

# Anhang: Beispielprogramme

Die folgenden Programme sind nach Themen gruppiert.

## Zahlentheorie und Zahlenfolgen

- [Primzahltest](#)
- [Primzahlen bis N](#)
- [Fakultaet](#)
- [Potenz](#)
- [Fibonacci](#)
- [KGT](#)
- [GGT](#)
- [Summe 1 bis N](#)
- [Summe gerade/ungerade](#)
- [Arithmetische Folge](#)
- [Geometrische Folge](#)
- [Quadratzahlen](#)
- [Kubikzahlen](#)

## Grundrechenarten und Operatoren

- [Ganzzahl-Division mit Rest](#)
- [Multiplikation durch Addition](#)
- [Division durch Subtraktion](#)
- [Modulo durch Subtraktion](#)
- [Summenquadrat vs. Quadratsumme](#)

## Vergleich und Logik

- [Zahlenvergleich](#)
- [Paritaetstest](#)
- [Betrag](#)
- [Abstand zweier Zahlen](#)
- [Swap](#)

## Ziffern und Darstellungen

- [Ziffernsumme](#)
- [Zifferanzahl](#)
- [Palindromtest](#)
- [Umkehrung einer Zahl](#)

# Geometrie und Physik

- [Rechteck Umfang und Flaeche](#)
- [Quadrat Umfang und Flaeche](#)
- [Kreis Umfang und Flaeche](#)
- [Dreiecksumfang](#)
- [Pythagoras](#)
- [Geschwindigkeit](#)

# Mathe-Funktionen (Festkomma)

- [Mathematische Funktionen](#)

# Statistik und Auswertung

- [Mittelwert](#)
- [Minimum und Maximum](#)
- [Einfache Statistik](#)

# Alltag und Wirtschaft

- [Dreisatz](#)
- [Prozentrechnung](#)
- [Zins \(einfach\)](#)
- [Zinseszins](#)
- [Rabattberechnung](#)
- [Versandkostenstaffel](#)
- [Waehrungsumrechnung](#)
- [Temperaturumrechnung](#)
- [Einheitenumrechnung](#)
- [Wechselgeld](#)
- [Notenbewertung](#)
- [BMI](#)

# Zeit und Kalender

- [Zeitumrechnung](#)
- [Uhrzeitdifferenz](#)
- [Tage zu Wochen](#)
- [Altersberechnung](#)

# Gleichungen und Interpolation

- [Lineare Gleichung](#)

- [Quadratische Gleichung](#)
- [Lineare Interpolation](#)

## Sonstiges

- [Countdown](#)
- [Schleifenzaehler](#)

## Schwierigkeit und Zeit

Legende: E = Einfach, M = Mittel, A = Anspruchsvoll.

Programm	Stufe	Zeit
Summe 1 bis N	E	10-15 min
Zahlenvergleich	E	10-15 min
Paritaetstest	E	10-15 min
Betrag	E	5-10 min
Abstand zweier Zahlen	E	5-10 min
Countdown	E	10-15 min
Arithmetische Folge	E	10-15 min
Geometrische Folge	M	15-20 min
Fibonacci	M	20-25 min
GGT	M	20-25 min
KGT	M	20-25 min
Primzahltest	M	20-25 min
Primzahlen bis N	A	25-35 min
Prozentrechnung	E	10-15 min
Zins (einfach)	E	10-15 min
Zinseszins	M	15-20 min

<b>Programm</b>	<b>Stufe</b>	<b>Zeit</b>
Rechteck Umfang und Flaeche	E	10-15 min
Kreis Umfang und Flaeche	M	15-20 min
Lineare Gleichung	M	15-20 min
Quadratische Gleichung	A	25-35 min
Ziffernsumme	M	15-20 min
Palindromtest	A	25-35 min

# Anhang: Beispiele ausfuehren

Die Beispiele liegen als PL/0-Quelltext in der Dokumentation. Zum Ausfuehren kopiere den Code in eine Datei und nutze die CLI.

## Schritte

1. Datei anlegen, z. B. `example.pl0`.
2. Programm einfuegen und mit `.` abschliessen.
3. Ausfuehren: `dotnet run --project src/Pl0.Cli -- run example.pl0`

## Tipp

Mit `--list-code` kannst du die generierte P-Code-Liste ausgeben.

# Anhang: Empfohlene Reihenfolge

Diese Reihenfolge ist fuer Auszubildende gedacht.

## Einstieg

1. [Summe 1 bis N](#)
2. [Zahlenvergleich](#)
3. [Paritaetstest](#)
4. [Betrag](#)
5. [Abstand zweier Zahlen](#)

## Schleifen und Folgen

6. [Countdown](#)
7. [Arithmetische Folge](#)
8. [Geometrische Folge](#)
9. [Fibonacci](#)

## Zahlentheorie

10. [GGT](#)
11. [KGT](#)
12. [Primzahltest](#)
13. [Primzahlen bis N](#)

## Anwendungen

14. [Prozentrechnung](#)
15. [Zins \(einfach\)](#)
16. [Zinseszins](#)
17. [Rechteck Umfang und Flaeche](#)
18. [Kreis Umfang und Flaeche](#)

# Anhang: Glossar

Begriffe aus PL/0, P-Code und der VM.

## Address

Die Speicherposition eines Wertes in der VM. Siehe auch: [VM-Instruction-Set](#).

## Argument

Das dritte Feld einer P-Code-Instruktion (OPCODE LEVEL ARG). Siehe auch: [P-Code-Handbuch](#).

## Basiszeiger (B)

Zeigt auf den aktuellen Stack-Frame. Siehe auch: [P-Code-Handbuch](#).

## Code-Area

Der Speicherbereich mit den P-Code-Instruktionen. Siehe auch: [P-Code-Handbuch](#).

## Compiler

Uebersetzt PL/0 in P-Code. Siehe auch: [Architektur](#).

## Dialekt

Variante der Sprache, z. B. classic oder extended. Siehe auch: [PL0-Handbuch](#).

## EndOfFile

Token, das das Dateiende markiert. Siehe auch: [API-Referenz](#).

## Instruktionszeiger (P)

Program Counter, zeigt auf die naechste Instruktion. Siehe auch: [P-Code-Handbuch](#).

## Lexer

Zerlegt Quelltext in Token. Siehe auch: [PL0-Handbuch](#).

## Level

Lexikalische Verschachtelungsebene fuer Variablenzugriff. Siehe auch: [P-Code-Handbuch](#).

## OPR

Opcode fuer arithmetische/vergleichende Operationen. Siehe auch: [OPR Referenz](#).

# Parser

Analysiert Token und erzeugt P-Code. Siehe auch: [PL0-Handbuch](#).

# P-Code

Zwischencode, den die VM ausfuehrt. Siehe auch: [P-Code-Handbuch](#).

# Stack

LIFO-Speicher fuer Werte und lokale Variablen. Siehe auch: [P-Code-Handbuch](#).

# Stack-Top (T)

Zeigt auf das aktuelle Stackende. Siehe auch: [P-Code-Handbuch](#).

# Symboltabelle

Speichert Deklarationen von Konstanten/Variablen/Prozeduren. Siehe auch: [PL0-Handbuch](#).

# Token

Kleinstes syntaktisches Element (z. B. Ident, Number). Siehe auch: [PL0-Handbuch](#).

# VM

Virtuelle Maschine zur Ausfuehrung von P-Code. Siehe auch: [VM-Instruction-Set](#).

# Anhang: Fibonacci

Dieses Beispiel berechnet die Fibonacci-Folge.

## Programm

```
var a, b, i, temp;  
begin  
    a := 0;  
    b := 1;  
    i := 0;  
    while i < 10 do  
        begin  
            ! a;  
            temp := a + b;  
            a := b;  
            b := temp;  
            i := i + 1  
        end  
    end.  
end.
```

## Erklaerung

- **a** und **b** halten die aktuellen Werte.
- **temp** speichert die Zwischensumme.
- Die Schleife gibt die ersten 10 Werte aus.

## Details

- Iterative Berechnung mit zwei Arbeitsvariablen.
- Gibt die ersten 10 Werte aus.

## Beispiel

Eingabe:

(no input)

Ausgabe:

0 1 1 2 3 5 8 13 21 34

## Testfaelle

- (no input) -> 0 1 1 2 3 5 8 13 21 34
- (no input) -> 0 1 1 2 3 5 8 13 21 34

## Ausfuehrung

Beispiel:

```
dotnet run --project src/Pl0.Cli -- run example.pl0
```

Tipp: Füge **--list-code** hinzu, um den P-Code zu sehen.

# Anhang: KGT

Dieses Beispiel berechnet das kleinste gemeinsame Vielfache (KGT) zweier Zahlen.

## Programm

```
var a, b, x, y;
procedure gcd;
var t;
begin
  while y # 0 do
  begin
    t := x;
    x := y;
    y := t - (t / y) * y
  end
end;
begin
  ? a;
  ? b;
  x := a;
  y := b;
  call gcd;
  ! (a / x) * b
end.
```

## Erklaerung

- Der groesste gemeinsame Teiler (GGT) wird mit dem Euklidischen Algorithmus ermittelt.
- Aus GGT wird das KGT berechnet: **KGT = (a / GGT) \* b.**

## Details

- KGT aus GGT abgeleitet:  $(a/GGT)*b$ .
- Nutzt Euklidischen Algorithmus.

## Beispiel

Eingabe:

12 18

Ausgabe:

## Testfaelle

- 6 8 -> 24
- 7 3 -> 21

## Ausfuehrung

Beispiel:

```
dotnet run --project src/Pl0.Cli -- run example.pl0
```

Tipp: Fuege **--list-code** hinzu, um den P-Code zu sehen.

# Anhang: GGT

Dieses Beispiel berechnet den groessten gemeinsamen Teiler (GGT) zweier Zahlen.

## Programm

```
var x, y, t;
begin
    ? x;
    ? y;
    while y # 0 do
        begin
            t := x;
            x := y;
            y := t - (t / y) * y
        end;
    ! x
end.
```

## Erklaerung

- Die Schleife implementiert den Euklidischen Algorithmus.
- Ergebnis ist der Wert von **x**.

## Details

- Euklidischer Algorithmus per wiederholter Subtraktion/Division.
- Ergebnis ist der GGT.

## Beispiel

Eingabe:

54 24

Ausgabe:

6

## Testfaelle

- 8 12 -> 4
- 7 3 -> 1

# Ausfuehrung

Beispiel:

```
dotnet run --project src/Pl0.Cli -- run example.pl0
```

Tipp: Füge **--list-code** hinzu, um den P-Code zu sehen.

# Anhang: Mathematische Funktionen

Hinweis: PL/0 arbeitet nur mit Integern. Die folgenden Beispiele nutzen einfache Festkomma-Naeherungen und sind didaktisch gedacht.

## sin (naehereungsweise)

```
const scale = 1000, six = 6;
var x, x2, x3, sinx;
begin
    ? x;          /* x in Milliradian */
    x2 := (x * x) / scale;
    x3 := (x2 * x) / scale;
    sinx := x - (x3 / six);
    ! sinx
end.
```

## cos (naehereungsweise)

```
const scale = 1000, two = 2;
var x, x2, cosx;
begin
    ? x;
    x2 := (x * x) / scale;
    cosx := scale - (x2 / two);
    ! cosx
end.
```

## tan (naehereungsweise)

```
var sinx, cosx;
begin
    ? sinx;
    ? cosx;
    if cosx = 0 then
        ! 0
    else
        ! (sinx / cosx)
end.
```

## Kreisberechnung (Flaeche)

```
const pi = 3141, scale = 1000;
var r, area;
begin
    ? r;
    area := (pi * r * r) / scale;
```

```
! area  
end.
```

## Details

- Festkomma-Naeherungen fuer trigonometrische Funktionen.
- Ergebnisse sind didaktisch, nicht numerisch exakt.

## Beispiel

Eingabe:

1000

Ausgabe:

approx

## Testfaelle

- 1000 -> approx
- 2000 -> approx

## Ausfuehrung

Beispiel:

```
dotnet run --project src/Pl0.Cli -- run example.pl0
```

Tipp: Fuege **--list-code** hinzu, um den P-Code zu sehen.

# Anhang: BMI

BMI in Festkomma (gewicht/(groesse^2)).

## Programm

```
const scale = 100;  
var w, h, bmi;  
begin  
    ? w;  
    ? h;  
    bmi := (w * scale) / ((h * h) / scale);  
    ! bmi  
end.
```

## Erklaerung

- Eingaben werden mit `?` gelesen.
- Ausgaben erfolgen ueber `!`.
- Alle Berechnungen sind ganzzahlig.

## Details

- Festkomma mit `scale=100`.
- Groesse in cm, Gewicht in kg.

## Beispiel

Eingabe:

`80 180`

Ausgabe:

`24`

## Testfaelle

- `60 170 -> 20`
- `90 180 -> 27`

## Ausfuehrung

Beispiel:

```
dotnet run --project src/Pl0.Cli -- run example.pl0
```

Tipp: Füge `--list-code` hinzu, um den P-Code zu sehen.

# Anhang: Zins (einfach)

Berechnet einfachen Jahreszins.

## Programm

```
var k, p, z;  
begin  
    ? k;  
    ? p;  
    z := (k * p) / 100;  
    ! z  
end.
```

## Erklaerung

- Eingaben werden mit **?**  gelesen.
- Ausgaben erfolgen ueber **!** .
- Alle Berechnungen sind ganzzahlig.

## Details

- Einfache Verzinsung, kein Zinseszins.
- **p** als Prozentwert verwenden.

## Beispiel

Eingabe:

**1000 5**

Ausgabe:

**50**

## Testfaelle

- 200 10 -> 20
- 1500 3 -> 45

## Ausfuehrung

Beispiel:

```
dotnet run --project src/Pl0.Cli -- run example.pl0
```

Tipp: Füge `--list-code` hinzu, um den P-Code zu sehen.

# Anhang: Wechselgeld

Greedy-Wechselgeld fuer feste Muenzen.

## Programm

```
var betrag, c, n;  
begin  
    ? betrag;  
    c := betrag / 50;  
    betrag := betrag - c * 50;  
    n := betrag / 20;  
    betrag := betrag - n * 20;  
    ! c;  
    ! n  
end.
```

## Erklaerung

- Eingaben werden mit **?** gelesen.
- Ausgaben erfolgen ueber **!**.
- Alle Berechnungen sind ganzzahlig.

## Details

- Greedy fuer 50 und 20 Einheiten.
- Rest wird nicht weiter aufgeteilt.

## Beispiel

Eingabe:

130

Ausgabe:

**2 1**

## Testfaelle

- 70 -> 11
- 40 -> 0 2

# Ausfuehrung

Beispiel:

```
dotnet run --project src/Pl0.Cli -- run example.pl0
```

Tipp: Füge **--list-code** hinzu, um den P-Code zu sehen.

# Anhang: Lineare Interpolation

Berechnet  $y = y_0 + (x-x_0) * (y_1-y_0)/(x_1-x_0)$ .

## Programm

```
var x0, y0, x1, y1, x, y;  
begin  
    ? x0;  
    ? y0;  
    ? x1;  
    ? y1;  
    ? x;  
    y := y0 + (x - x0) * (y1 - y0) / (x1 - x0);  
    ! y  
end.
```

## Erklaerung

- Eingaben werden mit **?** gelesen.
- Ausgaben erfolgen ueber **!**.
- Alle Berechnungen sind ganzzahlig.

## Details

- Lineare Interpolation zwischen  $(x_0,y_0)$  und  $(x_1,y_1)$ .
- $x_1$  darf nicht gleich  $x_0$  sein.

## Beispiel

Eingabe:

0 0 10 100 5

Ausgabe:

50

## Testfaelle

- 0 0 10 100 0 -> 0
- 0 0 10 100 10 -> 100

# Ausfuehrung

Beispiel:

```
dotnet run --project src/Pl0.Cli -- run example.pl0
```

Tipp: Füge **--list-code** hinzu, um den P-Code zu sehen.

# Anhang: Waehrungsumrechnung

Umrechnung mit festem Kurs.

## Programm

```
const kurs = 110;  
var eur, jpy;  
begin  
    ? eur;  
    jpy := eur * kurs;  
    ! jpy  
end.
```

## Erklaerung

- Eingaben werden mit **?** gelesen.
- Ausgaben erfolgen ueber **!**.
- Alle Berechnungen sind ganzzahlig.

## Details

- Fester Kurs als Konstante.
- Multiplikation ohne Rundung.

## Beispiel

Eingabe:

10

Ausgabe:

1100

## Testfaelle

- 1 -> 110
- 0 -> 0

## Ausfuehrung

Beispiel:

```
dotnet run --project src/Pl0.Cli -- run example.pl0
```

Tipp: Füge `--list-code` hinzu, um den P-Code zu sehen.

# Anhang: Dreisatz

Berechnet x aus a:b = c:x.

## Programm

```
var a, b, c, x;  
begin  
    ? a;  
    ? b;  
    ? c;  
    x := (b * c) / a;  
    ! x  
end.
```

## Erklaerung

- Eingaben werden mit **?** gelesen.
- Ausgaben erfolgen ueber **!**.
- Alle Berechnungen sind ganzzahlig.

## Details

- Direkte Proportion  $a:b = c:x$ .
- a darf nicht 0 sein.

## Beispiel

Eingabe:

2 10 6

Ausgabe:

30

## Testfaelle

- 4 20 5 -> 25
- 1 3 2 -> 6

## Ausfuehrung

Beispiel:

```
dotnet run --project src/Pl0.Cli -- run example.pl0
```

Tipp: Füge `--list-code` hinzu, um den P-Code zu sehen.

# Anhang: Geometrische Folge

Gibt eine geometrische Folge aus.

## Programm

```
var a, q, n, i, x;
begin
    ? a;
    ? q;
    ? n;
    i := 0;
    x := a;
    while i < n do
        begin
            ! x;
            x := x * q;
            i := i + 1
        end
    end.
```

## Erklaerung

- Eingaben werden mit `?` gelesen.
- Ausgaben erfolgen ueber `!`.
- Alle Berechnungen sind ganzzahlig.

## Details

- Startwert `a` und Faktor `q`.
- Gibt `n` Werte aus.

## Beispiel

Eingabe:

`2 3 4`

Ausgabe:

`2 6 18 54`

## Testfaelle

- 1 2 3 -> 1 2 4
- 3 3 2 -> 3 9

## Ausfuehrung

Beispiel:

```
dotnet run --project src/Pl0.Cli -- run example.pl0
```

Tipp: Füge **--list-code** hinzu, um den P-Code zu sehen.

# Anhang: Primzahltest

Testet, ob eine Zahl prim ist.

## Programm

```
var n, i, isPrime;  
begin  
    ? n;  
    if n < 2 then  
        isPrime := 0;  
    if n >= 2 then  
        begin  
            isPrime := 1;  
            i := 2;  
            while i * i <= n do  
                begin  
                    if (n / i) * i = n then  
                        isPrime := 0;  
                    i := i + 1  
                end  
            end;  
        ! isPrime  
    end.
```

## Erklaerung

- Eingaben werden mit **?** gelesen.
- Ausgaben erfolgen ueber **!**.
- Alle Berechnungen sind ganzzahlig.

## Details

- Schleife testet Teilbarkeit nur bis  $\sqrt{n}$ .
- Ergebnis: 1 = prim, 0 = nicht prim.

## Beispiel

Eingabe:

7

Ausgabe:

## Testfaelle

- 2 -> 1
- 9 -> 0

## Ausfuehrung

Beispiel:

```
dotnet run --project src/Pl0.Cli -- run example.pl0
```

Tipp: Fuege **--list-code** hinzu, um den P-Code zu sehen.

# Anhang: Notenbewertung

Punkte -> Note (vereinfacht).

## Programm

```
var p, n;  
begin  
    ? p;  
    n := 6;  
    if p >= 90 then  
        n := 1;  
    if p >= 75 then  
        n := 2;  
    if p >= 60 then  
        n := 3;  
    if p >= 45 then  
        n := 4;  
    if p >= 30 then  
        n := 5;  
    ! n  
end.
```

## Erklaerung

- Eingaben werden mit **?** gelesen.
- Ausgaben erfolgen ueber **!**.
- Alle Berechnungen sind ganzzahlig.

## Details

- Einfache Staffelung in Noten 1-6.
- Grenzen sind didaktisch gewaehlt.

## Beispiel

Eingabe:

78

Ausgabe:

2

## Testfaelle

- 95 -> 1
- 30 -> 5

## Ausfuehrung

Beispiel:

```
dotnet run --project src/Pl0.Cli -- run example.pl0
```

Tipp: Fuege **--list-code** hinzu, um den P-Code zu sehen.

# Anhang: Rechteck Umfang und Flaeche

Berechnet Umfang und Flaeche eines Rechtecks.

## Programm

```
var a, b, u, f;  
begin  
    ? a;  
    ? b;  
    u := 2 * (a + b);  
    f := a * b;  
    ! u;  
    ! f  
end.
```

## Erklaerung

- Eingaben werden mit **?** gelesen.
- Ausgaben erfolgen ueber **!**.
- Alle Berechnungen sind ganzzahlig.

## Details

- Umfang =  $2*(a+b)$ .
- Flaeche =  $a*b$ .

## Beispiel

Eingabe:

**3 4**

Ausgabe:

**14 12**

## Testfaelle

- 2 3 -> 10 6
- 5 5 -> 20 25

## Ausfuehrung

Beispiel:

```
dotnet run --project src/Pl0.Cli -- run example.pl0
```

Tipp: Füge **--list-code** hinzu, um den P-Code zu sehen.

# Anhang: Versandkostenstaffel

Einfache Staffelung nach Bestellwert.

## Programm

```
var wert, kosten;  
begin  
    ? wert;  
    kosten := 0;  
    if wert < 50 then  
        kosten := 5;  
    if wert >= 50 then  
        kosten := 0;  
    ! kosten  
end.
```

## Erklaerung

- Eingaben werden mit **?** gelesen.
- Ausgaben erfolgen ueber **!**.
- Alle Berechnungen sind ganzzahlig.

## Details

- Zwei Stufen: <50 kostet 5, sonst 0.
- Beispiel fuer einfache Regeln.

## Beispiel

Eingabe:

40

Ausgabe:

5

## Testfaelle

- 50 -> 0
- 49 -> 5

# Ausfuehrung

Beispiel:

```
dotnet run --project src/Pl0.Cli -- run example.pl0
```

Tipp: Füge **--list-code** hinzu, um den P-Code zu sehen.

# Anhang: Betrag

Berechnet  $|x|$ .

## Programm

```
var x;  
begin  
    ? x;  
    if x < 0 then  
        x := -x;  
    ! x  
end.
```

## Erklaerung

- Eingaben werden mit `? x` gelesen.
- Ausgaben erfolgen ueber `! x`.
- Alle Berechnungen sind ganzzahlig.

## Details

- Bedingung prueft negatives Vorzeichen.
- Ergebnis ist immer  $\geq 0$ .

## Beispiel

Eingabe:

-9

Ausgabe:

9

## Testfaelle

- 0  $\rightarrow$  0
- -1  $\rightarrow$  1

## Ausfuehrung

Beispiel:

```
dotnet run --project src/Pl0.Cli -- run example.pl0
```

Tipp: Füge `--list-code` hinzu, um den P-Code zu sehen.

# Anhang: Rabattberechnung

Berechnet Preis nach Rabatt in Prozent.

## Programm

```
var preis, rabatt, neu;  
begin  
    ? preis;  
    ? rabatt;  
    neu := preis - (preis * rabatt) / 100;  
    ! neu  
end.
```

## Erklaerung

- Eingaben werden mit **?** gelesen.
- Ausgaben erfolgen ueber **!**.
- Alle Berechnungen sind ganzzahlig.

## Details

- Preis minus Prozentanteil.
- Ergebnis ist ganzzahlig.

## Beispiel

Eingabe:

**200 10**

Ausgabe:

**180**

## Testfaelle

- 100 0 -> 100
- 50 20 -> 40

## Ausfuehrung

Beispiel:

```
dotnet run --project src/Pl0.Cli -- run example.pl0
```

Tipp: Füge `--list-code` hinzu, um den P-Code zu sehen.

# Anhang: Multiplikation durch Addition

Multipliziert  $a \cdot b$  durch Wiederholung.

## Programm

```
var a, b, i, p;
begin
    ? a;
    ? b;
    p := 0;
    i := 0;
    while i < b do
        begin
            p := p + a;
            i := i + 1
        end;
    ! p
end.
```

## Erklaerung

- Eingaben werden mit **?** gelesen.
- Ausgaben erfolgen ueber **!**.
- Alle Berechnungen sind ganzzahlig.

## Details

- Wiederholtes Addieren.
- $b$  bestimmt die Anzahl der Schritte.

## Beispiel

Eingabe:

6 7

Ausgabe:

42

## Testfaelle

- 0 9 -> 0

- 4 3 -> 12

## Ausfuehrung

Beispiel:

```
dotnet run --project src/Pl0.Cli -- run example.pl0
```

Tipp: Füge `--list-code` hinzu, um den P-Code zu sehen.

# Anhang: Temperaturumrechnung

Celsius in Fahrenheit (Integer).

## Programm

```
var c, f;  
begin  
    ? c;  
    f := (c * 9) / 5 + 32;  
    ! f  
end.
```

## Erklaerung

- Eingaben werden mit `? c` gelesen.
- Ausgaben erfolgen ueber `! f`.
- Alle Berechnungen sind ganzzahlig.

## Details

- Formel  $F = C \cdot 9/5 + 32$ .
- Ganzzahlige Rundung durch Division.

## Beispiel

Eingabe:

0

Ausgabe:

32

## Testfaelle

- 100 -> 212
- -40 -> -40

## Ausfuehrung

Beispiel:

```
dotnet run --project src/Pl0.Cli -- run example.pl0
```

Tipp: Füge `--list-code` hinzu, um den P-Code zu sehen.

# Anhang: Countdown

Zaeht von n nach 0.

## Programm

```
var n;  
begin  
    ? n;  
    while n >= 0 do  
        begin  
            ! n;  
            n := n - 1  
        end  
    end.
```

## Erklaerung

- Eingaben werden mit **?**  gelesen.
- Ausgaben erfolgen ueber **!** .
- Alle Berechnungen sind ganzzahlig.

## Details

- Zaeht von n bis 0 inklusive.
- N sollte nicht negativ sein.

## Beispiel

Eingabe:

3

Ausgabe:

3 2 1 0

## Testfaelle

- 0 -> 0
- 2 -> 2 1 0

## Ausfuehrung

Beispiel:

```
dotnet run --project src/Pl0.Cli -- run example.pl0
```

Tipp: Füge `--list-code` hinzu, um den P-Code zu sehen.

# Anhang: Kreis Umfang und Flaeche

Berechnet Umfang und Flaeche mit Festkomma.

## Programm

```
const pi = 3141, scale = 1000;  
var r, u, f;  
begin  
    ? r;  
    u := (2 * pi * r) / scale;  
    f := (pi * r * r) / scale;  
    ! u;  
    ! f  
end.
```

## Erklaerung

- Eingaben werden mit `?` gelesen.
- Ausgaben erfolgen ueber `!`.
- Alle Berechnungen sind ganzzahlig.

## Details

- `pi` als Festkomma (3141/1000).
- Ergebnisse sind Naehlerungen.

## Beispiel

Eingabe:

10

Ausgabe:

62 314

## Testfaelle

- 1 -> 6 3
- 2 -> 12 12

## Ausfuehrung

Beispiel:

```
dotnet run --project src/Pl0.Cli -- run example.pl0
```

Tipp: Füge **--list-code** hinzu, um den P-Code zu sehen.

# Anhang: Pythagoras

Berechnet  $c^2 = a^2 + b^2$ .

## Programm

```
var a, b, c2;  
begin  
    ? a;  
    ? b;  
    c2 := a * a + b * b;  
    ! c2  
end.
```

## Erklaerung

- Eingaben werden mit **?** gelesen.
- Ausgaben erfolgen ueber **!**.
- Alle Berechnungen sind ganzzahlig.

## Details

- Berechnet  $c^2$ , nicht  $c$ .
- Fuer  $c$  waere eine Wurzel noetig.

## Beispiel

Eingabe:

3 4

Ausgabe:

25

## Testfaelle

- 5 12 -> 169
- 8 15 -> 289

## Ausfuehrung

Beispiel:

```
dotnet run --project src/Pl0.Cli -- run example.pl0
```

Tipp: Füge `--list-code` hinzu, um den P-Code zu sehen.

# Anhang: Kubikzahlen

Gibt  $n^3$  fuer 1..N aus.

## Programm

```
var n, i;
begin
    ? n;
    i := 1;
    while i <= n do
        begin
            ! (i * i * i);
            i := i + 1
        end
    end.
```

## Erklaerung

- Eingaben werden mit **?** gelesen.
- Ausgaben erfolgen ueber **!**.
- Alle Berechnungen sind ganzzahlig.

## Details

- Ausgabe von  $i^3$  fuer 1..N.
- Eignet sich fuer Tabellen.

## Beispiel

Eingabe:

3

Ausgabe:

1 8 27

## Testfaelle

- 1 -> 1
- 2 -> 1 8

# Ausfuehrung

Beispiel:

```
dotnet run --project src/Pl0.Cli -- run example.pl0
```

Tipp: Füge **--list-code** hinzu, um den P-Code zu sehen.

# Anhang: Modulo durch Subtraktion

Berechnet  $a \bmod b$ .

## Programm

```
var a, b;  
begin  
    ? a;  
    ? b;  
    while a >= b do  
        a := a - b;  
    ! a  
end.
```

## Erklaerung

- Eingaben werden mit `?` gelesen.
- Ausgaben erfolgen ueber `!`.
- Alle Berechnungen sind ganzzahlig.

## Details

- Wiederholtes Subtrahieren liefert den Rest.
- Ergebnis entspricht  $a \bmod b$ .

## Beispiel

Eingabe:

`17 5`

Ausgabe:

`2`

## Testfaelle

- $8 \ 3 \rightarrow 2$
- $7 \ 7 \rightarrow 0$

## Ausfuehrung

Beispiel:

```
dotnet run --project src/Pl0.Cli -- run example.pl0
```

Tipp: Füge `--list-code` hinzu, um den P-Code zu sehen.

# Anhang: Summe gerade/ungerade

Summiert gerade und ungerade Zahlen bis N.

## Programm

```
var n, i, sEven, sOdd;  
begin  
    ? n;  
    sEven := 0;  
    sOdd := 0;  
    i := 1;  
    while i <= n do  
        begin  
            if (i / 2) * 2 = i then  
                sEven := sEven + i;  
            if (i / 2) * 2 # i then  
                sOdd := sOdd + i;  
            i := i + 1  
        end;  
    ! sEven;  
    ! sOdd  
end.
```

## Erklaerung

- Eingaben werden mit **?** gelesen.
- Ausgaben erfolgen ueber **!**.
- Alle Berechnungen sind ganzzahlig.

## Details

- Gerade pruefung per  $(i/2)*2$ .
- Ausgabe: zuerst gerade, dann ungerade Summe.

## Beispiel

Eingabe:

7

Ausgabe:

12 16

## Testfaelle

- 1 -> 0 1
- 4 -> 6 4

## Ausfuehrung

Beispiel:

```
dotnet run --project src/Pl0.Cli -- run example.pl0
```

Tipp: Fuege **--list-code** hinzu, um den P-Code zu sehen.

# Anhang: Paritaetstest

Pruft gerade/ungerade.

## Programm

```
var x, even;  
begin  
    ? x;  
    even := 0;  
    if (x / 2) * 2 = x then  
        even := 1;  
    ! even  
end.
```

## Erklaerung

- Eingaben werden mit `?` gelesen.
- Ausgaben erfolgen ueber `!`.
- Alle Berechnungen sind ganzzahlig.

## Details

- Gerade/ungerade ueber Division.
- Ausgabe: 1 = gerade, 0 = ungerade.

## Beispiel

Eingabe:

8

Ausgabe:

1

## Testfaelle

- 3 -> 0
- 10 -> 1

## Ausfuehrung

Beispiel:

```
dotnet run --project src/Pl0.Cli -- run example.pl0
```

Tipp: Füge `--list-code` hinzu, um den P-Code zu sehen.

# Anhang: Primzahlen bis N

Gibt Primzahlen bis zu einer oberen Grenze aus.

## Programm

```
var n, i, j, isPrime;  
begin  
    ? n;  
    i := 2;  
    while i <= n do  
        begin  
            isPrime := 1;  
            j := 2;  
            while j * j <= i do  
                begin  
                    if (i / j) * j = i then  
                        isPrime := 0;  
                    j := j + 1  
                end;  
            if isPrime = 1 then  
                ! i;  
            i := i + 1  
        end  
    end.  
end.
```

## Erklaerung

- Eingaben werden mit **?** gelesen.
- Ausgaben erfolgen ueber **!**.
- Alle Berechnungen sind ganzzahlig.

## Details

- Schleife testet Teilbarkeit nur bis  $\sqrt{n}$ .
- Ergebnis: 1 = prim, 0 = nicht prim.

## Beispiel

Eingabe:

10

Ausgabe:

2 3 5 7

## Testfaelle

- 5 -> 2 3 5
- 1 -> (keine Ausgabe)

## Ausfuehrung

Beispiel:

```
dotnet run --project src/Pl0.Cli -- run example.pl0
```

Tipp: Fuege **--list-code** hinzu, um den P-Code zu sehen.

# Anhang: Summenquadrat vs. Quadratsumme

Vergleicht  $(\text{sum})^2$  und  $\text{sum}(\text{x}^2)$ .

## Programm

```
var n, i, sum, sumsq;  
begin  
    ? n;  
    sum := 0;  
    sumsq := 0;  
    i := 1;  
    while i <= n do  
        begin  
            sum := sum + i;  
            sumsq := sumsq + i * i;  
            i := i + 1  
        end;  
    ! (sum * sum);  
    ! sumsq  
end.
```

## Erklaerung

- Eingaben werden mit **?** gelesen.
- Ausgaben erfolgen ueber **!**.
- Alle Berechnungen sind ganzzahlig.

## Details

- Umfang =  $4*a$ .
- Flaeche =  $a*a$ .

## Beispiel

Eingabe:

3

Ausgabe:

36 14

## Testfaelle

- 1 -> 11
- 2 -> 95

## Ausfuehrung

Beispiel:

```
dotnet run --project src/Pl0.Cli -- run example.pl0
```

Tipp: Füge **--list-code** hinzu, um den P-Code zu sehen.

# Anhang: Altersberechnung

Berechnet Alter aus Geburtsjahr und aktuellem Jahr.

## Programm

```
var geb, jahr, alter;  
begin  
    ? geb;  
    ? jahr;  
    alter := jahr - geb;  
    if alter < 0 then  
        alter := 0;  
    ! alter  
end.
```

## Erklaerung

- Eingaben werden mit **?** gelesen.
- Ausgaben erfolgen ueber **!**.
- Alle Berechnungen sind ganzzahlig.

## Details

- Alter = aktuelles Jahr - Geburtsjahr.
- Negative Werte werden auf 0 gesetzt.

## Beispiel

Eingabe:

**2000 2025**

Ausgabe:

25

## Testfaelle

- 2025 2025 -> 0
- 2010 2025 -> 15

## Ausfuehrung

Beispiel:

```
dotnet run --project src/Pl0.Cli -- run example.pl0
```

Tipp: Füge `--list-code` hinzu, um den P-Code zu sehen.

# Anhang: Potenz

Berechnet  $a^b$  durch Wiederholung.

## Programm

```
var a, b, i, p;
begin
    ? a;
    ? b;
    p := 1;
    i := 0;
    while i < b do
        begin
            p := p * a;
            i := i + 1
        end;
    ! p
end.
```

## Erklaerung

- Eingaben werden mit **?** gelesen.
- Ausgaben erfolgen ueber **!**.
- Alle Berechnungen sind ganzzahlig.

## Details

- Exponent wird durch wiederholte Multiplikation aufgebaut.
- b sollte nicht negativ sein.

## Beispiel

Eingabe:

**2 8**

Ausgabe:

256

## Testfaelle

- 3 0 -> 1

- 2 5 -> 32

## Ausfuehrung

Beispiel:

```
dotnet run --project src/Pl0.Cli -- run example.pl0
```

Tipp: Füge `--list-code` hinzu, um den P-Code zu sehen.

# Anhang: Einheitenumrechnung

cm -> m -> km.

## Programm

```
var cm, m, km;  
begin  
    ? cm;  
    m := cm / 100;  
    km := m / 1000;  
    ! m;  
    ! km  
end.
```

## Erklaerung

- Eingaben werden mit **?** gelesen.
- Ausgaben erfolgen ueber **!**.
- Alle Berechnungen sind ganzzahlig.

## Details

- cm -> m -> km, rein ganzzahlig.
- Rundungen durch Division.

## Beispiel

Eingabe:

12345

Ausgabe:

123 0

## Testfaelle

- 100 -> 1 0
- 100000 -> 1000 1

## Ausfuehrung

Beispiel:

```
dotnet run --project src/Pl0.Cli -- run example.pl0
```

Tipp: Füge `--list-code` hinzu, um den P-Code zu sehen.

# Anhang: Quadratzahlen

Gibt  $n^2$  fuer 1..N aus.

## Programm

```
var n, i;
begin
    ? n;
    i := 1;
    while i <= n do
        begin
            ! (i * i);
            i := i + 1
        end
    end.
end.
```

## Erklaerung

- Eingaben werden mit **?** gelesen.
- Ausgaben erfolgen ueber **!**.
- Alle Berechnungen sind ganzzahlig.

## Details

- Umfang =  $4 \cdot a$ .
- Flaeche =  $a \cdot a$ .

## Beispiel

Eingabe:

4

Ausgabe:

**1 4 9 16**

## Testfaelle

- 1 -> 1
- 2 -> 1 4

# Ausfuehrung

Beispiel:

```
dotnet run --project src/Pl0.Cli -- run example.pl0
```

Tipp: Füge **--list-code** hinzu, um den P-Code zu sehen.

# Anhang: Zinseszins

Berechnet Endkapital nach n Jahren.

## Programm

```
var k, p, n, i;
begin
    ? k;
    ? p;
    ? n;
    i := 0;
    while i < n do
        begin
            k := k + (k * p) / 100;
            i := i + 1
        end;
    ! k
end.
```

## Erklaerung

- Eingaben werden mit **?** gelesen.
- Ausgaben erfolgen ueber **!**.
- Alle Berechnungen sind ganzzahlig.

## Details

- Iterative Verzinsung pro Jahr.
- p als Prozentwert verwenden.

## Beispiel

Eingabe:

**1000 5 2**

Ausgabe:

**1102**

## Testfaelle

- 1000 10 1 -> 1100

- 1000 10 2 -> 1210

## Ausfuehrung

Beispiel:

```
dotnet run --project src/Pl0.Cli -- run example.pl0
```

Tipp: Füge **--list-code** hinzu, um den P-Code zu sehen.

# Anhang: Zahlenvergleich

Vergleicht a und b.

## Programm

```
var a, b, r;
begin
    ? a;
    ? b;
    r := 0;
    if a < b then
        r := -1;
    if a = b then
        r := 0;
    if a > b then
        r := 1;
    ! r
end.
```

## Erklaerung

- Eingaben werden mit **?** gelesen.
- Ausgaben erfolgen ueber **!**.
- Alle Berechnungen sind ganzzahlig.

## Details

- Ergebnis: -1 ( $a < b$ ), 0 ( $a = b$ ), 1 ( $a > b$ ).
- Drei getrennte if-Zweige.

## Beispiel

Eingabe:

5 9

Ausgabe:

-1

## Testfaelle

- 5 5 -> 0

- 9 4 -> 1

## Ausfuehrung

Beispiel:

```
dotnet run --project src/Pl0.Cli -- run example.pl0
```

Tipp: Füge `--list-code` hinzu, um den P-Code zu sehen.

# Anhang: Minimum und Maximum

Bestimmt Minimum und Maximum aus N Zahlen.

## Programm

```
var n, i, x, min, max;  
begin  
    ? n;  
    ? x;  
    min := x;  
    max := x;  
    i := 1;  
    while i < n do  
        begin  
            ? x;  
            if x < min then  
                min := x;  
            if x > max then  
                max := x;  
            i := i + 1  
        end;  
    ! min;  
    ! max  
end.
```

## Erklaerung

- Eingaben werden mit **?** gelesen.
- Ausgaben erfolgen ueber **!**.
- Alle Berechnungen sind ganzzahlig.

## Details

- Erstes Element initialisiert min/max.
- Danach Vergleich in der Schleife.

## Beispiel

Eingabe:

5 3 9 2 8 6

Ausgabe:

## Testfaelle

- 3 5 5 5 -> 5 5
- 4 -1 2 0 9 -> -1 9

## Ausfuehrung

Beispiel:

```
dotnet run --project src/Pl0.Cli -- run example.pl0
```

Tipp: Fuege **--list-code** hinzu, um den P-Code zu sehen.

# Anhang: Mittelwert aus N Zahlen

Liest N Zahlen und berechnet den Mittelwert (Integer).

## Programm

```
var n, i, sum, x;
begin
    ? n;
    sum := 0;
    i := 0;
    while i < n do
        begin
            ? x;
            sum := sum + x;
            i := i + 1
        end;
    ! (sum / n)
end.
```

## Erklaerung

- Eingaben werden mit `?` gelesen.
- Ausgaben erfolgen ueber `!`.
- Alle Berechnungen sind ganzzahlig.

## Details

- Ergebnis ist ganzzahlig gekuerzt.
- `n` muss groesser als 0 sein.

## Beispiel

Eingabe:

`4 10 20 30 40`

Ausgabe:

`25`

## Testfaelle

- `1 5 -> 5`

- 3 1 2 2 -> 1

## Ausfuehrung

Beispiel:

```
dotnet run --project src/Pl0.Cli -- run example.pl0
```

Tipp: Füge `--list-code` hinzu, um den P-Code zu sehen.

# Anhang: Ganzzahl-Division mit Rest

Berechnet Quotient und Rest.

## Programm

```
var a, b, q, r;  
begin  
    ? a;  
    ? b;  
    q := a / b;  
    r := a - q * b;  
    ! q;  
    ! r  
end.
```

## Erklaerung

- Eingaben werden mit `?` gelesen.
- Ausgaben erfolgen ueber `!`.
- Alle Berechnungen sind ganzzahlig.

## Details

- Rest wird als  $a - q \cdot b$  berechnet.
- $b$  darf nicht 0 sein.

## Beispiel

Eingabe:

`17 5`

Ausgabe:

`3 2`

## Testfaelle

- $20 \text{ } 4 \rightarrow 5 \text{ } 0$
- $9 \text{ } 2 \rightarrow 4 \text{ } 1$

## Ausfuehrung

Beispiel:

```
dotnet run --project src/Pl0.Cli -- run example.pl0
```

Tipp: Füge **--list-code** hinzu, um den P-Code zu sehen.

# Anhang: Dreiecksumfang

Berechnet Umfang eines Dreiecks.

## Programm

```
var a, b, c, u;  
begin  
    ? a;  
    ? b;  
    ? c;  
    u := a + b + c;  
    ! u  
end.
```

## Erklaerung

- Eingaben werden mit `?` gelesen.
- Ausgaben erfolgen ueber `!`.
- Alle Berechnungen sind ganzzahlig.

## Details

- Summe aller drei Seiten.
- Keine Plausibilitaetspruefung.

## Beispiel

Eingabe:

3 4 5

Ausgabe:

12

## Testfaelle

- 1 1 1 -> 3
- 5 7 9 -> 21

## Ausfuehrung

Beispiel:

```
dotnet run --project src/Pl0.Cli -- run example.pl0
```

Tipp: Füge `--list-code` hinzu, um den P-Code zu sehen.

# Anhang: Abstand zweier Zahlen

Berechnet  $|a-b|$ .

## Programm

```
var a, b, d;  
begin  
    ? a;  
    ? b;  
    d := a - b;  
    if d < 0 then  
        d := -d;  
    ! d  
end.
```

## Erklaerung

- Eingaben werden mit `?` gelesen.
- Ausgaben erfolgen ueber `!`.
- Alle Berechnungen sind ganzzahlig.

## Details

- Betrag der Differenz.
- Nutzt Bedingung fuer negatives Ergebnis.

## Beispiel

Eingabe:

5 12

Ausgabe:

7

## Testfaelle

- 10 3 -> 7
- -2 4 -> 6

## Ausfuehrung

Beispiel:

```
dotnet run --project src/Pl0.Cli -- run example.pl0
```

Tipp: Füge `--list-code` hinzu, um den P-Code zu sehen.

# Anhang: Division durch Subtraktion

Teilt a/b durch wiederholtes Subtrahieren.

## Programm

```
var a, b, q;  
begin  
    ? a;  
    ? b;  
    q := 0;  
    while a >= b do  
        begin  
            a := a - b;  
            q := q + 1  
        end;  
        ! q  
    end.
```

## Erklaerung

- Eingaben werden mit **?** gelesen.
- Ausgaben erfolgen ueber **!**.
- Alle Berechnungen sind ganzzahlig.

## Details

- Wiederholtes Subtrahieren bis  $a < b$ .
- Ergebnis ist Quotient ohne Rest.

## Beispiel

Eingabe:

17 5

Ausgabe:

3

## Testfaelle

- 9 3 -> 3
- 8 3 -> 2

# Ausfuehrung

Beispiel:

```
dotnet run --project src/Pl0.Cli -- run example.pl0
```

Tipp: Füge **--list-code** hinzu, um den P-Code zu sehen.

# Anhang: Uhrzeitdifferenz

Differenz zweier Zeiten in Minuten.

## Programm

```
var h1, m1, h2, m2, t1, t2, d;  
begin  
    ? h1;  
    ? m1;  
    ? h2;  
    ? m2;  
    t1 := h1 * 60 + m1;  
    t2 := h2 * 60 + m2;  
    d := t2 - t1;  
    if d < 0 then  
        d := -d;  
    ! d  
end.
```

## Erklaerung

- Eingaben werden mit **?** gelesen.
- Ausgaben erfolgen ueber **!**.
- Alle Berechnungen sind ganzzahlig.

## Details

- Berechnet Differenz in Minuten.
- Ergebnis ist absoluter Wert.

## Beispiel

Eingabe:

9 30 11 0

Ausgabe:

90

## Testfaelle

- 0 0 0 0 -> 0

- 10 0 11 30 -> 90

## Ausfuehrung

Beispiel:

```
dotnet run --project src/Pl0.Cli -- run example.pl0
```

Tipp: Füge **--list-code** hinzu, um den P-Code zu sehen.

# Anhang: Zifferanzahl

Zaeht die Ziffern einer Zahl.

## Programm

```
var n, c;  
begin  
    ? n;  
    if n = 0 then  
        c := 1;  
    if n # 0 then  
        begin  
            c := 0;  
            while n > 0 do  
                begin  
                    n := n / 10;  
                    c := c + 1  
                end  
            end;  
            ! c  
        end.  
    end.
```

## Erklaerung

- Eingaben werden mit `?`  gelesen.
- Ausgaben erfolgen ueber `!` .
- Alle Berechnungen sind ganzzahlig.

## Details

- Sonderfall  $n=0$  ergibt 1 Ziffer.
- Schleife zaeht durch Division.

## Beispiel

Eingabe:

1005

Ausgabe:

## Testfaelle

- 0 -> 1
- 42 -> 2

## Ausfuehrung

Beispiel:

```
dotnet run --project src/Pl0.Cli -- run example.pl0
```

Tipp: Fuege **--list-code** hinzu, um den P-Code zu sehen.

# Anhang: Lineare Gleichung

Loest  $ax + b = 0$  (ganzzahlig).

## Programm

```
var a, b, x;
begin
    ? a;
    ? b;
    if a = 0 then
        ! 0
    else
        begin
            x := -b / a;
            ! x
        end
    end.
end.
```

## Erklaerung

- Eingaben werden mit `? gelesen.`
- Ausgaben erfolgen ueber `!.`
- Alle Berechnungen sind ganzzahlig.

## Details

- Bei  $a=0$  wird 0 ausgegeben (keine/inf. Loesungen).
- $x$  wird ganzzahlig berechnet.

## Beispiel

Eingabe:

`2 4`

Ausgabe:

`-2`

## Testfaelle

- 0 5 -> 0
- 2 -6 -> 3

# Ausfuehrung

Beispiel:

```
dotnet run --project src/Pl0.Cli -- run example.pl0
```

Tipp: Füge **--list-code** hinzu, um den P-Code zu sehen.

# Anhang: Schleifenzaehler

Einfacher Zaehler fuer Laufzeit.

## Programm

```
var i, n;  
begin  
    ? n;  
    i := 0;  
    while i < n do  
        i := i + 1;  
    ! i  
end.
```

## Erklaerung

- Eingaben werden mit `?`  gelesen.
- Ausgaben erfolgen ueber `!` .
- Alle Berechnungen sind ganzzahlig.

## Details

- Dient zur Demonstration von Laufzeit/Schleifen.
- Ausgabe ist der Endzaehler.

## Beispiel

Eingabe:

5

Ausgabe:

5

## Testfaelle

- 0 -> 0
- 3 -> 3

## Ausfuehrung

Beispiel:

```
dotnet run --project src/Pl0.Cli -- run example.pl0
```

Tipp: Füge `--list-code` hinzu, um den P-Code zu sehen.

# Anhang: Geschwindigkeit

Berechnet Strecke  $s = v * t$ .

## Programm

```
var v, t, s;  
begin  
    ? v;  
    ? t;  
    s := v * t;  
    ! s  
end.
```

## Erklaerung

- Eingaben werden mit **?**  gelesen.
- Ausgaben erfolgen ueber **!**.
- Alle Berechnungen sind ganzzahlig.

## Details

- Strecke = Geschwindigkeit \* Zeit.
- Einheit konsistent halten.

## Beispiel

Eingabe:

**50 3**

Ausgabe:

**150**

## Testfaelle

- 0 10 -> 0
- 80 2 -> 160

## Ausfuehrung

Beispiel:

```
dotnet run --project src/Pl0.Cli -- run example.pl0
```

Tipp: Füge `--list-code` hinzu, um den P-Code zu sehen.

# Anhang: Fakultaet

Berechnet n! iterativ.

## Programm

```
var n, i, f;  
begin  
    ? n;  
    f := 1;  
    i := 2;  
    while i <= n do  
        begin  
            f := f * i;  
            i := i + 1  
        end;  
    ! f  
end.
```

## Erklaerung

- Eingaben werden mit **?** gelesen.
- Ausgaben erfolgen ueber **!**.
- Alle Berechnungen sind ganzzahlig.

## Details

- Laufvariable i startet bei 2, Ergebnis in f.
- Fuer n=0 bleibt f=1.

## Beispiel

Eingabe:

5

Ausgabe:

120

## Testfaelle

- 0 -> 1
- 6 -> 720

# Ausfuehrung

Beispiel:

```
dotnet run --project src/Pl0.Cli -- run example.pl0
```

Tipp: Füge **--list-code** hinzu, um den P-Code zu sehen.

# Anhang: Quadrat Umfang und Flaeche

Berechnet Umfang und Flaeche eines Quadrats.

## Programm

```
var a, u, f;  
begin  
    ? a;  
    u := 4 * a;  
    f := a * a;  
    ! u;  
    ! f  
end.
```

## Erklaerung

- Eingaben werden mit `? a` gelesen.
- Ausgaben erfolgen ueber `! u`.
- Alle Berechnungen sind ganzzahlig.

## Details

- Umfang =  $4 \cdot a$ .
- Flaeche =  $a \cdot a$ .

## Beispiel

Eingabe:

5

Ausgabe:

20 25

## Testfaelle

- 1 -> 4 1
- 3 -> 12 9

## Ausfuehrung

Beispiel:

```
dotnet run --project src/Pl0.Cli -- run example.pl0
```

Tipp: Füge `--list-code` hinzu, um den P-Code zu sehen.

# Anhang: Arithmetische Folge

Gibt eine arithmetische Folge aus.

## Programm

```
var a, d, n, i, x;
begin
    ? a;
    ? d;
    ? n;
    i := 0;
    x := a;
    while i < n do
        begin
            ! x;
            x := x + d;
            i := i + 1
        end
    end.
```

## Erklaerung

- Eingaben werden mit `?` gelesen.
- Ausgaben erfolgen ueber `!`.
- Alle Berechnungen sind ganzzahlig.

## Details

- Startwert `a` und Schritt `d`.
- Gibt `n` Werte aus.

## Beispiel

Eingabe:

`2 3 4`

Ausgabe:

`2 5 8 11`

## Testfaelle

- 1 1 3 -> 1 2 3
- 5 2 2 -> 5 7

## Ausfuehrung

Beispiel:

```
dotnet run --project src/Pl0.Cli -- run example.pl0
```

Tipp: Füge **--list-code** hinzu, um den P-Code zu sehen.

# Anhang: Tage zu Wochen

Berechnet Wochen und Resttage.

## Programm

```
var t, w, r;
begin
    ? t;
    w := t / 7;
    r := t - w * 7;
    ! w;
    ! r
end.
```

## Erklaerung

- Eingaben werden mit `?` gelesen.
- Ausgaben erfolgen ueber `!`.
- Alle Berechnungen sind ganzzahlig.

## Details

- Wochen =  $t/7$ , Resttage =  $t - \text{Wochen} \cdot 7$ .
- Beispiele mit kleinen Zahlen.

## Beispiel

Eingabe:

17

Ausgabe:

2 3

## Testfaelle

- 7 -> 1 0
- 0 -> 0 0

## Ausfuehrung

Beispiel:

```
dotnet run --project src/Pl0.Cli -- run example.pl0
```

Tipp: Füge `--list-code` hinzu, um den P-Code zu sehen.

# Anhang: Einfache Statistik

Summe, Mittel, Min, Max.

## Programm

```
var n, i, x, sum, min, max;
begin
    ? n;
    ? x;
    sum := x;
    min := x;
    max := x;
    i := 1;
    while i < n do
        begin
            ? x;
            sum := sum + x;
            if x < min then
                min := x;
            if x > max then
                max := x;
            i := i + 1
        end;
        ! sum;
        ! (sum / n);
        ! min;
        ! max
    end.
```

## Erklaerung

- Eingaben werden mit `?` gelesen.
- Ausgaben erfolgen ueber `!`.
- Alle Berechnungen sind ganzzahlig.

## Details

- Summe, Mittelwert, Min, Max.
- Mittelwert ist ganzzahlig.

## Beispiel

Eingabe:

4 2 8 4 6

Ausgabe:

20 5 2 8

## Testfaelle

- 1 5 -> 5 5 5 5
- 3 1 2 3 -> 6 2 1 3

## Ausfuehrung

Beispiel:

```
dotnet run --project src/Pl0.Cli -- run example.pl0
```

Tipp: Fuege **--list-code** hinzu, um den P-Code zu sehen.

# Anhang: Summe 1 bis N

Berechnet die Summe 1..N.

## Programm

```
var n, i, sum;  
begin  
    ? n;  
    sum := 0;  
    i := 1;  
    while i <= n do  
        begin  
            sum := sum + i;  
            i := i + 1  
        end;  
    ! sum  
end.
```

## Erklaerung

- Eingaben werden mit `?`  gelesen.
- Ausgaben erfolgen ueber `!` .
- Alle Berechnungen sind ganzzahlig.

## Details

- Summation per Schleife; alternative Formel  $(n*(n+1)/2)$  moeglich.
- $n$  sollte nicht negativ sein.

## Beispiel

Eingabe:

10

Ausgabe:

55

## Testfaelle

- 1 -> 1
- 5 -> 15

# Ausfuehrung

Beispiel:

```
dotnet run --project src/Pl0.Cli -- run example.pl0
```

Tipp: Füge `--list-code` hinzu, um den P-Code zu sehen.

# Anhang: Quadratische Gleichung

Diskriminante und ganzzahlige Loesung (vereinfachtes Beispiel).

## Programm

```
var a, b, c, d, x;
begin
    ? a;
    ? b;
    ? c;
    d := b * b - 4 * a * c;
    if d < 0 then
        ! 0
    else
        begin
            x := (-b + d) / (2 * a);
            ! x
        end
end.
```

## Erklaerung

- Eingaben werden mit `?` gelesen.
- Ausgaben erfolgen ueber `!`.
- Alle Berechnungen sind ganzzahlig.

## Details

- Diskriminante  $d$  entscheidet ueber Existenz der Loesung.
- Wurzel wird hier vereinfacht als  $d$  behandelt (didaktisch).

## Beispiel

Eingabe:

`1 -3 2`

Ausgabe:

`4`

## Testfaelle

- 1 0 -4 -> 4
- 1 0 4 -> 0

## Ausfuehrung

Beispiel:

```
dotnet run --project src/Pl0.Cli -- run example.pl0
```

Tipp: Füge **--list-code** hinzu, um den P-Code zu sehen.

# Anhang: Palindromtest

Prueft, ob eine Zahl ein Palindrom ist.

## Programm

```
var n, t, r, d;
begin
    ? n;
    t := n;
    r := 0;
    while t > 0 do
        begin
            d := t - (t / 10) * 10;
            r := r * 10 + d;
            t := t / 10
        end;
        if r = n then
            ! 1
        else
            ! 0
    end.
```

## Erklaerung

- Eingaben werden mit **?** gelesen.
- Ausgaben erfolgen ueber **!**.
- Alle Berechnungen sind ganzzahlig.

## Details

- Dreht Zahl um und vergleicht.
- Ergebnis: 1 = Palindrom, 0 = nein.

## Beispiel

Eingabe:

1221

Ausgabe:

1

## Testfaelle

- 123 -> 0
- 11 -> 1

## Ausfuehrung

Beispiel:

```
dotnet run --project src/Pl0.Cli -- run example.pl0
```

Tipp: Fuege **--list-code** hinzu, um den P-Code zu sehen.

# Anhang: Zeitumrechnung

Sekunden zu hh:mm:ss (ganzzahlig).

## Programm

```
var s, h, m;  
begin  
    ? s;  
    h := s / 3600;  
    s := s - h * 3600;  
    m := s / 60;  
    s := s - m * 60;  
    ! h;  
    ! m;  
    ! s  
end.
```

## Erklaerung

- Eingaben werden mit **?** gelesen.
- Ausgaben erfolgen ueber **!**.
- Alle Berechnungen sind ganzzahlig.

## Details

- Sekunden in h/m/s zerlegt.
- Ergebnisse einzeln ausgegeben.

## Beispiel

Eingabe:

3661

Ausgabe:

1 1 1

## Testfaelle

- 60 -> 0 1 0
- 59 -> 0 0 59

# Ausfuehrung

Beispiel:

```
dotnet run --project src/Pl0.Cli -- run example.pl0
```

Tipp: Füge `--list-code` hinzu, um den P-Code zu sehen.

# Anhang: Werte tauschen

Tauscht zwei Werte.

## Programm

```
var a, b, t;  
begin  
    ? a;  
    ? b;  
    t := a;  
    a := b;  
    b := t;  
    ! a;  
    ! b  
end.
```

## Erklaerung

- Eingaben werden mit `? a;` gelesen.
- Ausgaben erfolgen ueber `! a;`.
- Alle Berechnungen sind ganzzahlig.

## Details

- Tauscht Werte mit Hilfsvariable.
- Zwei Ausgaben zeigen Ergebnis.

## Beispiel

Eingabe:

`7 9`

Ausgabe:

`9 7`

## Testfaelle

- `1 2 -> 2 1`
- `-3 7 -> 7 -3`

# Ausfuehrung

Beispiel:

```
dotnet run --project src/Pl0.Cli -- run example.pl0
```

Tipp: Füge `--list-code` hinzu, um den P-Code zu sehen.

# Anhang: Ziffernsumme

Berechnet die Ziffernsumme einer Zahl.

## Programm

```
var n, sum, digit;  
begin  
    ? n;  
    sum := 0;  
    while n > 0 do  
        begin  
            digit := n - (n / 10) * 10;  
            sum := sum + digit;  
            n := n / 10  
        end;  
    ! sum  
end.
```

## Erklaerung

- Eingaben werden mit `? gelesen.`
- Ausgaben erfolgen ueber `!.`
- Alle Berechnungen sind ganzzahlig.

## Details

- Mod 10 per  $n - (n/10)*10$ .
- Schleife endet bei  $n=0$ .

## Beispiel

Eingabe:

472

Ausgabe:

13

## Testfaelle

- 0 -> 0
- 999 -> 27

# Ausfuehrung

Beispiel:

```
dotnet run --project src/Pl0.Cli -- run example.pl0
```

Tipp: Füge **--list-code** hinzu, um den P-Code zu sehen.

# Anhang: Prozentrechnung

Berechnet Prozentwert p% von grundwert g.

## Programm

```
var g, p, w;  
begin  
    ? g;  
    ? p;  
    w := (g * p) / 100;  
    ! w  
end.
```

## Erklaerung

- Eingaben werden mit **?**  gelesen.
- Ausgaben erfolgen ueber **!** .
- Alle Berechnungen sind ganzzahlig.

## Details

- Prozentwert = Grundwert \* Prozent / 100.
- Ergebnis ist ganzzahlig.

## Beispiel

Eingabe:

**200 15**

Ausgabe:

**30**

## Testfaelle

- 100 50 -> 50
- 80 25 -> 20

## Ausfuehrung

Beispiel:

```
dotnet run --project src/Pl0.Cli -- run example.pl0
```

Tipp: Füge `--list-code` hinzu, um den P-Code zu sehen.

# Anhang: Umkehrung einer Zahl

Kehrt die Ziffern einer Zahl um.

## Programm

```
var n, r, d;
begin
    ? n;
    r := 0;
    while n > 0 do
        begin
            d := n - (n / 10) * 10;
            r := r * 10 + d;
            n := n / 10
        end;
    ! r
end.
```

## Erklaerung

- Eingaben werden mit **?** gelesen.
- Ausgaben erfolgen ueber **!**.
- Alle Berechnungen sind ganzzahlig.

## Details

- Reversiert Ziffern durch Mod/Division.
- Ergebnis ist die umgekehrte Zahl.

## Beispiel

Eingabe:

12340

Ausgabe:

4321

## Testfaelle

- 100 -> 1
- 907 -> 709

# Ausfuehrung

Beispiel:

```
dotnet run --project src/Pl0.Cli -- run example.pl0
```

Tipp: Füge **--list-code** hinzu, um den P-Code zu sehen.