

Praktikum Objektorientierte Programmierung in C++ (WS 2023/2024)

[Dashboard](#) / [Meine Kurse](#) / [Wintersemester 2023/2024](#) / [Ingenieurwissenschaften](#) / [Informatik und Angewandte Kognitionswissenschaften](#) / [Praktikum OOP in C++ WS 2023/2024](#) / [Aufgabe 3/Task 3](#) / [A3 Teil 2: Präsenzaufgabe/Part 2: Presence Task](#)

A3 Teil 2: Präsenzaufgabe/Part 2: Presence Task

Erweitern Sie Ihren C++-Code aus Teil 1 bis zum Ende dieser Gruppenstunde folgendermassen:/

Extend your C++ code from part 1 until the end of this group hour as follows:

- Erweitern Sie den Ausgabeoperator für Daten vom Strukturtyp **year** um die Ausgabe der Zeichenkette **unit:** und der Einheit dahinter zusammen in runden Klammern./
Extend the output operator for data of the structure type **year** to include the output of string **unit:** and the unit behind it together in round brackets.
- Programmieren Sie eine **inline** Funktion namens **ones** (Einsen), die eine Referenzvariable vom Strukturtyp **year** als Operanden hat und eine Referenz vom Strukturtyp **year** zurück gibt.
Setzen Sie im Rumpf alle Viertelstunden-Intervallwerte auf **1.0** und geben danach die Referenzvariable als Funktionswert zurück./
Program an **inline** function called **ones**, which has a reference variable of structure type **year** as operand and returns a reference of structure type **year**.
In the body, set all quarter-hour interval values to **1.0** and return the reference variable as function value.
- Programmieren Sie einen überladenen binären Subtraktions-Operator **-** für zwei Referenzvariable vom Strukturtyp **year** als Operanden, der eine Variable vom Strukturtyp **year** zurück gibt.
Überprüfen Sie zuerst im Rumpf, dass die Jahreszahlen, der erste Wochentag im Jahr und die Einheiten der Operanden übereinstimmen. Initialisieren Sie danach eine neue Variable vom Strukturtyp **year**, subtrahieren für diese vom ersten Operanden elementweise alle Viertelstundenwerte gegeben im zweiten Operanden und geben die Ergebnis-Variable zurück./
Program an overloaded binary subtraction operator **-** for two reference variables of structure type **year** as operands, which returns a variable of structure type **year**.
In the body, first check that the number of the year, the first day of the week in the year and the units of the operands match. Then initialise a new variable of structure type **year**, subtract all quarter-hour values given in the second operand from the first operand element by element and return the result variable.
- Programmieren Sie einen überladenen binären Multiplikations-Operator ***** für eine Gleitpunktzahl und eine Referenzvariable vom Strukturtyp **year** als Operanden, der eine Variable vom Strukturtyp **year** zurück gibt.
Initialisieren Sie im Rumpf eine neue Variable vom Strukturtyp **year**, multiplizieren Sie elementweise alle Viertelstundenwerte gegeben im zweiten Operanden mit der Gleitpunktzahl im ersten Operanden und geben diese Variable zurück./
Program an overloaded binary multiplication operator ***** for a floating point number and a reference variable of structure type **year** as operands, which returns a variable of structure type **year**.
Initialise a new variable of structure type **year** in the body, multiply all quarter-hour values given in the second operand element by element with the floating point number in the first operand and return this variable.
- Definieren Sie eine Funktion namens **set_unit** mit einer Referenz vom Strukturtyp **year** als ersten und einer C++-Zeichenkette als zweitem Parameter ohne Rückgabe.
Setzen Sie im Rumpf die Komponente mit der Einheit der Werte (**unit**) in der Strukturvariable auf den Wert der Zeichenkette im zweiten Parameter./
Define a function called **set_unit** with a reference of the structure type **year** as the first parameter and a C++ character string as the second parameter without return.
In the body, set the component **unit** of the structure variable with the unit of the values to the value of the character string in the second parameter.
- Erweitern Sie in Ihrer Funktion **main** bei den Menüpunkten und deren Funktionalität (Beispiele siehe unten):/
Extend in your function **main** in the menu items and their functionality (see examples below):
 - m subtract actual from total (using operator -)**
hier ist nur **total = total - actual;** auszuführen./
here only **total = total - actual;** is to be executed.
 - s scalar multiplication**
lesen Sie einen Skalar ein und multiplizieren je nach Auswahl **actual** oder **total** mit diesem Wert über Ihren oben definierten überladenen Multiplikationsoperator./

read in a scalar and, depending on a selection, multiply **actual** or **total** by this value using your overloaded multiplication operator defined above.

- **c change unit**

lesen Sie eine Einheit als C++-Zeichenkette ein und ändern je nach Auswahl für **actual** oder **total** den Wert für Einheit über die oben definierte Funktion./

read in a unit as a C++ character string and, depending on the selection for **actual** or **total**, change the value for unit using the function defined above.

- **y set actual to ones (call function ones)**

rufen Sie Ihre Funktion **ones** für **actual** auf./

call your function **ones** for **actual**.

Laden Sie Ihren abgenommenen Programmcode in Moodle hoch./

[Upload your accepted program code in Moodle](#)

Beispiel Programmlauf/Example Program Run

YEARLY CONSUMPTION QUARTER HOUR

```

q quit
a add actual to total (using operator +)
m subtract actual from total (using operator -)
s scalar multiplication
c change unit
v sum up values
o output actual (using operator <<)
t output total (using operator <<)
u add consumption according to frequency of use (call functions add_consumption)
y set actual to ones (call function ones)
z set actual to zeros (call function zeros)
>> y
q quit
a add actual to total (using operator +)
m subtract actual from total (using operator -)
s scalar multiplication
c change unit
v sum up values
o output actual (using operator <<)
t output total (using operator <<)
u add consumption according to frequency of use (call functions add_consumption)
y set actual to ones (call function ones)
z set actual to zeros (call function zeros)
>> o
year: 2024 (unit: Watt)
day 0: Monday
  0:00      1.00      1.00      1.00      1.00
  1:00      1.00      1.00      1.00      1.00
  2:00      1.00      1.00      1.00      1.00

day 1: Tuesday
  0:00      1.00      1.00      1.00      1.00
  1:00      1.00      1.00      1.00      1.00
  2:00      1.00      1.00      1.00      1.00

day 2: Wednesday
  0:00      1.00      1.00      1.00      1.00
  1:00      1.00      1.00      1.00      1.00
  2:00      1.00      1.00      1.00      1.00

day 3: Thursday
  0:00      1.00      1.00      1.00      1.00
  1:00      1.00      1.00      1.00      1.00
  2:00      1.00      1.00      1.00      1.00

day 4: Friday
  0:00      1.00      1.00      1.00      1.00
  1:00      1.00      1.00      1.00      1.00
  2:00      1.00      1.00      1.00      1.00

day 5: Saturday
  0:00      1.00      1.00      1.00      1.00
  1:00      1.00      1.00      1.00      1.00
  2:00      1.00      1.00      1.00      1.00

day 6: Sunday
  0:00      1.00      1.00      1.00      1.00
  1:00      1.00      1.00      1.00      1.00
  2:00      1.00      1.00      1.00      1.00

day 7: Monday
  0:00      1.00      1.00      1.00      1.00
  1:00      1.00      1.00      1.00      1.00
  2:00      1.00      1.00      1.00      1.00

day 8: Tuesday
  0:00      1.00      1.00      1.00      1.00
  1:00      1.00      1.00      1.00      1.00
  2:00      1.00      1.00      1.00      1.00

day 9: Wednesday
  0:00      1.00      1.00      1.00      1.00
  1:00      1.00      1.00      1.00      1.00
  2:00      1.00      1.00      1.00      1.00

```

```
q quit
a add actual to total (using operator +)
m subtract actual from total (using operator -)
s scalar multiplication
c change unit
v sum up values
o output actual (using operator <<)
t output total (using operator <<)
u add consumption according to frequency of use (call functions add_consumption)
y set actual to ones (call function ones)
z set actual to zeros (call function zeros)
>> s
a for actual
t for total
a
value of scalar? 600
q quit
a add actual to total (using operator +)
m subtract actual from total (using operator -)
s scalar multiplication
c change unit
v sum up values
o output actual (using operator <<)
t output total (using operator <<)
u add consumption according to frequency of use (call functions add_consumption)
y set actual to ones (call function ones)
z set actual to zeros (call function zeros)
>> o
year: 2024 (unit: Watt)
day 0: Monday
0:00      600.00      600.00      600.00      600.00
1:00      600.00      600.00      600.00      600.00
2:00      600.00      600.00      600.00      600.00

day 1: Tuesday
0:00      600.00      600.00      600.00      600.00
1:00      600.00      600.00      600.00      600.00
2:00      600.00      600.00      600.00      600.00

day 2: Wednesday
0:00      600.00      600.00      600.00      600.00
1:00      600.00      600.00      600.00      600.00
2:00      600.00      600.00      600.00      600.00

day 3: Thursday
0:00      600.00      600.00      600.00      600.00
1:00      600.00      600.00      600.00      600.00
2:00      600.00      600.00      600.00      600.00

day 4: Friday
0:00      600.00      600.00      600.00      600.00
1:00      600.00      600.00      600.00      600.00
2:00      600.00      600.00      600.00      600.00

day 5: Saturday
0:00      600.00      600.00      600.00      600.00
1:00      600.00      600.00      600.00      600.00
2:00      600.00      600.00      600.00      600.00

day 6: Sunday
0:00      600.00      600.00      600.00      600.00
1:00      600.00      600.00      600.00      600.00
2:00      600.00      600.00      600.00      600.00

day 7: Monday
0:00      600.00      600.00      600.00      600.00
1:00      600.00      600.00      600.00      600.00
2:00      600.00      600.00      600.00      600.00

day 8: Tuesday
0:00      600.00      600.00      600.00      600.00
1:00      600.00      600.00      600.00      600.00
2:00      600.00      600.00      600.00      600.00

day 9: Wednesday
```

```
0:00      600.00      600.00      600.00      600.00
1:00      600.00      600.00      600.00      600.00
2:00      600.00      600.00      600.00      600.00

q quit
a add actual to total (using operator +)
m subtract actual from total (using operator -)
s scalar multiplication
c change unit
v sum up values
o output actual (using operator <<)
t output total (using operator <<)
u add consumption according to frequency of use (call functions add_consumption)
y set actual to ones (call function ones)
z set actual to zeros (call function zeros)
>> s
a for actual
t for total
a
value of scalar? 0.001
q quit
a add actual to total (using operator +)
m subtract actual from total (using operator -)
s scalar multiplication
c change unit
v sum up values
o output actual (using operator <<)
t output total (using operator <<)
u add consumption according to frequency of use (call functions add_consumption)
y set actual to ones (call function ones)
z set actual to zeros (call function zeros)
>> c
a for actual
t for total
a
what is the new unit? kW
q quit
a add actual to total (using operator +)
m subtract actual from total (using operator -)
s scalar multiplication
c change unit
v sum up values
o output actual (using operator <<)
t output total (using operator <<)
u add consumption according to frequency of use (call functions add_consumption)
y set actual to ones (call function ones)
z set actual to zeros (call function zeros)
>> o
year: 2024 (unit: kW)
day 0: Monday
0:00      0.60      0.60      0.60      0.60
1:00      0.60      0.60      0.60      0.60
2:00      0.60      0.60      0.60      0.60

day 1: Tuesday
0:00      0.60      0.60      0.60      0.60
1:00      0.60      0.60      0.60      0.60
2:00      0.60      0.60      0.60      0.60

day 2: Wednesday
0:00      0.60      0.60      0.60      0.60
1:00      0.60      0.60      0.60      0.60
2:00      0.60      0.60      0.60      0.60

day 3: Thursday
0:00      0.60      0.60      0.60      0.60
1:00      0.60      0.60      0.60      0.60
2:00      0.60      0.60      0.60      0.60

day 4: Friday
0:00      0.60      0.60      0.60      0.60
1:00      0.60      0.60      0.60      0.60
2:00      0.60      0.60      0.60      0.60

day 5: Saturday
```

0:00	0.60	0.60	0.60	0.60
1:00	0.60	0.60	0.60	0.60
2:00	0.60	0.60	0.60	0.60

day 6: Sunday

0:00	0.60	0.60	0.60	0.60
1:00	0.60	0.60	0.60	0.60
2:00	0.60	0.60	0.60	0.60

day 7: Monday

0:00	0.60	0.60	0.60	0.60
1:00	0.60	0.60	0.60	0.60
2:00	0.60	0.60	0.60	0.60

day 8: Tuesday

0:00	0.60	0.60	0.60	0.60
1:00	0.60	0.60	0.60	0.60
2:00	0.60	0.60	0.60	0.60

day 9: Wednesday

0:00	0.60	0.60	0.60	0.60
1:00	0.60	0.60	0.60	0.60
2:00	0.60	0.60	0.60	0.60

q quit

a add actual to total (using operator +)

m subtract actual from total (using operator -)

s scalar multiplication

c change unit

v sum up values

o output actual (using operator <<)

t output total (using operator <<)

u add consumption according to frequency of use (call functions add_consumption)

y set actual to ones (call function ones)

z set actual to zeros (call function zeros)

>> s

a for actual

t for total

a

value of scalar? 0.3

q quit

a add actual to total (using operator +)

m subtract actual from total (using operator -)

s scalar multiplication

c change unit

v sum up values

o output actual (using operator <<)

t output total (using operator <<)

u add consumption according to frequency of use (call functions add_consumption)

y set actual to ones (call function ones)

z set actual to zeros (call function zeros)

>> c

a for actual

t for total

a

what is the new unit? EUR

q quit

a add actual to total (using operator +)

m subtract actual from total (using operator -)

s scalar multiplication

c change unit

v sum up values

o output actual (using operator <<)

t output total (using operator <<)

u add consumption according to frequency of use (call functions add_consumption)

y set actual to ones (call function ones)

z set actual to zeros (call function zeros)

>> o

year: 2024 (unit: EUR)

day 0: Monday

0:00	0.18	0.18	0.18	0.18
1:00	0.18	0.18	0.18	0.18
2:00	0.18	0.18	0.18	0.18

day 1: Tuesday

0:00	0.18	0.18	0.18	0.18
------	------	------	------	------

1:00	0.18	0.18	0.18	0.18
2:00	0.18	0.18	0.18	0.18
day 2: Wednesday				
0:00	0.18	0.18	0.18	0.18
1:00	0.18	0.18	0.18	0.18
2:00	0.18	0.18	0.18	0.18
day 3: Thursday				
0:00	0.18	0.18	0.18	0.18
1:00	0.18	0.18	0.18	0.18
2:00	0.18	0.18	0.18	0.18
day 4: Friday				
0:00	0.18	0.18	0.18	0.18
1:00	0.18	0.18	0.18	0.18
2:00	0.18	0.18	0.18	0.18
day 5: Saturday				
0:00	0.18	0.18	0.18	0.18
1:00	0.18	0.18	0.18	0.18
2:00	0.18	0.18	0.18	0.18
day 6: Sunday				
0:00	0.18	0.18	0.18	0.18
1:00	0.18	0.18	0.18	0.18
2:00	0.18	0.18	0.18	0.18
day 7: Monday				
0:00	0.18	0.18	0.18	0.18
1:00	0.18	0.18	0.18	0.18
2:00	0.18	0.18	0.18	0.18
day 8: Tuesday				
0:00	0.18	0.18	0.18	0.18
1:00	0.18	0.18	0.18	0.18
2:00	0.18	0.18	0.18	0.18
day 9: Wednesday				
0:00	0.18	0.18	0.18	0.18
1:00	0.18	0.18	0.18	0.18
2:00	0.18	0.18	0.18	0.18

```

q quit
a add actual to total (using operator +)
m subtract actual from total (using operator -)
s scalar multiplication
c change unit
v sum up values
o output actual (using operator <<)
t output total (using operator <<)
u add consumption according to frequency of use (call functions add_consumption)
y set actual to ones (call function ones)
z set actual to zeros (call function zeros)
>> v
sum actual = 21.60 EUR
sum total = 0.00 Watt
q quit
a add actual to total (using operator +)
m subtract actual from total (using operator -)
s scalar multiplication
c change unit
v sum up values
o output actual (using operator <<)
t output total (using operator <<)
u add consumption according to frequency of use (call functions add_consumption)
y set actual to ones (call function ones)
z set actual to zeros (call function zeros)
>> c
a for actual
t for total
t
what is the new unit? EUR
q quit
a add actual to total (using operator +)

```

```

m subtract actual from total (using operator -)
s scalar multiplication
c change unit
v sum up values
o output actual (using operator <<)
t output total (using operator <<)
u add consumption according to frequency of use (call functions add_consumption)
y set actual to ones (call function ones)
z set actual to zeros (call function zeros)
>> a
q quit
a add actual to total (using operator +)
m subtract actual from total (using operator -)
s scalar multiplication
c change unit
v sum up values
o output actual (using operator <<)
t output total (using operator <<)
u add consumption according to frequency of use (call functions add_consumption)
y set actual to ones (call function ones)
z set actual to zeros (call function zeros)
>> o
year: 2024 (unit: EUR)
day 0: Monday
0:00      0.18      0.18      0.18      0.18
1:00      0.18      0.18      0.18      0.18
2:00      0.18      0.18      0.18      0.18

day 1: Tuesday
0:00      0.18      0.18      0.18      0.18
1:00      0.18      0.18      0.18      0.18
2:00      0.18      0.18      0.18      0.18

day 2: Wednesday
0:00      0.18      0.18      0.18      0.18
1:00      0.18      0.18      0.18      0.18
2:00      0.18      0.18      0.18      0.18

day 3: Thursday
0:00      0.18      0.18      0.18      0.18
1:00      0.18      0.18      0.18      0.18
2:00      0.18      0.18      0.18      0.18

day 4: Friday
0:00      0.18      0.18      0.18      0.18
1:00      0.18      0.18      0.18      0.18
2:00      0.18      0.18      0.18      0.18

day 5: Saturday
0:00      0.18      0.18      0.18      0.18
1:00      0.18      0.18      0.18      0.18
2:00      0.18      0.18      0.18      0.18

day 6: Sunday
0:00      0.18      0.18      0.18      0.18
1:00      0.18      0.18      0.18      0.18
2:00      0.18      0.18      0.18      0.18

day 7: Monday
0:00      0.18      0.18      0.18      0.18
1:00      0.18      0.18      0.18      0.18
2:00      0.18      0.18      0.18      0.18

day 8: Tuesday
0:00      0.18      0.18      0.18      0.18
1:00      0.18      0.18      0.18      0.18
2:00      0.18      0.18      0.18      0.18

day 9: Wednesday
0:00      0.18      0.18      0.18      0.18
1:00      0.18      0.18      0.18      0.18
2:00      0.18      0.18      0.18      0.18

q quit
a add actual to total (using operator +)
m subtract actual from total (using operator -)

```



```

s scalar multiplication
c change unit
v sum up values
o output actual (using operator <<)
t output total (using operator <<)
u add consumption according to frequency of use (call functions add_consumption)
y set actual to ones (call function ones)
z set actual to zeros (call function zeros)
>> t
year: 2024 (unit: EUR)
day 0: Monday
  0:00      0.18      0.18      0.18      0.18
  1:00      0.18      0.18      0.18      0.18
  2:00      0.18      0.18      0.18      0.18

day 1: Tuesday
  0:00      0.18      0.18      0.18      0.18
  1:00      0.18      0.18      0.18      0.18
  2:00      0.18      0.18      0.18      0.18

day 2: Wednesday
  0:00      0.18      0.18      0.18      0.18
  1:00      0.18      0.18      0.18      0.18
  2:00      0.18      0.18      0.18      0.18

day 3: Thursday
  0:00      0.18      0.18      0.18      0.18
  1:00      0.18      0.18      0.18      0.18
  2:00      0.18      0.18      0.18      0.18

day 4: Friday
  0:00      0.18      0.18      0.18      0.18
  1:00      0.18      0.18      0.18      0.18
  2:00      0.18      0.18      0.18      0.18

day 5: Saturday
  0:00      0.18      0.18      0.18      0.18
  1:00      0.18      0.18      0.18      0.18
  2:00      0.18      0.18      0.18      0.18

day 6: Sunday
  0:00      0.18      0.18      0.18      0.18
  1:00      0.18      0.18      0.18      0.18
  2:00      0.18      0.18      0.18      0.18

day 7: Monday
  0:00      0.18      0.18      0.18      0.18
  1:00      0.18      0.18      0.18      0.18
  2:00      0.18      0.18      0.18      0.18

day 8: Tuesday
  0:00      0.18      0.18      0.18      0.18
  1:00      0.18      0.18      0.18      0.18
  2:00      0.18      0.18      0.18      0.18

day 9: Wednesday
  0:00      0.18      0.18      0.18      0.18
  1:00      0.18      0.18      0.18      0.18
  2:00      0.18      0.18      0.18      0.18

q quit
a add actual to total (using operator +)
m subtract actual from total (using operator -)
s scalar multiplication
c change unit
v sum up values
o output actual (using operator <<)
t output total (using operator <<)
u add consumption according to frequency of use (call functions add_consumption)
y set actual to ones (call function ones)
z set actual to zeros (call function zeros)
>> v
sum actual = 21.60 EUR
sum total = 21.60 EUR
q quit

```

```
a add actual to total (using operator +)
m subtract actual from total (using operator -)
s scalar multiplication
c change unit
v sum up values
o output actual (using operator <<)
t output total (using operator <<)
u add consumption according to frequency of use (call functions add_consumption)
y set actual to ones (call function ones)
z set actual to zeros (call function zeros)
>> y
q quit
a add actual to total (using operator +)
m subtract actual from total (using operator -)
s scalar multiplication
c change unit
v sum up values
o output actual (using operator <<)
t output total (using operator <<)
u add consumption according to frequency of use (call functions add_consumption)
y set actual to ones (call function ones)
z set actual to zeros (call function zeros)
>> o
year: 2024 (unit: EUR)
day 0: Monday
0:00      1.00      1.00      1.00      1.00
1:00      1.00      1.00      1.00      1.00
2:00      1.00      1.00      1.00      1.00

day 1: Tuesday
0:00      1.00      1.00      1.00      1.00
1:00      1.00      1.00      1.00      1.00
2:00      1.00      1.00      1.00      1.00

day 2: Wednesday
0:00      1.00      1.00      1.00      1.00
1:00      1.00      1.00      1.00      1.00
2:00      1.00      1.00      1.00      1.00

day 3: Thursday
0:00      1.00      1.00      1.00      1.00
1:00      1.00      1.00      1.00      1.00
2:00      1.00      1.00      1.00      1.00

day 4: Friday
0:00      1.00      1.00      1.00      1.00
1:00      1.00      1.00      1.00      1.00
2:00      1.00      1.00      1.00      1.00

day 5: Saturday
0:00      1.00      1.00      1.00      1.00
1:00      1.00      1.00      1.00      1.00
2:00      1.00      1.00      1.00      1.00

day 6: Sunday
0:00      1.00      1.00      1.00      1.00
1:00      1.00      1.00      1.00      1.00
2:00      1.00      1.00      1.00      1.00

day 7: Monday
0:00      1.00      1.00      1.00      1.00
1:00      1.00      1.00      1.00      1.00
2:00      1.00      1.00      1.00      1.00

day 8: Tuesday
0:00      1.00      1.00      1.00      1.00
1:00      1.00      1.00      1.00      1.00
2:00      1.00      1.00      1.00      1.00

day 9: Wednesday
0:00      1.00      1.00      1.00      1.00
1:00      1.00      1.00      1.00      1.00
2:00      1.00      1.00      1.00      1.00

q quit
a add actual to total (using operator +)
```

```
m subtract actual from total (using operator -)
s scalar multiplication
c change unit
v sum up values
o output actual (using operator <<)
t output total (using operator <<)
u add consumption according to frequency of use (call functions add_consumption)
y set actual to ones (call function ones)
z set actual to zeros (call function zeros)
>> s
a for actual
t for total
a
value of scalar? 0.03
q quit
a add actual to total (using operator +)
m subtract actual from total (using operator -)
s scalar multiplication
c change unit
v sum up values
o output actual (using operator <<)
t output total (using operator <<)
u add consumption according to frequency of use (call functions add_consumption)
y set actual to ones (call function ones)
z set actual to zeros (call function zeros)
>> o
year: 2024 (unit: EUR)
day 0: Monday
0:00      0.03      0.03      0.03      0.03
1:00      0.03      0.03      0.03      0.03
2:00      0.03      0.03      0.03      0.03

day 1: Tuesday
0:00      0.03      0.03      0.03      0.03
1:00      0.03      0.03      0.03      0.03
2:00      0.03      0.03      0.03      0.03

day 2: Wednesday
0:00      0.03      0.03      0.03      0.03
1:00      0.03      0.03      0.03      0.03
2:00      0.03      0.03      0.03      0.03

day 3: Thursday
0:00      0.03      0.03      0.03      0.03
1:00      0.03      0.03      0.03      0.03
2:00      0.03      0.03      0.03      0.03

day 4: Friday
0:00      0.03      0.03      0.03      0.03
1:00      0.03      0.03      0.03      0.03
2:00      0.03      0.03      0.03      0.03

day 5: Saturday
0:00      0.03      0.03      0.03      0.03
1:00      0.03      0.03      0.03      0.03
2:00      0.03      0.03      0.03      0.03

day 6: Sunday
0:00      0.03      0.03      0.03      0.03
1:00      0.03      0.03      0.03      0.03
2:00      0.03      0.03      0.03      0.03

day 7: Monday
0:00      0.03      0.03      0.03      0.03
1:00      0.03      0.03      0.03      0.03
2:00      0.03      0.03      0.03      0.03

day 8: Tuesday
0:00      0.03      0.03      0.03      0.03
1:00      0.03      0.03      0.03      0.03
2:00      0.03      0.03      0.03      0.03

day 9: Wednesday
0:00      0.03      0.03      0.03      0.03
1:00      0.03      0.03      0.03      0.03
```

```

2:00      0.03      0.03      0.03      0.03

q quit
a add actual to total (using operator +)
m subtract actual from total (using operator -)
s scalar multiplication
c change unit
v sum up values
o output actual (using operator <<)
t output total (using operator <<)
u add consumption according to frequency of use (call functions add_consumption)
y set actual to ones (call function ones)
z set actual to zeros (call function zeros)
>> m
q quit
a add actual to total (using operator +)
m subtract actual from total (using operator -)
s scalar multiplication
c change unit
v sum up values
o output actual (using operator <<)
t output total (using operator <<)
u add consumption according to frequency of use (call functions add_consumption)
y set actual to ones (call function ones)
z set actual to zeros (call function zeros)
>> t
year: 2024 (unit: EUR)
day 0: Monday
0:00      0.15      0.15      0.15      0.15
1:00      0.15      0.15      0.15      0.15
2:00      0.15      0.15      0.15      0.15

day 1: Tuesday
0:00      0.15      0.15      0.15      0.15
1:00      0.15      0.15      0.15      0.15
2:00      0.15      0.15      0.15      0.15

day 2: Wednesday
0:00      0.15      0.15      0.15      0.15
1:00      0.15      0.15      0.15      0.15
2:00      0.15      0.15      0.15      0.15

day 3: Thursday
0:00      0.15      0.15      0.15      0.15
1:00      0.15      0.15      0.15      0.15
2:00      0.15      0.15      0.15      0.15

day 4: Friday
0:00      0.15      0.15      0.15      0.15
1:00      0.15      0.15      0.15      0.15
2:00      0.15      0.15      0.15      0.15

day 5: Saturday
0:00      0.15      0.15      0.15      0.15
1:00      0.15      0.15      0.15      0.15
2:00      0.15      0.15      0.15      0.15

day 6: Sunday
0:00      0.15      0.15      0.15      0.15
1:00      0.15      0.15      0.15      0.15
2:00      0.15      0.15      0.15      0.15

day 7: Monday
0:00      0.15      0.15      0.15      0.15
1:00      0.15      0.15      0.15      0.15
2:00      0.15      0.15      0.15      0.15

day 8: Tuesday
0:00      0.15      0.15      0.15      0.15
1:00      0.15      0.15      0.15      0.15
2:00      0.15      0.15      0.15      0.15

day 9: Wednesday
0:00      0.15      0.15      0.15      0.15
1:00      0.15      0.15      0.15      0.15
2:00      0.15      0.15      0.15      0.15

```

```
q quit
a add actual to total (using operator +)
m subtract actual from total (using operator -)
s scalar multiplication
c change unit
v sum up values
o output actual (using operator <<)
t output total (using operator <<)
u add consumption according to frequency of use (call functions add_consumption)
y set actual to ones (call function ones)
z set actual to zeros (call function zeros)
>> v
sum actual = 3.60 EUR
sum total = 18.00 EUR
q quit
a add actual to total (using operator +)
m subtract actual from total (using operator -)
s scalar multiplication
c change unit
v sum up values
o output actual (using operator <<)
t output total (using operator <<)
u add consumption according to frequency of use (call functions add_consumption)
y set actual to ones (call function ones)
z set actual to zeros (call function zeros)
>>
```

Zuletzt geändert: Montag, 20. November 2023, 00:03

[◀ A3 Upload Teil 1/Part 1](#)

Direkt zu:

[A3 Upload Teil 1+2/Upload Part 1+2 ▶](#)

- Deutsch (de)
- Dansk (da)

Deutsch (de)

English (en)

Español - España (es_es)

Español - Internacional (es)

Français (fr)

Polski (pl)

Türkçe (tr)

Русский (ru)

Українська (uk)

