How config CB MinGW64 GCC C_C++ compiler into CodeBlocks

Full name of tutorial : How config CB MinGW64 GCC C/C++ compiler into Code::Blocks on Windows 11 64 bits.

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

Presentation of MinGW64 included into CB

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 compiler GCC included in IDE CB directly into Code::Blocks on Windows 11 64 bits.

MinGW : A native Windows port of the GNU Compiler Collection (GCC), with freely distributable import libraries and header files for building native Windows applications; includes extensions to the MSVC runtime to support C99 functionality.
All of MinGW's software will execute on the 64bit Windows platforms.

How to install GNU GCC Compiler C/C++ included in IDE CB (version gcc 10.3.0, 64 bits) on Windows 11 64 bits ?

You can download last version of IDE Code::Blocks on Internet site of FossHub :
https://www.fosshub.com/Code-Blocks.html?dwl=codeblocks-20.03mingw-setup.exe

After download, you can click on this executable to install IDE Code::Blocks on directory C:\CodeBlocks by example, but version of GCC is enought old.
Then, it is recommended to replace by most recent version by download a "nigthly build" dated june 2023 :
https://sourceforge.net/projects/codeblocks/files/Binaries/Nightlies/2023/CB_20230604_rev13311_win64-setup-MinGW.exe/download

Idem, click on this executable to install this new version.

Configuration of MinGW64 included in CB into Code::Blocks

You must configure this new compiler into CB by selecting main menu "Settings" then submenu "Compiler..." into IDE interface, then, choose "GNU GCC Compiler (default)" and rename this like "CB GNU GCC Compiler 64bit" by example.

Then, you choose tab" "Toolchain executable" to position good environment like this

Toolchain executable :
C:\CodeBlocks\MinGW (subdirectory \bin will be searched automatically to access to binaries listed after)
- compilateur C : gcc.exe
- compilateur C++ : g++.exe
- linker for dynamic lib : g++.exe
- linker for static lib : gcc-ar.exe
- debugger : GDB (default)
- resource compiler : windres.exe
- make program : mingw32-make.exe

Then, you choose tab" "Search directories" to position good environment like this :

```
- to compiler :                 C:\CodeBlocks\MinGW\x86_64-w64-mingw32\include
and
                                            C:\CodeBlocks\MinGW\include
- to linker :           C:\CodeBlocks\MinGW\x86_64-w64-mingw32\lib    and
                                            C:\CodeBlocks\MinGW\lib
- to resource compiler : C:\CodeBlocks\MinGW\x86_64-w64-mingw32\include and
                                            C:\CodeBlocks\MinGW\include
```

Then, you can type this into command console Windows : "C:\CodeBlocks\MinGW\bin\g++.exe --version"; result here :

g++.exe (MinGW-W64 x86_64-ucrt-posix-seh, built by Brecht Sanders) 13.1.0
Copyright (C) 2023 Free Software Foundation, Inc.
This is free software; see the source for copying conditions. There is NO warranty;
not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

Test of "simple" code with MinGW64 included into CB with Code::Blocks

And, 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", and choose compiler "CB GNU GCC Compiler 64bit".
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).
If, you apply all of precedent instructions, compile and link of your program with this new compiler 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 GCC compiler of MinGW64 included in CB in command line (just to illustrate)

You can also use compiler GCC of MinGW64 include in Red-Panda in command console on Windows (CMD.EXE) with next instructions :

```
set PATHSAV=%PATH%
set PATH=C:\CodeBlocks\MinGW\bin;%PATH%
REM Next instructions are not mandatory, but you can set var if you want.
REM set CLANG=C:\CodeBlocks\MinGW\x86_64-w64-mingw32\include;C:\CodeBlocks\MinGW\include
REM set LIBRARY_PATH=C:\CodeBlocks\MinGW\x86_64-w64-mingw32\lib,C:\CodeBlocks\MinGW\lib
```

REM Generate console application in one pass
gcc hellowworld.c -o hellowworld.exe
REM Generate console application in two pass
gcc -c hellowworld.c -o hellowworld.obj
gcc hellowworld.obj -o hellowworld.exe -Wl,--subsystem console

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

set PATH=%PATHSAV%

But, it's much easy to use MinGW64 associated with CB directly into Code::Blocks IDE especially with complex C program (many C sources and many subdirectories ...) , and multiple targets by example : main DLL and console program to test this DLL, ... -)

PS3 : Syntax of tools GCC

Command "gcc" present a very "verbose" list of options, but to resume principal and useful options, you can use
by example :
- "-m16" Generate i386 16 bits object or executable.
- "-m32" Generate i386 32 bits object or executable.
- "-m64" Generate x64 64 bits object or executable.
- "-c" Compile and assemble, but do not link.
- "-D var[=value]" Define variable to be use by preprocessor, and optionnally affect an value at this variable.
- "-o " Place the output into .
- "-I " Add directory to search include files.
- "-L " Add directory to search library files.
- "-shared" Create a shared library.
- "-llibrary" Give name of library used by linker.
- "-pthread" Link with the POSIX threads library.
- "-Wl," Pass comma-separated on to the linker. Example of options :
-t trace all input files used by linker
--kill-at not add the function name decorations at-sign and number for stdcall functions
--add-stdcall-alias export functions with the stdcall decoration suffix (@nn) and also without this suffix
--dll create DLL on Win32 systems
--export-all-symbols all global symbols in the objects used to build a DLL will be exported by the DLL
--output-def file create a def file of all exported sysmbols
--out-implib file create a import library in parallel of creation of shared library

--subsystem ident create target based on "ident" that can be valued by "native, windows, console, posix, or xbox"

You can consult an summary of these options on site : https://gcc.gnu.org/onlinedocs/gcc/Option-Summary.html
And for "ld" (linker of gcc), you can consult too : http://sourceware.org/binutils/docs-2.16/ld/Options.html