TECHNISCHE UNIVERSITÄT DRESDEN

ALGORITHMEN UND DATENSTRUKTUREN

ÜBUNG 4: PROGRAMMIEREN MIT *C*

Eric Kunze

eric.kunze@mailbox.tu-dresden.de

TU Dresden, 20.11.2020

C

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

INPUT & OUTPUT

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

weiter Informationen:

`https://www.tutorials.at/c/03-dateneingabe-ausgabe.html`

BEDINGUNGEN

- `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 }
```

SCHLEIFEN

- `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 )
```

SCHLEIFEN

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

```
for ( Assignment ; BoolExp ; Assignment ) {
  Statement ;
}
```

zum Beispiel:

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

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

```c
#include <stdio.h>
int main() {
  int x, y, m;
  scanf("%d", &x);
  scanf("%d", &y);
  if ( x < y ) {
    m = y;
  } else { // x >= y
    m = x;
  }
  printf("Das Maximum von %d und %d ist %d.\n", x, y,
      m);
  return 0;
}
```

AUFGABE 1 — TEIL (B)

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

AUFGABE 1 — TEIL (B)

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

```c
#include <stdio.h>
int main(){
   int n, i, factorial = 1;
   scanf("%d", &n);
   for (i = 1; i <= n; i = i + 1){
      factorial = factorial * i;
   }
   printf("n! = %d\n", factorial);
   return 0;
}
```

AUFGABE 1 — TEIL (C)

AUFGABE 1 — TEIL (C)

```c
#include <stdio.h>
int main(){
   int i,j,n;
   scanf("%d", &n);
   for ( i=1; i<=n ; i=i+1 ) {
      for ( j=1; j<=n ; j=j+1 ) {
         printf("%4d", i*j);
      }
      printf("\n");
   }
   return 0;
}
```

AUFGABE 1 — TEIL (D)

AUFGABE 1 — TEIL (D)

```c
#include <stdio.h>
int main(){
  for (int i = 2; i <= 1000; i++) {
    int j = 2, prime = 1;
    while (j*j <= i) {
      if (i%j == 0) {
        prime = 0;
        break; // nicht notwendig
      }
      j++;
    }
    if (prime == 1)
      printf("%d ist prim\n", i);
  }
  return 0;
}
```

AUFGABE 2

- **A:** $\texttt{matches} > 0$
- **B:** $\texttt{turn} == 1$
- **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.");
```