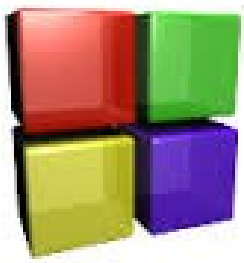
# Code::Blocks

## Table des matières



Code::Blocks est un environnement de développement intégré libre et multiplate-forme. Il est écrit en C++ grâce à la bibliothèque wxWidgets. Pour le moment, Code::Blocks est orienté C et C++, mais il supporte d'autres langages comme le D.



# 1. Introduction

Pour développer des programmes, il faut un logiciel qui permette :

- d'éditer du code (écrire le programme)
- de faire appel aux outils de compilation pour réaliser l'exécutable
- de débuguer le programme lors de son exécution

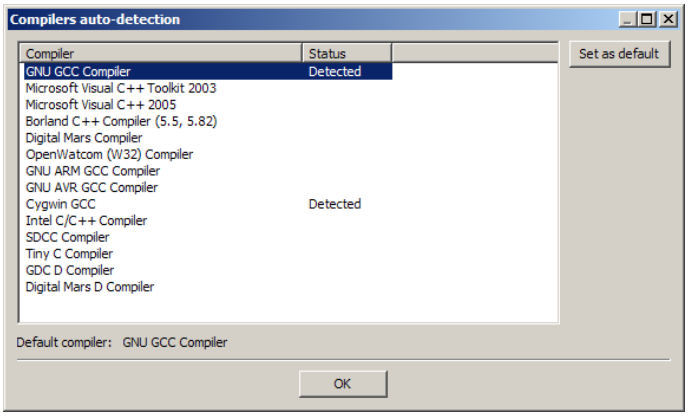
Un tel programme comme Code::Blocks s'appelle un IDE (Integrated Development Environment) :

- il est libre
- il est multi-plateformes (version pour Windows ou Linux)
- il est de taille raisonnable (installateur < 35Mo avec les outils de compilation)

Code::Blocks est capable de générer des applications écrites en C ou en C++ (en mode console ou non). Pour pouvoir utiliser Code::Blocks, il faut lui associer un compilateur tel MinGW[1]. Ce compilateur utilise les outils de compilations « libres » disponibles sur quasiment toutes les plateformes du marché.

# 2. Première exécution de Code::Blocks

Lors de la première exécution de Code::Blocks, si plusieurs compilateurs sont installés sur la machine, il faut choisir celui à utiliser par défaut. Le compilateur installé avec Code::Blocks est GNU[2] GCC[3] Compiler.




1 Minimalist GNU for Windows
2 Acronyme récursif : GNU is Not Unix
3 GNU C Compiler

# 3. Premier programme

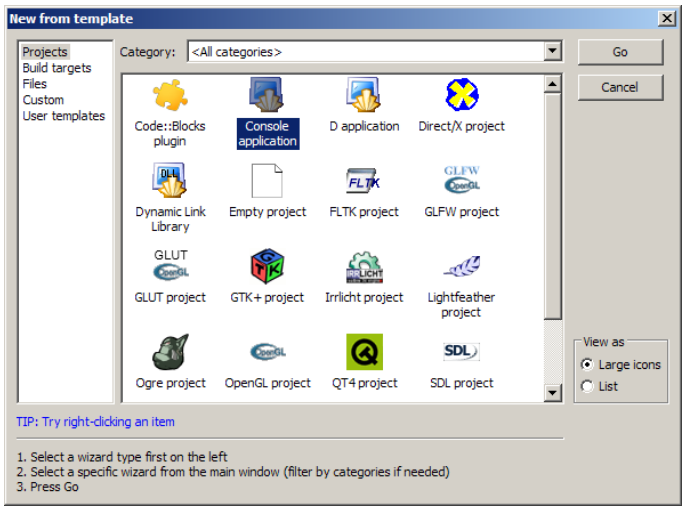
## *3.1. Créer un projet*

Un projet contient tous les éléments nécessaires pour compiler un programme.

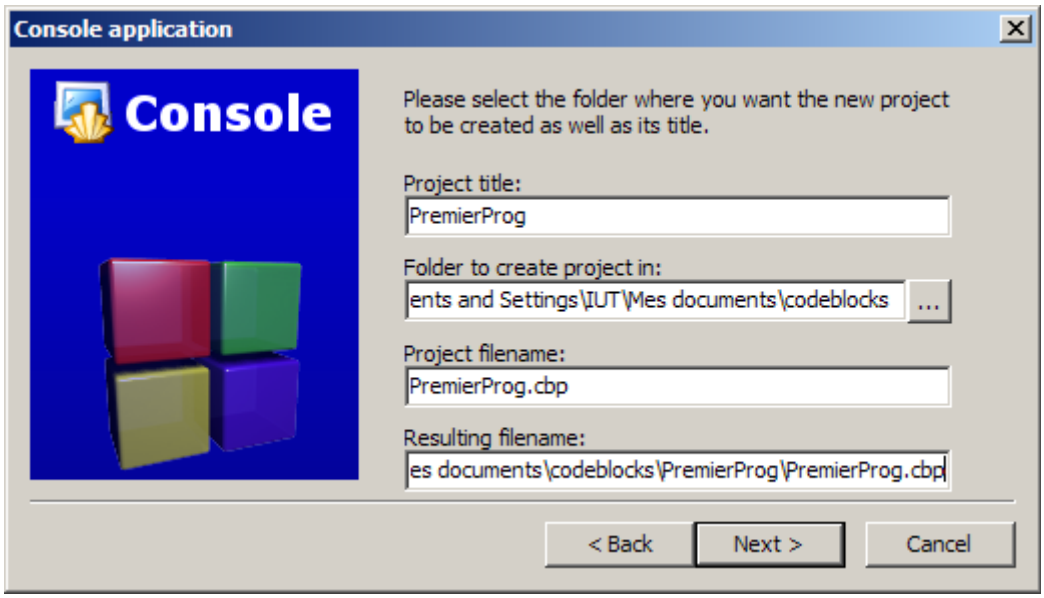
Vous pouvez créer un nouveau projet soit par le menu File → New → Project...


Vous pouvez aussi simplement cliquer sur Create a new project au milieu de l'écran d'accueil.

Une boîte de dialogue s'ouvre alors pour vous permettre de choisir le type de projet que vous souhaitez créer. Dans un premier temps, il est plus facile de créer des projets du type Console Application. Sélectionnez donc Console Application puis cliquez sur Go.



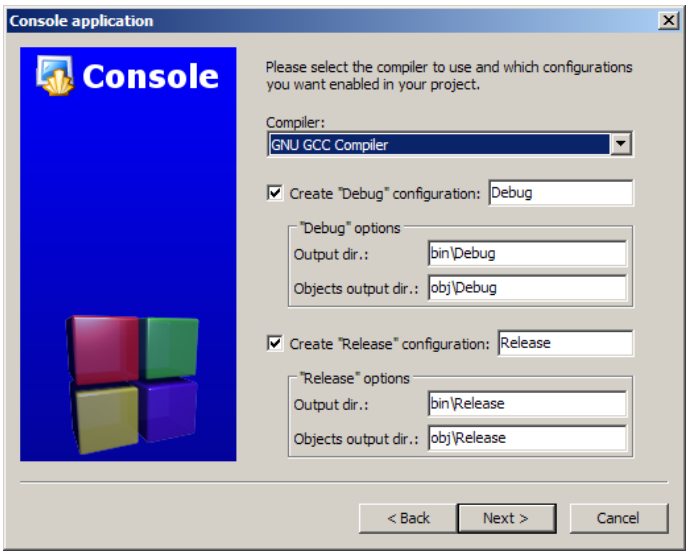

Vous devez maintenant choisir le nom du projet et son répertoire de sauvegarde.



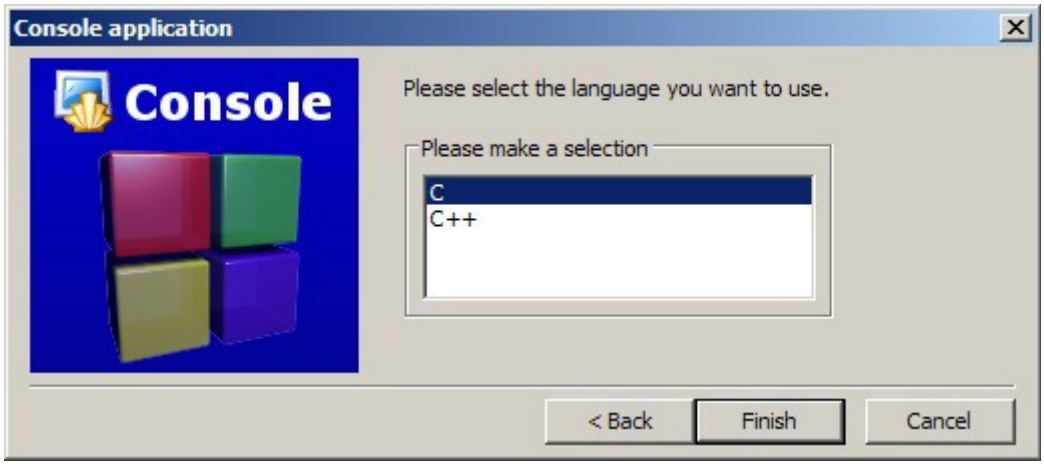

Code::Blocks place automatiquement tous les fichiers du projet dans un répertoire qui porte le nom du projet. Dans l'exemple ci-dessous, le répertoire du projet est Mes Documents\codeblocks et le nom du projet est PremierProg alors tous les fichiers du projets seront placés dans le répertoire Mes Documents\codeblocks\PremierProg

Pour le nom du projet, il est préférable de n'utiliser que des lettres et des chiffres et d'éviter tous les caractères spéciaux, en particulier les espaces.

Vous devez ensuite choisir le compilateur, il doit être automatiquement sélectionné à GNU GCC Compiler, et la création d'une version de débogage (Debug) et d'une version finale (Release) de votre programme.




Il reste à choisir un développement du programme en langage C.




Un clic sur Finish va maintenant créer le projet.

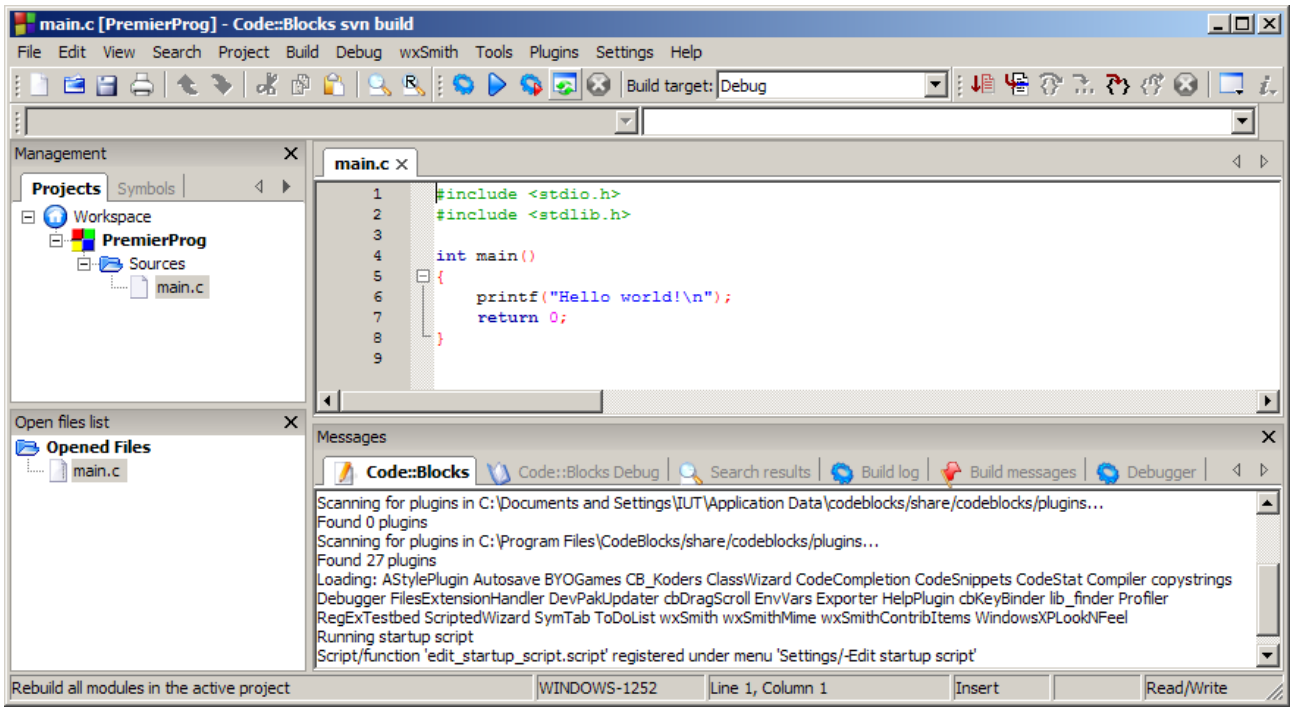
Dans le répertoire Mes Documents\codeblocks\PremierProg, deux fichiers ont été créés :

- PremierProg.cbp fichier du projet dont le format est propre à Code::Blocks et d'extension cbp pour Code::Blocks Project.
- main.c fichier en langage C.

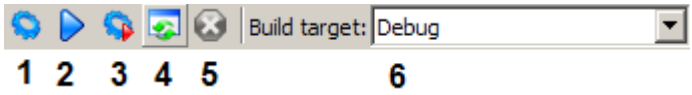
## *3.2. Construire et exécuter un projet*

Vous pouvez éditer le fichier main.c en allant à gauche dans la fenêtre Management dans Sources → main.c. Ce fichier comporte la fonction main, fonction principale du programme.

Le fichier main.c contient les instructions pour afficher Hello world! à l'exécution. Pour pouvoir exécuter le programme, il faut tout d'abord construire (build) le projet. Cette opération peut être réalisée soit par le menu Build, soit par la barre d'outils Compiler que l'on peut activer ou désactiver par View → Toolbars → Compiler.



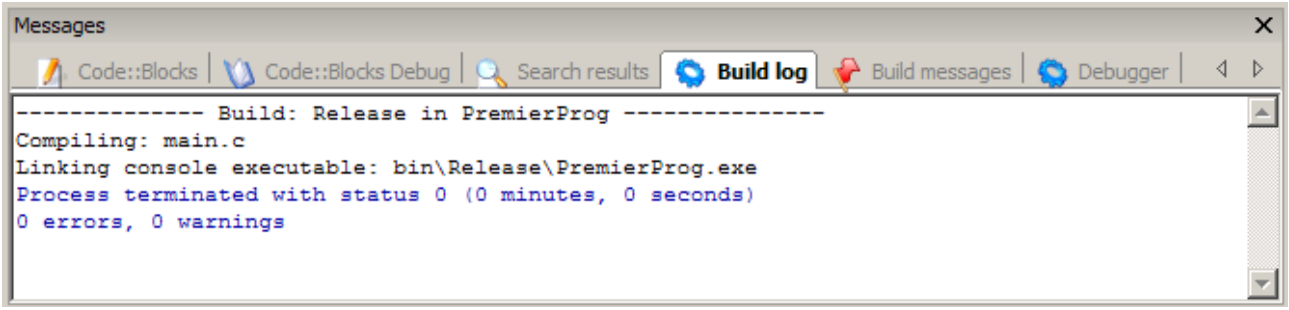

La barre d'outils Compiler :



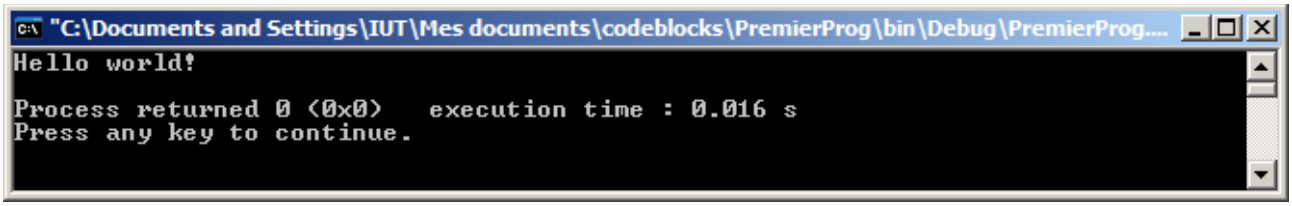

1. Construction du projet (Build)
2. Exécution du projet (Run)
3. Construction, puis exécution du projet (Build and run)
4. Tout reconstruire (Rebuild)
5. Termine l'exécution en cours (Abort)
6. Choix du type de cible, débogage (Debug) ou version finale (Release)

La version finale donne un fichier exécutable plus petit et généralement plus efficace mais elle n'autorise pas le débogage. Les fichiers sources compilés sont rangés dans le sous-répertoire obj du projet. Le programme exécutable se trouve dans le sous-répertoire bin\Debug pour la version de débogage et dans le sous-répertoire bin\Release pour la version finale. Sous Windows, le nom du programme correspond au nom du projet avec l'extension exe.

On obtient l'affichage du résultat d'un Build dans l'onglet Build log de la fenêtre Messages.

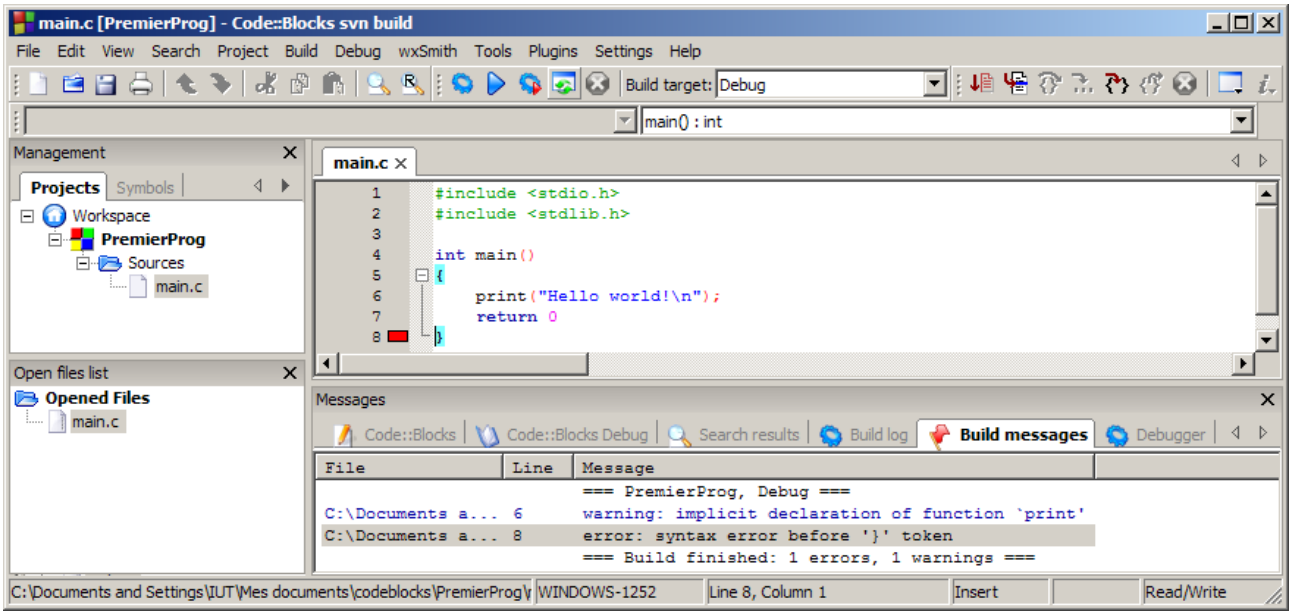



Si on lance le programme (Run), une fenêtre s'ouvre et affiche le résultat de l'exécution.




## 4. Erreur à la compilation

Les erreurs lors de la compilation du projet sont recensées dans l'onglet Build messages de la fenêtre Messages. Un simple clic sur l'erreur envoie le curseur dans le code à l'endroit où l'erreur est détectée. Un marqueur rouge dans la marge du code indique la ligne concernée. Attention, souvenez vous qu'en langage C, un point-virgule oublié en fin de ligne entraîne une erreur sur la ligne suivante. À titre d'exemple, créons volontairement deux erreurs dans le programme précédent.




Écrivons print à la place de printf et supprimons le point virgule après le return 0. Lors du Build, nous obtenons alors l'affichage ci-contre : l'absence de point-virgule apparaît comme une erreur.

À la compilation, l'usage de la fonction inconnue print (à la place de printf) génère un warning.

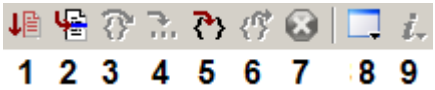
Elle aurait déclenché une erreur à l'édition de lien.

# 5. Utiliser le débogueur

## *5.1. Les commandes du débogueur*

Il est possible d'utiliser le débogueur soit à partir du menu Debug, soit grâce à la barre d'outils Debugger. On peut activer ou désactiver cette barre d'outils par View → Toolbars → Debugger.

La barre d'outils Debugger :



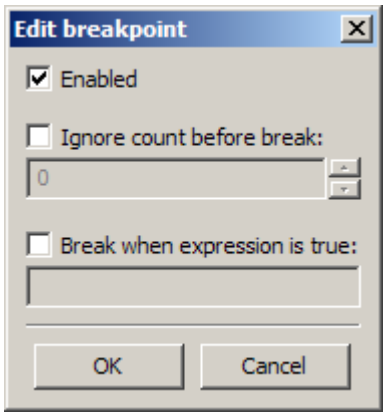

1. Exécute en mode débogueur (Debug/Continue)

   Attention, si aucun point d'arrêt n'est placé, le programme s'exécute normalement.
2. Exécute jusqu'au curseur placé dans le code (Run to cursor)
3. Exécute une ligne sans entrer dans les fonctions (Next line)
4. Exécute une instruction en assembleur (Next instruction)
5. Exécute une ligne en entrant dans les fonctions (Step into)
6. Exécute jusqu'à la fin de la fonction en cours (Step out)
7. Termine l'exécution en cours (Abort)
8. Menu d'ouverture des fenêtres de débogage
9. Menu d'informations

## *5.2. Les points d'arrêt (Breakpoint)*

Pour placer ou enlever un point d'arrêt (Breakpoint) dans le programme, il faut cliquer dans la marge juste après le numéro de la ligne. Un rond rouge doit apparaître. On peut aussi utiliser le menu Debug → Toggle breakpoint ou la touche F5. Lors de l'exécution en mode débogage, le programme s'arrêtera sur les points d'arrêt.

Il est possible de désactiver un point d'arrêt (sans le supprimer), de lui associer un nombre de passages avant arrêt ou une condition logique pour stopper. Un clic droit sur le point d'arrêt permet d'accéder à un menu d'édition du point d'arrêt (Edit breakpoint). Une boîte de dialogue s'ouvre alors et on peut choisir d'ignorer un certain nombre de passages avant de prendre en compte le point d'arrêt ou de ne s'arrêter que pour une condition logique particulière exprimée en langage C.




Lors d'un arrêt en mode débogage, une flèche jaune indique l'avancée de l'exécution du programme.

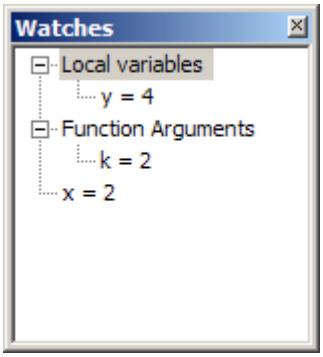
## *5.3. Visualiser l'état des variables (Watches)*

Lors du fonctionnement en mode débogage, Code::Blocks analyse automatiquement les valeurs des variables locales et des arguments des fonctions. Ces informations se trouvent dans la fenêtre

Watches.

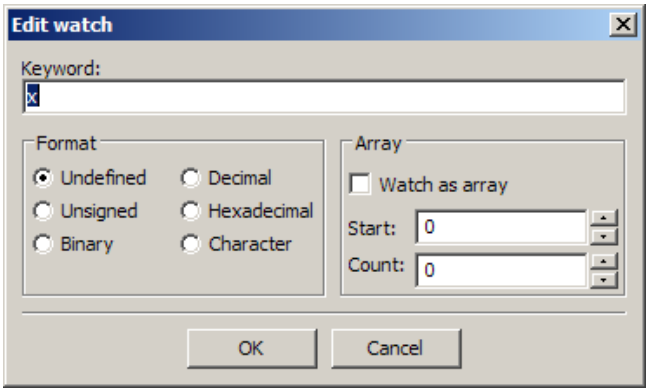
Vous pouvez ouvrir cette fenêtre grâce au menu Debug → Debugging windows → Watches. Vous pouvez aussi utiliser la barre d'outils Debugger et l'icône d'ouverture des fenêtre de débogage. La fenêtre Watches comprend par défaut deux listes, une pour les variables locales et une pour les arguments des fonctions. Les variables dans ces listes sont ajoutées et retirées de manière automatique en fonction du déroulement du programme. Un arrêt dans la fonction suivante :

```
double fct(double k)
{
      double y;

      y = k * x;

      return y;
}
```




où x est une variable globale donne l'affichage ci-dessus dans la fenêtre Watches (x a été ajouté manuellement).

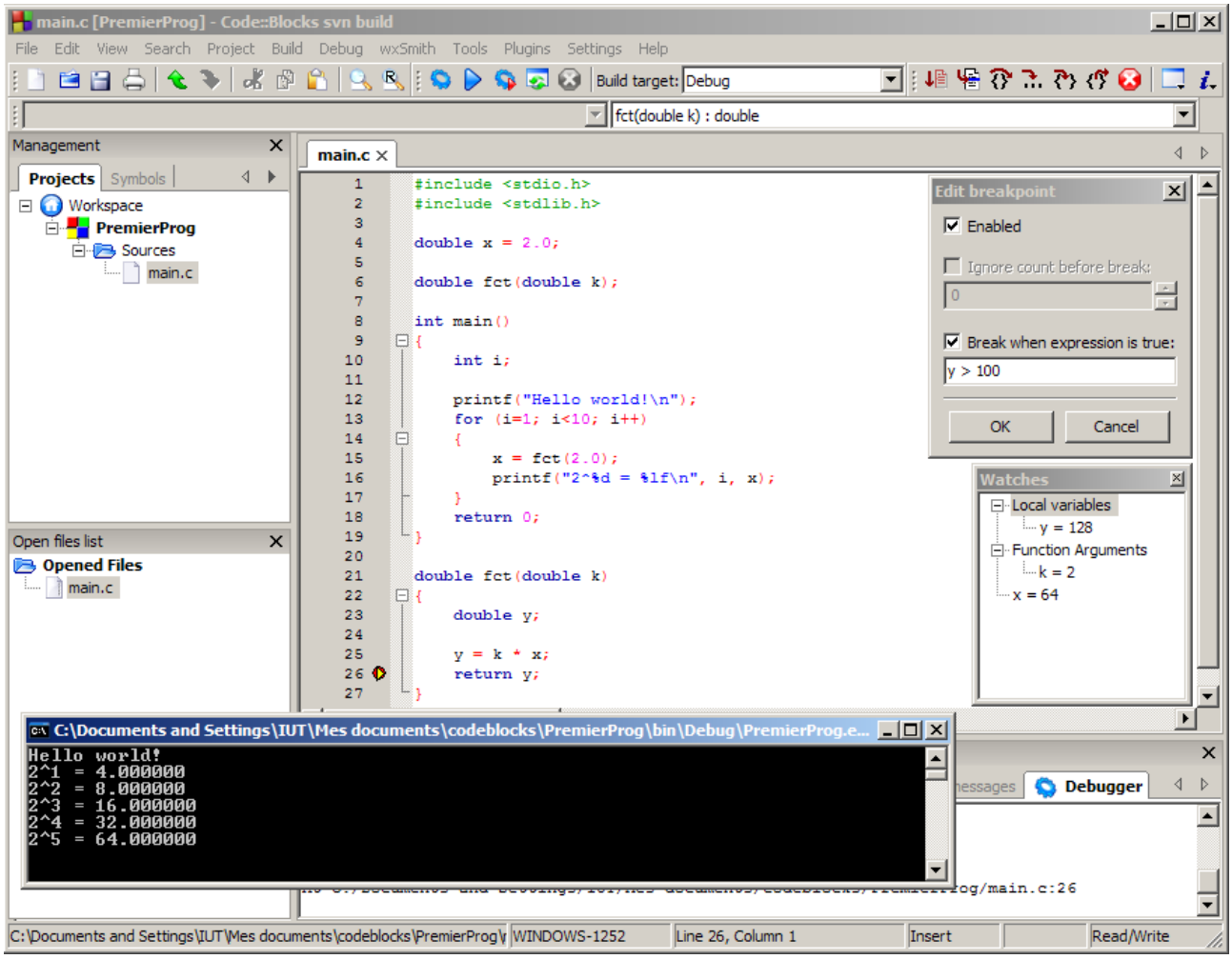
Un clic droit dans la fenêtre Watches permet d'accéder à un menu pour ajouter une variable (Add watch). Un clic droit sur une variable dans la fenêtre Watches permet de changer sa valeur (Change value) ou de modifier sa représentation (Edit watch).




Un clic droit sur une variable dans le code pendant une session de débogage permet d'obtenir un menu pour ajouter la visualisation de la variable.

### 5.4. Exemple d'une session de débogage

Le programme est arrêté par un point d'arrêt avec condition :




# 6. Autres fonctionnalités

### 6.1. Installer un fichier d'aide

Il est pratique d'obtenir rapidement de l'aide sur les fonctions des librairies standard du langage C. Pour cela, il faut associer un fichier d'aide à un appui sur la touche F1 quand le curseur est placé sur une fonction.

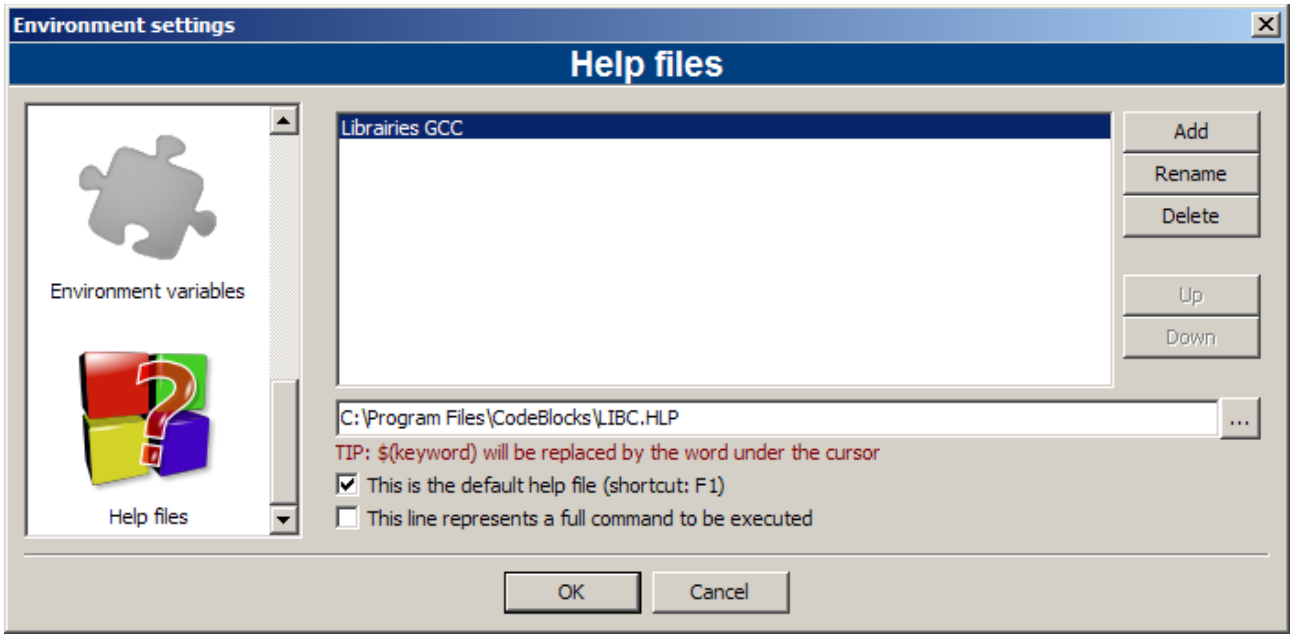
1. Allez dans le menu

   Settings → Environment...

2. Dans la fenêtre qui s'ouvre, faites défiler la partie gauche jusqu'à trouver Help files et cliquez sur Help files.

3. Cliquez sur le bouton Add et entrez un nom pour le fichier d'aide (par exemple Librairies GCC).

4. Cliquez sur Oui pour parcourir (browse) l'arborescence des fichiers.
5. Recherchez le fichier LIBC.HLP qui se trouve à l'emplacement C:\Program Files\CodeBlocks\LIBC.HLP pour une installation standard.
6. Cochez l'option This is the default help file (shortcut: F1) pour activer l'usage de la touche F1.
7. Terminez en cliquant sur OK.




Pour vérifier le fonctionnement, placez le curseur sur la fonction main et appuyez sur F1. Une fenêtre d'aide doit s'ouvrir pour vous donnez des informations sur la fonction main en langage C.

## 6.2. Formater automatiquement son code

Le langage C n'impose aucune contrainte d'écriture du code. Pour ceux qui écrivent en langage C comme des porcs, Code::Blocks intègre un outils de mise en forme automatique du code : AStyle. Pour l'utiliser, il suffit d'aller dans le menu Plugins → Source code formatter (AStyle).

On peut configurer la mise en forme du code en allant dans le menu Settings → Editor... puis en sélectionnant sur la gauche de la fenêtre Source formatter.

Avant AStyle

```
#include <stdio.h>
#include <stdlib.h>

double x = 1.0;
void fct(void);

int main()
{
int i;

printf("Hello world!\n");
for (i=1; i<10; i++) {
fct();
```

Après AStyle

```
#include <stdio.h>
#include <stdlib.h>

double x = 1.0;
void fct(void);

int main()
{
      int i;

      printf("Hello world!\n");
      for (i=1; i<10; i++)
      {
```

```
printf("2^%d = %lf\n", i, x);
}

return 0;
}

void fct(void){
x = 2 * x;
}
```

Pour en savoir plus, cliquer ici 

```
            fct();
            printf("2^%d = %lf\n", i, x);
      }

      return 0;
}

void fct(void)
{
      x = 2 * x;
}
```