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

A3 Teil 1: Hausaufgabe zur Vorbereitung auf die Präsenz-Gruppe/Part 1: Homework Task for Preparation of the Presence Group

Lernziele: inline-Funktionen, überladene Funktionen, überladene Operatoren./
Learning objectives: inline functions, overloaded functions, overloaded operators.

In dieser Aufgabe soll eine individuelle Berechnung des Stromverbrauchs für elektrische Verbraucher vorbereitet werden, die pauschale Berechnungen ersetzen kann. Die kleinste Zeiteinheit, in welcher der Verbrauch von Strom von Kunden und die Lieferung durch den Versorger überwacht und gehandelt werden, ist die Viertelstunde. Wie im Strommarkt in Deutschland soll deshalb in dieser Aufgabe der Verbrauch über das gesamte Jahr in Viertelstunden-Intervallen bearbeitet werden können./
In this task, an individual calculation of power consumption for power consumers is to be prepared, which can replace flat-rate calculations. The smallest unit of time in which the consumption of power by customers and the supply by the supplier are monitored and traded is the quarter of an hour. As in the electricity market in Germany, it should therefore be possible in this task to process consumption over the entire year in quarter-hour intervals.

1. Erweitern Sie die C++-Aufzählung namens **`Use`** (Häufigkeit der Benutzung) um die Werte **`monday`**, **`tuesday`**, **`wednesday`**, **`thursday`**, **`friday`**, **`saturday`** und **`sunday`**, belassen Sie diesen folgend die Aufzählungswerte **`once`** (einmal), **`daily`** (täglich), **`mo_fr`** (montags bis freitags) und **`sa_su`** (samstags und sonntags) und löschen Sie **`weekly`** (wöchentlich)./
   Add the values **`monday`**, **`tuesday`**, **`wednesday`**, **`thursday`**, **`friday`**, **`saturday`** and **`sunday`** to the C++ enumeration called **`Use`**, leave the enumeration values **`once`**, **`daily`**, **`mo_fr`** (Monday to Friday) and **`sa_su`** (Saturday and Sunday) and delete **`weekly`**.
2. Programmieren Sie einen überladenen binären Ausgabe-Operator **`<<`** für die Aufzählungswerte von **`Use`**, der obiger Reihenfolge entsprechend **`Monday`**, **`Tuesday`**, **`Wednesday`**, **`Thursday`**, **`Friday`**, **`Saturday`**, **`Sunday`**, **`once`**, **`daily`**, **`Monday to Friday`** beziehungsweise **`Saturday and Sunday`** auf den im ersten Parameter übergebenen Ausgabestrom schreibt./
   Program an overloaded binary output operator **`<<`** for the enumeration values of **`Use`**, which writes **`Monday`**, **`Tuesday`**, **`Wednesday`**, **`Thursday`**, **`Friday`**, **`Saturday`**, **`Sunday`**, **`once`**, **`daily`**, **`Monday to Friday`** or **`Saturday and Sunday`** to the output stream passed in the first parameter according to the above sequence of enumeration values.
3. Programmieren Sie einen überladenen unären Operator **`++`** für die Aufzählungswerte von **`Use`**, der den Wochentag inkrementiert, also **`tuesday`** für den Parameterwert **`monday`** zurück liefert, **`wednesday`** für den Parameterwert **`tuesday`**, ..., **`sunday`** für den Parameterwert **`saturday`**, **`monday`** für den Parameterwert **`sunday`**, und für **`once`**, **`daily`**, **`mo_fr`**, **`once`** und **`sa_su`** den gleichen Wert./
   Program an overloaded unary operator **`++`** for the enumeration values of **`Use`**, which increments the day of the week, i.e. returns **`tuesday`** for the parameter value **`monday`**, **`wednesday`** for the parameter value **`tuesday`**, ..., **`sunday`** for the parameter value **`saturday`**, **`monday`** for the parameter value **`sunday`**, and the same value for **`once`**, **`daily`**, **`mo_fr`**, **`once`** and **`sa_su`**.
4. Ändern Sie Ihre Funktion **`input_use`** passend auf alle obigen Aufzählungswerte ab (siehe auch Beispiele unten)./
   Modify your function **`input_use`** fitting to all above enumeration values (see also examples below).
5. Definieren Sie zwei weitere globale ganzzahlige konstante Variable mit den Werten **`365`** für die Anzahl Tage in einem Jahr (10 bei ersten Tests, siehe Hinweise und Programmlauf 1 unten) und **`96`** = 24 * 4 für die Anzahl Viertelstunden-Intervalle an einem Tag./
   Define two further global integer constant variables with the values **`365`** for the number of days in a year (10 for the first tests, see notes and program run 1 below) and **`96`** = 24 * 4 for the number of quarter-hour intervals in a day.

6.

quarter	00:00 [0]	00:15 [1]	00:30 [2]	00:45 [3]	01:00 [4]	01:15 [5]	01:30 [6]	01:45 [7]	02:00 [8]	...	22:45 [91]	23:00 [92]	23:15 [93]	23:30 [94]	23:45 [95]
[0]	2.5	2.5	2.5	2.5	2.5	2.5	2.5	2.5	2.5	...	2.5	2.5	2.5	2.5	2.5
[1]	2.5	2.5	2.5	2.5	2.5	2.5	2.5	2.5	2.5	...	2.5	2.5	2.5	2.5	2.5
[2]	2.5	2.5	2.5	2.5	2.5	2.5	2.5	2.5	2.5	...	2.5	2.5	2.5	2.5	2.5
[3]	2.5	2.5	2.5	2.5	2.5	2.5	2.5	2.5	2.5	...	2.5	2.5	2.5	2.5	2.5
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮		⋮	⋮	⋮	⋮	⋮
[362]	2.5	2.5	2.5	2.5	2.5	2.5	2.5	2.5	2.5	...	2.5	2.5	2.5	2.5	2.5
[363]	2.5	2.5	2.5	2.5	2.5	2.5	2.5	2.5	2.5	...	2.5	2.5	2.5	2.5	2.5
[364]	2.5	2.5	2.5	2.5	2.5	2.5	2.5	2.5	2.5	...	2.5	2.5	2.5	2.5	2.5

Definieren Sie eine Struktur mit Namen **`year`** (Jahr) mit folgenden Komponenten:

- ganzzahliger Wert für die Jahreszahl.
- Aufzählungswert vom obigen Typ **`Use`** für den ersten Wochentag im Jahr.
- C++-Zeichenkette für die Einheit der Werte gespeichert im Feld dieser Struktur.
- zweidimensionales Feld von Gleitpunktzahlen mit der Anzahl Zeilen für die Tage im Jahr gegeben über die erste in der vorherigen Teilaufgabe definierten Konstante und der Anzahl Spalten für die Viertelstunden-Intervalle gegeben über die zweite obige Konstante./

Define a structure called **`year`** with the following components:

- integer value for the year itself.
- enumeration value of the above type **`Use`** for the first day of the week in the year..
- C++ string for the unit of the values stored in the array of this structure.
- two-dimensional array of floating point numbers with the number of rows for the days given by the first constant defined in the previous subtask and the number of columns for the quarter-hour intervals given by the second constant above.

7. Programmieren Sie eine **`inline`** Funktion namens **`zeros`** (Nullen), 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 **`0.0`** und geben danach die Referenzvariable als Funktionswert zurück./
   Program an **`inline`** function called **`zeros`**, 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 **`0.0`** and then return the reference variable as function value.
8. Programmieren Sie eine **`inline`** Funktion namens **`time`** (Uhrzeit), die zwei ganzzahlige Werte für die Stunde **`h`** als ersten und für die Minute **`m`** als zweiten Parameter hat und eine ganzzahlige Rückgabe.
   Berechnen Sie im Rumpf die Minute des Tages (**`h * 60 + m`**) und geben diese als Funktionswert zurück, also bspw. 795 = 13 * 60 + 15 für 13:15 Uhr./
   Program an **`inline`** function called **`time`**, which has two integer values for the hour **`h`** as first parameter and for the minute **`m`** as second parameter and an integer return.
   Calculate the minute of the day (h * 60 + m) in the body and return this as a function value, e.g. 795 = 13 * 60 + 15 for 13:15 o'clock.
9. Programmieren Sie einen überladenen binären Ausgabe-Operator **`<<`** für eine Referenzvariable vom Strukturtyp **`year`**.
   Schreiben Sie im Rumpf zuerst **`year:`** und die Jahreszahl aus der Struktur auf den im ersten Parameter übergebenen Ausgabestrom.
   Schreiben Sie danach zeilenweise für jeden Tag im Jahr die Nummer des Tags und den Wochentag auf den Ausgabstrom, die Uhrzeiten aller vollen Stunden und die jeweils vier gespeicherten Viertelstundenwerte (siehe Beispiele unten).
   Verwenden Sie für die Ausgabe des jeweiligen Wochentags und die Berechnung des nächsten Wochentags den oben definierten Ausgabe-Operator **<<** und den Inkrement-Operator **++**./
   Program an overloaded binary output operator **<<** for a reference variable of structure type **`year`**.
   In the body, first write **`year:`** and the strucure's year onto the output stream passed in the first parameter. Then line by line write the number of the day and the day of the week for each day of the year onto teh output stream, the full hours clock time with following four stored quarter-hour values (see examples below).
   Use the output operator **<<** defined above and the increment operator **++** to output the respective day of the week and to calculate the next day of the week.
10. Programmieren Sie einen überladenen binären Additions-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`**, addieren für diese elementweise alle Viertelstundenwerte und geben diese Variable zurück./
    Program an overloaded binary addition 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`**, add for this all quarter-hour values element by element and return this variable.

11. Programmieren Sie eine überladene Funktion namens **`add_consumption`**, die eine Referenz vom Strukturtyp **`year`** als ersten Parameter hat, drei ganzzahlige Werte für einen bestimmten Tag im Jahr (also **`once`** (einmalig)), zwei Minutenwerte für diesen Tag von wann bis wann der Stromverbraucher an diesem Tag bei welcher Wattzahl (übergeben in einem fünften Parameter als Gleitpunktzahl) zum Verbrauch in dem Feld hinzu addiert werden soll. Die Funktion soll keine Rückgabe haben und im Rumpf für jede Verbrauchsminute den entsprechend berechneten Viertelstundenwert erhöhen.
    Hinweis: das jeweilige Viertelstunden-Intervall können Sie als Ganzzahlanteil der Division der Tagesminute durch 15 Minuten berechnen und die minütliche Wattzahl durch Division des Verbrauchswerts durch 60 Minuten./
    Program an overloaded function called **`add_consumption`**, which has a reference of the structure type **`year`** as the first parameter, three integer values for a certain day in the year (i.e. **`once`**), two minute values for this day from when to when the power consumer on this day is to be added to the consumption in the array at which wattage (passed in the fifth parameter as a floating point number). The function should have no return and increase the corresponding calculated quarter-hour value in the body for each minute of consumption.
    Note: the respective quarter-hour interval can be obtained as the integer part of the division of the minute value of the day by 15 minutes and the wattage per minute by dividing the consumption value by 60 minutes.
12. Programmieren Sie eine weitere überladene Funktion namens **`add_consumption`**, die als einzigen Unterschied zur vorherigen Funktion eine Variable vom Aufzählungstyp **`Use`** als zweiten Parameter hat und keine Rückgabe.
    Im Rumpf soll dann statt nur an einem bestimmten Tag wie in der Teilaufgabe zuvor täglich, montags bis freitags, samstags und sonntags oder an allen Wochentagen im kompletten Jahr der jeweilige Verbrauch in den angegebenen Zeiten von-bis bei den Viertelstundenwerten addiert werden (siehe Beispiele unten)./
    Program another overloaded function called **`add_consumption`**, which as the only difference to the previous function has a variable of the enumeration type **`Use`** as its second parameter and no return.
    In the body, the respective consumption in the specified time interval from-to should then be added to the quarter-hour values instead of just on one certain day as in the subtask before regarding daily, Mondays to Fridays, Saturdays and Sundays or all weekdays in the entire year (see examples below).
13. Programmieren Sie eine **`inline`** Funktion namens **`sum`** (Summe), die eine Referenzvariable vom Strukturtyp **`year`** als Operanden hat und eine Gleitpunktzahl zurück gibt.
    Summieren Sie im Rumpf alle Viertelstunde-Intervallwerte des gesamten Jahres auf und geben den Summenwert zurück./
    Program an **`inline`** function called **`sum`**, which has a reference variable of structure type **`year`** as operand and returns a floating point number.
    Sum up all quarter-hour interval values of the whole year in the body and return the sum value.
14. Schreiben Sie eine **`main`**-Funktion, in der die oben definierten Funktionalitäten getestet werden können.
    Definieren Sie hierzu neben allen weiteren von Ihnen benötigten Variablen zwei Strukturvariable namens **`actual`** und **`total`** vom Typ **`year`** und initialisieren diese mit der Jahreszahl **`2024`**, **`Montag`** als erstem Tag im Jahr, der Einheit **`Watt`** und Nullen für alle Viertelstunden-Intervallwerte.
    Initialisieren Sie als Preis für eine kWh Strom **`30`** ct/kWh wie in den Aufgaben A1 und A2 zuvor.
    Implementieren Sie dann ein kleines Menü wie in den Beispielen unten und rufen in den Menüpunkten jeweils die oben definierten Operatoren und Funktionen auf:
    - **`a`** hier ist nur **`total = total + actual;`** auszuführen.
    - **`c`** rufen Sie jeweils mit **`actual`** und auch mit **`total`** als Parameter Ihre Funktion **`sum`** auf und geben die Summenwerte für den Jahres-Stromverbrauch sowie die mit dem kWh-Preis multiplizierten Kosten aus (Hinweis: rechnen Sie hierzu Watt in Kilowatt um).
    - **`o`** hier ist nur **`cout << actual;`** aufzurufen.
    - **`t`** hier ist nur **`cout << total;`** aufzurufen.
    - **`u`** rufen Sie je nach Eingabe passend zur Nutzungsfrequenz und der weiteren Werte eine der beiden überladenen Funktionen **`add_consumption`** auf.
    - **`z`** rufen Sie Ihre Funktion **`zeros`** für **`actual`** auf./
    Write a function **`main`** in which the functionalities defined above can be tested.
    In addition to all the other variables you need, define two structure variables named **`actual`** and **`total`** of type year and initialise them with the year **`2024`**, **`Monday`** as the first day of the year, the unit **`Watt`** and zeros for all quarter-hour interval values.
    Initialise a price of 30 ct/kWh for one kWh of power as in tasks A1 and A2 above.
    Then implement a small menu as in the examples below and call the operators and functions defined above in the menu items:
    - **`a`** here only **`total = total + actual;`** is to be executed.
    - **`c`** call your function **`sum`** with **`actual`** and also with **`total`** as parameter and output the total values for the annual power consumption and the costs multiplied by the kWh price (note: convert watts to kilowatts for this)..
    - **`o`** here only **`cout << actual;`** is to be called.
    - **`t`** here only **`cout << total;`** is to be called.
    - **`u`** call one of the two overloaded functions **`add_consumption`** fitting to the inputted frequency of use and the other inputted values.
    - **`z`** call your function **`zeros`** for **`actual`**.

Wichtige Hinweise

Gehen Sie bei der Entwicklung am besten schrittweise vor - schreiben Ihr Programm und Menü erst einmal mit nur einer einfachen Funktionalität, und erweitern dieses dann schrittweise.
Da die Anzahl der Viertelstundenwerte in einem Jahr recht groß ist und die Ausgabe dafür sowohl lang und unübersichtlich wird als auch

zeitlich länger dauert, entwickeln und testen Sie Ihr Programm zunächst mit einem kleineren Wert für die Tage in einem Jahr (bspw. 10 bei ersten Tests) und auch einem kleineren Wert für die Viertelstundenwerte eines Tages (bspw. 12 bei ersten Tests; Uhrzeit nur bis 03:00 Uhr) - siehe Programmlauf 1.
Vergleichen Sie erst nach ausreichendem Testen Ihrer Funktionalität danach bei 365 Tagen im Jahr mit jeweils 96 Viertelstundenwerten die berechneten Summenwerte für den Jahresverbrauch und die Jahres-Stromkosten mit den pauschal berechneten Werten für eine Waschmaschine, einen Router und einen Office-PC aus Aufgabe A2 - siehe Programmlauf 2 (während Waschmaschine und Router gleiche Jahres-Verbrauchswerte und Stromkosten ergeben, zeigen sich kleine Rundungsfehler beim Office-PC mit Betriebs- und Standbyzeiten)./

Important notes

It is best to proceed step by step during development - write your program and menu with only one simple functionality at first, and then extend it step by step.
As the number of quarter-hour values in a year is quite large and the output is both long and confusing and takes longer, first develop and test your program with a smaller value for the days in a year (e.g. 10 for initial tests) and also a smaller value for the quarter-hour values of a day (e.g. 12 for initial tests; only time until 03:00 o'clock) - see program run 1.
Only after sufficiently testing your functionality, compare the calculated total values for the annual consumption and the annual power costs for 365 days a year, each with 96 quarter-hour values, with the flat-rate calculated values for a washing machine, a router and an office PC from task A2 - see program run 2 (while the washing machine and router result in the same annual consumption values and power costs, small rounding errors occur for the office PC with operating and standby times).

Beispiel Programmlauf 1 für 10 Tage und 12 Viertelstundenwerte/Example Program Run 1 for 10 days and 12 quarter hours

```
YEARLY CONSUMPTION QUARTER HOUR
q quit
a add actual to total (using operator +)
c annual consumption and cost (price for one kWh: 30.00 ct/kWh; calling function sum)
o output actual (using operator <<)
t output total (using operator <<)
u add consumption according to frequency of use (call functions add_consumption)
z set actual to zeros (call function zeros)
>> o
year: 2024
day 0: Monday
 0:00       0.00       0.00       0.00       0.00
 1:00       0.00       0.00       0.00       0.00
 2:00       0.00       0.00       0.00       0.00

day 1: Tuesday
 0:00       0.00       0.00       0.00       0.00
 1:00       0.00       0.00       0.00       0.00
 2:00       0.00       0.00       0.00       0.00

day 2: Wednesday
 0:00       0.00       0.00       0.00       0.00
 1:00       0.00       0.00       0.00       0.00
 2:00       0.00       0.00       0.00       0.00

day 3: Thursday
 0:00       0.00       0.00       0.00       0.00
 1:00       0.00       0.00       0.00       0.00
 2:00       0.00       0.00       0.00       0.00

day 4: Friday
 0:00       0.00       0.00       0.00       0.00
 1:00       0.00       0.00       0.00       0.00
 2:00       0.00       0.00       0.00       0.00

day 5: Saturday
 0:00       0.00       0.00       0.00       0.00
 1:00       0.00       0.00       0.00       0.00
 2:00       0.00       0.00       0.00       0.00

day 6: Sunday
 0:00       0.00       0.00       0.00       0.00
 1:00       0.00       0.00       0.00       0.00
 2:00       0.00       0.00       0.00       0.00

day 7: Monday
 0:00       0.00       0.00       0.00       0.00
 1:00       0.00       0.00       0.00       0.00
 2:00       0.00       0.00       0.00       0.00

day 8: Tuesday
 0:00       0.00       0.00       0.00       0.00
 1:00       0.00       0.00       0.00       0.00
 2:00       0.00       0.00       0.00       0.00

day 9: Wednesday
 0:00       0.00       0.00       0.00       0.00
 1:00       0.00       0.00       0.00       0.00
 2:00       0.00       0.00       0.00       0.00

q quit
a add actual to total (using operator +)
c annual consumption and cost (price for one kWh: 30.00 ct/kWh; calling function sum)
o output actual (using operator <<)
t output total (using operator <<)
u add consumption according to frequency of use (call functions add_consumption)
z set actual to zeros (call function zeros)
>> t
year: 2024
day 0: Monday
 0:00       0.00       0.00       0.00       0.00
 1:00       0.00       0.00       0.00       0.00
 2:00       0.00       0.00       0.00       0.00

day 1: Tuesday
```

```
 0:00       0.00       0.00       0.00       0.00
 1:00       0.00       0.00       0.00       0.00
 2:00       0.00       0.00       0.00       0.00

day 2: Wednesday
 0:00       0.00       0.00       0.00       0.00
 1:00       0.00       0.00       0.00       0.00
 2:00       0.00       0.00       0.00       0.00

day 3: Thursday
 0:00       0.00       0.00       0.00       0.00
 1:00       0.00       0.00       0.00       0.00
 2:00       0.00       0.00       0.00       0.00

day 4: Friday
 0:00       0.00       0.00       0.00       0.00
 1:00       0.00       0.00       0.00       0.00
 2:00       0.00       0.00       0.00       0.00

day 5: Saturday
 0:00       0.00       0.00       0.00       0.00
 1:00       0.00       0.00       0.00       0.00
 2:00       0.00       0.00       0.00       0.00

day 6: Sunday
 0:00       0.00       0.00       0.00       0.00
 1:00       0.00       0.00       0.00       0.00
 2:00       0.00       0.00       0.00       0.00

day 7: Monday
 0:00       0.00       0.00       0.00       0.00
 1:00       0.00       0.00       0.00       0.00
 2:00       0.00       0.00       0.00       0.00

day 8: Tuesday
 0:00       0.00       0.00       0.00       0.00
 1:00       0.00       0.00       0.00       0.00
 2:00       0.00       0.00       0.00       0.00

day 9: Wednesday
 0:00       0.00       0.00       0.00       0.00
 1:00       0.00       0.00       0.00       0.00
 2:00       0.00       0.00       0.00       0.00

q quit
a add actual to total (using operator +)
c annual consumption and cost (price for one kWh: 30.00 ct/kWh; calling function sum)
o output actual (using operator <<)
t output total (using operator <<)
u add consumption according to frequency of use (call functions add_consumption)
z set actual to zeros (call function zeros)
>> u
how often it will be used?
Monday    (0)
Tuesday   (1)
Wednesday (2)
Thursday  (3)
Friday    (4)
Saturday  (5)
Sunday    (6)
daily     (d)
mo_fr     (m)
once      (o)
sa_su     (s)?
d
from hour:minute? 00:20
to hour:minute? 01:37
how many watt it will have? 60
q quit
a add actual to total (using operator +)
c annual consumption and cost (price for one kWh: 30.00 ct/kWh; calling function sum)
o output actual (using operator <<)
t output total (using operator <<)
u add consumption according to frequency of use (call functions add_consumption)
z set actual to zeros (call function zeros)
>> o
```

```
year: 2024
day 0: Monday
 0:00       0.00      10.00      15.00      15.00
 1:00      15.00      15.00       7.00       0.00
 2:00       0.00       0.00       0.00       0.00

day 1: Tuesday
 0:00       0.00      10.00      15.00      15.00
 1:00      15.00      15.00       7.00       0.00
 2:00       0.00       0.00       0.00       0.00

day 2: Wednesday
 0:00       0.00      10.00      15.00      15.00
 1:00      15.00      15.00       7.00       0.00
 2:00       0.00       0.00       0.00       0.00

day 3: Thursday
 0:00       0.00      10.00      15.00      15.00
 1:00      15.00      15.00       7.00       0.00
 2:00       0.00       0.00       0.00       0.00

day 4: Friday
 0:00       0.00      10.00      15.00      15.00
 1:00      15.00      15.00       7.00       0.00
 2:00       0.00       0.00       0.00       0.00

day 5: Saturday
 0:00       0.00      10.00      15.00      15.00
 1:00      15.00      15.00       7.00       0.00
 2:00       0.00       0.00       0.00       0.00

day 6: Sunday
 0:00       0.00      10.00      15.00      15.00
 1:00      15.00      15.00       7.00       0.00
 2:00       0.00       0.00       0.00       0.00

day 7: Monday
 0:00       0.00      10.00      15.00      15.00
 1:00      15.00      15.00       7.00       0.00
 2:00       0.00       0.00       0.00       0.00

day 8: Tuesday
 0:00       0.00      10.00      15.00      15.00
 1:00      15.00      15.00       7.00       0.00
 2:00       0.00       0.00       0.00       0.00

day 9: Wednesday
 0:00       0.00      10.00      15.00      15.00
 1:00      15.00      15.00       7.00       0.00
 2:00       0.00       0.00       0.00       0.00

q quit
a add actual to total (using operator +)
c annual consumption and cost (price for one kWh: 30.00 ct/kWh; calling function sum)
o output actual (using operator <<)
t output total (using operator <<)
u add consumption according to frequency of use (call functions add_consumption)
z set actual to zeros (call function zeros)
>> c
sum actual = 770.00 Watt (costs: 0.23 EUR)
sum total = 0.00 Watt (costs: 0.00 EUR)
q quit
a add actual to total (using operator +)
c annual consumption and cost (price for one kWh: 30.00 ct/kWh; calling function sum)
o output actual (using operator <<)
t output total (using operator <<)
u add consumption according to frequency of use (call functions add_consumption)
z set actual to zeros (call function zeros)
>> a
q quit
a add actual to total (using operator +)
c annual consumption and cost (price for one kWh: 30.00 ct/kWh; calling function sum)
o output actual (using operator <<)
t output total (using operator <<)
u add consumption according to frequency of use (call functions add_consumption)
z set actual to zeros (call function zeros)
```

```
>> c
sum actual = 770.00 Watt (costs: 0.23 EUR)
sum total = 770.00 Watt (costs: 0.23 EUR)
q quit
a add actual to total (using operator +)
c annual consumption and cost (price for one kWh: 30.00 ct/kWh; calling function sum)
o output actual (using operator <<)
t output total (using operator <<)
u add consumption according to frequency of use (call functions add_consumption)
z set actual to zeros (call function zeros)
>> z
q quit
a add actual to total (using operator +)
c annual consumption and cost (price for one kWh: 30.00 ct/kWh; calling function sum)
o output actual (using operator <<)
t output total (using operator <<)
u add consumption according to frequency of use (call functions add_consumption)
z set actual to zeros (call function zeros)
>> c
sum actual = 0.00 Watt (costs: 0.00 EUR)
sum total = 770.00 Watt (costs: 0.23 EUR)
q quit
a add actual to total (using operator +)
c annual consumption and cost (price for one kWh: 30.00 ct/kWh; calling function sum)
o output actual (using operator <<)
t output total (using operator <<)
u add consumption according to frequency of use (call functions add_consumption)
z set actual to zeros (call function zeros)
>> u
how often it will be used?
Monday    (0)
Tuesday   (1)
Wednesday (2)
Thursday  (3)
Friday    (4)
Saturday  (5)
Sunday    (6)
daily     (d)
mo_fr     (m)
once      (o)
sa_su     (s)?
1
from hour:minute? 01:50
to hour:minute? 02:33
how many watt it will have? 100
q quit
a add actual to total (using operator +)
c annual consumption and cost (price for one kWh: 30.00 ct/kWh; calling function sum)
o output actual (using operator <<)
t output total (using operator <<)
u add consumption according to frequency of use (call functions add_consumption)
z set actual to zeros (call function zeros)
>> o
year: 2024
day 0: Monday
 0:00       0.00       0.00       0.00       0.00
 1:00       0.00       0.00       0.00       0.00
 2:00       0.00       0.00       0.00       0.00

day 1: Tuesday
 0:00       0.00       0.00       0.00       0.00
 1:00       0.00       0.00       0.00      16.67
 2:00      25.00      25.00       5.00       0.00

day 2: Wednesday
 0:00       0.00       0.00       0.00       0.00
 1:00       0.00       0.00       0.00       0.00
 2:00       0.00       0.00       0.00       0.00

day 3: Thursday
 0:00       0.00       0.00       0.00       0.00
 1:00       0.00       0.00       0.00       0.00
 2:00       0.00       0.00       0.00       0.00

day 4: Friday
 0:00       0.00       0.00       0.00       0.00
```

```
 1:00       0.00       0.00       0.00       0.00
 2:00       0.00       0.00       0.00       0.00

day 5: Saturday
 0:00       0.00       0.00       0.00       0.00
 1:00       0.00       0.00       0.00       0.00
 2:00       0.00       0.00       0.00       0.00

day 6: Sunday
 0:00       0.00       0.00       0.00       0.00
 1:00       0.00       0.00       0.00       0.00
 2:00       0.00       0.00       0.00       0.00

day 7: Monday
 0:00       0.00       0.00       0.00       0.00
 1:00       0.00       0.00       0.00       0.00
 2:00       0.00       0.00       0.00       0.00

day 8: Tuesday
 0:00       0.00       0.00       0.00       0.00
 1:00       0.00       0.00       0.00      16.67
 2:00      25.00      25.00       5.00       0.00

day 9: Wednesday
 0:00       0.00       0.00       0.00       0.00
 1:00       0.00       0.00       0.00       0.00
 2:00       0.00       0.00       0.00       0.00

q quit
a add actual to total (using operator +)
c annual consumption and cost (price for one kWh: 30.00 ct/kWh; calling function sum)
o output actual (using operator <<)
t output total (using operator <<)
u add consumption according to frequency of use (call functions add_consumption)
z set actual to zeros (call function zeros)
>> c
sum actual = 143.33 Watt (costs: 0.04 EUR)
sum total = 770.00 Watt (costs: 0.23 EUR)
q quit
a add actual to total (using operator +)
c annual consumption and cost (price for one kWh: 30.00 ct/kWh; calling function sum)
o output actual (using operator <<)
t output total (using operator <<)
u add consumption according to frequency of use (call functions add_consumption)
z set actual to zeros (call function zeros)
>> u
how often it will be used?
Monday    (0)
Tuesday   (1)
Wednesday (2)
Thursday  (3)
Friday    (4)
Saturday  (5)
Sunday    (6)
daily     (d)
mo_fr     (m)
once      (o)
sa_su     (s)?
3
from hour:minute? 00:00
to hour:minute? 00:16
how many watt it will have? 240
q quit
a add actual to total (using operator +)
c annual consumption and cost (price for one kWh: 30.00 ct/kWh; calling function sum)
o output actual (using operator <<)
t output total (using operator <<)
u add consumption according to frequency of use (call functions add_consumption)
z set actual to zeros (call function zeros)
>> o
year: 2024
day 0: Monday
 0:00       0.00       0.00       0.00       0.00
 1:00       0.00       0.00       0.00       0.00
 2:00       0.00       0.00       0.00       0.00
```

```
day 1: Tuesday
 0:00       0.00       0.00       0.00       0.00
 1:00       0.00       0.00       0.00      16.67
 2:00      25.00      25.00       5.00       0.00

day 2: Wednesday
 0:00       0.00       0.00       0.00       0.00
 1:00       0.00       0.00       0.00       0.00
 2:00       0.00       0.00       0.00       0.00

day 3: Thursday
 0:00      60.00       4.00       0.00       0.00
 1:00       0.00       0.00       0.00       0.00
 2:00       0.00       0.00       0.00       0.00

day 4: Friday
 0:00       0.00       0.00       0.00       0.00
 1:00       0.00       0.00       0.00       0.00
 2:00       0.00       0.00       0.00       0.00

day 5: Saturday
 0:00       0.00       0.00       0.00       0.00
 1:00       0.00       0.00       0.00       0.00
 2:00       0.00       0.00       0.00       0.00

day 6: Sunday
 0:00       0.00       0.00       0.00       0.00
 1:00       0.00       0.00       0.00       0.00
 2:00       0.00       0.00       0.00       0.00

day 7: Monday
 0:00       0.00       0.00       0.00       0.00
 1:00       0.00       0.00       0.00       0.00
 2:00       0.00       0.00       0.00       0.00

day 8: Tuesday
 0:00       0.00       0.00       0.00       0.00
 1:00       0.00       0.00       0.00      16.67
 2:00      25.00      25.00       5.00       0.00

day 9: Wednesday
 0:00       0.00       0.00       0.00       0.00
 1:00       0.00       0.00       0.00       0.00
 2:00       0.00       0.00       0.00       0.00

q quit
a add actual to total (using operator +)
c annual consumption and cost (price for one kWh: 30.00 ct/kWh; calling function sum)
o output actual (using operator <<)
t output total (using operator <<)
u add consumption according to frequency of use (call functions add_consumption)
z set actual to zeros (call function zeros)
>> u
how often it will be used?
Monday    (0)
Tuesday   (1)
Wednesday (2)
Thursday  (3)
Friday    (4)
Saturday  (5)
Sunday    (6)
daily     (d)
mo_fr     (m)
once      (o)
sa_su     (s)?
m
from hour:minute? 00:30
to hour:minute? 01:15
how many watt it will have? 60
q quit
a add actual to total (using operator +)
c annual consumption and cost (price for one kWh: 30.00 ct/kWh; calling function sum)
o output actual (using operator <<)
t output total (using operator <<)
u add consumption according to frequency of use (call functions add_consumption)
z set actual to zeros (call function zeros)
```

```
>> o
year: 2024
day 0: Monday
 0:00       0.00       0.00      15.00      15.00
 1:00      15.00       0.00       0.00       0.00
 2:00       0.00       0.00       0.00       0.00

day 1: Tuesday
 0:00       0.00       0.00      15.00      15.00
 1:00      15.00       0.00       0.00      16.67
 2:00      25.00      25.00       5.00       0.00

day 2: Wednesday
 0:00       0.00       0.00      15.00      15.00
 1:00      15.00       0.00       0.00       0.00
 2:00       0.00       0.00       0.00       0.00

day 3: Thursday
 0:00      60.00       4.00      15.00      15.00
 1:00      15.00       0.00       0.00       0.00
 2:00       0.00       0.00       0.00       0.00

day 4: Friday
 0:00       0.00       0.00      15.00      15.00
 1:00      15.00       0.00       0.00       0.00
 2:00       0.00       0.00       0.00       0.00

day 5: Saturday
 0:00       0.00       0.00       0.00       0.00
 1:00       0.00       0.00       0.00       0.00
 2:00       0.00       0.00       0.00       0.00

day 6: Sunday
 0:00       0.00       0.00       0.00       0.00
 1:00       0.00       0.00       0.00       0.00
 2:00       0.00       0.00       0.00       0.00

day 7: Monday
 0:00       0.00       0.00      15.00      15.00
 1:00      15.00       0.00       0.00       0.00
 2:00       0.00       0.00       0.00       0.00

day 8: Tuesday
 0:00       0.00       0.00      15.00      15.00
 1:00      15.00       0.00       0.00      16.67
 2:00      25.00      25.00       5.00       0.00

day 9: Wednesday
 0:00       0.00       0.00      15.00      15.00
 1:00      15.00       0.00       0.00       0.00
 2:00       0.00       0.00       0.00       0.00

q quit
a add actual to total (using operator +)
c annual consumption and cost (price for one kWh: 30.00 ct/kWh; calling function sum)
o output actual (using operator <<)
t output total (using operator <<)
u add consumption according to frequency of use (call functions add_consumption)
z set actual to zeros (call function zeros)
>> c
sum actual = 567.33 Watt (costs: 0.17 EUR)
sum total = 770.00 Watt (costs: 0.23 EUR)
q quit
a add actual to total (using operator +)
c annual consumption and cost (price for one kWh: 30.00 ct/kWh; calling function sum)
o output actual (using operator <<)
t output total (using operator <<)
u add consumption according to frequency of use (call functions add_consumption)
z set actual to zeros (call function zeros)
>> a
q quit
a add actual to total (using operator +)
c annual consumption and cost (price for one kWh: 30.00 ct/kWh; calling function sum)
o output actual (using operator <<)
t output total (using operator <<)
u add consumption according to frequency of use (call functions add_consumption)
```

```
z set actual to zeros (call function zeros)
>> c
sum actual = 567.33 Watt (costs: 0.17 EUR)
sum total = 1337.33 Watt (costs: 0.40 EUR)
q quit
a add actual to total (using operator +)
c annual consumption and cost (price for one kWh: 30.00 ct/kWh; calling function sum)
o output actual (using operator <<)
t output total (using operator <<)
u add consumption according to frequency of use (call functions add_consumption)
z set actual to zeros (call function zeros)
>> t
year: 2024
day 0: Monday
 0:00       0.00      10.00      30.00      30.00
 1:00      30.00      15.00       7.00       0.00
 2:00       0.00       0.00       0.00       0.00

day 1: Tuesday
 0:00       0.00      10.00      30.00      30.00
 1:00      30.00      15.00       7.00      16.67
 2:00      25.00      25.00       5.00       0.00

day 2: Wednesday
 0:00       0.00      10.00      30.00      30.00
 1:00      30.00      15.00       7.00       0.00
 2:00       0.00       0.00       0.00       0.00

day 3: Thursday
 0:00      60.00      14.00      30.00      30.00
 1:00      30.00      15.00       7.00       0.00
 2:00       0.00       0.00       0.00       0.00

day 4: Friday
 0:00       0.00      10.00      30.00      30.00
 1:00      30.00      15.00       7.00       0.00
 2:00       0.00       0.00       0.00       0.00

day 5: Saturday
 0:00       0.00      10.00      15.00      15.00
 1:00      15.00      15.00       7.00       0.00
 2:00       0.00       0.00       0.00       0.00

day 6: Sunday
 0:00       0.00      10.00      15.00      15.00
 1:00      15.00      15.00       7.00       0.00
 2:00       0.00       0.00       0.00       0.00

day 7: Monday
 0:00       0.00      10.00      30.00      30.00
 1:00      30.00      15.00       7.00       0.00
 2:00       0.00       0.00       0.00       0.00

day 8: Tuesday
 0:00       0.00      10.00      30.00      30.00
 1:00      30.00      15.00       7.00      16.67
 2:00      25.00      25.00       5.00       0.00

day 9: Wednesday
 0:00       0.00      10.00      30.00      30.00
 1:00      30.00      15.00       7.00       0.00
 2:00       0.00       0.00       0.00       0.00

q quit
a add actual to total (using operator +)
c annual consumption and cost (price for one kWh: 30.00 ct/kWh; calling function sum)
o output actual (using operator <<)
t output total (using operator <<)
u add consumption according to frequency of use (call functions add_consumption)
z set actual to zeros (call function zeros)
>> u
how often it will be used?
Monday    (0)
Tuesday   (1)
Wednesday (2)
Thursday  (3)
```

```
Friday    (4)
Saturday  (5)
Sunday    (6)
daily     (d)
mo_fr     (m)
once      (o)
sa_su     (s)?
o
on which day? 6
from hour:minute? 01:05
to hour:minute? 01:31
how many watt it will have? 12000
q quit
a add actual to total (using operator +)
c annual consumption and cost (price for one kWh: 30.00 ct/kWh; calling function sum)
o output actual (using operator <<)
t output total (using operator <<)
u add consumption according to frequency of use (call functions add_consumption)
z set actual to zeros (call function zeros)
>> o
year: 2024
day 0: Monday
 0:00       0.00       0.00      15.00      15.00
 1:00      15.00       0.00       0.00       0.00
 2:00       0.00       0.00       0.00       0.00

day 1: Tuesday
 0:00       0.00       0.00      15.00      15.00
 1:00      15.00       0.00       0.00      16.67
 2:00      25.00      25.00       5.00       0.00

day 2: Wednesday
 0:00       0.00       0.00      15.00      15.00
 1:00      15.00       0.00       0.00       0.00
 2:00       0.00       0.00       0.00       0.00

day 3: Thursday
 0:00      60.00       4.00      15.00      15.00
 1:00      15.00       0.00       0.00       0.00
 2:00       0.00       0.00       0.00       0.00

day 4: Friday
 0:00       0.00       0.00      15.00      15.00
 1:00      15.00       0.00       0.00       0.00
 2:00       0.00       0.00       0.00       0.00

day 5: Saturday
 0:00       0.00       0.00       0.00       0.00
 1:00       0.00       0.00       0.00       0.00
 2:00       0.00       0.00       0.00       0.00

day 6: Sunday
 0:00       0.00       0.00       0.00       0.00
 1:00    2000.00    3000.00     200.00       0.00
 2:00       0.00       0.00       0.00       0.00

day 7: Monday
 0:00       0.00       0.00      15.00      15.00
 1:00      15.00       0.00       0.00       0.00
 2:00       0.00       0.00       0.00       0.00

day 8: Tuesday
 0:00       0.00       0.00      15.00      15.00
 1:00      15.00       0.00       0.00      16.67
 2:00      25.00      25.00       5.00       0.00

day 9: Wednesday
 0:00       0.00       0.00      15.00      15.00
 1:00      15.00       0.00       0.00       0.00
 2:00       0.00       0.00       0.00       0.00

q quit
a add actual to total (using operator +)
c annual consumption and cost (price for one kWh: 30.00 ct/kWh; calling function sum)
o output actual (using operator <<)
t output total (using operator <<)
```

```
u add consumption according to frequency of use (call functions add_consumption)
z set actual to zeros (call function zeros)
>> c
sum actual = 5767.33 Watt (costs: 1.73 EUR)
sum total = 1337.33 Watt (costs: 0.40 EUR)
q quit
a add actual to total (using operator +)
c annual consumption and cost (price for one kWh: 30.00 ct/kWh; calling function sum)
o output actual (using operator <<)
t output total (using operator <<)
u add consumption according to frequency of use (call functions add_consumption)
z set actual to zeros (call function zeros)
>>
```

Beispiel Programmlauf 2 für 365 Tage und 96 Viertelstundenwerte/Example Program Run 2 for 365 days and 96 quarter hours

```
YEARLY CONSUMPTION QUARTER HOUR
q quit
a add actual to total (using operator +)
c annual consumption and cost (price for one kWh: 30.00 ct/kWh; calling function sum)
o output actual (using operator <<)
t output total (using operator <<)
u add consumption according to frequency of use (call functions add_consumption)
z set actual to zeros (call function zeros)
>> c
sum actual = 0.00 Watt (costs: 0.00 EUR)
sum total = 0.00 Watt (costs: 0.00 EUR)
q quit
a add actual to total (using operator +)
c annual consumption and cost (price for one kWh: 30.00 ct/kWh; calling function sum)
o output actual (using operator <<)
t output total (using operator <<)
u add consumption according to frequency of use (call functions add_consumption)
z set actual to zeros (call function zeros)
>> u
how often it will be used?
Monday    (0)
Tuesday   (1)
Wednesday (2)
Thursday  (3)
Friday    (4)
Saturday  (5)
Sunday    (6)
daily     (d)
mo_fr     (m)
once      (o)
sa_su     (s)?
d
from hour:minute? 00:00
to hour:minute? 24:00
how many watt it will have? 10
q quit
a add actual to total (using operator +)
c annual consumption and cost (price for one kWh: 30.00 ct/kWh; calling function sum)
o output actual (using operator <<)
t output total (using operator <<)
u add consumption according to frequency of use (call functions add_consumption)
z set actual to zeros (call function zeros)
>> c
sum actual = 87600.00 Watt (costs: 26.28 EUR)
sum total = 0.00 Watt (costs: 0.00 EUR)
q quit
a add actual to total (using operator +)
c annual consumption and cost (price for one kWh: 30.00 ct/kWh; calling function sum)
o output actual (using operator <<)
t output total (using operator <<)
u add consumption according to frequency of use (call functions add_consumption)
z set actual to zeros (call function zeros)
>> a
q quit
a add actual to total (using operator +)
c annual consumption and cost (price for one kWh: 30.00 ct/kWh; calling function sum)
o output actual (using operator <<)
t output total (using operator <<)
u add consumption according to frequency of use (call functions add_consumption)
z set actual to zeros (call function zeros)
>> c
sum actual = 87600.00 Watt (costs: 26.28 EUR)
sum total = 87600.00 Watt (costs: 26.28 EUR)
q quit
a add actual to total (using operator +)
c annual consumption and cost (price for one kWh: 30.00 ct/kWh; calling function sum)
o output actual (using operator <<)
t output total (using operator <<)
u add consumption according to frequency of use (call functions add_consumption)
z set actual to zeros (call function zeros)
>> z
q quit
a add actual to total (using operator +)
c annual consumption and cost (price for one kWh: 30.00 ct/kWh; calling function sum)
o output actual (using operator <<)
```

```
t output total (using operator <<)
u add consumption according to frequency of use (call functions add_consumption)
z set actual to zeros (call function zeros)
>> c
sum actual = 0.00 Watt (costs: 0.00 EUR)
sum total = 87600.00 Watt (costs: 26.28 EUR)
q quit
a add actual to total (using operator +)
c annual consumption and cost (price for one kWh: 30.00 ct/kWh; calling function sum)
o output actual (using operator <<)
t output total (using operator <<)
u add consumption according to frequency of use (call functions add_consumption)
z set actual to zeros (call function zeros)
>> u
how often it will be used?
Monday    (0)
Tuesday   (1)
Wednesday (2)
Thursday  (3)
Friday    (4)
Saturday  (5)
Sunday    (6)
daily     (d)
mo_fr     (m)
once      (o)
sa_su     (s)?
5
from hour:minute? 09:00
to hour:minute? 11:00
how many watt it will have? 2000
q quit
a add actual to total (using operator +)
c annual consumption and cost (price for one kWh: 30.00 ct/kWh; calling function sum)
o output actual (using operator <<)
t output total (using operator <<)
u add consumption according to frequency of use (call functions add_consumption)
z set actual to zeros (call function zeros)
>> c
sum actual = 208000.00 Watt (costs: 62.40 EUR)
sum total = 87600.00 Watt (costs: 26.28 EUR)
q quit
a add actual to total (using operator +)
c annual consumption and cost (price for one kWh: 30.00 ct/kWh; calling function sum)
o output actual (using operator <<)
t output total (using operator <<)
u add consumption according to frequency of use (call functions add_consumption)
z set actual to zeros (call function zeros)
>> a
q quit
a add actual to total (using operator +)
c annual consumption and cost (price for one kWh: 30.00 ct/kWh; calling function sum)
o output actual (using operator <<)
t output total (using operator <<)
u add consumption according to frequency of use (call functions add_consumption)
z set actual to zeros (call function zeros)
>> c
sum actual = 208000.00 Watt (costs: 62.40 EUR)
sum total = 295600.00 Watt (costs: 88.68 EUR)
q quit
a add actual to total (using operator +)
c annual consumption and cost (price for one kWh: 30.00 ct/kWh; calling function sum)
o output actual (using operator <<)
t output total (using operator <<)
u add consumption according to frequency of use (call functions add_consumption)
z set actual to zeros (call function zeros)
>> z
q quit
a add actual to total (using operator +)
c annual consumption and cost (price for one kWh: 30.00 ct/kWh; calling function sum)
o output actual (using operator <<)
t output total (using operator <<)
u add consumption according to frequency of use (call functions add_consumption)
z set actual to zeros (call function zeros)
>> c
sum actual = 0.00 Watt (costs: 0.00 EUR)
sum total = 295600.00 Watt (costs: 88.68 EUR)
```

```
q quit
a add actual to total (using operator +)
c annual consumption and cost (price for one kWh: 30.00 ct/kWh; calling function sum)
o output actual (using operator <<)
t output total (using operator <<)
u add consumption according to frequency of use (call functions add_consumption)
z set actual to zeros (call function zeros)
>> u
how often it will be used?
Monday    (0)
Tuesday   (1)
Wednesday (2)
Thursday  (3)
Friday    (4)
Saturday  (5)
Sunday    (6)
daily     (d)
mo_fr     (m)
once      (o)
sa_su     (s)?
m
from hour:minute? 08:30
to hour:minute? 17:00
how many watt it will have? 200
q quit
a add actual to total (using operator +)
c annual consumption and cost (price for one kWh: 30.00 ct/kWh; calling function sum)
o output actual (using operator <<)
t output total (using operator <<)
u add consumption according to frequency of use (call functions add_consumption)
z set actual to zeros (call function zeros)
>> c
sum actual = 443700.00 Watt (costs: 133.11 EUR)
sum total = 295600.00 Watt (costs: 88.68 EUR)
q quit
a add actual to total (using operator +)
c annual consumption and cost (price for one kWh: 30.00 ct/kWh; calling function sum)
o output actual (using operator <<)
t output total (using operator <<)
u add consumption according to frequency of use (call functions add_consumption)
z set actual to zeros (call function zeros)
>> u
how often it will be used?
Monday    (0)
Tuesday   (1)
Wednesday (2)
Thursday  (3)
Friday    (4)
Saturday  (5)
Sunday    (6)
daily     (d)
mo_fr     (m)
once      (o)
sa_su     (s)?
m
from hour:minute? 00:00
to hour:minute? 08:30
how many watt it will have? 0.5
q quit
a add actual to total (using operator +)
c annual consumption and cost (price for one kWh: 30.00 ct/kWh; calling function sum)
o output actual (using operator <<)
t output total (using operator <<)
u add consumption according to frequency of use (call functions add_consumption)
z set actual to zeros (call function zeros)
>> c
sum actual = 444809.25 Watt (costs: 133.44 EUR)
sum total = 295600.00 Watt (costs: 88.68 EUR)
q quit
a add actual to total (using operator +)
c annual consumption and cost (price for one kWh: 30.00 ct/kWh; calling function sum)
o output actual (using operator <<)
t output total (using operator <<)
u add consumption according to frequency of use (call functions add_consumption)
z set actual to zeros (call function zeros)
>> u
```

```
how often it will be used?
Monday    (0)
Tuesday   (1)
Wednesday (2)
Thursday  (3)
Friday    (4)
Saturday  (5)
Sunday    (6)
daily     (d)
mo_fr     (m)
once      (o)
sa_su     (s)?
m
from hour:minute? 17:00
to hour:minute? 24:00
how many watt it will have? 0.5
q quit
a add actual to total (using operator +)
c annual consumption and cost (price for one kWh: 30.00 ct/kWh; calling function sum)
o output actual (using operator <<)
t output total (using operator <<)
u add consumption according to frequency of use (call functions add_consumption)
z set actual to zeros (call function zeros)
>> c
sum actual = 445722.75 Watt (costs: 133.72 EUR)
sum total = 295600.00 Watt (costs: 88.68 EUR)
q quit
a add actual to total (using operator +)
c annual consumption and cost (price for one kWh: 30.00 ct/kWh; calling function sum)
o output actual (using operator <<)
t output total (using operator <<)
u add consumption according to frequency of use (call functions add_consumption)
z set actual to zeros (call function zeros)
>> a
q quit
a add actual to total (using operator +)
c annual consumption and cost (price for one kWh: 30.00 ct/kWh; calling function sum)
o output actual (using operator <<)
t output total (using operator <<)
u add consumption according to frequency of use (call functions add_consumption)
z set actual to zeros (call function zeros)
>> c
sum actual = 445722.75 Watt (costs: 133.72 EUR)
sum total = 741322.75 Watt (costs: 222.40 EUR)
q quit
a add actual to total (using operator +)
c annual consumption and cost (price for one kWh: 30.00 ct/kWh; calling function sum)
o output actual (using operator <<)
t output total (using operator <<)
u add consumption according to frequency of use (call functions add_consumption)
z set actual to zeros (call function zeros)
>>
```

Zuletzt geändert: Freitag, 3. November 2023, 21:21

◄ A2 Upload Teil 1/Part 1

Direkt zu:

A3 Upload Teil 1/Part 1 ►