

ALGORITHMEN UND DATENSTRUKTUREN

ÜBUNG 4: PROGRAMMIEREN MIT C

Eric Kunze

`eric.kunze@mailbox.tu-dresden.de`

TU Dresden, 20.11.2020

- ▶ Input / Output: `#include <stdio.h>`
- ▶ Variablentypen: z.B. `int`, `float`, `char`
- ▶ arithmetische Operatoren: `+`, `-`, `*`, `/`, `%`
- ▶ Vergleichsoperatoren: `==`, `<`, `<=`, `>`, `>=`
- ▶ Logikoperatoren:
 - ▷ Negation: `!`
 - ▷ Konjunktion: `&&`
 - ▷ Disjunktion: `||`
- ▶ Arrays: `int field[7]` (Indizierung beginnend bei 0)

- ▶ Einlesen: `scanf("%d", &n);`
- ▶ Ausgeben: `printf("n!=%d\n", factorial);`

weiter Informationen:

https:

[//www.tutorials.at/c/03-dateneingabe-ausgabe.html](https://www.tutorials.at/c/03-dateneingabe-ausgabe.html)

► **if-else-Statement:**

bedingte Ausführung eines Statements

```
1 if ( BoolExp ) {  
2     Statement ;  
3 } else {  
4     Statement ;  
5 }
```

► **switch-Statement:**

Fallunterscheidung mit mehr als zwei Fällen

```
1 switch ( Exp ) {  
2     case 0: StatementSeq ; break ;  
3     case 1: StatementSeq ; break ;  
4     default: StatementSeq ;  
5 }
```

- ▶ **while-Statement:** wiederholte Ausführung eines Statements (Schleifenrumpf)

```
1 while ( BoolExp ) {  
2     Statement ;  
3 }
```

- ▶ **do-while-Statement:** vergleichbar mit While-Statement, aber Schleifenbedingung wird nach Rumpf geprüft

```
1 do {  
2     Statement ;  
3 } while ( BoolExp )
```

- **for-Statement:** vor der Schleife steht Anzahl der Schleifendurchläufe fest

```
1 for ( Assignment ; BoolExp ; Assignment ) {  
2     Statement ;  
3 }
```

zum Beispiel:

```
1 for (i = 1; i <= n; i = i + 1){  
2     printf("i = %d\n", i);  
3 }
```

Übungsblatt 4

AUFGABE 1 — TEIL (A)

Maximum: $\max(n, m) = \begin{cases} m & \text{wenn } n < m \\ n & \text{sonst} \end{cases}$

AUFGABE 1 — TEIL (A)

Maximum: $\max(n, m) = \begin{cases} m & \text{wenn } n < m \\ n & \text{sonst} \end{cases}$

```
1 #include <stdio.h>
2 int main() {
3     int x, y, m;
4     scanf("%d", &x);
5     scanf("%d", &y);
6     if ( x < y ) {
7         m = y;
8     } else { // x >= y
9         m = x;
10    }
11    printf("Das Maximum von %d und %d ist %d.\n", x, y,
12           m);
13    return 0;
14 }
```

AUFGABE 1 — TEIL (B)

Fakultät: $n! = n \cdot (n - 1) \cdot (n - 2) \cdot \dots \cdot 2 \cdot 1$

AUFGABE 1 — TEIL (B)

Fakultät: $n! = n \cdot (n - 1) \cdot (n - 2) \cdot \dots \cdot 2 \cdot 1$

```
1 #include <stdio.h>
2 int main() {
3     int n, i, factorial = 1;
4     scanf("%d", &n);
5     for (i = 1; i <= n; i = i + 1) {
6         factorial = factorial * i;
7     }
8     printf("n! = %d\n", factorial);
9     return 0;
10 }
```

AUFGABE 1 — TEIL (C)

AUFGABE 1 — TEIL (C)

```
1 #include <stdio.h>
2 int main() {
3     int i,j,n;
4     scanf("%d", &n);
5     for ( i=1; i<=n ; i=i+1 ) {
6         for ( j=1; j<=n ; j=j+1 ) {
7             printf("%4d", i*j);
8         }
9         printf("\n");
10    }
11    return 0;
12 }
```

AUFGABE 1 — TEIL (D)

AUFGABE 1 — TEIL (D)

```
1 #include <stdio.h>
2 int main(){
3     for (int i = 2; i <= 1000; i++) {
4         int j = 2, prime = 1;
5         while (j*j <= i) {
6             if (i%j == 0) {
7                 prime = 0;
8                 break; // nicht notwendig
9             }
10            j++;
11        }
12        if (prime == 1)
13            printf("%d_list_prim\n", i);
14    }
15    return 0;
16 }
```

AUFGABE 2

► **A:** `matches > 0`

► **B:** `turn == 1`

► **C:**

```
z = (matches - 1) % 4;  
if (z == 0)  
    z = 1  
matches = matches - z;  
turn = 0;
```

► **D:**

```
printf("Der_Computer_zog_%d_Strichhoelzer;\n", z);  
printf("somit_verbleiben_%d_Streichhoelzer.\n", matches);
```

► **E:**

```
scanf("%d", &z);  
matches = matches - z;  
turn = 1;
```

► **F:**

```
if (turn == 1)  
    printf("Der_Computer_gewinnt.");  
else  
    printf("Der_Mensch_gewinnt.");
```