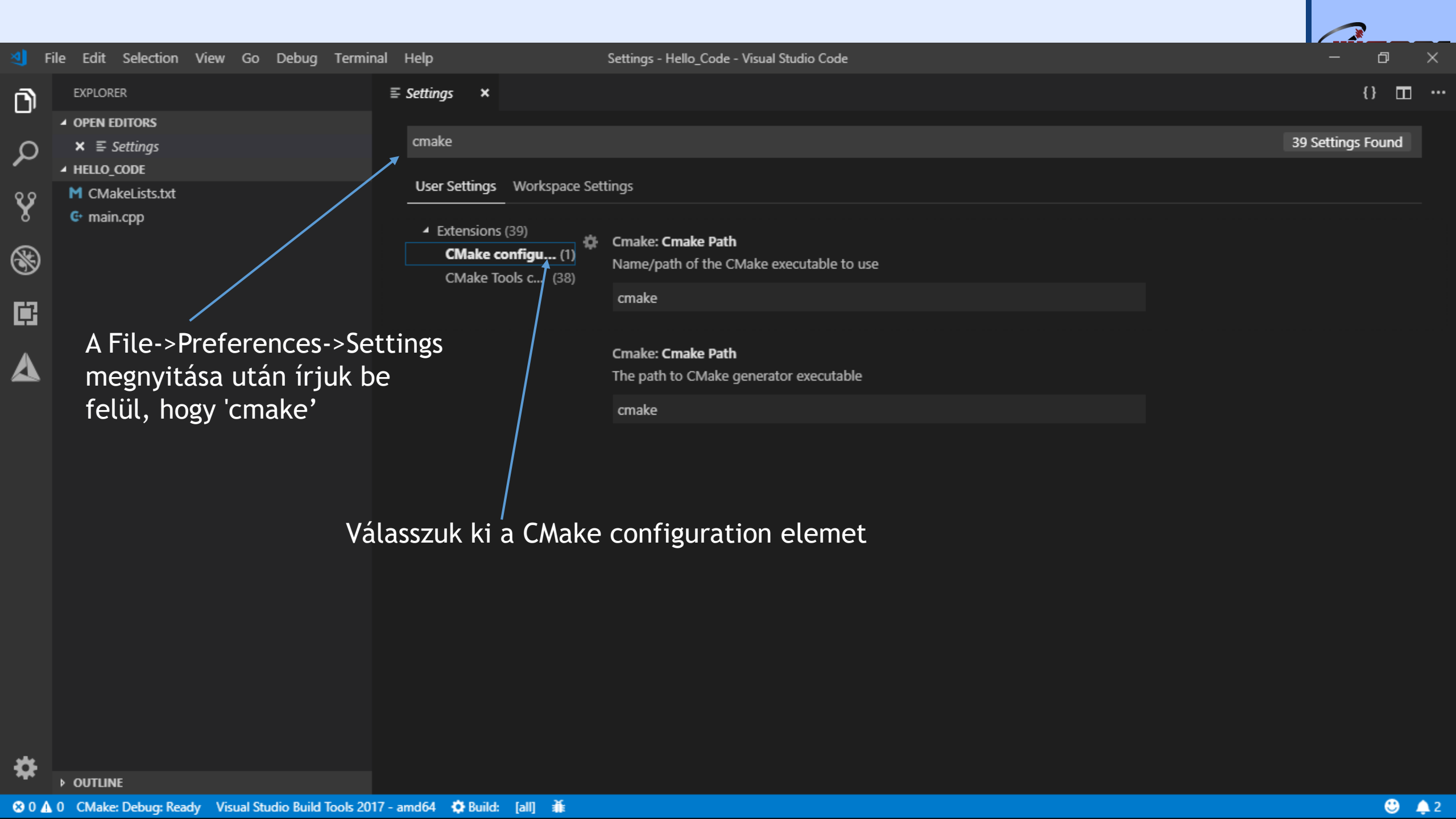
Gyakori hibák
Gyakori kérdések

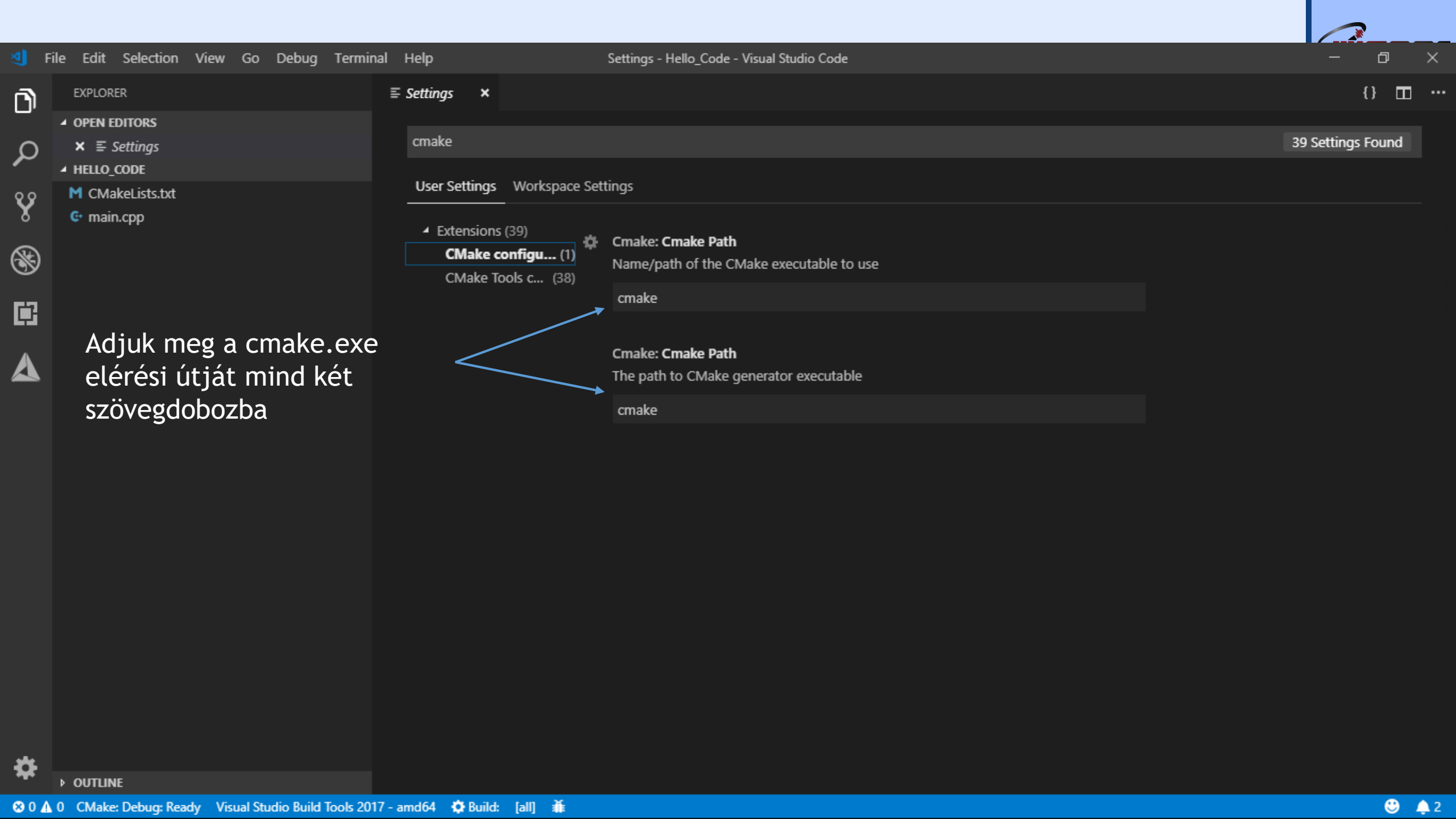
Haladó Alkalmazott Programozás
ELTE, 2019

VS Code és CMake konfigurálás

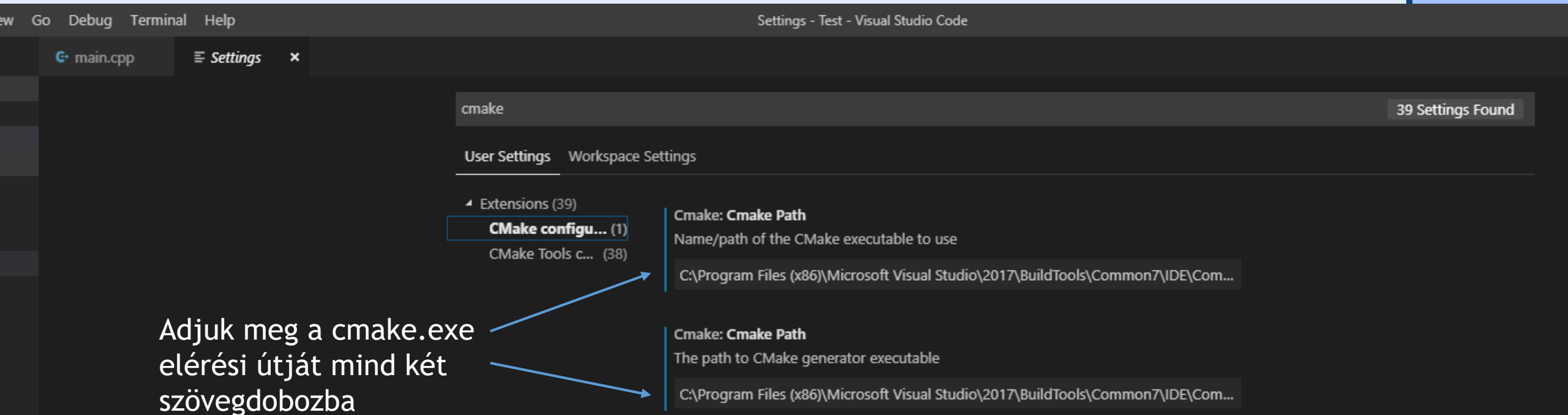
Ha a CMake nincs benne a PATH-ban, akkor a VS Code CMake extension-jének meg kell mondani, hol találja azt.

Ez a tipikus helyzet pl. Windowson, ha a Visual Studio-t vagy a Build Tools-t telepítettük csak.


File Edit Selection View Go Debug Terminal Help
Settings - Hello_Code - Visual Studio Code
EXPLORER
OPEN EDITORS
Settings
HELLO_CODE
CMakeLists.txt
main.cpp
Settings
cmake
39 Settings Found
User Settings
Workspace Settings
Extensions (39)
CMake configu... (1)
CMake Tools c... (38)
Cmake: Cmake Path
Name/path of the CMake executable to use
cmake
Cmake: Cmake Path
The path to CMake generator executable
cmake
A File->Preferences->Settings megnyitása után írjuk be felül, hogy 'cmake'
Válasszuk ki a CMake configuration elemet
OUTLINE
0 0 CMake: Debug: Ready Visual Studio Build Tools 2017 - amd64 Build: [all]
2


File Edit Selection View Go Debug Terminal Help
Settings - Hello_Code - Visual Studio Code
EXPLORER
OPEN EDITORS
Settings
HELLO_CODE
CMakeLists.txt
main.cpp
Settings
cmake
39 Settings Found
User Settings
Workspace Settings
Extensions (39)
CMake configu... (1)
CMake Tools c... (38)
Cmake: Cmake Path
Name/path of the CMake executable to use
cmake
Cmake: Cmake Path
The path to CMake generator executable
cmake
Adjuk meg a cmake.exe elérési útját mind két szövegdobozba
OUTLINE
0 0 CMake: Debug: Ready Visual Studio Build Tools 2017 - amd64 Build: [all]
2

VS Code és CMake konfigurálás



Adjuk meg a cmake.exe elérési útját mind két szövegdobozba

Alapesetben:

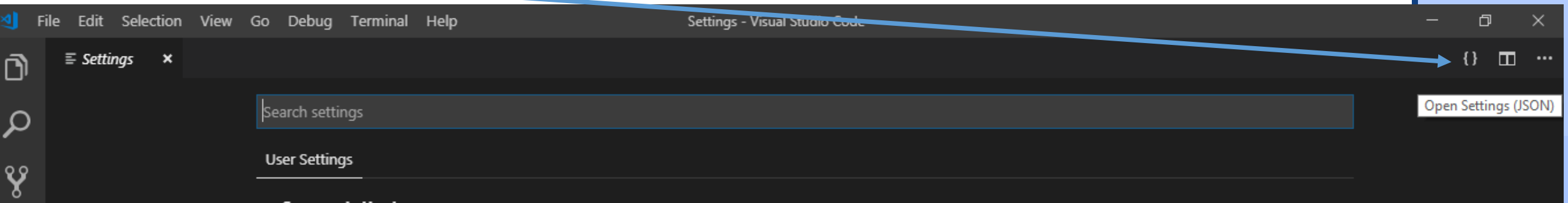
C:\Program Files (x86)\Microsoft Visual Studio\2017\Community\Common7\IDE\CommonExtensions\Microsoft\CMake\CMake\bin\cmake.exe

VS Code és CMake konfigurálás

A módosítások után érdemes újraindítani az alkalmazást

Ha még így sem találja a CMake-t, vagy hibaüzenettel leáll a konfiguráció, lehet, hogy kézzel meg kell adni a Ninja elérési útját is a konfigurációs json fájlban.

- Ehhez nyissuk meg az előző beállítások nézetet újra és kattintsunk jobb felül a {} jelekre, amellyel megnyílik az említett json fájl.



VS Code és CMake konfigurálás

A fájlban lehetőségünk van megadni közvetlenül a cmake elérési útját, illetve a cmake.configureArgs opcióval a ninja elérési útját (pár mappával a cmake.exe mellett van alapesetben).

Az elérési utakban itt \\-t kell elválasztónak írni és nincs sortörés az " " jelek között.

```
{
"cmake.cmakePath": "C:\\Program Files (x86)\\Microsoft Visual
Studio\\2017\\BuildTools\\Common7\\IDE\\CommonExtensions\\Microsoft\\CMake\\CMake\\bin\\cmake.exe",
"cmake.configureArgs": [
"-DCMAKE_MAKE_PROGRAM=C:\\Program Files (x86)\\Microsoft Visual
Studio\\2017\\BuildTools\\Common7\\IDE\\CommonExtensions\\Microsoft\\CMake\\Ninja\\ninja.exe"
]
}
```

VS Code és a fordítók konfigurálása

Ha telepítettünk egy új fordítót, újra kell szkennelni a rendszert, hogy a cmake tudjon róla

1. Nyissuk meg a parancs palettát (Ctrl+Shift+P),
2. írjuk be, hogy`cmake`, görgessünk le, és válasszuk ki a`scan for kits` parancsot.

VS Code és a fordítók konfigurálása

Ha nem találja a CMake a kívánt fordítót, kézzel hozzáadhatjuk a megfelelő konfigurációs json fájl módosításával:

1. Nyissuk meg a parancs palettát (Ctrl+Shift+P),
2. írjuk be, hogy`cmake` és válasszuk ki a`CMake: Edit user-local CMake kits` parancsot.
3. A megnyíló fájlba lehet kit-eket definiálni.

VS Code és a fordítók konfigurálása

Példa: az alap Visual Studio kit:

```
{
    "name": "Visual Studio Community 2017 - amd64",
    "visualStudio": "VisualStudio.15.0",
    "visualStudioArchitecture": "amd64",
    "preferredGenerator": {
        "name": "Visual Studio 15 2017",
        "platform": "x64"
    }
}
```

VS Code és a fordítók konfigurálása

Példa: Clang a Visual Studio kittel.

```
{
    "name": "Clang 8.0.0 for MSVC with Visual Studio Community 2017 (amd64)",
    "visualStudio": "VisualStudio.15.0",
    "visualStudioArchitecture": "amd64",
    "compilers": {
        "C": "C:\\Program Files\\LLVM\\bin\\clang-cl.exe",
        "CXX": "C:\\Program Files\\LLVM\\bin\\clang-cl.exe"
    }
}
```

Itt adjuk meg a fordító elérési útját a saját rendszerünkben!

Új CMake projekt VS Code-ban

- A VS Code-ban File->Open-el nyissunk meg egy mappát (akár a dialógus ablakban is létrehozhatunk új mappát jobb klikkel)
- A parancs palettába (Ctrl+Shift+P) írjuk be, hogy CMake, és a lehetőségek közül válasszuk a Quick Start-ot.

- Ez meg fogja kérdezni, hogy
  - melyik fordító Kit-et használja
    (Windows alatt a Visual Studio / Build Tools amd64 kitje ajánlott)
  - Mi legyen a projekt neve
  - És hogy könyvtárat, vagy futtathatót akarunk létrehozni (általában utóbbit)
  - Eredményül létrejön a mappában egy main.cpp és egy CMakeLists.txt

Új CMake projekt VS Code-ban

További leírások a Visual Studio Code és a CMake konfigurálásáról itt:

https://github.com/Wigner-GPU-Lab/Teaching/tree/master/CMake/Lesson4_ConfiguringVSCode

C++ gyakori hibák

Függvényben függvényt nem lehet bevezetni.

```
int main()
{
  int sq2(int x){ return x*x; }

  return sq2(x);
}
```

Ez nem megy!

C++ gyakori hibák

Függvényben függvényt nem lehet bevezetni.
De lambdát igen!

```
int main()
{
  auto sq2 = [](int x){ return x*x; };

  return sq2(x);
}
```

Ez viszont megy!

C++ gyakori hibák

Vezérlési szerkezeteknél a ( )-után és a { } után nem kell pontosvessző!

```
if()
{
}
else if()
{
}
else
{
}
```

Az egyetlen kivétel a do…while ciklus:

```
do
{
}while(    );
```

C++ gyakori hibák

Szimbólumokat legalább előre deklarálni kell, mielőtt használni lehet őket!

```
int main()
{
  return sqi(5);
}

int sqi(int x){ return x; }
```

Ez nem megy!

```
int sqi(int);

int main()
{
  return sqi(5);
}

int sqi(int x){ return x; }
```

Ez viszont megy!

C++ gyakori kérdések

`std::cout` formázás:

A `cout` ki tudja írni a beépített típusokat megfelelően minden extra nélkül:

```
int a = -4;
unsigned int b = 8;
float c = 3.1415;
double d = 0.70715432135465412;
std::cout << a << " " << b << " " << c << " " << d << "\n";
```

A finomabb testreszabás nem túl elegáns
(még a `printf` is szebb, de az nem típusbiztos), itt érdemes nézelődni: iomanip, io_base tagfüggvények: precision, width

C++ gyakori hibák

Amikor csak lehet, adjunk értéket a változóknak azonnal létrehozáskor:

```
int main()
{
  int a;
  a = 5;
  return a;
}
```

Hibalehetőség!

```
int main()
{
  int a = 5;
  return a;
}
```

Ez az ajánlott!

Ha a változó véletlen inicializálatlan marad, akkor nem definiált, hogy milyen érték van benne, egyáltalán nem biztos, hogy az nulla, vagy bármi más, és ha elkezdjük használni, elszabadul a pokol...

C++ gyakori hibák

C++ vs C headerek:

`#include <math.h>` Ez a C-s változat, ami nem ajánlott (deprecated)

`#include <cmath>` Ez a C++ változat, ami ajánlott

Bővebben: itt

C++ gyakori hibák

A lebegőpontos konstansok normál alakban gondolkoznak, de lehet 1-nél nagyobb számot is írni, csak figyeljünk arra, hogy ez mit is jelent!:

```
#include <iostream>
int main()
{
  std::cout.precision(16);
  std::cout << 1e-3 << "\n";
  std::cout << 10e-3 << "\n";
  std::cout << 100e-3 << "\n";
  return 0;
}
```

Ezeknek a kimenete:

```
0.001
0.01
0.1
```

C++ gyakori hibák

Abszolút érték függvényből sajnos több is van:

```
#include <cmath>
```

`abs`	Ez a C-s változat, ami egészekkel számol
`std::fabs`	Ez a C++ változat, ami csak lebegőpontos számokkal számol
`std::labs`	Ez a C++ változat, ami egész értékekkel számol
`std::abs`	Ez is C++ változat, ami az argumentum típusának megfelelő abszulút értéket számolja

C++ gyakori hibák

Abszolút érték függvényből sajnos több is van:

```
#include <cmath>
```

`abs`

`std::fabs`

`std::labs`

`std::abs` ← Ajánlott ezt használni!

C++ gyakori hibák

Ha header fájlt írunk, akkor:

- Pontosan azokat a headereket includeoljuk, amelyek mindenképp szükségesek az adott header funkcionalitásához. Ne többet, ne kevesebbet!

- A headerben ne használjunk globális using namespace-t, mert ezzel a header-t include-oló felhasználó névterét szennyezzük, amit ő nem tud könnyen megakadályozni, vagy visszafordítani.
- A header fájlok nevei kisbetű-nagybetű érzékenyek linuxon, windows-on viszont nem!