

ADATSZERKEZETEK ÉS ALGORITMUSOK

C++ környezet beállítása

CLion

- A félév során a gyakorlatokon használt szoftverek
 - JetBrains – CLion fejlesztőkörnyezet <https://www.jetbrains.com/clion/>
 - 2021.2, vagy újabb
 - Minden JetBrains termék ingyenes diákoknak, csak az egyetemi e-mail címet kell használni.

CLion

- A félév során a gyakorlatokon használt szoftverek
 - g++ 10.3, Make, CMake, vagy ennek megfelelő:
 - Windows: MinGW-w64 az MSYS2-n keresztül: <https://www.msys2.org/>
 - MSYS2 telepítés
 - Terminálban telepítés: `pacman -S mingw-w64-x86_64-gcc mingw-w64-x86_64-gdb mingw-w64-x86_64-make mingw-w64-x86_64-cmake`
 - Linux: g++ 10
 - Ubuntu LTS-eken: `sudo apt install g++-10 cmake`
 - Arch alapú: `sudo pacman -S base-devel`
 - Mac: Apple-Clang, lehetőleg 10 vagy újabb
 - Kisebb verziószám esetén néhány kód nem fog fordulni (C++17 support illene).
Részletek.: https://en.cppreference.com/w/cpp/compiler_support

Windows telepítés – MSYS2

- <https://www.msys2.org/> - Telepítési utasítások követése
- Fontos: pacman -Syu
 - Addig kell ismételni, amíg ez nem látható

```
$ pacman -Syu
:: Synchronizing package databases...
mingw32 is up to date
mingw64 is up to date
ucrt64 is up to date
clang64 is up to date
msys is up to date
:: Starting core system upgrade...
there is nothing to do
:: Starting full system upgrade...
there is nothing to do
```

Windows telepítés – MSYS2

- <https://www.msys2.org/> - Telepítési utasítások követése
- Ezután fel kell telepíteni a compilert, a Make-et és a CMake-et:
 - `pacman -S mingw-w64-x86_64-{gcc,gdb,make,cmake}`
 - Telepítés helye alapértelmezetten `c:\msys64\mingw64`

```
$ pacman -S mingw-w64-x86_64-{gcc,gdb,make,cmake}
resolving dependencies...
looking for conflicting packages...
```

Packages (56)

```
mingw-w64-x86_64-binutils-2.37-4 mingw-w64-x86_64-brotli-1.0.9-3 mingw-w64-x86_64-bzip2-1.0.8-2 mingw-w64-x86_64-c-ares-1.17.1-1
mingw-w64-x86_64-ca-certificates-20200601-3 mingw-w64-x86_64-crt-git-9.0.0.6294.f5ac9206e-1 mingw-w64-x86_64-curl-7.78.0-1
mingw-w64-x86_64-expat-2.4.1-1 mingw-w64-x86_64-gcc-libs-10.3.0-5 mingw-w64-x86_64-gettext-0.19.8.1-10 mingw-w64-x86_64-gmp-6.2.1-2
mingw-w64-x86_64-headers-git-9.0.0.6294.f5ac9206e-1 mingw-w64-x86_64-isl-0.24-1 mingw-w64-x86_64-jansson-2.13.1-1
mingw-w64-x86_64-jemalloc-5.2.1-2 mingw-w64-x86_64-jsoncpp-1.9.4-1 mingw-w64-x86_64-libarchive-3.5.1-3 mingw-w64-x86_64-libffi-3.3-4
mingw-w64-x86_64-libiconv-1.16-2 mingw-w64-x86_64-libidn2-2.3.1-1 mingw-w64-x86_64-libpsl-0.21.1-4 mingw-w64-x86_64-libssh2-1.9.0-5
mingw-w64-x86_64-libsystre-1.0.1-4 mingw-w64-x86_64-libtasn1-4.17.0-1 mingw-w64-x86_64-libtre-git-r128.6fb7206-2
mingw-w64-x86_64-libunistring-0.9.10-4 mingw-w64-x86_64-libuv-1.40.0-3 mingw-w64-x86_64-libwinpthread-git-9.0.0.6294.f5ac9206e-1
mingw-w64-x86_64-libxml2-2.9.12-3 mingw-w64-x86_64-lz4-1.9.3-1 mingw-w64-x86_64-mpc-1.2.1-1 mingw-w64-x86_64-mpdecimal-2.5.1-1
mingw-w64-x86_64-mpfr-4.1.0.p13-1 mingw-w64-x86_64-ncurses-6.2-3 mingw-w64-x86_64-nettle-3.7.3-3 mingw-w64-x86_64-nghttp2-1.43.0-1
mingw-w64-x86_64-openssl-1.1.1.l-1 mingw-w64-x86_64-p11-kit-0.24.0-1 mingw-w64-x86_64-pkgconf-1.7.4-2 mingw-w64-x86_64-python-3.9.6-4
mingw-w64-x86_64-readline-8.1.001-1 mingw-w64-x86_64-rhash-1.4.1-3 mingw-w64-x86_64-sqlite3-3.36.0-1 mingw-w64-x86_64-tcl-8.6.11-5
mingw-w64-x86_64-termcap-1.3.1-6 mingw-w64-x86_64-tk-8.6.11.1-2 mingw-w64-x86_64-windows-default-manifest-6.4-3
mingw-w64-x86_64-winpthreads-git-9.0.0.6294.f5ac9206e-1 mingw-w64-x86_64-xxhash-0.8.0-1 mingw-w64-x86_64-xz-5.2.5-2 mingw-w64-x86_64-zlib-1.2.11-9
mingw-w64-x86_64-zstd-1.5.0-1 mingw-w64-x86_64-cmake-3.21.2-1 mingw-w64-x86_64-gcc-10.3.0-5 mingw-w64-x86_64-gdb-10.2-2 mingw-w64-x86_64-make-4.3-1
```

```
Total Download Size: 114.79 MiB
Total Installed Size: 882.08 MiB
```

```
:: Proceed with installation? [Y/n]
```

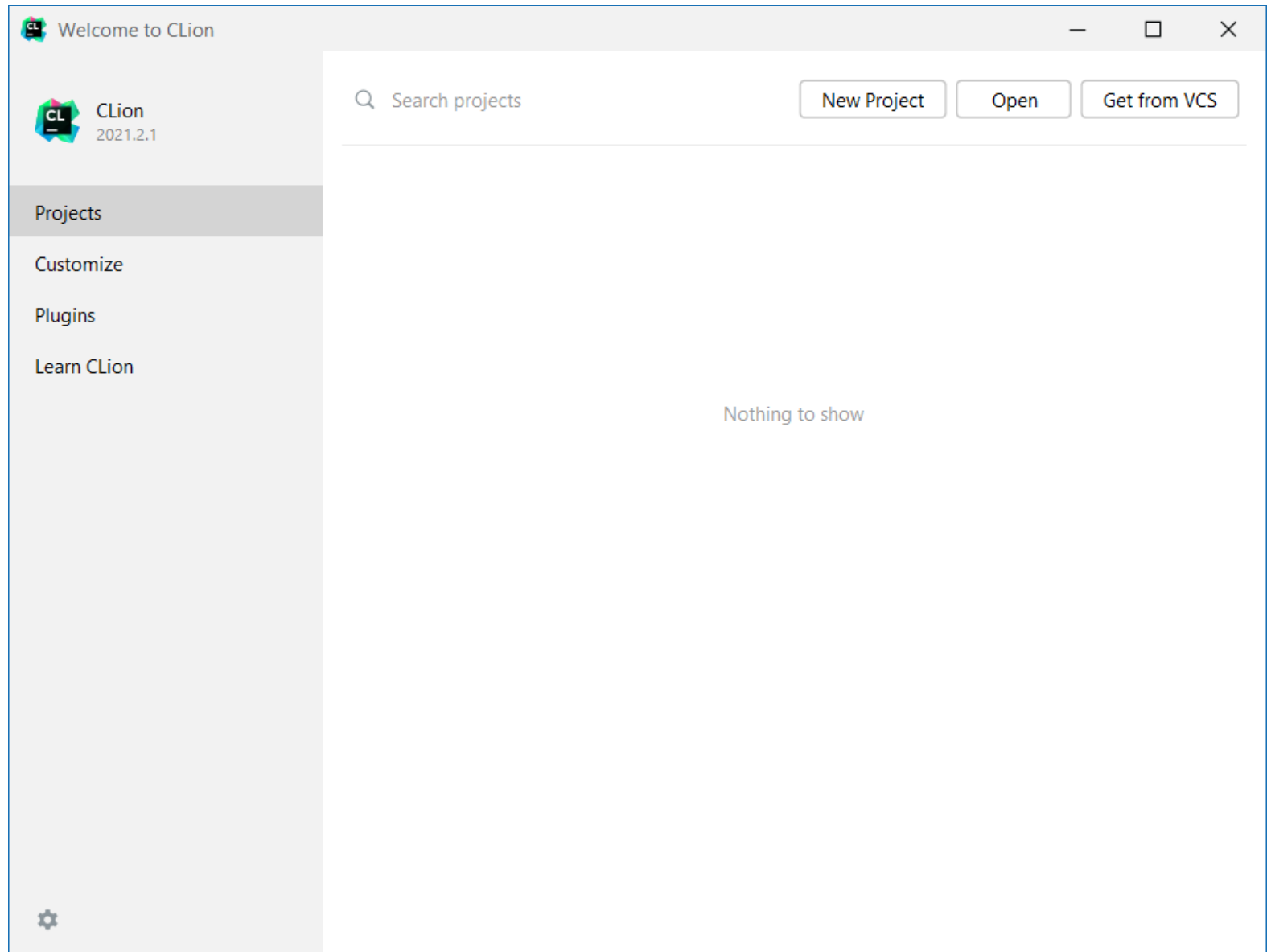
Windows telepítés – MSYS2

- <https://www.msys2.org/> - Telepítési utasítások követése
- Ezután fel kell telepíteni a compilert, a Make-et és a CMake-et:
 - `pacman -S mingw-w64-x86_64-{gcc,gdb,make,cmake}`
 - Telepítés helye alapértelmezetten `c:\msys64\mingw64`
 - Ellenőrzés:

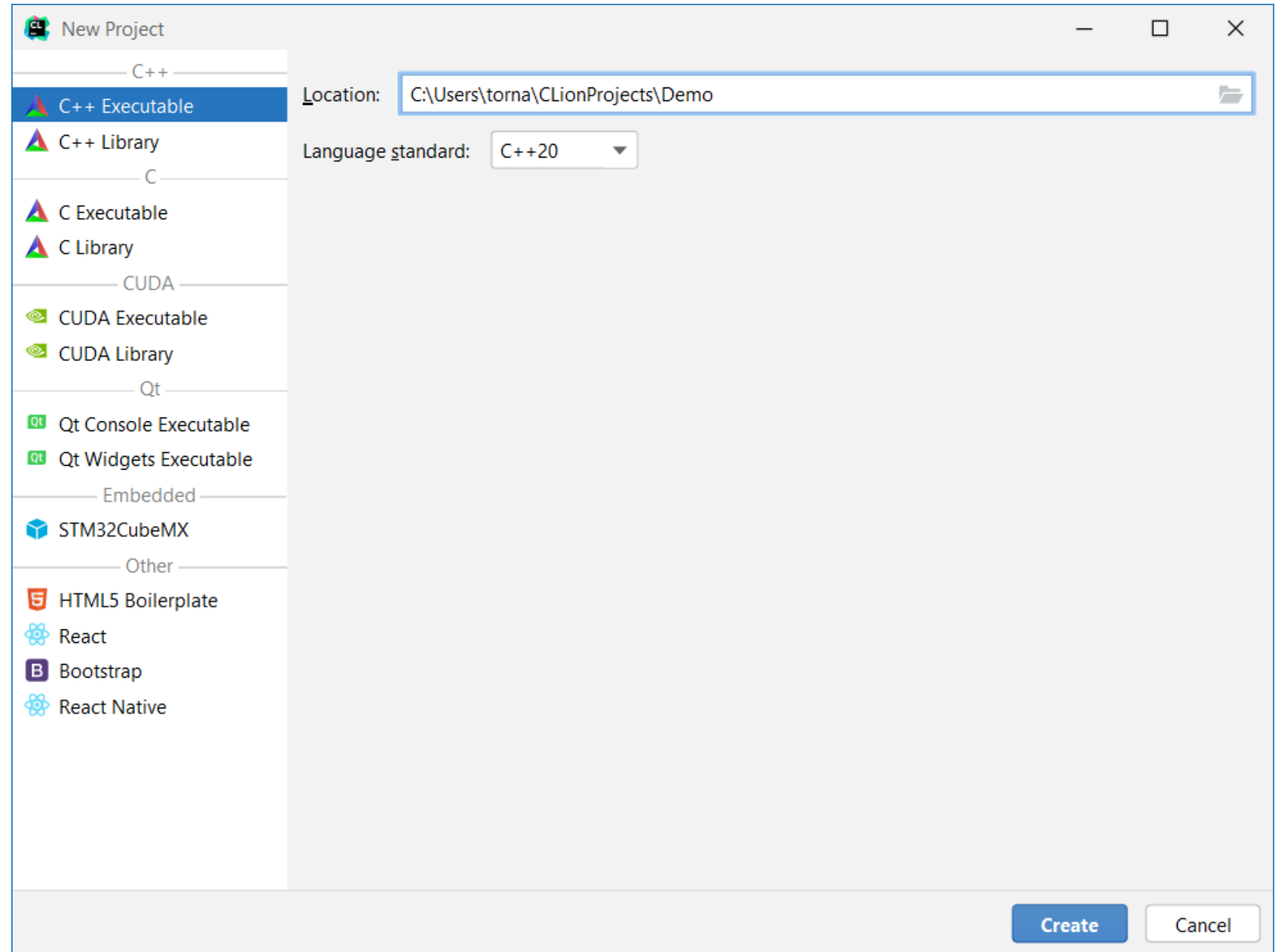
```
$ pacman -Qs gcc
local/gcc-libs 10.2.0-1 (msys2-devel)
Runtime libraries shipped by GCC
local/mingw-w64-x86_64-gcc 10.3.0-5 (mingw-w64-x86_64-toolchain)
GNU Compiler Collection (C,C++,OpenMP) for MinGW-w64
local/mingw-w64-x86_64-gcc-libs 10.3.0-5 (mingw-w64-x86_64-toolchain)
GNU Compiler Collection (libraries) for MinGW-w64

torna@staticus MSYS ~
$ pacman -Qs gdb
local/gdbm 1.19-1 (Database)
GNU database library
local/libgdbm 1.19-1 (libraries)
GNU database library
local/mingw-w64-x86_64-gdb 10.2-2 (mingw-w64-x86_64-toolchain)
GNU Debugger (mingw-w64)
```

Új projekt

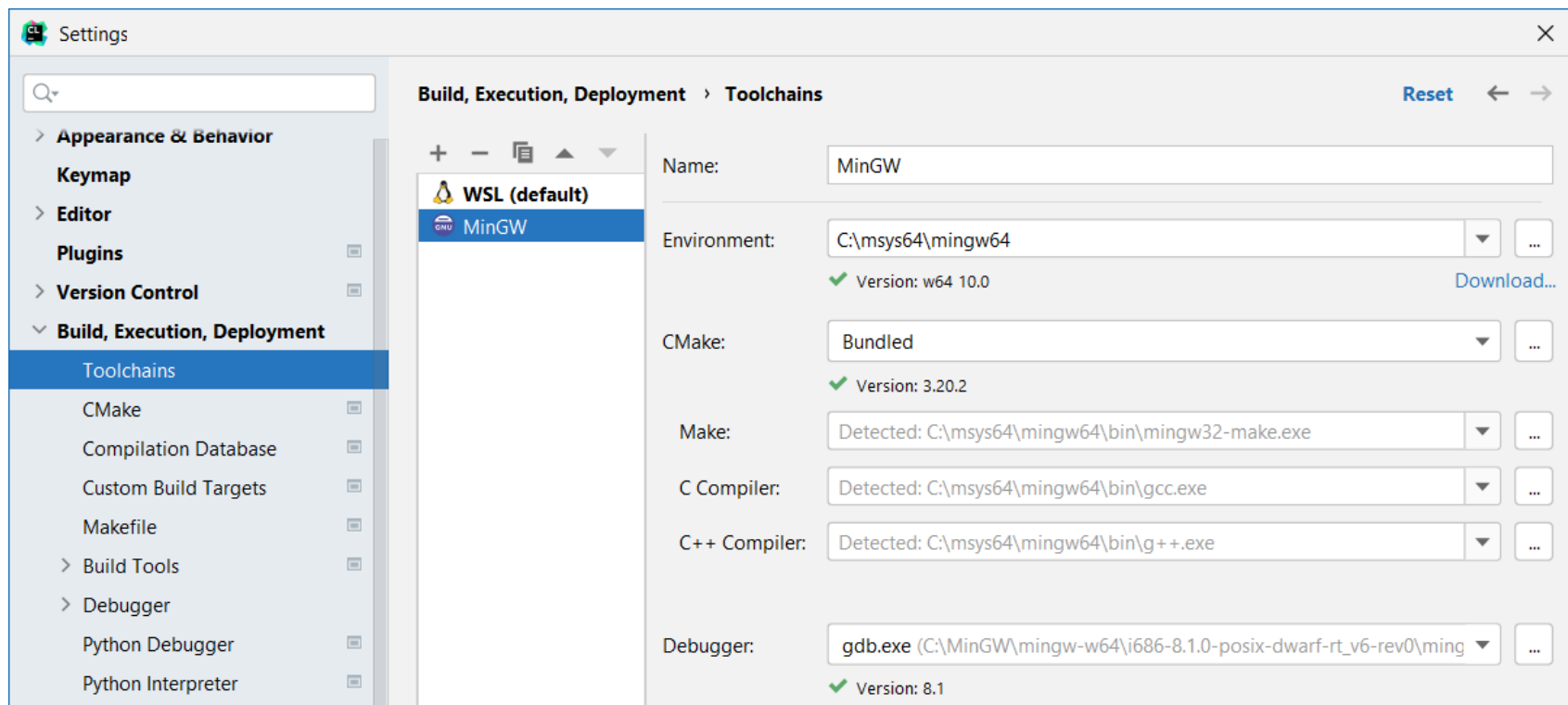


C++ Projekt

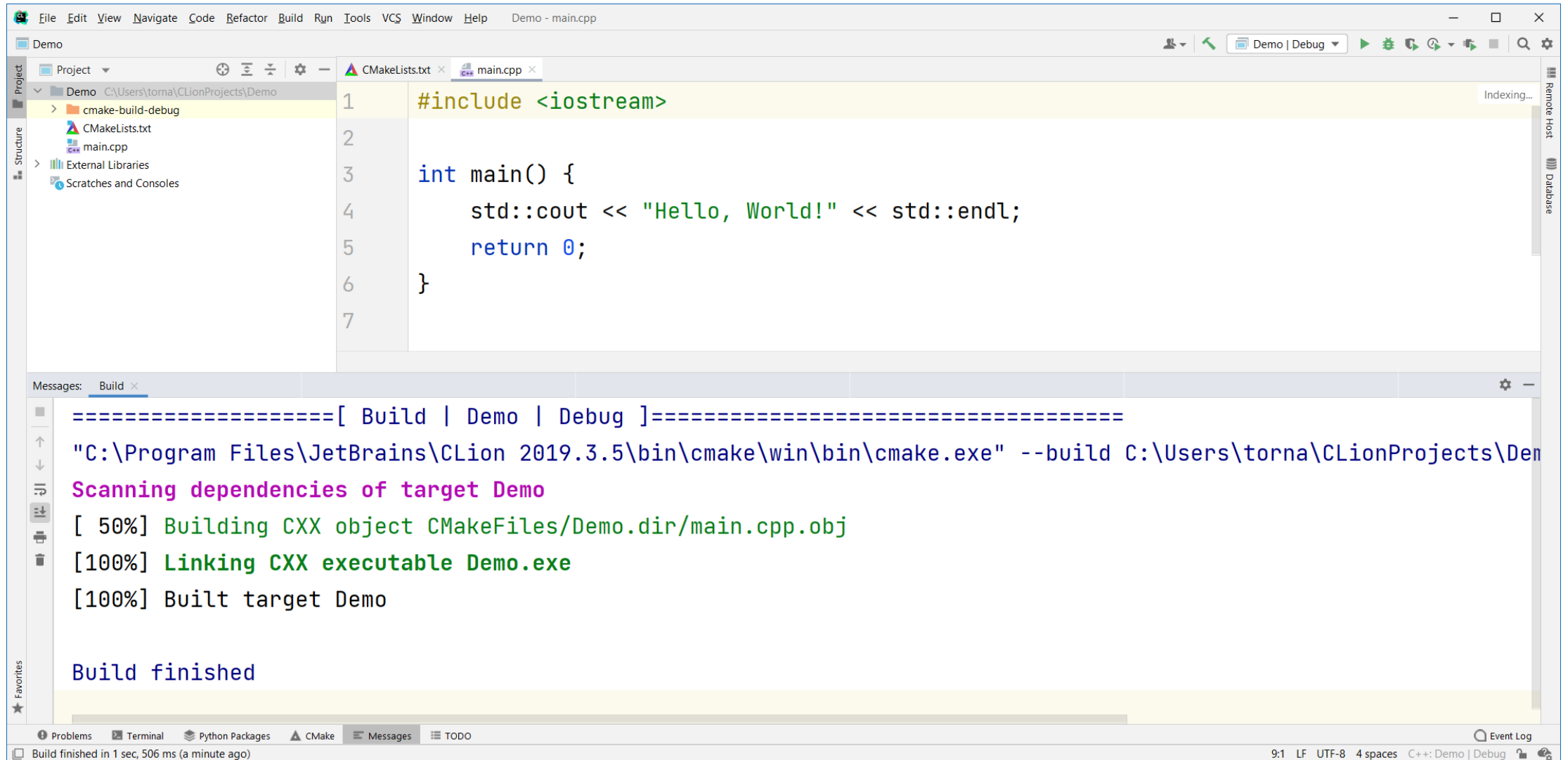


CLion beállítás

- Be kell állítani a fordító elérését:
 - File | Settings | Build, Execution, Deployment | Toolchains
 - + MinGW
 - Environment, CMake és Debugger elérési útvonalakat kell megadni a többi automatikus



Fordítás



The screenshot shows the CLion IDE interface. The main editor displays a C++ file named `main.cpp` with the following code:

```
1 #include <iostream>
2
3 int main() {
4     std::cout << "Hello, World!" << std::endl;
5     return 0;
6 }
7
```

The left sidebar shows the Project structure with the following items:

- Project
- Demo (C:\Users\torna\CLionProjects\Demo)
 - cmake-build-debug
 - CMakeLists.txt
 - main.cpp
 - External Libraries
 - Scratches and Consoles

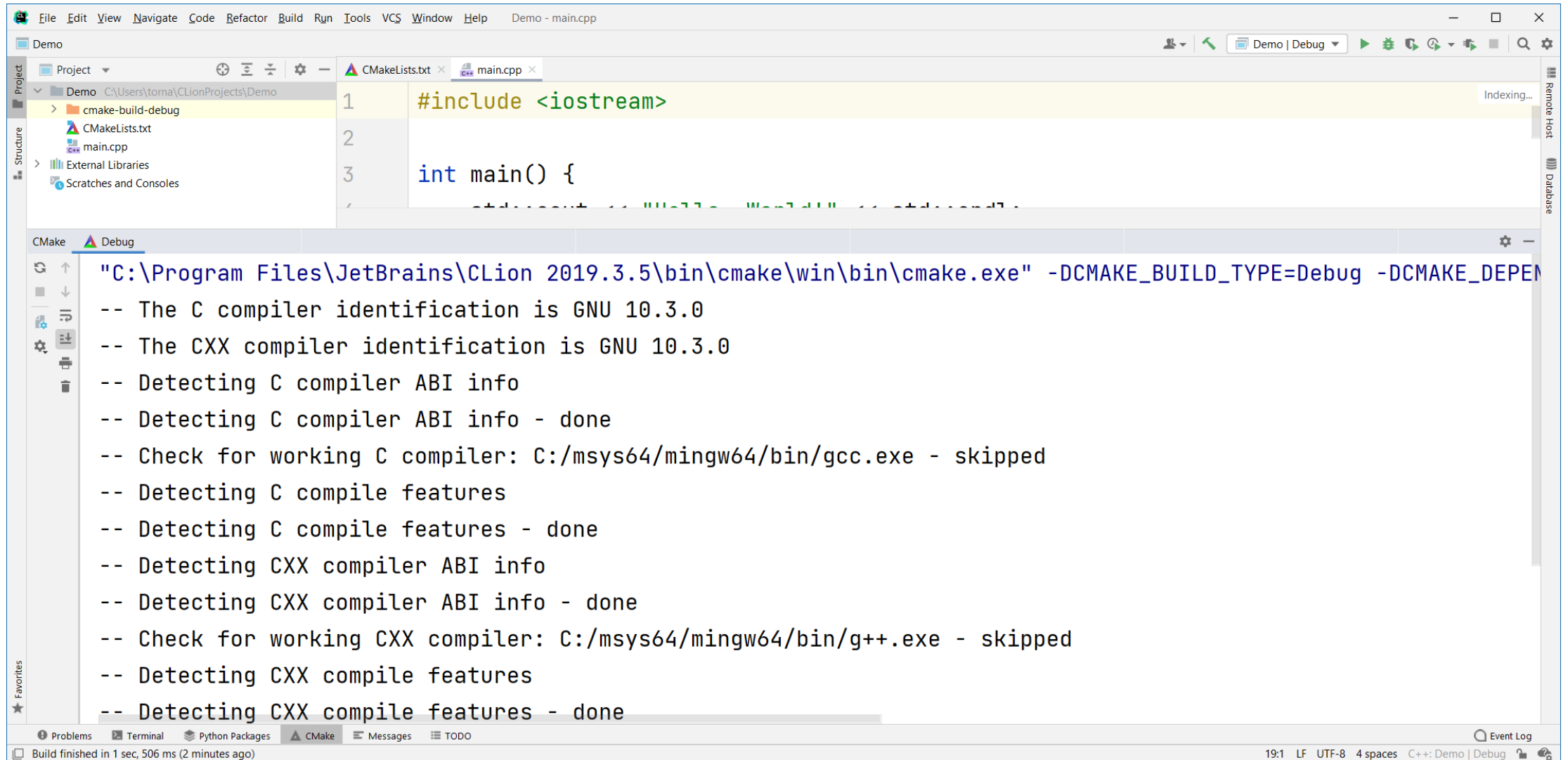
The bottom panel shows the Messages window with the following build output:

```
=====[ Build | Demo | Debug ]=====
"C:\Program Files\JetBrains\CLion 2019.3.5\bin\cmake\win\bin\cmake.exe" --build C:\Users\torna\CLionProjects\Demo
Scanning dependencies of target Demo
[ 50%] Building CXX object CMakeFiles/Demo.dir/main.cpp.obj
[100%] Linking CXX executable Demo.exe
[100%] Built target Demo

Build finished
```

The status bar at the bottom indicates: Build finished in 1 sec, 506 ms (a minute ago). The bottom right corner shows: 9:1 LF UTF-8 4 spaces C++: Demo | Debug.

Compiler ellenőrzése – Cmake fülön



The screenshot shows the CLion IDE interface. The top toolbar includes icons for File, Edit, View, Navigate, Code, Refactor, Build, Run, Tools, VCS, Window, and Help. The main editor displays the CMakeLists.txt file with the following content:

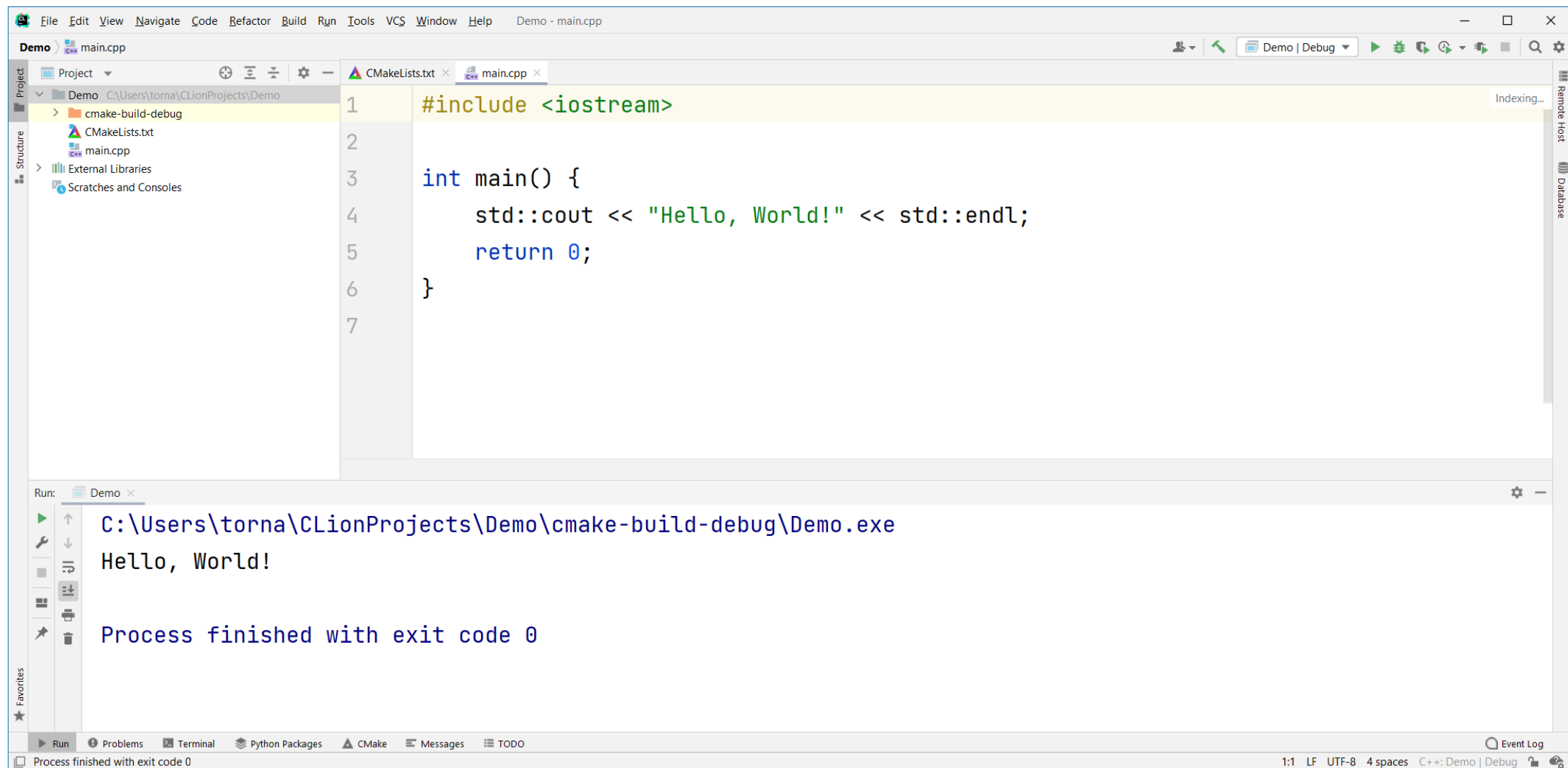
```
1 #include <iostream>
2
3 int main() {
4     std::cout << "Hello, World!" << std::endl;
5 }
```

The bottom panel shows the CMake Debug output, which contains the following text:

```
"C:\Program Files\JetBrains\CLion 2019.3.5\bin\cmake\win\bin\cmake.exe" -DCMAKE_BUILD_TYPE=Debug -DCMAKE_DEPEN
-- The C compiler identification is GNU 10.3.0
-- The CXX compiler identification is GNU 10.3.0
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Check for working C compiler: C:/msys64/mingw64/bin/gcc.exe - skipped
-- Detecting C compile features
-- Detecting C compile features - done
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Check for working CXX compiler: C:/msys64/mingw64/bin/g++.exe - skipped
-- Detecting CXX compile features
-- Detecting CXX compile features - done
```

The status bar at the bottom indicates "Build finished in 1 sec, 506 ms (2 minutes ago)" and "19:1 LF UTF-8 4 spaces C++: Demo | Debug".

Futtatás



The screenshot displays the CLion IDE interface. The main editor window shows a C++ source file named `main.cpp` with the following code:

```
1 #include <iostream>
2
3 int main() {
4     std::cout << "Hello, World!" << std::endl;
5     return 0;
6 }
7
```

The left sidebar shows the project structure for "Demo", including a `cmake-build-debug` directory and the `main.cpp` file. The bottom panel shows the "Run" output for the "Demo" configuration, indicating that the program executed successfully and printed "Hello, World!".

Run: Demo x

```
C:\Users\torna\CLionProjects\Demo\cmake-build-debug\Demo.exe
Hello, World!

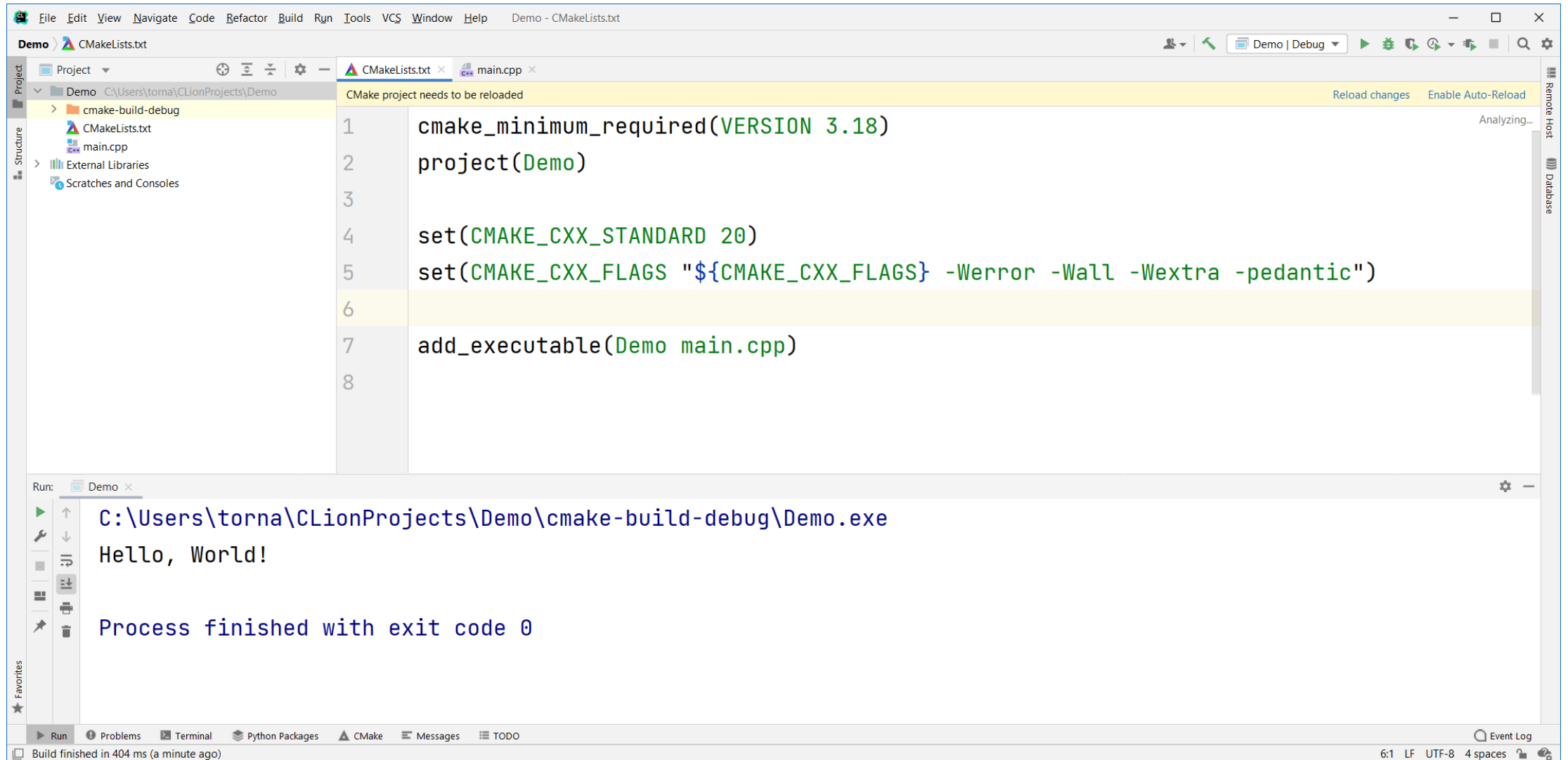
Process finished with exit code 0
```

The status bar at the bottom indicates the file is open at line 1, column 1, using UTF-8 encoding with 4 spaces, and the active configuration is C++: Demo | Debug.

Fordítás II. – CMake

- A CLion a fordítások kezelésére CMake-t használ.
 - Lényegében egy txt-ben leírjuk mi is szükséges a fordításunkhoz
 - És hagyjuk, hogy a CMake generáljon make fájlokat, amik alapján pedig a fordítás megtörténik
 - Az alapértelmezett CMakeLists.txt-ből hiányzik pár kapcsoló, amit kötelező használni a házik beadásakor.
 - Ezeket adjuk hozzá a következő sor beírásával:
 - `set(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} -Werror -Wall -Wextra -pedantic")`

Fordítás II. – CMake



CMake build targets

- A C++ kódok fordításakor a fordító programok nagyon sokat tudnak optimalizálni, hogy gyorsabb gépi kódot generáljanak.
- A CMake segítségével egy projekthez többféle fordítási beállítást tudunk használni, attól függően, épp mi a célunk.
 - Ezek a build target-ek
- Általában 2-3 build targetet szokás meghatározni:
 - **Debug**, hozzá tartozó flagek: -O0 -g, cél: fejlesztés alatt, debugolás (gépi kódból visszavezethető melyik sorban jár épp a kód)
 - **Release**, flagek: -O3, cél: lehető legjobb teljesítmény elérése (O0: compiler nem optimalizál, O3: a compilert megkértük, hogy ahol tud, agresszívan optimalizáljon)
 - **(RelWithDebInfo: -O2 -g** cél: profilozás. A compiler már sok optimalizálást csinál, de még követhető, hol jár a forráskódban.)
 - Ezek a flagek alapból be vannak állítva a CMake megfelelő változójában, ha akarjuk módosíthatjuk.
- Build targetek meghatározása:
 - File | Settings | Build, Execution, Deployment | Toolchains
 - Utána a Run Configuration-ök között választható