PRO1UEG*x* **Übungen zu Einführung in die Programmierung, Algorithmen und Datenstrukturen** **WS 25/26, Übung 2**

Abgabetermin: Mi, 22.10.2025

☒ **Gr. 1,** Dr. D. Auer **Name** Klemens Danner **Aufwand in h** 4

☐ **Gr. 2,** Dr. G. Kronberger

☐ **Gr. 3,** Dr. S. Wagner **Punkte** _____ **Kurzzeichen Tutor*in / Übungsleiter*in** ____ / ____

1. Einnahmen/Ausgaben-Rechnung (Wdh.) (8 Punkte)

Entwickeln Sie ein Pascal-Programm, das eine Folge ganzer Zahlen einliest und die Summe der positiven Zahlen (*Einnahmen*) sowie die Summe der negativen Zahlen (*Ausgaben*) berechnet. Wird der Wert 0 eingegeben, soll das Programm das Einlesen beenden und die beiden Summen sowie die Gesamtsumme (= Einnahmen – Ausgaben) ausgeben.

Beispiele:

1. Eingabe: `10 20 -30 40 -50 60 70 80 -90 0`
   Ausgabe:
   ```
   Einnahmen:   280
   Ausgaben:   -170
   Gesamt:      110
   ```
2. Eingabe: `0`
   Ausgabe:
   ```
   Einnahmen:     0
   Ausgaben:      0
   Gesamt:        0
   ```

2. Tägliche Zeiterfassung (8 Punkte)

Entwickeln Sie ein Pascal-Programm, das die tägliche Arbeitszeit in Form ganzzahliger Werte für Stunden und Minuten einliest und aus ggf. geleisteten Überstunden den Anspruch auf Zeitausgleich berechnet und ausgibt. Für die tägliche Arbeitszeit gelten (nur hier) folgende Regeln:

- Die ersten acht Stunden sind Normalarbeitszeit, es fallen keine Überstunden an.
- Die 9. und 10. Stunde kann als Zeitausgleich in Anspruch genommen werden.
- Die 11. und 12. Stunde kann mit Faktor 1.5 als Zeitausgleich in Anspruch genommen werden.
- Täglich dürfen höchstens 12 Stunden gearbeitet werden.

Beispiele:

1. Eingabe: `8 30`
   Ausgabe: `Anspruch auf Zeitausgleich: 0.50 Stunden`
2. Eingabe: `11 00`
   Ausgabe: `Anspruch auf Zeitausgleich: 3.50 Stunden`
3. Eingabe: `13 00`
   Ausgabe: `Taegliche Hoechstarbeitszeit ueberschritten`

3. Multiplikationstabelle (8 Punkte)

Implementieren Sie ein Pascal-Programm, das fortlaufend zwei ganzzahlige Werte n und m einliest und für eine gültige Eingabe (Wertebereich $1 \leq n, m \leq 20$) eine Multiplikationstabelle ausgibt. Wird $n = 0$ eingegeben, soll das Pascal-Programm beendet werden. Eine Multiplikationstabelle mit n Zeilen und m Spalten enthält in der Zeile i und Spalte j den Wert $i \cdot j$.

Beispiel für $n = 5$ und $m = 10$:

```
 1   2   3   4   5   6   7   8   9  10
 2   4   6   8  10  12  14  16  18  20
 3   6   9  12  15  18  21  24  27  30
 4   8  12  16  20  24  28  32  36  40
 5  10  15  20  25  30  35  40  45  50
```

Hinweise:

1. Geben Sie für alle Ihre Lösungen immer eine „Lösungsidee“ an.
2. Dokumentieren und kommentieren Sie Ihre Pascal-Programme.
3. Geben Sie immer auch Testfälle ab, an denen man erkennen kann, dass Ihr Pascal-Programm funktioniert, und dass es auch in Fehlersituationen entsprechend reagiert.

2.1.1 Kopie von Lösungsidee aus Übung 1

Variablen werden deklariert. Mittels Schleife werden solange Werte eingelesen, bis der eingelesene Wert gleich 0 ist. Alle Werte werden in der Schleife überprüft ob sie größer als 0 sind oder nicht und dann zu den jeweiligen Variablen addiert. Anschließend werden die Werte der Variablen ausgegeben.

2.1.2 Quellcode

```
1   PROGRAM EinnahmenAusgaben;
2
3     VAR
4     value, totalpos, totalneg: INTEGER;
5
6   BEGIN (*EinnahmenAusgaben*)
7     value := 1;
8     totalneg := 0;
9     totalpos := 0;
10
11    while value <> 0 do // Solange value ungleich 0 wird eingelesen
12    BEGIN
13      ReadLn(value);
14
15      //Zuweisungen je nach Vorzeichen
16      if value > 0 THEN
17        totalpos := totalpos + value
18      else
19        totalneg := totalneg + value
20    END;
21
22  //Ausgabe
23  WriteLn('Einnahmen: ', totalpos);
24  WriteLn('Ausgaben: ', totalneg);
25  WriteLn('Gesamt: ', (totalpos + totalneg));
26
27  END. (*EinnahmenAusgaben*)
```

2.1.3 Tests

Zahlen	Beschreibung	Ausgabe
10 20 -30 40 -50 60 70 80 -90 0	Positive und negative Zahlen	10 20 -30 40 -50 60 70 80 -90 0 Einnahmen: 280 Ausgaben: -170 Gesamt: 110
0	0 testen	0 Einnahmen: 0 Ausgaben: 0 Gesamt: 0

2.2.1 Lösungsidee

Die Anzahl der Minuten und Stunden werden eingelesen und den jeweiligen Variablen zugeordnet. Dann werden die Stunden und Minuten in eine Dezimalzahl mit der Einheit Stunden umgerechnet.

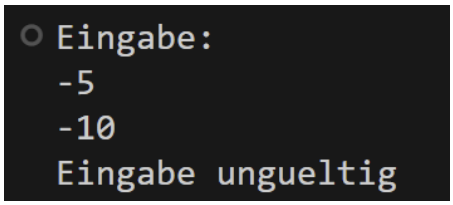
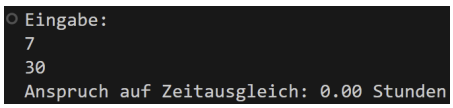
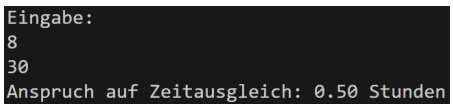
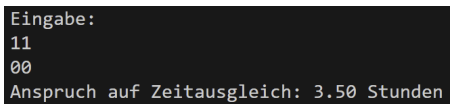
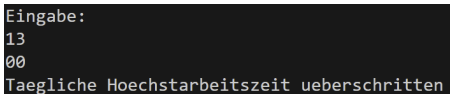
1. Wenn die Zahl kleiner als 0 ist, wird eine Fehlermeldung ausgegeben. Wenn die Zahl größer als 12 ist, wird ausgegeben, dass die Höchstarbeitszeit überschritten wurde.
2. Ansonsten: Von dieser real Zahl wird die Zahl 8.0 abgezogen. Wenn der Wert kleiner gleich 0 ist, soll 0 ausgegeben werden. Wenn die Differenz kleiner oder gleich 2 ist, soll der Wert als Zeitausgleich direkt ausgegeben werden.
3. Wenn der Wert größer als 2 ist, soll von der Differenz 2 abgezogen werden und die Zahl mit 1.5 multipliziert werden, dann wieder 2 addiert und dann ausgegeben.

2.2.2 Quellcode

```
PROGRAM Zeiterfassung;
  VAR
    min: INTEGER;
    hour: INTEGER;
    workTime: REAL;
    overWorkTime: REAL; //Gesamtzeit - Normalzeit (8h)
    compTimeOff: REAL; //Zeitausgleich
BEGIN (* Zeiterfassung *)
  //Einlesen der Arbeitszeit
  WriteLn('Eingabe: ');
  ReadLn(hour, min);
  //Berechnung der Gesamtzeit und Überstunden
  workTime := hour + (min/60);
  overWorkTime := workTime - 8;
  //Abfragen
  IF (workTime < 0) THEN BEGIN
    WriteLn('Eingabe ungueltig');
  END ELSE
    IF (workTime > 12) THEN BEGIN
      WriteLn('Taegliche Hoechstarbeitszeit ueberschritten');
    END ELSE
      IF (overWorkTime <= 0) THEN BEGIN //weniger oder genau 8 Stunden
gearbeitet
        compTimeOff := 0;
        WriteLn('Anspruch auf Zeitausgleich: ', compTimeOff:2:2, '
Stunden');
      END ELSE
        IF (overWorkTime > 0) AND (overWorkTime <=2) THEN BEGIN //zw. 0 und
2 Überstunden
          compTimeOff := overWorkTime;
          WriteLn('Anspruch auf Zeitausgleich: ', compTimeOff:2:2, '
Stunden');
        END ELSE
          IF (overWorkTime >= 2) THEN BEGIN //mehr als 2 Überstunden --> ab
11. Stunde *1.5
            compTimeOff := ((overWorkTime - 2)*1.5)+2;
            WriteLn('Anspruch auf Zeitausgleich: ', compTimeOff:2:2, '
Stunden');
          END; (* IF *)
END. (* Zeiterfassung *)
```

2.2.3 Tests

Es werden folgende Werte getestet:

Stunden	Minuten	Testbeschreibung	Testergebnis
-5	-10	Negative Eingaben (führen zu Fehlermeldung)	
7	30	Eingaben von 0 - 8 Stunden, sollen zu 0 Stunden Zeitausgleich führen	
8	30	Eingaben von 8 - 10 Stunden Sollen zu (Eingabe - 8) Stunden Zeitausgleich führen	
11	00	Eingaben von 10-12 Stunden sollen zu ((Eingabe - 10)*1.5 +2) Stunden Zeitausgleich führen	
13	00	Eingaben über 12 Stunden geben eine Meldung aus, dass die maximale Arbeitszeit überschritten wurde	

2.3.1 Lösungsidee

Solange in n ungleich 0 ist, wird das Folgende mittels while-Schleife wiederholt:
Ein Wert für n und einer für m wird eingelesen und den Variablen zugewiesen.
Dann werden zwei for-Schleifen verschachtelt implementiert. Die äußere Schleife zählt die Zeilen, von 1 bis n, der Index ist i. Die innere Schleife zählt die Spalten von 1 bis m mit dem Index j. Bei jedem Durchlauf wird das Produkt i*j ausgegeben. Um Abstand zwischen den Produkten zu gewährleisten, wird eine Mindestbreite bei der Ausgabe festgelegt.
Nach der inneren Schleife muss ein Zeilenumbruch durchgeführt werden.

2.3.2 Quelltext

```
PROGRAM Multiplikationstabelle;
  VAR
    n, m, i, j: INTEGER;

BEGIN
  //Initialisierung
  n := 1;
  m := 1;

  while n <> 0 do BEGIN

    Write('Zeilen: ');
    ReadLn(n);

    IF n <> 0 THEN BEGIN //Bei Eingabe von n=0 direkt abbrechen
      Write('Spalten: ');
      ReadLn(m);

      IF (n > 0) AND (m > 0) THEN BEGIN // Negative Werte ausschließen, Programm weiterführen

        for i := 1 to n do BEGIN

          for j := 1 to m do BEGIN

            Write((i*j):5)

          END; (*innere Zählschleife*)

          WriteLn
        END; (*äußere Zählschleife*)

      END (*if-Abfrage, neg. Werte ausschließen*) ELSE
        Write('Bitte nur positive ganze Zahlen eingeben.'); //Fehlermeldung ausgeben
        WriteLn;

    END (*if-Abfrage, bei n=0 abbrechen*)
  END (*While*)
END.
```

2.3.3 Tests

n	m	Beschreibung	Ausgabe
5	10	Positive ganze Zahlen	Zeilen: 5 Spalten: 10 1 2 3 4 5 6 7 8 9 10 2 4 6 8 10 12 14 16 18 20 3 6 9 12 15 18 21 24 27 30 4 8 12 16 20 24 28 32 36 40 5 10 15 20 25 30 35 40 45 50
-5	10	Eingabe einer negativen Zahl	Zeilen: -5 Spalten: 10 Bitte nur positive ganze Zahlen eingeben.
0	x	Eingabe von n=0, Programm soll direkt beendet werden	Zeilen: 0 Heap dump by heaptrc unit of C:\Users\kleme\OneDrive 0 memory blocks allocated : 0/0 0 memory blocks freed : 0/0 0 unfreed memory blocks : 0 True heap size : 65536 (160 used in System startup) True free heap : 65376