How config Red Panda MinGW64 GCC C_C++ compiler into CodeBlocks

Name of tutorial : How config Red Panda 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
```

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 Red Panda on Windows (fork of IDE Dev-Cpp because no activity
since 2016 ...).

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 Red Panda Dev-Cpp (version gcc 10.3.0, 64 bits) on Windows 11 64 bits ?

The IDE Dev-Cpp is not maintained since 2016, Red Panda is a fork of this, and last version is dated of 2022.
You can download last version of Red Panda on Internet site of SourceForge :
https://sourceforge.net/projects/redpanda-cpp/files/v3.1/RedPanda.C++.3.1.win64.MinGW64_11.4.Setup.exe

After download, you can click on this executable to install IDE "Red Panda" Dev-Cpp on directory C:\RedPanda-Cpp by example,
version of GCC 64 bit installed after is 11.2.0.

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 "Dev-Cpp GNU GCC Compiler 64bit" by example.

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

Toolchain executable :
C:\RedPanda-Cpp\MinGW64 (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.exe
resource compiler : windres.exe
make program : mingw32-make.exe

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

```
to compiler :                   C:\RedPanda-Cpp\MinGW64\x86_64-w64-
mingw32\include and C:\RedPanda-Cpp\MinGW64\include
to linker :                     C:\RedPanda-Cpp\MinGW64\x86_64-w64-mingw32\lib
and C:\RedPanda-Cpp\MinGW64\lib
to resource compiler :  C:\RedPanda-Cpp\MinGW64\x86_64-w64-mingw32\include and
C:\RedPanda-Cpp\MinGW64\include
```

Then, you can type this into command console Windows : "C:\RedPanda-Cpp\MinGW64\bin\g++.exe --version"; result here :

g++.exe (x86_64-posix-seh-rev1, Built by MinGW-W64 project) 11.2.0
Copyright (C) 2021 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.

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
"Dev-Cpp 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.

Question : Why choose IDE CB in priority and not IDE Red Panda Dev-Cpp ?
Response : In my mind, CB is very much open than Red Panda. It accept multiple C/C++ compilers, and not only MinGW'types of
these ... For me, it's very important that IDE can generate multiple targets on Windows with many free compilers.

Question : And why configure GCC included into IDE Red Panda Dev-Cpp into CB ?
Response : Just to verify that it's not difficult to add this compiler into CB, only create "new compiler" and configure this.
Simply.

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 : 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:\RedPanda-Cpp\MinGW64\bin;%PATH%
REM Next instructions are not mandatory, but you can set var if you want.
REM set CLANG=C:\RedPanda-Cpp\MinGW64\x86_64-w64-mingw32\include;C:\RedPanda-Cpp\MinGW64\include
REM set LIBRARY_PATH=C:\RedPanda-Cpp\MinGW64\x86_64-w64-mingw32\lib,C:\RedPanda-Cpp\MinGW64\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 GCC of MinGW64 directly into CB IDE especially with complex C program (many C sources
and many subdirectories ...) -)

PS3 : 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 préprocessor, 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 useful 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 of shared library
--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