Course code: GOC2126

The course is intended for developers on the .NET (Core) platform who want to get into the details of .NET and C # applications and learn how to write the most fast applications, use the right constructs or optimize memory consumption. At the same time also for the detection and diagnosis of these problems (post-mortem and current) using available tools. Emphasis is placed on the practical application of acquired knowledge. Students will be able to apply the acquired knowledge immediately after returning to the real world. The course is suitable for developers over the .NET Framework, .NET Core, .NET 5+, Xamarin, etc.

## Required input knowledge
- The course assumes knowledge and experience with programming in C # at the level of course GOC2124 and GOC2125

## Who is the course for?
- The course is designed for experienced developers who want to move a little further and better understand how memory is handled in .NET, what are the real possibilities of debugging, how code is executed, performance is measured or reflections are used, but also others advanced topics.

## Teaching methods
- Expert explanation with practical examples, exercises on computers.

## Studying materials
- Printed presentations of the subject matter.

## Course syllabus
Working with memory in .NET
- Value vs. reference types
- Allocation and functioning of GC (SOH, LOH, POH)
- Heap, stack, unmanaged heap
- Hidden allocations
- Stackalloc
- Span, Memory
- Unsafe
- In, ref structs, readonly
- Passing parameters
- Measurement of memory consumption, memory leaks
- Profiles
- Finalization
- Strings
- WeakReference
- Pooling

Algorithm complexity
- Big O
- Work with collections

Debugging
- Debugging in Visual Studio and all options of the Visual Studio debugger
- Work with symbols
- WinDBG debugging
- ProcDump, dotnet dump
- Post mortem debugging

- Application dump and analysis application
- Application status / operation monitoring (dotnet monitor)

Code execution in a .NET environment
- JIT compilation, profile optimizations, ngen, ready to run
- Optimization in JIT, multicore JIT
- Tiered JIT, PGO
- Intrinsics, vectorization, SIMD
- CPU operation and the effect of instruction on performance
- Profiles

Code performance measurement
- Benchmarking and its limitations
- Correct vs. erroneous measurements
- Profiling
- BenchmarkDotNet
- Introduction to assembly

Reflection and its effective use

Code generation
- IL Emit
- Expressions
- Source Generators

Lazy initialization

**GOPAS**®