20. Conclusion

Purpose and Format

Name the most important key words to each chapter. Checklist:
“does every notion make some sense for me?”

- motivating example for each chapter
- concepts that do not depend from the implementation (language)
- language (C++): all that depends on the chosen language
- examples from the lectures

1. Introduction

- Euclidean algorithm
- algorithm, Turing machine, programming languages, compilation, syntax and semantics
- values and effects, fundamental types, literals, variables
- include directive #include <iostream>
- main function int main(){...}
- comments, layout // Kommentar
- types, variables, L-value a, R-value a+b
- expression statement b=b*b; , declaration statement int a;, return statement return 0;

2. Integers

- Celsius to Fahrenheit
- associativity and precedence, arity
- expression trees, evaluation order
- arithmetic operators
- binary representation, hexadecimal numbers
- signed numbers, twos complement
- arithmetic operators 9 * celsius / 5 + 32
- increment / decrement expr++
- arithmetic assignment expr1 += expr2
- conversion int ↔ unsigned int
- Celsius to Fahrenheit, equivalent resistance
3. Booleans

- Boolean functions, completeness
- DeMorgan rules
- the type bool
- logical operators \( a \land \neg b \)
- relational operators \( x < y \)
- precedences \( 7 + x < y \land y \neq 3 \times z \)
- short circuit evaluation \( x \neq 0 \land z / x > y \)
- the assert-statement, \#include <cassert>
- Div-Mod identity.

4./5. Control Statements

- linear control flow vs. interesting programs
- selection statements, iteration statements
- (avoiding) endless loops, halting problem
- Visibility and scopes, automatic memory
- equivalence of iteration statement
- if statements \( \text{if} (a \land 2 == 0) \{ \ldots \} \)
- for statements \( \text{for} (\text{unsigned int} i = 1; i <= n; ++i) \{ \ldots \} \)
- while and do-statements \( \text{while} (n > 1) \{ \ldots \} \)
- blocks and branches \( \text{if} (a < 0) \text{continue}; \)
- sum computation (Gauss), prime number tests, Collatz sequence, Fibonacci numbers, calculator

6./7. Floating Point Numbers

- correct computation: Celsius / Fahrenheit
- fixpoint vs. floating point
- holes in the value range
- compute using floating point numbers
- floating point number systems, normalisation, IEEE standard 754
- guidelines for computing with floating point numbers
- types float, double
- floating point literals \( 1.23e-7f \)
- Celsius/Fahrenheit, Euler, Harmonic Numbers

8./9. Functions

- Computation of Powers
- Encapsulation of Functionality
- functions, formal arguments, arguments
- scope, forward declarations
- procedural programming, modularization, separate compilation
- Stepwise Refinement
- declaration and definition of functions double pow(double b, int e){ \ldots } function call pow (2.0, -2)
- the type void
- powers, perfect numbers, minimum, calendar
10. Reference Types

- Swap
- value- / reference- semantics, call by value, call by reference
- lifetime of objects / temporary objects
- constants
- reference type int & a
- call by reference, return by reference int & increment (int & i)
- const guideline, const references, reference guideline
- swap, increment

11./12. Arrays

- Iterate over data: array of Erathosthenes
- arrays, memory layout, random access
- (missing) bound checks
- vectors
- characters: ASCII, UTF8, texts, strings
- array types int a[5] = {4,3,5,2,1};
  characters and texts, the type char char c = 'a';, Konversion nach int
- multi-dimensional arrays, vectors of vectors
- sieve of Erathosthenes, Caesar-code, shortest paths, Lindenmayer systems

13./14. Pointers, Iterators and Containers

- arrays as function arguments
- pointers, chances and dangers of indirection
- random access vs. iteration, pointer arithmetics
- containers and iterators
- pointer int x;, Conversion array → pointer, null-pointer
- address and dereference operator int *ip = &i; int j = *ip;
- pointer and const const int *a;
- algorithms and iterators std::fill (a, a+5, 1);
- type definitions typedef std::set<char>::const_iterator Sit;
- filling an array, character salad

15./16. Recursion

- recursive матhe. functions
- recursion
- call stack, memory of recursion
- correctness, termination,
- recursion vs. iteration
- EBNF, formal grammars, streams, parsing
- evaluation, associativity
- factorial, GCD, Fibonacci, mountains
17. Structs and Classes I

- build your own rational number
- heterogeneous data types
- function and operator overloading
- encapsulation of data
- struct definition
  ```c
  struct rational {
  int n; int d;
  }
  ```
- member access
  ```c
  result.n = a.n * b.d + a.d * b.n;
  ```
- initialization and assignment,
- function overloading
  - `pow(2)` vs. `pow(3,3)`
- operator overloading
- rational numbers, complex numbers

18. Classes, Dynamic Data Types

- rational numbers with encapsulation, stack
- linked list, allocation, deallocation, dynamic data type
- classes
  ```c
  class rational {
  ...
  }
  ```
- access control
  ```c
  public:/private:
  ```
- member functions
- copy constructor, destructor, rule of three
- constructors
  ```c
  rational (int den, int nm)
  ```
- new and delete
- copy constructor, assignment operator, destructor
- linked list, stack

19. Tree Structures, Inheritance and Polymorphism

- expression trees,
- extension of expression trees
- inheritance
- trees
- inheritance
- polymorphism
- inheritance
  ```c
  class tree_node: public number_node
  ```
- virtual functions
  ```c
  virtual void size() const;
  ```
- expression tree, expression parsing, extension by abs-node

The End

End of the Course