

Algorithmen und Datenstrukturen

Aufgabe 1 Zahldarstellung (Beispiellösung)

- a) (i) $1234_{10} = 10011010010_2$ (vii) $14D_{16} = 333_{10} = 101001101_2$
(ii) $1234_{16} = 4660_{10} = 1001000110100_2$ (viii) $934_{10} = 1110100110_2$
(iii) $348_{10} = 101011100_2$ (ix) $523_{16} = 1315_{10} = 10100100011_2$
(iv) $229_{10} = 11100101_2$ (x) $10A_{16} = 266_{10} = 100001010_2$
(v) $825_{10} = 1100111001_2$ (xi) $842_{10} = 1101001010_2$
(vi) $ABC_{16} = 2748_{10} = 101010111100_2$ (xii) $192_{16} = 402_{10} = 110010010_2$
- b) (i) $101100100100_2 = 2852_{10}$ (iii) $CAFE_{16} = 51.966_{10}$
(ii) $111100000011_2 = 3843_{10}$ (iv) $15_{16} = 21_{10}$
- c) (i) $100100101111_2 = 2351_{10} = 92F_{16}$ (iii) $8372_{10} = 20B4_{16}$
(ii) $101011010000_2 = 2768_{10} = AD0_{16}$ (iv) $10_{10} = A_{16}$
- d) (i) $-4_{10}, n = 4$ (iv) $-133_{10}, n = 8$
 $\Rightarrow 1100_2$ \Rightarrow Überlauf!
(ii) $-17_{10}, n = 4$ (v) $-67_{10}, n = 8$
 \Rightarrow nicht darstellbar! $\Rightarrow 10111101_2$
(iii) $-17_{10}, n = 8$ (vi) $-35_{10}, n = 8$
 $\Rightarrow 11101111_2$ $\Rightarrow 11011101_2$
- e) (i) unsigned short (iii) $n = 5$, vorzeichenlos
 $\Rightarrow [0, 2^{16} - 1]$ $\Rightarrow [0, 2^5 - 1]$
(ii) signed long (iv) $n = 7$, vorzeichenbehaftet
 $\Rightarrow [-2^{31}, 2^{31} - 1]$ $\Rightarrow [-2^6, 2^6 - 1]$
- f) (i) 98_{10} dezimal (iii) 123_{10} hexadezimal
 $\Rightarrow 2$ $\Rightarrow 2$
(ii) 45_{10} binär (iv) 93_{10} binär
 $\Rightarrow 6$ $\Rightarrow 7$

- g) (i) $5453.289 \rightsquigarrow s = 0, m = 5.45, e = 3$
 $5452.183 \rightsquigarrow s = 0, m = 5.45, e = 3$
(ii) $934,917,234,384.293 \rightsquigarrow$ *Nicht darstellbar!*
- (iii) i. -253.192 iv. 0.0002039
 $\rightsquigarrow s = 1, m = 2.53192, e = 2$ $\rightsquigarrow s = 0, m = 2.039, e = -4$
ii. 9302.345 v. 42.012
 $\rightsquigarrow s = 0, m = 9.302345, e = 3$ $\rightsquigarrow s = 0, m = 4.2012, e = 1$
iii. -583.194 vi. 10294.284
 $\rightsquigarrow s = 1, m = 5.83194, e = 2$ $\rightsquigarrow s = 0, m = 1.0294284, e = 4$
- (iv) i. $\pi = 3.1415926536\dots$
 $\rightsquigarrow s = 0, m = 3.142, e = 0$
ii. $r_1 = 978.654$
 $\rightsquigarrow s = 0, m = \mathbf{9.787}, e = 2$
iii. $A = 3.142 \cdot 978.7 \cdot 978.7 = 3,009,576.294$ (statt $3,008,903.2521\dots$)
 $\rightsquigarrow s = 0, m = 3.010, e = 6$
iv. $r_2 = \sqrt{\frac{3,010,000}{3.142}} = 978.7688912\dots$ (statt 978.654)
 $\rightsquigarrow s = 0, m = \mathbf{9.788}, e = 2$
v. $r_1 \neq r_2!$

Aufgabe 2 Boolesche Logik (Beispiellösung)

a) NAND (NOT-AND, \uparrow):

a	b	$a \uparrow b = \neg(a \wedge b)$
0	0	1
0	1	1
1	0	1
1	1	0

NOT (\neg): $\neg a = a \uparrow a$:

a	$a \uparrow a$	$\neg a$
0	1	1
1	0	0

AND (\wedge) als NOT-NAND: $a \wedge b = \neg(a \uparrow b) = (a \uparrow b) \uparrow (a \uparrow b)$:

a	b	$a \uparrow b$	$(a \uparrow b) \uparrow (a \uparrow b)$	$a \wedge b$
0	0	1	0	0
0	1	1	0	0
1	0	1	0	0
1	1	0	1	1

OR (\vee) mittels De-Morgan $\neg(a \vee b) = (\neg a) \wedge (\neg b)$ und NOT:

$$\begin{aligned} a \vee b &= \neg((\neg a) \wedge (\neg b)) \\ &= (\neg a) \uparrow (\neg b) \\ &= (a \uparrow a) \uparrow (b \uparrow b) \end{aligned}$$

a	b	$a \uparrow a$	$b \uparrow b$	$(a \uparrow a) \uparrow (b \uparrow b)$	$a \vee b$
0	0	1	1	0	0
0	1	1	0	1	1
1	0	0	1	1	1
1	1	0	0	1	1

NOR (NOT-OR, \downarrow):

a	b	$a \downarrow b = \neg(a \vee b)$
0	0	1
0	1	0
1	0	0
1	1	0

NOT (\neg): $\neg a = a \downarrow a$

OR (\vee) als NOT-OR: $a \vee b = \neg(a \downarrow b) = (a \downarrow b) \downarrow (a \downarrow b)$

AND (\wedge) mittels De-Morgan $\neg(a \wedge b) = (\neg a) \vee (\neg b)$ und NOT:

$$\begin{aligned} a \wedge b &= \neg((\neg a) \vee (\neg b)) \\ &= (\neg a) \downarrow (\neg b) \\ &= (a \downarrow a) \downarrow (b \downarrow b) \end{aligned}$$

Tabellen für NOR analog.

b) Betrachte De-Morgan für NAND:

$$\neg(a \wedge b) = (\neg a) \vee (\neg b)$$

a	b	$a \wedge b$	$\neg(a \wedge b)$	$\neg a$	$\neg b$	$(\neg a) \vee (\neg b)$
0	0	0	1	1	1	1
0	1	0	1	1	0	1
1	0	0	1	0	1	1
1	1	1	0	0	0	0

Analog für De-Morgan für NOR: $\neg(a \vee b) = (\neg a) \wedge (\neg b)$

c) XOR (eXklusives OR, \oplus):

a	b	$a \wedge b$	$\neg(a \wedge b)$	$a \vee b$	$\neg(a \wedge b) \wedge (a \vee b)$	$a \oplus b$
0	0	0	1	0	0	0
0	1	0	1	1	1	1
1	0	0	1	1	1	1
1	1	1	0	1	0	0

d) (i) $a \vee c = 1$

(ii) $b \wedge c = 0$

\Rightarrow Lazy Evaluation!

Aufgabe 3 Algorithmen (Beispiellösung)

- a) $x = -3$: Ausgabe: " $x \leq 0$ ".
 $x = 7$: undefiniert ("Dangling Else").

b) **funktion**(n):

```
    hilfsmfunktion(1, n);
```

```
    hilfsmfunktion( $i$ ,  $n$ ):
```

```
        if ( $i \leq n$ ) {
            print( $i$ );
            hilfsmfunktion( $i + 1$ ,  $n$ );
        }
```

c) $checked = 0$;

```
    while ( $i \bmod 2 == 1 \wedge \neg checked$ ) {
        print("ungerade!");
        checked = 1;
    }
```

- d) (i) // Die Funktion 'verschlüsselt' einen gegebenen Text (in kleinen Buchstaben!) mittels
// ROT13. Dabei wird jedes Zeichen durch dasjenige ersetzt, das 13 Zeichen weiter
// hinten im Alphabet steht.

// Kommentare, sinnvolle Namen und Struktur machen das Verständnis leichter!

```
rot13(string):
```

```
    // Kopiere die Eingabenachricht
    // (Elementarer Verarbeitungsschritt)
    encrypted = string;
```

```
    // Laufe mit Index  $i$  über alle Zeichen der Nachricht
    // (Wiederholung)
```

```
    for  $i = 0$  to size(string)-1 {
        // Prüfe, ob Zeichen bei  $i$  im Bereich 'a' bis 'z' liegt. Diese Bedingung
        // nutzt die Konversion von Schriftzeichen zu Zahlen über die ASCII-
        // Tabelle, in der 'a' bis 'z' hintereinander sortiert sind.
        // (Bedingter Verarbeitungsschritt)
```

```
        if ( $string[i] \geq 'a' \wedge string[i] \leq 'z'$ ) {
            // Berechne den Abstand des Zeichens zu 'a'
            // (Elementarer Verarbeitungsschritt)
            abstand =  $string[i] - 'a'$ ;
```

```
            // 'Rotiere' diesen Abstand um 13 Zeichen; der Teilungsrest hält uns
            // im Bereich der 26 Buchstaben.
```

```
            // (Elementarer Verarbeitungsschritt)
            rotierter_abstand = ( $abstand + 13$ ) mod 26;
```

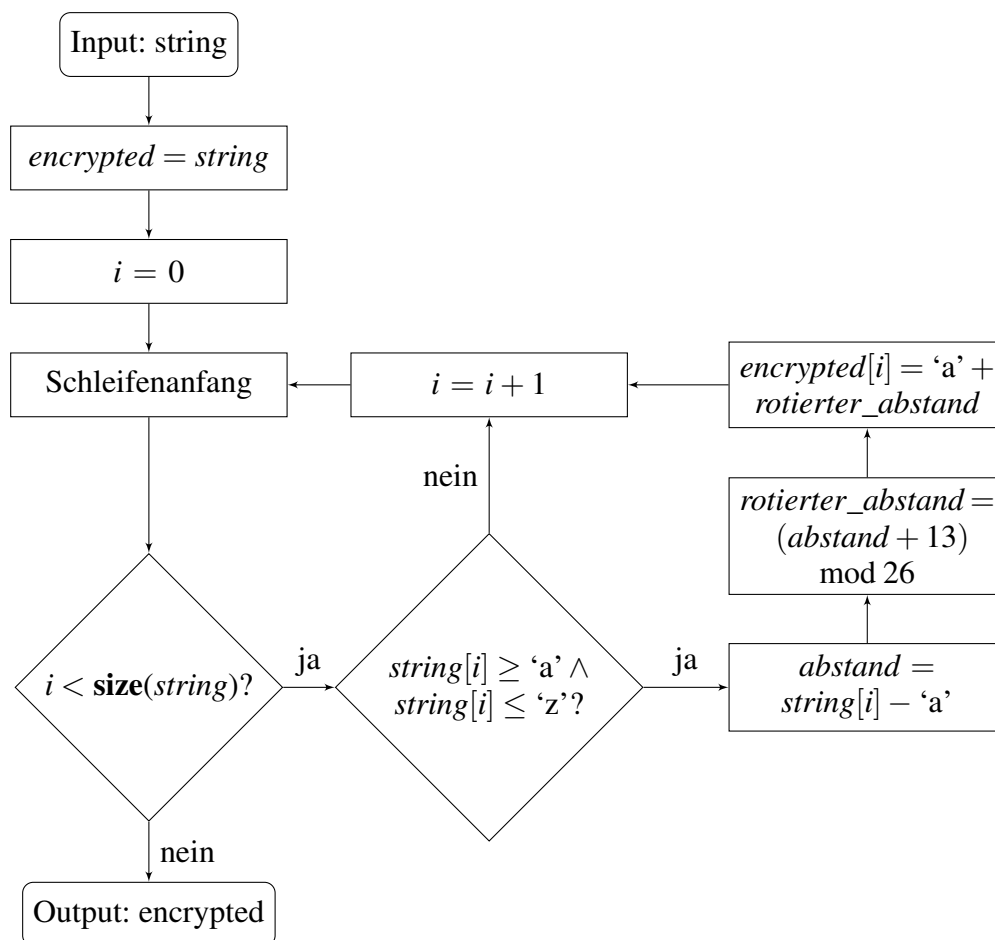
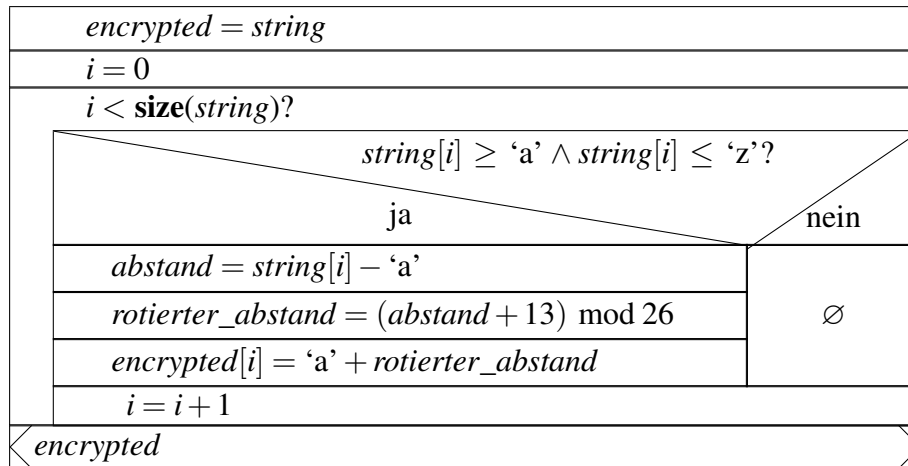
```
            // Der Abstand ist relativ zum Zeichen 'a'
            // (Elementarer Verarbeitungsschritt)
            encrypted[ $i$ ] = 'a' + rotierter_abstand;
```

```
        }
```

```
    // Gebe die verschlüsselte Nachricht zurück
    // (Elementarer Verarbeitungsschritt)
```

```
    return encrypted;
```

rot13(string)



(ii) // Die Funktion extrahiert das größte Element eines Arrays.

max_value(array):

// Prüfe, ob Array leer ist.

// (Bedingter Verarbeitungsschritt)

if (size(array) ≤ 0)

// (Elementarer Verarbeitungsschritt)

return -∞;

// Initialisiere größtes Element auf Wert des ersten Elements.

// (Elementarer Verarbeitungsschritt)

x = array[0];

// Laufe mit Index *i* über alle Elemente des Arrays

```
// (Wiederholung)
for i = 0 to size(array)-1 {
    // Prüfe, ob das aktuelle Element größer als das größte bekannte Element
    // ist.
    // (Bedingter Verarbeitungsschritt)
    if (array[i] > x)
        // Speichere dieses Element.
        // (Elementarer Verarbeitungsschritt)
        x = array[i];
}
```

```
// Gebe das größte Element zurück
// (Elementarer Verarbeitungsschritt)
return x;
```

