Tagesprogramm

Beispiele für unveränderliche Datenstrukturen

Aufgabe: Fehler in Spezifikation?

Welche Probleme hat `int toArray(int[] values)` in `FuncList`?

A Reihenfolge der Elemente im Array nicht spezifiziert

B Spezifikation ist widersprüchlich

C Implementierung ist nicht möglich

D Nicht spezifiziert was passiert wenn das Array zu klein ist

Aufgabe: Überprüfung von Array-Grenzen

Warum ist es gefährlich, auf die Überprüfung der Array-Grenzen zu verzichten?

A Weil der Interpreter Grenzen sowieso überprüft ist es nicht gefährlich.

B Weil Pufferüberläufe häufig ausgenutzte Sicherheitslücken sind.

C Weil entsprechene Ausnahmen nicht sicher abgefangen werden können.

D Weil Ausnahmen das Verstehen des Programmablaufs behindern.

Aufgabe: Off-by-One

Warum unterscheiden sich falsche Werte oft nur um 1 von den richtigen?

A Weil wir (halbwegs gut) schätzen statt genau zu überlegen.

B Weil oft unklar ist, ob wir bei 0 oder 1 zu zählen beginnen.

C Weil Operatoren wie < und <= leicht zu verwechseln sind.

D Weil oft nicht klar ist, ob wir das aktuelle Element schon gezählt haben.

Aufgabe: Map

Warum verwenden wir allgemein verwendbare Methoden wie `map` statt spezialisierter Methoden, die gleich das gewünschte machen?

A Weil `map` auch in unvorhergesehenen Fällen anwendbar ist

B Weil `map` effizienter ist

C Weil dadurch nachträgliche Änderungen von Interfaces vermieden werden

Aufgabe: Bäume

Finden Sie in Gruppen zu zwei bis drei Personen Antworten auf diese Frage:

Aus welchen Gründen können Bäume sinnvoll sein, wenn die effizente Suche nicht im Vordergrund steht?

Zeit: 2 Minuten

Aufgabe: FuncTree

Erweitern Sie `FuncTree` um Methoden für

- die Suche nach Elementen
- die Erzeugung veränderter Varianten von Bäumen (Einfügen, Löschen, etc.)
- die Verwendung von `map` (wie in `FuncList`)
- Objektvergleiche und die Erzeugung von Strings