Debugging

This lesson discusses how errors can be tracked and removed when programming in Go.

WE'LL COVER THE FOLLOWING

Debuggers in Go

Application programming needs excellent debugging support. Most bigger IDEs provide this kind of support. To use the debugger in **Visual Studio Code**, you must currently manually install delve (see the Installation Instructions for full details). On macOS, it requires creating a self-signed cert to sign the dlv binary. An explanation on how to get running with debugging can be found here: Debugging Go Code Using VS Code. It is also possible to remotely debug code using VS Code; read Remote Debugging. Go can be debugged with GDB, and many editors support that. GoClipse has fully featured GDB debugger support (reusing Eclipse CDT's GDB integration) comprising breakpoints and conditions, watch and view, disassembly view, viewing threads and stack frame contents. GoLand supports standard debugger features: *Watches, Evaluate Expression, Show Inline Values* and others. The debugger works for applications, as well as for tests.

Debugging strategy

If you don't want to use a debugger, the following is useful as a simple debugging strategy:

- Use print-statements (with the `fmt.Print` functions) at well-chosen places.

- In `fmt.Printf` functions, use the following specifiers to obtain complete info about variables:
    - **%v** gives us a complete output of the value with its fields.
    - **%#v** gives us a complete output of the value with its fields and qualified type name.
    - **%T** gives us the complete type specification.

- Use a `panic` statement to obtain a *stack trace* (a list of all the called functions up until that moment).
- Use the `defer` keyword in tracing code execution.
- A Windows-version gocomp.bat to compile all source files in the current directory is even shorter:

```
FOR %%X in (*.go) DO go build %%X >> compout
```

Debugging the errors and making a program error-free is one of the major concerns. Apart from this concern, designers are paying great attention to provide the best formatting ideas that make a program easily readable at the user-end. The next lesson discusses such techniques.