Principles of Software Design 3
Procedural programming, Secure C coding

Robert Lukoťka
lukotka@dcs.fmph.uniba.sk
www.dcs.fmph.uniba.sk/~lukotka

M-255

Procedural programming

Procedural programming is a programming paradigm, derived from imperative programming, based on the concept of the procedure call. Procedures (a type of routine or subroutine) simply contain a series of computational steps to be carried out. Any given procedure might be called at any point during a program's execution, including by other procedures or itself. [1]

How does this compare to other paradigms?
Procedural programming - Wikipedia

Procedural programming

OOP - features

I will try to persuade you, that there is not that much of a difference here.

Procedural programming

OOP concepts [2] we will be looking for in procedural programming:

- Class/Prototype
- Dynamic dispatch
- Encapsulation
- Composition, inheritance, and delegation
- Subtyping, polymorphism

Procedural programming

- Class/Prototype
    - Associating procedure calls with classes and methods is syntactic sugar. But there definitely are advantages (e.g. better autocomplete no need to use extra namespaces)
    - Constructors and destructors - so it is not up to programmer to make init call after declaration / match malloc and free calls for dynamic allocation.
- Dynamic dispatch
    - You have function pointers and they are fine.

Procedural programming

- Encapsulation
  - .h and .c files
  - Private functions and global variables can be hidden in .c files from compiler (use extern to expose global variable)
  - If you use static it is hidden from linker.
  - You have to share size of the variable if you want to allow client to use automatic variables
  - You may use pointers and handle alocation in init/free calls
- Composition, inheritance, and delegation
  - Composition over inheritance anyways. Function pointers are a good tool for composition and delegation
- Subtyping, polymorphism
  - It is a bit clumsy. In particular given the lack of constructors and destructors, you need to have init call to initialize function pointer fields

Polymorphism in procedural programming

- Tools to achieve polymorphism
  - passing pointers to functions
  - passing multiple function pointers
  - struct of function pointers
- You can generally achieve results similar to OOP. It is more or less like creating objects by hand.
- This has consequences for testing. You can do all the mocking and stuff like in OOP cmocka

Resources I

- Procedural programming - Wikipedia

Dangers I

- Undefined behaviour
    - sprintf quite unsafe design, new call snprintf
    - It is good idea to have array args always accompanied by known size
- Signed/unsigned int casting
- Going overboard with macros

Some tools to use I

- Const correctness
- Macros to avoid errors in code

References I

- Procedural programming - Wikipedia
- OOP - features