Grobe Themenübersicht

→ expression / evaluation
- ↳ operator precedence, associativity, arity (~how many operands)
- ↳ integers: loss of precision / rounding, over / underflow
- ↳ bools: short-circuiting

→ safe programming
- ↳ const
- ↳ assert

→ control flow
- ↳ if-else
- ↳ for, while, do-while loops
- ↳ scope of variables
- ↳ break / continue ⟷ to "break" from void func: use return; type func: return val;
- ↳ switch-case

→ floats
- ↳ type float vs type double ≙ single (32) vs double (64 bit) precision
- ↳ floating point number systems
- ↳ normalized numbers
- ↳ base conversion
- ↳ IEEE 754
- ↳ loss of precision → e.g. $f_{small} + f_{big}$ ⇒ Course page: floating point guidelines

→ functions
- ↳ uniqueness of func signature → name, args, return type (→ also important for operator overloading)
- ↳ pre / post cond.
- ↳ stepwise refinement
- ↳ forward declaration
- ↳ libs, include statements

→ references
- ↳ definition, initialization
- ↳ usage: pass by ref vs pass by val
- ↳ const refs

→ std::vectors
- ↳ mem. layout → contiguous
- ↳ rand access (vs. non-rand access e.g. in linked list)
- ↳ initialization
- ↳ .at(index) and out-of-bounds

→ strings / chars
  ↳ integer representation / ASCII
  ↳ std::string

→ recursion
  ↳ base case
  ↳ call stack → recursion tree
  ↳ backtracing e.g. dungeon

→ structs, classes
  ↳ definition and initialization
  ↳ custom operators / operator overloading
  ↳ constructors
  ↳ member functions
  ↳ const → const member funcs
  ↳ information hiding: "encapsulation" → public/ private

→ dynamic data types / pointers
  ↳ new → returns a ptr! int x = 5;
  ↳ delete
  ↳ address operator int * p = &x;
  ↳ dereference operator int val = *p;
  ↳ const ptr vs ptr to const var
  ↳ ptr arithmetic

→ containers in general
  ↳ std:: set, vector
  ↳ iterators
  ↳ templates

→ memory management (with classes)
  ↳ destructors
  ↳ copy constr.
  ↳ (copy) assignment operator
} rule of 3/5/0!