19. Zusammenfassung

Zweck und Format

Nennung der wichtigsten Stichwörter zu den Kapiteln.

- Ⓜ Motivation: Motivierendes Beispiel zum Kapitel
- Ⓚ Konzepte: Konzepte, die nicht von der Implementation (Sprache) $\mathrm{C{+}{+}}$abhängen
- Ⓢ Sprachlich ($\mathrm{C{+}{+}}$): alles was mit der gewählten Sprache zusammenhängt
- Ⓑ Beispiele: genannte Beispiele der Vorlesung

1. Einführung

Ⓜ
- Euklidischer Algorithmus

Ⓚ
- Algorithmus, Turingmaschine, Programmiersprachen, Syntax und Semantik
- Werte und Effekte, (Fundamental)typen, Literale, Variablen, Konstanten, Bezeichner, Objekte, Ausdrücke, Operatoren, Anweisungen

Ⓢ
- Include-Direktiven `#include <iostream>`
- Hauptfunktion `int main(){...}`
- Kommentare, Layout `// Kommentar`
- Typen, Variablen, Konstanten `const int s = 299792458;`
- L-Wert `a` , R-Wert `a+b`
- Ausdrucksanweisung `b=b*b;` , Deklarationsanweisung `int a;`, Rückgabeanweisung `return 0;`

2. Ganze Zahlen

Ⓜ
- Celsius to Fahrenheit

Ⓚ
- Assoziativität und Präzedenz, Stelligkeit
- Ausdrucksbäume, Auswertungsreihenfolge
- Arithmetische Operatoren
- Binärzahldarstellung, Hexadezimale Zahlen
- Wertebereiche, Zahlendarstellung mit Vorzeichen, Zweierkomplement

Ⓢ
- Arithmetische Operatoren `9 * celsius / 5 + 32`
- Inkrement / Dekrement `expr++`
- Arithmetische Zuweisungen `expr1 += expr2`
- Konversion `int` $\leftrightarrow$ `unsigned int`

Ⓑ
- Celsius to Fahrenheit, Ersatzwiderstand

3. Wahrheitswerte

Ⓚ
- Boole'sche Funktionen, Vollständigkeit*
- DeMorgan'sche Regeln

Ⓢ
- Der Typ `bool`
- Logische Operationen `a && !b`
- Relationale Operationen `x < y`
- Präzedenzen `7 + x < y && y != 3 * z`
- Kurzschlussauswertung `x != 0 && z / x > y`
- Assertions, Konstanten und die *Const-Richtlinie*

4./5. Kontrollanweisungen

Ⓜ
- Linearer Kontrollfluss vs. interessante Programme, Spaghetti-Code

Ⓚ
- Auswahlanweisungen, Iterationsanweisungen
- (Vermeidung von) Endlosschleifen, Halteproblem
- Sichtbarkeits- und Gültigkeitsbereich
- Automatische Speicherdauer
- Äquivalenz von Iterationsanweisungen

Ⓢ
- if Anweisungen `if (a % 2 == 0) {..}`
- for Anweisungen `for (unsigned int i = 1; i <= n; ++i) ...`
- while und do-Anweisungen `while (n > 1) {...}`
- Blöcke, Sprunganweisungen `if (a < 0) continue;`

Ⓑ
- Summenberechnung (Gauss, Primzahltest, Collatz-Folge, Fibonacci Zahlen

6./7. Fliesskommazahlen

Ⓜ
- Richtig Rechnen: Celsius / Fahrenheit

Ⓚ
- Fixkomma- vs. Fliesskommazahldarstellung
- (Löcher im) Wertebereich
- Rechnen mit Fliesskommazahlen, Umrechnung
- Fliesskommazahlensysteme, Normalisierung, IEEE Standard 754
- *Richtlinien für das Rechnen mit Fliesskommazahlen*

Ⓢ
- Typen `float`, `double`
- Fliesskommaliterale `1.23e-7f`

Ⓑ
- Celsius/Fahrenheit, Euler, Harmonische Zahlen

8./9. Funktionen

Ⓜ
- Potenzberechnung

Ⓚ
- Kapselung von Funktionalität
- Funktionen, formale Argumente, Aufrufargumente
- Gültigkeitsbereich, Vorwärts-Deklaration
- Prozedurales Programmieren, Modularierung, Getrennte Übersetzung
- *Stepwise Refinement*

Ⓢ
- Funktionsdeklaration, -definition

```
double pow(double b, int e){ ... }
```

- Funktionsaufruf `pow (2.0, -2)`
- Assertions `assert (e >=0 || b != 0);`
- Der typ `void`

Ⓑ
- Potenzberechnung, perfekte Zahlen, Minimum,

10. Referenztypen

Ⓜ
- Funktion Swap

Ⓚ
- Werte-/ Referenzsemantik, Call by Value / Call by Reference
- Lebensdauer von Objekten / Temporäre Objekte

Ⓢ
- Referenztyp `int& a`
- Call by Reference und Return by Reference

  `int& increment (int& i)`
- Const-Referenzen, Referenzrichtlinie

Ⓑ
- Swap, Inkrement

11./12. Felder (Arrays)

Ⓜ
- Iteration über Daten: Feld des Eratosthenes

Ⓚ
- Felder, Speicherlayout, Wahlfreier Zugriff
- (Fehlende) Grenzenprüfung
- Vektoren
- ASCII, UTF8, Texte, Strings

Ⓢ
- Feldtypen `int a[5] = {4,3,5,2,1};`
- Zeichen und Texte, der Typ char `char c = 'a';`, Konversion nach `int`
- Mehrdimensionale Felder, Vektoren von Vektoren

Ⓑ
- Sieb des Eratosthenes, Caesar-Code, Kürzeste Wege, Lindenmayer-Systeme

13./14. Zeiger, Iteratoren, Container

Ⓜ
- Arrays als Funktionsargumente

Ⓚ
- Zeiger, Möglichkeiten und Gefahren der Indirektion
- Wahlfreier Zugriff vs. Iteration, Zeiger-Arithmetik
- Container und Iteratoren

Ⓢ
- Zeiger `int* x;`, Konversion Feld $\rightarrow$ Zeiger, Nullzeiger
- Adress-, Dereferenzoperator `int *ip = &i; int j = *ip;`
- Zeiger und Const `const int *a;`
- Algorithmen und Iteratoren `std::fill(a, a+5, 1);@`
- Typdefinitionen `typedef std::set<char>::const_iterator Sit;`
- Überladen von Funktionen `pow(2)` vs. `pow(3,3);`

Ⓑ
- Füllen eines Feldes, Buchstabensalat

15. Rekursion

Ⓜ
- Rekursive math. Funktionen

Ⓚ
- Rekursion
- Aufrufstapel, Gedächtnis der Rekursion
- Korrektheit, Terminierung,
- Rekursion vs. Iteration
- BNF, Parsen
- Auswertung, Assoziativität

Ⓑ
- Fakultät, GGT, Fibonacci, Berge, Taschenrechner

16. Structs und Klassen I

Ⓜ
- Datentyp Rationale Zahlen selber bauen

Ⓚ
- Inhomogene Datenstruktur
- Operator Overloading
- Datenkapselung

Ⓢ
- Struct Definition `struct rational {int n; int d;};`
- Mitgliedszugriff `result.n = a.n * b.d + a.d * b.n;`
- Initialisierung und Zuweisung, Überladen von Operatoren

Ⓑ
- rationale Zahlen, komplexe Zahlen

17. Klassen

Ⓜ
- Rationale Zahlen
- Stack

Ⓚ
- Verkettete Liste
- Allokation, Deallokation, dynamische Objekte

Ⓢ
- Klassen `class rational { ... };`
- Zugriffssteuerung `public:` / `private:`
- Mitgliedsfunktionen `int rational::denominator () const`
- Copy-Konstruktor, Destruktor, Dreierregel
- Konstruktoren `rational (int den, int nm): d(den), n(no) {}`
- `new` und `delete`
- Copy-Konstruktor, Zuweisungs-Operator, Destruktor

Ⓑ
- Verkettete Liste, Stack

18. Vererbung und Polymorphie

Ⓜ
- Ausdrucksbäume,
- Erweiterung von Ausdrucksbäumen

Ⓚ
- Baumstrukturen
- Vererbung
- Polymorphie

Ⓢ
- Vererbung `class tree_node: public number_node`
- Virtuelle Funktionen `virtual void size() const;`

Ⓑ
- Ausdrucksbaum, Parser auf Ausdrücken, Erweiterung um abs-Knoten

Ende

Ende der Vorlesung.