Klassen

- Was passiert, wenn ein neues Objekt ohne runde Klammern erzeugt wird?
- Vor allem bei einer Klasse, die gar keinen Konstruktor hat? → `wallet.cpp`
- Unser Beispiel mit komplexen Zahlen lässt sich nun auch als eine Klasse implementieren → `complex-class.cc`

Präprozessor

- Die Präprozessoranweisung `#define` ist meist eine "mechanische" Ersetzung
- Ob die Anweisung überhaupt sinnvoll/korrekt ist, wird vom Präprozessor gar nicht geprüft.
- Dabei können auch einzelne C++ Befehle durch andere Anweisungen ersetzt werden → `german.cpp`
- `g++ -E -P code.cpp` zeigt den Code nach der Bearbeitung durch den Präprozessor (nützlich für das Debuggen)
- Auch Ersetzungen mit Argumenten sind möglich → `moddiv.cpp`

Nachbesprechung Übungsaufgaben

- Aufgabe 19: $\rightarrow$ `a19-rechteck.cc`
- Aufgabe 20: $\rightarrow$ `a20-hanoi.cc`
  - 1. Aufgabe: die Quellstange A leer bekommen
  - 2. Aufgabe den Zielturm bei B richtig aufbauen
  - `schiebe(n, von A, nach B, über C)`, ruft sich selbst für $n-1$ zweimal auf
  - Der 1. Aufruf von `schiebe(n-1, von A, nach C, über B)` + eine zusätzliche Verschiebung der n. Scheibe erledigt die 1. Aufgabe
  - Der 2. Aufruf von `schiebe(n-1, von C, nach B, über A)` erledigt die 2. Aufgabe
- Der Prozess lässt sich visualisieren $\rightarrow$ `a20-hanoi-stangen.cc`
- Gleiches mit Klassen $\rightarrow$ `a20-hanoi-stangen-cl.cc`

Nachbesprechung Übungsaufgaben

- Aufgabe 21: $\rightarrow$ `a21-quersumme.cc`
- $\mathrm{qs}(392) = \underbrace{392\%10}_{=2} + \mathrm{qs}\underbrace{(392/10)}_{39} = 2 + \mathrm{qs}(39)$

  $\mathrm{qs}(39) = \underbrace{39\%10}_{=9} + \mathrm{qs}\underbrace{(39/10)}_{3} = 9 + \mathrm{qs}(3)$

  $\mathrm{qs}(3) = \underbrace{3\%10}_{=3} + \mathrm{qs}\underbrace{(3/10)}_{0} = 3 + \mathrm{qs}(0) = 3$

  Somit $\mathrm{qs}(392) = 2 + 9 + 3 = 14$
- $\mathrm{totalqs}(392) \rightarrow \mathrm{totalqs}(\underbrace{\mathrm{qs}(392)}_{=14}) \rightarrow \mathrm{totalqs}(\underbrace{\mathrm{qs}(14)}_{=5}) \rightarrow \mathrm{totalqs}(\underbrace{\mathrm{qs}(5)}_{=5}) = 5$

Vorbesprechung Übungsaufgaben

- Aufgabe 22: Beispiel für die Partitionierung: `A = {3,5,1,7,2}`
- `s=0, t=4, l=0`, `l` als Listentrenner
- `pivot = A[0] = 3`
- i=1: `A[1] = 5 < 3 ?` Nein, keine Änderung: `A = {3,5,1,7,2}, l=0`
- i=2: `A[2] = 1 < 3 ?` Ja, daher `A[2]` $\leftrightarrow$ `A[1]`: `A = {3,1,5,7,2}, l=1`
- i=3: `A[3] = 7 < 3 ?` Nein, keine Änderung: `A = {3,1,5,7,2}, l=1`
- i=4: `A[4] = 2 < 3 ?` Ja, daher `A[4]` $\leftrightarrow$ `A[2]`: `A = {3,1,2,5,7}, l=2`
- Anschließend `A[0]` $\leftrightarrow$ `A[2]`: `A = {2,1,3,5,7}`
- Wichtig für den Algorithmus: Die Stelle des Pivotelements ändert sich nach jeder Partitionierung.
- Die Teillisten links und rechts vom Pivotelement können nach dem gleichen Verfahren sortiert werden $\rightarrow$ Rekursion!

Vorbesprechung Übungsaufgaben

- Aufgabe 23: Sehr ähnlich zu unserer Klasse der komplexen Zahlen
- Zahlen in Summanden zerlegen: Betrachte Paare aus der ursprünglichen Zahl n und dem größten Summanden m, z.B. $[4, 3]$.
- Wie oft ist größte Summand in der Zahl enthalten? $\Rightarrow$ restfreie Division
- Rekursive Implementierung, bei der der größte Summand jeweils um eins verkleinert wird.
- Die ursprüngliche Zahl wird ebenfalls um $i \times m$ verkleinert, i läuft von 0 bis n/m
- Sobald $n = 0$ erreicht ist, hat man eine Zerlegung gefunden.
- $m = 0$ bedeutet keine Zerlegung
- Was passiert für $m > n$ und gerade m-Werte?
- Was ist, wenn $m = 1$ gilt?

- Beispiel einer Zerlegungssuche
  - $[4,4]$, $4/4 = 1$, betrachte: $[4 - 0 \times 4, 3]$ und $[4 - 1 \times 4, 3]$
    $[4,3]$ muss weiter untersucht werden
    $[0,3]$ entspricht $4 = (4)$, $\# = 1$
  - $[4,3]$, $4/3 = 1$, betrachte: $[4 - 0 \times 3, 2]$ und $[4 - 1 \times 3, 2]$
    $[4,2]$ muss weiter untersucht werden
    Aus $[1,2]$ wird $[1,1]$, das entspricht $4 = (3+1)$, $\# = 2$
  - $[4,2]$, $4/2 = 2$, betrachte: $[4 - 0 \times 2, 1]$, $[4 - 1 \times 2, 1]$ und $[4 - 2 \times 2, 1]$
    $[4,1]$ entspricht $4 = 1+1+1+1$, $\# = 3$
    $[2,1]$ entspricht $4 = 2+1+1$, $\# = 4$
    $[0,1]$ entspricht $4 = 2+2$, $\# = 5$
- Somit $4 = (4), (3+1), (1+1+1+1), (2+1+1), (2+2)$
- Sofern unklar, die gleichen Schritte für $n = 5$ (7 Zerlegungen) durchgehen.