Strukturiertes Programmieren Übung 5 Oliver Heil ITB1\_2a

# „Das fehlende Element“

1. Searchformissingnumber(↓Integer numbers[], ↑Integer missingNum)
2. Integer sumofnumbers = 0
3. Integer totalsum = 0
4. Integer i = 0
6. **while** (i < numbers.length()) **do**
7. sumofnumbers = numbers[i]
8. totalsum += i
9. i++
10. **end**
11. missingNum = (totalsum + i) - sumofnumbers
12. **end**

Lösungsidee

Man addiert alle Werte in dem Array und zeitgleich bildet man eine Summe aus den Zahlen die es sein sollten. Anschließend subtrahiert man die Summe der kompletten Zahlen und die Summe des Arrays. Das Ergebnis ergibt die fehlende Zahl.

numbers = {2,0,1,4}

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| i | sumofnumbers | totalsum | Überprüfung | Befehl |
| 0 | 0 | 0 | i < numbers.length() | - |
| 0 | 2 | 0 | true | sumofnumbers = numbers[i] |
| 0 | 2 | 0 | - | totalsum += i |
| 1 | 2 | 0 | - | i++ |
| 1 | 2 | 0 | i < numbers.length() | - |
| 1 | 2 | 0 | true | sumofnumbers = numbers[i] |
| 1 | 2 | 1 | - | totalsum += i |
| 2 | 2 | 1 | - | i++ |
| 2 | 2 | 1 | i < numbers.length() | - |
| 2 | 3 | 1 | true | sumofnumbers = numbers[i] |
| 2 | 3 | 3 | - | totalsum += i |
| 3 | 3 | 3 | - | i++ |
| 3 | 3 | 3 | i < numbers.length() | - |
| 3 | 7 | 3 | true | sumofnumbers = numbers[i] |
| 3 | 7 | 6 | - | totalsum += i |
| 4 | 7 | 6 | - | i++ |
| 4 | 7 | 6 | i < numbers.length() | - |
| 4 | 7 | 6 | false |  |
| 4 | 7 | 10 | - | totalsum + i |
|  |  |  | missingNum = totalsum  - sumofnumbers | missingNum = 3 |

# Bestimmung von lokalen Minima und Maxima

1. findlocalminmax(↓Integer data[], ↑Integer minimas, ↑Integer maximas)
2. Integer local\_minimas = 0
3. Integer local\_maximas = 0
4. Integer i = 1
5. **While** (i < data.length()-1) **do**
6. **if** (data[i-1] > data[i] && data[i+1] > data[i])**then**
7. ++local\_minimas
8. **end**
9. **if** (data[i-1] < data[i] && data[i+1] < data[i])**then**
10. ++local\_maximas
11. **end**
12. i = i+1
13. **end**
14. minimas = local\_minimas
15. maximas = local\_maximas
16. **end**

Lösungsidee

Ein Zähler läuft durch das ganze Array mit Ausnahme der randwerte. An jedem Punkt wird überprüft ob ein lokales Minima oder ein lokales Maxima besteht.

**data =** {1,3,5,4,6};

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| i | local\_minimas | local\_maxima | Überprüfung | Befehl |
| 1 | 0 | 0 | - | - |
| 1 | 0 | 0 | i < data.length()- 1 | - |
| 1 | 0 | 0 | true | - |
| 1 | 0 | 0 | if minima | - |
| 1 | 0 | 0 | if maxiam | - |
| 2 | 0 | 0 | true | - |
| 2 | 0 | 0 | if minima | - |
| 2 | 0 | 1 | if maxiam | local\_maximas++ |
| 3 | 0 | 1 | true | - |
| 3 | 1 | 1 | if minima | local\_minimas++ |
| 3 | 1 | 1 | if maxiam | - |
| 3 | 1 | 1 | false | - |

## Entfernen von Negativen Werten

1. delnegativnumbers(↓Integer data[], ↑Integer Newsize, ↑Integer Newdata[])
2. Integer validcounter = 0;
3. Integer i = 0
4. Integer n = 0
5. **while** (i< data.length()) **do**
6. **if** (data[i] >= 0) **then**
7. data[n] = data[i]
8. ++n
9. ++validcounter
10. **end**
11. ++i
12. **end**
13. Newsize = validcounter
14. Newdata = data
15. **end**

Lösungsidee

Durch das Array wird schritt für schritt von 0-n durchgegangen. Falls eine negativzahl gefunden wurde, wird diese übersprungen. Bei positiven Zahlen wird der Wert an diesem Punkt auf das nächste gültige Feld geschrieben und der Zähler für diesen Bereich um eins erhöht. Am Ende gibt der Zähler des gültigen Bereiches die neue länge an.

data = {1,-4,3,5,-1};

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| i | n | data[i] | Überprüfung | Befehl |
| 0 | 0 | 1 | i< data.length() | - |
| 0 | 0 | 1 | true | - |
| 0 | 0 | 1 | data[i] >= 0 | - |
| 0 | 0 | 1 | true | - |
| 0 | 1 | 1 | - | n++, data[n]=data[i] |
| 0 | 1 | 1 | i< data.length() | i++ |
| 1 | 1 | -4 | true | - |
| 1 | 1 | -4 | data[i] >= 0 | - |
| 1 | 1 | -4 | false | - |
| 1 | 1 | -4 | i< data.length() | i++ |
| 2 | 1 | 3 | true | - |
| 2 | 1 | 3 | data[i] >= 0 | - |
| 2 | 1 | 3 | true | - |
| 2 | 2 | 3 | - | n++, data[n]=data[i] |
| 2 | 2 | 3 | i< data.length() | i++ |
| 3 | 2 | 5 | true | - |
| 3 | 2 | 5 | data[i] >= 0 | - |
| 3 | 2 | 5 | true | - |
| 3 | 3 | 5 | - | n++, data[n]=data[i] |
| 3 | 3 | 5 | i< data.length() | i++ |
| 4 | 3 | -1 | true | - |
| 4 | 3 | -1 | data[i] >= 0 | - |
| 4 | 3 | -1 | false | - |
| 4 | 3 | -1 | i< data.length() | i++ |

Newdata = {1,3,5,5,-1}

validcounter = 3