Démo environnement de travail IFT1169

Ce document contient une série d'exercices permettant de vous familiariser avec les différents outils utilisés dans le cadre du cours IFT1169. Nous vous recommandons de faire les exercices dans l'ordre.

Exercice 1 : déterminer la version du compilateur g++.

Exercice 2 : compiler un programme « C++ » en ligne de commande.

Exercice 3 : configurer l'interface de développement codelite.

Exercice 4 : création d'un nouvel espace de travail puis d'un nouveau projet « C++ ».

Exercice 5 : création d'un projet « console wxWidgets ».

Exercice 6 : création d'un projet « interface graphique wxWidgets ».

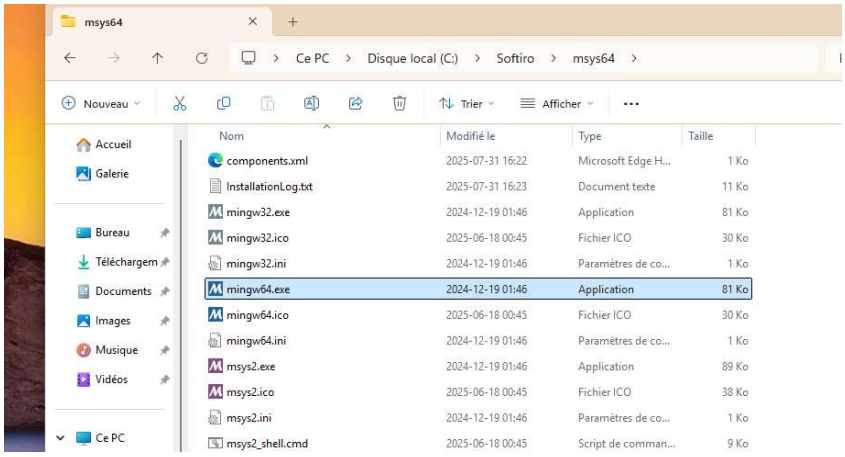
Exercice 7 : nettoyer/compiler tout l'espace de travail.

Exercice 8 : exemples du cours, chapitre « interfaces graphiques ».

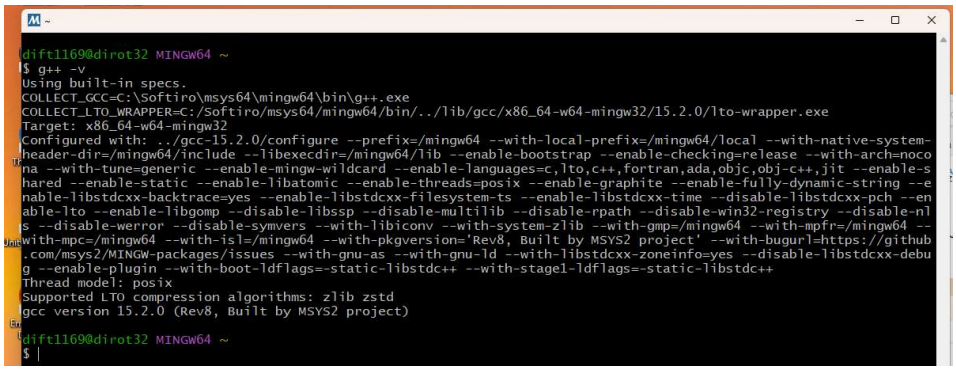
Exercice 1 : déterminer la version du compilateur g++.

Exécuter le programme mingw64 qui se trouve à cet endroit :

« C:\Softiro\msys64\mingw64.exe »



Dans la nouvelle fenêtre, exécuter la commande « g++ -v » :



La version du compilateur « g++ » installée sur une machine de la DESI est « 15.2 ».

La fenêtre « mingw » n'est qu'un terminal permettant d'exécuter une série de commandes.

Question : que va afficher en sortie la commande « ls », exécutée dans ce terminal?

Exercice 2 : compiler un programme « C++ » en ligne de commande.

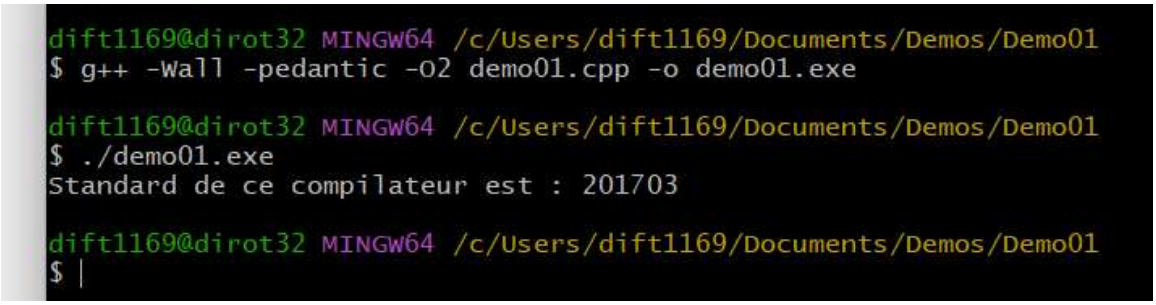
Sauvegarder ces lignes de code en C++ dans le fichier « demo01.cpp » :

```
#include <iostream>

using namespace std;

int main(){
	std ::cout << "Standard de ce compilateur est : "  << __cplusplus <<std ::endl;
	return 0;
}
```

En se positionnant dans le répertoire où se trouve votre fichier, lancer cette commande dans une fenêtre « mingw64 » :

g++ -Wall -pedantic -O2 demo01.cpp -o demo01.exe

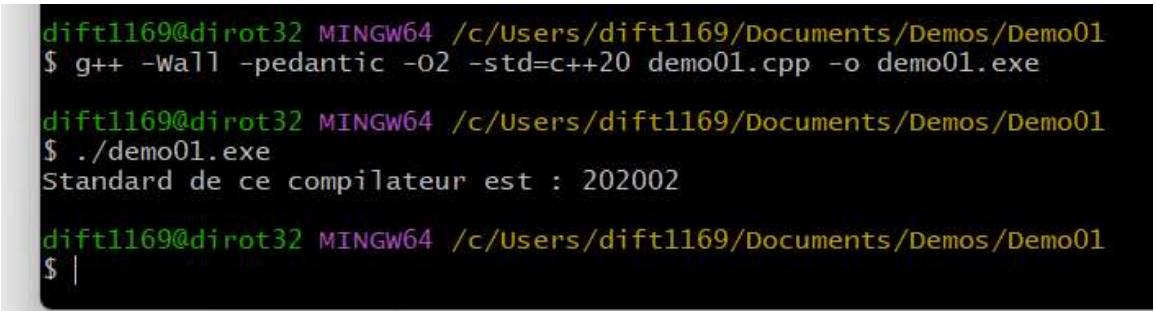
Exécuter par la suite le programme « demo01.exe ».



Ce compilateur utilise un standard normalisé en mars 2017. Il correspond à « C++17 ».

Nous allons maintenant forcer l'utilisation du standard « C++20 » à l'aide de l'option « std », comme suit :

```
g++ -Wall -pedantic -O2 -std=c++20 demo01.cpp -o demo01.exe
```

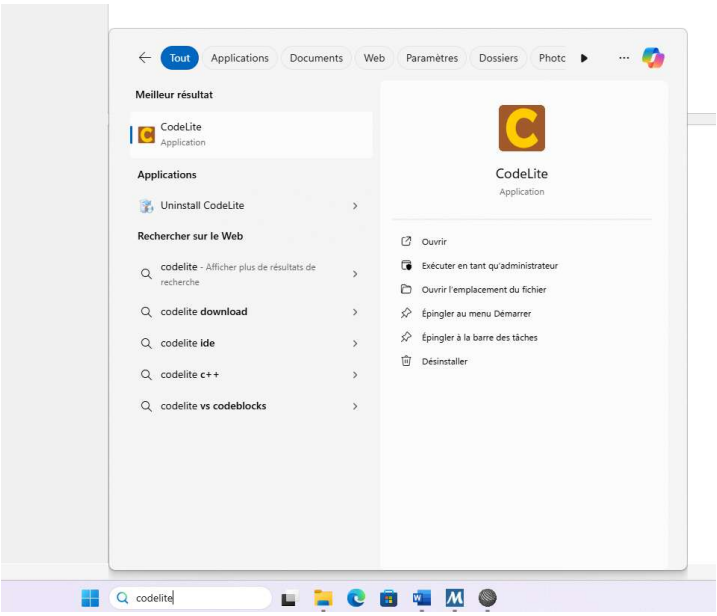


Ce compilateur utilise maintenant le standard normalisé en février 2020. Il correspond à « C++20 ».

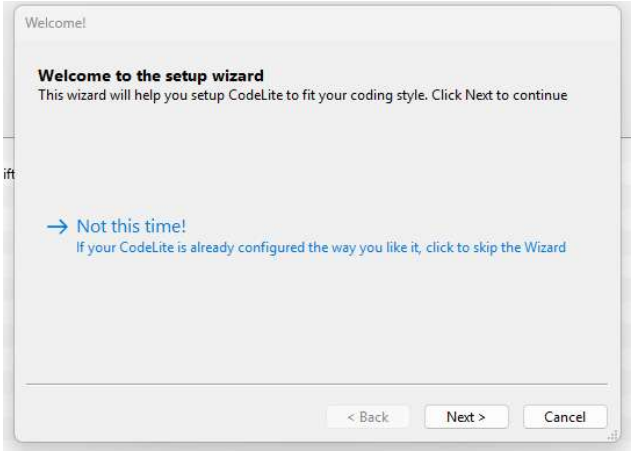
Question : quel mois de l'année 2023, le standard « C++23 » a-t-il été normalisé?

Exercice 3 : configurer l'interface de développement codelite.

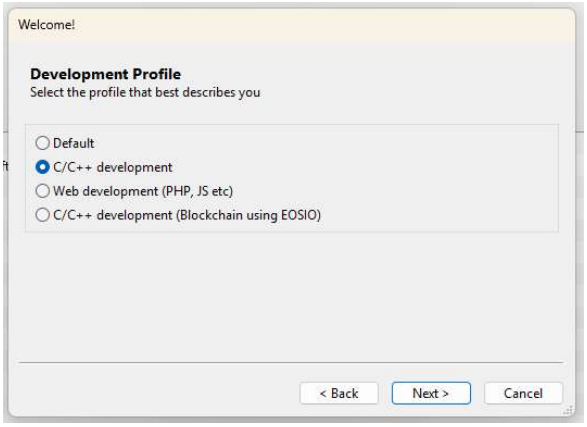
Dans la fenêtre de recherche de Windows, taper « codelite » :



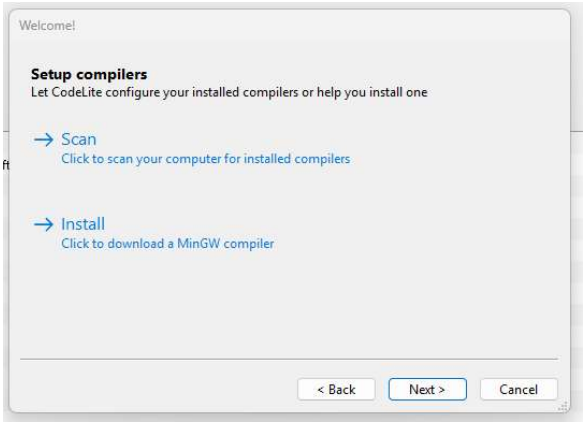
Lancer par la suite le programme « codelite ». Vous allez obtenir lors du premier lancement cette fenêtre d'accueil :



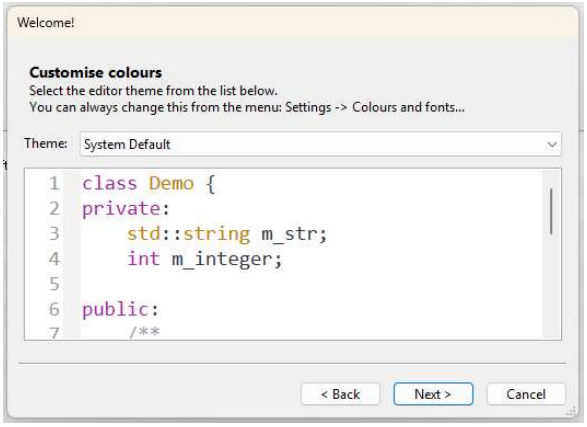
Cliquer sur « Next » …



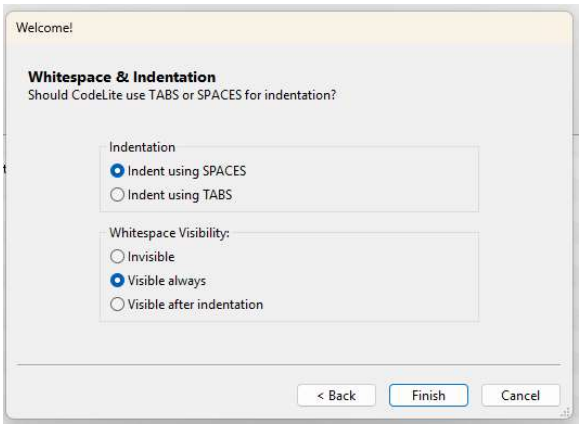
Sélectionner « c/c++ development », puis cliquer sur « Next » …



Comme le compilateur a déjà été configuré sur les postes de la DESI, nous pouvons sauter cette étape. Cliquer sur « Next » …



Sélectionner le mode d'affichage puis cliquer sur « Next » …



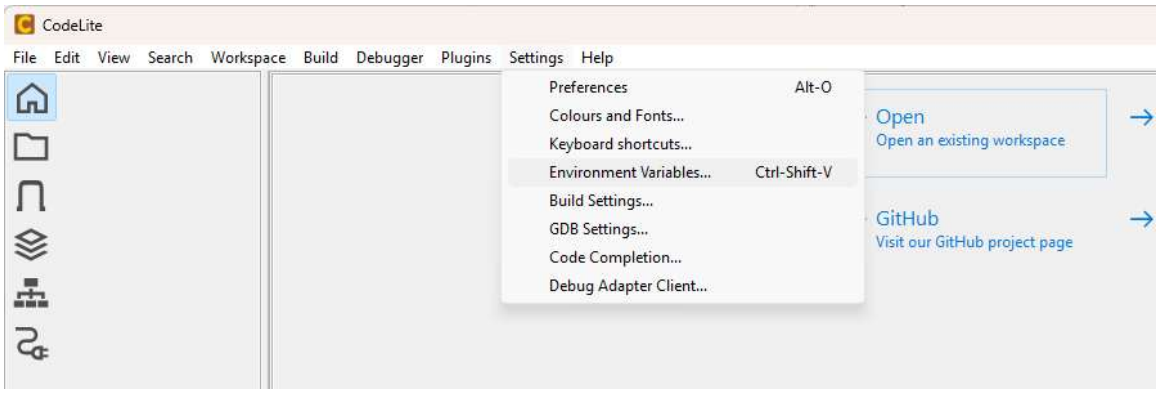
Sélectionner les options désirées, puis cliquer sur « Finish ».

Codelite va redémarrer afin d'activer les options choisies.

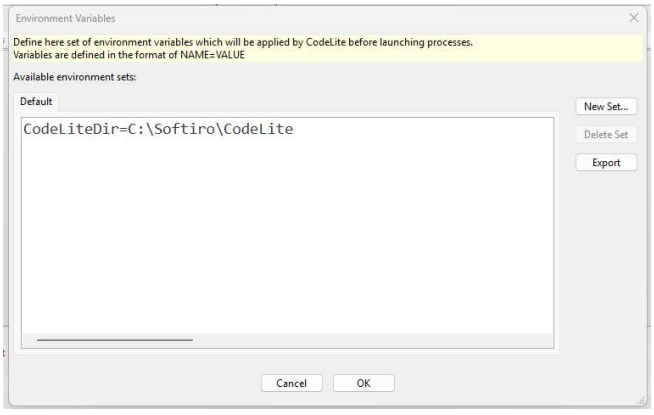
Question : comment peut-on relancer ce configurateur une nouvelle fois à partir de l'interface de codelite?

Il reste une dernière étape pour que codelite soit complètement opérationnel sur les postes de la DESI.

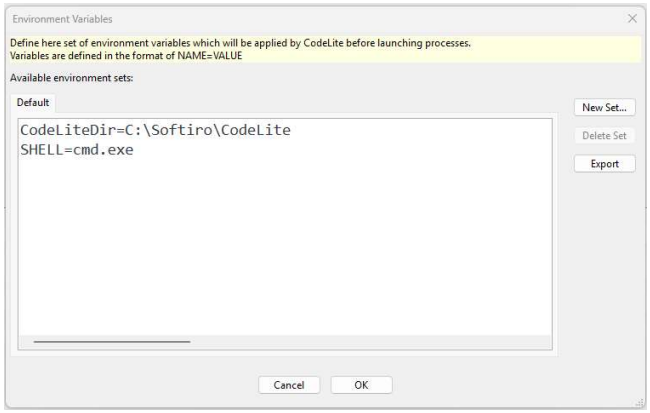
Dans le menu de codelite, sélectionner « Settings », puis « Environment Variables », comme suit :



Vous allez obtenir cette fenêtre :



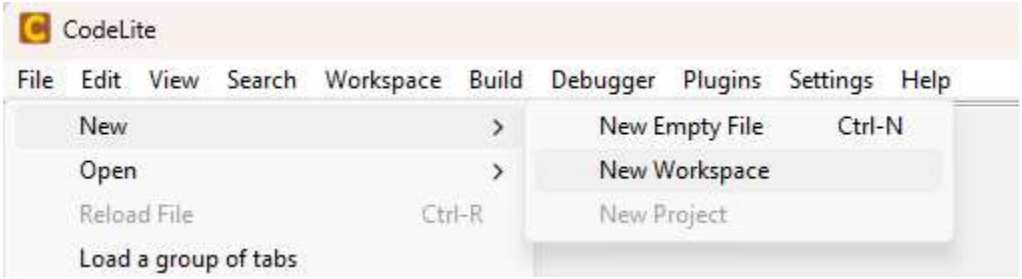
Ajouter cette variable « SHELL=cmd.exe », puis cliquer sur « OK » :



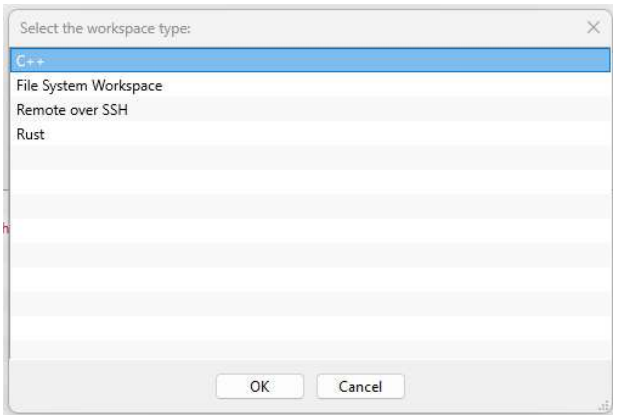
Codelite est maintenant complètement configuré.

Exercice 4 : création d'un nouvel espace de travail puis d'un nouveau projet « C++ ».

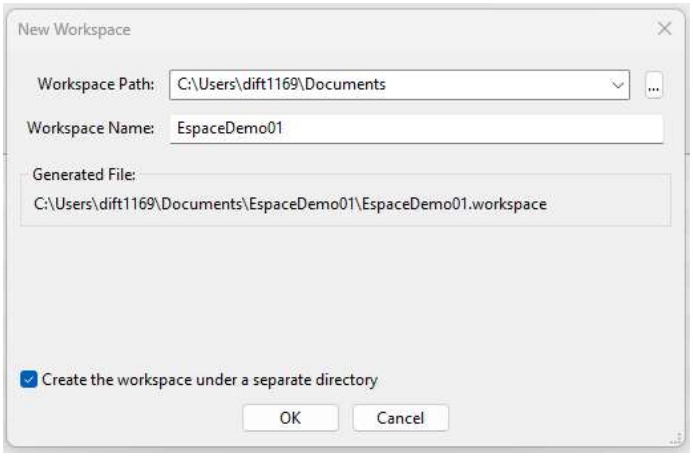
Cliquer sur « File », « New », « New Workspace », comme suit :



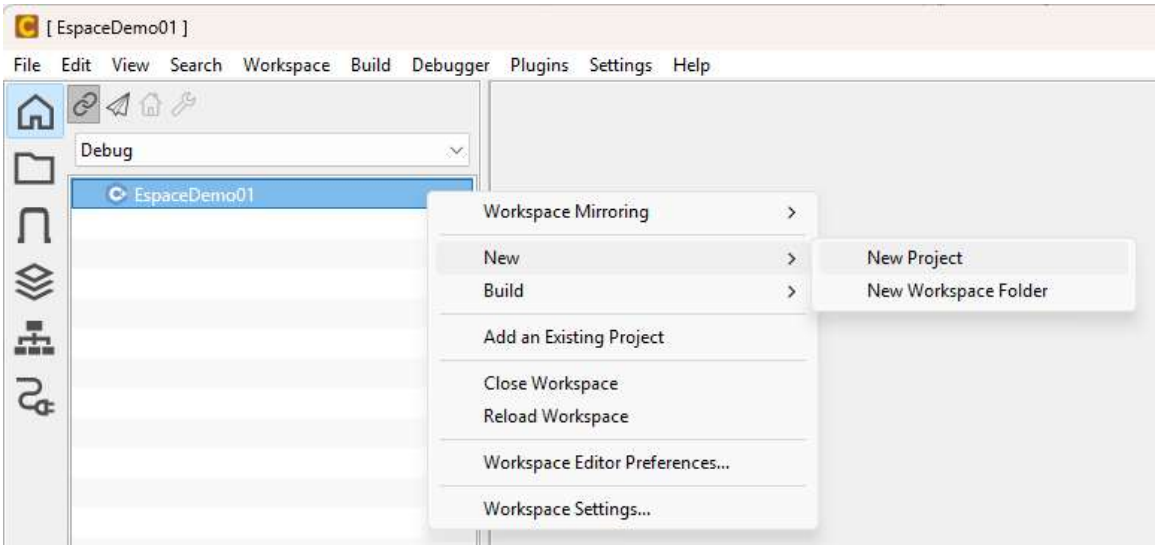
Vous allez obtenir la fenêtre ci-dessous. Vous allez sélectionner « C++ », puis cliquer sur « OK » :



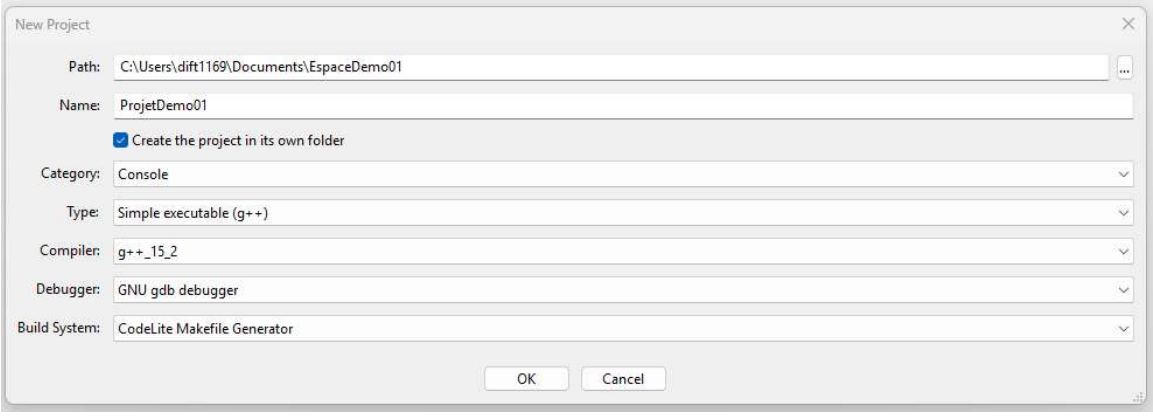
Choisir un nom et un emplacement pour l’espace de travail, puis cliquer sur « OK » :



Nous allons créer maintenant un nouveau projet dans cet espace de travail. Cliquer sur l’espace de travail avec le bouton droit de la souris, et sélectionner « New », puis « New Project », comme suit :

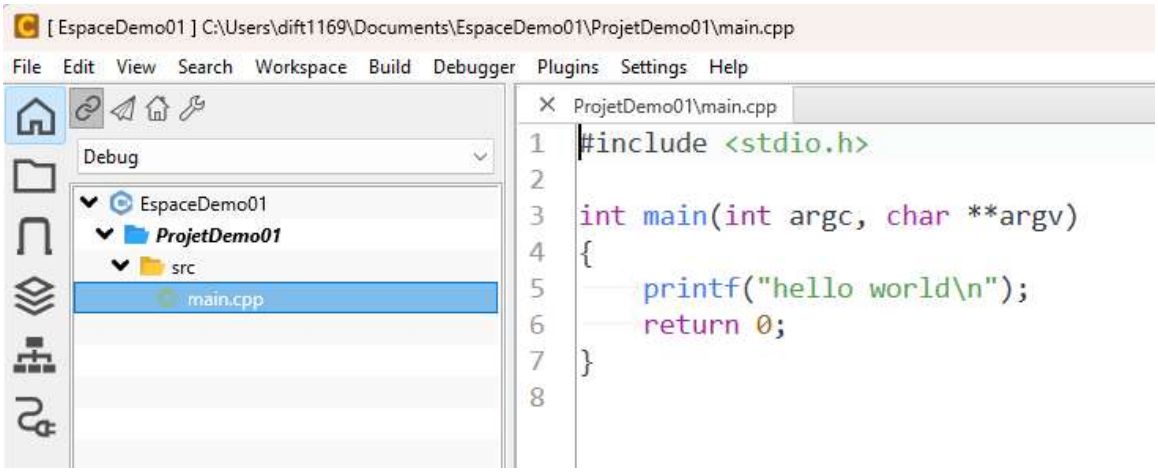


Donner un nom à votre projet et ajuster les différents paramètres comme suit :



Nous avons décidé de créer le projet dans un dossier séparé pour une meilleure fluidité. Le programme est un code en « C++ » qui sera exécuté dans une console régulière. Nous allons utiliser le compilateur « g++_15_ » installé sur les postes de la DESI.

Après avoir cliqué sur « OK » dans la précédente fenêtre, vous allez obtenir ce résultat :



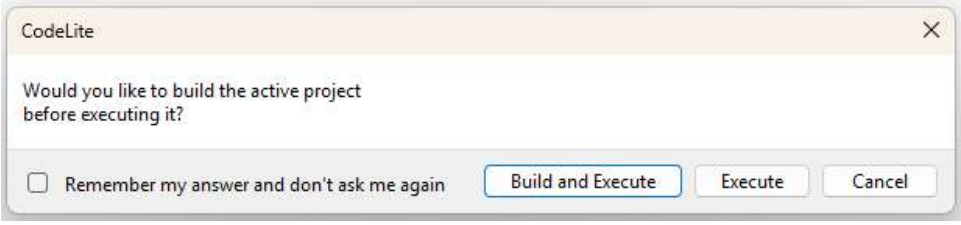
Un programme par défaut « main.cpp » a été créé. Modifier les lignes 1 et 5 pour le rendre conforme au langage C++, en utilisant à la place « iostream » et « std::cout ».

Après avoir sauvegardé ces modifications, cliquer sur F7 pour compiler ce projet ou bien à partir du menu « Build, Build Project ».

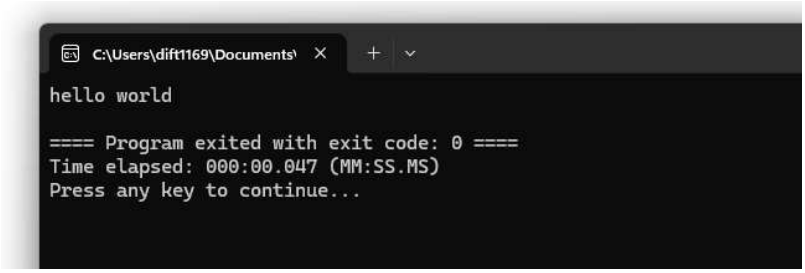
Normalement, vous allez obtenir ce résultat :

=== build completed successfully (0 errors, 0 warnings) ===

Exécuter maintenant le programme en cliquant sur les touches CTRL-F5 ou bien à partir du menu « Build, Run ». Vous allez obtenir cette fenêtre :



Vous avez le choix de recompiler le projet, puis de l'exécuter ou l'exécuter directement. Comme, le projet a été compilé lors de la précédente étape, il n'est pas nécessaire de le compiler une deuxième fois. L'exécution du programme va donner ce qui suit :



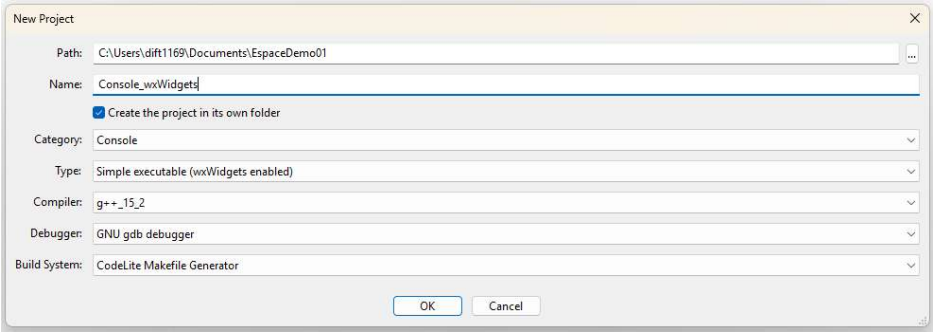
Le programme s'est exécuté correctement. Appuyer sur n'importe quelle touche pour quitter la console.

Question : refaire cet exercice avec le programme de l'exercice 2. L'idée est de créer un nouveau projet en utilisant directement le fichier « demo01.cpp ». Vous n'allez pas faire un copier-coller d'un contenu!

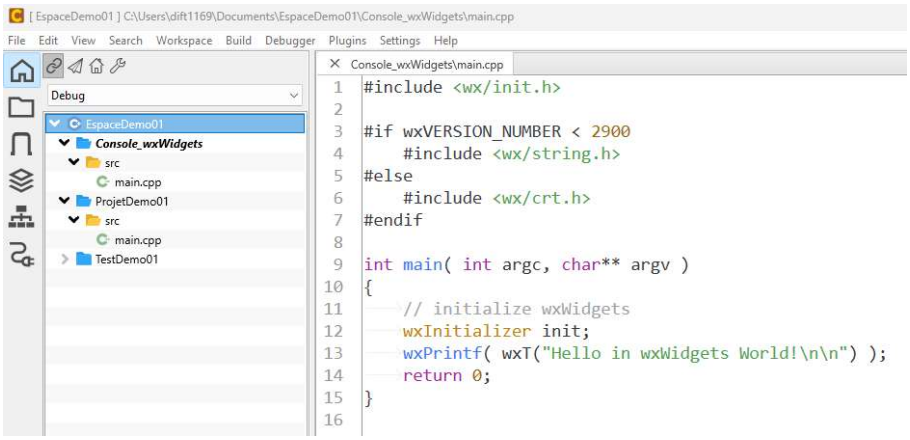
Par la suite, chercher dans codelite comment activer la version du standard souhaitée (l'équivalent de l'option -std).

Exercice 5 : création d'un projet « console wxWidgets ».

Pour ce projet, nous allons créer un programme console en utilisant la librairie wxWidgets. Nous allons refaire les premières étapes de l'exercice 4, mais avec les paramètres suivants :

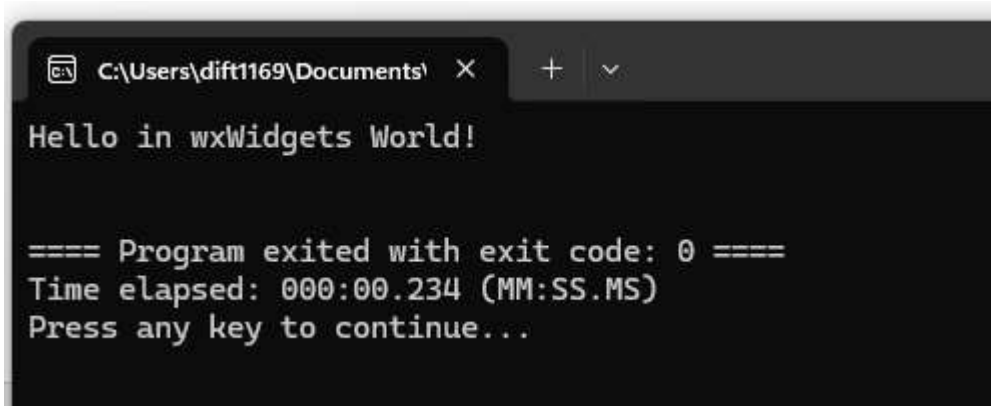


Ce qui change est le nom du projet et le type. Nous avons choisi « Simple executable (wxWidgets enabled »). Cliquer par la suite sur « OK ».



Il faut faire un double clic sur le nom du projet pour s'assurer qu'il est sélectionné (en gras). Compiler, puis exécuter le programme, comme dans l'exercice 4.

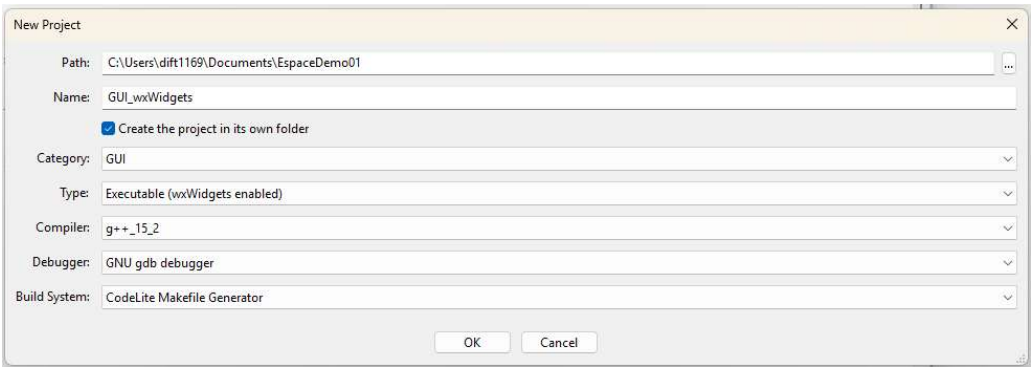
Si vous avez suivi correctement ces étapes, vous allez obtenir ce résultat :



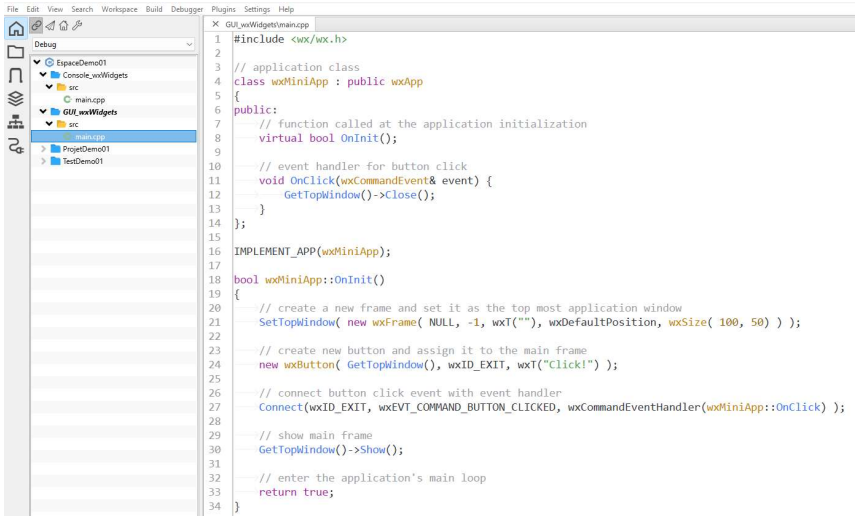
Le tout fonctionne correctement!

Exercice 6 : création d'un projet « interface graphique wxWidgets ».

Pour ce projet, nous allons créer une interface graphique de base en utilisant la librairie wxWidgets. Nous allons refaire les premières étapes de l'exercice 5, mais avec les paramètres suivants :

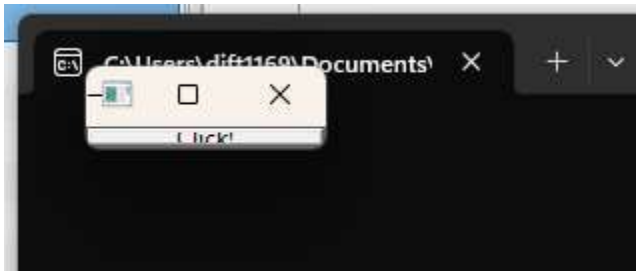


Nous avons choisi un nom approprié pour le projet, puis nous avons modifié la catégorie pour avoir « GUI ». Le type reste le même « Simple executable (wxWidgets enabled ») . Cliquer par la suite sur « OK ».



```cpp
#include <wx/wx.h>

// application class
class wxMiniApp : public wxApp
{
public:
    // function called at the application initialization
    virtual bool OnInit();

    // event handler for button click
    void OnClick(wxCommandEvent& event) {
        GetTopWindow()->Close();
    }
};

IMPLEMENT_APP(wxMiniApp);

bool wxMiniApp::OnInit()
{
    // create a new frame and set it as the top most application window
    SetTopWindow( new wxFrame( NULL, -1, wxT(""), wxDefaultPosition, wxSize( 100, 50) ) );

    // create new button and assign it to the main frame
    new wxButton( GetTopWindow(), wxID_EXIT, wxT("Click!") );

    // connect button click event with event handler
    Connect(wxID_EXIT, wxEVT_COMMAND_BUTTON_CLICKED, wxCommandEventHandler(wxMiniApp::OnClick) );

    // show main frame
    GetTopWindow()->Show();

    // enter the application's main loop
    return true;
}
```

Il faut s'assurer d'avoir sélectionné le bon projet avant de le compiler. Exécuter le projet par la suite. Vous allez obtenir ce résultat :

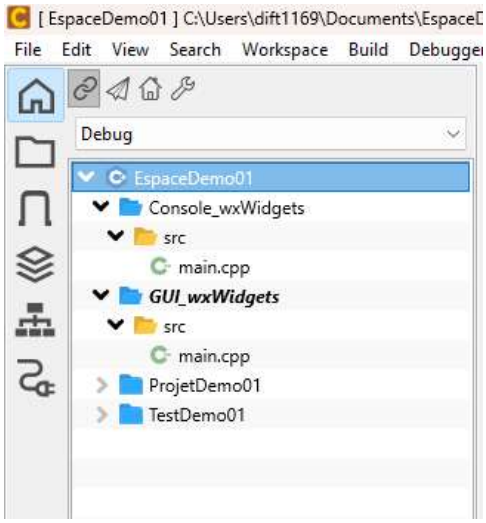


Vous pouvez élargir la fenêtre de l'interface afin de cliquer sur le bouton « Click! ».

Question : modifier le programme pour avoir une interface plus large afin de bien visualiser le bouton « Click! », puis recompiler et relancer le programme.

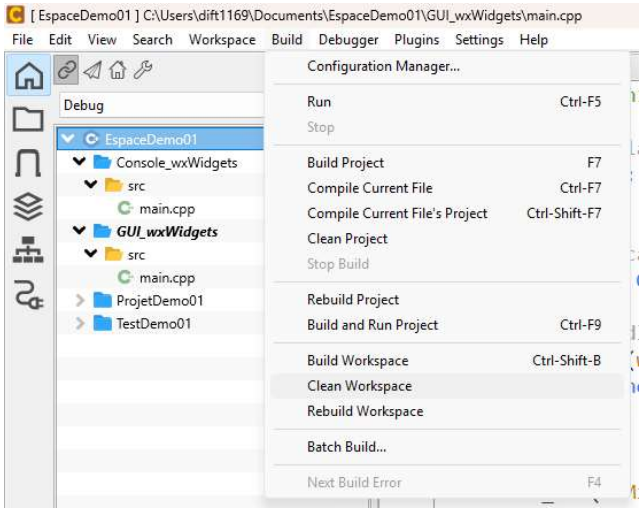
Exercice 7 : nettoyer/compiler tout l'espace de travail.

L'espace de travail contient plusieurs projets :

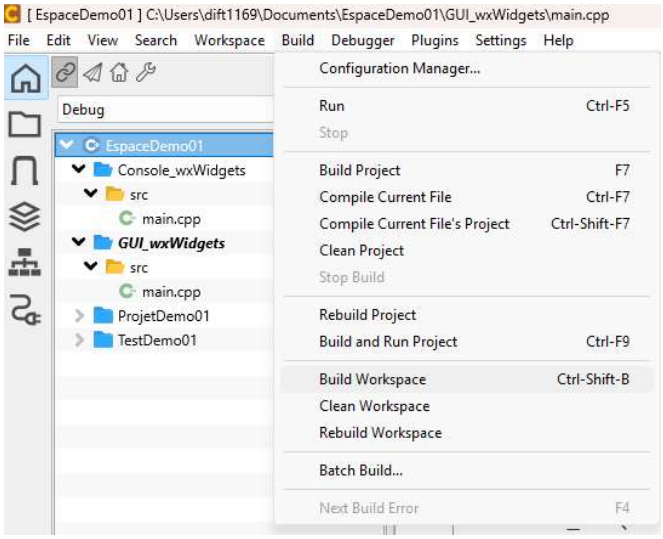


Codelite offre la possibilité d'effacer les exécutables de tous les projets afin de les recompiler une nouvelle fois. Il offre aussi la possibilité de compiler tous les projets d'un seul coup. Par contre, on ne peut exécuter qu'un programme à la fois.

Nettoyer tous les projets dans l'espace (Build, Clean Workspace) :



Construire tous les projets dans l'espace (Build, Build Workspace) :



Question : quelle est l'utilité de la commande « Build, Rebuild Workspace »?

Exercice 8 : exemples du cours, chapitre « interfaces graphiques ».

Refaire l’exercice 7 sur les exemples du chapitre du cours sur les interfaces graphiques.