PRG1x & ADE1x	**Üb. zu Einf. in die Programmierung und Element. Alg. u. Datenstrukt.**	**WS 20/21, Übung 3**

Abgabetermin: Mi, 28. 10. 2020

☐ ~~**Gr. 1, Dr. D. Auer**~~
✓ **Gr. 2, Dr. G. Kronberger**
☐ ~~**Gr. 3, Dr. S. Wagner**~~

Name ANGELOS ANGELIS **Aufwand in h** 12

Punkte _____ **Kurzzeichen Tutor / Übungsleiter** _____ / _____

1. Varianz (6 Punkte)

Entwickeln Sie ein Pascal-Programm, das eine Folge von reellen Zahlen einliest und in einem Feld speichert. Die Eingabe der Zahlenfolge wird mit 0 beendet. Ihr Programm soll das arithmetische Mittel und die Varianz für die Zahlenfolge berechnen und ausgeben. Verwenden Sie den Datentyp `RealArray` um die Zahlen zu speichern. Sie können davon ausgehen, dass maximal 100 Zahlen eingegeben werden.

```
TYPE RealArray = ARRAY[1..100] OF REAL.
```

Zur Erinnerung: Die Varianz für eine Zahlenfolge $(x_i)_{i=1..N}$ mit dem Mittelwert $\bar{x}$ kann folgendermaßen berechnet werden:

$$\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})^2$$

```
Eingabe: 4 3 4 4 5 0
Average: 4.0, Variance: 0.50
```

2. Terminüberschneidungen (12 Punkte)

Entwickeln Sie ein Pascal-Programm, das eine Folge von Terminen für einen einzelnen Tag einliest und überprüft ob es Überschneidungen gibt. Falls zumindest eine Überschneidung gefunden wird sollen die beiden Termine ausgegeben werden, die sich am stärksten überschneiden.

Für jeden Termin werden zuerst der Betreff und dann die Startzeit sowie die Endzeit eingegeben. Die Eingabe wird mit einem leeren Titel beendet. Der Betreff darf Leerzeichen aber keine Umbrüche enthalten. Die Eingabe von Start und Endzeitpunkt erfolgt als Tupel von vier Zahlen und kann mit `ReadLn(startHour, startMinute, endHour, endMinute)` eingelesen werden.

Ihr Programm muss strukturierte Datentypen für Termine (`Appointment`) und Zeiten (`Time`) definieren und verwenden.

Sie können davon ausgehen, dass die Termine nach Startzeit sortiert eingegeben werden und Termine nur auf Minuten genau geplant werden. Bei mehreren gleich langen Überschneidungen soll die frühere Überschneidung ausgegeben werden.

Beispiel:

```
GRI
 9 00   10 30
FH>>next
12 00   17 30
ADE
13 00   14 35
PRG
14 40   16 15

Conflict FH>>next and ADE (95 min)
```

3. Caesar-Chiffre (6 Punkte)

Implementieren Sie ein Programm das eine Eingabe mithilfe der Caesar-Chiffre ver- und entschlüsselt. Bei der Caesar-Chiffre wird jeder Buchstabe in der Eingabe durch einen Buchstaben ersetzt, der im Alphabet um n Stellen versetzt folgt. Implementieren Sie den speziellen Fall bei dem eine Verschiebung um 13 Stellen erfolgt (Rot-13 Algorithmus). Dabei ergibt sich folgende Codierungstabelle:

ABCDEFGHIJKLMNOPQRSTUVWXYZ

NOPQRSTUVWXYZABCDEFGHIJKLM

Sie können davon ausgehen, dass der Klartext nur Großbuchstaben, Zahlen und Satzzeichen enthält; Zahlen und Satzzeichen sollen aber nicht verschlüsselt werden!

Beispiel:

```
Eingabe: SAPERE AUDE!
Ausgabe: FNCRER NHQR!
```

1. Varianz

Lösungsidee:
Erst mal muss man den Durchschnitt von den vom User angegebene Zahlen berechnen. Das habe ich mit einer while Schleife gemacht. Um die Varianz dann zu berechnen muss man die angegebene Formel benutzen. Um das zu machen habe ich erstmal den Quotient ($(x_i - \bar{x})_2$) berechnet wobei hier auch der schon berechnete Durchschnitt benutzt wird. Anschließend muss nur der Quotient durch die Anzahl der angegebenen Zahlen minus 1 geteilt werden.

Quelltext:

```
PROGRAM varianz;
TYPE
RealArray = ARRAY[1..100] OF REAL;
VAR
Eingabe,s : String;
n,summe, quotient : REAL;
zahlenFolge : RealArray;
i,j,total : INTEGER;

BEGIN
  Read(n);
  Str(n:7:3,s);
  Eingabe := Concat(s, ' ');

  i := 1;
  summe := 0;
  total := 0;
  quotient := 0;

  while n <> 0 do BEGIN
    zahlenFolge[i] := n;
    summe := summe + n;
    Inc(i);
    Inc(total);
    Read(n);

    Str(n:7:3,s); // Convert n to String and save it to s (Dient nur zur Darstellung)
    Eingabe := Concat(Eingabe, s, ' '); //Verkette Eingabe, s und ' ' in Eingabe (Dient nur zur Dars
tellung)
  END;

  FOR j := 1 TO total DO BEGIN
    quotient := quotient + Sqr(zahlenFolge[j] - (summe / total));
  END;

  WriteLn('Eingabe: ', Eingabe);
  WriteLn('Average : ', (summe / total):7:3);
  WriteLn('Varianz : ', (quotient / (total - 1)):7:3)
END.
```

Testfälle:

1)

```
4 3 4 4 5 0
Eingabe:   4.000   3.000   4.000   4.000   5.000   0.000
Average :   4.000
Varianz :   0.500
```

2)

```
2 2 2 2 2 0
Eingabe:   2.000   2.000   2.000   2.000   2.000   0.000
Average :   2.000
Varianz :   0.000
```

3)

```
12 13 14 15 0
Eingabe:  12.000  13.000  14.000  15.000   0.000
Average :  13.500
Varianz :   1.667
```

2. Terminüberschneidung

Lösungsidee:

Eingabe: Meine Idee war es die Aufgabe mit einem Array vom Typ Record zu lösen. Das Record besteht dann vom Namen des Termins, Anfangs Stunde und Minute und End Stunde und Minute des Termins. Die Termine werden mit Hilfe einer For Schleife eingelesen.

Man geht ja laut der Aufgabe davon aus dass die Termine aufsteigend anhand der Start Stunde sortiert eingegeben werden. Daher wird es einfacher die Überschneidungen zu berechnen. Um die Überschneidung zu berechnen habe ich zwei for Schleifen Benutzt. Die erste Stellt sozusagen die Endstunden dar und die zweite stellt die Start Stunden dar. In den for Schleifen wird dann Überprüft ob es eine eindeutige Überschneidung bei den Stunden der Termine gibt. Falls ja wird dann überprüft welcher Termin zu erst Endet und anhand dem die Überschneidung berechnet(in Minuten).

Anschließend darauf wird dann überprüft ob es davor eine noch größere Überschneidung gab. Falls ja wird die wird die berechnete Überschneidung ignoriert und es werden die nächsten Termine überprüft. Falls nein wird die berechnete Überschneidung in der Variable größteüberschneidung gespiechert.

Wenn es bei den Stunden der Termine keine eindeutige Überschneidung gibt wird überprüft ob sich die Termine bei den Minuten überschneiden. Dann wird ähnlich die Überschneidung berechnet.

Quelltext:

```
PROGRAM terminueberschneidungen;
TYPE
AppointmentREC = RECORD
  APname : STRING;
  startHour, startMinute, endHour, endMinute : INTEGER;
END;
AppointmentArray = ARRAY[1..100] OF AppointmentREC;

VAR
appointments : AppointmentArray;
termin,ueberschneidung,groessteUeberschneidung,positionZ,positionX : INTEGER;
i,j,z,x : INTEGER;

BEGIN
  termin := 1;
  i := 1;
  ueberschneidung := 0;
  groessteUeberschneidung := 0;

  for i:=1 TO Length(appointments) do BEGIN//EINGABE
    WriteLn('Bitte Terminname fuer Termin ', termin,' eingaben (Leerzeichen falls eingabe zu ende)')
;
    ReadLn(appointments[i].APname);
    if (appointments[i].APname = ' ') then
      BREAK;
    WriteLn('Bitte startHour, startMinute, endHour, endMinute fuer Termin ', termin,' eingeben');
    ReadLn(appointments[i].startHour,appointments[i].startMinute,appointments[i].endHour,appointment
s[i].endMinute);
    Inc(termin);
  END;//EINGABE ENDE

  for z := 1 to termin-1 do BEGIN // z ist immer endhour
    for x := 1 to termin-1 do BEGIN// x ist immer starthour
      if (appointments[z].endHour > appointments[x].startHour) AND (x >= z+1)(*alle starthours die N
ACH endhour kommen*) then BEGIN //eindeutig dass sich bei den stunden etwas überschneidet
        if appointments[z].endhour > appointments[x].endHour then BEGIN
          ueberschneidung := appointments[x].endHour - appointments[x].startHour;
          ueberschneidung := ueberschneidung * 60;
          ueberschneidung := ueberschneidung + appointments[x].endMinute - appointments[x].startMinu
te;
        end;
        if appointments[z].endhour < appointments[x].endHour then BEGIN
          ueberschneidung := appointments[z].endHour - appointments[x].startHour;
          ueberschneidung := ueberschneidung * 60;
          ueberschneidung := ueberschneidung + appointments[z].endMinute - appointments[x].startMinu
te;
        end;
        if ueberschneidung > groessteUeberschneidung then BEGIN
          groessteUeberschneidung := ueberschneidung;
          positionZ := z;
          positionX := x;
        END;//if
      END;//eindeutig dass sich bei den stunden etwas überschneidet

      if (appointments[z].endHour = appointments[x].startHour) AND NOT (appointments[z].endMinute =a
ppointments[x].endMinute) then BEGIN //Nur minuten unterscheiden sich
```

```
        if appointments[z].endMinute > appointments[x].startMinute then BEGIN
          ueberschneidung := appointments[z].endMinute - appointments[x].startMinute;
        END;
        if appointments[z].endMinute < appointments[x].startMinute then BEGIN
          ueberschneidung := appointments[x].startMinute - appointments[z].endMinute;
        END;
        if ueberschneidung > groessteUeberschneidung then BEGIN
          groessteUeberschneidung := ueberschneidung;
          positionZ := z;
          positionX := x;
        END;//if
      END;//if //NUR MINUTEN UNTERSCHEIDEN SICH

    END;//for
  END;//for

  for j := 1 to termin - 1 do BEGIN//AUSGABE
    WriteLn(appointments[j].APname);
    WriteLn(appointments[j].startHour,':',appointments[j].startMinute,' - ',appointments[j].endHour,
':',appointments[j].endMinute);
  END;
  if groessteUeberschneidung = 0 then
    WriteLn('Es gibt keine Ueberschneidungen');
  if groessteUeberschneidung > 0 then
    WriteLn('Staerkste ueberschneidung an position: ', appointments[positionZ].APname,' und ',appoin
tments[positionX].APname,' um ', groessteUeberschneidung, ' minuten ');
END.
```

Testfälle

1)

```
GRI
9:0 - 10:30
FH>>next
12:0 - 17:30
ADE
13:0 - 14:35
PRG
14:40 - 16:15
Staerkste ueberschneidung an position: FH>>next und ADE um 95 minuten
```

2)

```
test 1
12:0 - 13:0
test 2
14:0 - 15:0
Es gibt keine Ueberschneidungen
```

3)

```
test 1
13:0 - 14:0
test 2
14:0 - 15:50
test 3
15:0 - 16:0
Staerkste ueberschneidung an position: test 2 und test 3 um 50 minuten
```

3. Caesar Chiffre

Erstmals wird der eingelesene String mithilfe einer for schleife zu einem char array konvertiert. Mithilfe dieses char arrays wird es jetzt einfacher die Buchstaben immer um 13 Plätze im Alphabet zu verschieben. Dann wird mithilfe der ASCII Tabelle überprüft ob die Buchstaben Sonderzeichen oder eine Leerzeile sind. Falls ja werden sie nicht geändert. Falls die Buchstaben sich Anhand der ASCII Tabelle zwischen Platz 65 und 90 befinden soll die Verschlüsselung (Findet ebenfalls mithilfe der ASCII Tabelle statt) stattfinden

Quelltext:

```
PROGRAM caesarChiffre;
TYPE
charArray = ARRAY[1..100] OF CHAR;
VAR
str : STRING;
eingabe,ausgabe : charArray;
i,j,z,x,p : INTEGER;
BEGIN
  Read(str);
  for i := 1 to Length(str) do BEGIN
    eingabe[i] := str[i];
  END;
  ausgabe := eingabe;

  for j := 1 to Length(ausgabe) do BEGIN
    if ((Ord(ausgabe[j]) > 47) AND (Ord(ausgabe[j]) < 58)) OR (Ord(ausgabe[j]) =
32) THEN BEGIN
      ausgabe[j] := ausgabe[j];
    END;
    if ((Ord(ausgabe[j]) > 64) AND (Ord(ausgabe[j]) < 91)) then BEGIN
      if (Ord('Z') - Ord(ausgabe[j])) >= 13 THEN BEGIN
        ausgabe[j] := CHAR(Ord(ausgabe[j])+(13));
```

```
      END
      else if (Ord('Z') - Ord(ausgabe[j])) < 13 THEN BEGIN
        x := Ord('Z') - Ord(ausgabe[j]);
        ausgabe[j] := CHAR(Ord('A')+(12-x));
      END;//else if
    END;//if
  END;//for

  for z := 1 to Length(ausgabe) do BEGIN
    Write(ausgabe[z])
  END;
  (*n:= CHAR(65);
  WriteLn(CHAR(Ord(n)+1))*)
END.
```

Testfälle:

1)

```
SAPERE AUDE!
FNCRER NHQR!
```

2)

```
ANGELOS ANGELIS
NATRYBF NATRYVF
```

3)

```
DANKE FÜRS KORRIGIEREN
QNAXR SÜEF XBEEVTVRERA
```