NPRG041 – C++

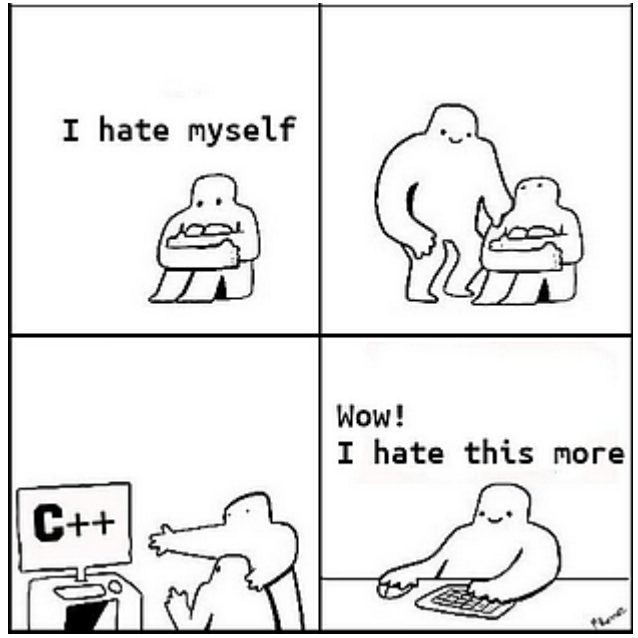
cvičení – Jiří Klepl

mattermost: ulita/2425ZS: nprg041-cpp-klepl (inv na SIS nástěnce)

Klepl@d3s.mff.cuni.cz

Agenda

- Cmake
- find_package
- FetchContent
- vcpkg



CMake – základy

```
cmake_minimum_required(VERSION 3.20)

project("Simple Application"
    VERSION 0.1
    DESCRIPTION "Demonstration of CMake"
    LANGUAGES CXX
)

# Set the C++ standard to C++23 (globally)
set(CMAKE_CXX_STANDARD 23) # suggest using C++23
set(CMAKE_CXX_STANDARD_REQUIRED True) # Enforce C++23
set(CMAKE_CXX_EXTENSIONS OFF) # Disable extensions

# Add include directory (globally)
include_directories(include)

# Add executable
add_executable(application src/main.cpp
                           features/cool_feature.cpp)
```

CMake – základy

```
cmake_minimum_required(VERSION 3.20)

project("Simple Application"
    VERSION 0.1
    DESCRIPTION "Demonstration of CMake"
    LANGUAGES CXX
)

# Set the C++ standard to C++23 (globally)
set(CMAKE_CXX_STANDARD 23) # suggest using C++23
set(CMAKE_CXX_STANDARD_REQUIRED True) # Enforce C++23
set(CMAKE_CXX_EXTENSIONS OFF) # Disable extensions

# Add include directory (globally)
include_directories(include)

# Add executable
add_executable(application src/main.cpp
                           features/cool_feature.cpp)
```

Minimální ç̣ǎlê ŵêsș̣îǫ̣ŋ

Informace o projektu

CMake – základní použití

- **Začínáme ve složce se CMakeLists.txt** a vytvoříme **podsložku build**
- Vejdeme do složky build a spustíme konfiguraci: **cmake -DCMAKE_BUILD_TYPE=<type> ..** („..“ je cesta k projektu)
  - Build typ typicky: **Debug**, **Release**, **RelWithDebInfo** (tenhle např. na profiling)
- Potom konečně sestavení: **cmake --build .**
  - Useful optiony:
    - **-j [<jobs>]** – multithreaded kompilace
    - **--clean-first** – vyčištění cílové složky
    - **-t <target>** – sestavení pouze zadaného cíle
- Pokud chceme testovat (musíme definovat v cmake filu), tak: **ctest**

CMake – základní direktivy pro definici targetu

- **add_executable(<name> <sources>...)**
  - Vytvoří spustitelný soubor
- **add_library(<name> [STATIC | SHARED | MODULE] <sources>...)**
  - „Normální“ knihovna – staticky linkovaná, dynamicky, dynamicky na vyžádání
- **add_library(<name> INTERFACE <sources>...)**
  - „Interface“ knihovna – např. když je „header-only“, negeneruje žádné objekty
    - To se nám hodí např. na definici generických std-like knihoven – ty je potřeba generovat podle zadaných typů, nejde je předvyrobit
- **add_custom_target(<name> <command with args>)**
  - Typicky použijeme, pokud target vygeneruje nějaký script

CMake – základní konfigurace targetů

- Pro většinu těchto direktiv máme globální a target-specific verzi
  - Často se hodí každý target nakonfigurovat jinak; jindy chceme všechny nakonfigurovat stejně
  - **set** a **set_target_properties, include_directories** a **target_include_directories**
- Zde seznam těch hlavních (target-specific verze):
  - **target_include_directories** – kde má target hledat header fily
  - **target_link_libraries** – co chceme k targetu přilinkovat za knihovnu (přidá i potřebné includy)
  - **set_target_properties** – tím můžeme nastavit standard atd (viz předchozí slajd)
  - **target_compile_options** – zde můžeme nastavovat warningy atd. (*compiler-specific*)
  - **target_compile_definitions** – nastavení definů: MY_FLAG WITH_VALUE=1
    - Přeloží se na compilerem podporované optiony, pro gcc: -DMY_FLAG -DWITH_VALUE=1
- JE potřeba znát rozdíl mezi PRIVATE a PUBLIC: PRIVATE konfigurace se nepředává dál (třeba kdybychom dělali library a měli bychom header file s definicí něčeho, co nemá být v API: **target_include_directories(target PRIVATE header.hpp)**

CMake – externí projekty

- **find_package(<package> [REQUIRED])**
  - + spousta optional argumentů: často doporučeny podle externího projektu
  - Přidá externí dependenci
    Na to většinou navážeme: **target_link_libraries(target PRIVATE package::...)**
  - Ten **package musí být nainstalován** (ukážeme si přenositelně s **vcpkg**, ale třeba na Ubuntu/Debian stačí nainstalovat přes apt)
- Pro malé/specifické projekty můžeme použít (místo find_package):

```
Include(FetchContent)
FetchContent_Declare(
  <package_name>
  GIT_REPOSITORY <url, např.: https://github.com/...>
  GIT_TAG <git branch, nebo tag name (release)>
)
FetchContent_MakeAvailable(<package_name>)
```

  - To package stáhne do cílové složky jako dependenci
  - Typicky ho přímo na místě i sestaví (zatímco u find_package už je nainstalován)
  - Bývá to mnohem jednodušší na použití, pokud ten package nevyžaduje něco nainstalovat

find_package(<package> [REQUIRED])

- + spousta optional argumentů: často doporučeny podle externího projektu
- Přidá externí dependenci
  Na to většinou navážeme: **target_link_libraries(target PRIVATE package::...)**
- Ten **package musí být nainstalován** (ukážeme si přenositelně s **vcpkg**, ale třeba na Ubuntu/Debian stačí nainstalovat přes apt)

Include(FetchContent)

- Pro malé/specifické projekty můžeme použít (místo **find_package**):
  FetchContent_Declare(
  <package_name>
  GIT_REPOSITORY <url, např.: https://github.com/...>
  GIT_TAG <git branch, nebo tag name (release)>
)
  FetchContent_MakeAvailable(<package_name>)
- Stáhne package do cílové složky jako dependenci
  - Přímo na místě i sestaví (zatímco u **find_package** už je nainstalován)

vcpkg

- https://learn.microsoft.com/en-us/vcpkg/get_started/get-started
- https://vcpkg.io/en/packages
- Jednoduchá instalace C++ packagů mimo systém
- Ale občas je potřeba něco doinstalovat – pak poradí přesně jak
- Spolupracuje s nástrojem Cmake **(ujistěte se, že vcpkg je v PATH)**:
  - CMake musíme při konfiguraci říct, kde hledat package (přidáme option):

    -DCMAKE_TOOLCHAIN_FILE=[vcpkgroot]/scripts/buildsystems/vcpkg.cmake

- Pak můžeme pohodlně používat **find_package(...)** s packagi ve vcpkg

Příklad: labs/dvd

- Jednoduchý CMake projekt s externí dependencí na **sfml**
  - sfml: https://www.sfml-dev.org/ https://vcpkg.io/en/package/sfml
- Úkol: aplikace otevře okno, ve kterém bude jezdit obrázek (ten se bude odrážet od hran okna)
  - Obrázek může být libovolné logo/nápis/...

https://www.google.com/search?q=dvd+bouncing+logo