

Ganze Zahlen

Aufgabe 1. Erstellen Sie ein solches Programm:

```
class meineErstenSchritte
{
    static void Main(string[] args)
    {
        Console.WriteLine(5+13);
    }
}
```

Beachten Sie, dass das Semikolon am Ende der Anweisung nicht weggelassen werden darf!

Erklären Sie, welche Bedeutung die Anweisung hat, die in diesem Programm gemacht wird.

Aufgabe 2. Testen Sie ausgehend von obigem Beispiel auch die Grundrechenarten Subtraktion und Multiplikation.

Hinweis: Die Multiplikation erreicht man in C# mit „“.*

Bevor wir zur Division kommen, betrachten wir eine kleine Aufgabe zur Vorbereitung.

Aufgabe 3. Andrea macht eine Schreinerlehre. Sie soll ein 7 m langes Brett in 2 m lange Stücke schneiden. Berechnen Sie, wie viele Stücke sie erhalten wird und wie viel Abfall übrig bleibt.

Aufgabe 4. Division in C#

(a) Welche Ausgaben liefern die folgenden Anweisungen:


```
Console.WriteLine(15/3);
Console.WriteLine(12/4);
Console.WriteLine(20/2);
Console.WriteLine(0/3);
```

(b) Testen Sie diese Anweisungen:

```
Console.WriteLine(21/0);
Console.WriteLine(12/0);
Console.WriteLine(0/0);
```

(c) Betrachten Sie die Ausgaben, die diese Anweisungen liefern und erklären, welche Rechnung hinter diesen steckt.

```
Console.WriteLine(13/2);
```

	<p style="text-align: center;">Lernfeld 5 <u>Software zur Verwaltung von</u> <u>Daten anpassen</u></p>	<p style="text-align: right;">27.09.2022 ITF-22-e</p>
---	---	---

```
Console.WriteLine(21/5);
Console.WriteLine(27/5);
Console.WriteLine(7/2);
```

Tipp: Rufen Sie sich nochmal Aufgabe 3 ins Gedächtnis!

- (d) In C# kann man nicht nur mit „/“ eine Division erreichen, sondern auch mit „%“. Diese müssen wir uns ebenfalls genau anschauen. Testen Sie dazu diese Beispiele:

```
Console.WriteLine(13%2);
Console.WriteLine(21%5);
Console.WriteLine(27%5);
Console.WriteLine(7%2);
```

Betrachten Sie noch weitere Beispiele und erklären dann, welche Bedeutung die Division mit „%“ hat.

Aufgabe 5. Andrea soll ein 734 cm langes Brett in 13 cm lange Stücke schneiden.

Berechne Sie mit C#, wie viele Stücke sie erhalten wird und wie viel Abfall übrig bleibt.

Aufgabe 6. Genau um 0 Uhr wurde eine besonders aufwändige Berechnung an einem Großrechner gestartet. Die Berechnung hat insgesamt 143 Stunden gedauert.

Ermitteln Sie mithilfe von C#, wie spät es war als die Rechnung beendet wurde. Ermitteln Sie auch, wie viele volle Tage der Großrechner an der Rechnung gearbeitet hat.

Aufgabe 7. Angenommen, es ist jetzt genau 14 Uhr.

Berechnen Sie mithilfe von C#, wie spät es in 17 Stunden sein wird.

Aufgabe 8. Neben der Anweisung

```
Console.Write(„Anweisung“);
```


gibt es auch noch die Nachfolgende um Daten in der Konsole anzeigen zu lassen.

```
Console.WriteLine(„Anweisung“);
```

Erkläre Sie den Unterschied zwischen beiden. Wo wäre die jeweilige Anweisung von Vorteil?

Dezimalzahlen

Aufgabe 9. Betrachten Sie ausgehend von diesem Beispiel die Addition, Subtraktion und Multiplikation von Dezimalzahlen in C#:

	<p style="text-align: center;">Lernfeld 5 <u>Software zur Verwaltung von</u> <u>Daten anpassen</u></p>	<p style="text-align: right;">27.09.2022 ITF-22-e</p>
---	---	---

```
Console.WriteLine(2.5+5.1);
```

Hinweis: Beachten Sie, dass man einen Punkt statt eines Kommas verwendet!

Aufgabe 10. Vergewissern Sie sich anhand einiger Beispiele, dass eine solche Anweisung tatsächlich die ganz gewöhnliche Division zweier Dezimalzahlen darstellt:

```
Console.WriteLine(8.5/2.5);
```

Aufgabe 11. Andrea macht immer noch ihre Schreinerlehre. Sie soll ein 2,5 m langes Brett in 1,2 m lange Stücke schneiden. Berechnen Sie, wie viele Stücke sie erhalten wird und wie viel Abfall übrig bleibt.

Aufgabe 12. Man kann in C# auch Dezimalzahlen mithilfe von % dividieren. Ermitteln Sie, wie dies uns bei Aufgabe 11 helfen kann.

Aufgabe 13. Andrea soll ein 734,5 cm langes Brett in 13,2 cm lange Stücke schneiden.

Berechne Sie mit C#, wie viele Stücke sie erhalten wird und wie viel Abfall übrig bleibt.

Aufgabe 14. Angenommen, es ist jetzt genau 14:45 Uhr.

Berechne Sie mithilfe von C#, wie spät es in 17,5 Stunden sein wird.

Variablen

Aufgabe 15. Erstellen Sie ein Programm, in dem eine Variable namens „meineZahl“ vom Typ „**int**“ angelegt wird und in der die Zahl „5“ abgelegt wird.

Fügen Sie danach noch eine zweite Variable mit Namen „meineZweiteZahl“ hinzu, die ebenfalls vom Typ „**int**“ ist. Speichern Sie in dieser die Zahl „10“.

(a) Testen und erklären Sie, was die folgenden Anweisungen bewirken:

```
Console.WriteLine ( meineZahl + meineZweiteZahl ) ;
Console.WriteLine ( meineZahl - meineZweiteZahl ) ;
Console.WriteLine ( meineZahl * meineZweiteZahl ) ;
Console.WriteLine ( meineZahl / meineZweiteZahl ) ;
Console.WriteLine ( meineZahl % meineZweiteZahl ) ;
```


(b) Testen und erklären Sie, was die folgende Anweisung bewirkt:

```
meineZahl = meineZahl + 3;
```

Tipp: Lassen Sie sich den Wert von „meineZahl“ zur Kontrolle nach dieser Anweisung ausgeben!

Aufgabe 16. Ermitteln Sie zuerst im Kopf und dann mit C#, welchen Wert die Variable „meineZahl“ am Ende dieser Anweisungen hat:

```
int meineZahl;
meineZahl = 5;
```

	<p style="text-align: center;">Lernfeld 5 <u>Software zur Verwaltung von</u> <u>Daten anpassen</u></p>	<p style="text-align: right;">27.09.2022 ITF-22-e</p>
---	---	---

```
meineZahl = meineZahl * 2;
meineZahl = meineZahl - 1;
```

Aufgabe 17. Ermitteln Sie zuerst im Kopf und dann mit C#, welche Werte die Variablen „meineZahl“ und „meineZweiteZahl“ am Ende dieser Anweisungen haben:

```
int meineZahl;
meineZahl = 5;
int meineZweiteZahl;
meineZweiteZahl = 10;
meineZahl = meineZahl * meineZweiteZahl;
meineZahl = meineZahl - 10;
meineZweiteZahl = meineZweiteZahl + meineZahl;
```

Aufgabe 18. Zu Beginn des Schuljahres wird in der Klasse 5b eine neue Klassenkasse angelegt. Weil die Fünftklässler noch nicht gut mit Dezimalzahlen umgehen können, wird darauf geachtet, dass immer nur ganze Eurobeträge eingezahlt oder entnommen werden. Im Laufe des Schuljahres geschehen nun diese Vorgänge:

- Anfangs wird die Klassenkasse eingerichtet. Sie ist noch leer.
- 27 Schüler zahlen ihren Beitrag von 5 Euro ein.
- 3 Nachzügler zahlen auch noch ihren Beitrag. Sie müssen aber wegen der Verspätung 6 Euro einzahlen.
- Mit dem Geld der Klassenkasse wird Bastelmaterial gekauft. Es kostet insgesamt 30 Euro.
- Im Sommer geht die Klasse ins Schwimmbad. Der Eintritt ins Schwimmbad kosten insgesamt 60 Euro und wird mit der Klassenkasse finanziert.

- (a) Simulieren Sie die obige Situation mithilfe von Variablen in C#.
- (b) Berechnen Sie mit Hilfe von C#, wie viel Geld jeder der 30 Schüler am Ende des Schuljahres aus der Klassenkasse ausgezahlt bekommt, wenn man das Geld gleichmäßig auf alle aufteilt aber nur ganzzahlige Eurobeträge auszahlt. Ermitteln Sie, wie viel dann in der Klassenkasse übrig bleibt.
- (c) Berechnen Sie, wie viel Geld jeder der 30 Schüler am Ende des Schuljahres aus der Klassenkasse ausgezahlt bekommt, wenn man das Geld gleichmäßig auf alle aufteilt ohne einen Rest in der Kasse zu lassen.

Variablen vergleichen

Aufgabe 19. Erstellen Sie ein Programm, in dem zwei Variablen vom Typ „**int**“ mit den Namen „meineZahl“ und „deineZahl“ angelegt werden. Speichern Sie in der Variablen „meineZahl“ die Zahl „5“ und in „deineZahl“ die Zahl „10“.

Testen und erklären Sie dann, was die folgenden Anweisungen bewirken:

```
Console.WriteLine ( meineZahl < deineZahl );  
Console.WriteLine ( meineZahl > deineZahl );  
Console.WriteLine ( meineZahl == 8 );  
Console.WriteLine ( meineZahl == 5 );
```

Aufgabe 20. In C# gibt es die folgenden sechs Möglichkeiten, zwei Zahlen miteinander zu vergleichen:

== != < > <= >=


Testen Sie diese anhand verschiedener (eigener) Beispiele.

Halten Sie dann in einer Tabelle fest, welche Bedeutung die verschiedenen Symbole, oder wie man auch sagt *Vergleichsoperatoren*, haben.

Aufgabe 21. Zwei Personen spielen ein kleines Spiel, bei dem man ständig Punkte gewinnen aber auch wieder verlieren kann. Es gibt nur ganzzahlige Punkte. Wir betrachten den folgenden Spielverlauf:

- Anfangs haben beide Spieler noch 0 Punkte.
- Der erste Spieler gewinnt 10 Punkte, der zweite Spieler 5 Punkte.
- Der erste Spieler verliert 3 Punkte, der zweite Spieler verdoppelt seine Punktzahl.
- Der erste Spieler gewinnt 11 Punkte, der zweite Spieler verliert einen Punkt.
- Der erste Spieler darf zu seinen Punkten die Punktzahl des zweiten Spielers addieren, die Punktzahl des zweiten Spielers bleibt dabei unverändert.
- Der erste Spieler verliert 9 Punkte, der zweite Spieler gewinnt 9 Punkte.

- (a) Simulieren Sie obige Situation mithilfe von Variablen in C#.
- (b) Ermitteln Sie danach, ob beide Spieler am Ende dieselbe Punktzahl haben ohne die Punktzahlen konkret anzeigen zu lassen. Ermittle gegebenenfalls, welcher Spieler die größere Punktzahl hat — auch hier sollen die Punktzahlen nicht angezeigt werden, sondern die obigen Vergleichsoperatoren benutzt werden.
- (c) Ermitteln Sie mithilfe der Vergleichsoperatoren, ob einer der Spieler mehr als 20 Punkte hat.

	<p style="text-align: center;">Lernfeld 5 <u>Software zur Verwaltung von</u> <u>Daten anpassen</u></p>	<p style="text-align: right;">27.09.2022 ITF-22-e</p>
---	---	---

*Bisher haben wir nur Variablen vom Typ **int** benutzt. Der Umgang mit denen vom Typ **double** funktioniert aber ganz genauso. Auch dazu ein Beispiel.*

Aufgabe 22. Ein Sprinter absolviert drei Trainingsläufe. Dabei erreicht er die folgenden Zeiten die alle in **Sekunden** gemessen wurden.

10,1 10,5 10,2

- (a) Speichern Sie diese drei Zeiten in drei Variablen vom Typ **double**.
- (b) Ermitteln Sie mit C# den Mittelwert dieser drei Laufzeiten.

Ein zweiter Sprinter erreicht im Training die nachfolgenden drei Zeiten, die ebenfalls in Sekunden gemessen werden.

9,9 10,3 10,4

Ermitteln Sie mithilfe der Vergleichsoperatoren, ob sein Mittelwert genau so groß, größer oder kleiner als der des ersten Sprinters ist.

Zusatzaufgabe 23. Versuchen Sie einmal, den Wert 0,5 in einer Variablen vom Typ **int** zu speichern und erklären Sie die Fehlermeldung, die dadurch ausgelöst wird.

Speichern Sie dann einmal den Wert 7 in einer Variablen vom Typ **double**, und lassen Sie danach den gespeicherten Wert noch einmal anzeigen und achten dabei auf Details. Erklären Sie die Ausgabe, die man erhält.

Zusatzaufgabe 24. Wie wir gesehen haben, ist C# sehr streng bei der Unterscheidung zwischen **int**- und **double**-Werten. Nun stellt sich die Frage, ob C# bereit ist, solche Werte miteinander zu vergleichen.

Ermitteln Sie selbst, ob C# (mithilfe der Vergleichsoperatoren) **int**- und **double**-Werte korrekt miteinander vergleichen kann.

Zusatzaufgabe 25. Denken Sie sich selbst eine Situation aus, die man mithilfe der uns bisher bekannten Mittel in C# simulieren kann.

Halten Sie diese Situation schriftlich fest, damit sie später an andere verteilt werden kann.