ETH Zürich

Informatik I
Übungsstunde 09

Johannes Odenthal

Hausaufgabenreview · Exercise 08

Reverse Digits · Ausgabe einmal

```cpp

// PRE: n >= 0
// POST: prints digits in reverse
void reverse(int n) {
  std::cout << n % 10;
  if (n >= 10) {
    reverse(n / 10);
  }
}
```

Rounding · Conditions und Formatierung

```cpp

// PRE: nearest integer fits in int
// POST: nearest; ties away from 0
int round_number(double x) {
  if (x < 0) { return x - 0.5; }
  return x + 0.5;
}
```

Auch eigene Hilfsfunktionen mit Pre-Condition und Post-Condition versehen

Hausaufgabenfragen

Welche Fragen sind bei der letzten Übung offen geblieben?

Structs

Structs

```
#include <vector>

struct strange {
  int n;
  bool b;
  std::vector<int> a;
};
```

Structs

```cpp
#include <vector>

struct strange {
  int n;
  bool b;
  std::vector<int> a;
};

int main() {
  strange x = {1, true, {1, 2, 3}};
  strange y = x;
  y.a[2] = 9;
  std::cout << x.a[2] << " " << y.a[2];
}
```

Was wird ausgegeben?

```
3 9
```

Structs · Ohne Member-Standardwerte

```
1  struct point {
2    double x;
3    double y;
4  };
5
6  point a;                // x, y uninitialisiert
7  point b{};              // x = 0.0, y = 0.0
8  point c = {};           // x = 0.0, y = 0.0
9  point d = {2.0, 3.0};   // x = 2.0, y = 3.0
10 point e = {2.0};        // x = 2.0, y = 0.0
```

Structs · Mit Member-Standardwerten

```
1  struct point {
2    double x = 1.0;
3    double y = 2.0;
4  };
5
6  point a;                 // x = 1.0, y = 2.0
7  point b{};               // x = 1.0, y = 2.0
8  point c = {};            // x = 1.0, y = 2.0
9  point d = {8.0, 9.0};    // x = 8.0, y = 9.0
10 point e = {8.0};         // x = 8.0, y = 2.0
```

Geometry Exercise

CodeExpert

Fragen?

Pause

09:00–09:15

Recursion

Power Set

CodeExpert

Power Set · Das letzte Element hinzufügen

Schritt für Schritt ein Element mehr:

$$\begin{aligned}
\mathcal{P}(\{\}) &= \{\{\}\} \\
\mathcal{P}(\{a\}) &= \{\{\}, \{a\}\} \\
\mathcal{P}(\{a,b\}) &= \{\{\}, \{a\}, \{b\}, \{a,b\}\} \\
\mathcal{P}(\{a,b,c\}) &= \{\{\}, \{a\}, \{b\}, \{a,b\}, \{c\}, \{a,c\}, \{b,c\}, \{a,b,c\}\}
\end{aligned}$$

> Alte Teilmengen behalten + Kopien mit dem neuen Element ergänzen.

Power Set · Vom Muster zur Rekursion

1 · Ein Element wegnehmen

Bei {a, b, c} nehmen wir c heraus. Übrig bleibt {a, b}.

2 · Dieselbe Funktion aufrufen

Wir brauchen alle Teilmengen von {a, b}. Das kann unsere Funktion selbst — jetzt mit einem Element weniger.

3 · Bei der leeren Menge stoppen

Die leere Menge hat genau eine Teilmenge: die leere Menge selbst. Dieses Ergebnis geben wir zurück.

4 · Beim Zurückkehren ergänzen

Alle zurückgegebenen Teilmengen behalten. Jede davon kopieren, c hinzufügen und die neue Teilmenge ebenfalls aufnehmen.

Immer kleiner: {a, b, c} → {a, b} → {a} → {}
Dann zurück: Mit a ergänzen → mit b ergänzen → mit c ergänzen.

Fragen?

Fragen?

Anhang · Power Set

Power Set · Base Case

Was ist die Potenzmenge der leeren Menge?

$$\mathcal{P}(\{\}) = \{\{\}\}$$

Ein Element: die leere Menge

`CharSet()` ist leer

`SetOfCharSets(CharSet())` enthält eine leere Menge

Power Set · Kleine Mengen

Welche Teilmengen haben {a} und {a, b}?

$$\mathcal{P}(\{a\}) = \{\{\}, \{a\}\}$$

$$\mathcal{P}(\{a, b\}) = \{\{\}, \{a\}, \{b\}, \{a, b\}\}$$

Ohne b: {}, {a}
Mit b: {b}, {a, b}

Power Set · Mit und ohne c

$S = \{a, b, c\} \cdot x = c \cdot R = \{a, b\}$

Ohne c
$\mathcal{P}(R)$
$\{\{\}, \{a\}, \{b\}, \{a, b\}\}$

Mit c
$Y \cup \{c\}$ für jedes Y aus $\mathcal{P}(R)$
$\{\{c\}, \{a, c\}, \{b, c\}, \{a, b, c\}\}$

Jede Teilmenge liegt in genau einer der beiden Gruppen

Power Set · Rekursiver Aufbau

Wie entsteht die Potenzmenge aus einem kleineren Problem?

1. Falls $S = \{\}$: $\{\{\}\}$ zurückgeben
2. Sonst x aus S auswählen und $R = S \setminus \{x\}$ bilden
3. $Q = \mathcal{P}(R)$ genau einmal rekursiv berechnen
4. Ergebnis mit einer Kopie von Q beginnen

Für jedes Y aus Q: $Y \cup \{x\}$ in das Ergebnis einfügen
Ergebnis zurückgeben

Power Set · Terminierung und Vollständigkeit

Warum endet die Rekursion, und fehlen keine Teilmengen?

- Pro Aufruf ein Zeichen weniger; bei der leeren Menge stoppen
- Jede Teilmenge enthält x oder enthält x nicht
  Beide Gruppen sind disjunkt und zusammen vollständig
- Beide Gruppen enthalten gleich viele Teilmengen
  Anzahl verdoppelt sich pro Zeichen: 2^n Teilmengen bei n Zeichen

Optional

Recursion to Iteration 1

CodeExpert

Recursion to Iteration 2

CodeExpert

Floating Point Number Systems

$F^*(2, 3, -4, 4)$ · drei signifikante Binärziffern

Berechnet mit Rundung nach jeder Addition

Zur nächsten Zahl runden; bei Gleichstand ties to even

$$(10 + 0.5) + 0.5$$

$10 = 1.01_2 \times 2^3$
10.5 liegt zwischen 10 und 12
Näher bei 10: $10.5 \to 10$
Nochmals: $10.5 \to 10$
Ergebnis: 10

$$(0.5 + 0.5) + 10$$

$0.5 + 0.5 = 1$
$1 + 10 = 11$
Gleich weit von 10 und 12 entfernt

$10 = 1.01_2 \times 2^3$
$12 = 1.10_2 \times 2^3$
Letztes Bit von 1.10_2 ist gerade
Ergebnis: 12

The Towers of Hanoi

CodeExpert