

Programmation en C/C++

Série 1

Présentation de l'UE Présentation du langage C Notion de compilation Environnement de Développement Intégré

Objectifs

Présentation de l'UE, présentation du langage C, mots clés, notion de compilation, installer un EDI sur votre propre ordinateur, premier programme C.



@Copyright : L'ensemble des documents qui vous seront fournis dans le cadre de cet apprentissage est destiné à un usage personnel ou familial. Il est interdit de les modifier, publier, donner, vendre, diffuser, reproduire ou de les utiliser dans un cadre collectif.



Pédagogie utilisée

Ces séries comportent des informations et données permettant de comprendre et assimiler des bases solides en programmation. Vous n'aurez pas besoin d'acheter de livre. Ces supports, utilisés durant les cours présentiels, ou en FOD, comportent tous un ensemble important de programmes sources écrits en langage C qui vont être repris et expliqués en cours. Chaque série comporte également un nombre d'exercices corrigés important afin que vous puissiez continuer à vous entraîner chez vous car tout ne pourra pas être traité en séance.

A chaque fois, vous trouverez en premier le **cours** et la syntaxe des notions abordées puis des exemples concrets d'explications puis des lots d'exercices corrigés et de **niveau progressif**.

Pour cette UE, on pratiquera un enseignement "**en immersion**" : toutes les séances sont en salle machines ; nous serons constamment devant l'ordinateur. On parlera le minimum possible, on sera concret et pragmatique ; on passera un maximum de temps à programmer sur la machine. Au début de chaque séance, l'enseignant

présentera les nouvelles notions essentielles et les auditeurs passeront à la partie programmation immédiatement après.

En cours de séance, l'enseignant répondra aux questions, débloquera les auditeurs, reprendra devant tous, des notions de cours supplémentaires ou non comprises ; expliquera certains exercices demandant des précisions ou explications. L'enseignant jouera un rôle de coach.

L'entre-aide entre auditeurs, à voix basse est autorisée.



Attention ! Les exercices proposés dans une série sont d'un niveau progressif ; il en est de même entre une série et la suivante. Le rythme est très soutenu tout au long du semestre. Pour comprendre les notions d'une série il faut traiter celles-ci du début à la fin puis passer à la suivante. L'enchaînement des cours et des exercices a été réfléchi et pensé d'un point de vue pédagogique.



Pour passer à la série n il faut avoir assimilé la série $n-1$. Car, plus on avance, plus on mélange et utilise les notions vues précédemment.



Celui qui lit les programmes sources proposés sans les chercher ne progressera pas et, au final, ne saura pas programmer. Excepté pour le premier programme en C, n'utilisez pas la fonction *copier/coller* pour compiler et exécuter les programmes : cherchez-les et écrivez-les !!

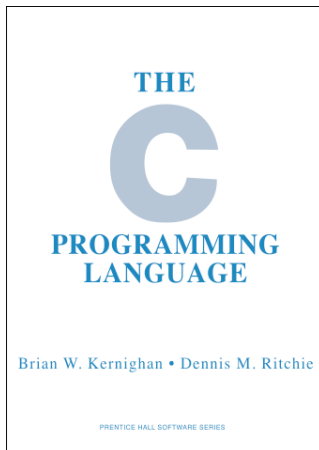
Pour devenir un bon développeur en C, il vous faudra refaire et chercher **tous** les exercices ; cela prendra du temps mais vous ne le regretterez pas en fin de semestre.



Il est vivement conseillé de programmer au moins 2 fois 2 heures par semaine, à la maison, en plus de la séance hebdomadaire.

Introduction

40 ans mais le plus utilisé dans le monde !



Le langage C a été inventé en 1972 par les américains Dennis Ritchie et Ken Thompson.

En 1978, Kernighan publie le livre de référence *The C Programming Language* qui décrit le langage.

C est le langage le plus utilisé dans le monde à égalité avec le langage Java ; loin devant les autres. Il est enseigné dans les universités et écoles du monde entier.

Un langage normalisé

Le langage a été normalisé en 1989 par l'ANSI (*American National Standards Institute*) sous la norme dite ANSI C ou C89. Il a été ensuite normalisé en 1990 par l'ISO (*International Standards Organization*) puis en 1994, 1996, 1999 et 2011.

La normalisation permet d'utiliser des standards communs et assure une compatibilité des codes sources.

32 mots clés

Selon la norme ANSI, le langage C comporte uniquement 32 mots clés :

if ; else ; switch ; case ; default ; auto ; register

extern ; typedef

do ; while ; for

char ; double ; float ; int ; long ; short ; const ; signed ; volatile ;

unsigned ; void ; enum ; union ; struct ; static ;

printf ; scanf ; return ; break ; goto ; continue ; sizeof

C'est donc en jouant uniquement avec ces quelques mots clés que l'on pourra écrire de très longs et très beaux programmes...

Espaces, casse, fin d'instruction

Le C est sensible à la casse : des identifiants (nom de variable, nom de fonction...) en majuscules ou avec des minuscules sont considérés comme différents.

Les espaces, tabulations, retour à la ligne, lignes blanches peuvent être utilisés librement pour la mise en page des programmes sources.

Chaque instruction (expression écrite en langage C) est terminée par un point virgule.

Bloc d'instructions

On parle de bloc d'instructions lorsque plusieurs instructions sont regroupées entre des accolades.

Exemple

```
if (val == 0)
{
    val++
    val2 = val
}
```



Sur MacBook

- Une "{" s'obtient en tapant sur les touches ALT et (
- Une "}" s'obtient en tapant sur les touches ALT et)

Ligne et bloc de commentaires (// et /* */)

On peut écrire une ligne de commentaires en la faisant commencer par //

Exemple

```
// Saisie des données
```

Une ou plusieurs lignes de commentaires peuvent être ajoutées entre les opérateurs /* et */

Exemple

```
/* Traitement des données
Puis affichage des données*/
```

C'est une bonne habitude de documenter son code source en y ajoutant des commentaires. Les commentaires ne sont pas pris en compte durant l'opération de compilation.

Un langage typé

Les variables, qui permettront de stocker des informations sont typées. Avant d'utiliser une variable on devra donc la déclarer en précisant son nom et le type de données qu'elles contiendra (entier, entier long, réel, caractère...)

Il existe des types prédéfinis de variables : *char*, *short*, *int*, *long*, *float*, *double*, *unsigned*. Le type booléen n'existe pas.

Un code source compact

Le langage C est réputé être court, syntaxiquement. En effet, il est possible de rendre le code source des programmes plus compact en utilisant différents opérateurs ou techniques de programmation que nous verrons par la suite.

Avantages de C

Le langage est normalisé, mondialement connu et utilisé. Il bénéficie d'une grande communauté d'utilisateurs, de librairies qui peuvent grandement aider les développeurs.

Un bon programme en langage C permet de minimiser les besoins en mémoire, de maximiser les performances en fonction de la machine cible. L'exécutable ne nécessite aucun environnement, outils ou bibliothèque logicielle pour fonctionner. Du coup, C est très rapide à l'exécution : c'est pour cela qu'il est utilisé dans le calcul scientifique, dans le domaine embarqué, pour la réalisation de systèmes d'exploitation comme Unix, Windows ou Linux, pour l'écriture de nouveaux compilateurs.

C dispose de peu de mots clés qui seront donc vite connus.

Inconvénients de C

Les compilateurs C ne vérifient pas certains types d'erreurs et le programme peut donner des résultats surprenants à l'exécution ou planter : il faudra être rigoureux lors de l'écriture du code.

C gère mal les exceptions et les chaînes de caractères ; il ne prend pas en charge la programmation orientée objet.

La gestion de la mémoire est assez complexe.

Des problèmes peuvent survenir à l'exécution du fait que le comportement des exécutables peut être lié à la machine qui les exécute.



Nom du fichier source

Le programme source, écrit en langage C, devra être enregistré dans un fichier dont le nom comporte au maximum 31 caractères et commence par une lettre. Le nom ne devra comporter ni d'espaces ni de caractères de contrôles excepté le tiret bas "_". Le fichier aura pour extension ".c". Le nom du fichier ne doit pas être un mot clé du langage C.

Structure et organisation d'un programme

- 1- Directives à destination du préprocesseur
- 2- Déclaration (éventuelle) des variables externes
- 3- Écriture (éventuelle) des fonctions secondaires

4- Écriture du *main()* /* Programme principal */
{
/* corps du programme principal */
}

Le "main" est le programme principal.

Nous reviendrons sur la structure d'un programme à la fin de cette série et dans la série sur les fonctions.

Un langage compilé

Le C est un langage compilé.

Le développeur écrit un code source qui est ensuite soumis au verdict d'un compilateur qui détecte les éventuelles erreurs d'écriture.

Notion de compilation

Un compilateur est un programme qui permet de transformer un code source (écrit dans un langage de programmation – ici le langage C -) en un programme "exécutable" par la machine sous réserve que le code source ne comporte pas d'erreur. En effet, il faut respecter la syntaxe du langage de programmation qui est utilisé. Le compilateur va signaler, éventuellement, des erreurs (**errors**) et des avertissements (**warnings**). Les erreurs sont à corriger obligatoirement pour générer un code exécutable. Les avertissements doivent être lus avec attention car, souvent, ils soulignent une possibilité d'erreur et donc, un risque éventuel lors de l'exécution future du programme (par exemple, une variable a été déclarée mais n'est pas utilisée, on n'utilise pas le résultat obtenu dans une variable...)

Tant que des erreurs (de syntaxe ou de sémantique) sont présentes il faudra les corriger et compiler à nouveau le programme source.

En résumé,

- 1- Il est préférable de disposer d'un Environnement de Développement Intégré (EDI), comme nous allons le voir, et d'un compilateur C installé sur l'ordinateur,
- 2- On crée un programme dit "programme source" écrit en langage C ; on corrige ses erreurs,
- 3- On compile ce programme afin de générer un programme dit "exécutable"
- 4- On exécute le programme.

L'opération de compilation

Le code source est en quelque sorte un fichier texte écrit par le développeur : il n'est pas compréhensible par le processeur de la machine. Il va devoir être compilé et l'opération de compilation comporte quatre phases :

1- Précompilation par le préprocesseur : le code source est nettoyé, les espaces et tabulations supprimées, les commentaires remplacés par des blancs. Les directives de précompilation commençant par un hashtag ou croisillon (#) sont utilisées ; les autres codes sources appelés via des fonctions ou des bibliothèques sont ajoutés...

2- Compilation

Cette étape importante vérifie la cohérence des codes sources du programme et identifie les erreurs de syntaxe (mauvaise orthographe de mots clés, mauvaise syntaxe d'une instruction...) ou de sémantique (mauvaise correspondance de type, erreur d'affectation, paramètres incohérents...). Des codes en assembleur sont générés s'il n'y a pas d'erreur. Le code est également optimisé lors de cette étape.

3- Assemblage

Cette étape transforme tous les fichiers en assembleur en des fichiers binaires (fichiers "objet") utilisables par le processeur de l'ordinateur.

4- Edition de liens

Cette étape lie les différents fichiers objets ainsi que les bibliothèques de fonctions utilisées. S'il n'y a pas d'erreur de référencement, le fichier "exécutable" est enfin généré.



Les fichiers source ont l'extension `.c`
Les fichiers objet ont l'extension `.obj` ou `.o`
Les fichiers exécutables ont l'extension `.out`

Lors de l'opération de compilation, il est possible, grâce à des options, d'optimiser plus ou moins le code exécutable.

Remarque

Il est à noter que les erreurs de logique ne sont pas détectables par le compilateur : si vous avez divisé deux nombres au lieu de les additionner, le programme exécutable sera généré mais donnera un résultat faux à l'exécution.

Environnement de développement

Pour saisir, corriger, compiler et visualiser nos programmes nous utiliserons un Environnement de Développement Intégré (EDI en français ; IDE en anglais) : cette année, nous avons choisi l'EDI gratuit nommé "Codeblocks" qui intègre également un compilateur C/C++.

Cet EDI est déjà installé sur toutes les machines de la salle informatique que nous utilisons ; vous n'avez donc rien à installer !

Un EDI est pratique et fait gagner du temps : les lignes de commentaires ou mots clés sont coloriés on peut indenter le code source, utiliser un débbugger, utiliser des fonctions remplacer, rechercher...



Installation de l'environnement de développement Codeblocks (pour programmer chez soi !)

Étape 1

Téléchargement de l'EDI Codeblocks qui intègre le compilateur C (chez vous !)

Taper *codeblocks* sur www.google.fr,

Cliquer le lien qui renvoie au site de codeblocks (<http://www.codeblocks.org/home>)

Cliquer sur l'onglet *downloads* (téléchargements)

Cliquer sur *download binary release* (on télécharge la version exécutable)

Choisissez la version adaptée à votre système d'exploitation présent sur votre ordinateur (Windows XP, 7, 8, 10, MacOSX...)

Choisissez une version "mingw" car elle intègre un compilateur C : par exemple, `codeblocks-17.12mingw-setup.exe` ou une version plus récente. Cliquez ensuite sur le lien BerliOS ou Sourceforge.net pour télécharger le fichier.



• **Pour MacBook** télécharger la version 13.12 ou supérieure.

Avant de lancer *Codeblocks* ; lancez le *Terminal* à partir du *Dock* et taper le code suivant :

`xcode-select --install`

Xcode sera alors téléchargé et installé et permettra le bon fonctionnement de Codeblocks.

Étape 2

Installation de l'EDI Codeblocks qui intègre le compilateur C (chez vous !)

Une fois le fichier téléchargé, double cliquez dessus pour l'installer (l'EDI et le compilateur vont être installés) ;



ne modifiez pas le répertoire d'installation qui est proposé par défaut !).

Une fois l'EDI Codeblocks installé, une icône de raccourci sera présente sur le



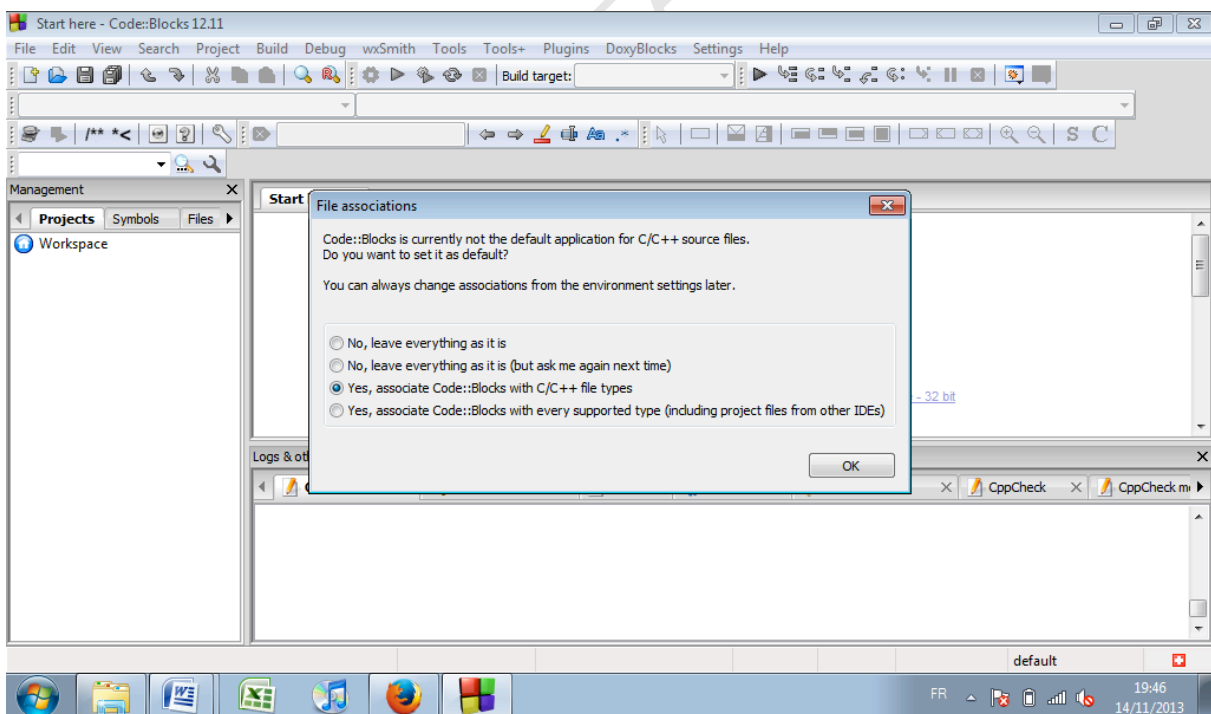
bureau : Cliquez dessus pour lancer Codeblocks!

Étape 3

Paramétrage de l'EDI Codeblocks (chez vous !)

Lancer Codeblocks afin de le paramétrer.

Codeblocks va vous demander si vous souhaitez le définir comme EDI gérant les programmes C/C++ par défaut. (Voir capture d'écran ci-après). Il faut accepter.



Il va falloir ensuite définir un projet (une fois pour toutes !) où seront regroupés tous vos programmes sources (*Project*).

Pour créer ce projet :

File/new/Project... Dans le menu catégories choisir "*Console application*" puis cliquer *Go* puis *Next*.

Sélectionner le langage de programmation

GNU GCC Compiler : choisir "C" puis cliquer *Next*.

Titre du projet

Dans la fenêtre qui s'ouvre alors, taper par exemple "sourcesC" comme titre de projet (*Project title*).

Répertoire de sauvegarde

Pour le répertoire de sauvegarde de ces fichiers sources écrits en langage C (*Folder to create project in* :) choisir, par exemple, C:\ et un dossier "sourcesC" va être créé à la racine du disque dur C:\.

Vos programmes seront donc stockés sur le disque dur dans le répertoire C:\sourcesC.

Sur la gauche de l'écran, dans la fenêtre *Management* apparaît maintenant votre projet nommé sourcesC.



Enlever le fichier main.c()

Cliquez sur votre projet à gauche de l'écran puis cliquez droit sur le fichier *main()* puis supprimer ce fichier (*remove file from project*).



Dans un même projet **un seul *main()* n'est possible** donc l'une des solutions possibles consiste à travailler toujours sur le même fichier source et à le modifier d'un exercice à l'autre. Une autre façon consiste à supprimer le fichier source précédent et à créer un nouveau fichier source.

A partir de ce moment-là vous pouvez commencer à programmer.



Créer un nouveau programme source en C

A chaque fois que vous souhaitez créer un nouveau programme en langage C il faudra réaliser :

File/new/Empty file (fichier vide) ; *Codeblocks* vous demandera si ce fichier doit être ajouté au projet en cours et il faudra répondre oui !

Release ou Debug

Suivant les versions, il vous sera demandé de sélectionner la cible du fichier

Sélectionnez "Release" puis validez.

Le mode **Release** va permettre de créer un exécutable optimisé mais ne comportant pas des informations de débogage.

Le mode **Debug** va permettre de créer un exécutable non optimisé mais comportant des informations de débogage.



Il faudra alors **choisir un nom de fichier** pour l'enregistrement du code source (**n'oubliez pas l'extension ".c"**). Exemple : *prg1.c*



Éviter les espaces et caractères spéciaux dans les noms de fichiers ainsi que les mots clés des langages C/C++.

Premier programme en C

! Tous les programmes proposés ici devront être cherchés, écrits par vos soins, compilés puis exécutés afin de vérifier leur bon fonctionnement. Recopier des morceaux de code source et les compiler n'a pas d'intérêt. Devenir un bon développeur informatique demande de passer de nombreuses heures à écrire des lignes de code et corriger ses erreurs.

Nous allons créer un **premier programme** écrit en langage C.


Créez un nouveau fichier : cliquez sur *File* (fichier) puis *New* (nouveau) puis *Empty file* (nouveau fichier vide). Donnez un nom à votre fichier (par exemple : *essai1.c*). Le fichier créé sera donc *essai1.c*

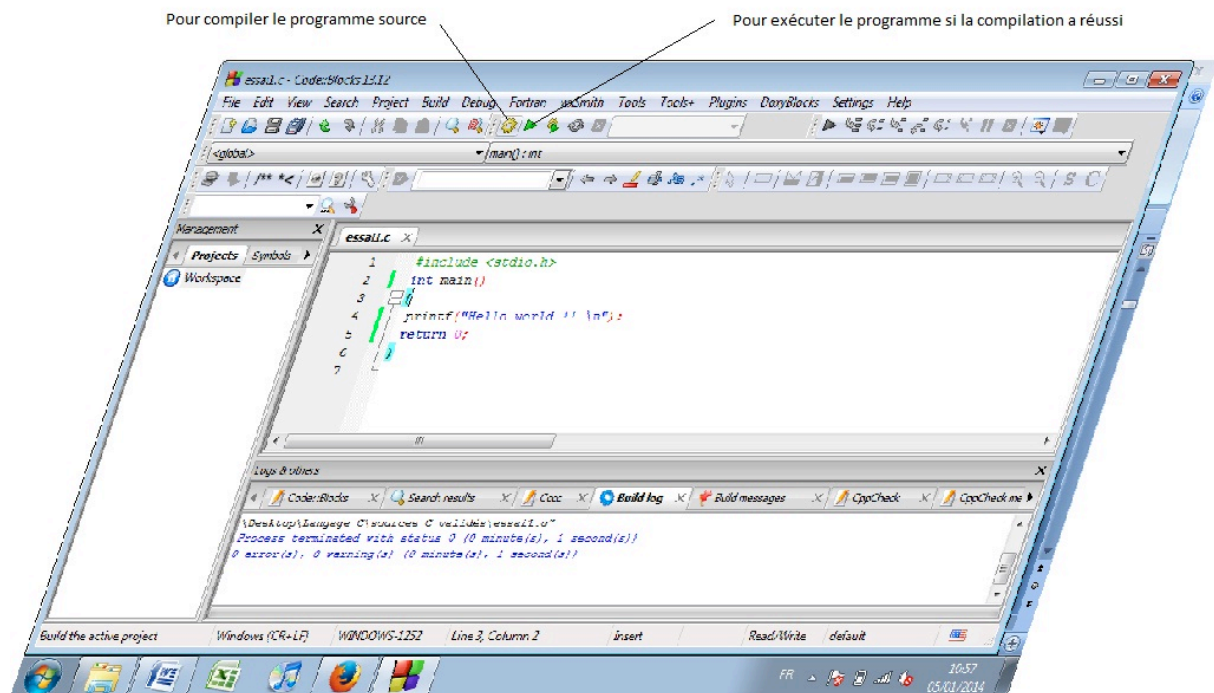
Saisissez (pas de copier/coller !) tout le code source ci-après sans modification.


```
/* 1er programme en langage C */  
#include <stdio.h>
```

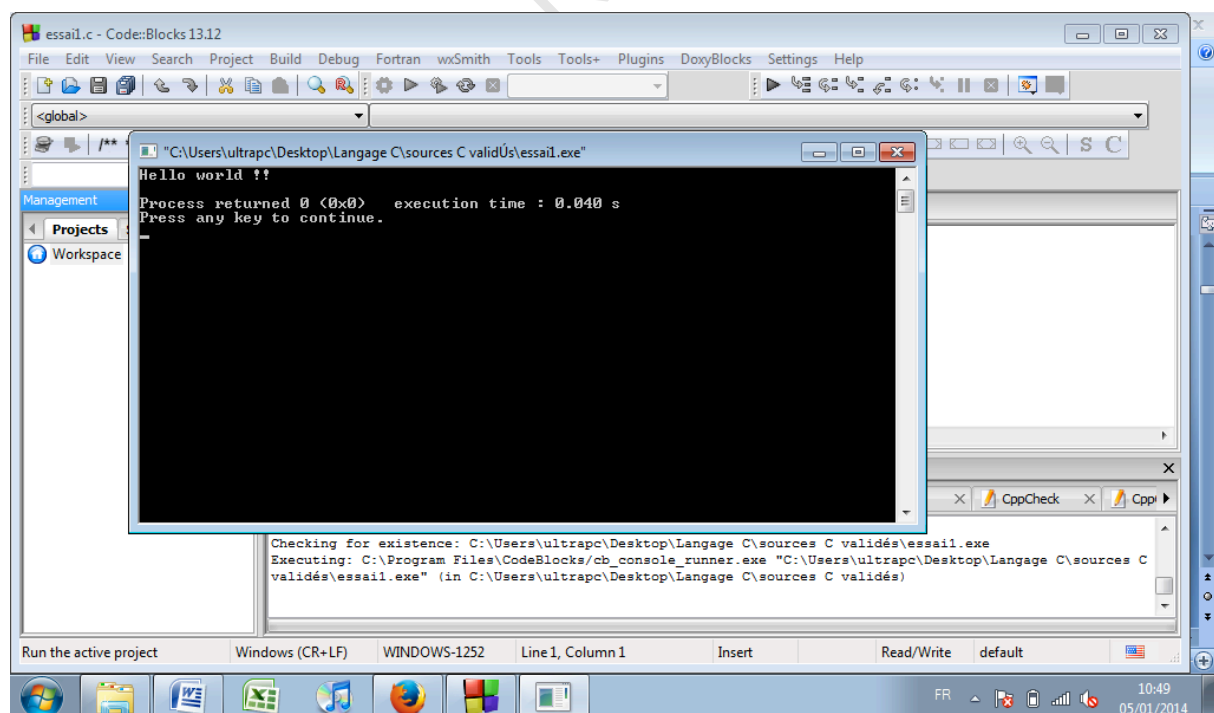
```
int main()  
{  
    printf("Hello world !! \n");  
    return 0;  
}
```

Enregistrez ce programme source : (*file/save as*) : votre programme source est désormais stocké et enregistré sur votre disque dur dans un fichier nommé *essai1.c*.

! Une fois le programme saisi, corrigez les erreurs éventuelles puis enregistrez à nouveau puis **lancez la compilation** (engrenage jaune ). En bas de l'écran, il s'ouvre une fenêtre : la compilation est réussie lorsqu'il s'affiche *0 error(s)* c'est-à-dire aucune erreur (voir écran ci-après). S'il y a des erreurs, il faut les corriger toutes en suivant les indications affichées par le compilateur puis enregistrer le fichier puis lancer à nouveau la compilation.



! Si la compilation a réussi, vous pouvez alors exécuter le programme que vous avez compilé (cliquez sur *run* -flèche verte- ): une fenêtre s'ouvre alors et il s'affiche bien "Hello world !! " :



Remarque

L'instruction **printf()** permet d'afficher à l'écran du texte placé entre guillemets ou le contenu de variables.

Syntaxe : `printf("texte... ");`



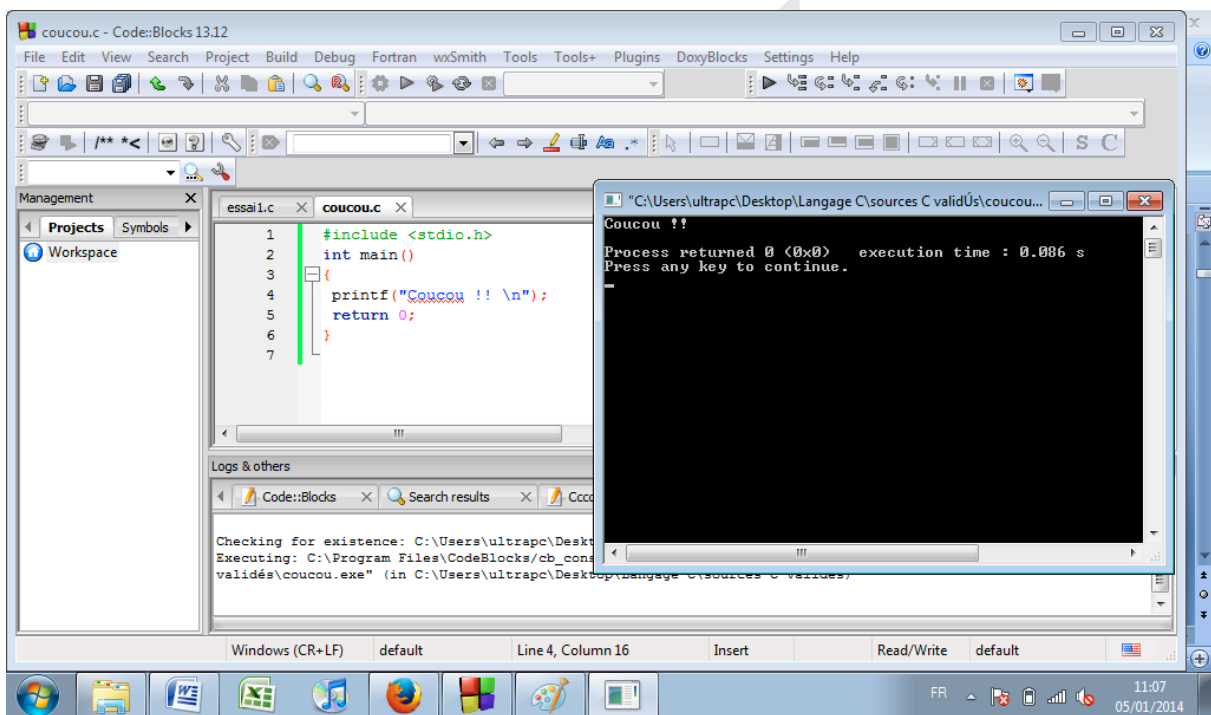
L'opérateur `\n` : après l'affichage, le curseur passera au début de la ligne suivante.



Attention ! Avant de recompiler un programme ou d'en créer et compiler un nouveau, pensez à fermer la fenêtre où s'est exécuté le programme précédent.

Exercice 1

Créez un nouveau programme source ; recopiez le précédent et modifiez-le afin qu'il affiche "Coucou !!" Vous enregistrerez votre programme source dans un fichier nommé *coucou.c*.



Exercice 2

Créez un nouveau programme qui affiche A, B, C, D, E sur une ligne et 1, 2, 3, 4, 5 sur la suivante.