

Utiliser Dev-C++

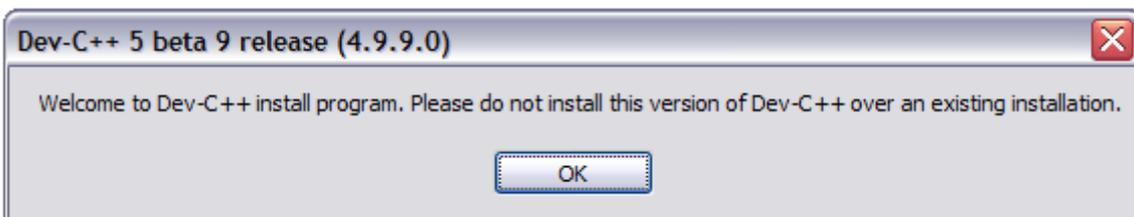
Voici quelques explications sur l'utilisation de Dev-C++, un excellent environnement de développement en C et C++ pour Windows qu'on peut télécharger librement depuis le site www.bloodshed.net (téléchargements : www.bloodshed.net/dev/devcpp.html).

Table des matières

1. [Installer Dev-C++](#)
2. [Utiliser simplement Dev-C++](#)
3. [Compiler et exécuter votre programme](#)
4. [Empêcher la fermeture de la console](#)
5. [Travailler avec un projet](#)
6. [Débuguer votre programme](#)
7. [Installation d'un nouveau package](#)
8. [Configuration du « Help »](#)
9. [Options diverses](#)

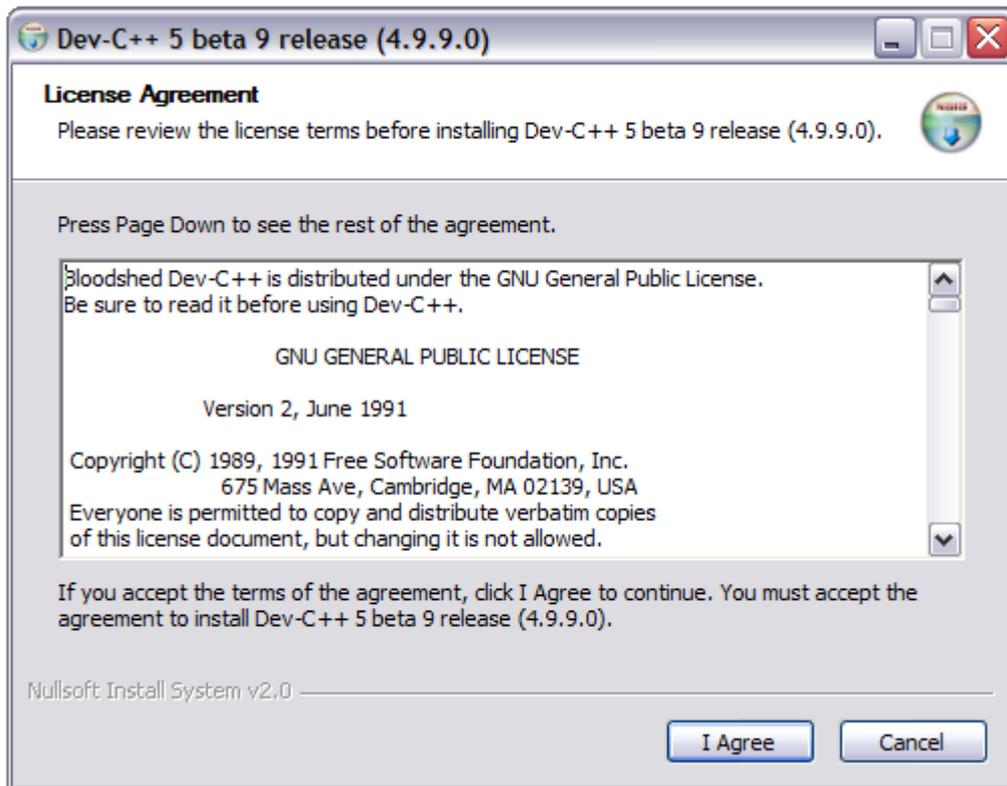
.1 Installation de Dev-C++

L'installation de Dev-C++ ne pose pas de problème mais, pour que chacun soit bien rassuré, nous en montrons ici les étapes. Pour démarrer, exécutez `devcpp4990setup.exe`, le fichier téléchargé : après un premier panneau qui vous rappelle qu'il vaut mieux *commencer par supprimer toute éventuelle installation précédente*¹ de Dev-C++

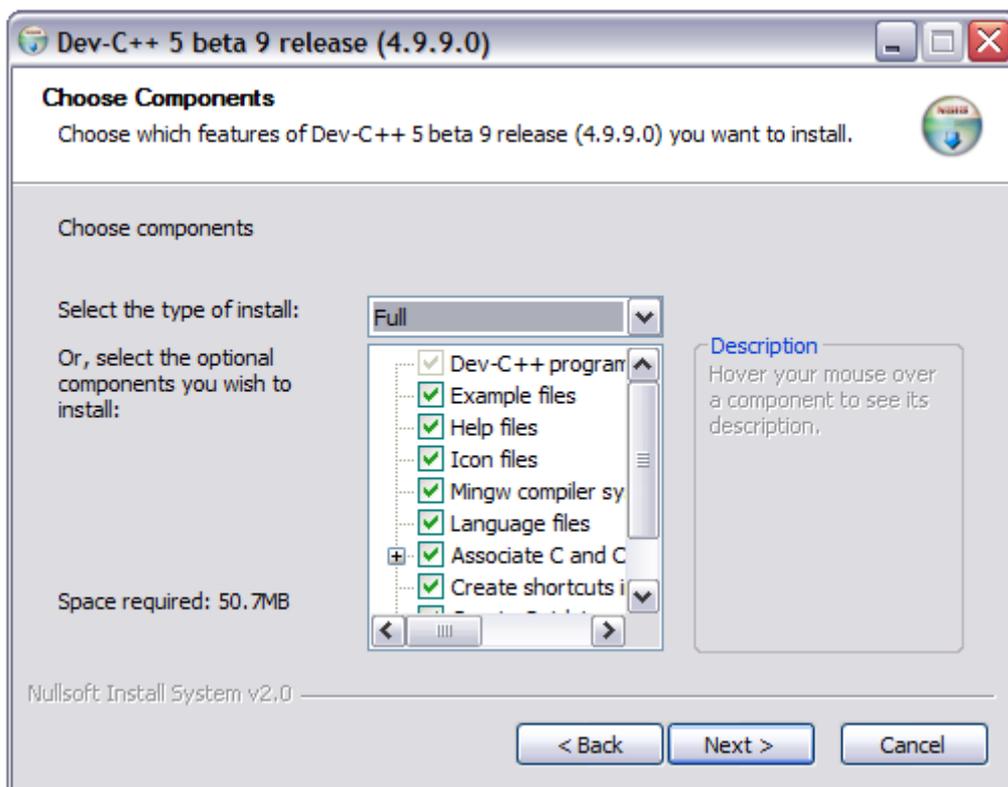


Vous obtenez la licence du produit (à lire attentivement :-)

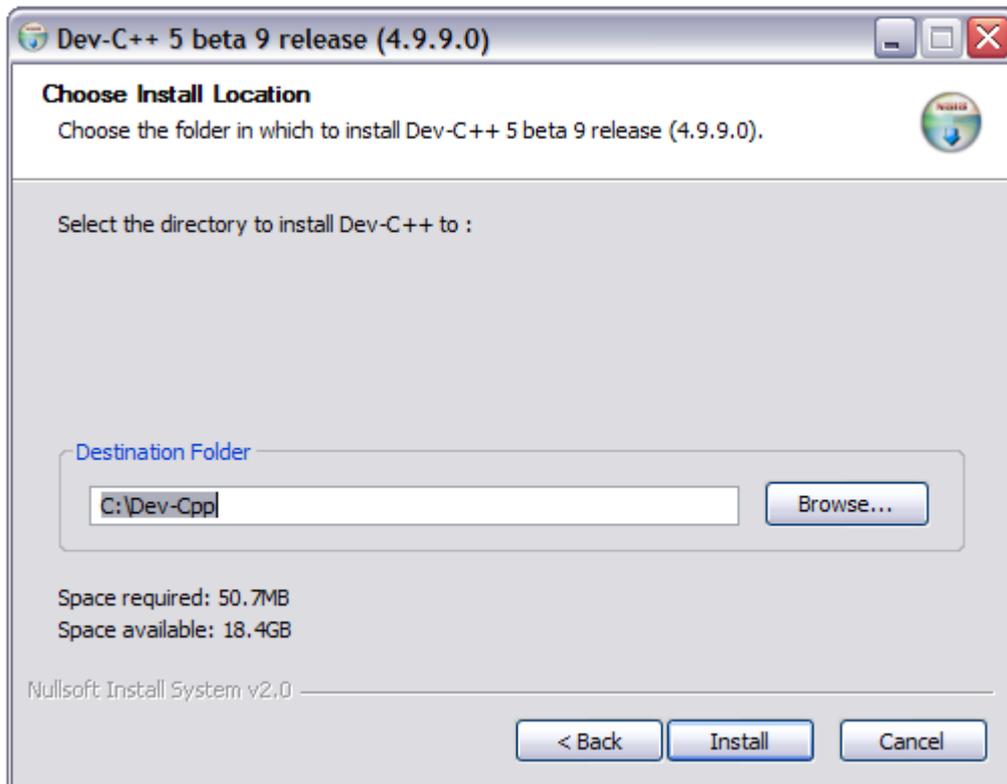
Note. Ces explications sont tirées d'un document rédigé par Henri Garreta de la Faculté des Sciences de Luminy, augmentées de quelques informations.



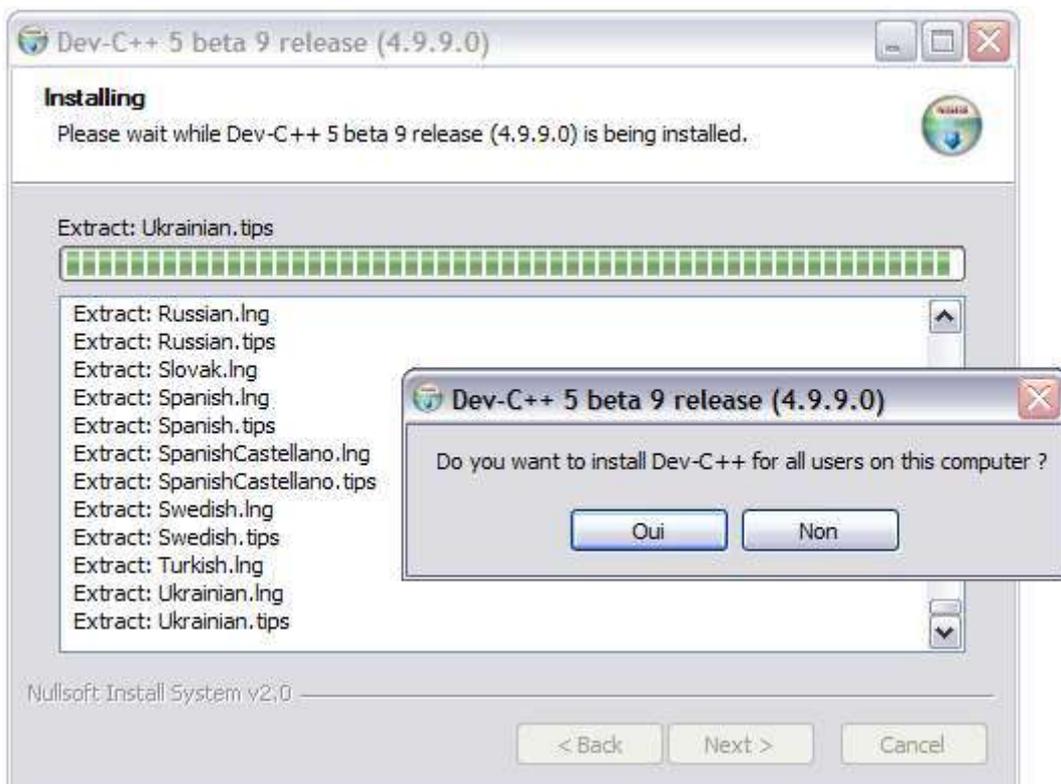
Cliquez sur « I Agree ». Apparaît alors un panneau pour choisir les éléments à installer :



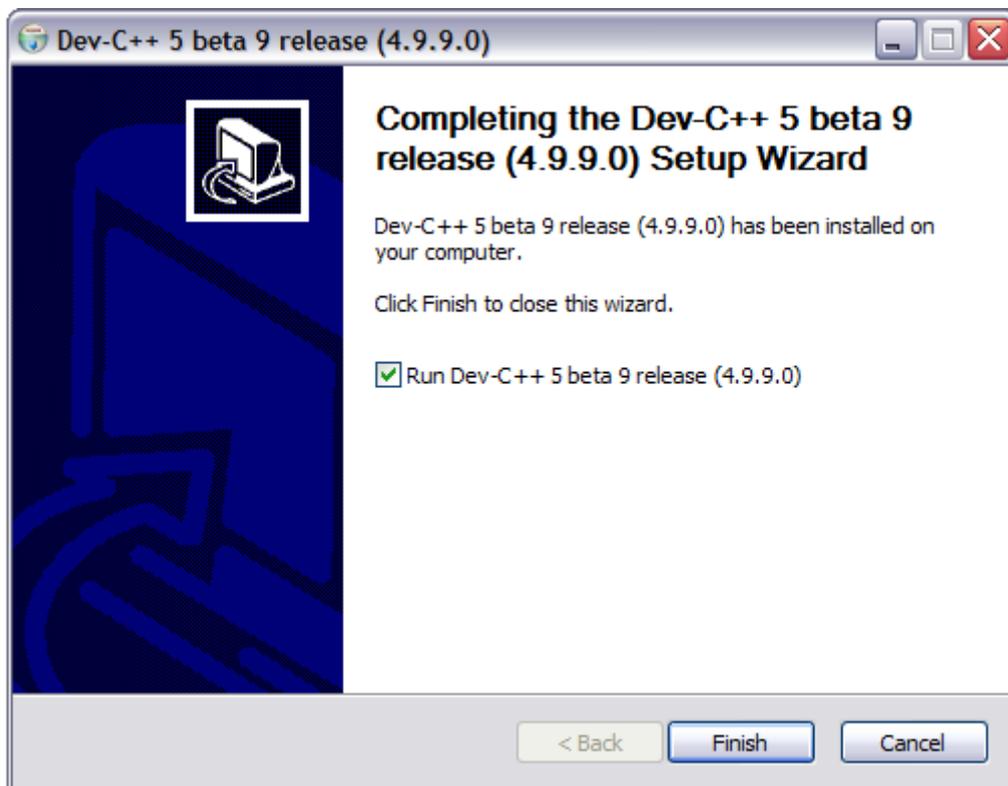
Laissez les cases à cocher comme elles sont (il est difficile de réduire significativement la place occupée par ce logiciel, qui n'est pas très encombrant) puis cliquez sur « Next > ». Il est question alors du dossier d'installation :



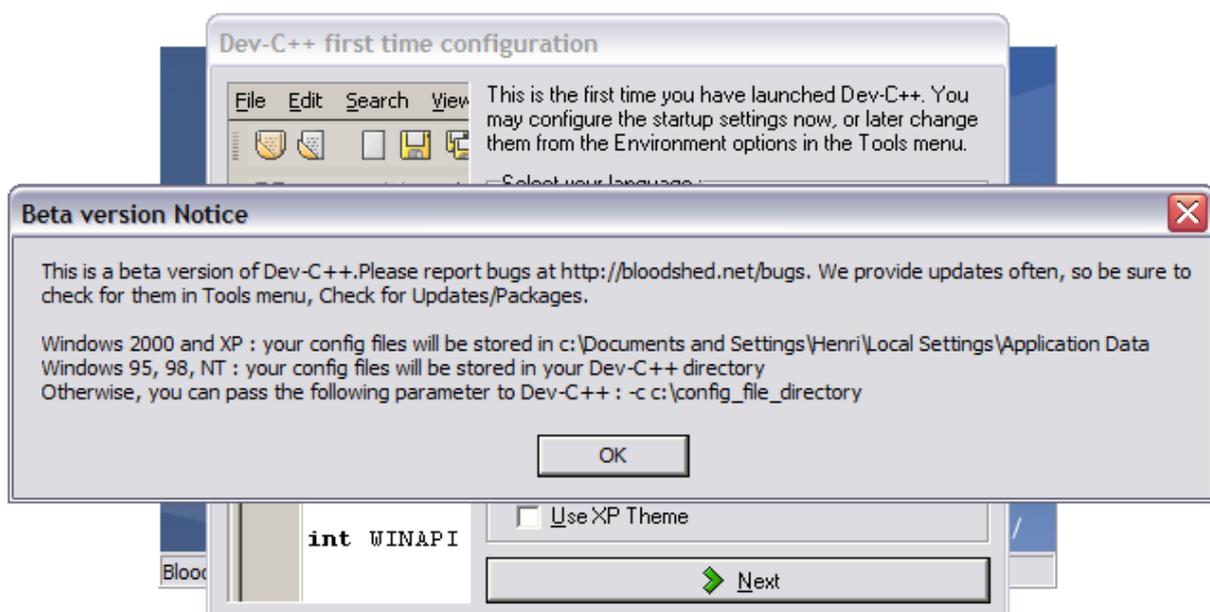
Cliquez sur « Install », l'installation se fait quasiment sans besoin d'aide. Selon le système que vous utilisez, une question rituelle vous sera posée : « Voulez-vous installer Dev-C++ pour *tous les utilisateurs* de cet ordinateur ? »



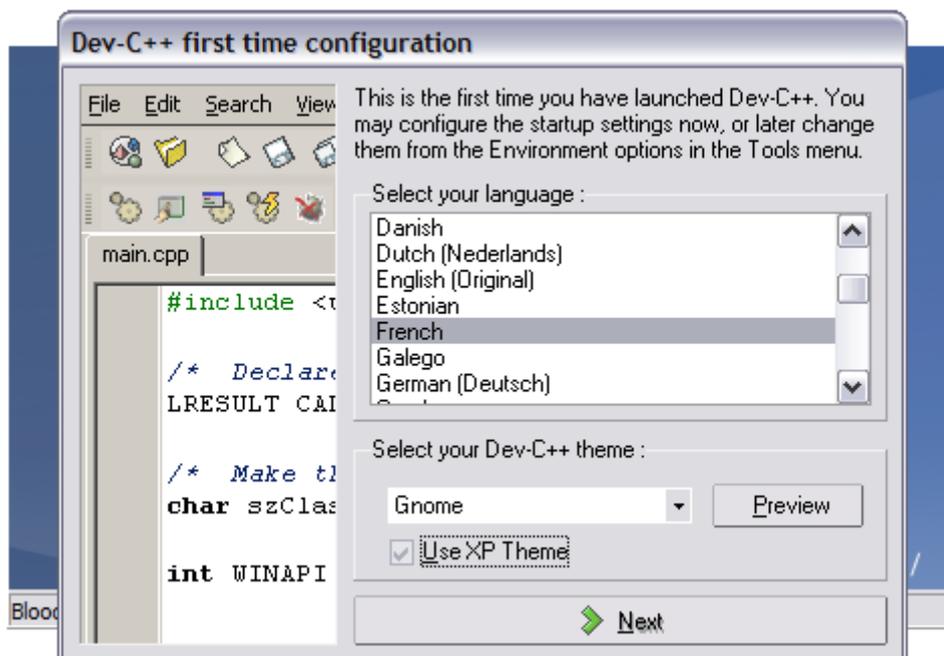
Si vous êtes en train d'installer Dev-C++ sur un système partagé répondez « Non », car vous n'avez pas probablement le droit de faire une « installation pour tous » ; s'il s'agit de votre propre système, répondez ce que vous voulez.



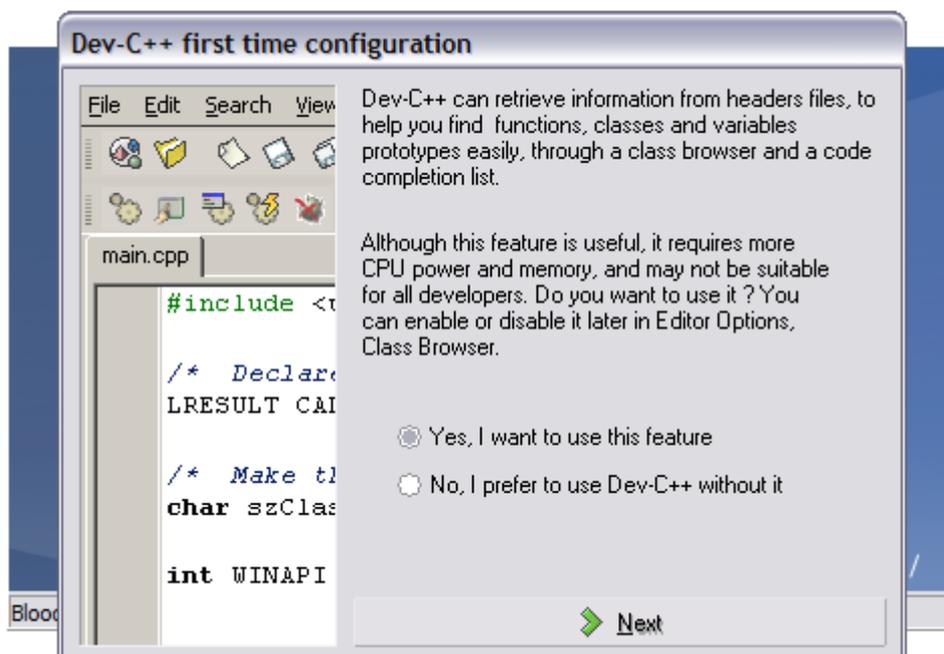
L'installation est maintenant terminée. Quand vous cliquerez sur « Finish », Dev-C++ sera lancé et, *si votre machine n'a pas gardé des traces d'une installation précédente*, vous aurez un message informatif :



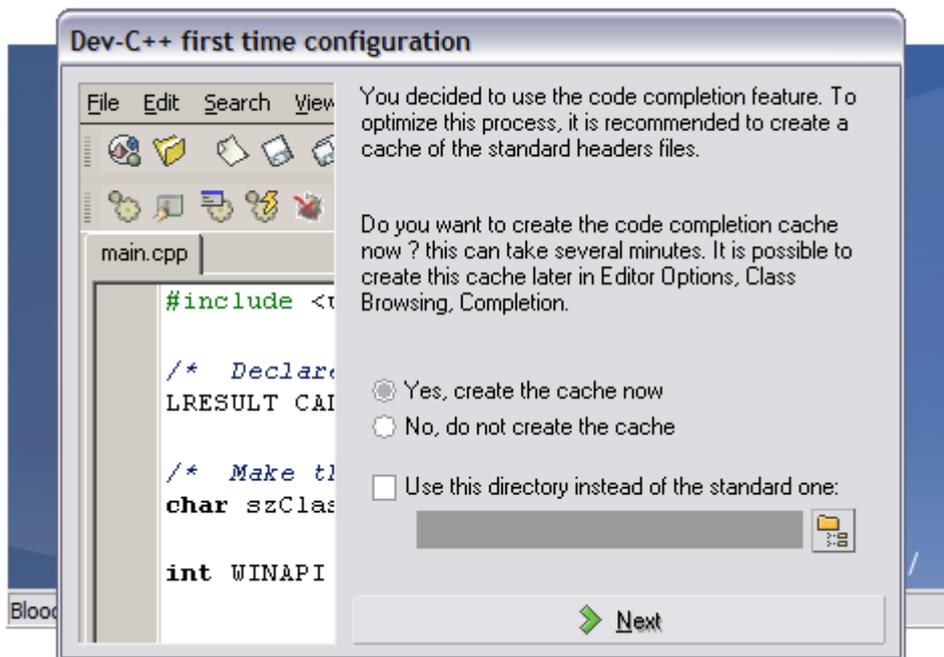
et vous devrez ensuite choisir la langue et le thème (c'est-à-dire l'aspect des boutons) de l'interface. Les illustrations montrées dans la suite de cette notice correspondent aux choix : French, Gnome et XP Theme



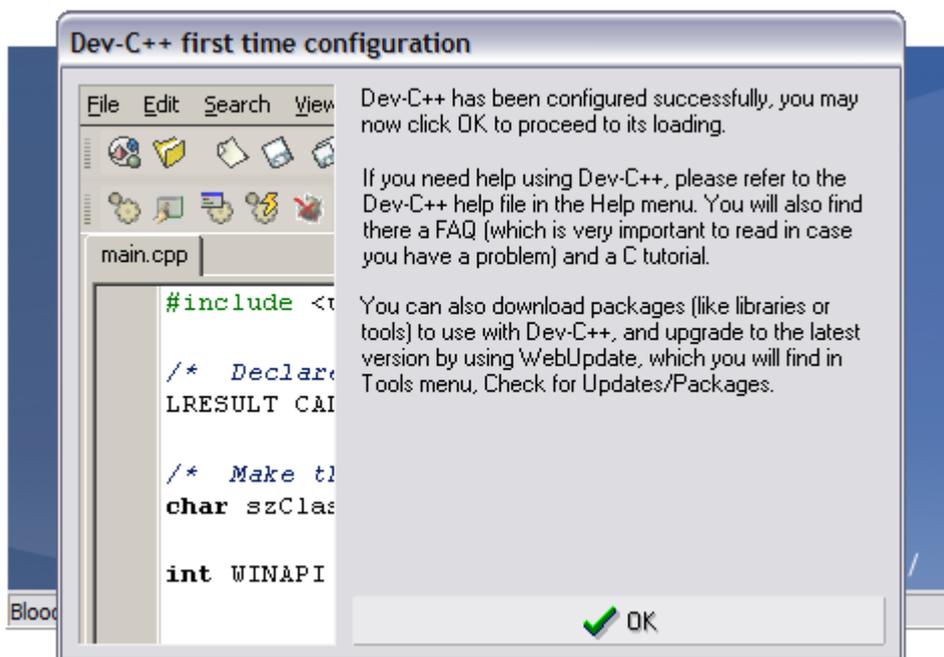
Vous pouvez en outre activer des fonctions d'aide à la composition, comme la navigation parmi les classes et la complétion de code (ces fonctions sont surtout utiles si on programme en C++) :



Il est alors recommandé de laisser Dev-C++ créer un « cache » pour ces fonctions :

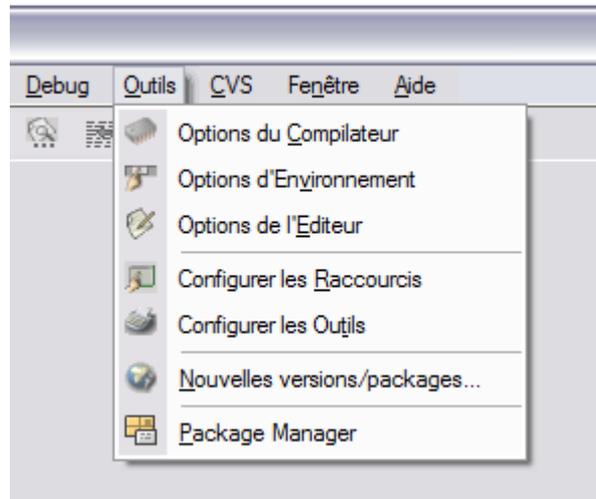


C'est un peu long, mais cela finit par se terminer :

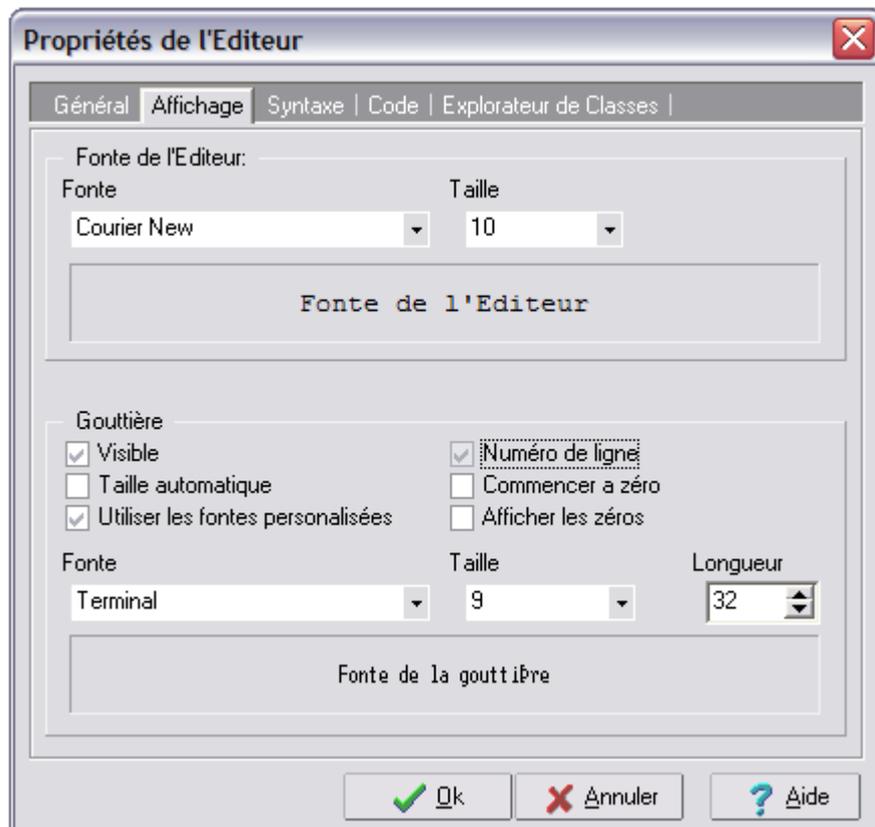


Durant l'installation, un raccourci pointant sur Dev-C++ aura probablement été créé et placé sur le bureau ou dans la barre des tâches. Si ce n'était pas le cas, vous devez créer vous même un raccourci sur le fichier `devcpp.exe` qui se trouve dans le dossier `Dev-Cpp`. Par la suite, vous lancerez Dev-C++ en double-cliquant sur ce raccourci.

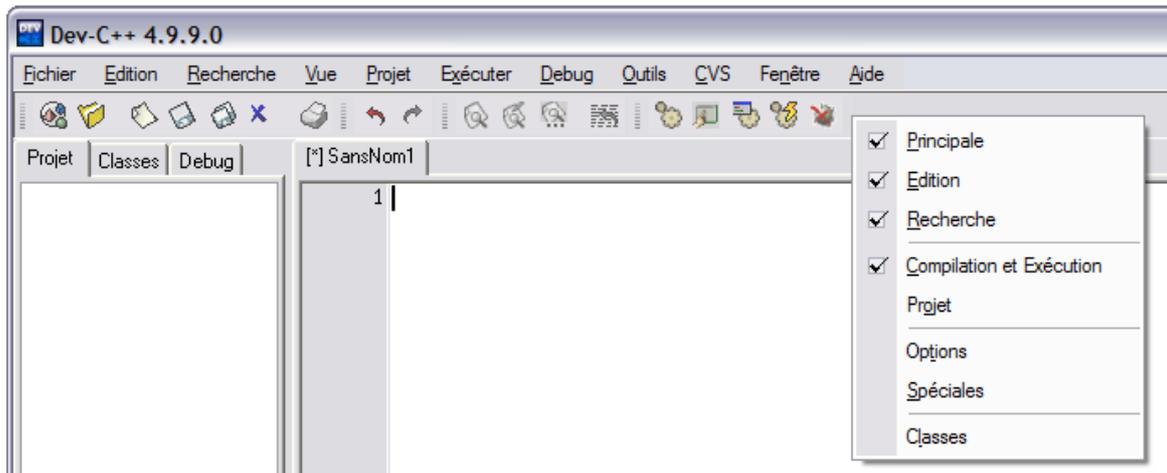
Selon vos goûts, vous pourrez ultérieurement personnaliser un certain nombre d'autres éléments de l'interface et de l'éditeur, en actionnant les commandes du menu **Outils** :



Par exemple, vous pouvez faire en sorte que les numéros de ligne apparaissent dans la gouttière (la marge gauche), comme dans les exemples de cette notice :



Notez enfin que vous pouvez alléger la barre d'outils en choisissant les groupes de boutons qui y apparaissent. Pour cela, cliquez avec le bouton droit de la souris dans la barre d'outils (mais pas sur un bouton) :



.2Utilisation simple de Dev-C++

Fichier nouveau. Si votre programme tient dans un seul fichier et n'a pas besoin de bibliothèques particulières, vous pouvez utiliser Dev-C++ *sans créer de projet*. Pour cela il vous suffit de lancer Dev-C++ puis de créer un fichier source : commande **Nouveau | Fichier Source** du menu **Fichier** (beaucoup de commandes des menus s'obtiennent aussi par des boutons de la barre d'outils et/ou par des raccourcis clavier).

Enregistrez immédiatement ce fichier à l'aide des commandes **Sauvegarder** ou **Sauvegarder Sous...** du menu **Fichier**.

Attention :

- le fichier doit avoir un **nom se terminant par .c**
- faites **attention au dossier** dans lequel le fichier sera rangé (ce doit être un dossier que vous aurez créé en vue d'y ranger vos travaux, non un dossier appartenant au système ou à Dev-C++)

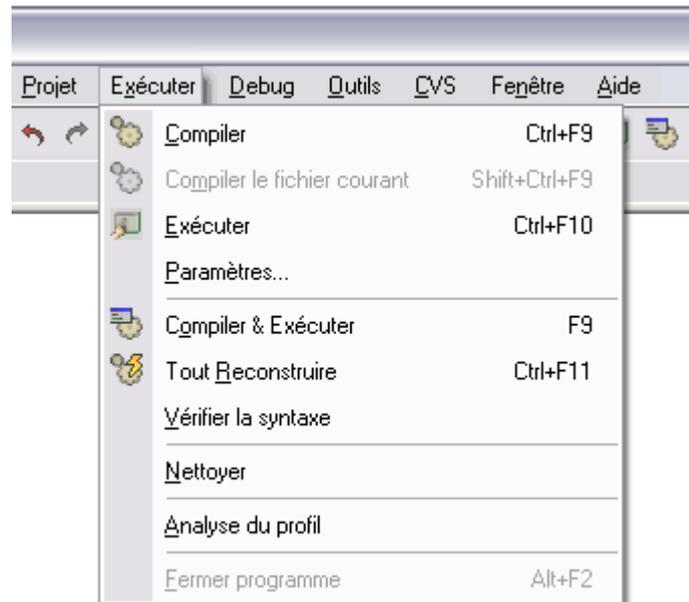
Fichier existant. Dans le cas où vous voulez travailler sur un fichier qui existe déjà, vous pouvez l'ouvrir dans Dev-C++ par la commande **Ouvrir Projet ou Fichier...** du menu du menu **Fichier**. D'autre part, si Windows est bien configuré (c'est le cas, en principe, si l'installation s'est bien passée), les icônes des fichiers **.c** ressemblent à l'une des suivantes :



et vous pouvez alors lancer Dev-C++ directement en double-cliquant sur une telle icône.

.3Compilation et exécution de votre programme

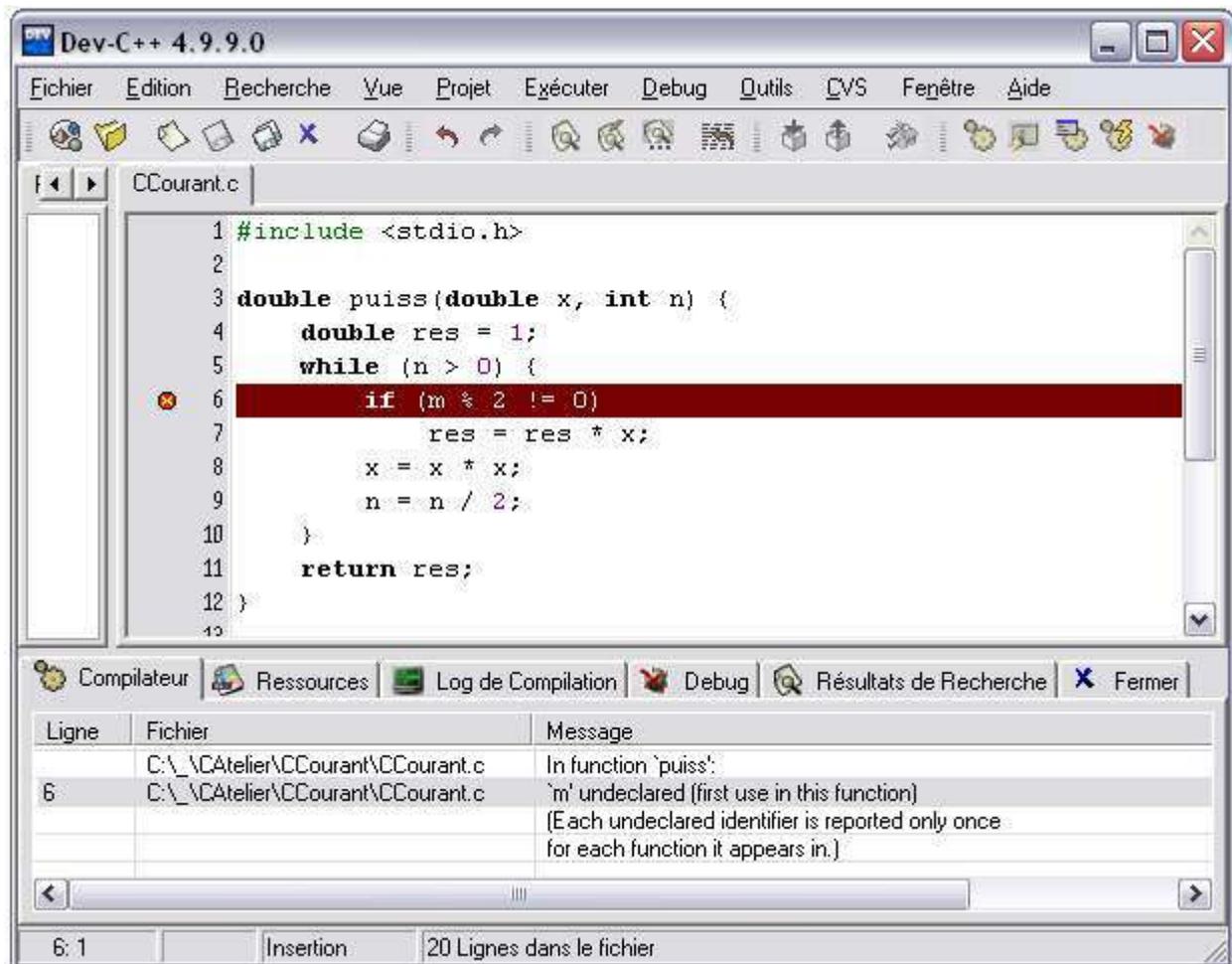
Cela concerne le menu **Exécuter** :



Compilez votre programme à l'aide d'une des commandes du menu **Exécuter** : **Compiler**, **Compiler le fichier courant**, **Compiler & Exécuter** ou **Tout Reconstruire** (dans le cas d'un unique fichier source, toutes ces commandes en produisent la compilation).

Les erreurs à la compilation sont affichées dans une fenêtre en bas de l'écran. En double-cliquant sur un message d'erreur on obtient l'affichage, dans la fenêtre principale, du texte de l'erreur signalé par une couleur spéciale et une marque dans la marge.

A titre d'exemple, observez l'image ci-dessous : les quatre lignes non vides de la fenêtre Compilateur constituent le signalement d'une erreur. On y apprend que dans la fonction `puiss`, plus précisément à la ligne 6 du fichier `c:_CAtelier\CCourant\CCourant.c`, l'identificateur `m` n'a pas été déclaré. Pas avare de détails, le compilateur nous dit en outre que c'est la première utilisation [de `m`] dans cette fonction, et que chaque identificateur non déclaré est signalé une seule fois dans chaque fonction où il apparaît.



En principe, le volet **Classes** de la fenêtre de gauche montre les structures, les variables globales et les fonctions de votre programme. En cliquant sur un de ces éléments la fenêtre principale se positionne sur l'entité en question.

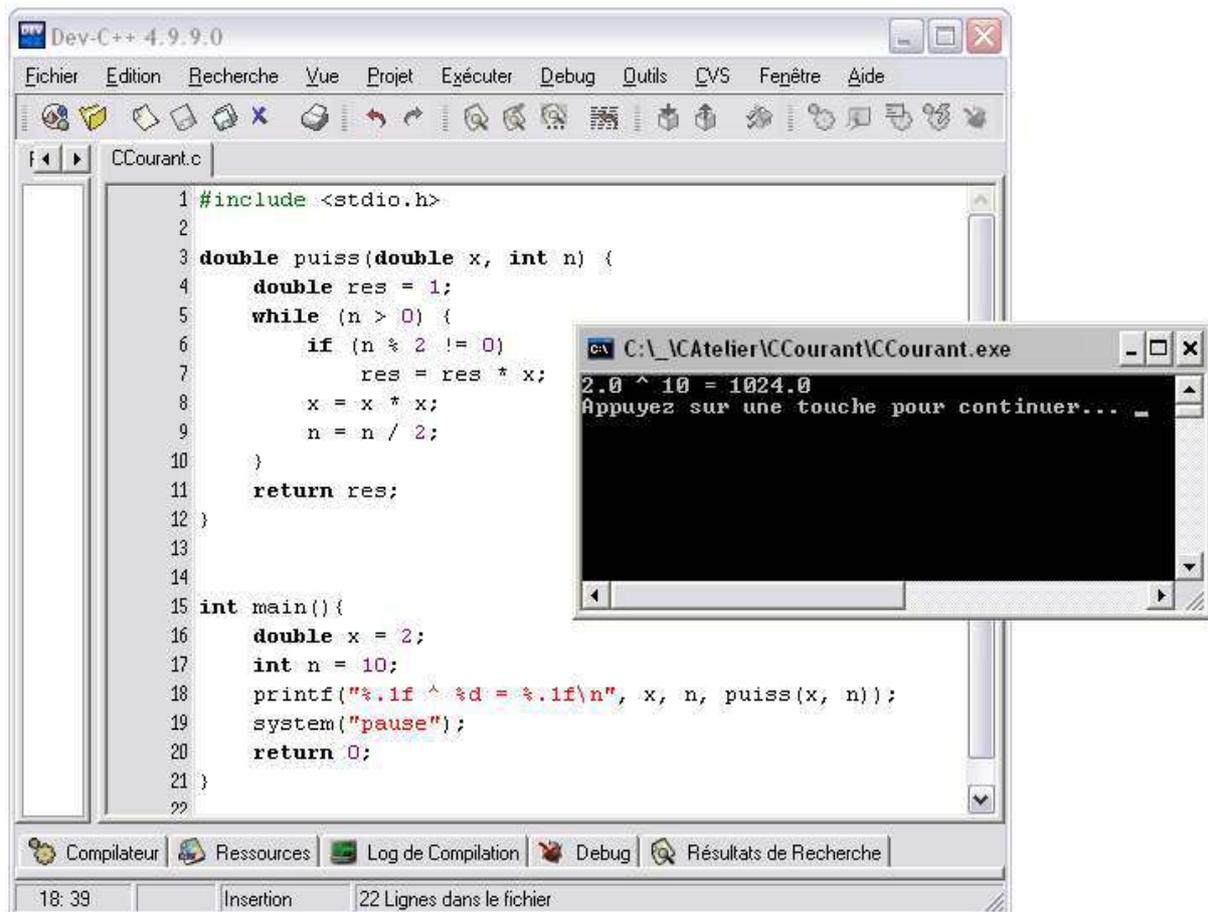
Exécutez votre programme par une des commandes **Exécuter** ou **Compiler & Exécuter**.

.4Empêcher la fermeture de la console d'exécution

La console d'exécution se ferme automatiquement à la fin de l'exécution d'un programme, ce qui ne laisse guère le temps de lire les éventuels résultats affichés. Pour empêcher cela, vous pouvez ajouter à la fin de votre programme la ligne

```
system("pause");
```

(**system** est une fonction standard C, **pause** est une commande MS-DOS/Windows qui produit l'affichage du message « Appuyez sur une touche pour continuer... » et met le système en attente de la frappe d'une touche) :



Une autre manière de garder ouverte la console d'exécution consiste à lancer votre programme depuis une fenêtre MS-DOS (appelée, selon le système, « Invite de commandes », « Console MS-DOS », etc.) : vous saisissez et compilez votre programme à l'intérieur de Dev-C++ mais pour l'exécuter vous vous placez dans une fenêtre « Invite de commandes ». Dans ce cas vous ne devez pas ajouter l'instruction `system("pause")` ; à la fin de votre programme.

Il y a éventuellement un petit travail à faire pour se placer dans le répertoire qui contient le programme. La commande pour changer de répertoire est `cd`, celle pour lister les fichiers d'un répertoire est `dir`. Enfin, pour lancer un programme il suffit de taper son nom. Voyez l'image suivante :

```
Microsoft Windows XP [version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\> cd exosC

C:\exosC> dir
Le volume dans le lecteur C n'a pas de nom.
Le numéro de série du volume est 180F-7E2A

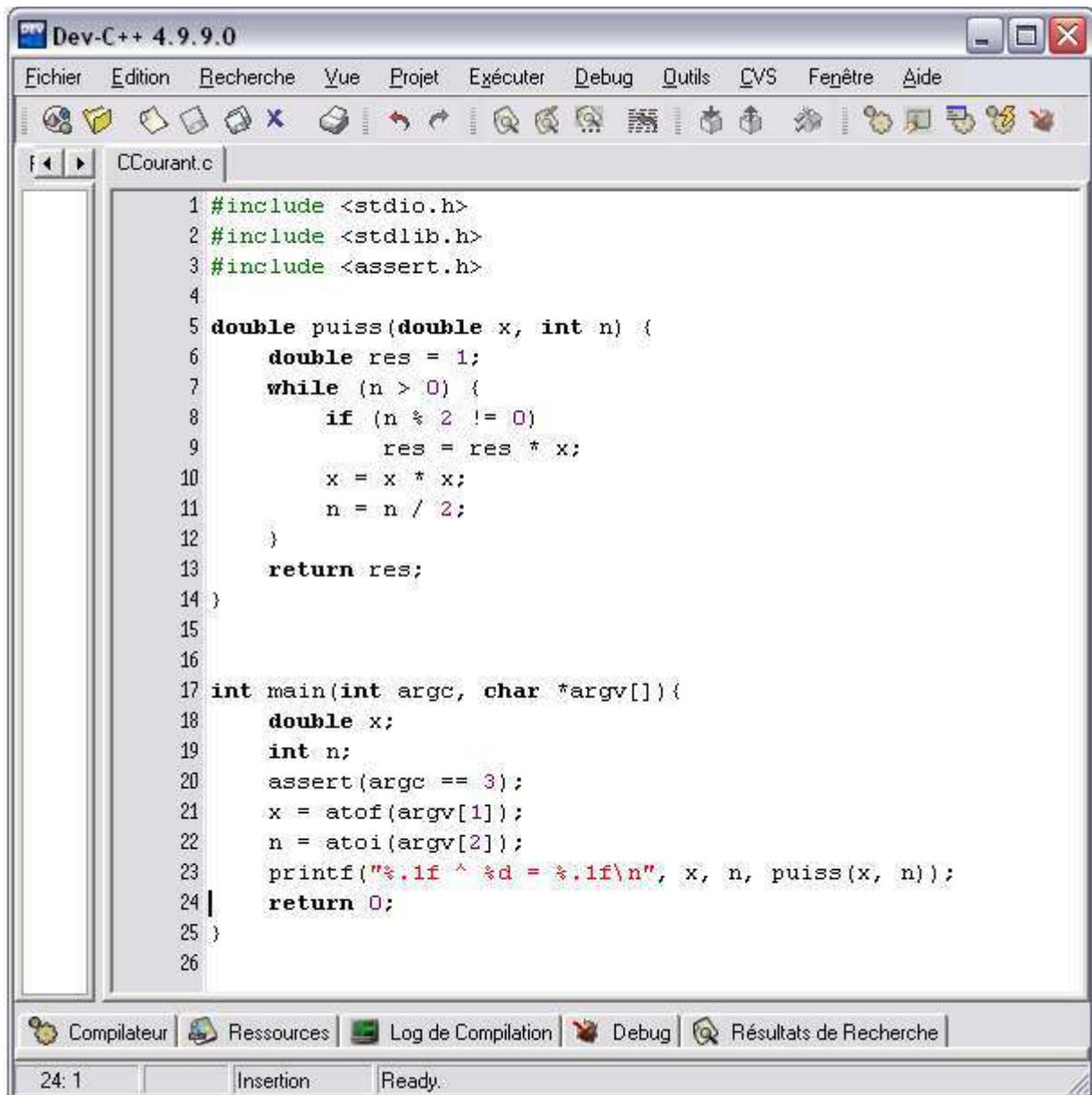
Répertoire de C:\exosC

08/10/2003  09:50    <REP>          .
08/10/2003  09:50    <REP>          ..
08/10/2003  10:00                23 248 CCourant.exe
08/10/2003  10:00                350 CCourant.c
08/10/2003  09:55                796 CCourant.dev
                3 fichier(s)          24 394 octets
                2 Rép(s)          99 820 544 octets libres

C:\exosC> CCourant
2.0 ^ 10 = 1024.0

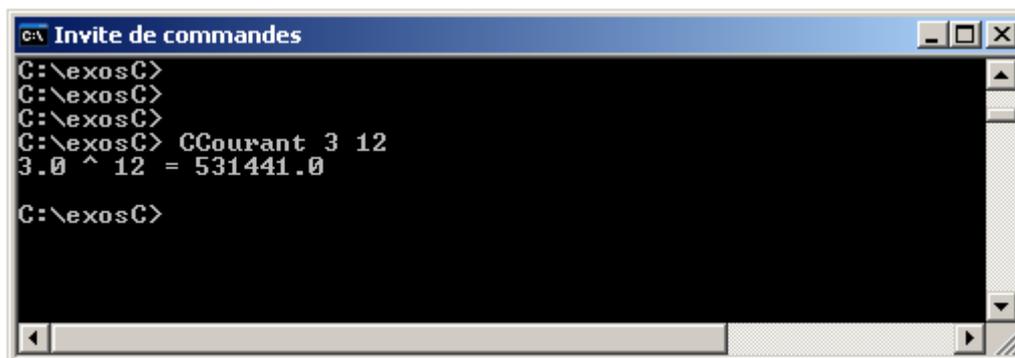
C:\exosC>_
```

Il est important de noter que cette manière d'exécuter son programme permet d'utiliser les *arguments de la ligne de commande*. Exemple, nouveau code :



```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <assert.h>
4
5 double puiss(double x, int n) {
6     double res = 1;
7     while (n > 0) {
8         if (n % 2 != 0)
9             res = res * x;
10        x = x * x;
11        n = n / 2;
12    }
13    return res;
14 }
15
16
17 int main(int argc, char *argv[]){
18     double x;
19     int n;
20     assert(argc == 3);
21     x = atof(argv[1]);
22     n = atoi(argv[2]);
23     printf("%.1f ^ %d = %.1f\n", x, n, puiss(x, n));
24     return 0;
25 }
26
```

Nouvelle exécution :



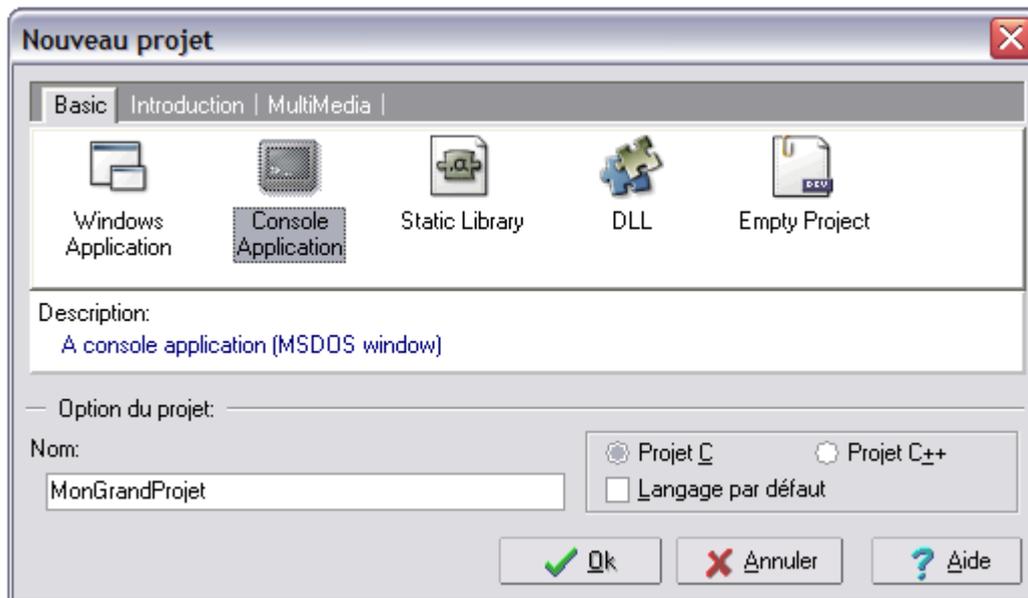
```
C:\ Invite de commandes
C:\exosC>
C:\exosC>
C:\exosC>
C:\exosC> CCourant 3 12
3.0 ^ 12 = 531441.0
C:\exosC>
```

.5Travailler avec un projet

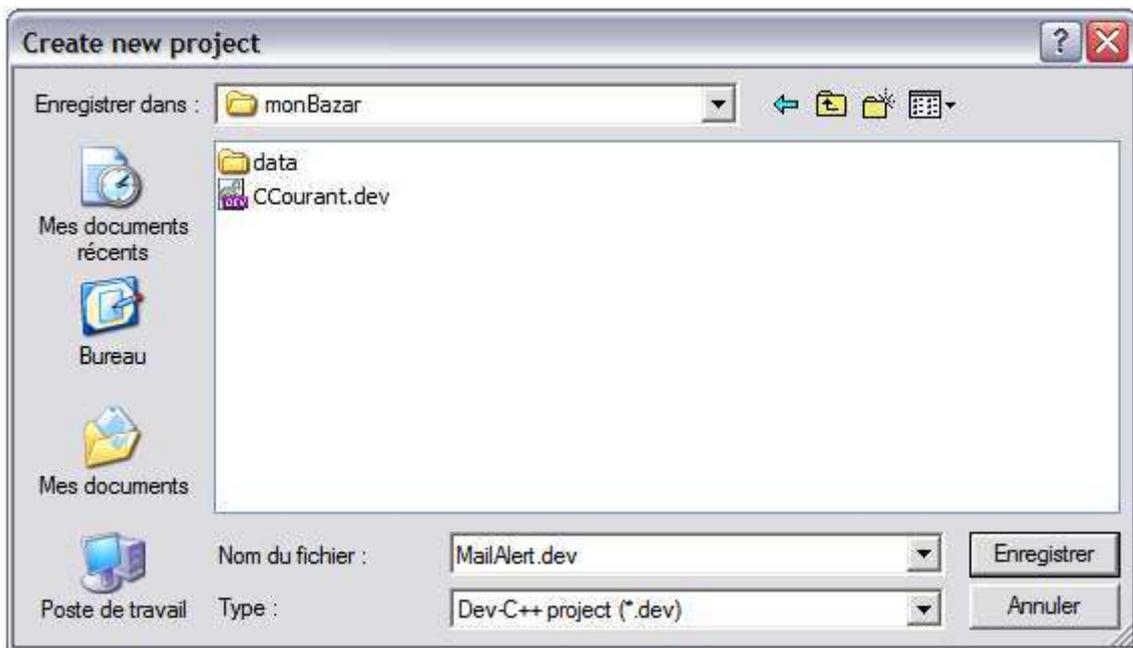
Si votre programme comporte plusieurs fichiers sources, ce qui est le cas normal quand on développe autre chose que de petits exercices, ou bien s'il requiert des bibliothèques spéciales (graphiques, mathématiques, etc.) alors il est nécessaire de travailler avec un projet.

Fondamentalement, un projet est une liste de fichiers et un ensemble d'options, mémorisés ensemble dans un fichier d'extension **.dev**.

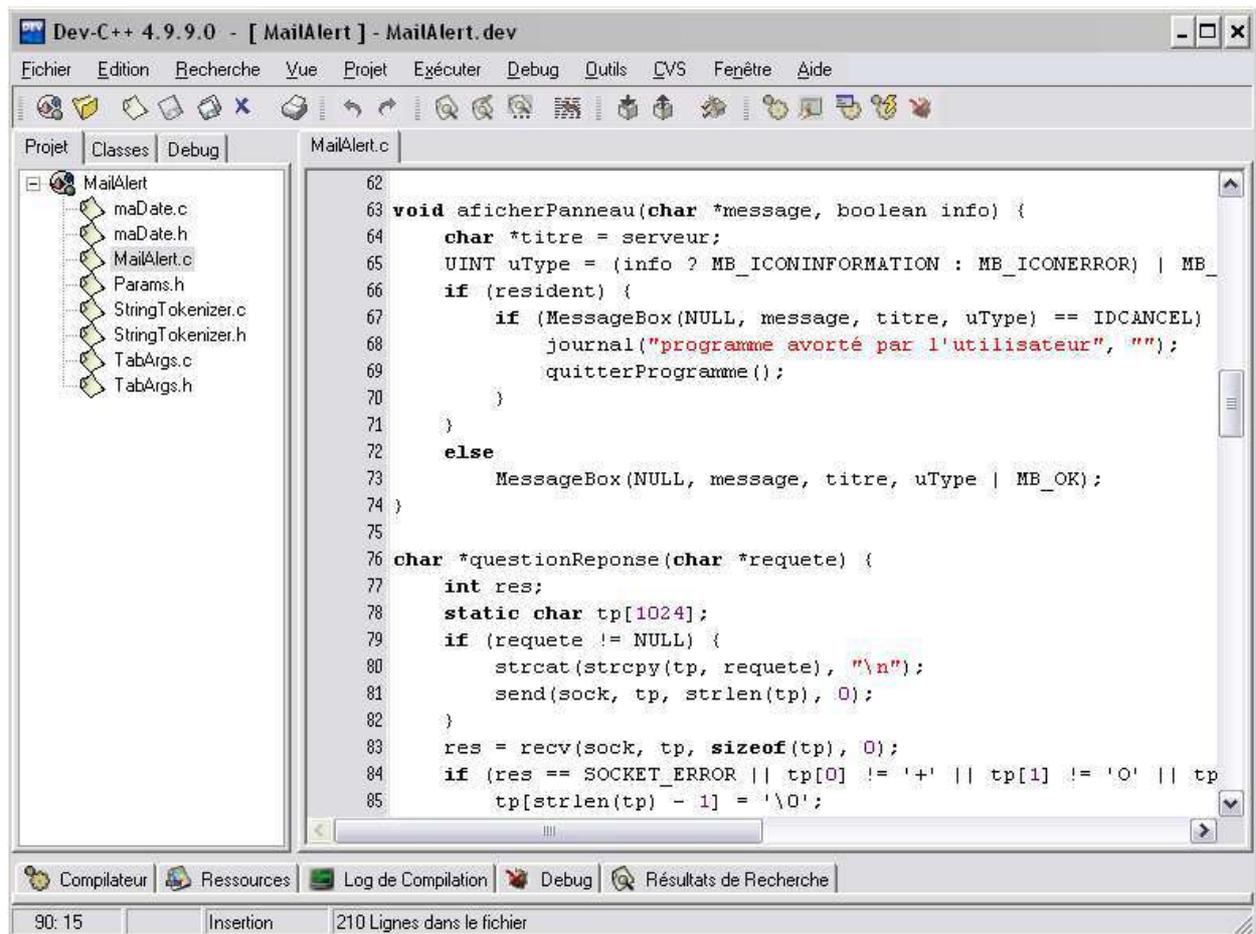
On crée un projet par la commande **Projet...** du menu **Nouveau**



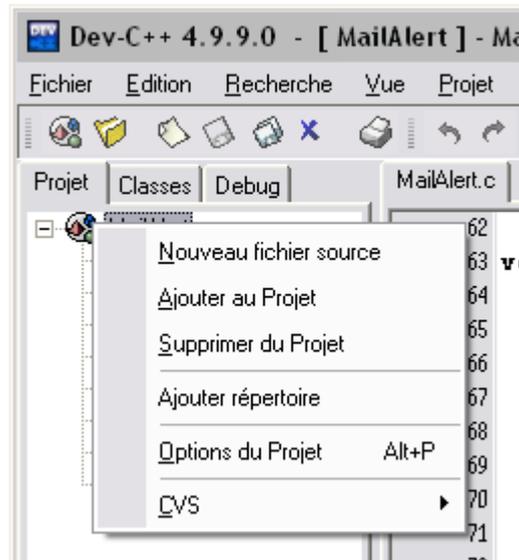
Pour ce qui nous occupe ici, choisir **Console Application**, **Projet C** et **Langage par défaut** comme ci-dessus. Il faut aussi trouver un nom pour le projet (ici *MonGrandProjet*), qui servira à l'étape suivante. Dès qu'on clique sur **Ok** on vous demande de sauver le projet (la fenêtre suivante peut être différente sur votre système) :



Une fois le projet créé, les commandes **Nouveau | Fichier Source** du menu **Fichier** et **Projet | Ajouter au Projet** du menu **Projet** permettent d'ajouter les divers fichiers sources. Les noms de ces fichiers s'affichent dans le volet **Projet** de la fenêtre de gauche :



Des clics avec le bouton droit de la souris sur les éléments du volet **Projet** font apparaître un menu contextuel permettant également d'ajouter ou enlever des fichiers au projet :



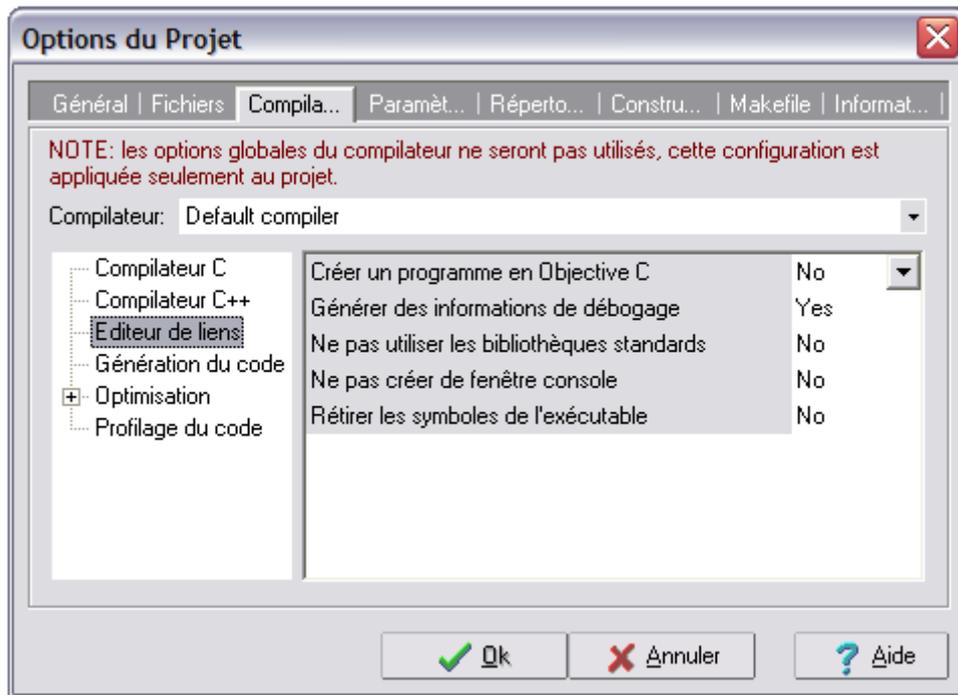
Quand on travaille avec un projet comportant plusieurs fichiers sources, la commande **Compiler** du menu **Exécuter** est optimisée de telle manière qu'elle produit la compilation *uniquement des fichiers qui en ont besoin*, c'est-à-dire ceux qui n'ont jamais été compilés et ceux qui ont été modifiés depuis leur dernière compilation (autrement dit: sont compilés les fichiers .c pour lesquels le fichier .o correspondant soit n'existe pas, soit a une date plus ancienne que celle du fichier .c).

La commande **Tout Reconstruire** du menu **Exécuter** produit la compilation de *tous les fichiers*, indépendamment de ces considérations de dates. Utilisez-la dès que vous avez l'impression que la commande **Compiler** ne fait pas le travail qu'elle devrait et, en particulier, à la suite d'une modification des options du projet (ces modifications ne touchent pas la date des fichiers sources).

.6Déboguer votre programme

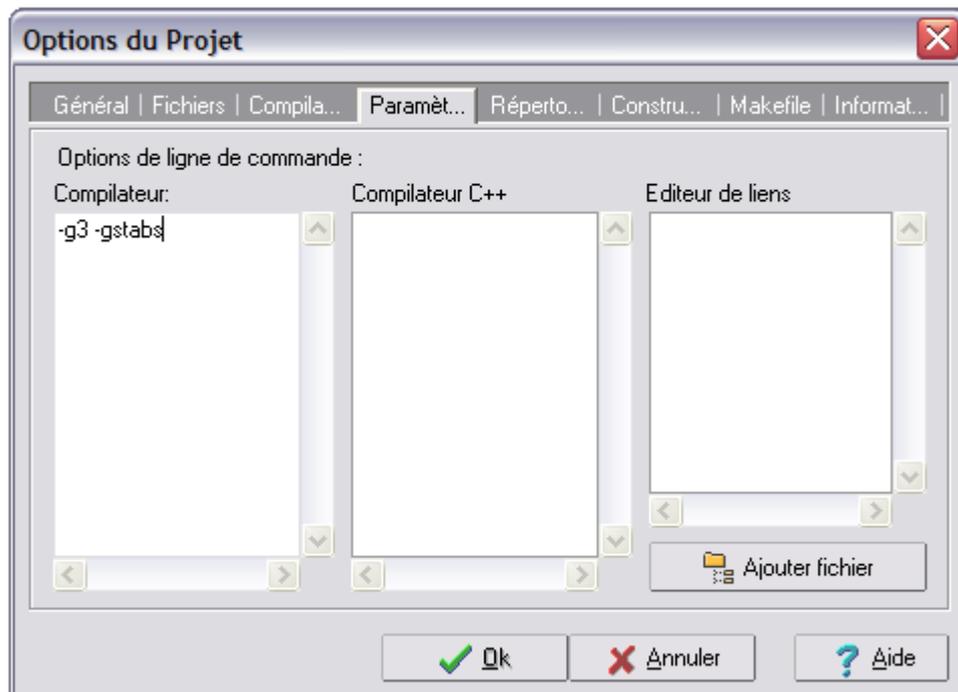
Un *débogueur* est un outil pour exécuter un programme pas à pas et en permettant d'examiner le contenu des variables. Cela permet de comprendre le comportement de l'application et comment ses variables évoluent. C'est un moyen précieux pour trouver les fautes de programmation, et aussi pour parfaire sa connaissance de la programmation en examinant de l'intérieur comment les programmes marchent.

Pour qu'un programme puisse être contrôlé par le débogueur il faut que le fichier exécutable ait gardé certaines informations symboliques, comme les noms des variables et des fonctions, qui sont habituellement éliminées durant la compilation. A cet effet il faut positionner une option de l'éditeur de liens : commande **Options du Projet** du menu **Projet**, volet **Compilation**, choisir **Editeur de liens** et donner la valeur **Yes** à l'option **Générer des informations de débogage** (laisser les autres options à **No**).



Après avoir mis à *Yes* l'option *Générer les informations de débogage* il faut recompiler le programme avec la commande **Tout Reconstruire** du menu **Exécuter** (la commande *Compiler* risquerait de ne pas faire le travail).

Note 1. Dans certains cas, les actions précédentes ne suffisent pas à mettre Dev-C++ dans un état rendant possible le débogage. Une manière d'atteindre cet état à coup sûr consiste à ajouter la ligne « `-g3 -gstabs` » dans la fenêtre **Compilateur:** du volet **Paramètres** du panneau **Options du projet** (commande **Options du Projet** du menu **Projet**) :



Note 2. L'une et l'autre des manipulations précédentes peuvent se faire en agissant sur des panneaux plus ou moins analogues obtenus à travers la commande **Options du compilateur** du menu **Outils**. Ces actions portent alors sur tous les projets que vous créez et non uniquement sur le projet en cours :



Le volet **Debug** en bas de l'écran montre les principales commandes du débogueur :



Attention. Il faut être tolérant, le débogueur n'est pas un programme très robuste et, dans certaines circonstances, ses commandes semblent ne pas avoir d'effet. En outre, faites attention à ne pas laisser des sessions de débogage actives par inadvertance, car cela met Dev-C++ dans un état malsain. En principe, la commande **Arrêter l'exécution** du menu **Debug** fait quitter le débogage et remet Dev-C++ dans l'état « normal ».

Il a deux manières principales de lancer le débogueur :

- placer un point d'arrêt (*breakpoint*) puis actionner la commande **Debugger**

- placer le curseur au début d'une instruction puis actionner la commande **Executer jusqu'au curseur**

La manière la plus simple de placer un point d'arrêt consiste à cliquer dans la gouttière (la marge de gauche). Une marque dans la gouttière indique le point d'arrêt, ainsi qu'un surlignage de la ligne concernée. D'autre part, une flèche dans la gouttière montre constamment la ligne sur laquelle l'exécution est arrêtée. Par exemple, la figure ci-dessous montre un moment d'une session de débogage, avec l'exécution arrêtée à la ligne 20, un point d'arrêt étant placé à la ligne 24 (les couleurs avec lesquelles sont surlignées certaines lignes peuvent être redéfinies par la commande **Options de l'éditeur** du menu **Outils**, volet **Syntaxe**, types **Breakpoints** et **Active breakpoints**) :

```

17 int main(int argc, char *argv[]){
18     double u, v;
19     int k;
20     u = 2;
21     k = 10;
22     v = puiss(u, k);
23
24     printf("%g ^%d = %g\n", u, k, v);
25     return 0;
26 }

```

The image shows a code editor window with a C program. The code is as follows:

```

17 int main(int argc, char *argv[]){
18     double u, v;
19     int k;
20     u = 2;
21     k = 10;
22     v = puiss(u, k);
23
24     printf("%g ^%d = %g\n", u, k, v);
25     return 0;
26 }

```

Line 20 is highlighted in blue, indicating that the execution is currently stopped at this line. Line 24 has a red circle icon in the left margin, representing a breakpoint. The gutter on the left shows a blue arrow pointing to line 20 and a red circle at line 24.

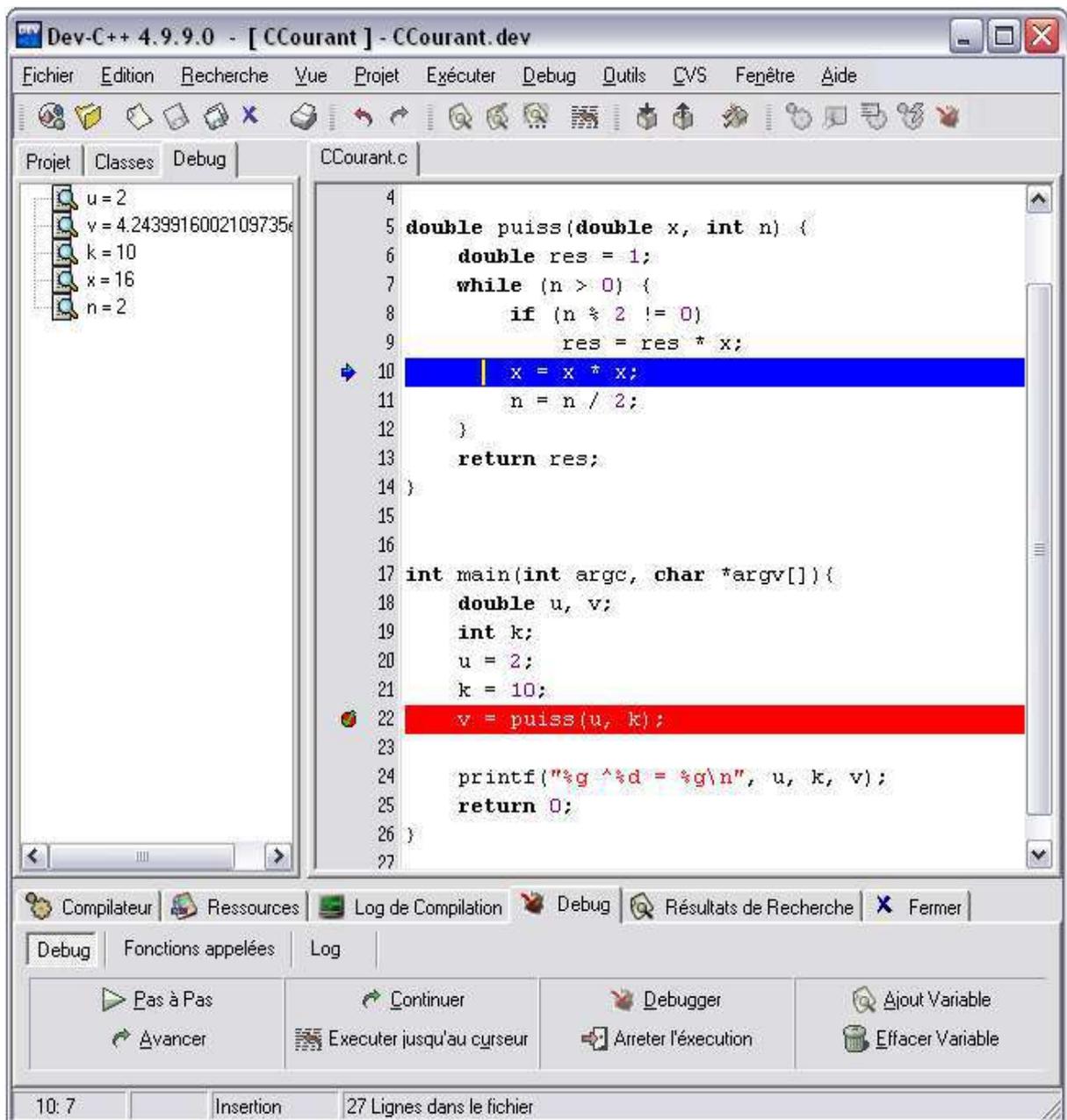
Un programme ne peut être arrêté que sur des instructions, évitez de mettre des points d'arrêt sur des lignes constituées de déclarations (des déclarations il ne reste aucune trace après la compilation).

Lorsque le débogueur est bloqué (sur un point d'arrêt ou consécutivement à l'emploi de la commande *Executer jusqu'au curseur*) on doit le débloquent par une des commandes :

- **Pas à Pas** (*Next Step*) : exécuter une instruction, en considérant qu'un appel de fonction est une instruction atomique qu'il n'y a pas lieu de détailler,
- **Avancer** (*Step Into*) : avancer d'une instruction, en s'arrêtant, le cas échéant, à l'intérieur des fonctions appelées,
- **Continuer** : relancer l'exécution du programme, jusqu'au prochain point d'arrêt ou, s'il n'y en a plus, jusqu'à la fin.

Examiner les variables. Pour faire afficher une variable dans le volet *Debug* à gauche de l'écran il suffit de presser le bouton **Ajout variable** ou bien de double-cliquer sur la variable. En fait, passer (lentement, soyez patients) le curseur sur la variable suffit la plupart du temps pour l'ajouter au volet *Debug*. La variable et sa valeur sont ensuite constamment affichées et on peut en observer l'évolution pendant que le programme est exécuté.

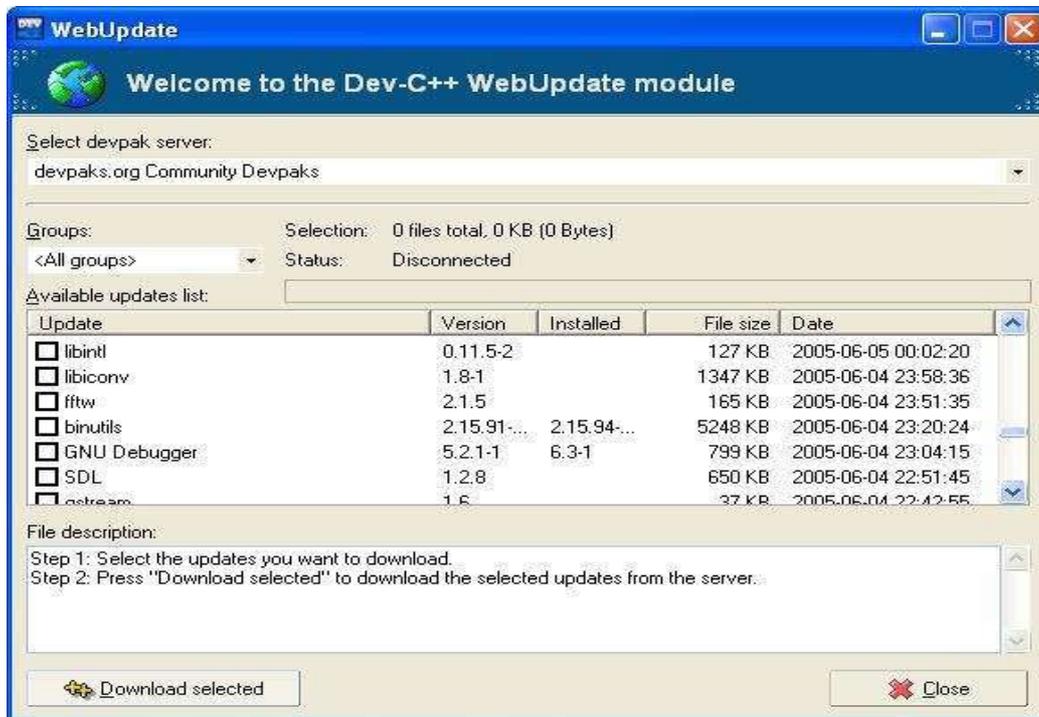
Lorsque la variable est complexe, le volet *Debug* permet d'en examiner les éléments.



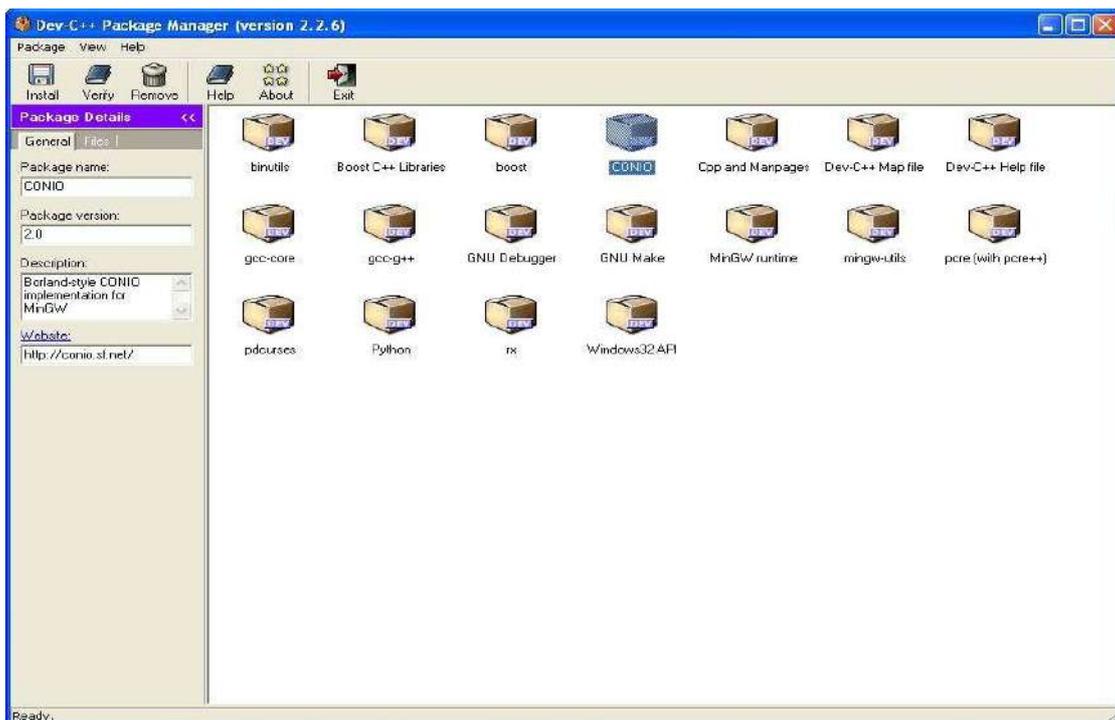
Note 1. Désinstaller toute trace d'une installation précédente est particulièrement important si vous cherchez à réparer une installation de Dev-C++ qui ne fonctionne plus.

.7 Installation d'un nouveau package

Le menu « Outils/Nouvelles versions/Packages... » permet de mettre à jour Dev-C++ et d'installer des extensions ou de nouvelles bibliothèques sous une forme très conviviale :



Pour connaître l'ensemble des « packages » existants sur sa machine, on peut utiliser l'outil « package manager » disponible dans le menu « Outils/Package Manager » :



.8 Configuration du « Help »

On peut enrichir le menu « Aide » grâce au menu « Aide/Editer menu d'aide » :



L'icône « Ajouter » permet de sélectionner un nouveau fichier d'aide et de le configurer. Ces fichiers d'aide, au format « hlp » ou « chm » se trouvent facilement sur le net.

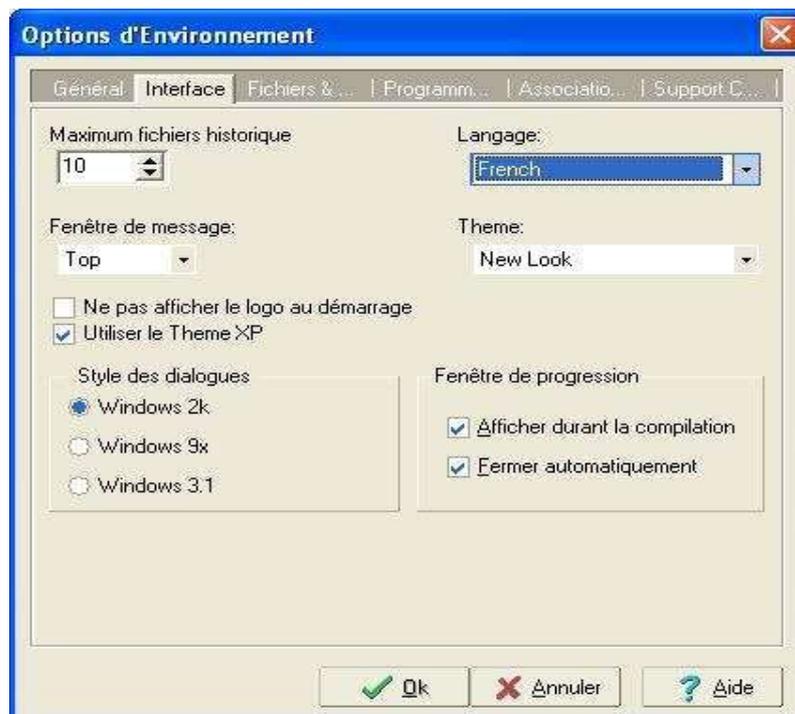
.9Options diverses

Le menu « Outils » permet de configurer différentes options de travail.

Options utiles du compilateur :



Options utiles d'environnement :



Options utiles de l'éditeur :

