Mohamad Raafat Baki

Übung 2 – Arrays :

2.1.1. Implementieren Sie folgenden Algorithmus anzahl\_null in Pseudocode: Eingabe ist ein Array A[1..n] ganzer Zahlen. Ergebnis (return Wert) ist die Anzahl der Elemente des Arrays, die gleich Null sind. Verwenden Sie eine for-Schleife! Wie oft wird diese ausgeführt? Testen Sie Ihre Implementierung mit Arrays der Länge 1, 2, 3 und 4; geben Sie dazu den (selbstgewählten) Input und die Ergebnisse an!

var A = [1,2,3,0,0]

func anzahl\_null(A[1..n])

var b = 0

for i = 1 to n

if A[i] == 0

b = b + 1

return b

print anzahl\_null(A)

2.1.2. Implementieren Sie folgenden Algorithmus put\_last in Pseudocode: Eingabe Array A[1..n] und eine ganze Zahl x. Alle Elemente von A sollen um eine Position nach links verschoben werden, x soll in A an letzter Position (Index n) gespeichert werden. Der alte Inhalt des ersten Elements A[1] geht verloren. Verwenden Sie eine for-Schleife! Wie oft wird diese ausgeführt? Testen Sie Ihre Implementierung mit Arrays der Länge 1, 2, 3 und 4; geben Sie dazu den (selbstgewählten) Input und die Ergebnisse an!

var A = [2,3,5,7]

func printArray(A[1..k])

for j = 1 to k

print A[j]

func put\_last(A[1..n],x)

for i = 1 to n-1

A[i] = A[i+1]

A[n] = x

put\_last(A,123)

printArray(A)

2.1.3. Implementieren Sie folgenden Algorithmus put\_first in Pseudocode: Eingabe Array A[1..n] und eine ganze Zahl x. Alle Elemente von A sollen um eine Position nach rechts verschoben werden, x soll in A an erster Position (Index 1) gespeichert werden. Der alte Inhalt des letzten Elements A[n] geht verloren. Verwenden Sie eine for-Schleife! Wie oft wird diese ausgeführt? Testen Sie Ihre Implementierung mit Arrays der Länge 1, 2, 3 und 4; geben Sie dazu den Input und die Ergebnisse an!

var A = [2,3,5,7]

func printArray(A[1..k])

for j = 1 to k

print A[j]

func put\_first(A[1..n],x)

for i = 0 to n-2

A[n-i] = A[n-(i+1)]

A[1] = x

put\_first(A,0)

printArray(A)

2.1.4. Implementieren Sie folgenden Algorithmus mirror in Pseudocode: Eingabe Array A[1..n]. Der Algorithmus „spiegelt“ das Array, d.h. Das erste Element des Arrays wird mit dem letzten vertauscht, das zweite mit dem vorletzten usw.. Benutzen Sie swap(A,i,j) zum paarweisen Vertauschen zweier Elemente. Testen Sie Ihre Implementierung mit Arrays der Länge 1, 2, 3 und 4; geben Sie dazu den (selbstgewählten) Input und die Ergebnisse an!

var A = [2,3,5,7,8,9]

func printArray(A[1..n])

for i = 1 to n

print A[i]

func swap(A[1..n],i,j)

var t = A[i]

A[i] = A[j]

A[j] = t

func mirror(A[1..n])

for i = 1 to n/2

swap(A,i,n-i+1)

mirror(A)

printArray(A)

2.1.5. Implementieren Sie folgenden Algorithmus max\_diff in Pseudocode: Eingabe ein Array A[1..n] ganzer Zahlen. Ergebnis (return Wert) ist die größte aller Differenzen zwischen je zwei beliebigen Elementen des Arrays.

var A = [2,3,5,7,11]

func max\_diff(A[1..n])

var max = 0

for i = 1 to n

for j = 1 to n

var diff = A[i] - A[j]

if diff > max

max = diff

return max

print max\_diff(A)

2.1.6. Implementieren Sie folgenden Algorithmus min\_diff in Pseudocode: Eingabe ein Array A[1..n] ganzer Zahlen. Ergebnis (return Wert) ist der kleinste Abstand (absoluter Betrag der Differenz) zwischen je zwei verschiedenen Elementen des Arrays. Sortieren darf nicht verwendet werden!

var A = [4,45,981,990,1000]

func min\_diff(A[1..n])

var min = A[1] -A[2]

if min < 0

min = min \* (-1)

var t = 0

for i = 1 to n

for j = 1 to n

t = A[i] - A[j]

if t < 0

t = t \* (-1)

if t < min

min = t

return min

print min\_diff(A)

2.1.7. Ein Array A der Länge n enthalte (in beliebiger Reihenfolge) jede der n+1 ganzen Zahlen von 0 bis n genau einmal, bis auf eine, die fehlt. Implementieren Sie einen Algorithmus missing in Pseudocode, der die fehlende Zahl ermittelt und als Ergebnis zurückgibt. Testen Sie Ihre Implementierung mit Arrays der Länge 1, 2, 3 und 4; geben Sie dazu den (selbstgewählten) Input und die Ergebnisse an! Sortieren darf nicht verwendet werden!

Beispiel: n=4 A = [1|3|0|4] ► Ergebnis: 2.

var A = [1,3,0,4]

func missing(A[1..n])

var s = 0

var r = 0

var miss = 0

for i = 1 to n

s = s + i

for j = 1 to n

r = r + A[j]

miss = s - r

return miss

print missing(A)

2.1.8. Implementieren Sie folgenden Algorithmus even\_first in Pseudocode: Eingabe ein Array A[1..n] natürlicher Zahlen. Der Algorithmus soll die Elemente in einem Array so paarweise vertauschen, dass am Ende alle geraden Zahlen links von allen ungeraden Zahlen stehen. D.h. wenn A[i] gerade ist und A[j] ungerade, dann ist i<j. Testen Sie Ihre Implementierung mit Arrays der Länge 1, 2, 3 und 4; geben Sie dazu den (selbstgewählten) Input und die Ergebnisse an!

var A = [0,1,2,3,4,5]

func printArray(A[1..n])

for i = 1 to n

print A[i]

func swap(A[1..n],i,j)

var t = A[i]

A[i] = A[j]

A[j] = t

func even\_first(A[1..n])

var j = 1

for i = 1 to n

if A[i] mod 2 == 0

swap(A,i,j)

j = j + 1

even\_first(A)

printArray(A)

2.1.9. Implementieren Sie folgenden Algorithmus anzahl\_null2 in Pseudocode: Eingabe ist ein Array A[1..n] ganzer Zahlen. Ergebnis (return Wert) ist die Anzahl der Paare von Elementen des Arrays, deren Summe gleich Null sind. Jedes Paar soll nur einmal gezählt werden, wenn z.B. A[3]+A[5]=0 ist, soll A[5]+A[3]=0 nicht erneut gezählt werden. Testen Sie Ihre Implementierung mit Arrays der Länge 1, 2, 3 und 4; geben Sie dazu den (selbstgewählten) Input und die Ergebnisse an!

var A = [1,2,-2,4,-4,5]

func anzahl\_null2(A[1..n])

var s = 0

for i = 1 to n-1

for j = 1 to n

var r = A[i] + A[j]

if r == 0

s = s + 1

return s

print anzahl\_null2(A)

2.1.12. Implementieren Sie den Luhn-Algorithmus in Pseudocode:

Eingabe ist eine Kreditkartennummer als ein Array A[1..16]. Die Elemente sind Dezimalziffern 0 bis 9. Die Ausgabe soll 1 sein, wenn die Kreditkartennummer gültig ist und 0 sonst. Wikipedia definiert den Luhn-Algorithmus wie folgt (die rechteste Ziffer ist A[16]): „Der Algorithmus läuft in drei Schritten ab. Im ersten Schritt wird jede zweite Ziffer, beginnend bei der zweiten von rechts, verdoppelt. Wenn das Resultat größer als 9 ist, wird die Quersumme des Resultats der Verdopplung gebildet (2 wird zu 4, 7 wird zu 5). Im zweiten Schritt werden alle Zahlen summiert. Schließlich im letzten Schritt wird das Resultat durch 10 dividiert. Wenn der Rest gleich 0 ist, ist die originale Ziffernfolge, die es zu überprüfen galt, gültig.“

Beispiel: 4913268009926396 ist eine gültige Kreditkartennummer.

var A = [4,9,1,3,2,6,8,0,0,9,9,2,6,3,9,6]

func luhn (A[1..n])

var k = n - 1

for i = 1 to n/2

A[k] = A[k] \* 2

while A[k] > 9

A[k] = A[k] - 9

k = k - 2

var s = 0

for j = 1 to n

s = s + A[j]

while s > 0

s = s - 10

if s == 0

return 1

else

return 0

print luhn(A)

2.2. Dezimalzahlen als Arrays

2.2.1. Implementieren Sie einen Algorithmus wert10(A[1..n]), der aus der Dezimaldarstellung A[1..n] die durch diese Dezimaldarstellung repräsentierte natürliche Zahl k berechnet und als Ergebnis zurückgibt.

var A = [3,5,8]

func wert10(A[1..n])

var k = 0

for i = 1 to n

k = k + A[i]

if i != n

k = k \* 10

return k

print wert10(A)

2.2.2. Implementieren Sie einen Algorithmus, der als Eingabe ein Array mit der Dezimaldarstellung einer natürlichen Zahl k hat. Das Ergebnis des Algorithmus soll 1 sein, wenn k eine gerade Zahl ist und 0 sonst. (Rest der Division von k durch 2).

var A = [3,5,8]

func wert10(A[1..n])

var k = 0

for i = 1 to n

k = k + A[i]

if i != n

k = k \* 10

while k > 0

k = k - 2

if k == 0

return 1

else

return 0

print wert10(A)

2.2.3. Implementieren Sie einen Algorithmus, der als Eingabe ein Array mit der Dezimaldarstellung einer natürlichen Zahl k hat. Das Ergebnis des Algorithmus soll der Rest der Division von k durch 3 sein (0, 1 oder 2).

var A = [3,5,3]

func wert10(A[1..n])

var k = 0

for i = 1 to n

k = k + A[i]

if i != n

k = k \* 10

while k >= 3

k = k - 3

return k

print wert10(A)

2.3. Binärzahlen als Arrays

2.3.1. Implementieren Sie einen Algorithmus wert2(A[1..n]) der aus der Binärdarstellung A[1..n] die durch diese Binärdarstellung repräsentierte natürliche Zahl k berechnet und als Ergebnis zurückgibt.

var A = [1,0,1,0,0]

func wert2(A[1..n])

var a = n

var dez = 0

for i = 0 to n-1

A[a] = A[a] \* 2^(0 + i)

a = a - 1

for j = 1 to n

dez = dez + A[j]

return dez

print wert2(A)