How config Borland C_C++ compiler (32 bits only) into CodeBlocks

Name of tutorial : How config Borland C/C++ compiler (32 bits only) 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 an compiler on Windows : Borland C/C++ compiler.

How to install Borland C/C++ compiler (very old today ... last version 2000 ... and restrict to 32 bits) ?

You can download it from Internet site :
https://developerinsider.co/download-and-install-borland-c-compiler-on-windows-10/

File proposed to download is : "BorlandCPP.zip", and you must decompress on your computer. This archive contains multiple
directories. To simplify, you can choose to decompress in C:\temp directory, and after transfer subdirectory \Borland to C:.
To purge all files of archive, think to delete directories "__MACOSX" and "Borland C++" and all subdirectories under, present
on C:\temp.

Normally, after that, next run of CB detect presence of this compiler and proposed it in list of available compiler in main
menu "Settings" and after submenu "Compiler..." : "Borland C++ Compiler (5.5, 5.82)". If you select this, verify that fields
describe next are parametered into CB.

In tab "Toolchain executable", you must find in field "Compiler installation directory" :
C:\Borland\BCC55 (subdirectory "\bin" automatically searched after this "top" directory),
and in subtab "Program Files", list next :
compilateur C : bcc32.exe
compilateur C++ : bcc32.exe

linker for dynamic lib : ilink32.exe
linker for static lib : tlib.exe
debugger :
resource compiler : brcc32.exe
make program : make.exe

If CB propose different values of fields described below, you can change/force it.

After,you select tab "Search directories", and into each subtab, you write with "add" button, if not searched by default :
to compiler : C:\Borland\BCC55\include
to linker : C:\Borland\BCC55\lib
to resource compiler : C:\Borland\BCC55\include

Before generate to first program, think to verify or change two configuration files into directory "C:\Borland\BCC55\bin",
bcc32.cfg and ilink32.cfg (respectively to compiler and to linker of Borland C/C++) like this :

file "bcc32.cfg" :

```
-I"C:\Borland\BCC55\include"
-L"C:\Borland\BCC55\Lib\PSDK;C:\Borland\BCC55\Lib"
```

file "ilink32.cfg" :

```
-L"C:\Borland\BCC55\Lib\PSDK;C:\Borland\BCC55\Lib"
```

Normally, nothing to do, but it's mandatory to point respectively to "good" include files and libraries directories,
by example, if you install this compiler on different directory (D:\ ...), or if you install in "most short" directory like
"C:\BCC55".

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" by default, and choose
compiler "Borland C++ Compiler (5.5, 5.82)".
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 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 : Borland C/C++ compiler can be used directly into command console of Windows (CMD.EXE) and next command lines configure it :

```
SET PATHSAV=%PATH%
SET BCC=C:\Borland\BCC55
SET PATH=%BCC%\bin;%PATH%
REM Thank's to configuration files present to be used in conjonction of commands "bcc32" and "ilink32"
REM No need to define INCLUDE or LIB variable ...
```

By example, to generate "console" executable :

```
REM Generate console executable in one pass
bcc32 -WC hellowworld.c -ehelloworld.exe
REM Generate console executable in two pass
bcc32 -c -WC hellowworld.c -ohellowworld.obj
ilink32 hellowworld.obj, helloworld.exe
```

After, work with Borland C/C++ compiler, but, at the end of your work, think to return in initial state ... to avoid difficulties :

```
.....
SET PATH=%PATHSAV%
```

PS3 : Syntax and options of command line "bcc32.exe"

```
bcc32
Borland C++ 5.5.1 for Win32 Copyright (c) 1993, 2000 Borland
Syntax is: BCC32 [ options ] file[s] = default; -x- = turn switch x off
```

-*3* 80386 Instructions -4 80486 Instructions

-5 Pentium Instructions -6 Pentium Pro Instructions

-Ax Disable extensions -B Compile via assembly

-C Allow nested comments -Dxxx Define macro

-Exxx Alternate Assembler name -Hxxx Use pre-compiled headers

-Ixxx Include files directory -K Default char is unsigned

-Lxxx Libraries directory -M Generate link map

-N Check stack overflow -Ox Optimizations

-P Force C++ compile -R Produce browser info

-RT *Generate RTTI -S Produce assembly output*

-Txxx Set assembler option -Uxxx Undefine macro

-Vx Virtual table control -X Suppress autodep. output

-aN Align on N bytes -b Treat enums as integers

-c Compile only -d Merge duplicate strings

-exxx Executable file name -fxx Floating point options

-gN Stop after N warnings -iN Max. identifier length

-jN Stop after N errors -k *Standard stack frame*

-lx Set linker option -nxxx Output file directory

-oxxx Object file name -p Pascal calls

-tWxxx Create Windows app -u Underscores on externs

-v Source level debugging -wxxx Warning control

-xxxx Exception handling -y Produce line number info

-zxxx Set segment names

The following table is an alphabetical listing of the Borland C/C++ compiler options:

Option Description

@ Read compiler options from the response file filename

+ Use alternate compiler configuration file filename

-3 Generate 80386 protected-mode compatible instructions (Default)

-4 Generate 80386/80486 protected-mode compatible instructions

-5 Generate Pentium instructions

-6 Generate Pentium Pro instructions

-A Use ANSI keywords and extensions

-AK Use Kernighan and Ritchie keywords and extensions

-AT Use Borland C++ keywords and extensions (also -A-)

-AU Use UNIX V keywords and extensions

-a Default (-a4) data alignment; -a- is byte alignment (-a1)

-a n Align data on "n" boundaries, where 1=byte, 2=word (2 bytes),
4=double word (4 bytes), 8=quad word (8 bytes), 16=paragraph (16 bytes)

(Default: -a4)
-B Compile to .ASM (-S), then assemble to .OBJ
-b Make enums always integer-sized (Default: -b makes enums integer size)
-b- Makes enums byte-sized when possible
-C Turn nested comments on (Default: -C- turn nested comments off)
-CP Enable code paging (for MBCS)
-c Compile to .OBJ, no link
-D Define "name" to the null string
-D<name=string> Define "name" to "string"
-d Merge duplicate strings
-d- Does not merge duplicate strings (Default)
-E Specify assembler
-e Specify executable file name
-f Emulate floating point
-f- No floating point
-ff Fast floating point
-fp Correct Pentium FDIV flaw
-gb Stop batch compilation after first file with warnings (Default = OFF)
-g n Warnings: stop after n messages (Default = 255)
-G, -G- Optimize for size/speed; use - O1 and -O2 instead
-H Generate and use precompiled headers
-H- Does not generate or use precompiled headers (Default)
-H= Set the name of the file for precompiled headers
-H"xxx" Stop precompiling after header file xxx
-Hc Cache precompiled header (Must be used with -H or -H"xxx"
-He Enable precompiled headers with external type files (Default)
-Hh=xxx Stop precompiling after header file xxx
-Hs Enable smart cached precompiled headers (Default)
-Hu Use but do not generate precompiled headers
-I

Syntax and options of command line "ilink2.exe"

ilink32
Turbo Incremental Link 5.00 Copyright (c) 1997, 2000 Borland
Syntax: ILINK32 objfiles, exefile, mapfile, libfiles, deffile, resfiles
@xxxx indicates use response file xxxx
General Options: -Af:nnnn Specify file alignment
-C Clear state before linking -Ao:nnnn Specify object alignment
-wxxx Warning control -ax Specify application type

```
-Enn Max number of errors -b:xxxx Specify image base addr
-r Verbose linking -Txx Specify output file type
-q Supress banner -H:xxxx Specify heap reserve size
-c Case sensitive linking -Hc:xxxx Specify heap commit size
-v Full debug information -S:xxxx Specify stack reserve size
-Gn No state files -Sc:xxxx Specify stack commit size
-Gi Generate import library -Vd.d Specify Windows version
-GD Generate .DRC file -Dstring Set image description
Map File Control: -Vd.d Specify subsystem version
-M Map with mangled names -Ud.d Specify image user version
-m Map file with publics -GC Specify image comment string
-s Detailed segment map -GF Set image flags
-x No map -Gl Static package
Paths: -Gpd Design time only package
-I Intermediate output dir -Gpr Runtime only package
-L Specify library search paths -GS Set section flags
-j Specify object search paths -Gt Fast TLS
Image Control: -Gz Do image checksum
-d Delay load a .DLL -Rr Replace resources
```

Just to illustrate, an example of generation of console application with source file and resource file :

```
bcc32 -c -w -w-par -w-inl -W -a1 -O2 -6 -DNDEBUG -I%INCLUDE% -
oobjBC55\Release%NAME_APPLI%.obj src%NAME_APPLI%.c
brcc32 -32 -i%INCLUDE% -foobjBC55\Release%NAME_APPLI%.res src%NAME_APPLI%.rc
ilink32 -q -aa -V4.0 -c -x -Gn -L"%LIB%" c0x32w.obj objBC55\Release%NAME_APPLI%.obj,
binBC55\Release%NAME_APPLI%.exe,
, import32.lib cw32.lib glu32.lib opengl32.lib gdi32.lib advapi32.lib comdlg32.lib winmm.lib
user32.lib kernel32.lib, ,objBC55\Release%NAME_APPLI%.res
```

And to terminate, an example of generation of DLL application with source file and resource file :

```
bcc32 -c -w -w-par -w-inl -W -a1 -O2 -6 -v -DDEBUG -D_DEBUG -I%INCLUDE% -
oobjBC55\Debug%NAME_APPLI%.obj src%NAME_APPLI%.c
brcc32 -32 -i%INCLUDE% -foobjBC55\Debug%NAME_APPLI%.res src%NAME_APPLI%.rc
ilink32 -q -Tpd -aa -V4.0 -c -x /Gi -L"%LIB%" c0d32.obj objBC55\Debug%NAME_APPLI%.obj,
binBC55\Debug%NAME_APPLI%.dll,
, import32.lib cw32.lib glu32.lib opengl32.lib gdi32.lib advapi32.lib comdlg32.lib winmm.lib
user32.lib kernel32.lib, ,objBC55\Debug%NAME_APPLI%.res
REM Next command is not mandatory, because option "/Gi" before, generate it. Note exact
```

```
syntax option "/Gi" because option "-Gi" don't work !!!
REM implib -c binBC55\Debug%NAME_APPLI%.lib binBC55\Debug%NAME_APPLI%.dll
```