KIV/CPP – Programování v jazyce C++

10b. Ladění kódu

Martin Úbl

KIV ZČU

2023/2024

Debugging

- ladění kódu
- možné použít jakýkoliv debugger nativního kódu
  - MSVS debugger
  - gdb
  - lldb
  - ...
- lépe vysokoúrovňový – podpora C++
  - např. name mangling
  - rozpoznání (polymorfních) typů
  - ...
- v rámci předmětu – MSVS debugger

Debugging

- ladění kódu
- debug informace uloženy v binárce nebo separátně
  - MSVS přibaluje .pdb soubor
  - gcc/clang na GNU/Linuxu vkládají do binárky
- formáty debug informací
  - DWARF
  - PE/COFF
  - stabs
  - OMF
  - ...

Debugging

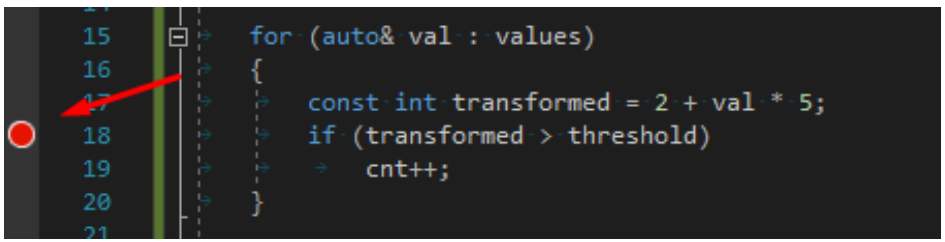
- ladění kódu
- debug informace
  - názvy funkcí/metod
  - typy proměnných
  - mapování binárního kódu na zdrojový kód
  - ...

Debugging

- debugger
- načítá debug informace
- registruje se operačnímu systému pro daný program
- dovoluje:
  - instrukční breakpointy
  - datové breakpointy
  - data watch
  - modifikace paměti
  - a další...

Debugging

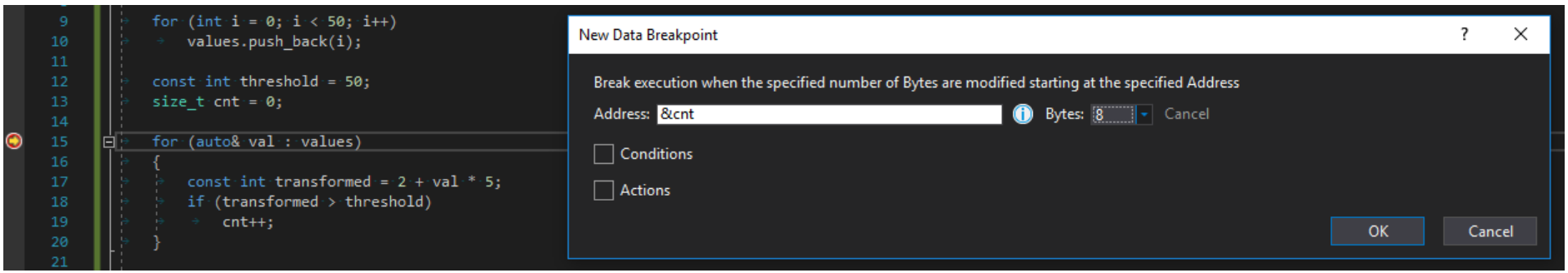
- instrukční breakpoint
- často s podporou hardware – ladicí registry
- nebo čistě softwarové – vkládání instrukcí
- zastaví běh programu na dané instrukci (před ní)
- signalizuje debugger
- může být podmíněný (výrazně pak zpomaluje běh programu)

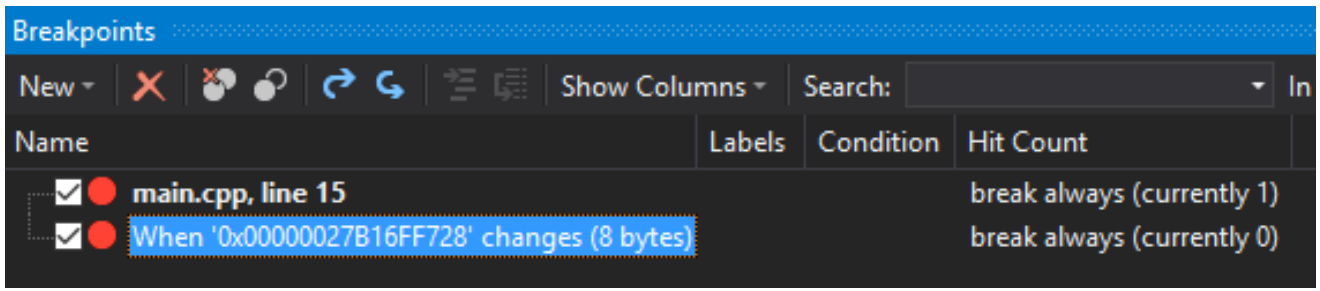


Obrázek: Instrukční breakpoint nastavený v MSVS

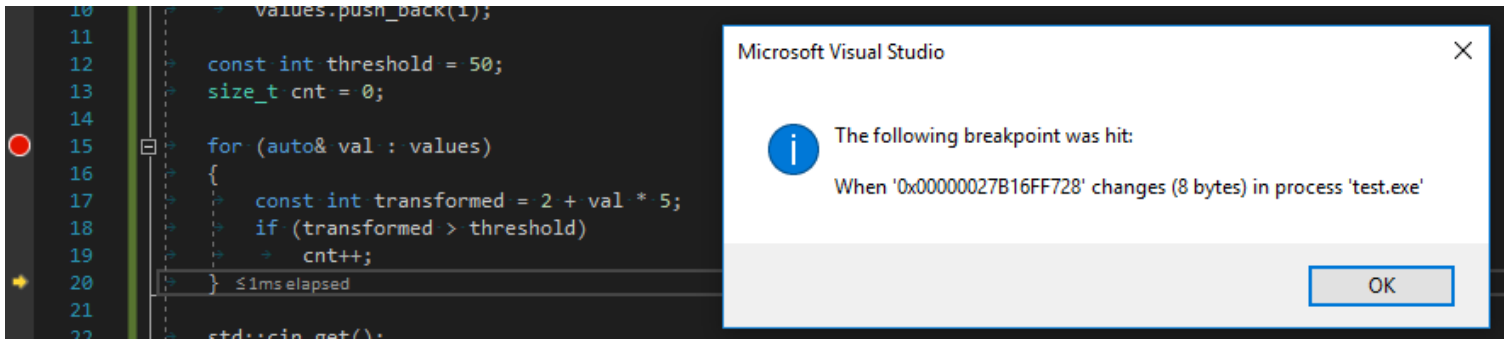
Debugging

- datový breakpoint
- dovoluje zastavit provádění programu při změně hodnoty paměti (proměnné)





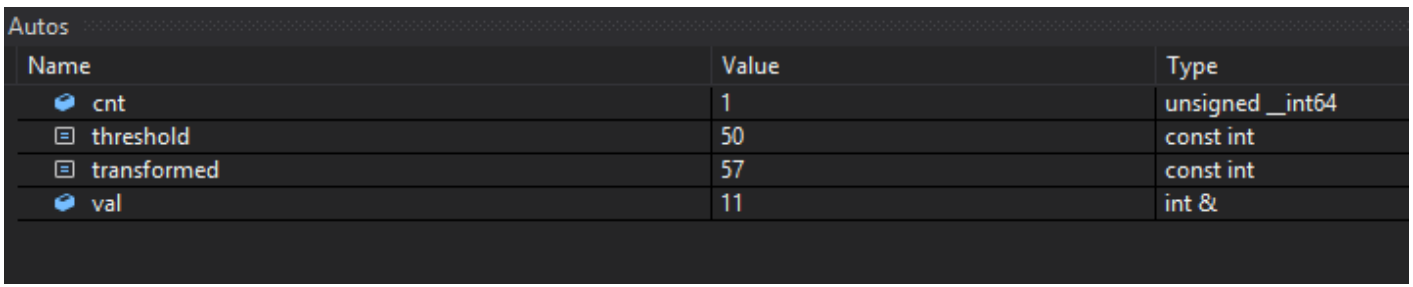
Debugging



- datový breakpoint
- nevýhoda: adresy se mění, je třeba nastavit instrukční breakpoint v místě, kde už známe adresu a až potom přidat datový breakpoint

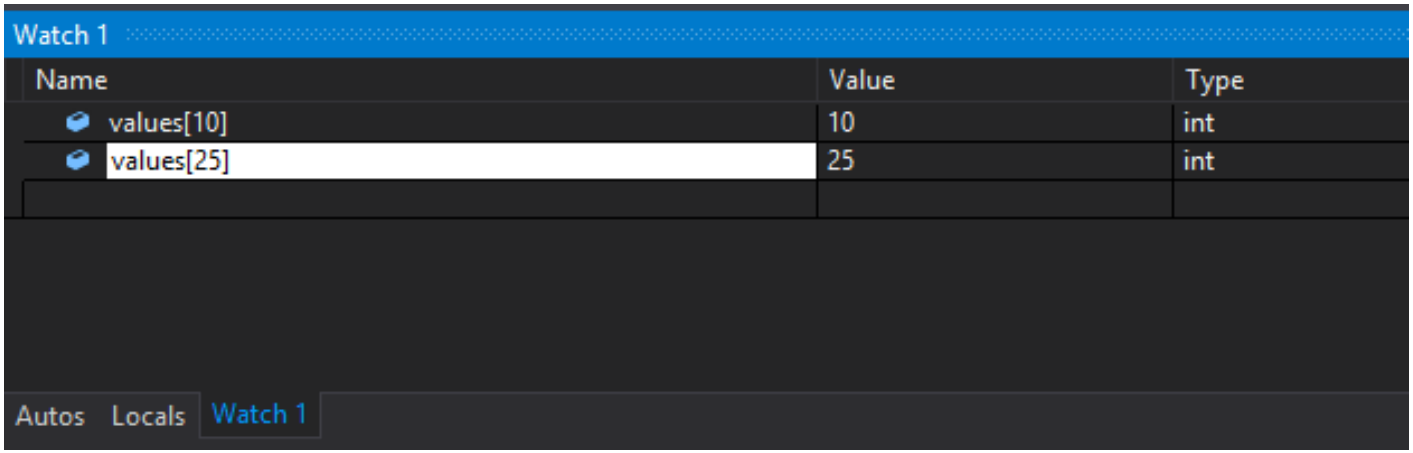
Debugging

- watch
- pohled na kus paměti při debuggování
- má smysl při krokování
- ukazuje aktuální hodnoty paměti
- automatický watch
  - debugger vydedukuje, co by mohlo zajímat
- ruční watch



Obrázek: Automatický watch v MSVS

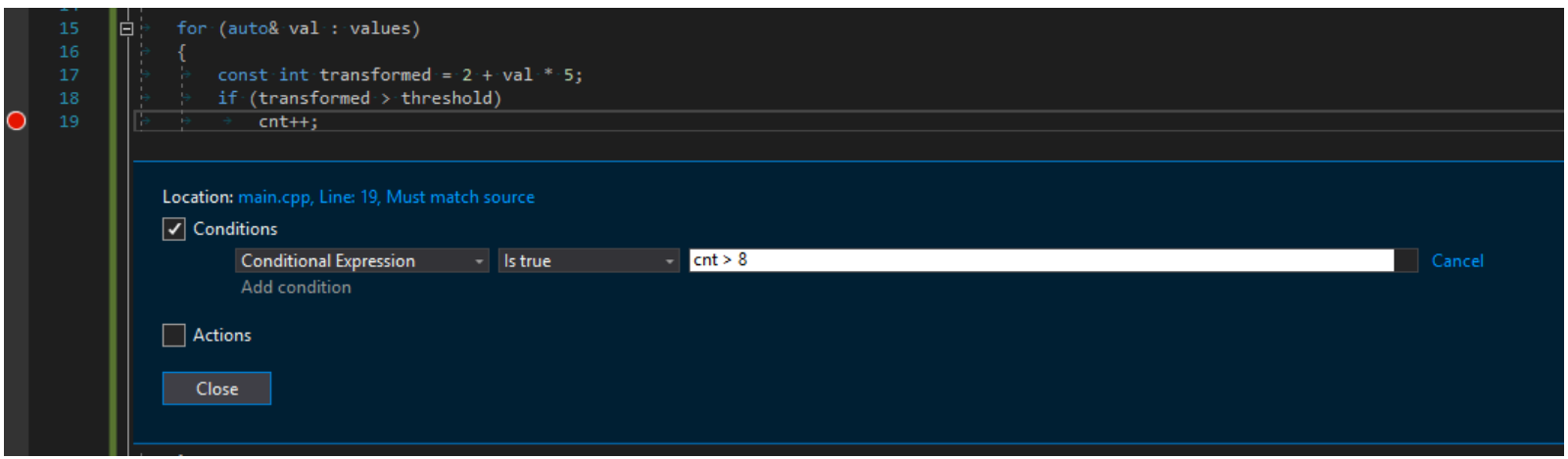
Debugging



Obrázek: Ruční watch v MSVS

Debugging

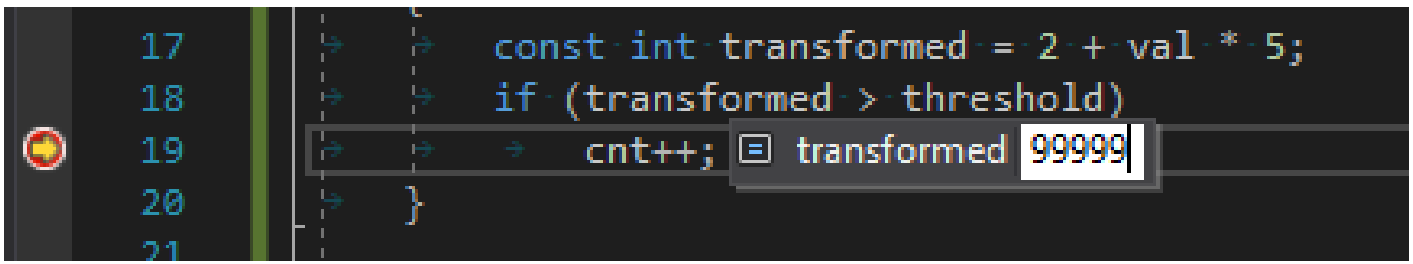
- podmíněný instrukční breakpoint
- lze nastavit podmínku v podobě logického výrazu nebo počtu průchodů (hit count)
- při průchodu breakpointem se podmínka ověří a při splnění breakne



Obrázek: Podmíněný instrukční breakpoint v MSVS

Debugging

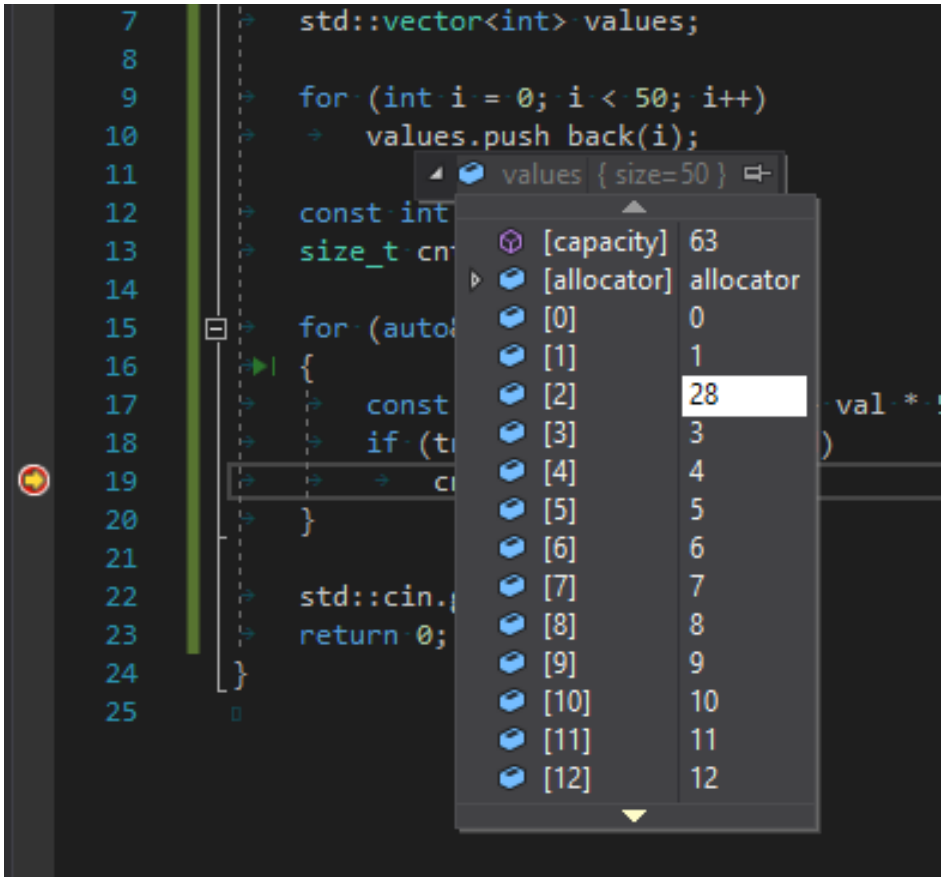
- hodnoty paměti lze za běhu měnit, když je program pozastaven



Obrázek: Editace paměti v MSVS

Debugging

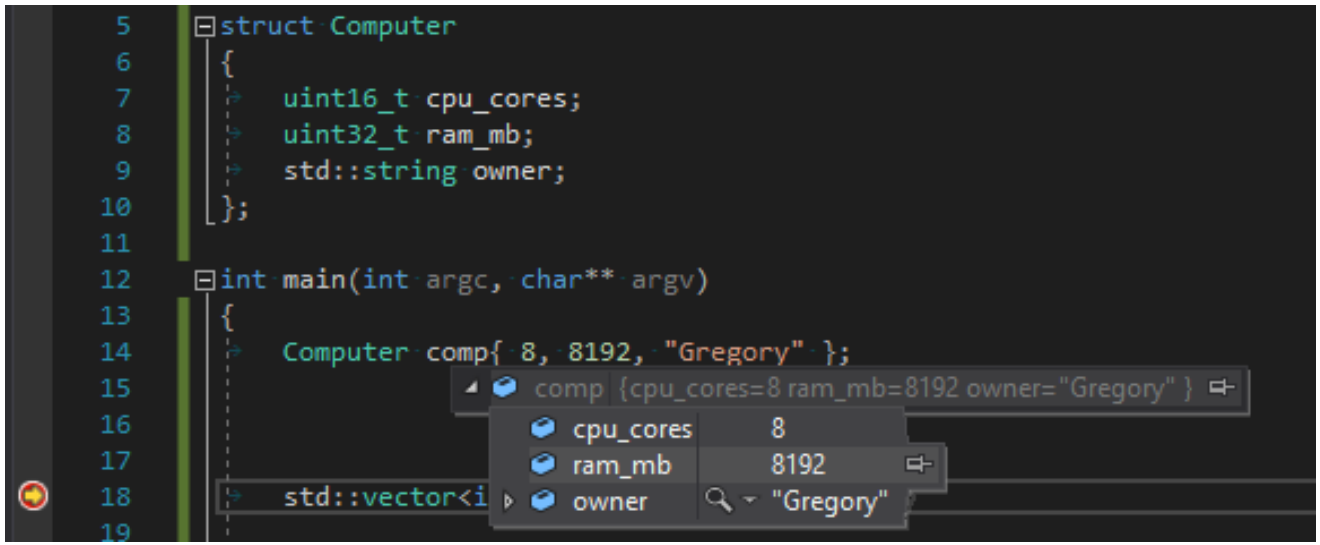
- MSVS rozpoznává i komponované typy
- lze jejich vnitřnosti „rozbalit“ a zde i editovat



Obrázek: Editace paměti v MSVS

Debugging

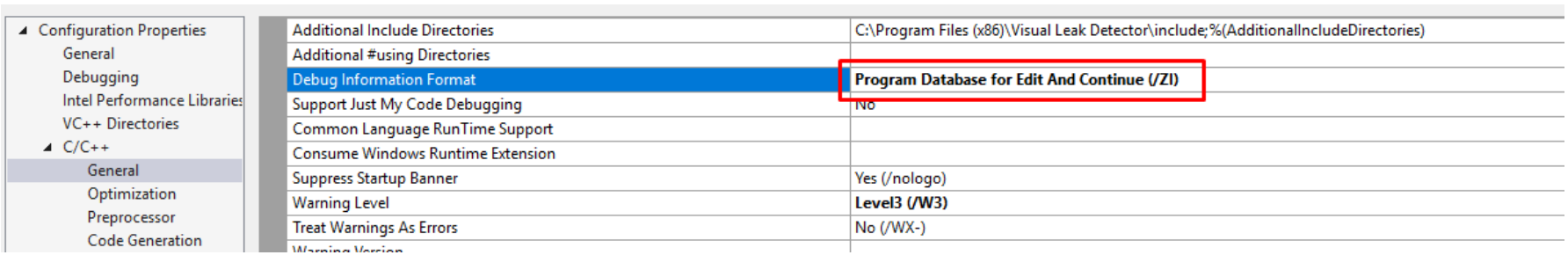
- lze inspektovat i vlastní objekty



Obrázek: Editace paměti v MSVS

Debugging

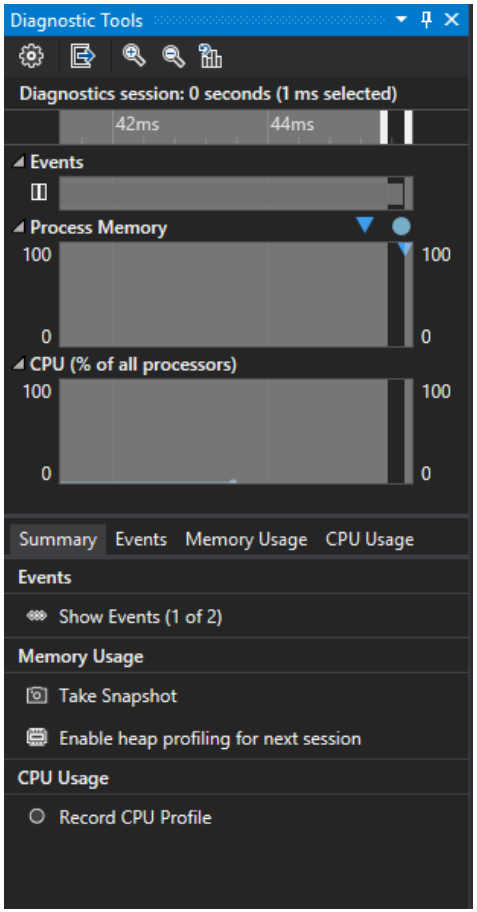
- Edit and continue
- pozastavený program lze za určitých okolností modifikovat, překompilovat a nahradit kód za běhu
- nutno přepnout formát debug informace na `/ZI`



Obrázek: Nastavení formátu debug informací v MSVS

Debugging

- diagnostic tools
- heap profiler

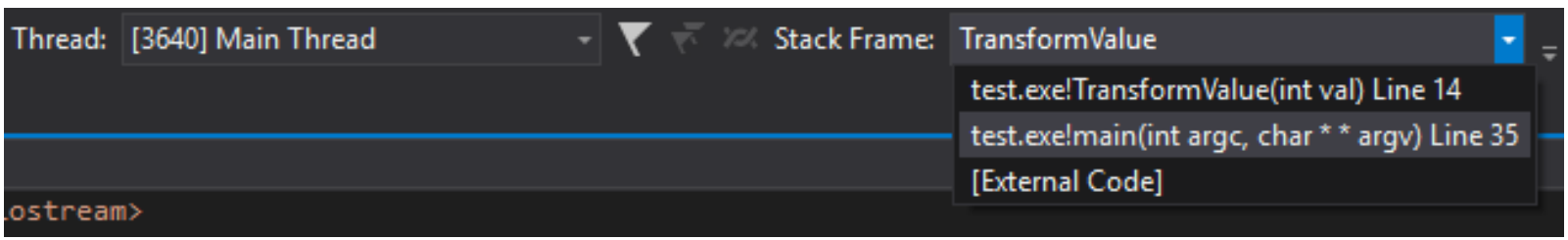


Debugging

- *continue* – pokračování v provádění
- *pause* – pozastavení programu v daném momentě
- *stop* – zastavení programu
- *restart*
- krokování
  - *step into* – zanoření do funkce
  - *step over* – překročení na další řádku kódu
  - *step out* – pokračování do návratu z funkce

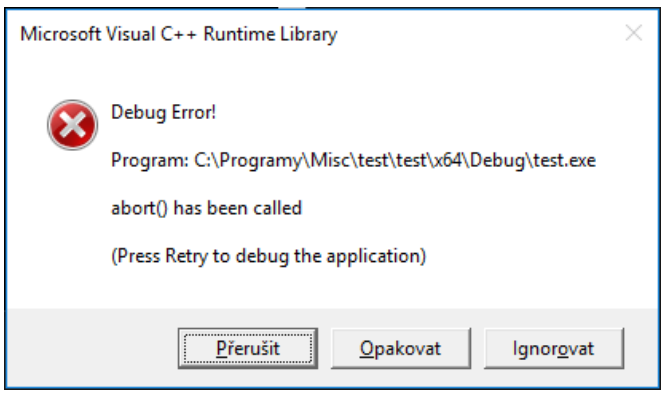
Debugging

- inspekce stavu zásobníku volání
  - lze se přepnout na libovolnou z úrovní zanoření při funkčním volání
- přepínání kontextů vláken
  - vícevláknové programy dovolují pozorovat i odlišné kontexty vláken



Debugging

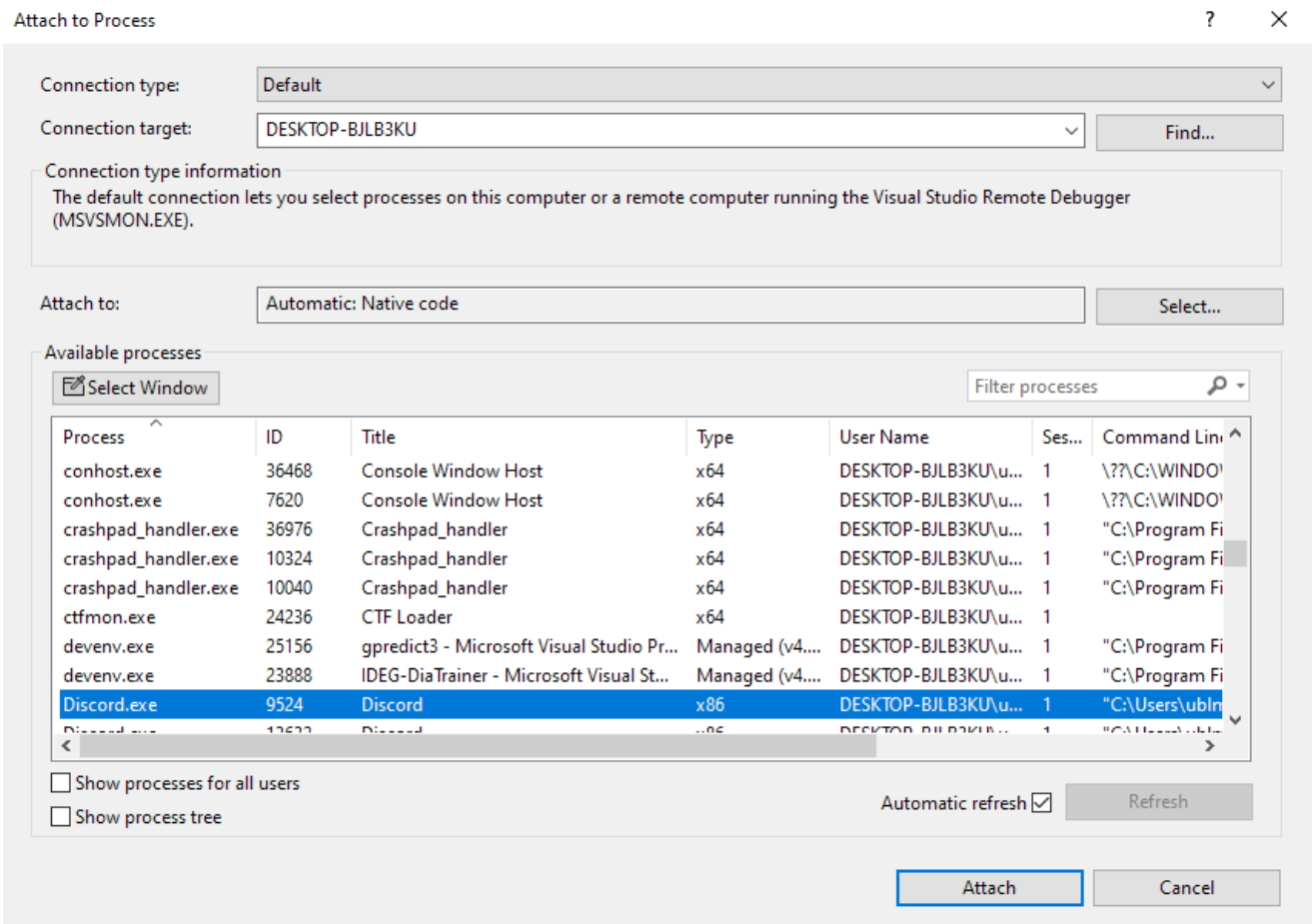
- runtime assertion
- `#include <assert.h>`
- makro `assert(...)`
- uvnitř je podmínka, při nesplnění je signalizován operační systém
- Windows – po zvolení „Opakovat“ lze předat signál do debuggeru




```
assert(count == 5 && "Pocet neni spravny");
```

Debugging

- debugger lze připojit i k jinému procesu
- nemusí jít o proces spuštěný z Visual Studia (popř. gdb, ...)
- pro úspěšné ladění je třeba mít ladicí symboly (PDB, ...)



Debugging

- debugger lze připojit i na „vzdálený“ proces
  - přes síť na jiném PC (SSH, ...)
  - ve WSL
  - v cloudu (Azure)
  - interpretující jiný podporovaný jazyk (Javascript, Python, ...)
  - na jiná prostředí (Unity, Unreal Engine, qemu, ...)