

# Algorithmique et Programmation 1

## Introduction au langage de programmation cible

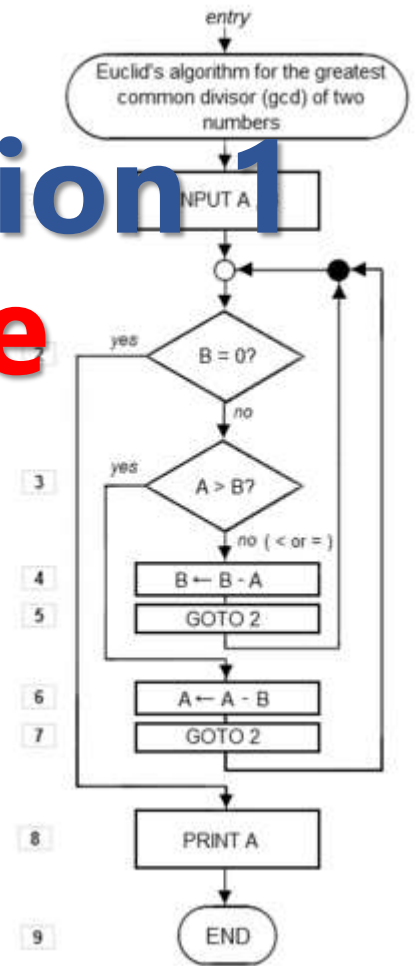
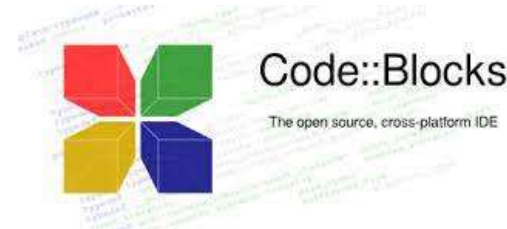
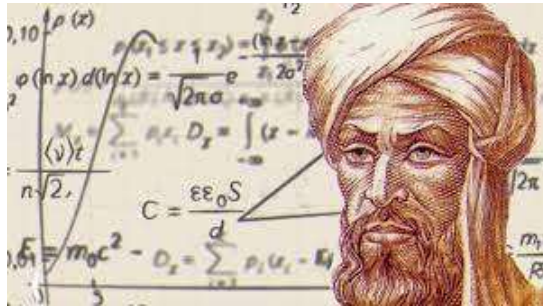
### Langage C

Prof. Ousmane SALL

Université de THIES - UFR Sciences et Technologies

Département Informatique

[osall@univ-thies.sn](mailto:osall@univ-thies.sn)



# A propos de moi



- Enseignant-Chercheur à l'UFR SET- Université Iba DER THIAM de THIES  
<https://sites.google.com/a/univ-thies.sn/osall751/>
- Enseignements:
  - Algorithmique et Programmation(C, Java, PHP)
  - Programmation WEB dynamique(HTML 5 CSS, PHP, MySQL, CMS,...)
  - Programmation Java, Dart, TypeScript
  - Programmation Java, Jakarta EE, JSF, Spring, SpringBoot, Angular
  - Technologies Mobiles Android, Xamarin, Ionic, Flutter
  - Programmation .Net, C#
  - Gestion de Projet Informatique
  - Génie Logiciel, Qualité et Métrique du Logiciel
- Contact:
  - [osall@univ-thies.sn](mailto:osall@univ-thies.sn)
  - UFR SET, Université de THIES -Dpt Informatique, BP 967 THIES.



# Une sagesse chinoise...

*« J'écoute et j'oublie; je lis et je comprends; je fais et j'apprends »*  
[Proverbe chinois]



# Programme d'Algorithmique et **Programmation 1**

- Introduction Générale
- Notions de base en Algorithmique
- Saisie et Affichage en algorithmique
- **Introduction au langage de programmation cible: Langage C**
- Les Structures de Contrôle
- Les tableaux
- Sous-algorithmes/Fonctions

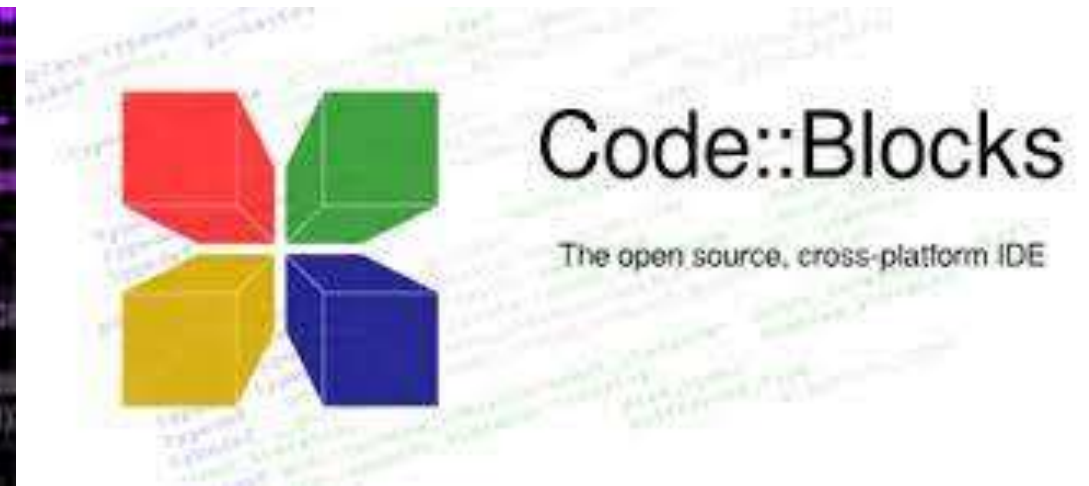
# Contenu de cette partie

- Programmation C
- Installation et Configuration de l'environnement de développement
- Structure générale d'un programme en C
- Concepts de base d'un programme en C
- Compléments et Exemples



# Objectifs/Compétences visé(e)s

- Ce chapitre présente le **langage C et ses concepts de base**



# Programmation C

# Langages informatiques

- Un langage informatique est un outil permettant de donner des ordres (**instructions**) à la machine
- A chaque instruction correspond une action du processeur
- Intérêt : écrire des programmes (suite consécutive d'instructions) destinés à effectuer une tâche donnée. Exemple: un **programme** de gestion de comptes bancaires
- Contrainte: être compréhensible par la machine



# Programme

- Une fois que l'on a un algorithme, on pourrait par exemple l'implémenter en langage C :

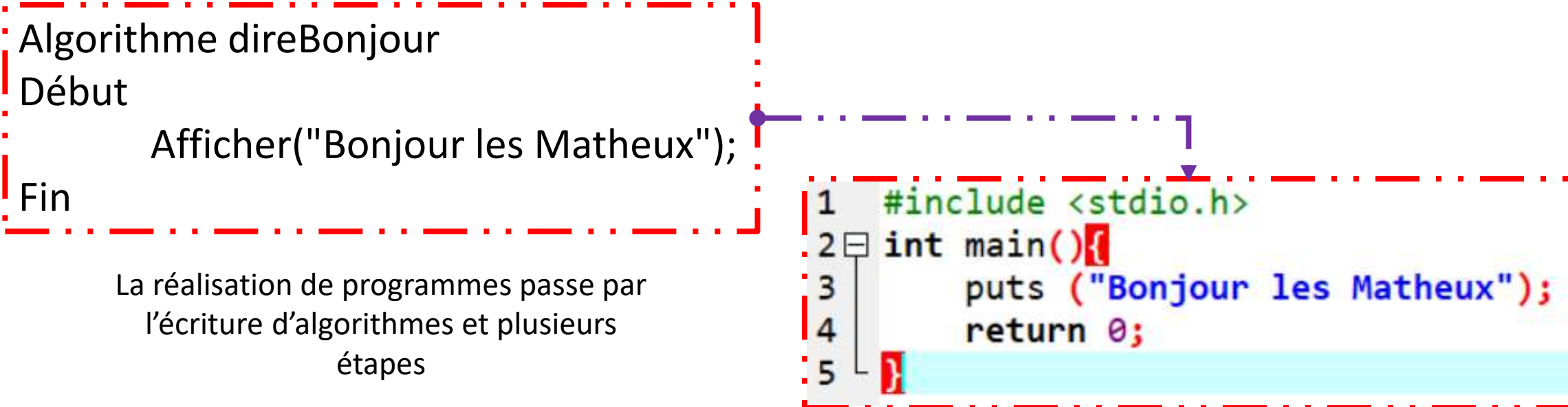
Algorithme direBonjour

Début

Afficher("Bonjour les Matheux");

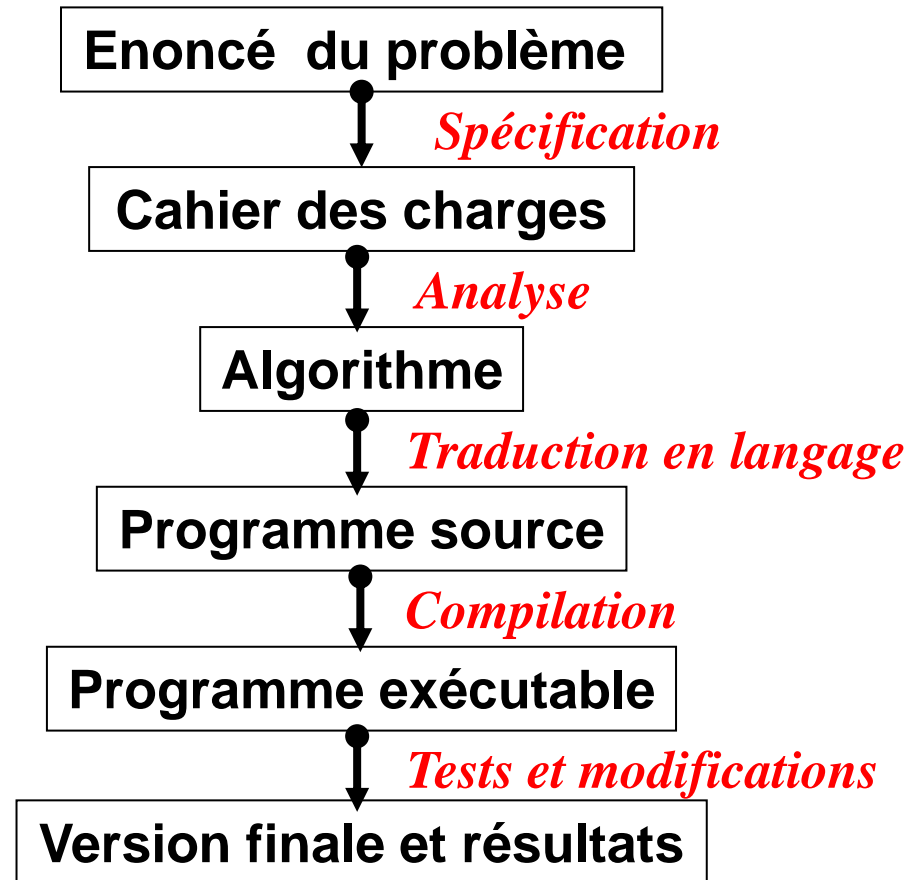
Fin

La réalisation de programmes passe par  
l'écriture d'algorithmes et plusieurs  
étapes



```
1 #include <stdio.h>
2 int main(){
3     puts ("Bonjour les Matheux");
4     return 0;
5 }
```

# Etapes de réalisation d'un programme



Langage C par exemple

La réalisation de programmes passe par l'écriture d'algorithmes  
⇒ D'où l'intérêt de l'**Algorithmique**

# Langage C

- En 1972, Dennis RITCHIE a créé le langage C, un langage de haut niveau, pour écrire le système d'exploitation Unix. La conception de ce langage a été régie par les pré requis suivants :
  - la souplesse
  - la fiabilité
  - la portabilité
  - les possibilités de l'assembleur



# Langage C: Historique

- **1960 – Algol60**

Très abstrait, donne le Pascal, PL/I et CPL

- **1967 – BCPL par Martin Richards**

Basic Combined Programming Language

- **1970 – Langage B par Ken Thompson**

Afin d'assurer l'évolution de Unix écrit en assembleur,  
son créateur crée ce langage inspiré du BCPL

- **1972 – Langage C par Dennis Ritchie et Ken Thompson**

Après modification du langage B



# Structure général d'un programme C

Le programme C le plus simple... et le plus inutile (car il ne fait rien !) est le suivant :

```
#include<stdio.h>
int main(){
    return 0;
}
```

Ce programme crée le **programme** Rien.c contenant la **fonction** main, qui est nécessaire pour produire un code **exécutable**. De plus il déclare que:

- cette fonction est vide : son **corps** (délimité par { et }) est vide,
- cette fonction ne retourne que 0 : c'est le **int** devant le nom de la fonction.

# Structure général d'un programme C

Un programme C se présente de la manière suivante :

**directives au préprocesseur**  
**déclarations de variables globales**  
**fonctions secondaires**

```
int main ()  
{  
    déclaration de variables internes  
    instructions  
    return 0;  
}
```



# La fonction main

- En-tête de la fonction main :

```
int main(){  
    return 0;  
}
```

- Pour commencer :
  - En-tête standard à apprendre par cœur
  - Chaque programme possède une fonction **main**
- Lors du démarrage du programme :
  - La fonction **main** est recherchée
  - Son bloc d'instructions est exécuté
  - S'il n'y a pas de fonction **main**, le programme ne démarre pas

# Interpréteur/Compilateur

- Comment rendre les instructions plus sophistiquées compréhensibles par l'ordinateur ?
- **Traduire** les séquences d'instructions de haut niveau en instructions-machine directement exécutables par le microprocesseur
- Selon ses caractéristiques, un tel traducteur est appelé **compilateur** ou **interpréteur**.
- L'ensemble des instructions de plus haut niveau qu'un compilateur ou un interpréteur est capable de traiter constitue un **langage de programmation**.

# Compilateur

Un compilateur informatique est un programme qui traduit le code source (compréhensible par les humains) en code binaire (compréhensible par les machines). Le but étant de générer un programme exécutable par un ordinateur.



bonjour.c

gcc



bonjour.exe

```
1  #include <stdio.h>
2  int main() {
3      puts ("Bonjour les Matheux");
4      return 0;
5  }
```



Compilateur



Programme exécutable

La compilation informatique désigne le procédé de traduction d'un programme, écrit et lisible par un humain, en un programme exécutable par un ordinateur.

# La compilation

- C est un langage compilé. Cela signifie qu'un programme C est décrit par un fichier texte appelé fichier source. Ce fichier n'est pas exécutable par le microprocesseur, il faut le traduire en langage machine. Cette opération est effectuée par un programme appelé compilateur.
- **La compilation d'un programme C se décompose en 4 phases successives:**
  1. **Le traitement par le préprocesseur** : le fichier source est analysé par un programme appelé préprocesseur qui effectue des transformations purement textuelles (remplacement de chaînes de caractères, inclusion d'autres fichiers source, etc.).
  2. **La compilation** : au cours de cette étape, le fichier engendré par le préprocesseur est traduit en assembleur, c'est à dire en une suite d'instructions qui sont chacune associées à une fonctionnalité du microprocesseur (faire une addition, une comparaison, etc.).

<http://www.lifl.fr/~marquet/ens/pdc/tpic-002.html>

# La compilation

- **La compilation d'un programme C se décompose en 4 phases successives:**

- 3. L'assemblage** : cette opération transforme le code assembleur en un fichier binaire, c'est-à-dire en instructions directement compréhensibles par le processeur. Le fichier produit par l'assemblage est appelé fichier objet (.o).
- 4. L'édition de liens** : un programme est souvent séparé en plusieurs fichiers source (ceci permet d'utiliser des bibliothèques de fonctions standard déjà écrites comme les fonctions d'affichage par exemple). Une fois le code source assemblé, il faut donc lier entre eux les différents fichiers objets. L'édition de liens produit alors un fichier exécutable.

**La commande gcc invoque tour à tour ces différentes phases.** Des options de gcc permettent de stopper le processus de compilation après chaque des phases.


<http://www.lifl.fr/~marquet/ens/pdc/tpic-002.html>


# Utiliser un éditeur sous Windows

- Utiliser un **environnement de développement intégré (EDI)**, par exemple **Code::Blocks**
- Le programmeur n'appelle pas explicitement les commandes **cc**, mais elles sont appelées par l'EDI.
- **Code::Blocks** est un environnement de développement intégré libre et multiplate-forme. Il est écrit en C++ et utilise la bibliothèque wxWidgets. Code::Blocks est orienté C et C++, mais il supporte d'autres langages comme FORTRAN ou le D.

<https://fr.wikipedia.org/wiki/Code::Blocks>

**Code::Blocks**





Code::Blocks sur Mac OS

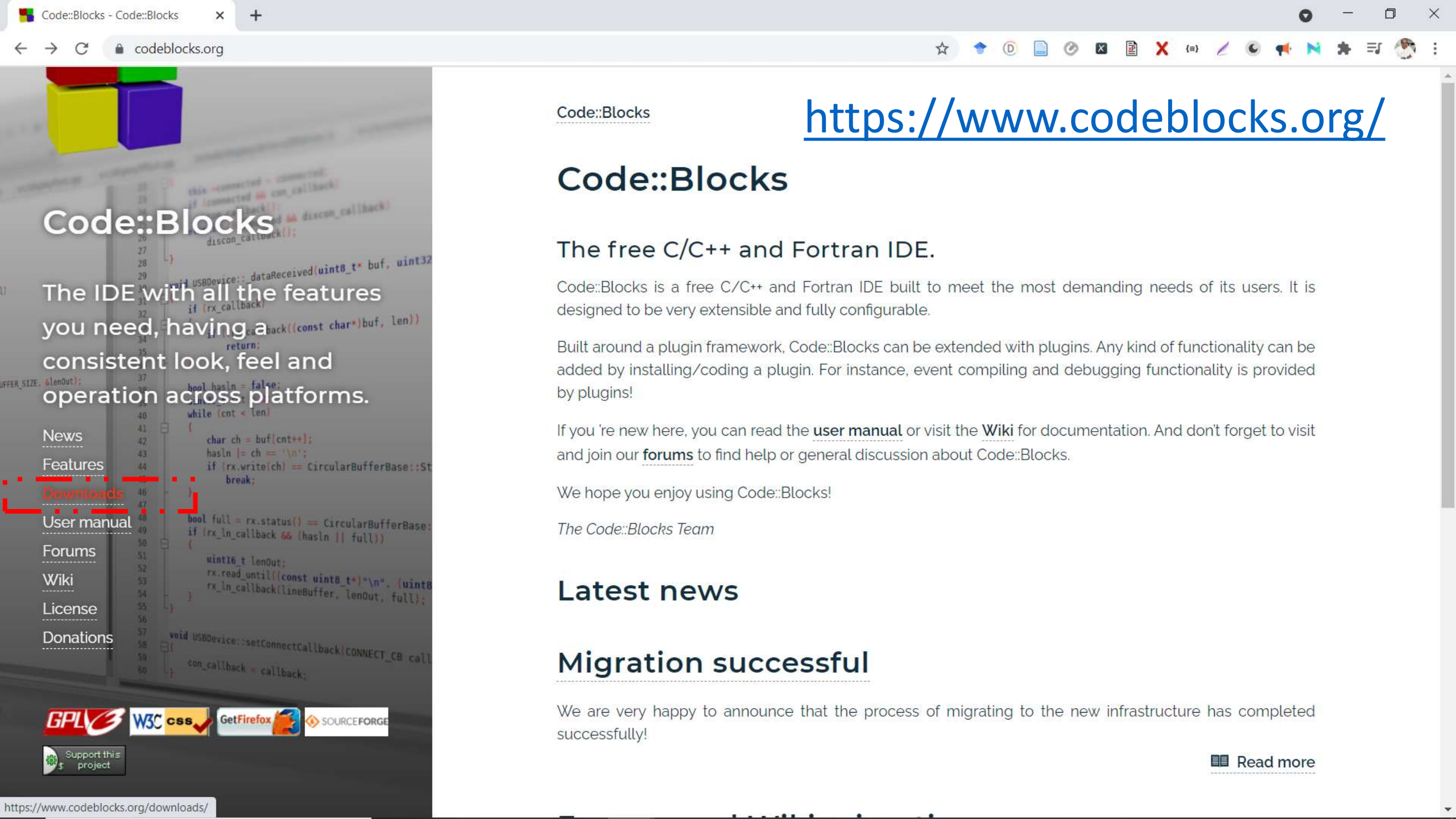
**Informations**

Développé par	The Code::Blocks team
Dernière version	20.03 (29 mars 2020)
Dépôt	<a href="https://svn.code.sf.net/p/codeblocks/code/trunk">svn.code.sf.net/p/codeblocks/code/trunk</a>
Écrit en	C++ et wxWidgets
Interface	WxWidgets
Système d'exploitation	Multiplateforme
Environnement	Multiplate-forme
Formats lus	Code::Blocks Dependencies (d), Code::Blocks Project (d), Code::Blocks Workspace Layout (d), Code::Blocks lexer (d) et Code::Blocks wxSmith resource (d)
Formats écrits	Code::Blocks Project (d)
Type	Environnement de développement intégré
Licence	Licence publique générale GNU version 3
Site web	<a href="http://www.codeblocks.org">www.codeblocks.org</a>

[modifier](#) - [modifier le code](#) - [voir Wikidata](#) ([aide](#))



# Installation et Configuration de l'environnement de développement



# Code::Blocks

The IDE with all the features you need, having a consistent look, feel and operation across platforms.

[News](#)

[Features](#)

[Downloads](#)

[User manual](#)

[Forums](#)

[Wiki](#)

[License](#)

[Donations](#)

Code::Blocks

<https://www.codeblocks.org/>

## Code::Blocks

### The free C/C++ and Fortran IDE.

Code::Blocks is a free C/C++ and Fortran IDE built to meet the most demanding needs of its users. It is designed to be very extensible and fully configurable.

Built around a plugin framework, Code::Blocks can be extended with plugins. Any kind of functionality can be added by installing/coding a plugin. For instance, event compiling and debugging functionality is provided by plugins!

If you're new here, you can read the [user manual](#) or visit the [Wiki](#) for documentation. And don't forget to visit and join our [forums](#) to find help or general discussion about Code::Blocks.

We hope you enjoy using Code::Blocks!

*The Code::Blocks Team*

## Latest news

### Migration successful

We are very happy to announce that the process of migrating to the new infrastructure has completed successfully!

[Read more](#)



The IDE with all the features you need, having a consistent look, feel and operation across platforms.

News

## Features

## Downloads

User manual

## Forums

Wiki

## License

## Donations



<https://www.codeblocks.org/downloads/binaries>

## Downloads

There are different ways to download and install Code::Blocks on your computer:

- Download the binary release

This is the easy way for installing Code::Blocks. Download the setup file, run it on your computer and Code::Blocks will be installed, ready for you to work with it. Can't get any easier than that!

- Download a nightly build

There are also more recent so-called nightly builds available in the [forums](#). Please note that we consider nightly builds to be stable, usually, unless stated otherwise.

- Other distributions usually follow provided by the community (big "Thank you!" for that!). If you want to provide some, make sure to announce in the forums such that we can put it on the official C::B homepage.

- Download the source code

If you feel comfortable building applications from source, then this is the recommended way to download Code::Blocks. Downloading the source code and building it yourself puts you in great control and also makes it easier for you to update to newer versions or, even better, create patches for bugs you may find and contributing them back to the community so everyone benefits.

- Retrieve source code from SVN

This option is the most flexible of all but requires a little bit more work to setup. It gives you that much more flexibility though because you get access to any bug-fixing we do at the time we do it. No need to





The IDE with all the features you need, having a consistent look, feel and operation across platforms.

News

## Features

## Downloads

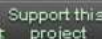
User manual

## Forums

Wiki

## License

## Donations



<https://www.fosshub.com/Code-Blocks.html?dwl=codeblocks-20.03mingw-setup.exe>

## Binary releases

- Windows XP / Vista / 7 / 8.x / 10
- Linux 32 and 64-bit
- Mac OS X

**NOTE:** For older OSes use older releases. There are releases for many OS version and platforms on the [Sourceforge.net](#) page.

**NOTE:** There are also more recent nightly builds available in the [forums](#) or (for Ubuntu users) in the [Ubuntu PPA repository](#). Please note that we consider nightly builds to be stable, usually.

**NOTE:** We have a [Changelog for 20.03](#), that gives you an overview over the enhancements and fixes we have put in the new release.

**NOTE:** The default builds are 64 bit (starting with release 20.03). We also provide 32bit builds for convenience.



File

codeblocks-20.03-setup.exe

codeblocks-20.03-setup-nonadmin.exe

codeblocks-20.03-nosetup.zip

codeblocks-20.03mingw-setup.exe

codeblocks-20.03mingw-nosetup.zip

Download from

FossHUB or Sourceforge.net

FossHUB or Sourceforge.net

FossHUB or Sourceforge.net

FossHUB or Sourceforge.net

FossHUB or Sourceforge.net

**Code::Blocks est  
téléchargé avec le  
compilateur  
intégré MinGW**



- CATEGORIES
- ANTI MALWARE
- AUDIO EDITORS
- AUDIO PLAYERS
- BACKUP TOOLS
- BROWSERS
- BURNING TOOLS
- CHAT CLIENTS
- CODEC PACKS
- DATABASE
- DEVELOPER TOOLS
- DISK ANALYSERS
- EBOOK APPS

DEVELOPER TOOLS

# Code Blocks

DEVELOPER: <http://www.codeblocks.org>

★★★★★ 5 / 11

Donate to remove ad

Donate to remove ad

DOWNLOAD

Code Blocks Wind

Code Blocks Wind

Enregistrer sous

« Cours Algorithmique et programmation en langage C » Algo 1 » 2020-2021

Rechercher dans : 2020-2021

Organiser Nouveau dossier

Nom	Modifié le	Type	Taille
Aucun élément ne correspond à votre recherche.			

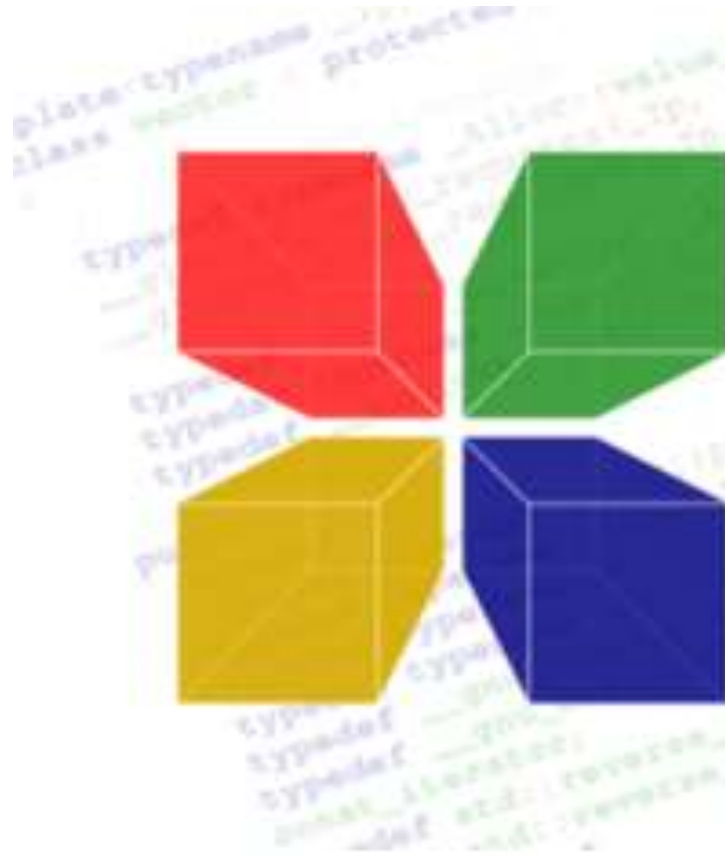
Nom du fichier : codeblocks-20.03mingw-setup.exe

Type : Application (\*.exe)

Masquer les dossiers

Enregistrer Annuler



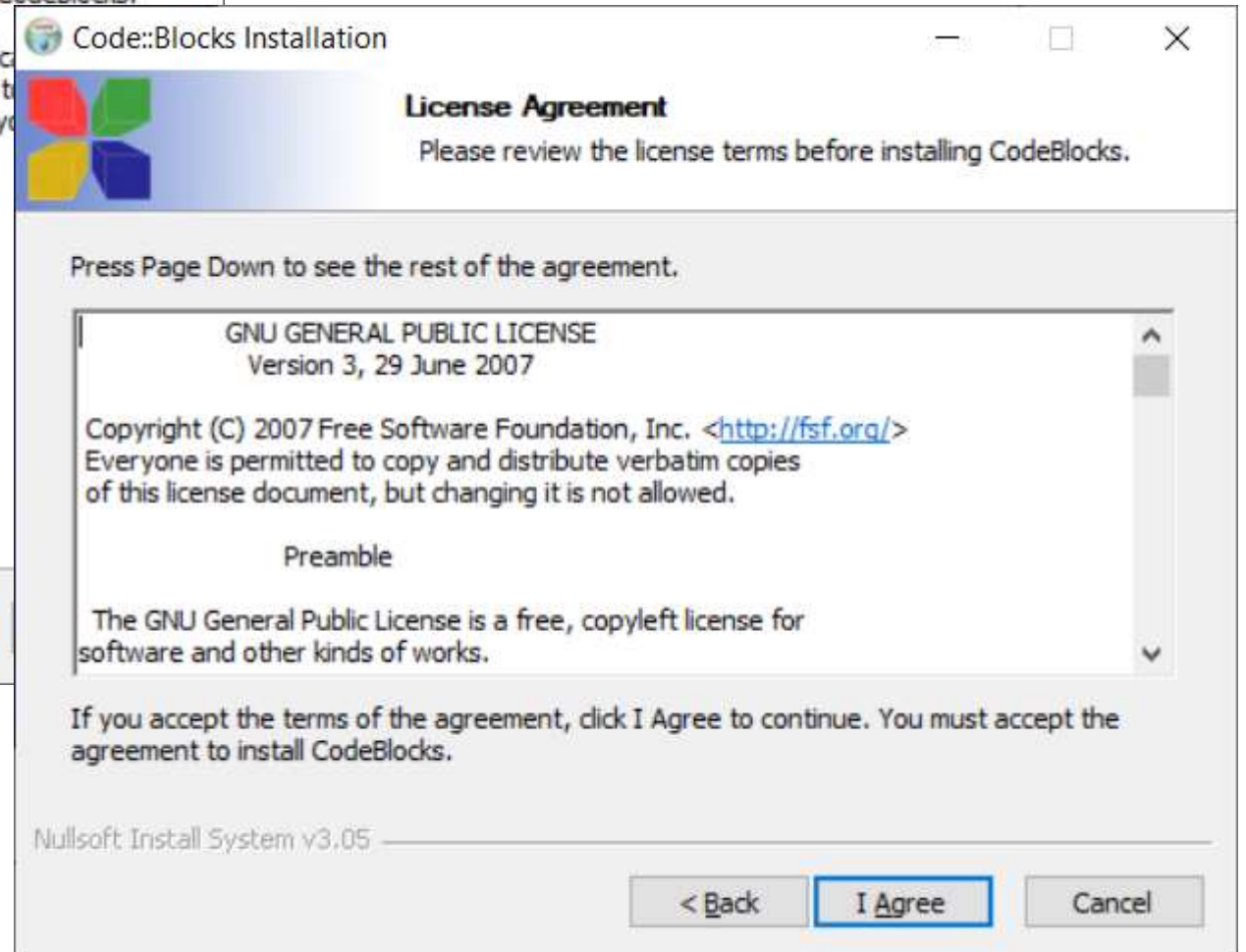
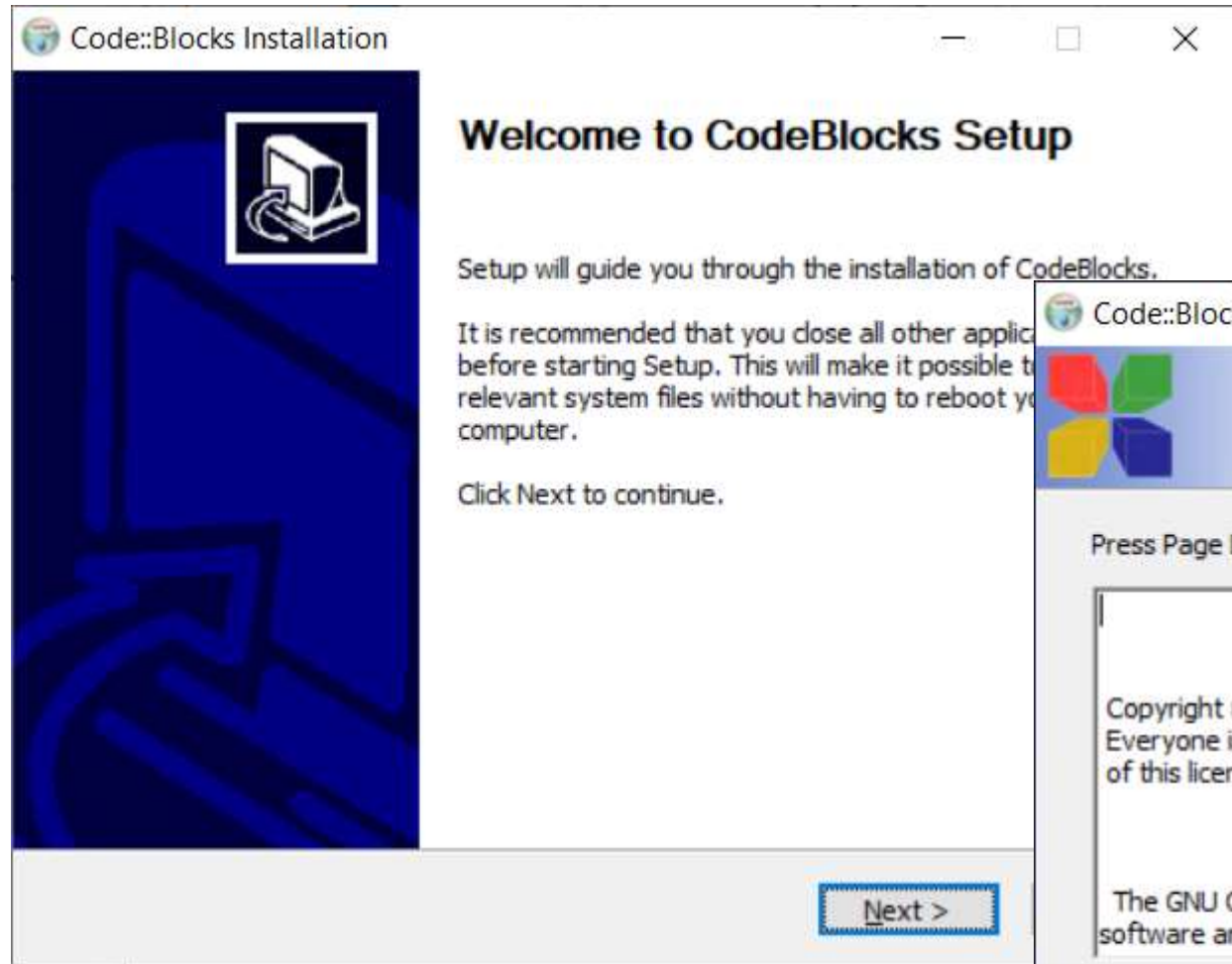


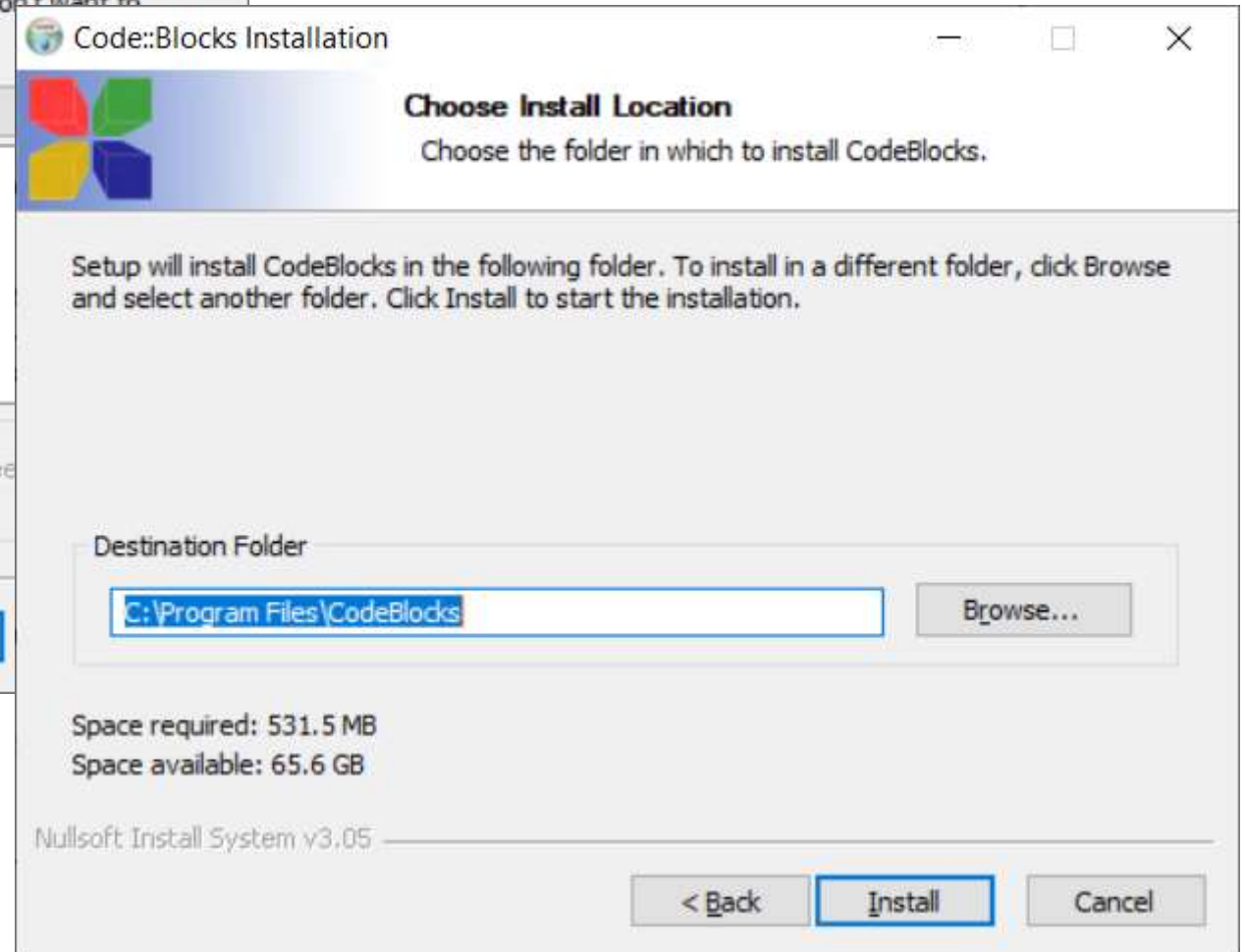
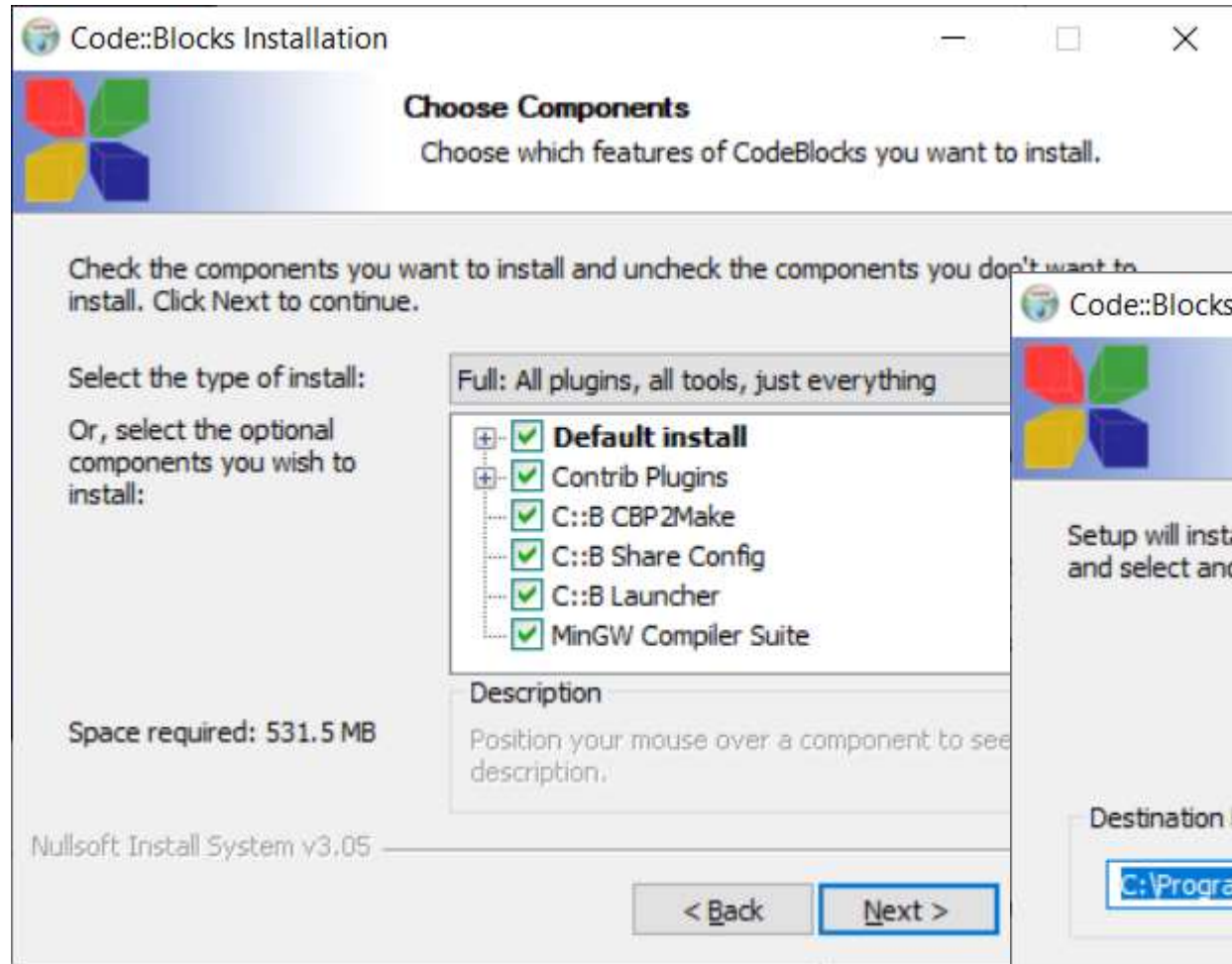
# Code::Blocks

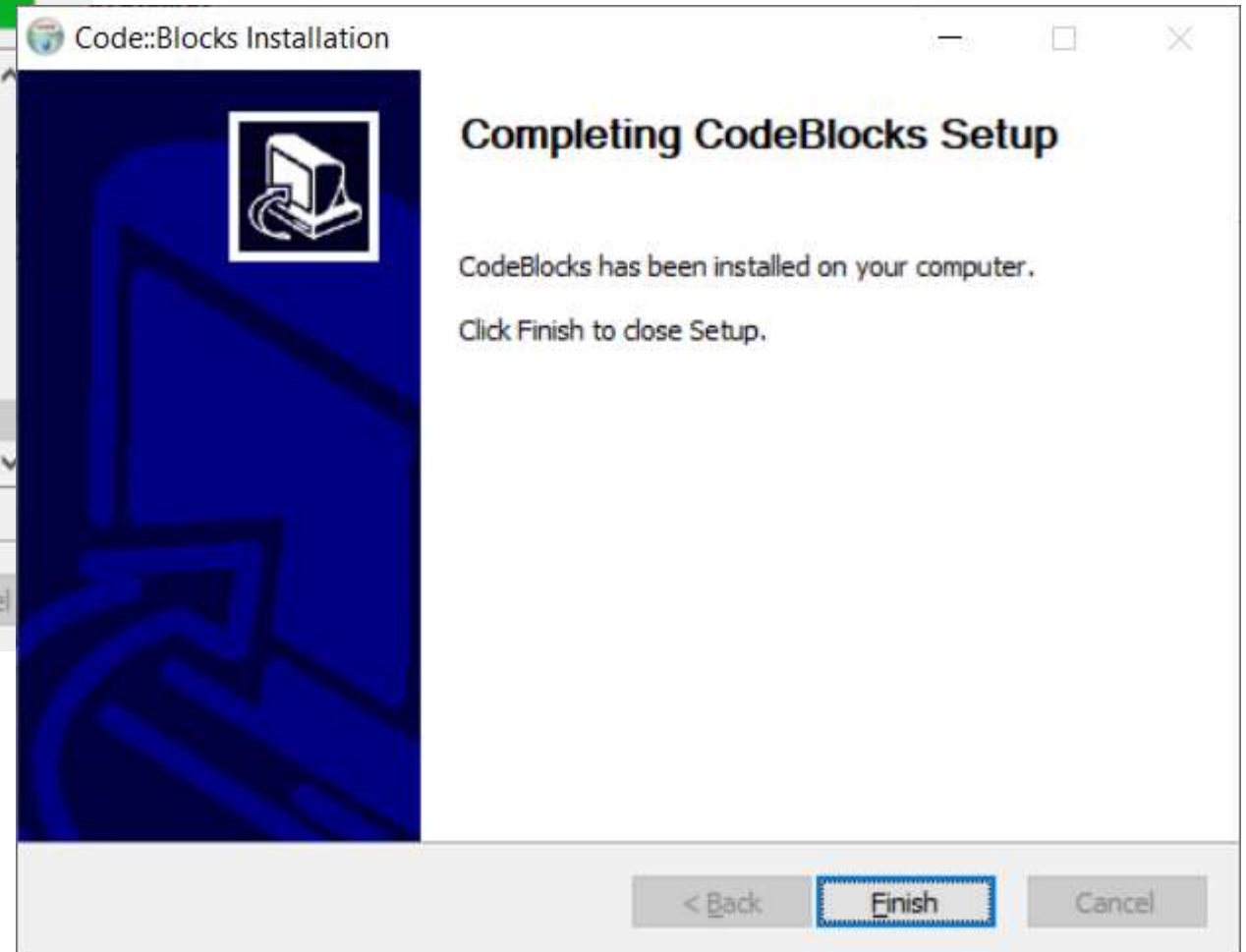
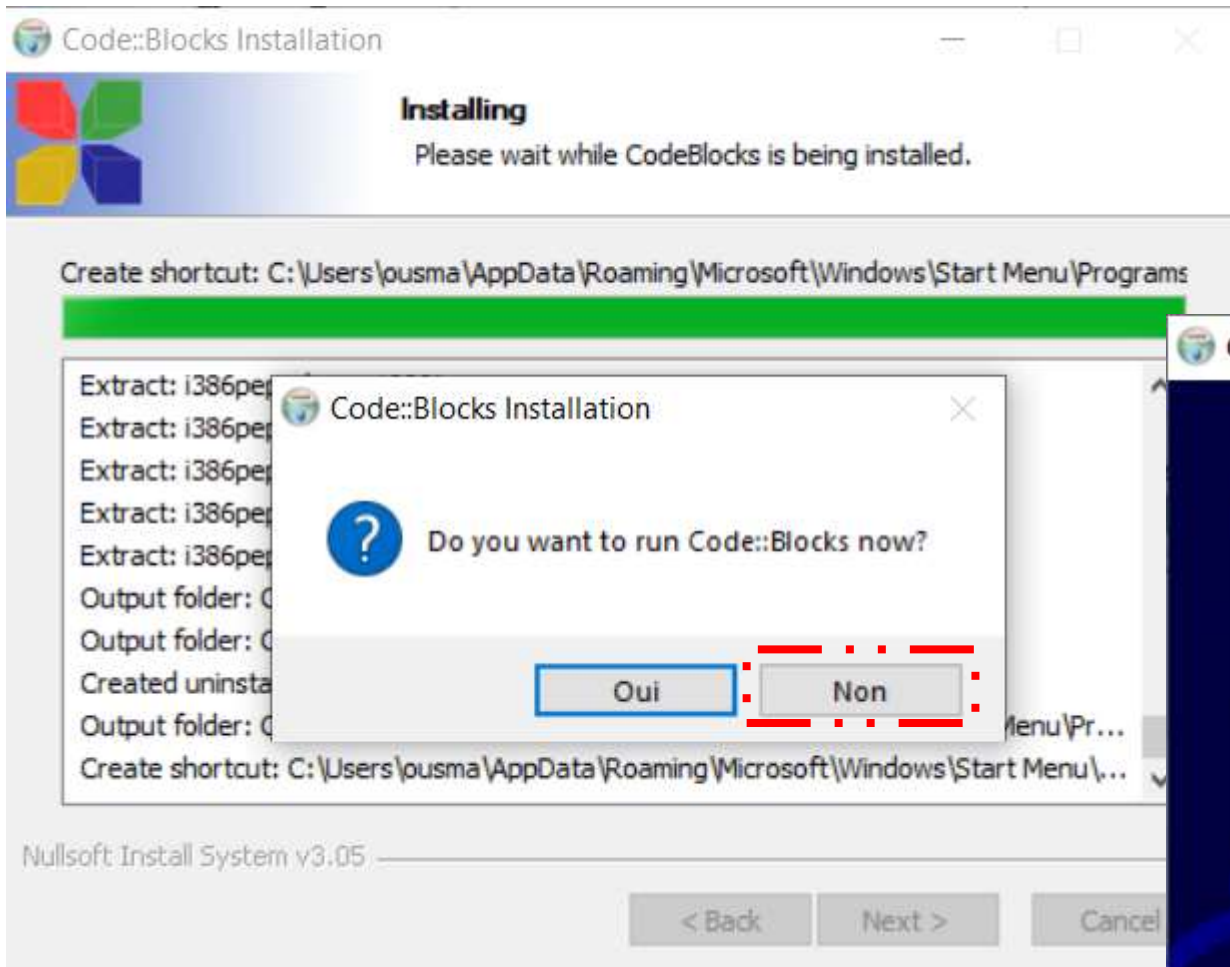
The open source, cross-platform IDE

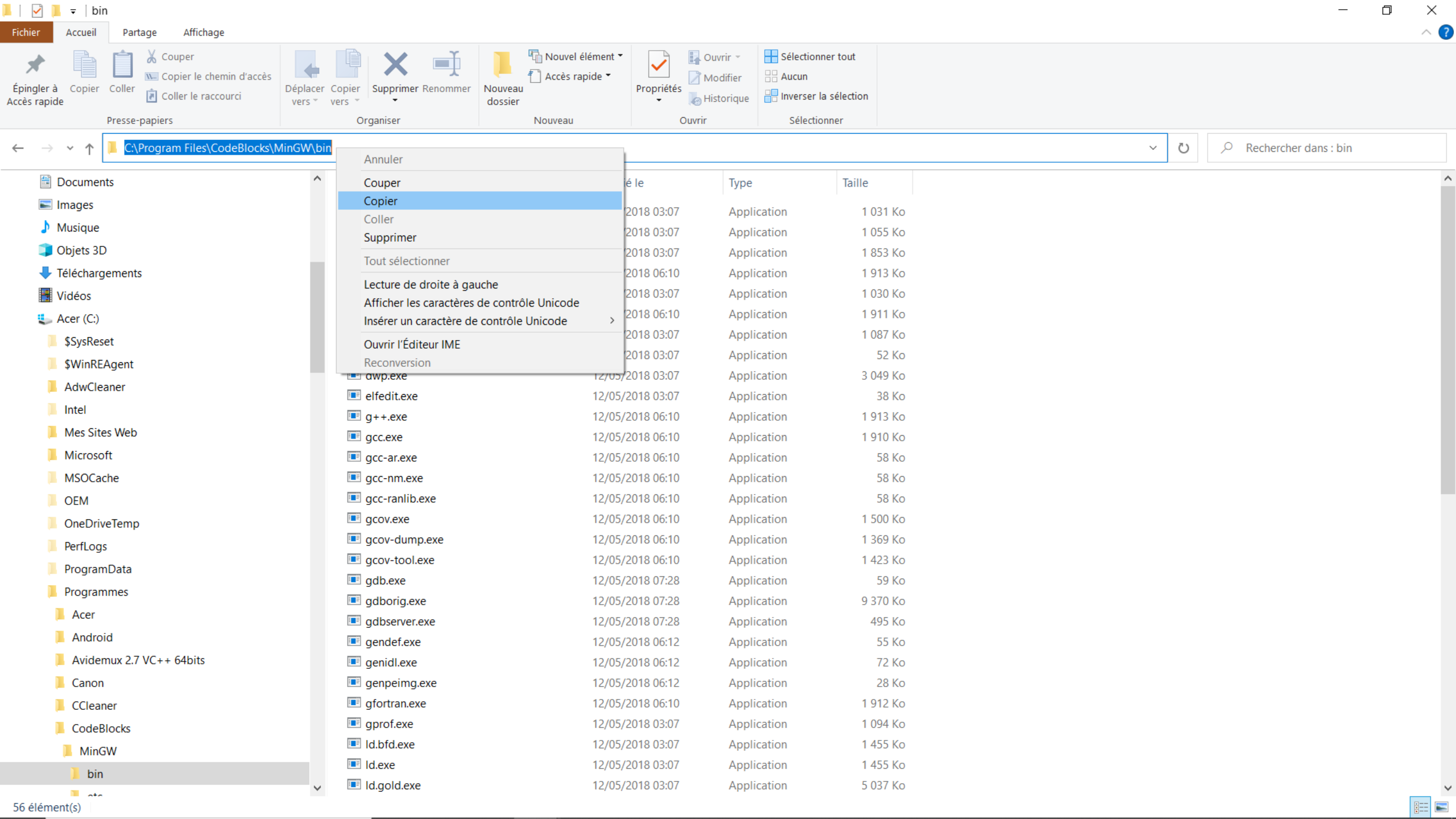
**20.03**



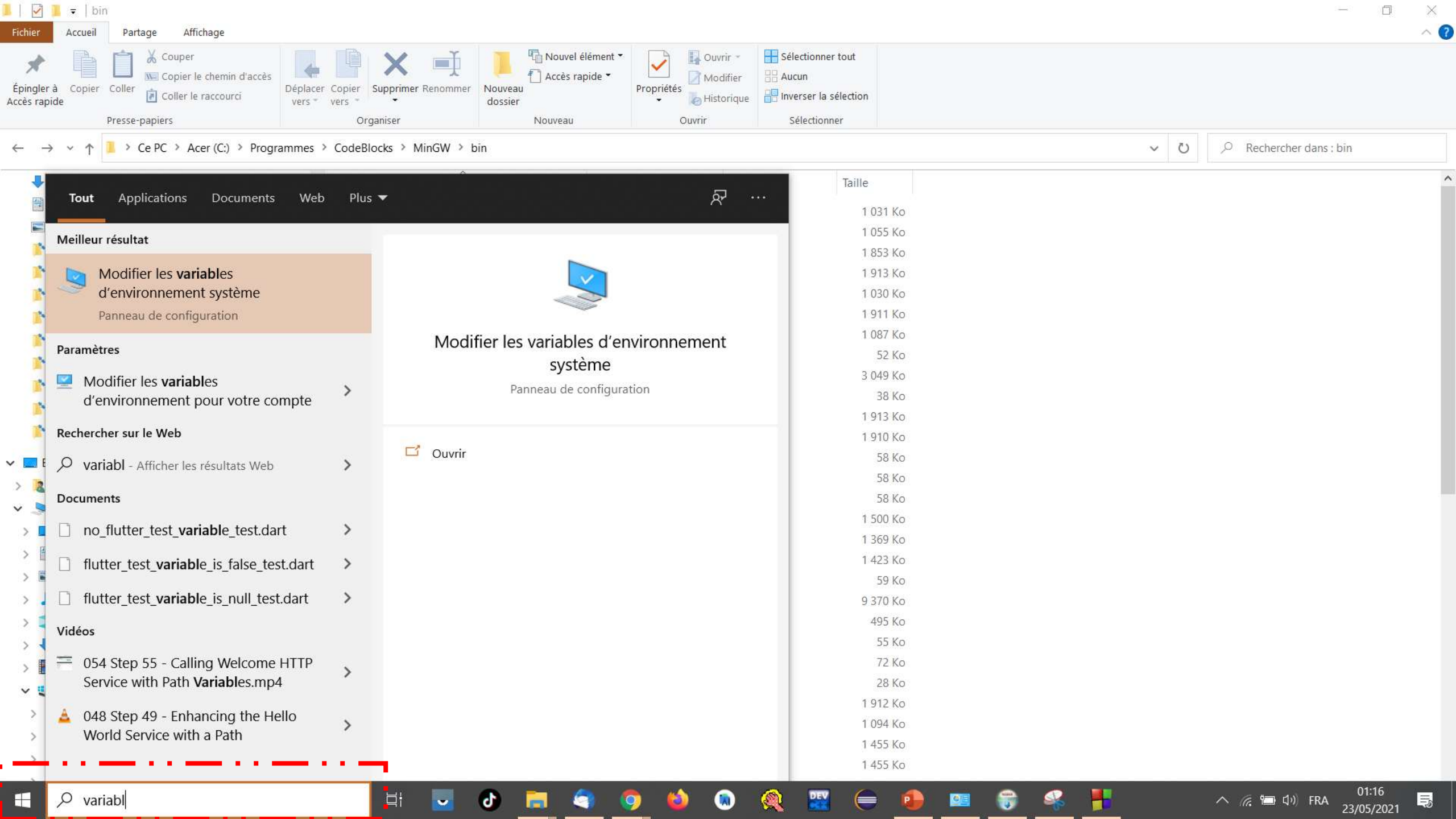












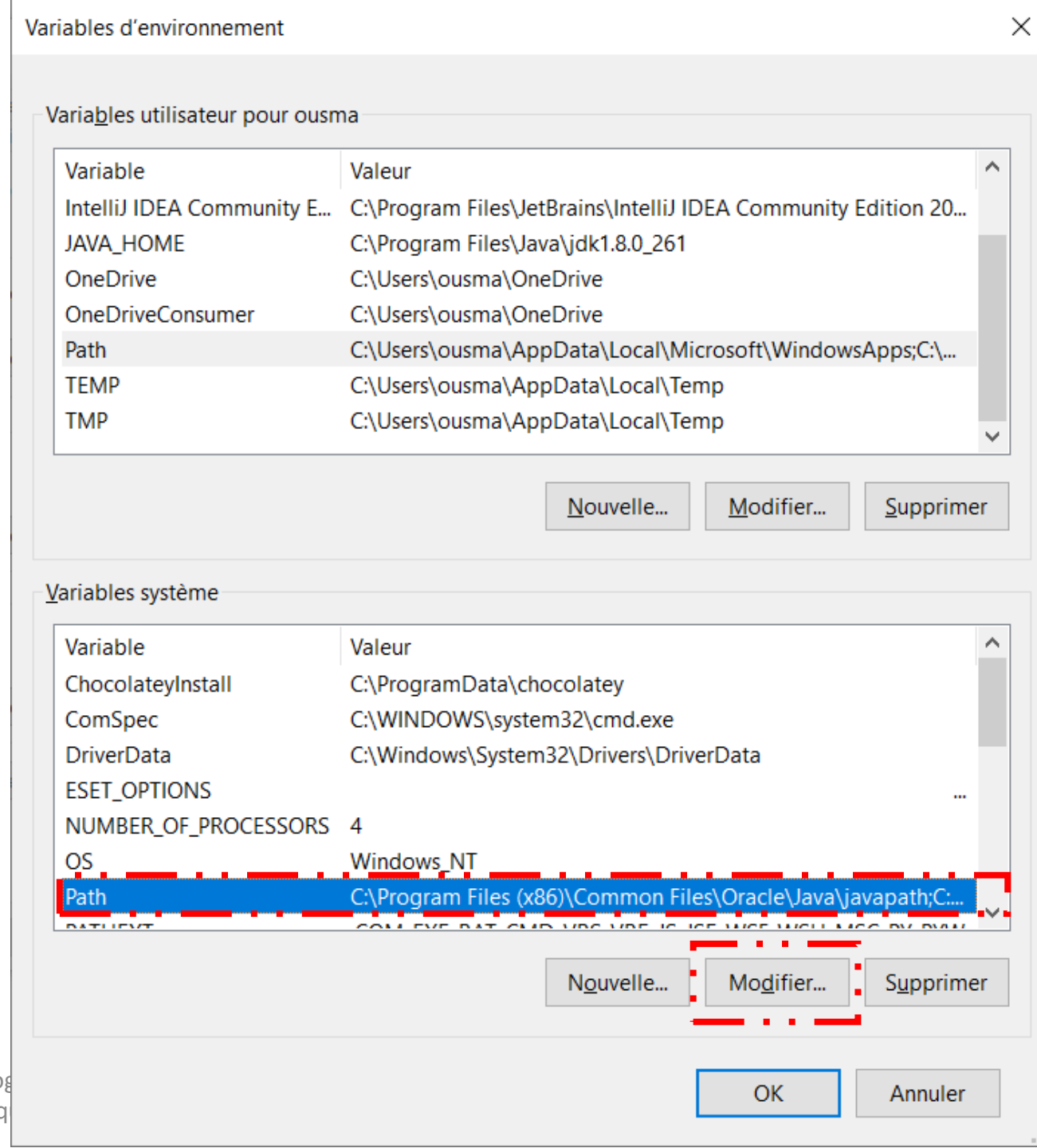
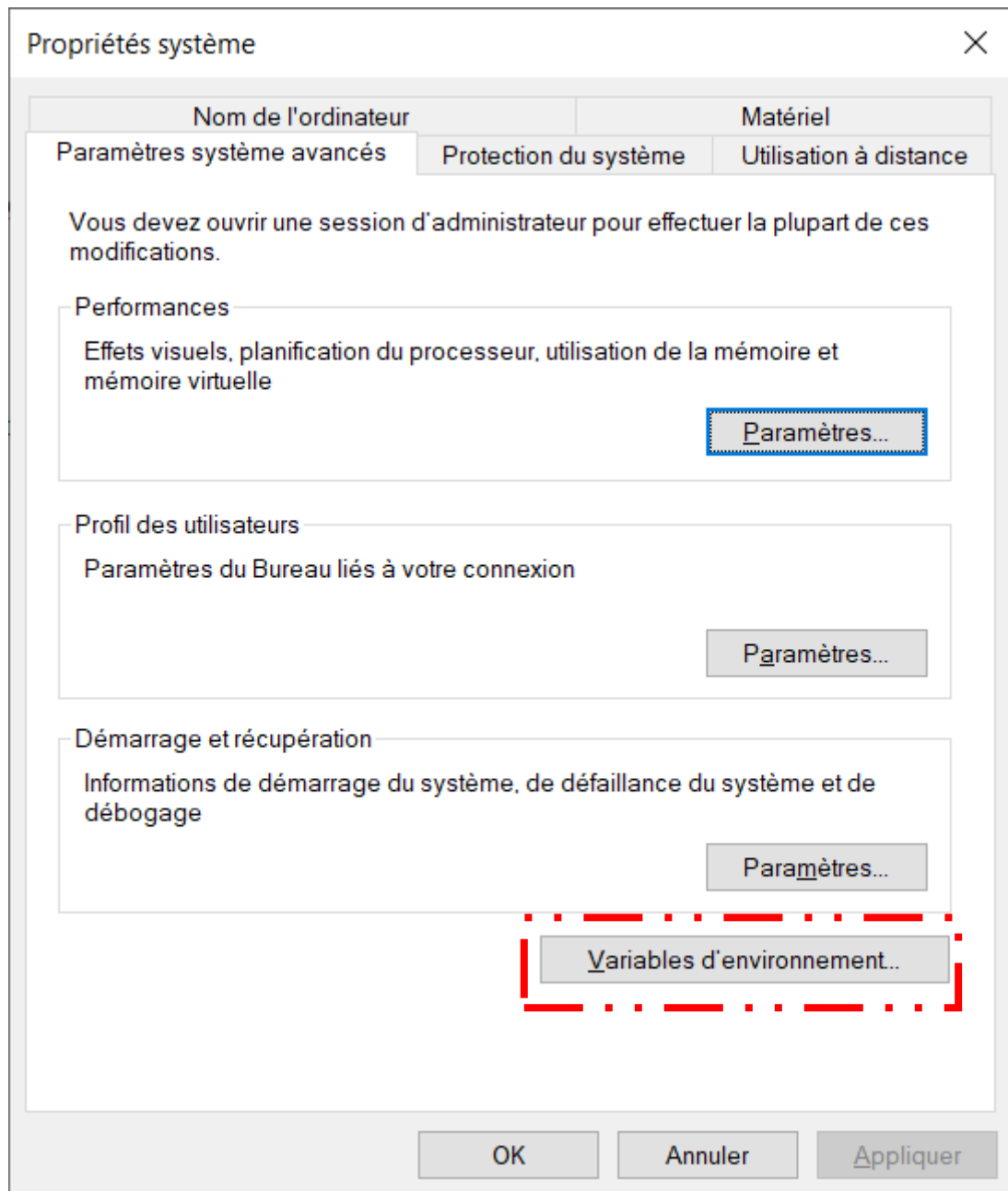




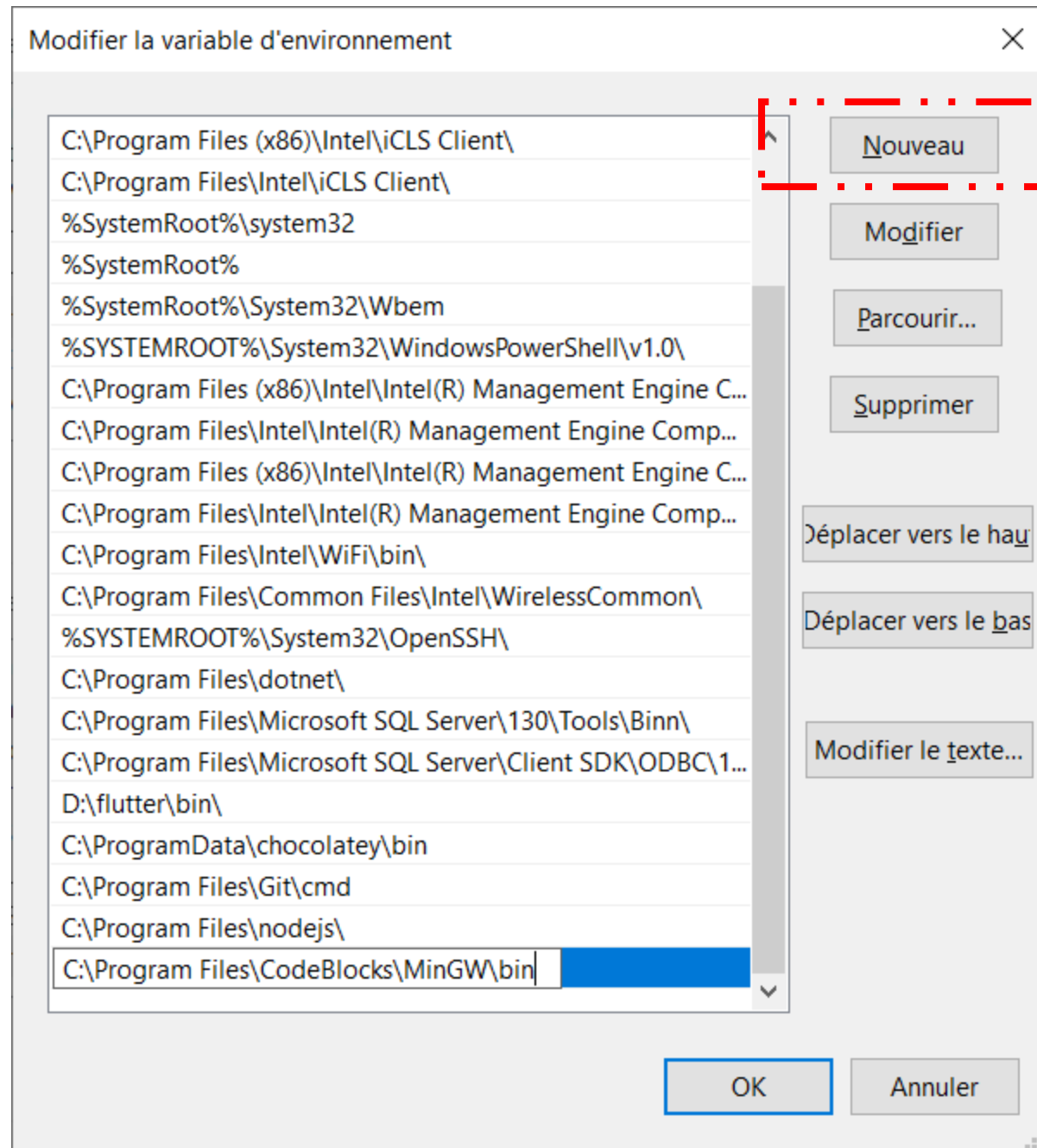


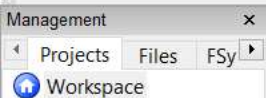


Tout	Applications	Documents	Web	Plus
<b>Meilleur résultat</b>				
 <b>Modifier les variables d'environnement système</b> Panneau de configuration				
<b>Paramètres</b>				
 <b>Modifier les variables d'environnement pour votre compte</b> >				
<b>Rechercher sur le Web</b>				
 <b>variabl - Afficher les résultats Web</b> >				
<b>Documents</b>				
 <b>no_flutter_test_variable_test.dart</b> >				
 <b>flutter_test_variable_is_false_test.dart</b> >				
 <b>flutter_test_variable_is_null_test.dart</b> >				
<b>Vidéos</b>				
 <b>054 Step 55 - Calling Welcome HTTP Service with Path Variables.mp4</b> >				
 <b>048 Step 49 - Enhancing the Hello World Service with a Path</b> >				
<b>Taille</b>				
1 031 Ko				
1 055 Ko				
1 853 Ko				
1 913 Ko				
1 030 Ko				
1 911 Ko				
1 087 Ko				
52 Ko				
3 049 Ko				
38 Ko				
1 913 Ko				
1 910 Ko				
58 Ko				
58 Ko				
58 Ko				
1 500 Ko				
1 369 Ko				
1 423 Ko				
59 Ko				
9 370 Ko				
495 Ko				
55 Ko				
72 Ko				
28 Ko				
1 912 Ko				
1 094 Ko				
1 455 Ko				
1 455 Ko				










- Environment...
- Editor...
- Compiler...
- Debugger...
- Global variables...
- Scripting...
- Edit startup script






Start here




# Code::Blocks

The open source, cross-platform IDE

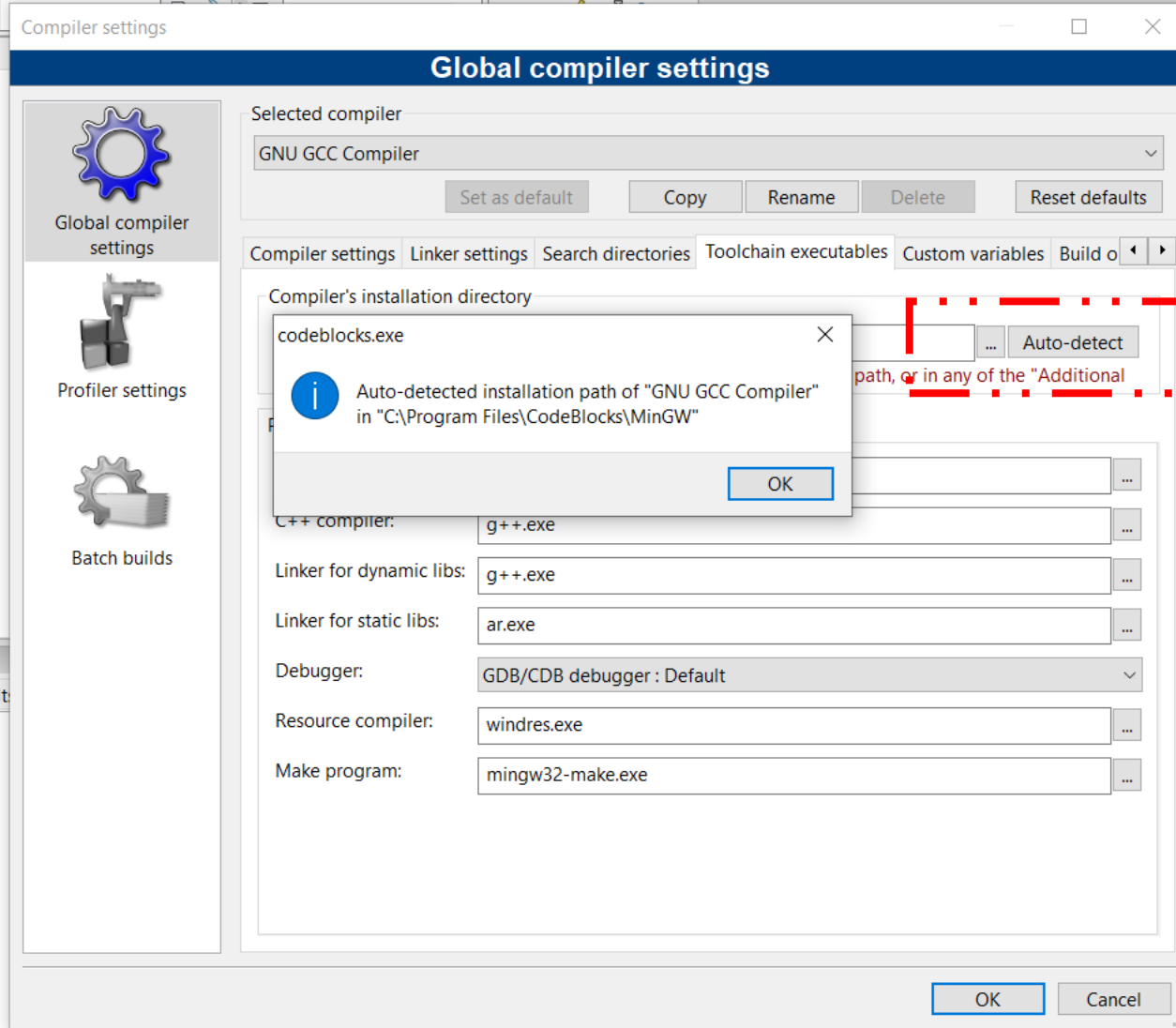
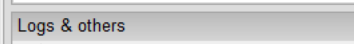
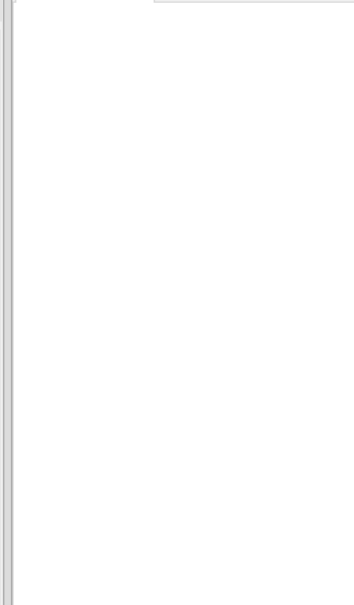
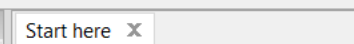
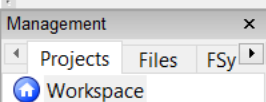
[Release 20.03 rev 11983 \(2020-03-12 18:24:30\) gcc 8.1.0 Windows/unicode - 64 bit](#)

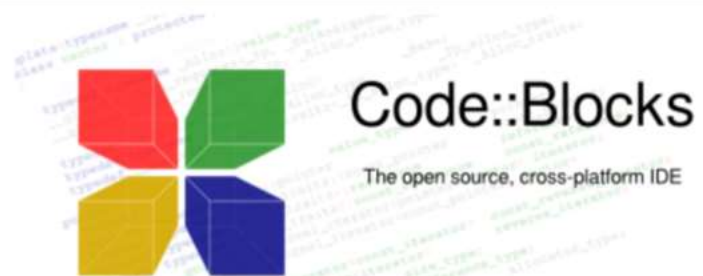
 [Create a new project](#)  [Open an existing project](#)  [Tip of the Day](#)

 [Visit the Code::Blocks forums](#) [Report a bug or request a new feature](#)

Logs & others

- Code::Blocks
- Search results
- Cccc
- Build log
- Build messages
- CppCheck/Vera++
- CppCheck/Vera++ messages
- Cscope
- Debugger
- DoxyBlocks
- Fortran info
- Close





[Release 20.03 rev 11983 \(2020-03-12 18:24:30\) gcc 8.1.0 Windows/unicode - 64 bit](#)



[Report a bug or request a new feature](#)

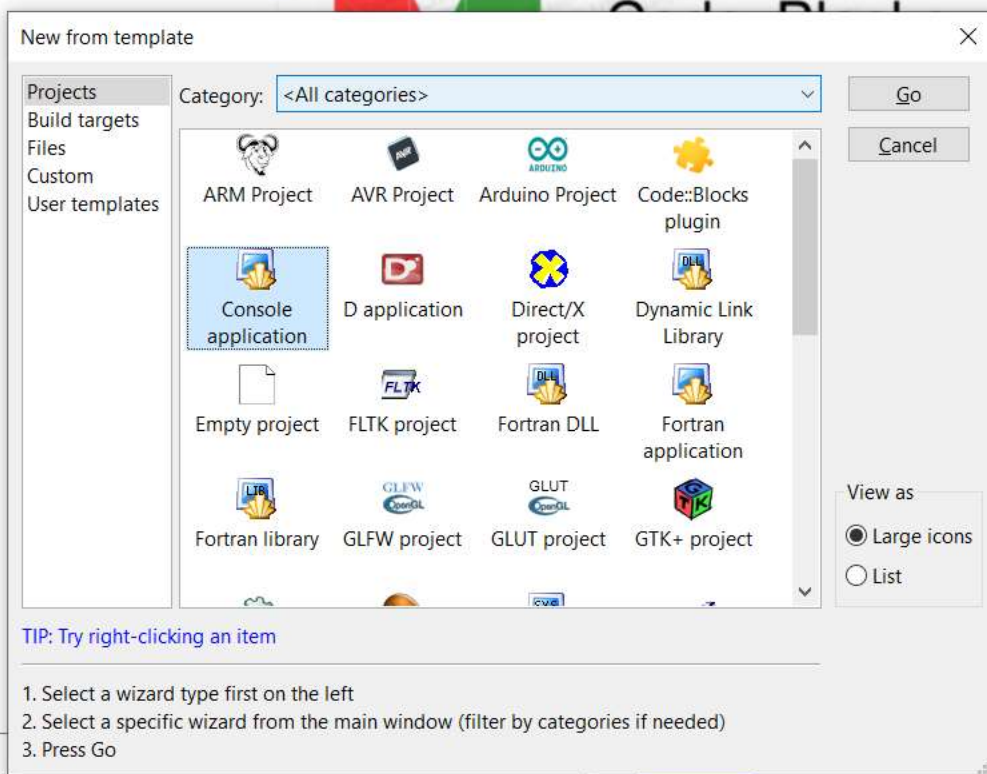
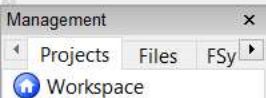
#### Recent projects

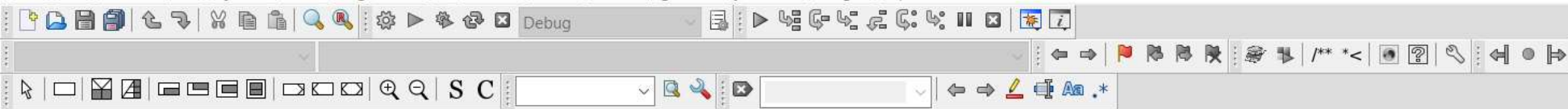
No recent projects

#### Recent files

No recent files

© 2004 - 2018, The [Code::Blocks](#) Team.





## Console application



Welcome to the new console application wizard!  
This wizard will guide you to create a new  
console application.

When you're ready to proceed, please click  
"Next"...

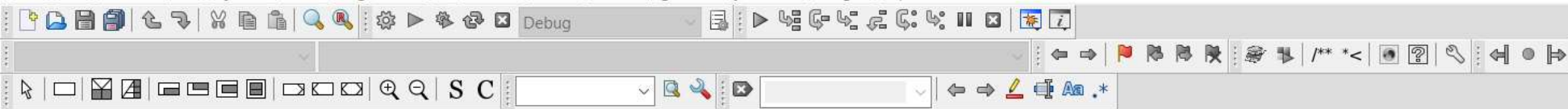
☐ Skip this page next time

&lt; Back

Next &gt;

Cancel





## Console application



Please select the language you want to use.

Please make a selection

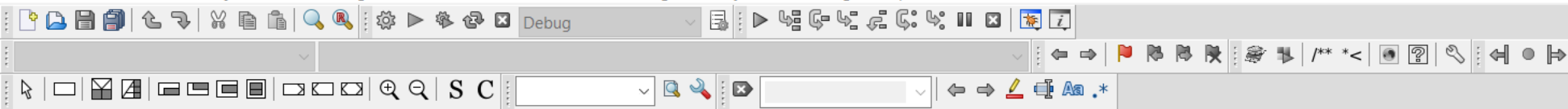
C

C++

&lt; Back

Next &gt;

Cancel



## Console application

**Console**

Please select the folder where you want the new project to be created as well as its title.

Project title:

Exemple1

Folder to create project in:

C:\Users\ousma\Desktop\

Project filename:

Exemple1.cbp

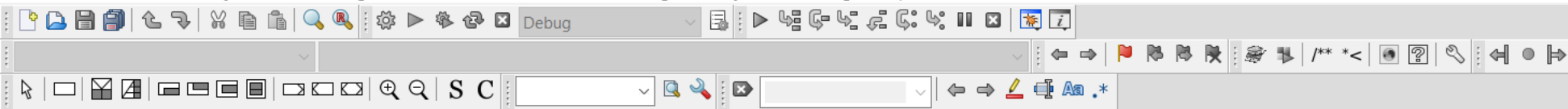
Resulting filename:

C:\Users\ousma\Desktop\Exemple1\Exemple1.cbp

&lt; Back

Next &gt;

Cancel



## Console application

**Console**

Please select the compiler to use and which configurations you want enabled in your project.

Compiler:

GNU GCC Compiler

☒ Create "Debug" configuration:

Debug

"Debug" options

Output dir.: bin\Debug\

Objects output dir.: obj\Debug\

☒ Create "Release" configuration:

Release

"Release" options

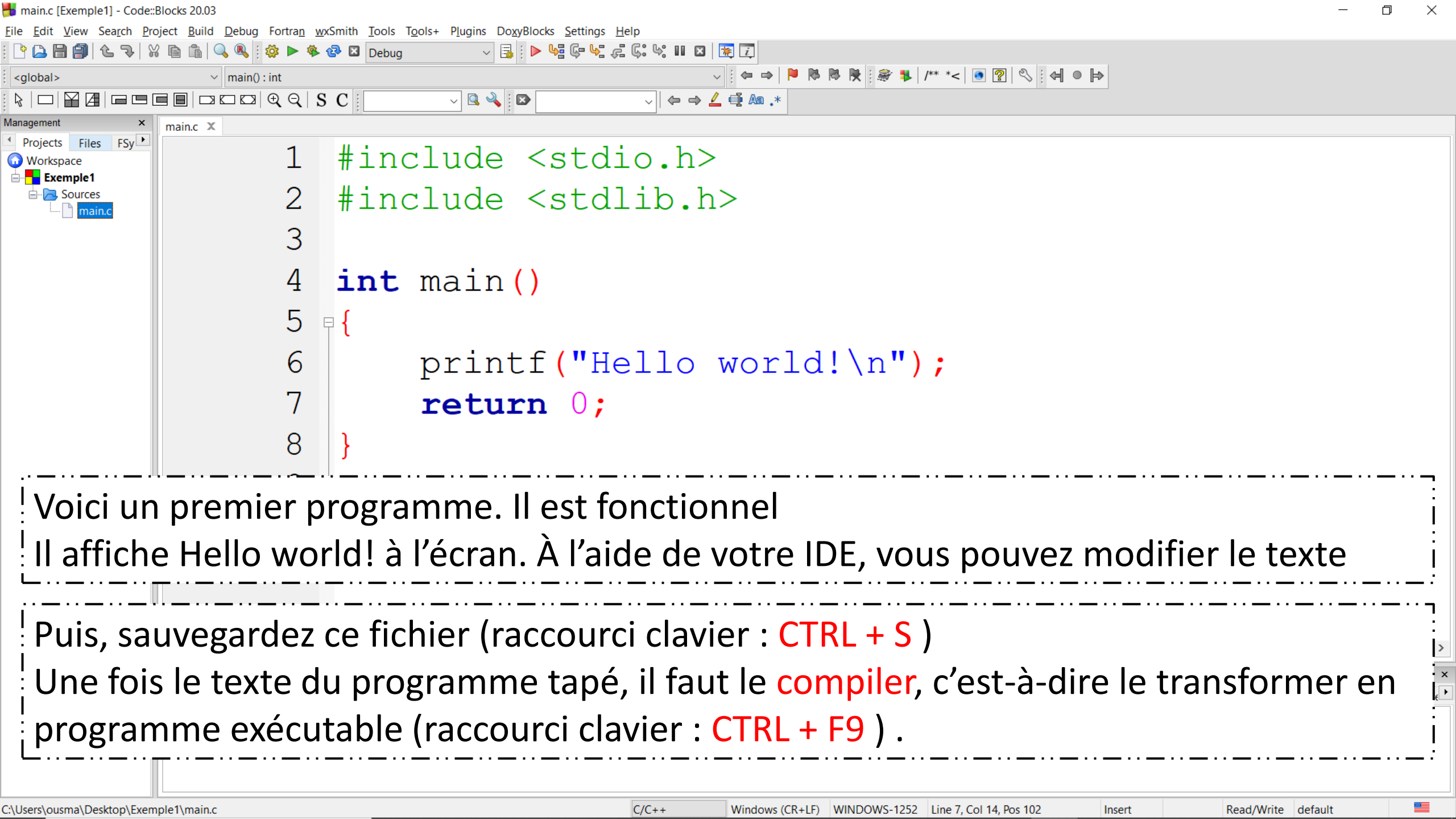
Output dir.: bin\Release\

Objects output dir.: obj\Release\

&lt; Back

Finish

Cancel



Voici un premier programme. Il est fonctionnel

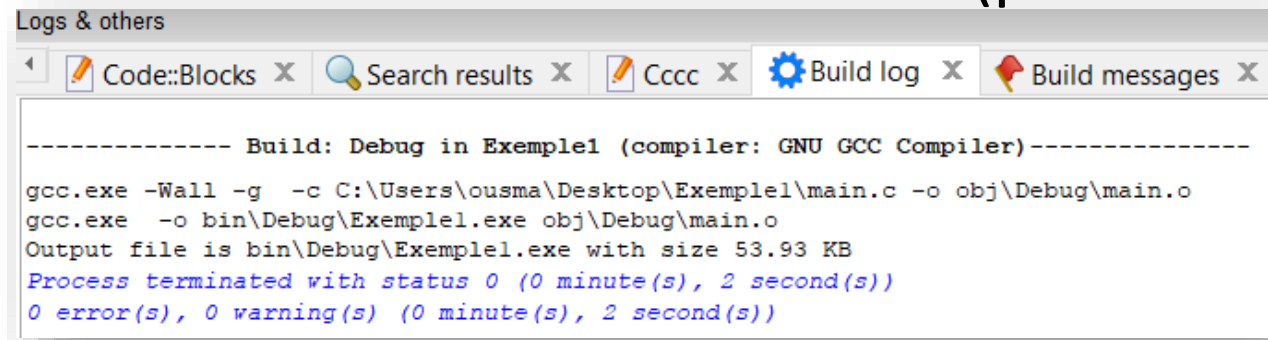
Il affiche Hello world! à l'écran. À l'aide de votre IDE, vous pouvez modifier le texte

Puis, sauvegardez ce fichier (raccourci clavier : **CTRL + S** )

Une fois le texte du programme tapé, il faut le **compiler**, c'est-à-dire le transformer en programme exécutable (raccourci clavier : **CTRL + F9** ) .

# Exemple de programme

- Si vous n'avez pas fait d'erreurs, la ligne précédente provoquera l'affichage d'une fenêtre semblable à celle-ci (pas de nouvelle, bonne nouvelle. . . )

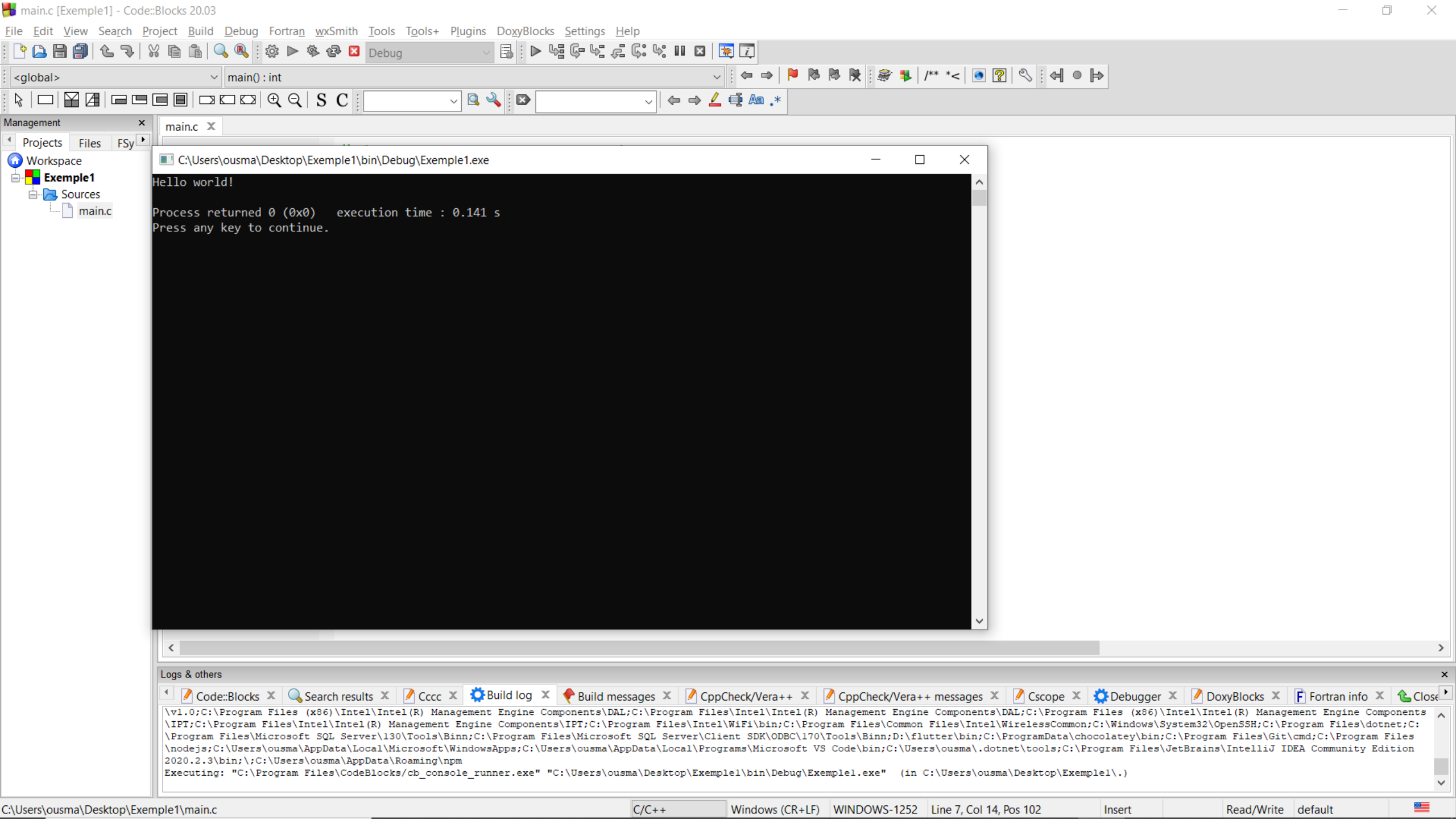


```
----- Build: Debug in Exemple1 (compiler: GNU GCC Compiler)-----
gcc.exe -Wall -g -c C:\Users\ousma\Desktop\Exemple1\main.c -o obj\Debug\main.o
gcc.exe -o bin\Debug\Exemple1.exe obj\Debug\main.o
Output file is bin\Debug\Exemple1.exe with size 53.93 KB
Process terminated with status 0 (0 minute(s), 2 second(s))
0 error(s), 0 warning(s) (0 minute(s), 2 second(s))
```

Le fichier **main.c** est un « **fichier source** ». Un fichier source désigne un fichier qu'un être humain peut comprendre par opposition à un exécutable que seule la machine arrive à comprendre. Il ne reste plus qu'à compiler et exécuter le programme pour cela appuyer sur la touche **F9**.

La machine affichera alors Bonjour et attendra que vous appuyiez sur la touche « entrée ».

Remarquez l'apparition d'un nouveau fichier **main.exe** qui est un « fichier exécutable ».





# Structure générale d'un programme en C

# Fichier C (extension .c)

```
*main.c x
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main()
5  {
6      printf("Hello world!\n");
7      return 0;
8  }
```

En C le programme principal s'appelle toujours **main**

Plusieurs **instructions** peuvent être regroupées par un **bloc**. Les **blocs d'instructions** sont définis par des accolades **{** marque le début d'un bloc et **}** marque la fin d'un bloc.

# Directives au préprocesseur –

## #include <fichier en-tête bibliothèque>

0

- Le langage C n'est qu'un nombre restreint d'instructions et un ensemble de **bibliothèques**. Le compilateur y trouve les fonctions et les applications qui lui permettent de créer un programme exécutable.
- Certaines bibliothèques (les plus courantes) sont incluses par défaut, ce qui permet à notre programme de se compiler. La fonction **printf** est stockée dans la bibliothèque standard d'entrées-sorties, incluse par défaut permet d'afficher sur l'écran.
- L'utilisation d'une bibliothèque nécessite que nous informions le compilateur de notre souhait de l'utiliser : il suffit d'ajouter **#include <fichier en-tête bibliothèque>** en début de programme.

# Directives au préprocesseur – La fonction `main`

- Ainsi, puisque nous utilisons la fonction `printf`, qui est dans la librairie standard d'entrées/sorties, nous indiquerons en début de programme: `#include <stdio.h>` ③
- Un autre point à corriger est l'ajout de la ligne `return 0`. Tout programme doit renvoyer une valeur de retour, tout à la fin. Cette valeur de retour permet de savoir si le programme que l'on exécute s'est correctement terminé. ④
- En général 0 signifie une terminaison sans erreur. Enfin, il faut transformer la ligne `main ()` en `int main()`. Ce point sera détaillé par la suite lorsque nous parlerons des fonctions...

# Petit mot sur ce qu'est une bibliothèque

- À l'instar de l'étudiant qui recherche dans des livres, nous pouvons dire que le « .h » représente l'index du livre et le « .c » le contenu du chapitre concerné.
- Après avoir lu (et retenu) le contenu des fichiers .h inclus, si le compilateur rencontre l'appel à la fonction *printf*, il est en mesure de vérifier si la fonction figurait dans un des *include*. Si ce n'est pas le cas, il émettra un avertissement.

# Squelette de programme

- On peut définir le squelette d'un programme C de la façon suivante :

```
1  /* Déclaration des fichiers d'en-têtes de bibliothèques */
2  #include<stdio.h>
3  int main () {
4      /* Déclaration des variables (cf. chapitres suivants...) */
5      /* Corps du programme */
6      getchar(); /* Facultatif mais permet d'attendre l'appui d'une touche */
7      return 0; /* Aucune erreur renvoyée */
8  }
```



# Blocs 2 5

- La partie de programme située entre deux accolades est appelée un **bloc**. On conseille de prendre l'habitude de faire une tabulation après l'accolade. Puis retirer cette tabulation au niveau de l'accolade fermante du bloc.
- Ainsi, on obtient :

```
1 int main () {  
2     Tabulation  
3     Tout le code est frappé à cette hauteur  
4 }  
5 Retrait de la tabulation  
6 Tout le texte est maintenant frappé à cette hauteur.
```

# Commentaires

- Bien commenter un programme signifie qu'une personne ne connaissant pas votre code doit pouvoir le lire et le comprendre. Les commentaires sont indispensables dans tout bon programme. Ils peuvent être placés à n'importe quel endroit. Ils commencent par `/*` et se terminent par `*/` :

`/* Commentaire */`

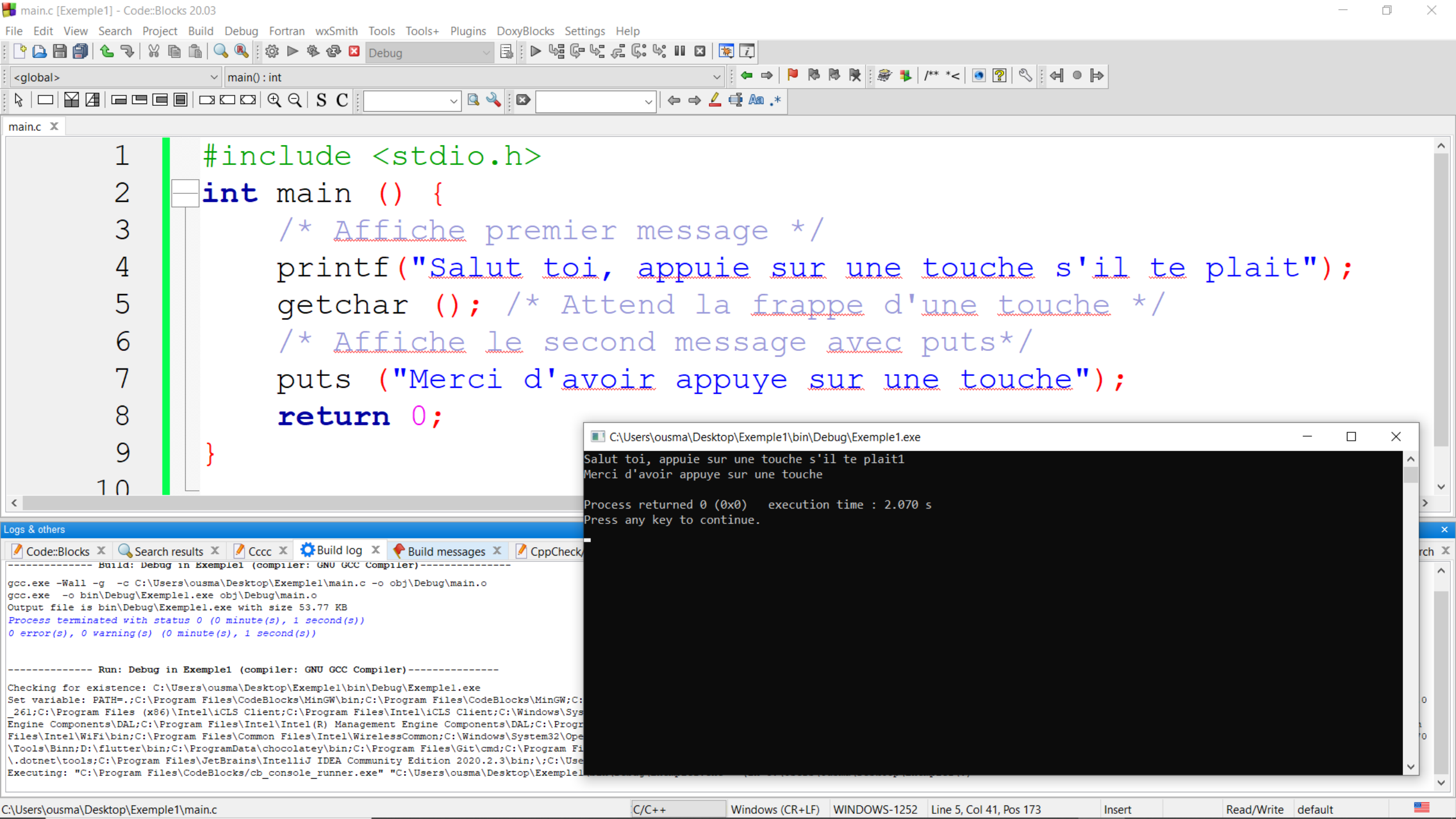
- Comme nous l'avons déjà mentionné, vous trouverez aussi parfois des commentaires C++ :

`//` Le reste de la ligne est un commentaire

# Exercice d'application



- Écrivez algorithme puis un programme qui :
  - affiche « Salut toi, appuie sur une touche s'il te plaît » ;
  - attend l'appui d'une touche ;
  - affiche : « Merci d'avoir appuyé sur une touche ».
- Une fois que vous serez satisfait de votre solution, vous pourrez la comparer avec la solution qui apparaît au slide suivant.



# Concepts de base d'un programme en C

# Langage C: Identificateurs et mots-clés

- **Identificateur** : nom donné par le programmeur à un élément de programmation : fonction, variable, constante symbolique, type, etc.
- La règle principale pour un identificateur, est qu'il doit être unique, dans une certaine mesure qui dépend de l'élément concerné : variable, constante, fonction, etc.
- Pour une variable, l'identificateur doit être unique dans le bloc qui la contient et dans tous ses sous-blocs.
- Le nom d'une variable peut contenir jusqu'à 256 caractères.



# Langage C: Identificateurs et mots-clés

- Un identificateur doit aussi respecter un certain nombre de contraintes pour être valide.
- Pour simplifier, on utilisera les règles suivantes dans le cadre de ce cours :
  - On utilisera seulement des lettres (minuscules et majuscules), des chiffres et le caractère tiret-bas (aussi appelé underscore : `_`) ;
  - On n'utilisera pas de signes diacritiques sur les lettres (accents, trémas, tilde, etc.)
  - Le premier caractère de l'identificateur sera toujours une lettre.
- Remarque : le compilateur C fait la distinction entre lettres minuscules et majuscules. Par exemple, les identificateurs **var** et **VAR** sont considérés comme différents
- **Une dernière restriction concernant les noms des variables: il est interdit d'utiliser un des 32 mots réservés du langage C**

# Mots réservés du langage C

<b>auto</b>	<b>double</b>	<b>int</b>	<b>struct</b>
<b>break</b>	<b>else</b>	<b>long</b>	<b>switch</b>
<b>case</b>	<b>enum</b>	<b>register</b>	<b>typedef</b>
<b>char</b>	<b>extern</b>	<b>return</b>	<b>union</b>
<b>const</b>	<b>float</b>	<b>short</b>	<b>unsigned</b>
<b>continue</b>	<b>for</b>	<b>signed</b>	<b>void</b>
<b>default</b>	<b>goto</b>	<b>sizeof</b>	<b>volatile</b>
<b>do</b>	<b>if</b>	<b>static</b>	<b>while</b>

# Langage C: Types scalaires prédéfinis

Nom	Taille	Constante	Min	Max
<b>char</b>	1 octet	'a', 97	-128 (SCHAR_MIN)	+127 (SCHAR_MAX)
unsigned char	1 octet	'a', 97	0	255 (UCHAR_MAX)
short	2 octets	-125, 312	-32768 (SHRT_MIN)	32767 (SHRT_MAX)
unsigned short	2 octets	273U	0	65535 (USHRT_MAX)
<b>int</b>	4 octets (parfois 2 ou 8)	-3000, 3213	-2 milliards env. (INT_MIN)	+2 milliards env. (INT_MAX)
<b>unsigned int</b>	4 octets (parfois 2 ou 8)	3124U	0	4 milliards env. (UINT_MAX)
long	4 octets (parfois 8 ou 16)	3167L	-2 milliards env. (INT_MIN)	+2 milliards env. (INT_MAX)
unsigned long	4 octets (parfois 8 ou 16)	243UL	0	4 milliards env. (UINT_MAX)
float	4 octets	21.3F, 3.12e4F	-3.4e38 env. (-FLT_MAX) 1.17e-38 (FLT_MIN)	+3.4e38 env. (FLT_MAX)
<b>double</b>	8 octets	12.4, 125.2e5	-1.79e308 env. (-DBL_MAX) 2.22e-308 env. (DBL_MIN)	1.79e308 (DBL_MAX)

# Langage C: Autres types introduits par la norme C99

- Outre les nouveaux types entiers dont nous avons parlé, la **norme C99** introduit :
  - le type booléen, sous le nom **bool** ; une variable de ce type ne peut prendre que l'une des deux valeurs : vrai (noté true) et faux (noté false) ;
  - des types complexes, sous les noms **float complex**, **double complex** et **long double complex** ; la constante **I** correspond alors à la constante mathématique  $i$  (racine de  $-1$ ).

# Les expressions

- Correspondent à une combinaison d'éléments tel que des :
  - **identificateurs**
  - **constantes**
  - **variables**
  - tableaux
  - pointeurs
  - structures
  - unions
  - **appels de fonctions**
  - **opérateurs unaires ou binaires**
  - ...
  - chaque ligne de code excluant celles ayant uniquement des commentaires

# Déclaration des variables

- La déclaration des variables se fait selon le modèle suivant:

**type** **variable1**, **variable2**, ... ;

- Remarquez la **virgule** séparant deux variables ainsi que le **point-virgule** à la fin.
- Exemples:

<b>int nbre;</b>	définit nbre comme une variable entière
<b>int a,b,c;</b>	définit trois variables entières dont les noms respectifs sont a, b et c.
<b>double x;</b>	définit x comme une variable réelle.

# Langage C: l'affectation

- L'instruction **variable ← expression** se traduit en C par  
**variable = expression ;**
- **Remarque 1:** Observez le caractère ' ; ' qui termine toujours une instruction en C.
- **Remarque 2:** Le symbole **=** n'est pas un opérateur de comparaison en C. Il s'agit d'un opérateur d'affectation.  
Nous reviendrons sur ce sujet dans un prochain cours.



.....

à ne pas mettre de "ll"

.....

Je dirais que la cour

.....

Downloaded from <http://ajphaphysocpharm.sagepub.com/> at 11:01 11 November 2014

Code::Blocks Search results Cccc Build Log Build messages CppCheck/Vera++ CppCheck/Vera++ messages Cscope Debugger DoxyBlocks Fortran info Close

File	Line	Message
		=== Build: Debug in Exemple1 (compiler: GNU GCC Compiler) ===


```
C:\Users\ou... 7 warning: variable 'n' set but not used [-Wunused-but-set-variable]
C:\Users\ou... 7
C:\Users\ou... 7 warning: unused variable 'n' [-Wunused-variable]
```

```

C:\Users\ou... 5 warning: unused variable 'x' [-Wunused-variable]
===== Build failed: 1 error(s), 2 warning(s) (0 minute(s), 0 second(s)) =====

```

DATA: DATUM = DATUM(0); WERTUNG(0) = 0; ANZAHL(0) = 0; ZEICHEN(0);

sample1\main.c	C/C++	Windows (CR+LF)	WINDOWS-1252	Line 9, Col 1, Pos 140	Insert		Read/Write	default	
----------------	-------	-----------------	--------------	------------------------	--------	--	------------	---------	---

# Langage C: Les opérateurs

- **Opérateurs arithmétiques**

Symbole	Opération
+	addition
-	soustraction
*	multiplication
/	division entière
%	reste de la division entière (cette opération n'est pas valable sur des réels)
a=b	affectation (a reçoit la valeur de b)

# Langage C: Les opérateurs

- Opérateurs arithmétiques raccourcis :

Symbole	Opération
<code>--</code>	notation contractée : <code>a -= b</code> donne <code>a = a - b</code>
<code>+=</code>	notation contractée : <code>a += b</code> donne <code>a = a + b</code>
<code>*=</code>	notation contractée : <code>a *= b</code> donne <code>a = a * b</code>
<code>/=</code>	notation contractée : <code>a /= b</code> donne <code>a = a / b</code>
<code>--</code>	post décrémentation : <code>a=i--</code> . "a" reçoit la valeur de "i", puis "i" est décrémenté de 1. pré décrémentation : <code>a=--i</code> . "i" est décrémenté de 1 puis sa valeur est attribuée à "a"
<code>++</code>	post incrémentation : <code>a=i++</code> . "a" reçoit la valeur de "i", puis "i" est incrémenté de 1. pré incrémentation : <code>a=++i</code> . "i" est incrémenté de 1 puis sa valeur est attribuée à "a"

# Langage C: Les opérateurs

- **Opérateurs logiques (sur entiers uniquement) :**

Symbole	Opération bit à bit
<code>~</code>	complémentation (non logique)
<code>&amp;</code>	ET logique
<code> </code>	OU logique
<code>^</code>	OU exclusif
<code>&lt;&lt;n</code>	décalage à gauche de n bits
<code>&gt;&gt;n</code>	décalage à droite de n bits

# Langage C: Les opérateurs

- **Opérateurs de comparaison :**

Symbole	Fonction
<code>==</code>	égale à
<code>!=</code>	différent de
<code>&gt;</code>	supérieur à
<code>&lt;</code>	inférieur à
<code>&gt;=</code>	supérieur ou égal à
<code>&lt;=</code>	inférieur ou égal à

# Langage C: Les opérateurs - L'opérateur **cast**

- On peut forcer la conversion d'une expression grâce à l'opérateur cast

**(type) expression\_a\_caster**

```
1  #include <stdio.h>
2
3  int main()
4  {
5      double d;
6      int i, j;
7      i = 20;
8      j = 7;
9      d = i / j; /* d vaut 2 */
10     d = (double) (i/j); /* d vaut 2.0 */
11     d = (double) (i)/j; /* d vaut 2.857143 */
12     d = 20 / 7; /* d vaut 2 */
13     d = 20.0 / 7.0; /* d vaut 2.857143 */
14     return 0;
15 }
16
```

# Langage C: Priorité des opérateurs

**Si deux opérateurs possèdent la même priorité, ils seront exécuté de gauche à droite selon l'ordre d'apparition sur la ligne d'instruction**

Ordre	Opérateurs
1	( ) [ ] . ->
2	++ -- +(signe) -(signe) *(déréférence) &(adresse) ! ~ (type)(« type cast ») sizeof
3	*(multiplication) / %
4	+(addition) -(soustraction)
5	>> <<
6	== !=
7	&
8	^
9	
10	&&
11	
12	? :
13	= += -= *= /= %= &= ^=  = <<= >>=
14	,



# Exercices d'Application

**Exercice 1:** Soit les déclarations suivantes :

`int n = 10 , p = 4 ;`

`long q = 2 ;`

`float x = 1.75 ;`

Donner le type et la valeur de chacune des expressions suivantes :

a) `n + q`

b) `n + x`

c) `n % p + q`

d) `n < p`

e) `n >= p`

f) `n > q`

g) `q + 3 * (n > p)`

h) `q && n`

i) `(q-2) && (n-10)`

j) `x * (q==2)`

k) `x *(q=5)`

# Langage C Affichage de variables : **printf()**

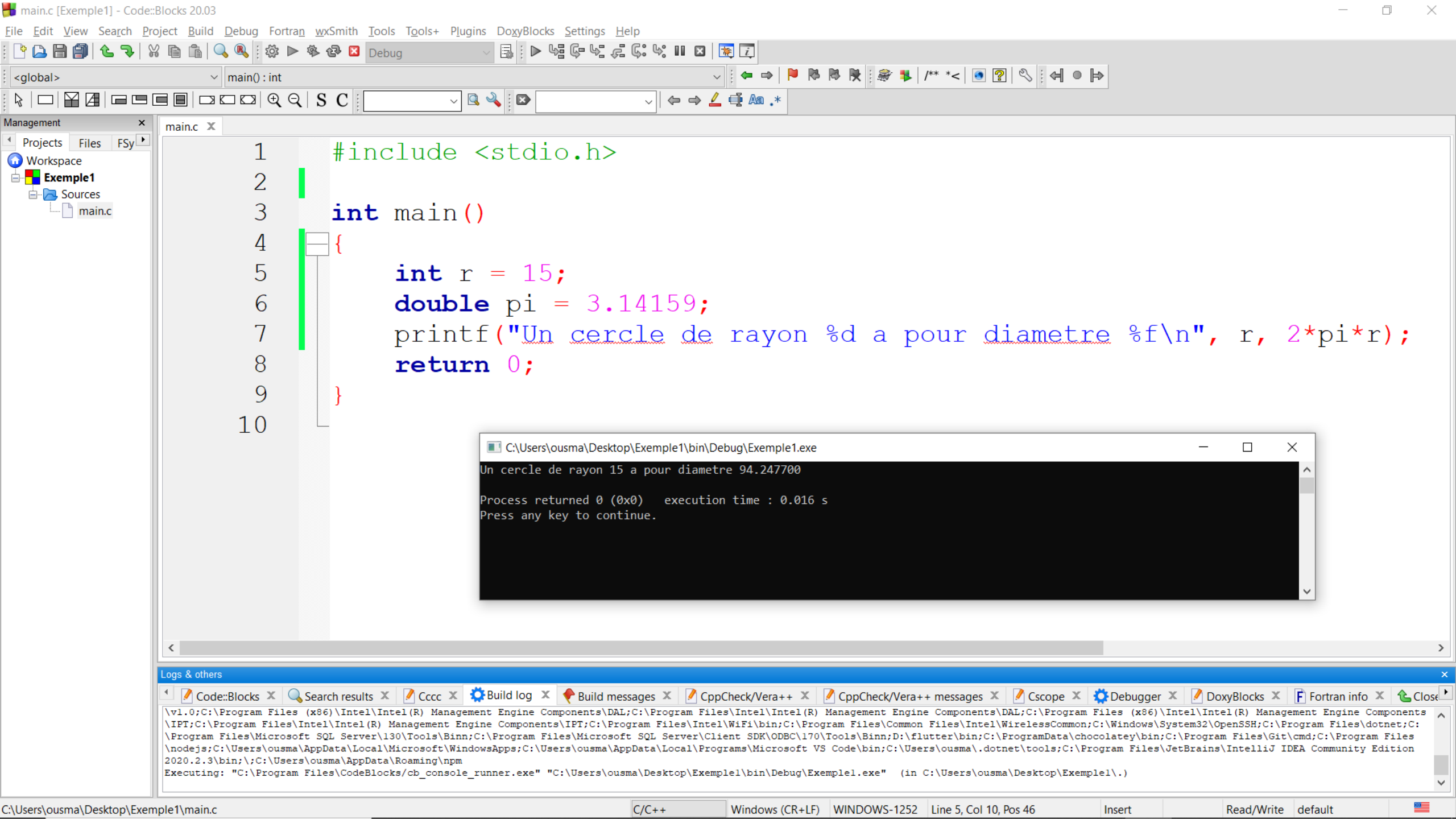
L'instruction **Afficher(variable1)** est exprimée en C de la façon suivante:

**printf("formatVariable1",variable1)**

**printf("formatVariable1 formatVariable2 ...",variable1, variable2, ...)**

Format	Type de variable correspondant
%c	Caractère de type char
%d ou %i	Entier de type int
%x	Entier de type int affiché en notation hexadécimale
%u	Entier non signé de type unsigned int
%f	Nombre à virgule flottante de type float ou double
%e ou %E	Nombre à virgule flottante affiché en notation exponentielle

**OUTPUT**



# Formatage des nombres avec **printf()**

Format	Résultat	Description
%6d	123	Largeur minimum de champ de 6 caractères
%06d	000123	Largeur minimum de champ de 6 caractères, remplissage avec des zéros
%-6d	123	Largeur minimum de champ de 6 caractères, cadrage à gauche
%6.3f	3.141	Largeur minimum de champ de 6 caractères avec 3 chiffres après la virgule

# Formatage des nombres avec **printf()**

- Le signe moins (-), placé immédiatement après le symbole % (comme dans %-4d ou %-10.3f), demande de cadrer l'affichage à gauche au lieu de le cadrer (par défaut) à droite ; les éventuels espaces supplémentaires sont donc placés à droite et non plus à gauche de l'information affichée.
- Le caractère \* figurant à la place d'un gabarit ou d'une précision signifie que la valeur effective est fournie dans la liste des arguments de printf. En voici un exemple dans lequel nous appliquons ce mécanisme à la précision :

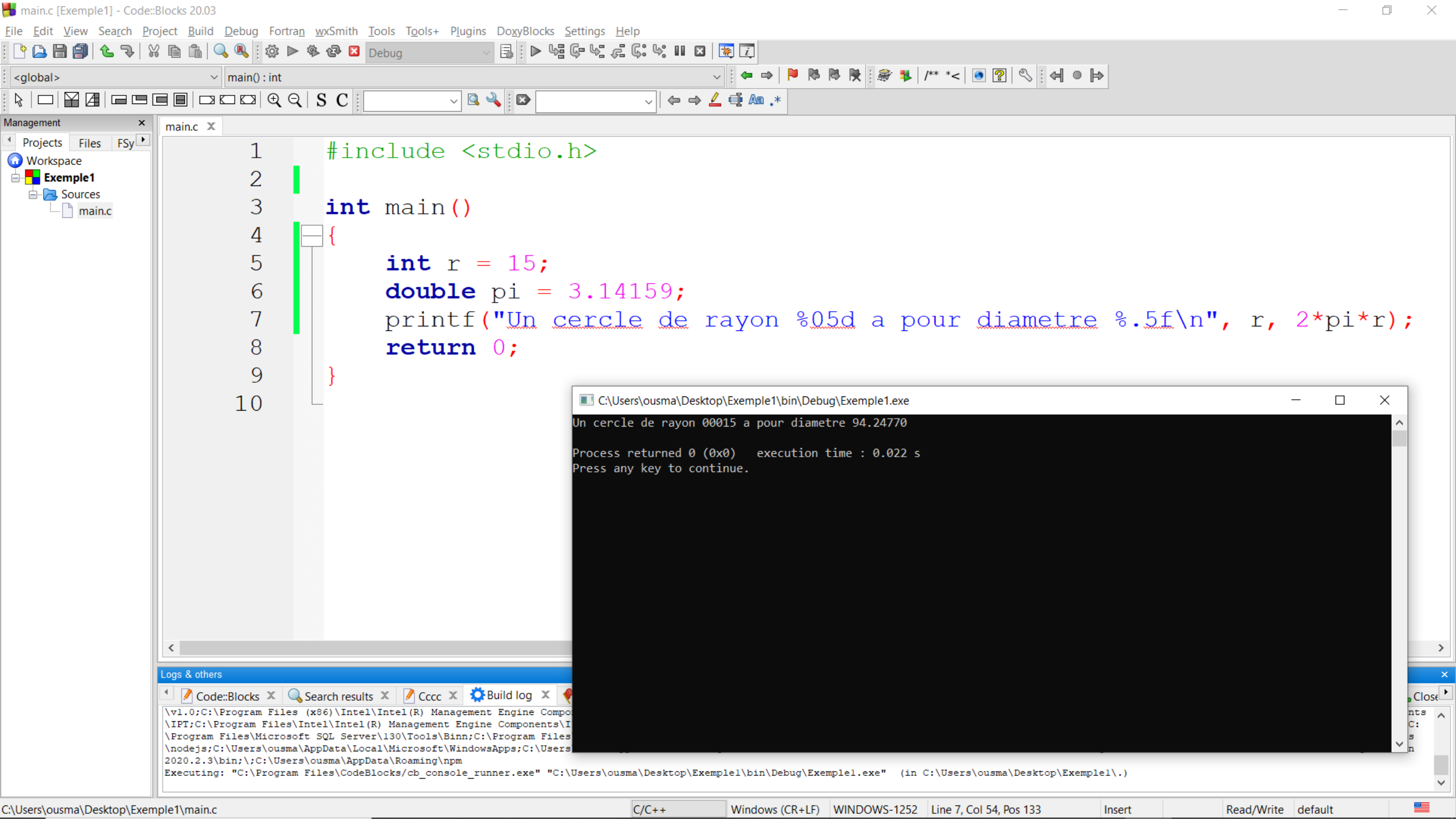
```
printf ("%8.*f", n, x) ;
```

```
n = 1      x = 1.2345
```

```
^^^^1.2
```

```
n = 3      x = 1.2345
```

```
^^^1.234
```



# Exercices d'Application

**Exercice 2:** Quels résultats fournit le programme suivant ?

```
#include <stdio.h>
main ()
{
    int n = 543 ;
    int p = 5 ;
    float x = 34.5678;
    printf ("A : %d %f\n", n, x) ;
    printf ("B : %4d %10f\n", n, x) ;
    printf ("C : %2d %3f\n", n, x) ;
    printf ("D : %10.3f %10.3e\n", x, x) ;
    printf ("E : %*d\n", p, n) ;
    printf ("F : %*.*f\n", 12, 5, x) ;
}
```



# Langage C: Lecture d'une valeur au clavier

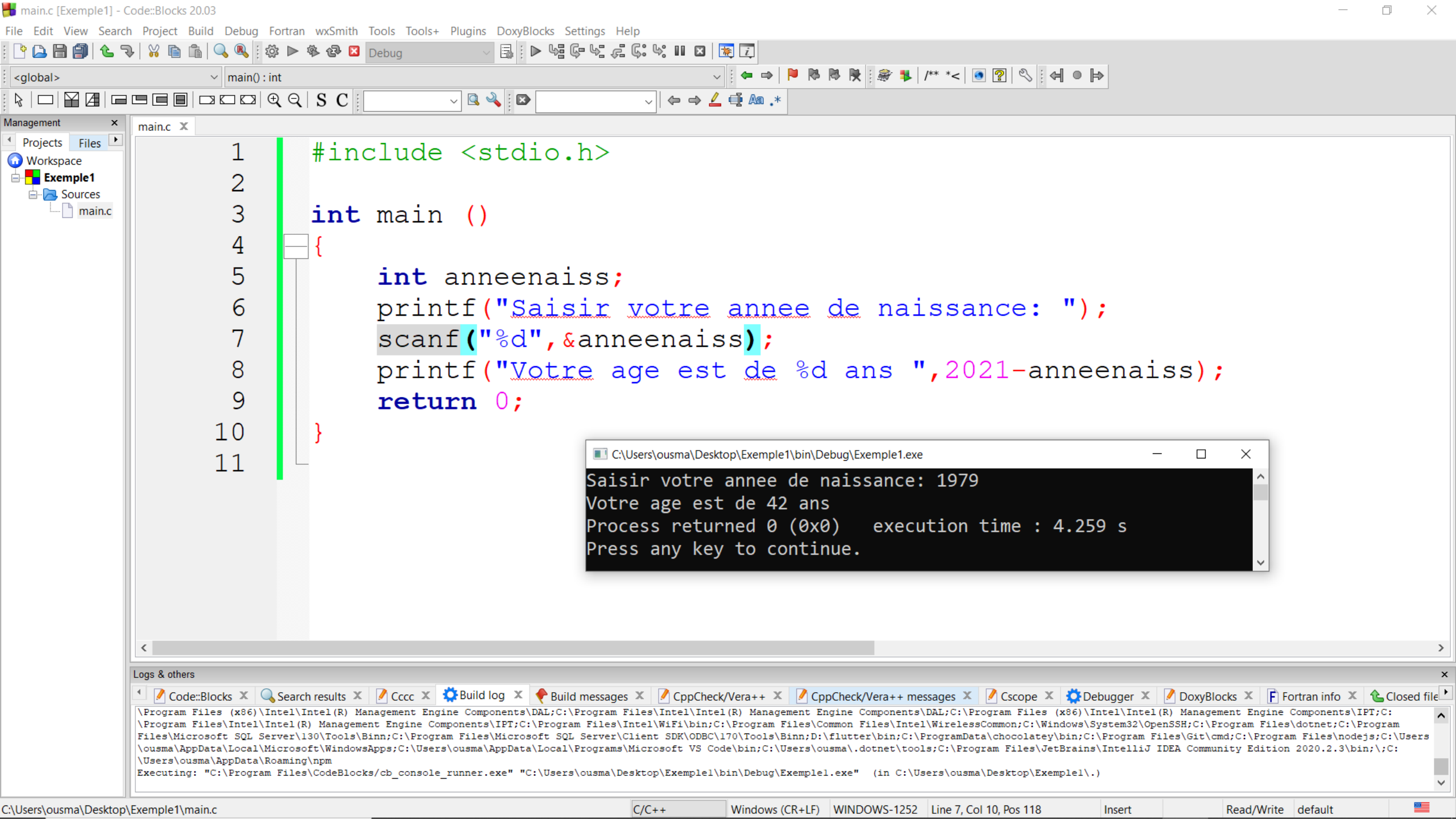
L'instruction **lire(variable1)** est exprimée en C de la façon suivante:

**scanf("format",variable1);**

**scanf("formatVariable1 formatVariable2 ...",&variable1, &variable2, ...)**

Variable	Instruction
int i;	scanf("%i", &i);
double d;	scanf("%lf", &d);
float f;	scanf("%f", &f);
short int h;	scanf("%hi", &h);
long int l;	scanf("%li", &l);
char c;	scanf("%c", &c); OU c=getchar();

**INPUT**



# Langage C: Lecture d'une valeur au clavier -

## Indication de la largeur maximale

- Pour tous les spécificateurs, nous pouvons indiquer la largeur maximale du champ à évaluer pour une donnée. Les chiffres qui passent au-delà du champ défini sont attribués à la prochaine variable qui sera lue !
- Exemple: Soient les instructions:  
    int A,B;  
    scanf("%4d %2d", &A, &B);
- Si nous entrons le nombre 1234567, nous obtiendrons les affectations suivantes:  
    A=1234  
    B=56
- le chiffre 7 sera gardé pour la prochaine instruction de lecture.

# Langage C: Lecture d'une valeur au clavier - **Formats 'spéciaux'**

- Si la chaîne de format contient aussi d'autres caractères que des signes d'espacement, alors ces symboles doivent être introduits exactement dans l'ordre indiqué.

```
int jour, mois, annee;  
scanf("%d/%d/%d", &jour, &mois, &annee);
```

Accepte les entrées:	Rejette les entrées:
<b>12/4/1980</b>	<b>12 4 1980</b>
<b>12/04/01980</b>	<b>12 /4 /1980</b>

# Exercices d'Application

**Exercice 3:** Quelles seront les valeurs lues dans les variables  $n$  et  $p$  (de type *int*), par l'instruction suivante ?

```
scanf ("%4d %2d", &n, &p) ;
```

lorsqu'on lui fournit les données suivantes (le symbole ^ représente un espace et le symbole @ représente une fin de ligne, c'est-à-dire une validation) ?

a) 12^45@

b) 123456@

c) 123456^7@

d) 1^458@

e) ^^^4567^^8912@

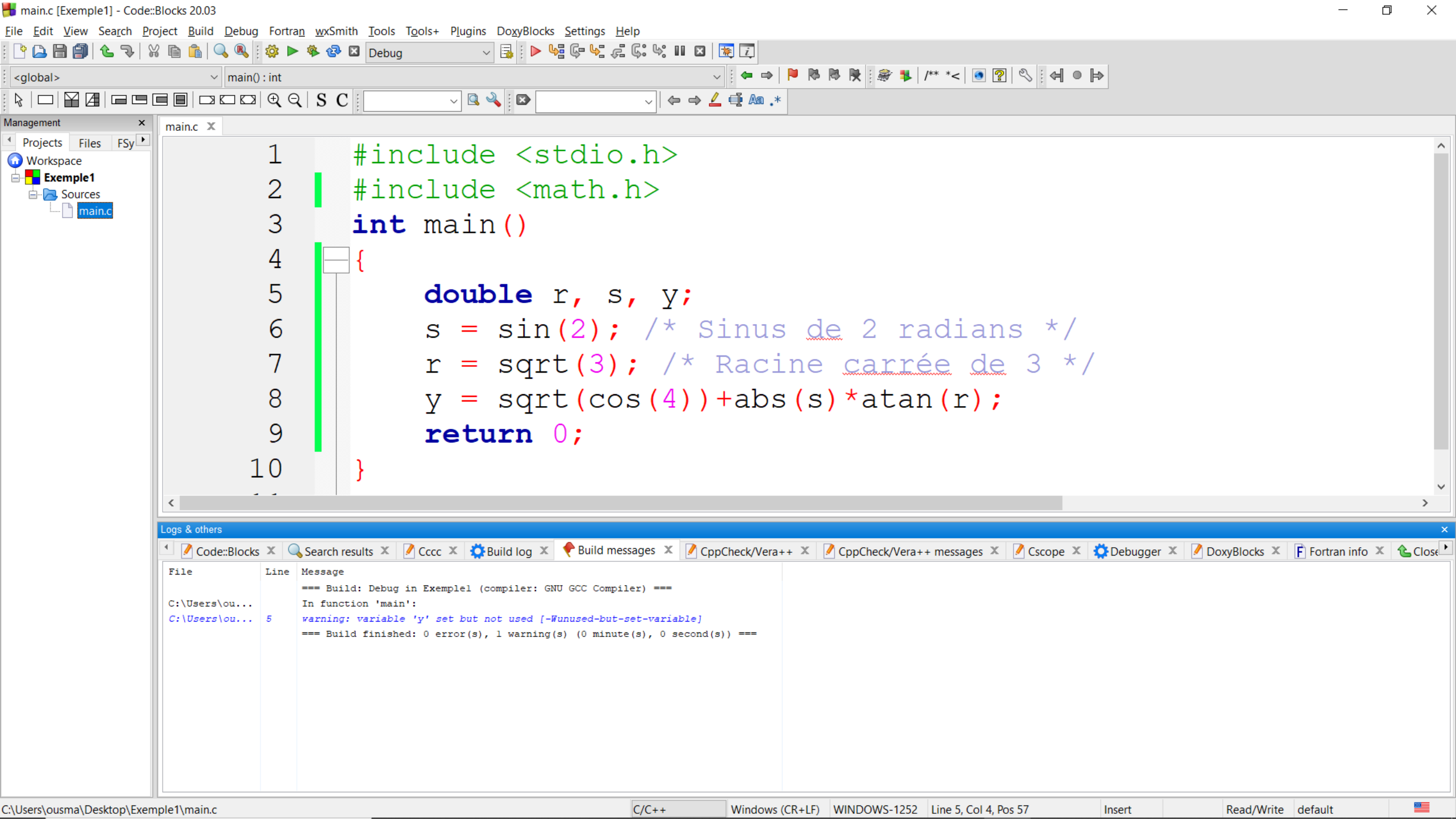
# Compléments et Exemples

# Fonctions mathématiques standard

Les fonctions mathématiques suivantes sont disponibles en C (parmi d'autres), elles sont déclarées dans le fichier **math.h**

La librairie mathématique	
Attention : cette librairie travaille avec des <u>nombre</u> s décimaux !	
# include <math.h>	Directive préprocesseur
fabs(nombre)	Valeur absolue d'un décimal
Valeur absolue d'un entier : <b>abs(nombre)</b> - dispo dans <stdlib.h>	
ceil(nombre)	Retourne le nombre entier supérieur
floor(nombre)	Nombre entier inférieur
pow(nombre, exposant)	Calcul de puissances
sqrt(nombre)	Racine carée
sin(x), cos(x), tan(x) asin(x), acos(x), atan(x)	Fonctions trigonométriques. 1 radiant = 1 degré * 180 / Pi
exp(nombre)	Exponentielle d'un nombre décimal
log(nombre)	Logarithme népérien (de Néper)
log10(nombre)	Logarithme décimal (en base 10)
# define PI 3.14159265359	Définir le nombre Pi





```
1  #include <stdio.h>
2  #include <math.h>
3  int main()
4  {
5      double r, s, y;
6      s = sin(2); /* Sinus de 2 radians */
7      r = sqrt(3); /* Racine carrée de 3 */
8      y = sqrt(cos(4)) + abs(s) * atan(r);
9      return 0;
10 }
```

## Logs &amp; others

File	Line	Message
=== Build: Debug in Exemple1 (compiler: GNU GCC Compiler) ===		
In function 'main':		
C:\Users\ou...	5	warning: variable 'y' set but not used [-Wunused-but-set-variable]
=== Build finished: 0 error(s), 1 warning(s) (0 minute(s), 0 second(s)) ===		

# Erreurs communes

- Ne pas confondre `==` et `=`  
`=` est l'opérateur d'affectation, `==` est l'opérateur de test d'égalité
- Ne pas oublier le signe `&` dans l'instruction `scanf` (nombres et caractères)  
Exemple: `scanf("%d", &i);`
- Division réelle et division entière: La division est faite en réels si l'un au moins des deux opérandes est réel, en entiers sinon
- Pas d'espaces, de virgules, `\n`, etc..., dans la chaîne de format de `scanf`  
Exemple: `scanf("%d%d", &i, &j);`
- Utiliser `getchar()` pour éliminer les `\n` qui traînent: problèmes de tampon

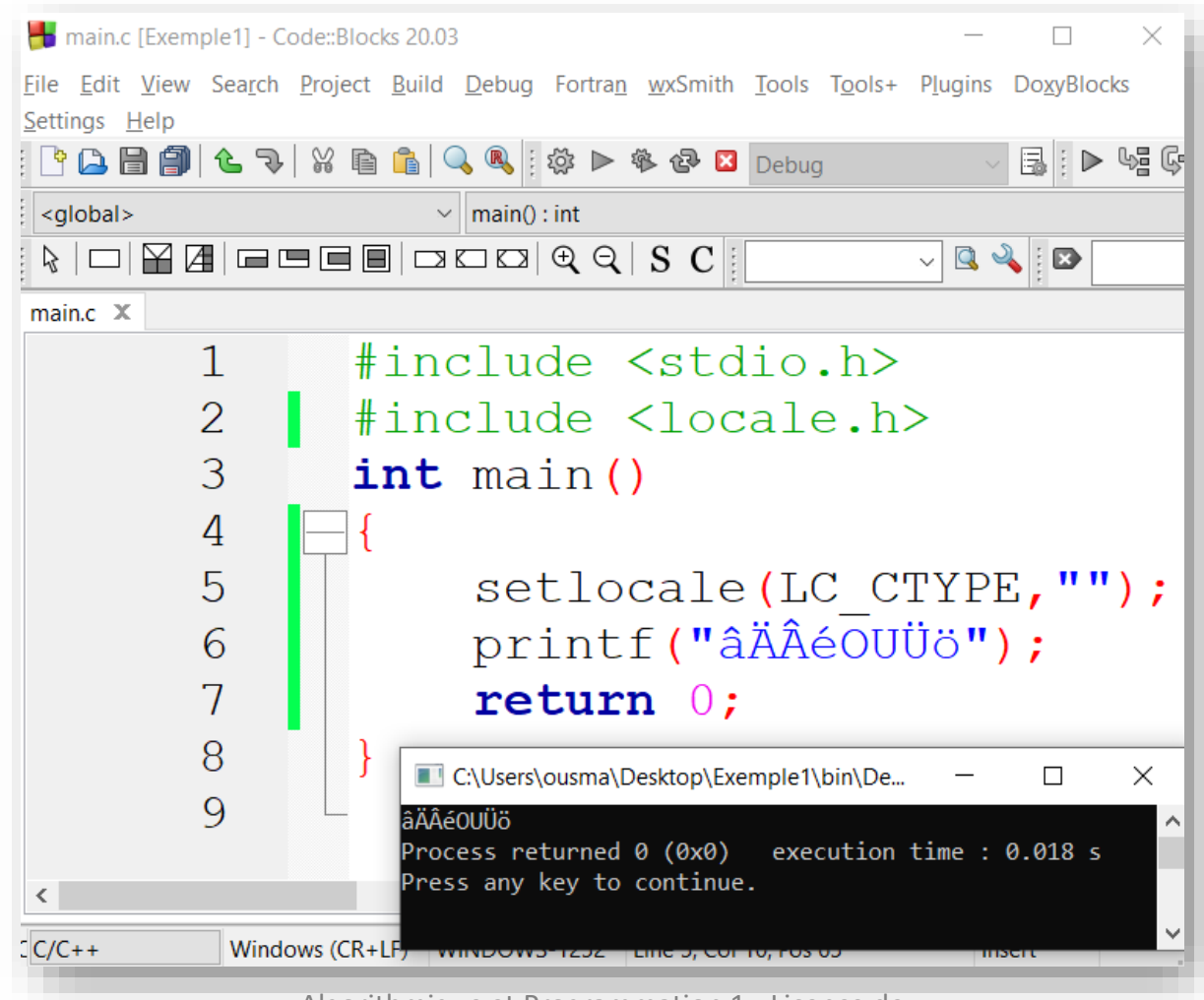
# Conseils et suggestions

- Utiliser des **doubles** plutôt que des **float** (meilleure précision)

Rappel: **printf("%f", dbl);** et **scanf("%lf", &dbl);** Une erreur fréquente consiste à penser que %lf équivaut à *double*. Si cela est vrai dans un *scanf ()*, il en est autrement dans le *printf ()*

- Pensez aux **#includes** : **math.h**, **stdlib.h**
  - **math.h**: fonctions mathématiques (**ceil**, **floor**, **pow**...)
  - **stdlib.h**: essentiellement les fonctions **rand()** et **srand()**
- Utiliser **x\*x** pour élever x au carré plutôt que **pow(x,2)**

# Conseils et suggestions: Pour les caractères accentués – utiliser **locale.h**



```
main.c [Exemple1] - Code::Blocks 20.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks
Settings Help
<global> main() : int
main.c x
1 #include <stdio.h>
2 #include <locale.h>
3 int main()
4 {
5     setlocale(LC_CTYPE, "");
6     printf("âÄÂéOUÜö");
7     return 0;
8 }
9
```

C:\Users\ousma\Desktop\Exemple1\bin\De...  
âÄÂéOUÜö  
Process returned 0 (0x0) execution time : 0.018 s  
Press any key to continue.

# Arithmétique entière et à virgule flottante

**double** x;

**int** i, j;

instruction	signification
<code>i = 10 / 3;</code>	i de type <b>int</b> reçoit la valeur 3, résultat de la division entière
<code>i = 10.0 / 3;</code>	i de type <b>int</b> reçoit la valeur 3 par arrondi automatique de la division réelle
<code>x = 10 / 3;</code>	x de type <b>double</b> reçoit la valeur 3, résultat de la division entière
<code>x = 10/3 * 3;</code>	x reçoit la valeur 9.0, le calcul est fait en entiers puis transformé en réel.
<code>x = 10.0/3 * 3;</code>	x reçoit la valeur 10.0 la division et la multiplication étant effectués en réels.
<code>x=i/j;</code>	Si les entiers i et j valent 7 et 2, x de type <b>double</b> reçoit la valeur 3, résultat de la division entière
<code>x=(double) i/j;</code>	Si les entiers i et j valent 7 et 2, x de type <b>double</b> reçoit la valeur 3.5, résultat de la division réelle de i, préalablement converti explicitement en réel, par j automatiquement converti en réel

# Constructions abrégées

Construction abrégée	Construction équivalente
<code>a++;</code>	<code>a = a+1;</code>
<code>++a;</code>	<code>a = a+1;</code>
<code>a--;</code>	<code>a = a-1;</code>
<code>--a;</code>	<code>a = a-1;</code>
<code>a += b;</code>	<code>a = a+b;</code>
<code>a -= b;</code>	<code>a = a-b;</code>
<code>a *= b;</code>	<code>a = a*b;</code>
<code>a /= b;</code>	<code>a = a/b;</code>
<code>a=(b&gt;0) ? 12 : c;</code>	<pre>if (b&gt;0) {     a = 12; } else {     a = c; }</pre>

```
int a = 12;  
int b;
```

`b = a++;`



**b vaut 12  
a vaut 13**

`b = ++a;`



**b vaut 13  
a vaut 13**

`b = a--;`



**b vaut 12  
a vaut 11**

`b = --a;`



**b vaut 11  
a vaut 11**

# Nombres pseudo-aléatoires

- Voici un petit exemple de programme qui permet d'obtenir des nombres pseudo-aléatoires entre 0 et 99 :

```
1 #include <stdio.h>
2 #include <stdlib.h> // sert pour les fonctions srand et rand
3 #include <time.h>
4 int main() {
5     int nb_alea=0;
6     /* Initialisation du générateur de nombres
7      basée sur la date et l'heure */
8     srand (time (NULL));
9     nb_alea = rand() % 100;
10    printf ("Nombre aleatoire : %d\n",nb_alea);
11    getchar();
12    return 0;
13 }
```

**rand** est la fonction qui retourne un nombre aléatoire à chaque appel. Ce nombre est compris entre 0 et **RAND\_MAX**.

La fonction **srand** permet d'initialiser le générateur de nombres. Elle ne doit être appelée qu'une seule fois avant tout appel à **rand**.



# Nombres pseudo-aléatoires

- **srand (time (NULL))** permet d'initialiser le générateur de nombres pseudo-aléatoire.
- **rand()** renvoie un nombre entier compris entre **0** et **RAND\_MAX**.
- **rand()%100** est donc le reste de la division entière d'un nombre pseudo-aléatoire (éventuellement très grand) par 100, c'est-à-dire un nombre compris entre 0 et 99...

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 int main() {
4     /* Pour notre information */
5     printf ("RAND_MAX : %ld\n", RAND_MAX);
6     return 0;
7 }
```

# Norme C99: Types complexes

```
1  #include <stdio.h>
2  #include <complex.h>
3  int main ()
4  {
5      double complex x = 1 + 2*I;
6      double complex y = 2 - 3*I;
7      double complex somme = x + y;
8      printf("x+y == %.1f%+.1fi\n", creal(somme), cimag(somme));
9      return 0;
10 }
11
```

```
C:\Users\ousma\Desktop\Exemple1\bin\Debug\Exemple1.exe
x+y == 3.0-1.0i

Process returned 0 (0x0)   execution time : 0.096 s
Press any key to continue.
```

# Norme C99: le type **bool** de **<stdbool.h>**

- Permet de définir un pseudo type booléen (en fait un entier non signé) afin d'améliorer la lisibilité de vos signatures de fonctions et du typage de vos variables. Ainsi, les fonctions manipulant des booléens apparaîtront avec plus d'évidences.
- Il est à noter que cette entête a été introduite avec C ISO 1999 (C99).

## Type *bool* dans *stdbool.h*

```
#define bool unsigned int  
#define true 1  
#define false 0
```

main.c [Exemple1] - Code::Blocks

File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help

<global>

main() : int

Management

Projects

Workspace

Exemple1

Sources

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

#include <stdio.h>

#include <stdbool.h>

int main ()

{

bool genre;

printf("Etes-vous un homme ?\n");

printf("(Saisir 1 ou true pour homme et 0 ou false sinon): ");

scanf("%d",&genre);

genre==true ?

printf("Vous etes donc un homme !!!");

printf("Vous etes donc une femme !!!");

return 0;

}

C:\Users\ousma\Desktop\Exemple1\bin\Debug\Exemple1.exe

Etes-vous un homme ?  
(Saisir 1 ou true pour homme et 0 ou false sinon): true  
Vous etes donc une femme !!!  
Process returned 0 (0x0) execution time : 5.358 s  
Press any key to continue.

Logs & others

Code::Blocks Search results Cccc Build log Build messages CppCheck/Vera++ CppCheck/Vera++ messages Cscope Debugger DoxyBlocks Fortran info Closed files li

261:C:\Program Files (x86)\Intel\iCLS Client;C:\Program Files\Intel\iCLS Client;C:\Windows\System32;C:\Windows;C:\Windows\System32\wbem;C:\Windows\System32\WindowsPowerShell\v1.0;C:\Program Files (x86)\Intel\Intel(R) Management Engine Components\DAL;C:\Program Files\Intel\Intel(R) Management Engine Components\DAL;C:\Program Files (x86)\Intel\Intel(R) Management Engine Components\IPT;C:\Program Files\Intel\Intel(R) Management Engine Components\IPT;C:\Program Files\Intel\WiFi\bin;C:\Program Files\Common Files\Intel\WirelessCommon;C:\Windows\System32\OpenSSH;C:\Program Files\dotnet;C:\Program Files\Microsoft SQL Server\130\Tools\Binn;C:\Program Files\Microsoft SQL Server\Client SDK\ODBC\170\Tools\Binn;D:\flutter\bin;C:\ProgramData\chocolatey\bin;C:\Program Files\Git\cmd;C:\Program Files\nodejs;C:\Users\ousma\AppData\Local\Microsoft\WindowsApps;C:\Users\ousma\AppData\Local\Programs\Microsoft VS Code\bin;C:\Users\ousma\.dotnet\tools;C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2020.2.3\bin;;C:\Users\ousma\AppData\Roaming\npm

Executing: "C:\Program Files\CodeBlocks\cb\_console\_runner.exe" "C:\Users\ousma\Desktop\Exemple1\bin\Debug\Exemple1.exe" (in C:\Users\ousma\Desktop\Exemple1\.)

C:\Users\ousma\Desktop\Exemple1\main.c

C/C++ Windows (CR+LF) WINDOWS-1252 Line 5, Col 16, Pos 73 Insert Read/Write default

# Caractères spéciaux

**\a** - *beep* -

**\b** retour arrière

**\f** chargement de page

**\n** saut de ligne

**\r** retour en début ligne

**\t** tabulation horizontale

**\v** tabulation verticale

**\\** barre oblique inverse

**\'** apostrophe

**\"** guillemet

**\?** point d'interrogation

**\nnn** valeur ASCII en octal

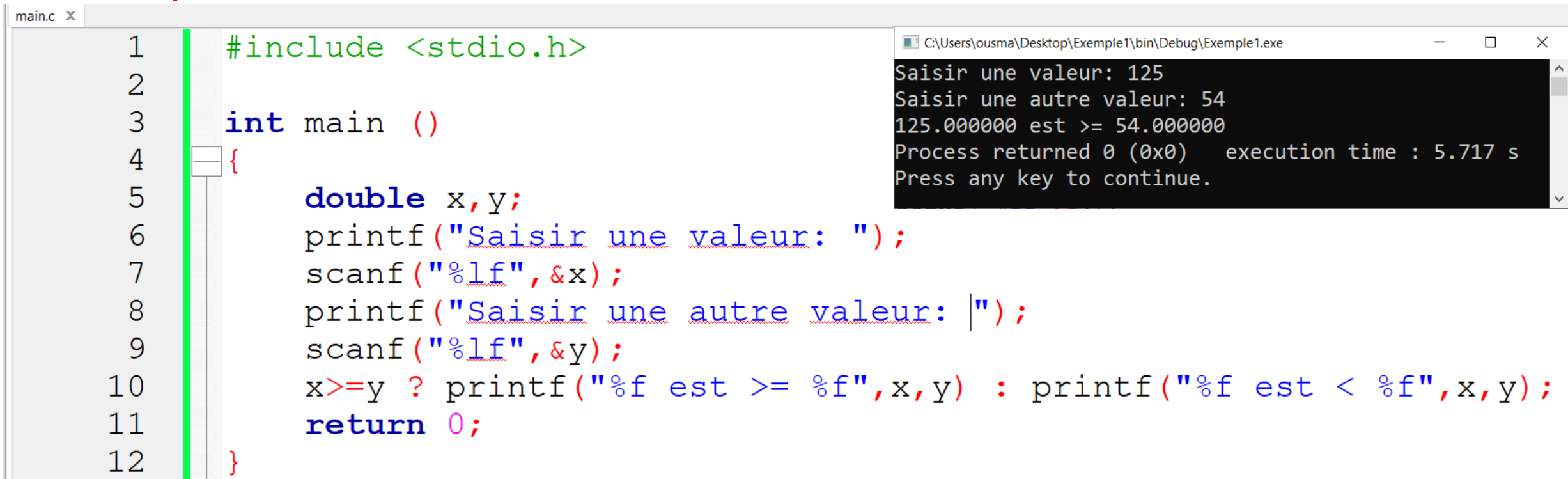
**\xnnn** valeur ASCII en hexadécimal

**\0** fin de la chaîne de caractère

# Langage C: Les opérateurs - **L'opérateur conditionnel**

- Ternaire – Est évalué de gauche à droite

**exp ? val1 : val2**      retourne **val1** si **exp** est **vraie** sinon



The image shows a code editor window titled 'main.c' and a console window titled 'C:\Users\ousma\Desktop\Exemple1\bin\Debug\Exemple1.exe'.

The code in 'main.c' is as follows:

```
1  #include <stdio.h>
2
3  int main ()
4  {
5      double x, y;
6      printf("Saisir une valeur: ");
7      scanf("%lf", &x);
8      printf("Saisir une autre valeur: ");
9      scanf("%lf", &y);
10     x >= y ? printf("%f est >= %f", x, y) : printf("%f est < %f", x, y);
11     return 0;
12 }
```

The console output shows the execution of the program:

```
Saisir une valeur: 125
Saisir une autre valeur: 54
125.000000 est >= 54.000000
Process returned 0 (0x0)   execution time : 5.717 s
Press any key to continue.
```

# Exercices d'Application

**Exercice 4:** Écrire plus simplement l'instruction suivante :

$$z = (a > b ? a : b) + (a \leq b ? a : b) ;$$

**Exercice 5:** n étant de type int, écrire une expression qui prend la valeur :

- -1 si n est négatif,
- 0 si n est nul,
- 1 si n est positif.

# Exercices d'Application

**Exercice 6:** Quels résultats fournit le programme suivant ?

```
main.c x
1  #include <stdio.h>
2  int main()
3  {
4      int n=10, p=5, q=10, r ;
5      r = n == (p = q) ;
6      printf ("A : n = %d p = %d q = %d r = %d\n", n, p, q, r) ;
7      n = p = q = 5 ;
8      n += p += q ;
9      printf ("B : n = %d p = %d q = %d\n", n, p, q) ;
10     q = n < p ? n++ : p++ ;
11     printf ("C : n = %d p = %d q = %d\n", n, p, q) ;
12     q = n > p ? n++ : p++ ;
13     printf ("D : n = %d p = %d q = %d\n", n, p, q) ;
14 }
15
```



## Exemple 1 (saisie et affichage)



**Ecrivez un algorithme puis le programme en langage C correspondant qui demande à l'utilisateur de saisir un nombre entier , puis qui calcule et affiche le double de ce nombre**

# Exemple 1 (saisie et affichage)

**Algorithme** Calcul\_double

**variables** A, B : **entier**;

**Début**

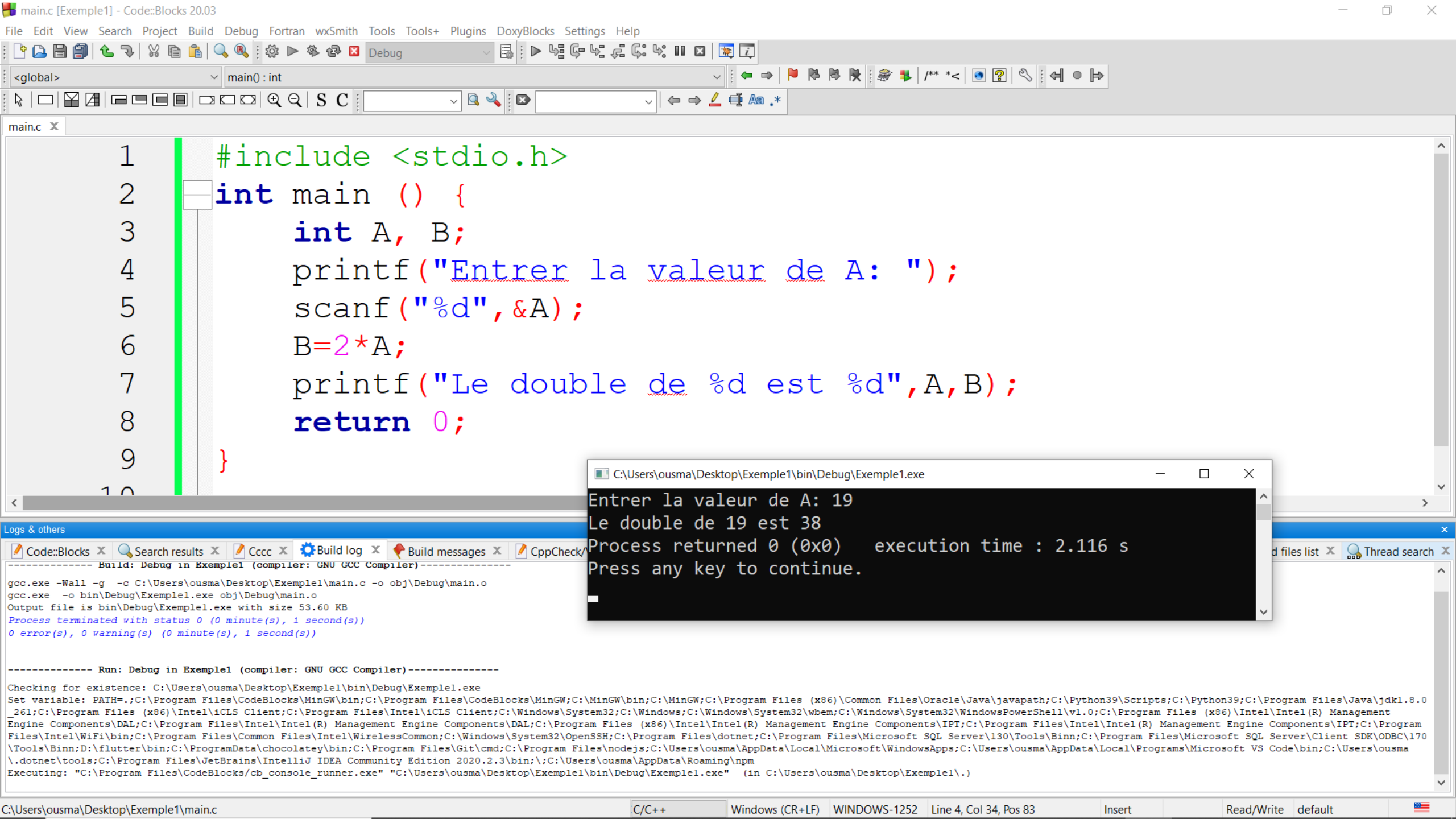
**Afficher**("Entrer la valeur de A ");

**Lire**(A);

$B \leftarrow 2 * A$ ;

**Afficher**("Le double de ", A, "est :", B);

**Fin**



## Exemple 2 (saisie et affichage)



Ecrivez un algorithme puis le programme en langage C correspondant qui **à partir du prix hors taxe PHT d'un produit et du taux de TVA** saisie au clavier **calcule et affiche le prix toutes taxes comprises PTTC.**

# Exemple 2 (saisie et affichage)

## Algorithme ParExemple

*{Saisie un prix HT et affiche le prix TTC correspondant}*

**Constantes** TVA  $\leftarrow$  20.6 ;

Titre  $\leftarrow$  "Résultat";

**variables** prixHT, prixTTC : réels;

*{déclarations}*

**Début**

*{préparation du traitement}*

**Afficher**("Donnez-moi le prix hors taxe :");

**Lire**(prixHT);

prixTTC  $\leftarrow$  prixHT \* (1+TVA/100);

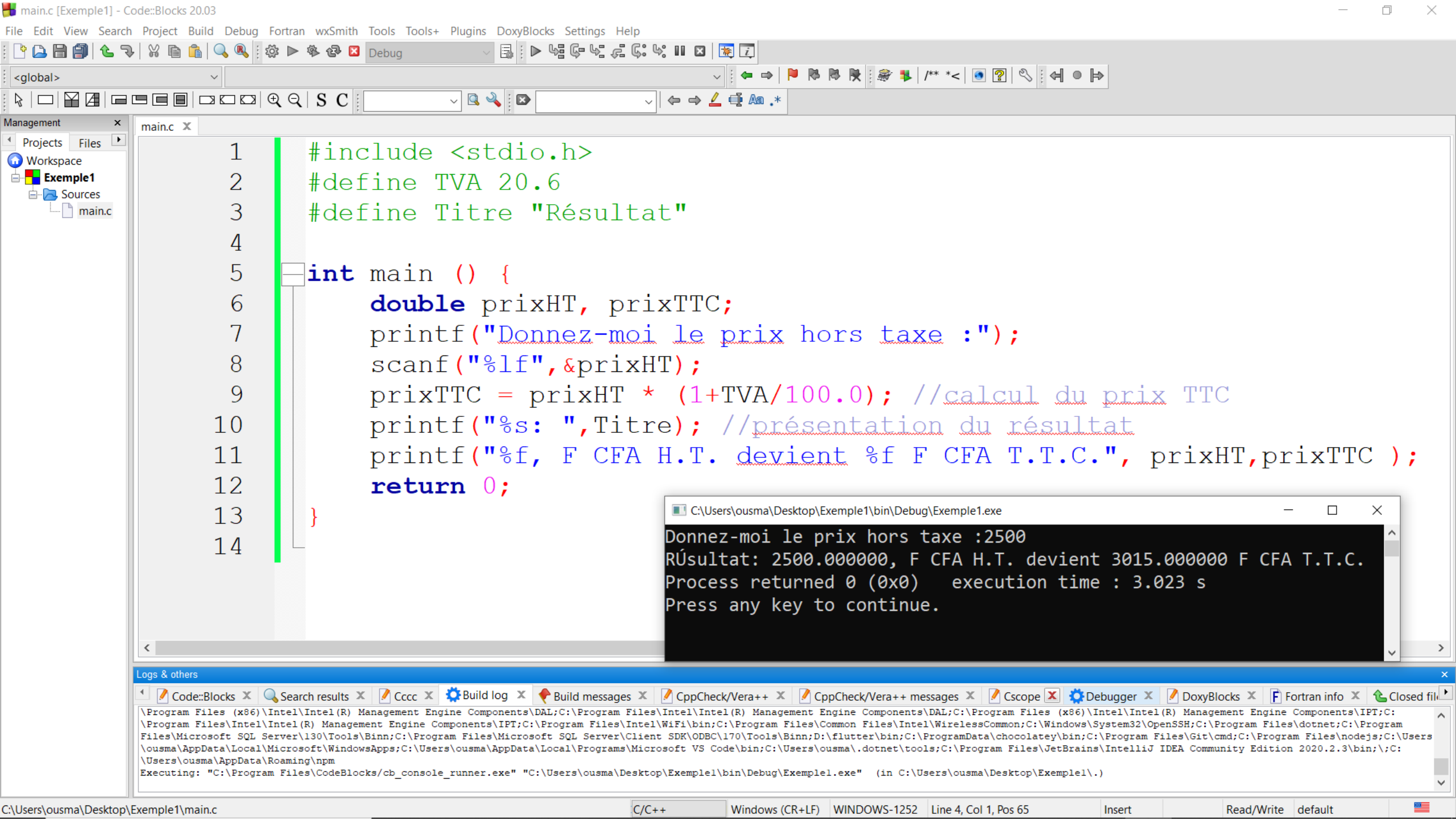
*{calcul du prix TTC}*

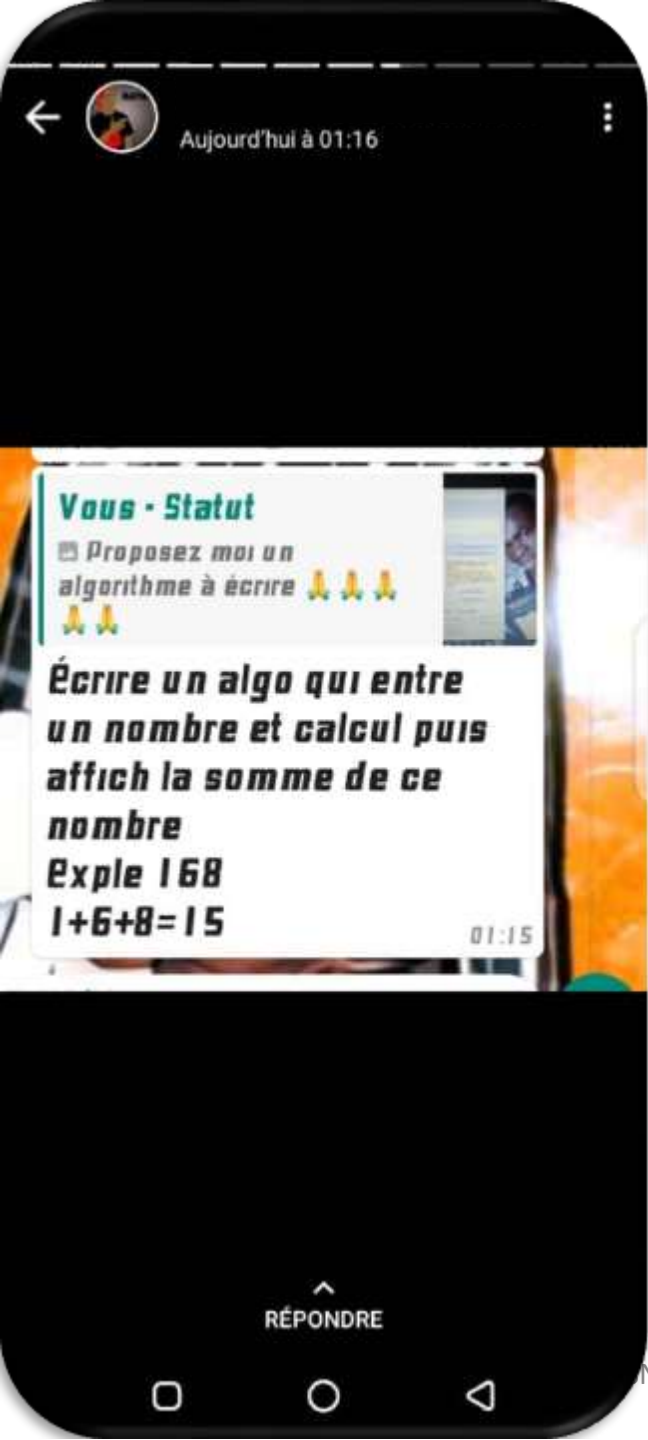
**Afficher**(Titre);

*{présentation du résultat}*

**Afficher**(prixHT, "F CFA H.T. devient ", prixTTC, " F CFA T.T.C.");

**Fin**





## Exemple3 (saisie et affichage)



Ecrivez un algorithme puis le programme en langage C correspondant qui **demande de saisir un nombre à 3 chiffres puis calcule et affiche la somme des chiffres qui le compose**

# Exemple3 (saisie et affichage)

## **Algorithme SommeChiffres**

**variables** nombre, nombre: **entier**;

### **Début**

**Afficher**("Saisir un nombre inférieur ou égal à 999: ");

**Lire**(nombre);

**somme**  $\leftarrow$  nombre/100;

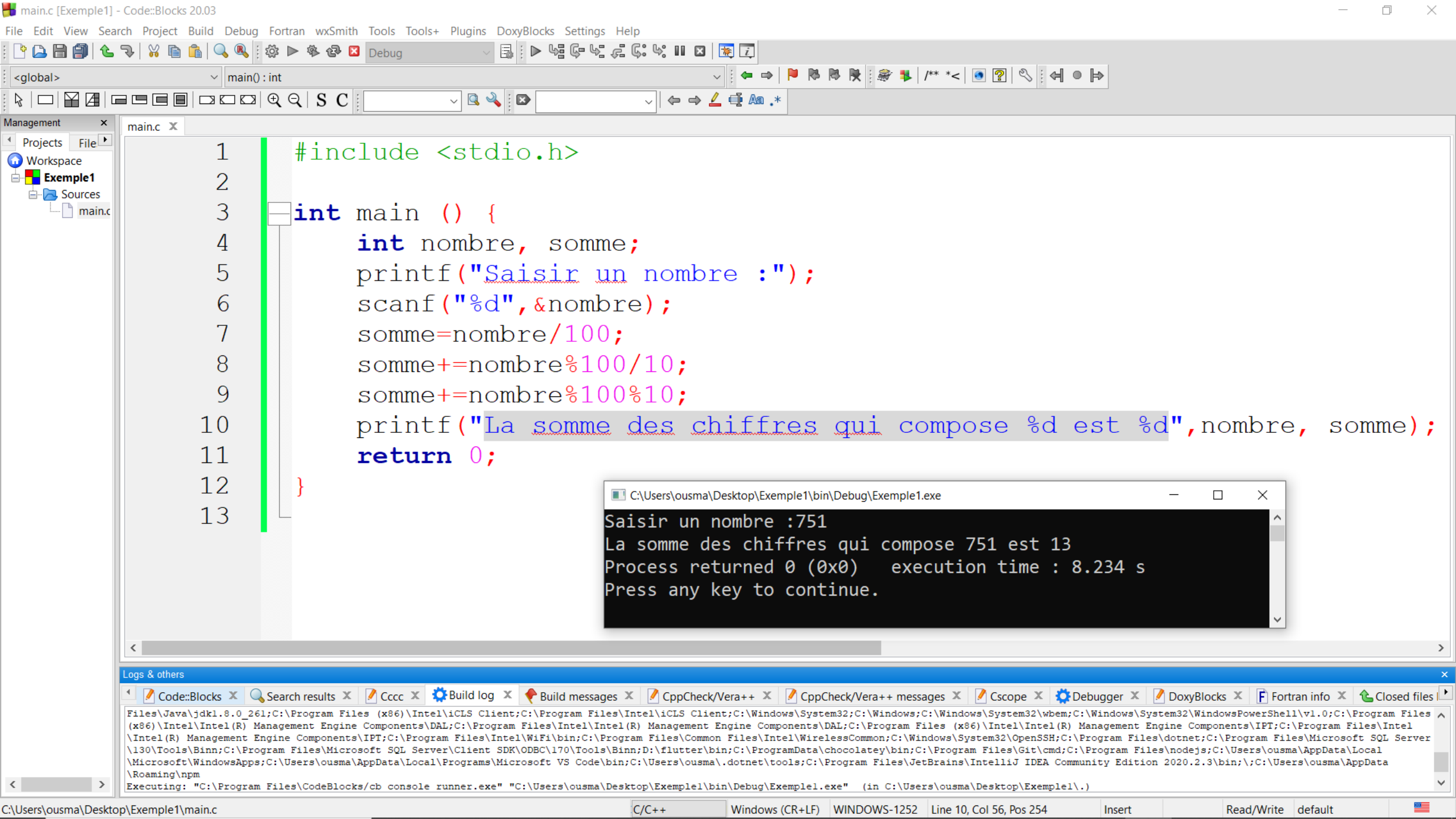
**somme**  $\leftarrow$  somme + nombre%100/10;

**somme**  $\leftarrow$  somme + nombre%100%10;

**Afficher**(" La somme des chiffres qui composent ", nombre, " est ",somme);

### **Fin**





Programmation 1 - Travaux Dirigés n°1



# Préparer la fiche de TD n°1 de Programmation 1

# Programmation 1 - Travaux Dirigés n°1

- **Exercice 01:** Ecrire un algorithme puis un programme en C qui demande à l'utilisateur un nombre , puis calcule son cube.
- **Exercice 02:** Ecrire un algorithme puis un programme en C qui demande à l'utilisateur deux nombres et qui calcule leur moyenne.
- **Exercice 03:** Ecrire un programme en langage C qui demande à l'utilisateur de donner un nombre de jours et affiche son équivalent en nombre d'années, semaines et jours restant. Une année compte 365 jours et une semaine 7 jours.

```
Donnez le nombre de jours : 1329
1329 jours correspond à 3 années, 33 semaines et 3jours.
```

# Programmation 1 - Travaux Dirigés n°1

- **Exercice 04:** Ecrire un programme qui déclare trois variables D, P et S et affecter respectivement à ces variables les valeurs du diamètre, du périmètre et de la surface d'un cercle dont le rayon est R. On affichera à l'écran le contenu de ces différentes variables selon le format suivant

***Un cercle de rayon WW a pour diamètre XX, pour circonférence YY et pour surface ZZ.***

- **Exercice 05:** Ecrire un algorithme puis un programme en C qui demande à l'utilisateur de donner les longueurs des trois côtés d'un triangle ABC. On veut savoir si le triangle est rectangle en C.

# Programmation 1 - Travaux Dirigés n°1

- **Exercice 06:** On lance un dé à six faces, non truqué. Si on obtient six, on a gagné. Ecrire un algorithme qui permet de simuler un tel lancer de dé et qui affiche « C'est gagné » quand le résultat est six. Avec Algobox, la fonction RANDOM renvoie un nombre aléatoire compris entre 0 et 1. Ecrire un algorithme puis un programme en C
- **Exercice 07:** Exercice 7 : Ecrire l'algorithme qui lit les coordonnées de deux vecteurs  $u$  et  $v$ , et de calculer leur norme et leur produit scalaire
- **Exercice 08:** Ecrivez un algorithme qui à partir du rayon d'un cercle saisie au clavier, calcule son diamètre, sa surface et sa circonférence.

# Programmation 1 - Travaux Dirigés n°1

- **Exercice 09:** Ecrire un algorithme puis un programme en C qui permet la saisie d'afficher les résultats des opérations de base sur les nombres complexes (Somme, Soustraction, Produit, conjugué d'un nombre complexe)
- **Exercice 10:** Dans la librairie `<math.h>` La fonction `abs` permet d'obtenir la valeur absolue d'un nombre entier. Ecrire le programme permettant de saisir `a` et `b` :
  - La fonction `fabsf` permet d'obtenir la valeur absolue d'un float. Utilisez cette dernière fonction pour calculer la valeur absolue de  $(a-b)$ .
  - La fonction `ceilf` permet d'obtenir l'arrondi entier supérieur d'un flottant. Utilisez cette fonction pour calculer l'arrondi supérieur de  $(a/b)$ .

# Bibliographie et Webographie

- Tapez "cours langage c" sur GOOGLE <http://www.google.sn/>
- <https://openclassrooms.com/fr/courses/4366701-decouvrez-le-fonctionnement-des-algorithmes>
- Cours : Algorithmique et Programmation 1  
<http://foad.ugb.sn/course/view.php?id=267>
- <https://algo.developpez.com/cours/>
- Algorithmique,...
  - Tapez "cours Algorithmique" sur GOOGLE <http://www.google.sn/>
- ...