NPRG 041 – cvičení Programování v C++

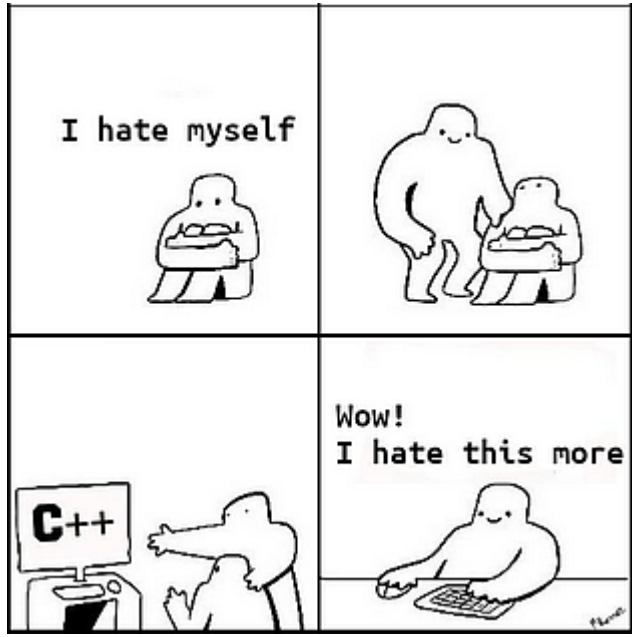
Jiří Klepl

mail:
klepl@d3s.mff.cuni.cz

mattermost:
https://ulita.ms.mff.cuni.cz/mattermost/ar2324zs/channels/nprg041-cpp-klepl

Agenda

- Organizační záležitosti
- Nástroje a dodatečné zdroje
- Mini-úloha: Hello, user
- První úloha: Násobilka



Obecné informace

- Na cvičení budeme spolu řešit základy praxe v **moderním** C++ (20+)
  - Ale naučit se lze jenom **pokusem a omylem**
- O čem je C++ cvičení – na co se budeme zaměřovat
  - **Portabilní a čitelný kód, "good practices", efektivita** (nepočítat zbytečně)
- Co když mám s něčim problém (klidně i z přednášky)?
  - Můžu položit **otázku na cvičení**
    - Možnost krátké diskuze o souvisejících tématech
  - Můžu napsat **otázku na mattermost nebo mailem**
    - Pro větší problémy jde domluvit konzultaci
  - Nebojte se ptát, problémy je nejlepší vyřešit co nejrychleji

Organizace cvičení

- **Úlohy na cvičeních**: vypracování probíraných úloh a jejich nahrání na GitLabovou skupinu
  - Přítomnost na cvičení není povinná, ale výrazně zjednoduší plnění úloh
  - Odevzdání úloh vždy do půlnoci před následujícím cvičením pro **feedback**
- **Průběžné 2-3 domácí úkoly** v Recodexu (Nepovinné)
  - neovlivňují známku ani udělení zápočtu, ale jsou nejlepším zdrojem feedbacku
- **Závěrečná domácí úloha** v Recodexu
  - Bude se od ní odvíjet zápočtový test (rozšíření úlohy)
- **Zápočtový test**: poslední cvičení (**POVINNÉ!**), jeden opravný termín
  - Úprava závěrečné úlohy na školním počítači bez externí pomoci
  - Nutná podmínka pro připuštění ke zkoušce i udělení zápočtu

Zápočtový program

1. **Návrh zadání do poloviny listopadu** (stručné, stačí hlavní myšlenka)
2. **Schválení zadání do 30.11.** – specifikace, rozhraní, externí knihovny
3. **Technické demo do konce výuky ZS** – ukázka automatizovaného sestavení včetně všech externích knihoven pomocí CMake, **multiplatformnost**, základní funkčnost "naoko"
4. **Odevzdání kompletního programu do 30.4.** (první pokus, feedback)
   - Program by se měl (polo-)automatizovaně sestavit po stažení z repositáře
   - Kompletní funkcionalita programu, bude fungovat multiplatformě
   - **Finální odevzdání i s dokumentací do konce výuky v LS**

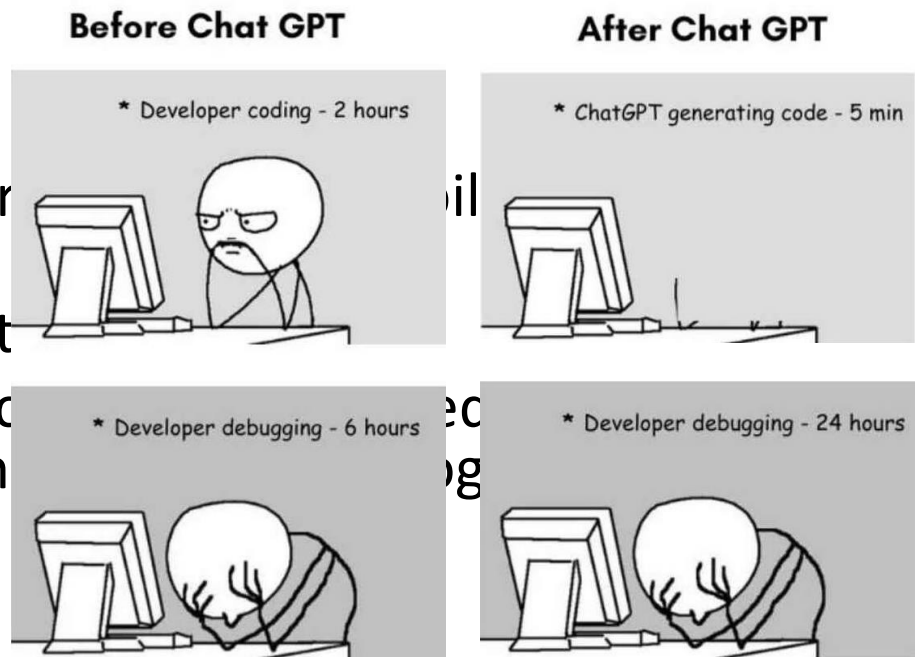
Vývoj v GitLabu, samostatný commit pro každou podstatnější změnu, např. přidání nové komponenty řešení (nebo alespoň po každém dnu vývoje)

Zkouška

- Praktickou formou na **školních počítačích** v laboratoři
- Je zadána úloha prověřující znalosti používání jazyka **a kvality kódu**
  - Odevzdávání do Recodexu (testy budou prověřovat základní funkčnost)
  - Lze ztratit body za prohřešky nerozpoznávány testem
    - Skryté bugy: např. memory leaky, corner-casy v algoritmu řešení, …
    - Kvalita: nečitelnost kódu, výrazná neefektivita řešení, chybné použití prvků jazyka, …
- Celá známka vychází právě ze zkoušky
- Lze podstoupit až **po zápočtovém testu**
- **Žádná ústní část prověřující teorii**

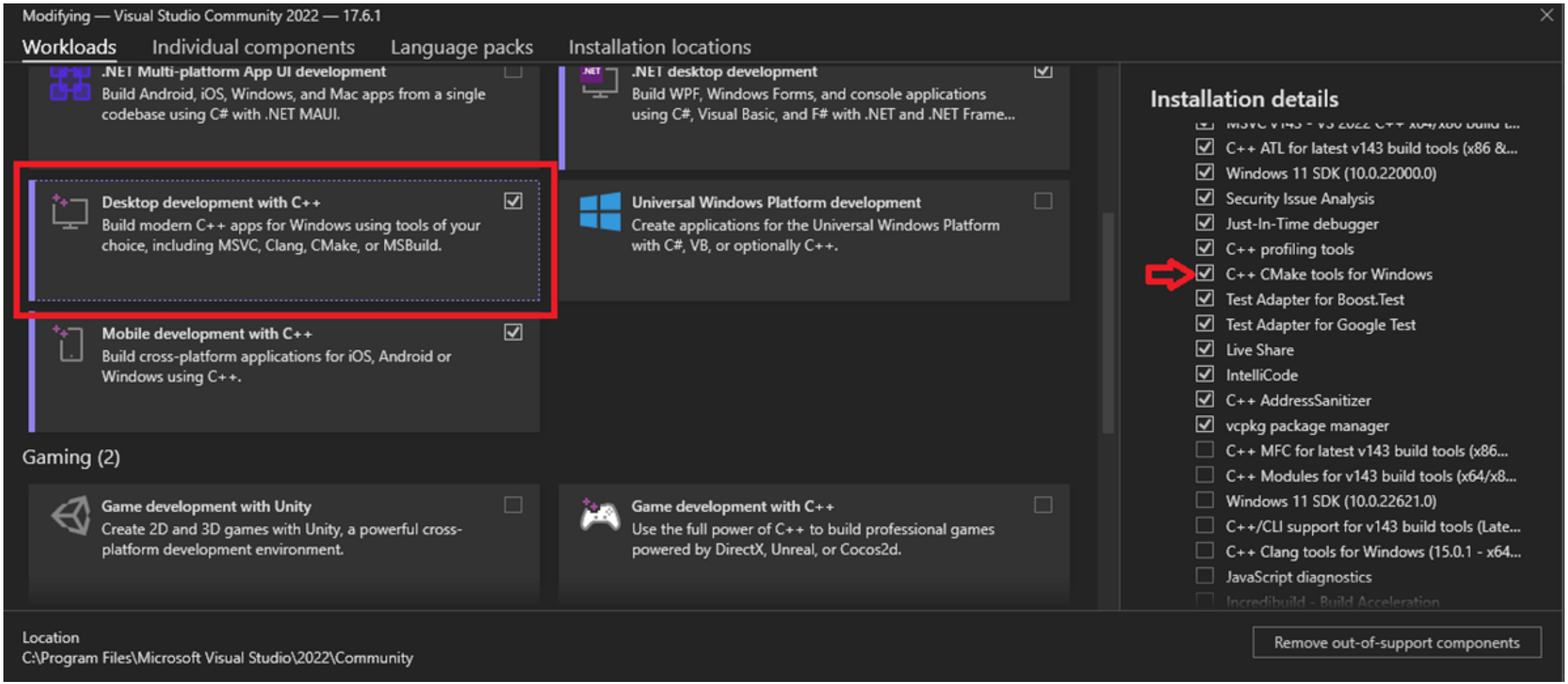
AI

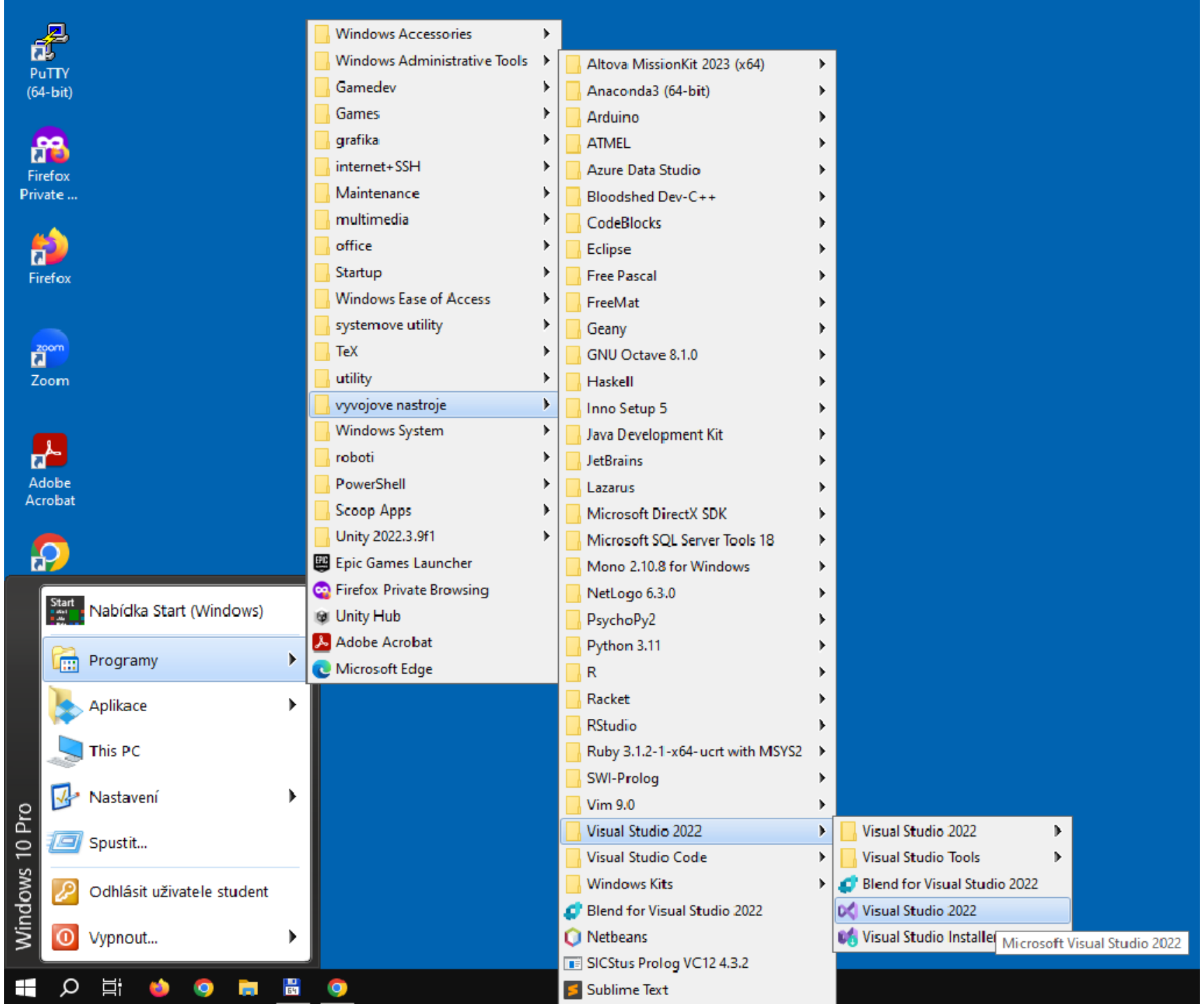
- Používání AI modelů je povoleno pro úlohy i domácí úkoly
- AI NENÍ POVOLENO NA ZÁPOČTOVÉM TESTU ANI ZKOUŠCE
- **Žádným** AI modelům často nejde věřit
  - AI modely často „halucinují“ – když neví odpověď, vymyslí si jí
  - Jsou naučeny na zastaralém/nekvalitním kódu
- Na co jsou AI modely supr
  - **Zvýšení efektivity** psaní „nudného“ kódu (zde opr il
  - **Inspirace** – AI model viděl víc kódu než uživatel a elegantnější/jednodušší postupy okoukané od ost
  - **Cizí pohled** – může dávat programátorovi feedbac ec kódu (model kódu rozumí nebo nerozumí); má jin g



Visual Studio IDE

- **Stačí Community verze**
- Nainstalovat "**Desktop Development with C++**" a "**CMake tools**"
- Bude se používat na cvičeních, ale není nutností
- Dobré alternativy:
  - Visual Studio Code + compiler (e.g. GCC, Clang, …)
    - To používám já
    - VS Code není IDE, ale poskytuje mnoho rozšíření, kterými se mu dokáže vyrovnat
    - Je nutno nastavit (a doinstalovat) compiler - dokáže pracovat i s několika
  - CLion IDE (používá Clang)
  - … (opravdu jich je mnoho)


Modifying — Visual Studio Community 2022 — 17.6.1
Workloads
Individual components
Language packs
Installation locations
.NET Multi-platform App UI development
Build Android, iOS, Windows, and Mac apps from a single codebase using C# with .NET MAUI.
.NET desktop development
Build WPF, Windows Forms, and console applications using C#, Visual Basic, and F# with .NET and .NET Frame...
Desktop development with C++
Build modern C++ apps for Windows using tools of your choice, including MSVC, Clang, CMake, or MSBuild.
Universal Windows Platform development
Create applications for the Universal Windows Platform with C#, VB, or optionally C++.
Mobile development with C++
Build cross-platform applications for iOS, Android or Windows using C++.
Gaming (2)
Game development with Unity
Create 2D and 3D games with Unity, a powerful cross-platform development environment.
Game development with C++
Use the full power of C++ to build professional games powered by DirectX, Unreal, or Cocos2d.
Installation details
C++ ATL for latest v143 build tools (x86 &...
Windows 11 SDK (10.0.22000.0)
Security Issue Analysis
Just-In-Time debugger
C++ profiling tools
C++ CMake tools for Windows
Test Adapter for Boost.Test
Test Adapter for Google Test
Live Share
IntelliCode
C++ AddressSanitizer
vcpkg package manager
C++ MFC for latest v143 build tools (x86...
C++ Modules for v143 build tools (x64/x8...
Windows 11 SDK (10.0.22621.0)
C++/CLI support for v143 build tools (Late...
C++ Clang tools for Windows (15.0.1 - x64...
JavaScript diagnostics
Location
C:\Program Files\Microsoft Visual Studio\2022\Community
Remove out-of-support components


PuTTY (64-bit)
Firefox Private ...
Firefox
Zoom
Adobe Acrobat
Windows Accessories
Windows Administrative Tools
Gamedev
Games
grafika
internet+SSH
Maintenance
multimedia
office
Startup
Windows Ease of Access
systemove utility
TeX
utility
vyvojove nastroje
Windows System
roboti
PowerShell
Scoop Apps
Unity 2022.3.9f1
Epic Games Launcher
Firefox Private Browsing
Unity Hub
Adobe Acrobat
Microsoft Edge
Altova MissionKit 2023 (x64)
Anaconda3 (64-bit)
Arduino
ATMEL
Azure Data Studio
Bloodshed Dev-C++
CodeBlocks
Eclipse
Free Pascal
FreeMat
Geany
GNU Octave 8.1.0
Haskell
Inno Setup 5
Java Development Kit
JetBrains
Lazarus
Microsoft DirectX SDK
Microsoft SQL Server Tools 18
Mono 2.10.8 for Windows
NetLogo 6.3.0
PsychoPy2
Python 3.11
R
Racket
RStudio
Ruby 3.1.2-1-x64-ucrt with MSYS2
SWI-Prolog
Vim 9.0
Visual Studio 2022
Visual Studio Code
Windows Kits
Blend for Visual Studio 2022
Netbeans
SICStus Prolog VC12 4.3.2
Sublime Text
Visual Studio 2022
Visual Studio Tools
Blend for Visual Studio 2022
Visual Studio 2022
Visual Studio Installe
Microsoft Visual Studio 2022
Start
Nabídka Start (Windows)
Programy
Aplikace
This PC
Nastavení
Spustit...
Odhlásit uživatele student
Vypnout...
Windows 10 Pro

Na co si dávat pozor při sestavování

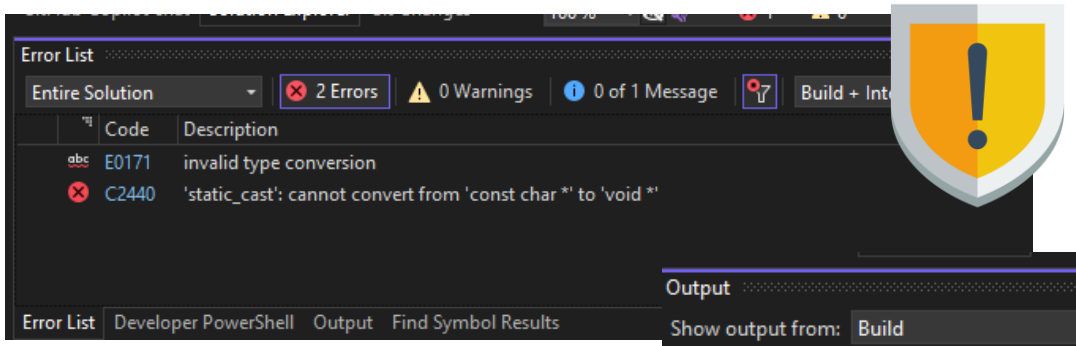
- **C++ standard** (většina compilátorů nedefaultuje na C++20 ani C++23)
- Nejlepší dva compilery (pokud jde o pokrytí C++ featur)
  - MSVC 19.28+ (ověřit v *Developer PowerShell for VS*: `cl --version`)
    - Součást Visual Studio 2022
    - Kliknout project > Properties > C/C++ > Language > C++ Language Standard => /std:c++20
    - Pozor na přepínač Debug/Release (Debug verze může být OPRAVDU pomalejší)
  - GCC 13+ (ověřit: `g++ --version`)
    - Nastavení standardu (to samé clang): `-std=c++20`
- Nastavení warning levelu: `/W4` (msvc) `-Wall -Wextra` (gcc, clang)
- Nastavení optimalizací (VS to řeší za nás; manuálně `/O2` a `-O2`)

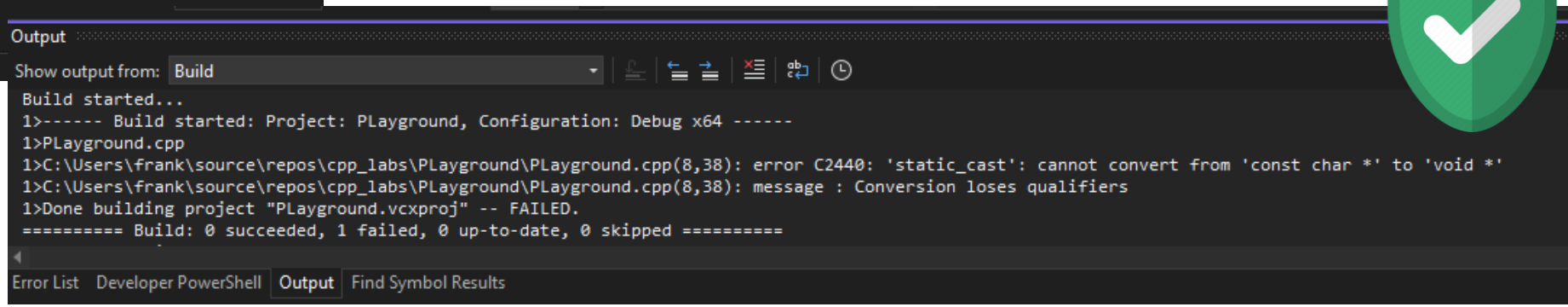
CheatSheat pro nastavení

- **Set the C++ language standard version.**
  - Right click project > Properties > C/C++ > Language > C++ Language Standard => /std:c++20 or latest.
  - Equivalent of `` `-std=c++20` `` in GCC/Clang.
- **Add command line args to your debugged program.**
  - Right click project > Properties > Debugging > Command Arguments => `` `-t -v --some=cool` ``.
- **Set up a warning level.**
  - Right click project > Properties > C/C++ > General > Warning Level => W4.
  - (Somewhat) equivalent of `` `-Wall` `` in GCC/Clang.
- **Add additional include directories.**
  - Right click project > Properties > C/C++ > General > Additional Include Directories => Add the desired dirs.
  - Equivalent of `` `-I <some dir>` `` in GCC/Clang.
- **Add additional directories where to look for libs for linking.**
  - Right click project > Properties > Linker > General > Additional Library Directories=> Add the desired libs.
  - Equivalent of `` `-L /opt/libdir/` `` in GCC/Clang.
- **Add additional libraries to link with.**
  - Right click project > Properties > Linker > Input > Additional Dependencies => e.g. somelib.lib.
  - Equivalent of `` `-l somelib` `` in GCC/Clang.

Debugging ve Visual Studiu (Compilation errory)

- Dívejte se do „Output“ okna, ne „Error List“
  - Output se objeví po spuštění kompilace (ctrl + shift + B)
  - Dvojklik skočí na místo v kódu, kde vznikla chyba
- Vždy se zaměřte na opravu první chyby v „Output“
  - Další chyby mohly vzniknout kvůli té první



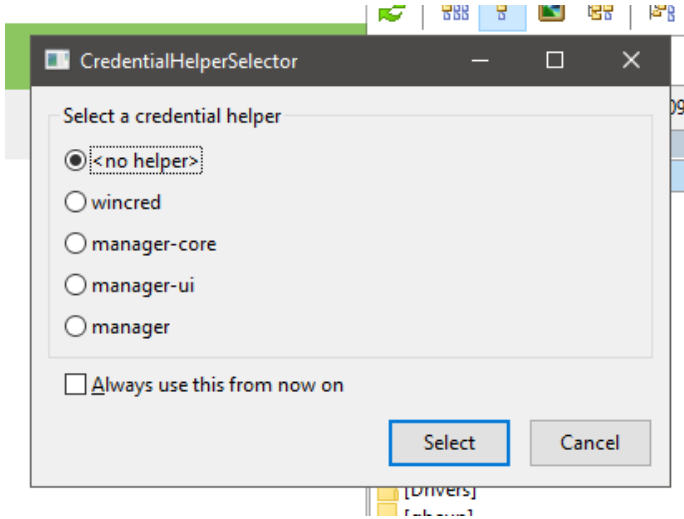


GIT

- Basic commands
  - Clone(copy) remote repository: git clone https://url.com/repository.git
  - Update changes (by coworkers) to the local repository: git pull
  - Creating a new GIT commit:
    - Register (stage) a changed file: git add path/file.cpp
    - Wrap-up the commit: git commit –m "What the commit achieves (e.g., Add file.cpp)"
  - Upload changes to the origin (the cloned one) remote repository : git push
    - Upload to a specific remote repository: git push REPOSITORY [branch]
  - Queue-up current state of the local repository: git status
  - Create new branch: git branch new-branch
  - switch branch: git checkout branch-name
- More useful commands: git commit --amend, git checkout –b new-branch

GIT na školních počítačích

- Nepoužívejte SSH klíče (ostatní se k nim můžou dostat)
  - Všichni studenti sdílí stejný „students“ adresář
  - Doma/na vlastních strojích s tim samozřejmě není problém (i pro další body)
- Nastavte v repozitáři následující (nedávejte --global)
  - git config user.email "emailova@adre.sa"
  - git config user.name “Jméno Příjmení"
- Používejte HTTPS verzi (klonujte pomocí https odkazu, ne ssh)
  - Nepoužívejte žádný password manager, musíte zadat jméno+heslo manuálně

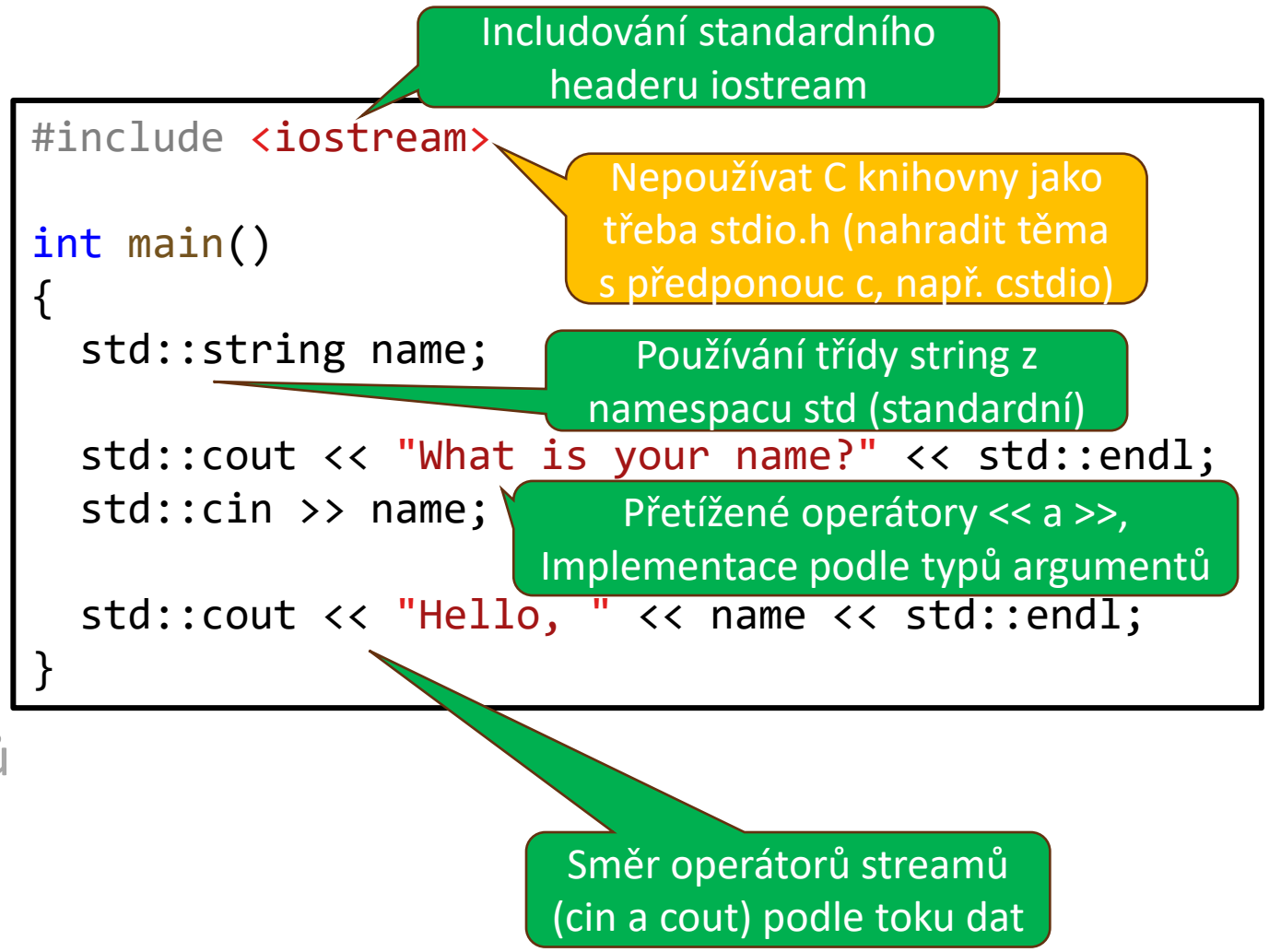


Kvalitní zdroje dodatečných informací, nástroje

- Přednášky: https://www.ksi.mff.cuni.cz/teaching/nprg041-web/
- CppReference: https://en.cppreference.com/w/
- clang-tidy, cppcheck
  - Nástroje na statickou analýzu (linting) kódu
  - Jejich warningy bývají pravdivé
    - Určitě neodhalí vše (false negative)
    - Občas se spletou (false positive)
- clang-format (nebo podobný): formatter, pomáhá udržet konzistentní styl kódu; IDE typicky mají vlastní formatter
- LLMs (chat.openai.com, GH copilot, …) mohou usnadnit psaní kódu
  - Jako oheň: dobrý sluha, ale špatný pán; je potřeba kontrolovat, co vymyslí

Hello, user

- Vytvořit projekt C++
- Vytvořit main.cpp s obsahem:
- Sestrojit projekt
- Přidat podporu pro víceslovné jméno (Jméno Příjmení)
- Přidat podporu pro --help
  - Problém se srovnáváním stringů



Streamy (proudové vstupy, výstupy)

- 3 základní streamy na komunikaci s uživatelem:
  - std::cout (mnemotechnika "console output") - standardní výstup
  - std::cin (mnemotechnika "console input") - standardní vstup
  - std::cerr - výstup pro chybové hlášky, logy, warningy
- Typicky se dělí na istreamy (input streamy), ostreamy (output streamy) a iostreamy (kombinace obou)
  - istream >> arg (čte "parsuje" vstup podle typu arg), istream.get, istream.read
  - ostream << arg (správný výstup podle typu arg), ostream.put, ostream.write
- Soubory: (i/o)fstream (budeme řešit později)

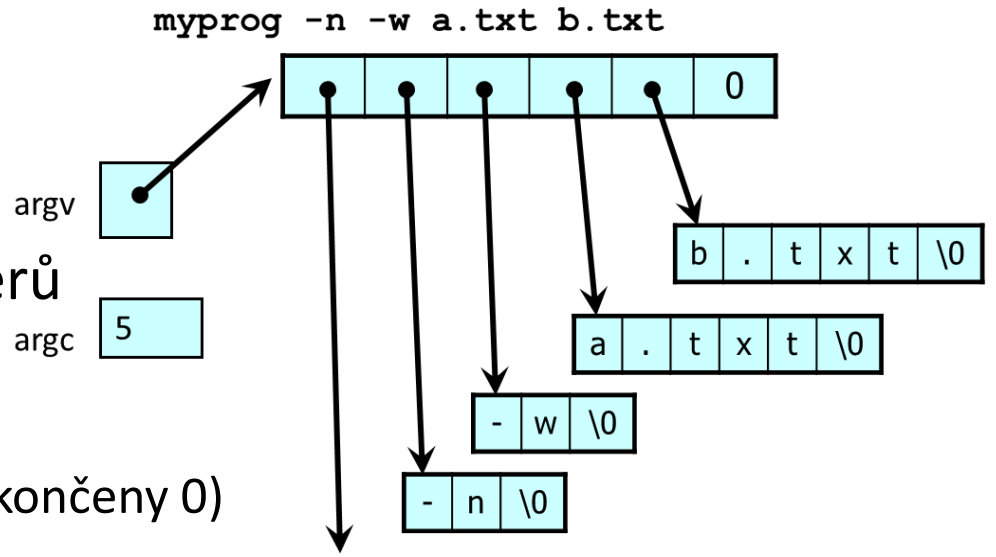
Hello, user

- Vytvořit projekt C++
- Vytvořit main.cpp s obsahem:
- Sestrojit projekt
- Přidat podporu pro víceslovné jméno (Jméno Příjmení)
- Přidat podporu pro --help
  - Problém se srovnáváním stringů

```cpp
#include <iostream>

int main()
{
  std::string name;

  std::cout << "What is your name?" << std::endl;
  std::getline(std::cin, name);

  std::cout << "Hello, " << name << std::endl;
}
```

Funkce main

- Volaná při startu programu, **nikdy jindy**
- Vrací integer
  - `0` značí úspěch (defaultně – pro každou jinou non-void funkci explicitně zavolejte `return`)
- Pokud má argumenty, tak jsou následující:
  - **`int argc`** - počet cmd argumentů
  - **`char *argv[]`** - null-terminated pole pointerů na jednotlivé cmd argumenty
    - `argv[0]` je název programu
    - Argument reprezentovány **C-stringy** (pole bajtů zakončeny 0)



#include, #define, #ifndef

- Znakem # začínají preprocesorová makra
  - `#include <standardni-header>`
    `#include <externi-header.hpp>`
    `#include "lokalni-header.hpp"`
    - To přímo okopíruje header do zdrojáku (standardní nemají koncovky, jinak .h nebo .hpp)
  - Headery začínají tzv. header-guardem (a končí jeho endifem):
    ```
    #ifndef CELY_NAZEV_IDEALNE_S_CESTOU_HPP_
    #define CELY_NAZEV_IDEALNE_S_CESTOU_HPP_
    … (tady je tělo headeru)
    #endif
    ```
    - To předchází tomu, aby byl header includován víckrát
  - Mimo includy a header-guardy se makrům vyhýbejte jak to jen jde

Nepoužívejte #define pro konstanty

- Preprocesor obecně nerozumí kódu
  - Jen nahrazuje kusy textu
  - Není safe, nezná typy
  - Pokud obsahuje výraz, dosadí ho všude, kde ho použijeme (ne jeho hodnotu)

```
constexpr size_t N = 0;
constexpr float M = 42.3;
constexpr const char* STR1 = "hello";
constexpr char STR2[] = "hello";
```

- Pro konstanty slouží klíčové slovo `constexpr`

  - To znamená „vypočítej za kompilace“, takže používání výrazů je safe
  - `constexpr` může být i funkce (před návratovým typem) – tím naznačujeme, že je možné ji vypočítat za kompilace (ale jde ji použít i v runtimu)
  - `constexpr` implikuje `const` (že to jde jen číst)
    - Pozor u pointerů: `const char*` (pointer na readonly) vs `char* const` (readonly pointer)
    - Zde implikuje právě to druhé (proto v rámečku explicitní `const char*`)

Hello, user

- Vytvořit projekt C++
- Vytvořit main.cpp s obsahem:
- Sestrojit projekt
- Přidat podporu pro víceslovné jméno (Jméno Příjmení)
- Přidat podporu pro --help
  - Problém se srovnáváním stringů

```
#include <iostream>

using namespace std;

int main(int argc, char *argv[])
{
  if (argc > 1 && argv[1] == "--help") {
    cout << "USAGE: " << argv[0] << endl;
  }

  string name;

  cout << "What is your name?" << endl;
  getline(cin, name);

  cout << "Hello, " << name << endl;
}
```

Rozbalení namespacu std
přenese vše z std sem

Tento příkaz nikdy nepoužívat v headeru

Problém

Hello, user

- Vytvořit projekt C++
- Vytvořit main.cpp s obsahem:
- Sestrojit projekt
- Přidat podporu pro víceslovné jméno (Jméno Příjmení)
- Přidat podporu pro --help
  - Problém se srovnáváním stringů

```cpp
#include <iostream>
#include <vector>

using namespace std;

int main(int argc, char *argv[])
{
  vector<string> args(argv, argv + argc);
  if (args.size() > 1 && args[1] == "--help") {
    cout << "USAGE: " << argv[0] << endl;
  }

  string name;

  cout << "What is your name?" << endl;
  getline(cin, name);

  cout << "Hello, " << name << endl;
}
```

Překopírujeme do stringů ve vectoru

Pracujeme jen s args
Dopadne správně

Souhrn

- Dnes jsme probrali
  - Jak založit a sestavit projekt, práce s nástroji na C++ development
  - Základy includování knihoven
  - Základy streamů (standardní vstup/výstup)
  - Čtení cmd argumentů
- Domácí příprava na příští cvičení
  - Nasetupovat Development Environment a nástroje (GIT, AI?, …)
  - Připojit se na Mattermost
  - Přidat se do ReCodex skupiny, vyzkoušet mff GitLab