How config MSVC Compiler (32 and 64 bits) into CodeBlocks

Full name of tutorial : How config MSVC Compiler (32 and 64 bits) into Code::Blocks on Windows 11 64 bits.

```
Code::Blocks : the best and great free IDE for Windows, Linux and ... Mac OS
```

Presentation of MSVC compiler (if needed ...)

During first run of CB on Windows, this IDE detect automatically some compilers, or present one list of them pre-configured.
It's very good functionnality, but, sometimes, you must "force" these configurations proposed by default to run correctly.
This tuto describe how configure one major compiler C\C++ on Windows : MSVC Compiler (32 and 64 bits).

How to install MSVC Compiler (32 and 64 bits) ?

This compiler is available on site Microsoft, it's included in package Visual Studio 2022 Community by example :
https://visualstudio.microsoft.com/fr/vs/community/ (community, only "free" version available)

But it's not enought, because you must donwload also last SDK for Windows (needed because contain all include and libraries files) :
https://developer.microsoft.com/fr-fr/windows/downloads/windows-sdk/

After download clic on these files to install resuired components on your system. It's recommended to run "Visual Studio Installer" to verify last available version, but also to install "Visual Studio Build Tools 2022"

Configuration of MSVC compiler C/C++ into CB

First, you must access into IDE CB at menu "Settings" and submenu "Compilers" to select "Visual Studio 2022", normally autodetected by IDE.
To distinguish version 32 bits or 64 bits into CB, you must rename this ident of compiler in

"Visual Studio 2022 (64 bits)" (click on button Rename, simply).
An after, you must verify field "Compiler's installation directory" that must contain "C:\Program Files\Microsoft Visual Studio\2022\Community\VC\Tools\MSVC\14.40.33807\bin\Hostx64\x64". (by example)

You must verify list of tools like this (in tab "Program Files") :
- C Compiler : cl.exe
- C++ Compiler : cl.exe
- Linker for dynamic libs : link.exe
- Linker for statics libs : lib.exe
- Debugger :
- Resource compiler : rc.exe (this included in SDK Windows 11)
- Make program : nmake.exe (this included in Visual Studio Build Tools 2002)

After, you must add in tab "Additional Paths" :
- "C:\Program Files (x86)\Windows Kits\10\bin\10.0.22621.0\x64" (by example)
- and : C:\Program Files (x86)\Windows Kits\10\bin\x64
- and : C:\Program Files\Microsoft Visual Studio\2022\Community\MSBuild\Current\Bin\amd64
It's mandatory to access at "rc.exe" and "nmake".

WARNING : After update, all number included into name of directories of MSVC/SDK can change !!!
To integrate these evolutions, it's seem rational to define environment variables on system like this :
- VS_VERSION define to 2022 (at date)
- VS_NUM define to 14.40.33807 (at date) and
- KIT_VERSION define to 10 (at date)
- KIT_NUM define to 10.0.22621.0 (at date)
Then, you can use these variables into configuration of CB with use of %var% into directory name to be independant of evolutions.

And, it's not all, you must verify in tab "Search Directories" and select "Compiler", or "Linker" or "Resource Compiler" subtabs.

For compiler, search directories of "include files" are (and it's the same for "Resource Compiler") :
- C:\Program Files\Microsoft Visual Studio%VS_VERSION%\Community\VC\Tools\MSVC%VS_NUM%\include
- C:\Program Files\Microsoft Visual Studio%VS_VERSION%\Community\VC\Auxiliary\VS\include
- C:\Program Files (x86)\Windows Kits%KIT_VERSION%\Include%KIT_NUM%\ucrt

- C:\Program Files (x86)\Windows Kits%KIT_VERSION%\Include%KIT_NUM%\um
- C:\Program Files (x86)\Windows Kits%KIT_VERSION%\Include%KIT_NUM%\shared
- C:\Program Files (x86)\Windows Kits%KIT_VERSION%\Include%KIT_NUM%\winrt
- C:\Program Files (x86)\Windows Kits%KIT_VERSION%\Include%KIT_NUM%\cppwinrt

For linker, search directories of "lib files" are :
- C:\Program Files\Microsoft Visual Studio%VS_VERSION%\Community\VC\Tools\MSVC%VS_NUM%\lib\x64
- C:\Program Files (x86)\Windows Kits%KIT_VERSION%\Lib%KIT_NUM%\ucrt\x64
- C:\Program Files\Microsoft Visual Studio%VS_VERSION%\Community\VC\Tools\MSVC%VS_NUM%\lib\x64\store
- C:\Program Files (x86)\Windows Kits%KIT_VERSION%\Lib%KIT_NUM%\um\x64

Remark : If you wan't use environment variables, you can translate these by "real value" of directory into CB.

To terminate with 64 bits version of MSVC, you must add in tab "Linker settings" and in zone "Other linker options" :"/MACHINE:X64". MSVC must position automatically this option, but it's a precaution.

It's some tedious, but with these "full" configurations, you can build an program on Windows 11 64 bits with success.

Now, you must copy "Visual Studio 2022 (64 bits)" in list of available compilers into CB (menu "Settings" submenu "Compilers"), and rename it by "Visual Studio 2022 (32 bits)" by example.

New configurations for this compiler in version 32 bits are next :

In tab "Toolchain executable", you must change value by :
C:\Program Files\Microsoft Visual Studio%VS_VERSION%\Community\VC\Tools\MSVC%VS_NUM%\bin\Hostx86\x86

In subtab "Additional Paths" :
- C:\Program Files (x86)\Windows Kits%KIT_VERSION%\bin%KIT_NUM%\x86
- C:\Program Files (x86)\Windows Kits%KIT_VERSION%\bin\x86
- C:\Program Files\Microsoft Visual Studio%VS_VERSION%\Community\MSBuild\Current\Bin\amd64

In tab "Search Paths", only changes are mandatory in subtab "Linker" ("include" directory files are same) :
- C:\Program Files\Microsoft Visual Studio%VS_VERSION%\Community\VC\Tools\MSVC%VS_NUM%\lib\x86
- C:\Program Files (x86)\Windows Kits%KIT_VERSION%\Lib%KIT_NUM%\ucrt\x86

- C:\Program Files\Microsoft Visual Studio%VS_VERSION%\Community\VC\Tools\MSVC%VS_NUM%\lib\x86\store
- C:\Program Files (x86)\Windows Kits%KIT_VERSION%\Lib%KIT_NUM%\um\x86

And, to terminate, you must "force" an option in tab "Linker Settings" to add in zone "Other linker options" : "/MACHINE:X86".
MSVC must position automatically this option, but it's a precaution.

Test of "simple" code with MSVC compiler C/C++ into CB

With simply source "hellowworld.c", you can test generation of program into IDE CB, choosing "create new project" in main windows of CB, and choose "console application" with no source proposed by default, because named "main.c" by default, and choose compiler "Visual Studio 2022 (64 bits)".
You can select good directory/source with option "add file" after first creation of project into CB.

One time project created, you can generate it with selecting main menu "Build" and choose submenu "Rebuild..." (or CTRL-F11), ans save project file (or "save everythings").

To test 32 bits, return into main menu "Project" and in submenu "Build options" to change the compiler to "Visual Studio 2022 (32 bits)", and rebuild with CTRL-F11.

If, you apply all of precedent instructions, compile and link of your program must be succeeded.

Pleasure of programming is open for you, your imagination is illimited, at your keyboard ! Enjoy !

PS : source file "hellowworld.c" :

/ *Basic example in language C : hellowworld.c* /

```
#include <stdio.h>

int main(int argc, char argv[]) {
/ printf() displays the string inside quotation */
printf("Hello, World!");
return 0;
}
```

PS2 : Use of MSVC compiler with command line (just to illustrate)

You can also use MSVC compiler in command console on Windows (CMD.EXE) with next instructions :

```
set PATHSAV=%PATH%
set INCSAV=%INCLUDE%
set LIBSAV=%LIB%
set PATH=C:\Program Files\Microsoft Visual Studio%VS_VERSION%\Community\VC\Tools\MSVC%VS_NUM%\bin\Hostx64\x64;C:\Program Files (x86)\Windows Kits%KIT_VERSION%\bin%KIT_NUM%\x64;%PATH%
set PATH=C:\Program Files (x86)\Windows Kits%KIT_VERSION%\bin\x64;C:\Program Files\Microsoft Visual Studio%VS_VERSION%\Community\MSBuild\Current\Bin\amd64;%PATH%
set INCLUDE=C:\Program Files\Microsoft Visual Studio%VS_VERSION%\Community\VC\Tools\MSVC%VS_NUM%\include;C:\Program Files\Microsoft Visual Studio%VS_VERSION%\Community\VC\Auxiliary\VS\include;C:\Program Files (x86)\Windows Kits%KIT_VERSION%\Include%KIT_NUM%\ucrt;%INCLUDE%
set INCLUDE=C:\Program Files (x86)\Windows Kits%KIT_VERSION%\Include%KIT_NUM%\um;C:\Program Files (x86)\Windows Kits%KIT_VERSION%\Include%KIT_NUM%\shared;C:\Program Files (x86)\Windows Kits%KIT_VERSION%\Include%KIT_NUM%\winrt;%INCLUDE%
set LIB=C:\Program Files\Microsoft Visual Studio%VS_VERSION%\Community\VC\Tools\MSVC%VS_NUM%\lib\x64;C:\Program Files\Microsoft Visual Studio%VS_VERSION%\Community\VC\Tools\MSVC%VS_NUM%\lib\x64\store
set LIB=C:\Program Files (x86)\Windows Kits%KIT_VERSION%\Lib%KIT_NUM%\ucrt\x64;C:\Program Files (x86)\Windows Kits%KIT_VERSION%\Lib%KIT_NUM%\um\x64;%LIB%

REM Compile + link in one pass
cl hellowworld.c /Fe:hellowworld.exe
REM Compile + link in two pass
cl /c hellowworld.c /Fo:hellowworld.obj
link hellowworld.obj /OUT:hellowworld.exe /MACHINE:X64 /SUBSYSTEM:CONSOLE
```

Continue with use of MSVC compiler, and don't forgive, at the end of your work, to return to initial state :

```
set PATH=%PATHSAV%
set INCLUDE=%INCSAV%
```

set LIB=%LIBSAV%

And, with precedent example, you can also generate version 32 bits with "cl" but, you must change values of PATH, INCLUDE and LIB with good directories described before (don't forgive to change "/MACHINE:X86" during call of "link" if two pass).

But, it's much easy to use MSVC compiler C/C++ directly into CB IDE especially with complex C program (many C/C++ sources and many subdirectories ...) , and multiple targets by example : main DLL and console program to test this DLL, ... -)

PS3 : Syntax of tools MSVC

Syntax of MSVC compiler "cl" on command line is next :

cl /help
Compilateur d'optimisation Microsoft (R) C/C++ version 19.40.33813 pour x64
Copyright (C) Microsoft Corporation. Tous droits réservés.

```
                        OPTIONS DU COMPILATEUR C/C++


                             -OPTIMISATION-
```

/O1 optimisations maximales (favoriser l'espace)
/O2 optimisations maximales (favoriser la vitesse)
/Ob expansion inline (n=0 par défaut)
/Od désactiver les optimisations (par défaut)
/Og activer l'optimisation globale
/Oi[-] activer les fonctions intrinsèques
/Os privilégier l'espace du code /Ot privilégier la vitesse du code
/Ox optimisations (favoriser la vitesse)
/favor:<blend|AMD64|INTEL64|ATOM> sélectionner le processeur à optimiser, au choix :
blend - combinaison d'optimisations pour plusieurs processeurs x64 différents
AMD64 - processeurs AMD 64 bits
INTEL64 - processeurs d'architecture Intel(R)64
ATOM - processeurs Intel(R) Atom(TM)

```
                           -GÉNÉRATION DE CODE-
```

/Gu[-] permet de vérifier que les fonctions distinctes ont des adresses distinctes
/Gw[-] séparer les variables globales pour l'Éditeur de liens
/GF activer le regroupement des chaînes en lecture seule
/Gy[-] séparer les fonctions pour l'Éditeur de liens
/GS[-] activer les vérifications de la sécurité
(appuyez sur <entrée> pour continuer)
/GR[-] activer C++ RTTI
/guard:cf[-] activer CFG (protection du flux de contrôle)
/guard:ehcont[-] activer les métadonnées de continuation EH (CET)
/EHs activer C++ EH (sans exception SEH)
/EHa activer C++ EH (avec exceptions SEH)
/EHc extern " C" a pour valeur par défaut nothrow
/EHr génère toujours des contrôles d'arrêt de runtime noexcept
/fp:<contract|except[-]|fast|precise|strict> choisir le modèle à virgule flottante :
contract - prend en compte les contractions en virgule flottante lors de la génération de code.
except[-] - prendre en considération les exceptions de virgule flottante lors de la génération du code
fast - modèle de virgule flottante « rapide » ; les résultats sont moins prévisibles
precise - modèle de virgule flottante « précise » ; les résultats sont prévisibles
strict - modèle de virgule flottante « stricte » (implique /fp:except)
/Qfast_transcendentals générer des intrinsèques FP inline même avec /fp:except
/Qspectre[-] activer les atténuations pour CVE 2017-5753
/Qpar[-] activer la génération de code parallèle
/Qpar-report:1 diagnostic du paralléliseur automatique ; indiquer les boucles parallélisées
/Qpar-report:2 diagnostic du paralléliseur automatique ; indiquer les boucles non parallélisées
/Qvec-report:1 diagnostic du vectoriseur automatique ; indiquer les boucles vectorisées
/Qvec-report:2 diagnostic du vectoriseur automatique ; indiquer les boucles non vectorisées
/GL[-] activer la génération du code durant l'édition des liens
/volatile:<iso|ms> choisir le modèle de volatile :
iso - Les sémantiques acquire/release ne sont pas garanties sur les accès volatiles
ms - Les sémantiques acquire/release sont garanties sur les accès volatiles
/GA optimiser pour les applications Windows
/Ge forcer le contrôle de pile pour toutes les fonctions
/Gs[nombre] contrôler les appels de contrôle de pile
/Gh activer l'appel de la fonction _penter
/GH activer l'appel de la fonction _pexit
(appuyez sur <entrée> pour continuer)
/GT générer les accès TLS à fibres sécurisées
/RTC1 Activer les contrôles rapides (/RTCsu)
/RTCc Contrôles de la conversion des types les plus petits

/RTCs Contrôles à l'exécution des frames de pile
/RTCu Contrôles de l'utilisation des variables locales non initialisées
/clr[:option] compile for common language runtime, where option is:
pure : produce IL-only output file (no native executable code)
safe : produce IL-only verifiable output file
netcore : produce assemblies targeting .NET Core runtime
noAssembly : do not produce an assembly
nostdlib : ignore the system .NET framework directory when searching for assemblies
nostdimport : do not import any required assemblies implicitly
initialAppDomain : enable initial AppDomain behavior of Visual C++ 2002
implicitKeepAlive- : turn off implicit emission of System::GC::KeepAlive(this)
char_t- : turn off metadata support for char8_t, char16_t and char32_t
/fsanitize=address Activer la génération de code d'address sanitizer
/homeparams forcer l'écriture dans la pile des paramètres passés dans les registres
/GZ Activer les contrôles de pile (/RTCs)
/Gv convention d'appel __vectorcall
/arch:<SSE2|SSE4.2|AVX|AVX2|AVX512> exigences minimales d’architecture du processeur, au choix :
SSE2 – (par défaut) activer l’utilisation des instructions disponibles avec les UC compatibles SSE2
SSE 4.2 – activer l’utilisation des instructions disponibles avec les UC compatibles SSE 4.2
AVX – activer l’utilisation des instructions disponibles avec les UC compatibles AVX
AVX2 – activer l’utilisation des instructions disponibles avec les UC compatibles AVX2
AVX512 – activer l’utilisation des instructions disponibles avec les processeurs compatibles AVX-512
/QIntel-jcc-erratum permet d'activer les atténuations pour l'erratum JCC d'Intel
/Qspectre-load Active les atténuations de spectre pour toutes les instructions qui se chargent en mémoire
/Qspectre-load-cf Active les atténuations de spectre pour toutes les instructions de flux de contrôle qui se chargent en mémoire
/Qspectre-jmp[-] Activer les atténuations de Spectre pour les instructions de saut inconditionnel
(appuyez sur <entrée> pour continuer)
/fpcvt:<IA|BC> compatibilité de conversion entre virgule flottante et entier non signé
IA - résultats compatibles avec l'instruction VCVTTSD2USI
BC - résultats compatibles avec le compilateur VS2017 et les versions antérieures
/jumptablerdata Place les tables de raccourcis pour les instructions switch case dans la section .rdata

```
-FICHIERS DE SORTIE-
```

/Fa[fichier] nommer le fichier listing d'assembly
/FA[scu] configurer le listing assembleur
/Fd[fichier] nommer le fichier .PDB
/Fe nommer le fichier exécutable
/Fm[fichier] nommer le fichier de mappage
/Fo nommer le fichier objet
/Fp nommer le fichier d'en-tête précompilé
/Fr[fichier] nommer le fichier browser source
/FR[fichier] nommer le fichier .SBR étendu
/Fi[fichier] nommer le fichier prétraité/Fd: nommer le fichier .PDB
/Fe: nommer le fichier exécutable
/Fm: nommer le fichier de mappage
/Fo: nommer le fichier objet /Fp: nommer le fichier .PCH
/FR: nommer le fichier .SBR étendu
/Fi: nommer le fichier prétraité
/Ft

emplacement des fichiers d'en-tête générés pour #import
/doc[fichier] traiter les commentaires de documentation XML et nommer éventuellement le fichier .xdc

```
-PREPROCESSOR-
```

/AI<rép> ajouter au chemin de recherche des assemblys
/FU importer un assembly/module .NET
(appuyez sur <entrée> pour continuer)
/FU:asFriend importer un assembly/module .NET en tant qu'ami
/C ne pas supprimer les commentaires /D{=|#} définir une macro
/E prétraiter dans stdout /EP prétraiter dans stdout, sans #line
/P prétraiter dans un fichier
/Fx fusionner le code injecté dans un fichier
/FI nommer le fichier Include forcé
/U supprimer la macro prédéfinie
/u supprimer toutes les macros prédéfinies
/I<rép> ajouter au chemin de recherche Include
/X ignorer les "places standard"
/PH permet de générer #pragma file_hash durant le prétraitement
/PD affiche toutes les définitions de macro

```
-LANGUE-
```

/std:<c++14|c++17|c++20|c++latest> version standard C++
c++14 – ISO/IEC 14882:2014 (par défaut)
c++17 – ISO/IEC 14882:2017
c++20 – ISO/CEI 14882:2020
c++latest – dernier projet de norme (ensemble de fonctionnalités susceptible de changer)
/std :<c11|c17|clatest> C version standard
c11 - ISO/IEC 9899:2011
c17 - ISO/IEC 9899:2018
clatest - dernier brouillon standard (ensemble de fonctionnalités susceptible d'être modifié)
/permissive[-] permet de compiler du code non conforme
(ensemble de fonctionnalités susceptible d'être modifié) (désactivé par défaut en C++20 et versions ultérieures)
/Za désactive les extensions (non recommandé pour C++)
/ZW permet d'activer les extensions de langage WinRT
/Zs vérifier la syntaxe uniquement
(appuyez sur <entrée> pour continuer)
/await activer l'extension des fonctions pouvant être reprise
/await:strict activer la prise en charge de la coroutine standard C++ 20 avec les versions précédentes du langage
/constexpr:depth limite de profondeur de récursivité pour l'évaluation de constexpr (valeur par défaut : 512)
/constexpr:backtrace permet d'afficher N évaluations de constexpr dans les diagnostics (valeur par défaut : 10)
/constexpr:steps permet de mettre fin à l'évaluation de constexpr après N étapes (valeur par défaut : 1048576)
/Zi permet d'activer les informations de débogage
/Z7 permet d'activer les informations de débogage selon l'ancien format
/Zo[-] permet de générer des informations de débogage plus détaillées pour du code optimisé (activé par défaut)
/ZH:[MD5|SHA1|SHA_256] algorithme de hachage pour le calcul de la somme de contrôle du fichier dans les informations de débogage (par défaut : SHA_256)
/Zp[n] permet de compresser les structs à la limite n-octet
/Zl omettre le nom de bibliothèque par défaut dans .OBJ
/vd{0|1|2} permet de désactiver/d'activer vtordisp
/vm type des pointeurs vers les membres
/Zc:arg1[,arg2] conformité au langage, où les arguments peuvent être :
forScope[-] permet d'appliquer la norme C++ pour les règles de portée
wchar_t[-] wchar_t est le type natif et non un typedef
auto[-] permet d'appliquer la nouvelle signification C++ standard pour auto

trigraphs[-] permet d'activer les trigraphes (désactivés par défaut)
rvalueCast[-] permet d'appliquer les règles de conversion de type explicite standard C++
(activé par défaut en C++20 ou version ultérieure, implicitement par /permissive-)
strictStrings[-] désactive string-literal sur [char|wchar_t]*
conversion (activée par défaut en C++20 ou version ultérieure, implicite par /permissive-)
implicitNoexcept[-] permet d'activer un noexcept implicite sur les fonctions obligatoires
threadSafeInit[-] permet l'initialisation statique locale thread-safe
inline[-] supprime la fonction ou les données non référencées si elle est
COMDAT ou a une liaison interne uniquement (désactivée par défaut)
sizedDealloc[-] activer la désallocation de taille globale C++14
fonctions (activées par défaut)
throwingNew[-] suppose que l'opérateur new lève une exception en cas d'échec (désactivé par défaut)
referenceBinding[-] un élément temporaire ne sera pas lié à un élément non const
référence lvalue (activée par défaut en C++20 ou version ultérieure, implicite par /permissive-)
twoPhase- permet de désactiver la recherche de nom en deux phases
ternary[-] applique les règles C++11 pour l'opérateur conditionnel
(activé par défaut en C++20 ou version ultérieure, implicitement par /permissive-)
noexceptTypes[-] permet d'appliquer les règles de noexcept C++17 (activé par défaut dans C++17 ou version ultérieure)
alignedNew[-] permet d'activer l'alignement C++17 des objets alloués dynamiquement (activé par défaut)
hiddenFriend[-] appliquer les règles friend masquées C++ standard
(activé par défaut en C++20 ou version ultérieure, implicitement par /permissive-)
externC[-] applique les règles C++ standard pour les fonctions « extern C »
(activé par défaut en C++20 ou version ultérieure, implicitement par /permissive-)
lambda[-] meilleure prise en charge lambda à l'aide du processeur lambda plus récent
(activé par défaut en C++20 ou version ultérieure, implicitement par /permissive-)
tlsGuards[-] génère des vérifications au moment de l'exécution pour l'initialisation des variables TLS (activé par défaut)
zeroSizeArrayNew[-] appelle le membre new/delete pour les tableaux d'objets de taille égale à 0 (activé par défaut)
static_assert[-] gestion stricte de « static_assert » (activé par défaut en C++20 ou version ultérieure,
implicite par /permissive-)
gotoScope[-] ne peut pas passer au-delà de l'initialisation d'une variable (implicitement par /permissive-).
templateScope[-] appliquer les règles de masquage des paramètres de modèle C++ standard

enumTypes[-] activer les types enum sous-jacents C++ Standard (désactivé par défaut)
checkGwOdr[-] applique les violations de règle de définition standard C++ one
quand /Gw a été activé (désactivé par défaut)
nrvo[-] activer la copie et le déplacement facultatifs (activé par défaut en C++20 ou version ultérieure,
implicite par /permissive- ou /O2)
STDC définit **STDC** à 1 en C
cplusplus[-] macro cplusplus signale la norme C++ prise en charge (désactivée par défaut)
char8_t[-] activer la prise en charge littérale 'u8' native de C++20 en tant que 'const char8_t'
(activé par défaut en C++20 ou version ultérieure)
externConstexpr[-] active la liaison externe pour les variables constexpr en C++
(activé par défaut en C++20 ou version ultérieure, implicitement par /permissive-)
préprocesseur[-] activer le préprocesseur conforme standard en C/C++
(activé par défaut en C11 ou version ultérieure)
/ZI active modifier et continuer les informations de débogage
/openmp activer les extensions de langage OpenMP 2.0
/openmp:experimental active les extensions de langage OpenMP 2.0 plus certaines extensions de langage OpenMP 3.0+
/openmp:llvm extensions de langage OpenMP utilisant le runtime LLVM

```
-DIVERS-
```

@ fichier réponse des options /?, /help afficher ce message d'aide
/bigobj générer un format d'objet étendu
/c compiler uniquement, pas d'édition des liens
/FC utiliser des chemins complets dans les diagnostics
/H longueur maximale du nom externe
/J type de caractère par défaut non signé
/MP[n] utiliser jusqu'à 'n' processus pour la compilation
/nologo supprimer le message de copyright
/showIncludes afficher les noms de fichiers Include
/Tc compiler le fichier comme .c
/Tp compiler le fichier comme .cpp
/TC compiler tous les fichiers comme .c
/TP compiler tous les fichiers comme .cpp
/V définir la chaîne de version /Yc[fichier] créer un fichier .PCH
/Yd placer les informations de débogage dans chaque .OBJ
/Yl[sym] injecter la référence .PCH pour la bibliothèque de débogage

/Yu[fichier] utiliser le fichier .PCH /Y- désactiver toutes les options PCH
/Zm mémoire max. allouée (% de la valeur par défaut)
/FS utilisation forcée de MSPDBSRV.EXE
(appuyez sur <entrée> pour continuer)
/source-charset:<nom_iana>|.nnnn définit le jeu de caractères source
/execution-charset:<nom_iana>|.nnnn définit le jeu de caractères d'exécution
/utf-8 définit le jeu de caractères source et d'exécution à UTF-8
/validate-charset[-] valide les fichiers UTF-8 uniquement pour les caractères conformes
/fastfail[-] activer le mode fast-fail
/JMC[-] active Uniquement mon code en mode natif
/presetPadding[-] initialisation par remplissage à l'aide de zéros des types de classe basés sur une pile
/volatileMetadata[-] générer des métadonnées sur les accès à la mémoire volatile
/sourcelink [file] fichier contenant les informations de lien source

```
-ÉDITION DES LIENS-
```

/LD Créer une .DLL
/LDd Créer une bibliothèque .DLL de débogage
/LN Créer un .netmodule /F définir la taille de la pile
/link [options et bibliothèques de l'Éditeur de liens]
/MD lier avec MSVCRT.LIB /MT lier avec LIBCMT.LIB
/MDd lier avec la bibliothèque de débogage MSVCRTD.LIB
/MTd lier avec la bibliothèque de débogage LIBCMTD.LIB

```
-ANALYSE DU CODE-
```

/analyze[-] Activer l'analyse native
/analyze:quiet[-] Aucun avertissement sur la console
/analyze:log Avertissements sur le fichier
/analyze:autolog Enregistrer le journal sur *.pftlog*
/analyze:autolog:ext Enregistrer le journal sur .
/analyze:autolog- Pas de fichier journal/analyze:WX- Avertissements récupérables
/analyze:stacksize Frame de pile maximal
(appuyez sur <entrée> pour continuer)
/analyze:max_paths Chemins d'accès maximum
/analyze:only Analyse, pas de génération de code

```
-DIAGNOSTICS-
```

/diagnostics:<arguments,...> contrôle le format des messages de diagnostic :
classic - conserve le format antérieur
column[-] - affiche les informations de la colonne
caret[-] - affiche la colonne et la ligne de code source indiquée
/Wall activer tous les avertissements
/w permet de désactiver tous les avertissements
/W définir le niveau d'avertissement (par défaut n=1)
/Wv:xx[.yy[.zzzzz]] désactiver les avertissements introduits après la version xx.yy.zzzzz
/WX traiter les avertissements comme des erreurs
/WL activer les diagnostics sur une ligne
/wd désactiver l'avertissement n
/we traiter l'avertissement n comme une erreur
/wo émettre l'avertissement n une fois
/w définir le niveau d'avertissement 1-4 pour n
/external:I - emplacement des en-têtes externes
/external:env: - *variable d'environnement avec des emplacements d'en-têtes externes*
/external:anglebrackets - permet de traiter tous les en-têtes inclus via <> en tant qu'en-têtes externes
/external:W - niveau d'avertissement des en-têtes externes
/external:templates[-] - permet d'évaluer le niveau d'avertissement à travers la chaîne d'instanciation de modèle
/sdl activer des avertissements et des fonctionnalités de sécurité supplémentaires
/options : les options du compilateur strictes non reconnues sont une erreur

Syntax of "link" command is next :

link /link
Microsoft (R) Incremental Linker Version 14.40.33813.0

utilisation : LINK [options] [fichiers] [@fichiercommandes]

options :

```
   /ALIGN:#
   /ALLOWBIND[:NO]
   /ALLOWISOLATION[:NO]
   /APPCONTAINER[:NO]
   /ASSEMBLYDEBUG[:DISABLE]
   /ASSEMBLYLINKRESOURCE:nomfichier
   /ASSEMBLYMODULE:nomfichier
   /ASSEMBLYRESOURCE:nomfichier[,[nom][,PRIVATE]]
```

```
/BASE:{adresse[,taille]|@nomfichier,clé}
/CLRIMAGETYPE:{IJW|PURE|SAFE|SAFE32BITPREFERRED}
/CLRLOADEROPTIMIZATION:{MD|MDH|NONE|SD}
/CLRSUPPORTLASTERROR[:{NO|SYSTEMDLL}]
/CLRTHREADATTRIBUTE:{MTA|NONE|STA}
/CLRUNMANAGEDCODECHECK[:NO]
/DEBUG[:{FASTLINK|FULL|NONE}]
/DEF:nomfichier
/DEFAULTLIB:bibliothèque
/DELAY:{NOBIND|UNLOAD}
/DELAYLOAD:dll
/DELAYSIGN[:NO]
/DEPENDENTLOADFLAG:flag
/DLL
/DRIVER[:{UPONLY|WDM}]
/DYNAMICBASE[:NO]
/EMITVOLATILEMETADATA[:NO]
/ENTRY:symbol
/ERRORREPORT:{NONE|PROMPT|QUEUE|SEND}
/EXPORT:symbol
/EXPORTPADMIN[:size]
/FASTFAIL[:NO]
/FASTGENPROFILE[:{COUNTER32|COUNTER64|EXACT|MEMMAX=#|MEMMIN=#|NOEXACT|
                 NOPATH|NOTRACKEH|PATH|PGD=filename|TRACKEH}]
/FILEALIGN:#
/FIXED[:NO]
/FORCE[:{MULTIPLE|UNRESOLVED}]
/FUNCTIONPADMIN[:taille]
/GUARD:{CF|NO}
/GENPROFILE[:{COUNTER32|COUNTER64|EXACT|MEMMAX=#|MEMMIN=#|NOEXACT|
             NOPATH|NOTRACKEH|PATH|PGD=filename|TRACKEH}]
/HEAP:réserve[,validation]
/HIGHENTROPYVA[:NO]
/IDLOUT:nomfichier
/IGNORE:#
/IGNOREIDL
/ILK:filename
/IMPLIB:nomfichier
/INCLUDE:symbole
/INCREMENTAL[:NO]
```

```
/INTEGRITYCHECK
/KERNEL
/KEYCONTAINER:nom
/KEYFILE:nomfichier
/LARGEADDRESSAWARE[:NO]
/LIBPATH:rép
/LINKREPRO:dir
/LINKREPROTARGET:filename
/LTCG[:{INCREMENTAL|NOSTATUS|OFF|STATUS|}]
/LTCGOUT:nom_fichier
/MACHINE:{ARM|ARM64|ARM64EC|ARM64X|EBC|X64|X86}
/MANIFEST[:{EMBED[,ID=#]|NO}]
/MANIFESTDEPENDENCY:dépendance de manifeste
/MANIFESTFILE:nomfichier
/MANIFESTINPUT:nomfichier
/MANIFESTUAC[:{NO|UAC fragment}]
/MAP[:nomfichier]
/MAPINFO:{EXPORTS|PDATA}
/MERGE:depuis=vers
/MIDL:@fichiercommandes
/NATVIS:filename
/NOASSEMBLY
/NODEFAULTLIB[:bibliothèque]
/NOENTRY
/NOEXP
/NOIMPLIB
/NOLOGO
/NXCOMPAT[:NO]
/OPT:{ICF[=iterations]|LBR|NOICF|NOLBR|NOREF|REF}
/ORDER:@nomfichier
/OUT:nomfichier
/PDB:nomfichier
/PDBINJECT:{MAPFILE}
/PDBSTRIPPED[:filename]
/PROFILE
/RELEASE
/SAFESEH[:NO]
/SECTION:nom,[[!]{DEKPRSW}][,ALIGN=#]
/SOURCELINK:filename
/STACK:reserve[,commit]
```

```
   /STUB:filename
   /SUBSYSTEM:{BOOT_APPLICATION|CONSOLE|EFI_APPLICATION|
               EFI_BOOT_SERVICE_DRIVER|EFI_ROM|EFI_RUNTIME_DRIVER|
               NATIVE|POSIX|WINDOWS|WINDOWSCE}[,#[.##]]
   /SWAPRUN:{CD|NET}
   /TLBID:#
   /TLBOUT:nomfichier
   /TIME
   /TSAWARE[:NO]
   /USEPROFILE[:{AGGRESSIVE|PGD=filename}]
   /VERBOSE[:{CLR|ICF|INCR|LIB|REF|SAFESEH|UNUSEDDELAYLOAD|UNUSEDLIBS}]
   /VERSION:#[.#]
   /WINMD[:{NO|ONLY}]
   /WINMDDELAYSIGN[:NO]
   /WINMDFILE:nomfichier
   /WINMDKEYCONTAINER:name
   /WINMDKEYFILE:filename
   /WHOLEARCHIVE[:library]
   /WX[:NO]
   /WX[:nnnn[,nnnn...]]
```

You can consult many documentation on site : https://learn.microsoft.com/fr-fr/visualstudio/?view=vs-2022