

TP N°1 : Programmation en langage C LE – Informatique et Mathématique

Objectif : Initiation au langage C

Prérequis : Installation du logiciel **Code::Blocks**

Important : Les solutions des exercices du TP doivent être codées en langage C, compilées et exécutées. Ensuite, le travail à rendre sur la plateforme sera sous forme d'un **rapport** (un fichier **word** ou **pdf**) contenant pour chaque exercice les captures d'écran du code source ainsi que celles des résultats après l'exécution.

1. Téléchargement et installation du logiciel Code::Blocks

Pour télécharger le logiciel **Code::Blocks**, allez sur le lien:

<http://www.codeblocks.org/downloads/26>

Une liste de versions est proposée, vous choisissez celle appropriée selon votre système d'exploitation (32 ou 64 bits):

- Pour 64 bits, téléchargez la version : **codeblocks-20.03mingw-setup.exe**
- Pour 32 bits, téléchargez la version : **codeblocks-20.03mingw-32bit-setup.exe**



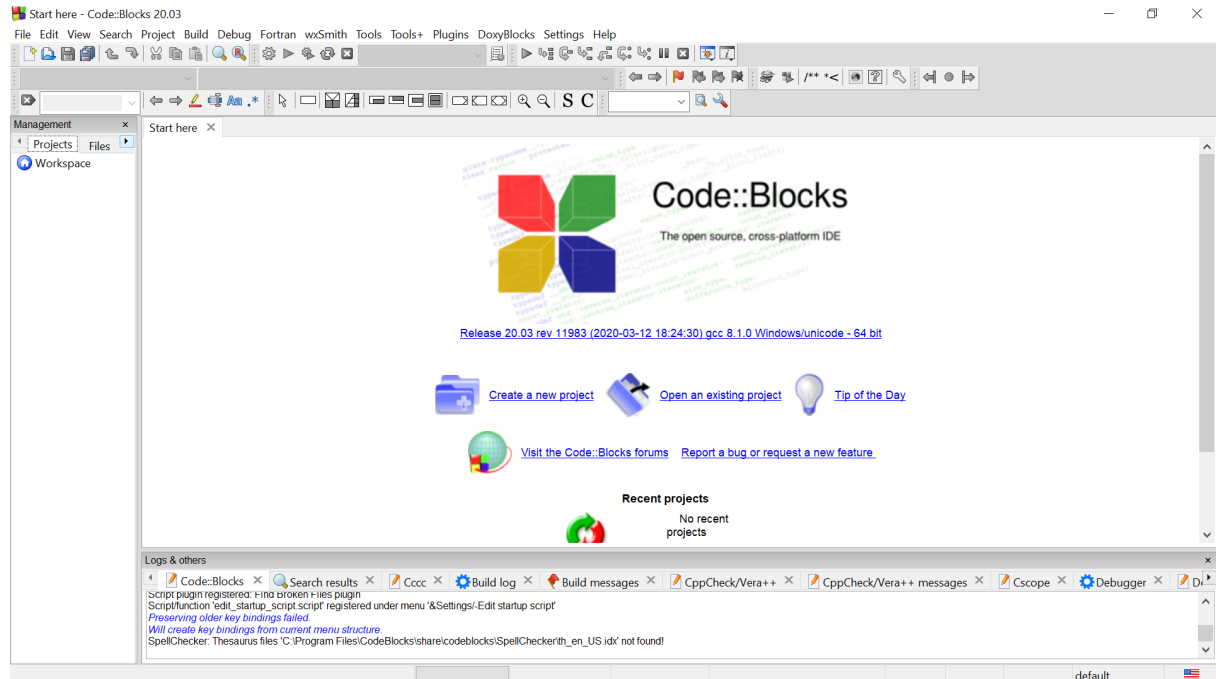
Windows XP / Vista / 7 / 8.x / 10:

File	Date	Download from
codeblocks-20.03-setup.exe	29 Mar 2020	FossHUB or Sourceforge.net
codeblocks-20.03-setup-nonadmin.exe	29 Mar 2020	FossHUB or Sourceforge.net
codeblocks-20.03-nosetup.zip	29 Mar 2020	FossHUB or Sourceforge.net
codeblocks-20.03mingw-setup.exe	29 Mar 2020	FossHUB or Sourceforge.net
codeblocks-20.03mingw-nosetup.zip	29 Mar 2020	FossHUB or Sourceforge.net
codeblocks-20.03-32bit-setup.exe	02 Apr 2020	FossHUB or Sourceforge.net
codeblocks-20.03-32bit-setup-nonadmin.exe	02 Apr 2020	FossHUB or Sourceforge.net
codeblocks-20.03-32bit-nosetup.zip	02 Apr 2020	FossHUB or Sourceforge.net
codeblocks-20.03mingw-32bit-setup.exe	02 Apr 2020	FossHUB or Sourceforge.net
codeblocks-20.03mingw-32bit-nosetup.zip	02 Apr 2020	FossHUB or Sourceforge.net

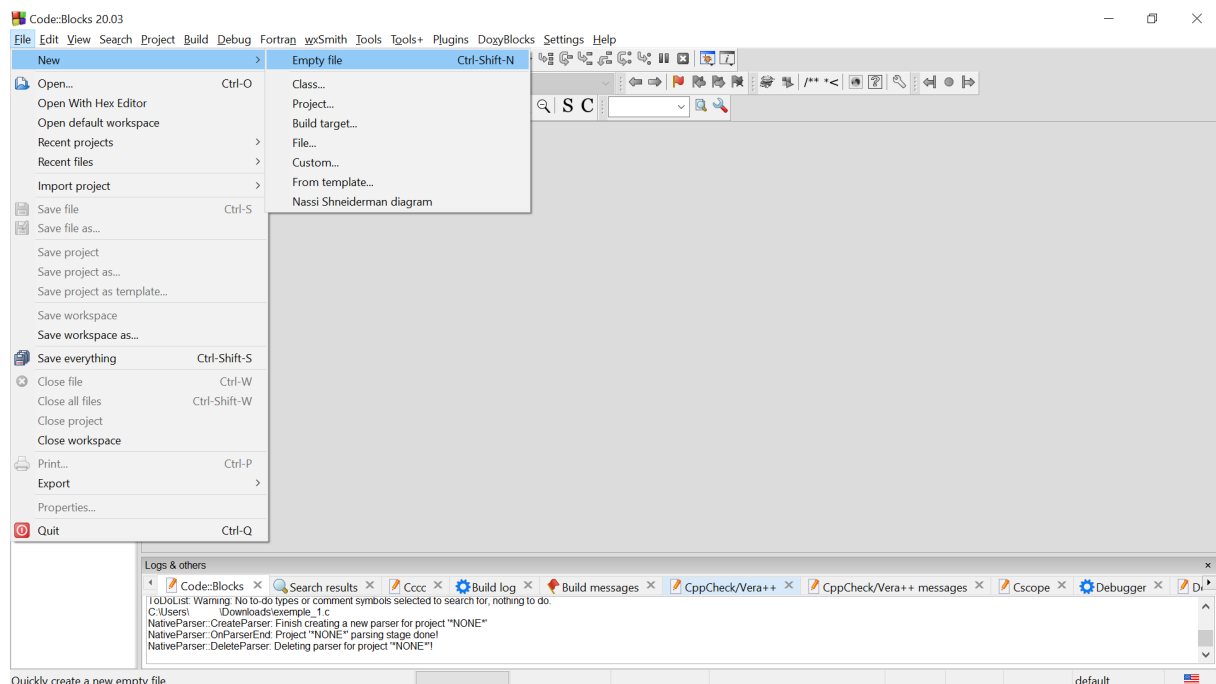
Une fois le logiciel est téléchargé, lancer l'installation.

2. Initiation à l'environnement :

Une fois l'installation est terminée, vous cliquez sur l'icône du logiciel pour le lancer. La première interface d'ouverture du logiciel est la suivante :



Pour créer votre premier programme, vous allez sur **File → New → Empty file**

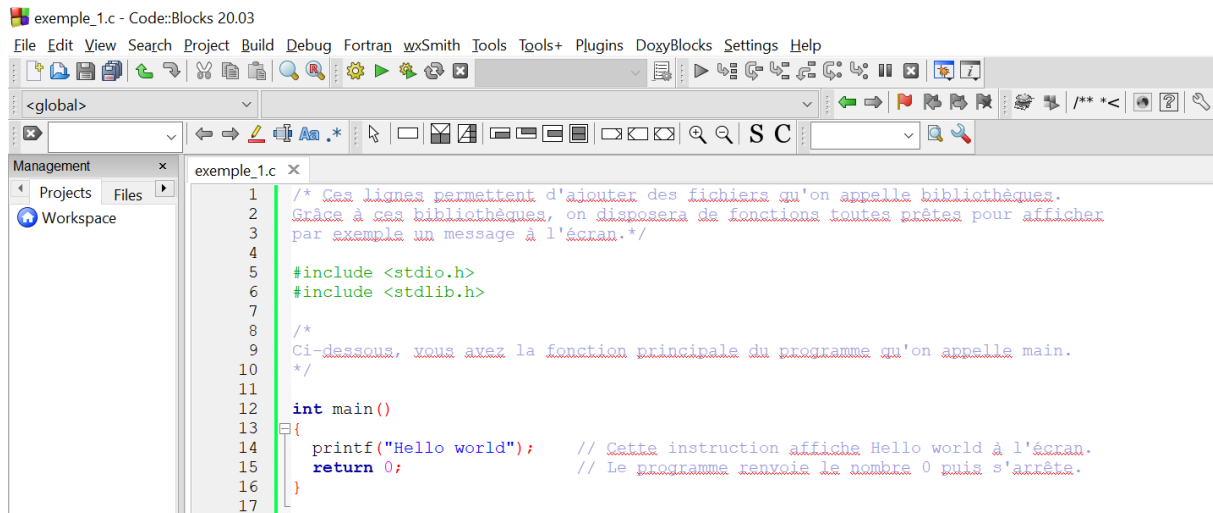


3. Prise en mains :

Exemple 1 : Création d'un premier programme

Le but de l'exercice est de s'initier à la syntaxe du langage C et ainsi écrire un programme qui affiche la phrase " Hello world "

Pour ce faire, nous allons ouvrir un nouveau fichier **File → New → Empty file**. Et nous allons taper le code source comme suit :

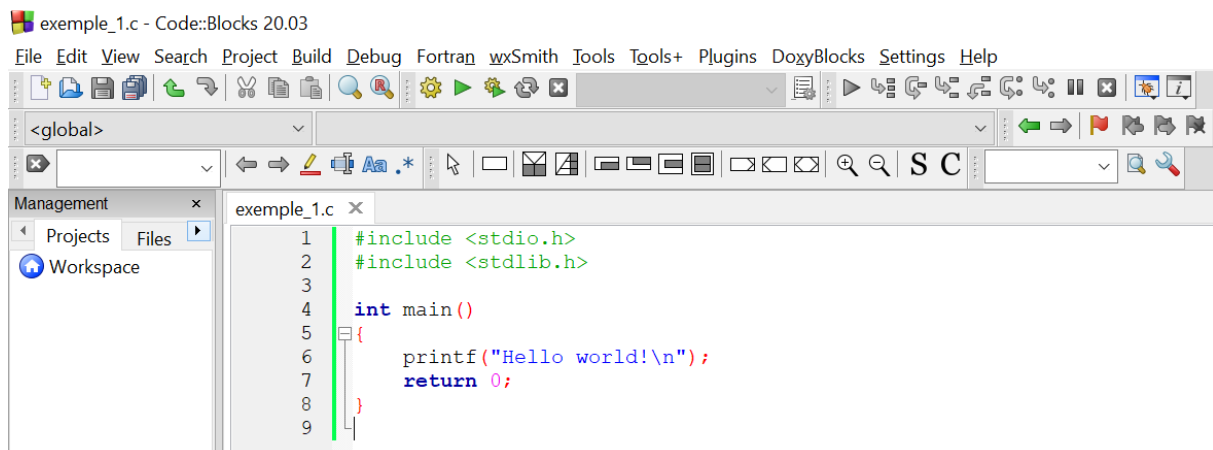


```

1  /* Ces lignes permettent d'ajouter des fichiers qu'on appelle bibliothèques.
2  Grâce à ces bibliothèques, on disposera de fonctions toutes prêtes pour afficher
3  par exemple un message à l'écran.*/
4
5  #include <stdio.h>
6  #include <stdlib.h>
7
8  /*
9  Ci-dessous, vous avez la fonction principale du programme qu'on appelle main.
10 */
11
12 int main()
13 {
14     printf("Hello world");    // Cette instruction affiche Hello world à l'écran.
15     return 0;                // Le programme renvoie le nombre 0 puis s'arrête.
16 }
17

```

Remarque : Dans l'exemple ci-dessus nous avons inséré des commentaires pour une meilleure compréhension. Et comme les commentaires n'ont aucun effet sur le code source, vous pouvez les omettre. Ce qui donne la version réduite suivante :



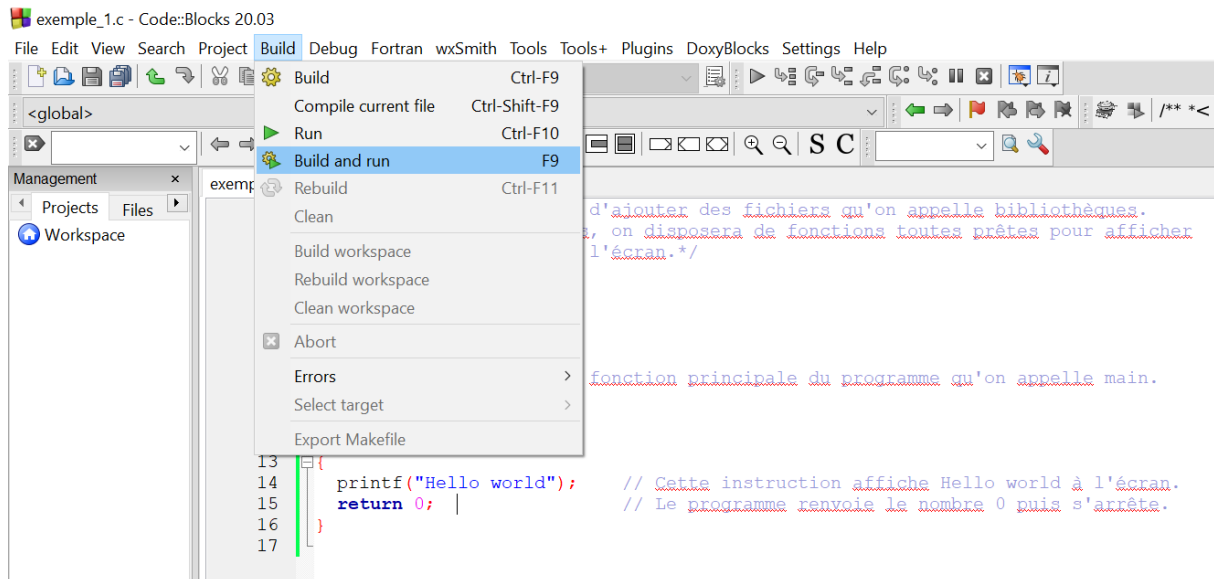
```

1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main()
5  {
6      printf("Hello world!\n");
7      return 0;
8  }
9

```

Sauvegardez ensuite votre programme en cliquant sur l'icône de *sauvegarde* ou bien **ctrl+s** en attribuant un nom à votre programme. Dans cet exemple, le nom attribué est *exemple_1*

Nous passons ensuite à compiler et exécuter le programme, en cliquant sur **Build** → **Build and run** :



Le résultat du programme après exécution est affiché dans la console, il s'agit d'un simple affichage de la phrase "Hello world" :

```

Hello world
Process returned 0 (0x0)   execution time : 0.263 s
Press any key to continue.

```

Exemple 2 : Création d'un deuxième programme

Dans cet exemple, le programme doit demander à l'utilisateur de fournir deux nombres entiers **a** et **b**. Ensuite, le programme doit calculer leur somme, leur produit et le résultat de la division de **a** sur **b**. Et finalement, afficher les résultats sur l'écran. On doit donc utiliser les instructions d'entrée / sortie (*printf* et *scanf*).

Le code source est le suivant :

```

exemple_2.c x
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main(int argc, char *argv[]) // équivalent de int main()
5  {
6      int a = 0, b = 0; // on initialise les variables a et b à 0
7      int som, prod;
8      float div;
9
10     printf("Saisir un nombre entier : ");
11     scanf("%d", &a); // scanf() est équivalente à lire() en algorithmique
12
13     printf("Saisir un autre nombre entier : ");
14     scanf("%d", &b);
15
16     som = a + b;
17     prod = a * b;
18     div = a / b;
19
20     printf("La somme de %d et %d est : %d\n", a, b, som); // %d indique qu'on affiche un entier
21     printf("Le produit de %d et %d est : %d\n", a, b, prod);
22     printf("La division de %d par %d est : %f\n", a, b, div); // %f indique qu'on affiche un réel
23
24     return 0;
25 }
26

```

Il ne reste qu'à compiler et exécuter le programme.

Attention : Il est souvent possible de commettre des erreurs de frappe et de syntaxe lors de l'écriture des codes sources. Si c'est le cas, une fois vous lancez la compilation, elle s'arrête en vous affichant des messages d'erreurs.

Exemple de code source contenant des erreurs de syntaxe :

L'erreur ici concerne le type prédéfini *float*.

L'éditeur vous indique qu'il y a une erreur dans votre programme et qu'il ne peut pas être compilé tel qu'il est avec l'erreur au niveau de la ligne 8 ayant le rectangle rouge, en vous proposant de la corriger.

```

exemple_2.c x
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main(int argc, char *argv[]) // équivalent de int main()
5  {
6      int a = 0, b = 0; // on initialise les variables a et b à 0
7      int som, prod;
8      floet div;
9
10     printf("Saisir un nombre entier : ");
11     scanf("%d", &a); // scanf() est équivalente à lire() en algorithmique
12
13     printf("Saisir un autre nombre entier : ");
14     scanf("%d", &b);
15
16     som = a + b;
17     prod = a * b;
18     div = a / b;
19
20     printf("La somme de %d et %d est : %d\n", a, b, som); // %d indique qu'on affiche un entier
21     printf("Le produit de %d et %d est : %d\n", a, b, prod);
22     printf("La division de %d par %d est : %f\n", a, b, div); // %f indique qu'on affiche un réel
23
24     return 0;
25 }
26

```

Logs & others

Code:Blocks x Search results x Cccc x Build log x Build messages x CppCheck/Vera++ x CppCheck/Vera++ messages x Cs

File	Line	Message
C:\Users\...		=== Build file: "no target" in "no project" (compiler: unkn...
C:\Users\...		In function 'main':
C:\Users\...	8	error: unknown type name 'floet'; did you mean 'float'?
C:\Users\...		=== Build failed: 1 error(s), 0 warning(s) (0 minute(s), 0

Une fois compilé, votre programme s'exécute, en vous demandant de saisir le premier entier **a**:

```
Saisir un nombre entier : _
```

Une fois saisi, le programme demande le deuxième entier **b**, (dans ce cas **a = 14** et **b = 5**).

Le programme ensuite calcule et affiche les résultats comme suit:

```
Saisir un nombre entier : 14
Saisir un autre nombre entier : 5
La somme de 14 et 5 est : 19
Le produit de 14 et 5 est : 70
La division de 14 par 5 est : 2.000000

Process returned 0 (0x0)   execution time : 7.702 s
Press any key to continue.
```

Travail à rendre :

Important : Les solutions des exercices du TP doivent être codées en langage C, compilées et exécutées. Ensuite, le travail à rendre sur la plateforme sera sous forme d'un **rapport** (*un fichier word ou pdf*) contenant pour chaque exercice **les captures d'écran du code source ainsi que celles des résultats après l'exécution**.

Remarque : Les exercices suivants ont été déjà traités en TD, il suffit juste de traduire les algorithmes en langage C.

Exercice 1:

Écrire un programme qui demande un nombre entier à l'utilisateur, puis qui calcule et affiche son carré.

Exercice 2 – Échange :

Écrire un programme qui demande à l'utilisateur 4 valeurs entières, qui les permute et les affiche ensuite.

Exercice 3 – Calcul de valeur d'une fonction :

Écrire un programme permettant de calculer la valeur de la fonction $f(x) = x^2 - 3x + 20$ pour un x donné.

Exercice 4 – Calcul de la remise (Conditions Simples):

Écrire un programme qui demande à l'utilisateur d'entrer le montant de la facture à payer. Si le montant est compris entre 400 dhs et 700 dhs, une remise de 2% est appliquée. Sinon, s'il dépasse 700 dhs, une remise de 4% est appliquée. Ensuite, le programme affiche le montant total à payer.

Exercice 5 – Conditions composées:

Écrire un programme qui demande à l'utilisateur deux entiers entrés au clavier, et qui l'informe ensuite si le produit des deux entiers est positif ou négatif sans calculer le produit.

Exercice 6 – Conditions imbriquées:

Écrire un programme qui demande à l'étudiant d'entrer trois notes et qui affiche:

- La décision 'Admis' si la moyenne est supérieure ou égale à 10, sinon affiche 'Non admis'.
- La mention selon la moyenne :
 - "Faible" si inférieure à 10.
 - "Passable" si comprise entre 10 et 12.
 - "Assez bien" si comprise entre 12 et 14.
 - "Bien" si comprise entre 14 et 16.
 - "Excellent" si supérieure à 16.

Exercice 7 – Structure à choix multiple:

Écrire un programme qui demande deux entiers et un opérateur arithmétique, et affiche ensuite l'opération ainsi que le résultat de l'opération indiquée.