programmes permettant de manipuler des compteurs entiers

```
-- compteur prend pour valeur initiale valeur
procédure initialiser (sortie compteur <Entier>, entrée valeur <Entier>) ;

-- compteur devient égal à compteur + 1
procédure ajouterUn (màj compteur <Entier>) ;

-- compteur devient égal à compteur + valeur
procédure ajouterN (màj compteur <Entier>, entrée valeur <Entier>) ;

-- compteur devient égal à compteur - 1
procédure soustraireUn (màj compteur <Entier>) ;

-- retourne VRAI si valeur1 = valeur2 et FAUX sinon
fonction valeursEgales (entrée valeur1 <Entier>, entrée valeur2 <Entier>)
retourne <Booléen> ;
```

IMPORTANT :

Pour chaque exercice, l'état initial de la machine est indéterminé. On admettra donc que :

- * la position du stylet est indéterminée,
- * la direction initiale du stylet est indéterminée,
- * le stylet peut être soit levé, soit baissé,
- * on utilisera uniquement le nom des compteurs fournis (c1, c2, c3, c4, c5).

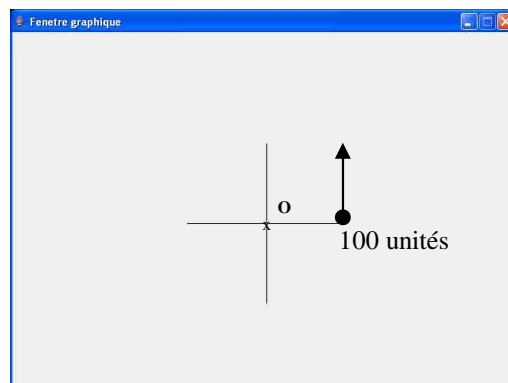
À l'issue de chaque tracé le stylet devra être positionné à l'origine et levé.

Toute fenêtre graphique ouverte doit être fermée à la fin de l'algorithme.

Attention donc aux initialisations ! Pensez-y !

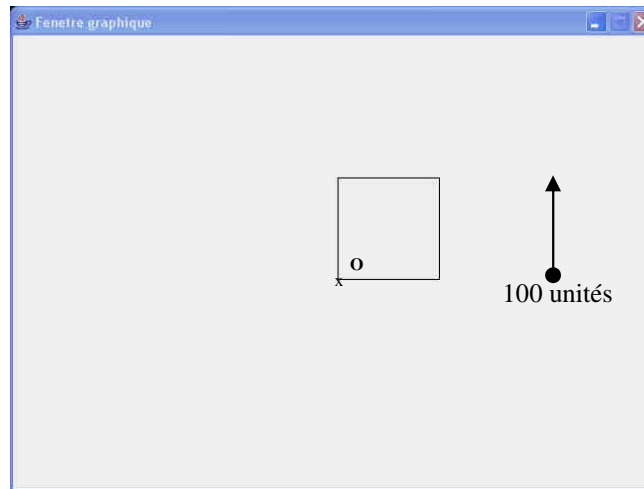
1) Tracé d'une croix simple

✎ Écrire l'algorithme permettant de tracer une croix centrée à l'origine de la surface de travail. Chaque branche aura 100 unités de longueur. *Note : la lettre o désigne le centre de la surface.*



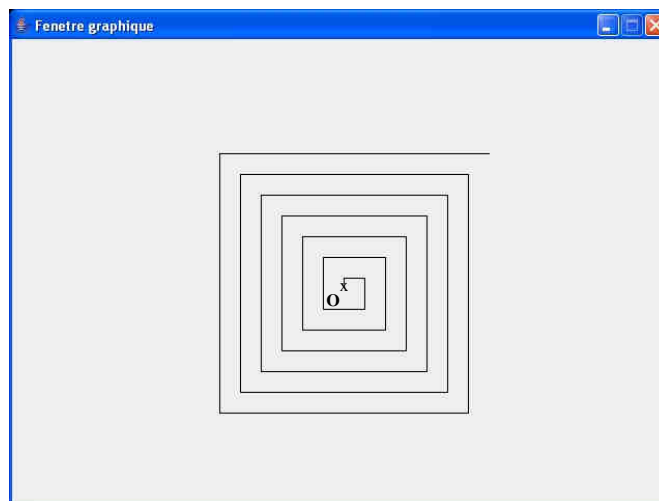
2) Tracé d'un carré

✎ Écrire l'algorithme permettant de tracer un carré dont le coin bas gauche est situé à l'origine de la surface de travail. Chaque côté mesure 100 unités.



3) Tracé d'une spirale

✎ Écrire l'algorithme permettant de tracer une spirale dont la dernière branche mesure 260 unités. Le départ de la spirale est situé à l'origine. Les différentes branches mesurent (en unités) 10, 20, 30, 40 ... Notez que la spirale que vous obtenez doit être identique à celle de la figure.

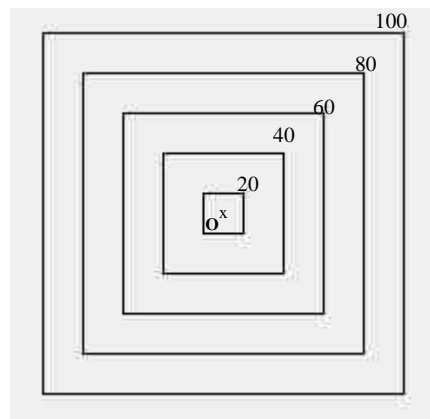


2 - Dessinez c'est gagné ! Le RETOUR !

Répertoire : TRACE

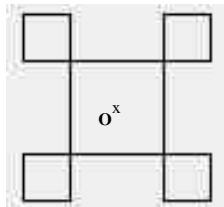
1) *Tracé de carrés gigognes*

- ✎ Écrire l'algorithme permettant de tracer la figure ci-dessous sachant que :
 - le plus petit carré fait 20 unités de côté,
 - le petit carré est centré sur l'origine,
 - chaque carré est plus grand de 20 unités que son précédent.
- ✎ Reprendre l'algorithme précédent pour introduire des sous-programmes tels que :
 - tracer un segment d'une taille donnée en fonction de l'orientation actuelle
 - tracer un carré d'une taille donnée
 - ...



2) *Tracé de 4 carrés aux coins d'un grand carré*

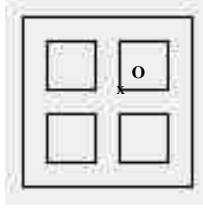
- ✎ Écrire l'algorithme permettant de tracer la figure ci-dessous sachant que :
 - les petits carrés font 20 unités de côté,
 - le grand carré centré fait 40 unités de côté.
- L'algorithme doit favoriser la répétition de motifs.



3) *Tracé de 4 carrés dans un carré*

- ✎ Écrire l'algorithme permettant de tracer la figure ci-dessous sachant que :
 - les petits carrés font 20 unités de côté,
 - le grand carré fait 70 unités de côté,

- le petit carré en haut à droite possède son coin bas et gauche centré en O,
 - l'espace entre les carrés vaut 10 unités.
- L'algorithme doit favoriser la répétition de motifs.



DANS LES EXERCICES DES PROCHAINS TP, NOUS VOUS DEMANDONS DE FAIRE OBLIGATOIREMENT APPEL À DES SOUS-PROGRAMMES POUR DÉCOMPOSER LES PROBLÈMES.

Pour aller plus loin....

4) Tracé d'un carré crénelé

- ✎ Écrire l'algorithme permettant de tracer la figure ci-dessous sachant que :
- les créneaux possèdent les dimensions suivantes : 20 unités en hauteur et 20 unités en largeur,
 - le coin bas droit du carré se trouve à l'origine O.
- L'algorithme doit favoriser la répétition de motifs.

