

☐ Gr. 1, Dr. D. Auer☐ Gr. 2, Dr. G. Kronberger

Gr. 3, Dr. H. Gruber

Name Andreas Roither Aufwand in h 5 h

Punkte \_\_\_\_\_ Kurzzeichen Tutor / Übungsleiter \_\_\_\_\_ / \_\_\_\_\_

**1. Telefonverzeichnis****(16 Punkte)**

Implementieren Sie ein elektronisches Telefonverzeichnis, welches es ermöglicht, Einträge zu speichern, abzurufen und zu löschen. Verwenden Sie dazu die folgenden Deklarationen:

```
CONST
  max = 10;
TYPE
  Entry = RECORD
    firstName: STRING[20];
    lastName:  STRING[30];
    phoneNumber: INTEGER;
  END; (*RECORD*)
  PhoneBook = ARRAY [1 .. max] OF Entry;
```

Das Telefonverzeichnis soll lückenlos gefüllt werden, auf eine Sortierung können Sie verzichten. Beim Löschen eines Eintrags ist die Lücke durch Verschieben der restlichen Einträge zu schließen. Zugriffe auf das Verzeichnis dürfen nur über die Prozeduren *AddEntry*, *DeleteEntry*, *SearchName*, *SearchNumber* und die Funktion *NrOfEntries* erfolgen (wählen Sie die Parameter passend):

- PROC. **AddEntry**(...);  
Erweitert das Verzeichnis um einen Eintrag. Bei einem Überlauf muss eine Fehlermeldung ausgegeben werden; das Verzeichnis darf in diesem Fall nicht verändert werden.
- PROC. **DeleteEntry**(...);  
Versucht, einen durch Vor- und Nachnamen gegebenen Eintrag zu entfernen. Ist dieser nicht vorhanden, so ist eine Fehlermeldung auszugeben.
- PROC. **SearchNumber**(...);  
Sucht einen Eintrag mit Vor- und Nachnamen. Ist kein solcher vorhanden, so ist eine Fehlermeldung auszugeben. Bei mehrfachen Einträgen sind alle auszugeben.
- PROC. **SearchName**(...);  
Sucht einen Eintrag nach der Telefonnummer. Ist kein Eintrag mit dieser Nummer vorhanden, so ist eine Fehlermeldung auszugeben.
- FUNC. **NrOfEntries**(...): INTEGER;  
Liefert die aktuelle Anzahl der Einträge.

**2. Feldverarbeitung mit offenen Feldparametern****(8 Punkte)**

Gegeben sind zwei beliebig große Felder *a1* und *a2*, die positive ganze Zahlen in aufsteigend sortierter Reihenfolge enthalten. Gesucht ist eine Pascal-Prozedur

```
PROC. Merge(a1, a2: ARRAY OF INT.; VAR a3: ARRAY OF INT.; VAR n3: INT.);
```

die ein aufsteigend sortiertes Feld *a3* liefert, das alle *n3* Zahlen enthält, die in *a1* oder in *a2* aber nicht in *a1* und *a2* vorkommen (vgl. *XOR*). Beachten Sie, dass in jedem Feld (also auch im Ergebnisfeld *a3*) Werte mehrfach vorkommen können. Im Fehlerfall (Feld *a3* würde überlaufen) soll *n3* auf -1 gestellt werden. *Beispiel*:

<i>a1</i> =	2	4	4	10	15	15
<i>a2</i> =	3	4	5	10		
<i>a3</i> =	2	3	5	15	15	

und *n3* = 5

# Übung 5

## Aufgabe 1

### Lösungsidee

Beim Telefonverzeichnis werden Procedures geschrieben die das Verwalten eines Telefonverzeichnisses vereinfachen. Dabei wird ein Typ Entry und ein Array verwendet. Mithilfe der Procedures wird auf das dictionary zugegriffen und verändert.

```

1 program phonedictionary;
3 CONST    max = 10;
5 TYPE     Entry = RECORD
        firstName:    STRING[20];
        lastName:     STRING[30];
        phoneNumber:  INTEGER;
9 END; (*RECORD*)
11 PhoneBook = ARRAY [1 .. max] OF Entry;
13 (*adds entry to a Phonebook*)
PROCEDURE AddEntry(fname,lname : STRING; pn :INTEGER; VAR dictionary :
        Phonebook);
15 VAR i : Integer;
    VAR addpossible: Boolean;
17 BEGIN
        addpossible := False;
19
        FOR i := 1 TO length(dictionary) DO
21             IF dictionary[i].firstName = '' THEN BEGIN
                    dictionary[i].firstName := fname;
23                     dictionary[i].lastName := lname;
                    dictionary[i].phoneNumber := pn;
25                     addpossible := True;
                    break;
27             end;
29
        IF NOT addpossible THEN
            WriteLn('AddEntry not possible');
31 END;
33 (*deletes entry and moves later entries down (no gaps)*)
PROCEDURE DeleteEntry(fname, lname : STRING; VAR dictionary : Phonebook);
35 VAR i, i2 : Integer;
    VAR found : Boolean;
37 BEGIN
        found := False;
39        FOR i := 1 TO length(dictionary) DO
            IF (dictionary[i].firstName = fname) AND (dictionary[i].lastName = lname)
            THEN BEGIN
41                dictionary[i].firstName := '';
                    dictionary[i].lastName := '';
43                dictionary[i].phoneNumber := 0;
                    found := True;
45

```

```

47     FOR i2 := i TO length(dictionary)-1 DO BEGIN
        dictionary[i2].firstName := dictionary[i2+1].firstName;
        dictionary[i2].lastName := dictionary[i2+1].lastName;
49     dictionary[i2].phoneNumber := dictionary[i2+1].phoneNumber;
        end;
51     end;

53     IF NOT found THEN
        WriteLn('DeleteEntry: Entry not found');
55 END;

57 (*returns number if entry matches with strings*)
PROCEDURE SearchNumber(fname, lname : String; dictionary : Phonebook);
59 VAR i : Integer;
    VAR found : Boolean;
61 BEGIN
    found := False;
63     FOR i := 1 TO length(dictionary) DO BEGIN
        IF (dictionary[i].firstName = fname) AND (dictionary[i].lastName = lname)
        THEN BEGIN
65             WriteLn('Phonenumber of ', fname, ' ', lname, ': ', dictionary[i].
                phoneNumber);
                found := True;
67             end;
        end;
69
        IF NOT found THEN
71             WriteLn('SearchNumber: Entry not found');
        END;
73

    (*prints out the number of the entries in a Phonebook*)
75 FUNCTION NrofEntries(dictionary : Phonebook): Integer;
    VAR i, count : INTEGER;
77 BEGIN
    count := 0;
79     FOR i := 1 TO length(dictionary) DO
        IF dictionary[i].firstName <> '' THEN
81         count := count +1;

83     NrofEntries := count;
    END;
85

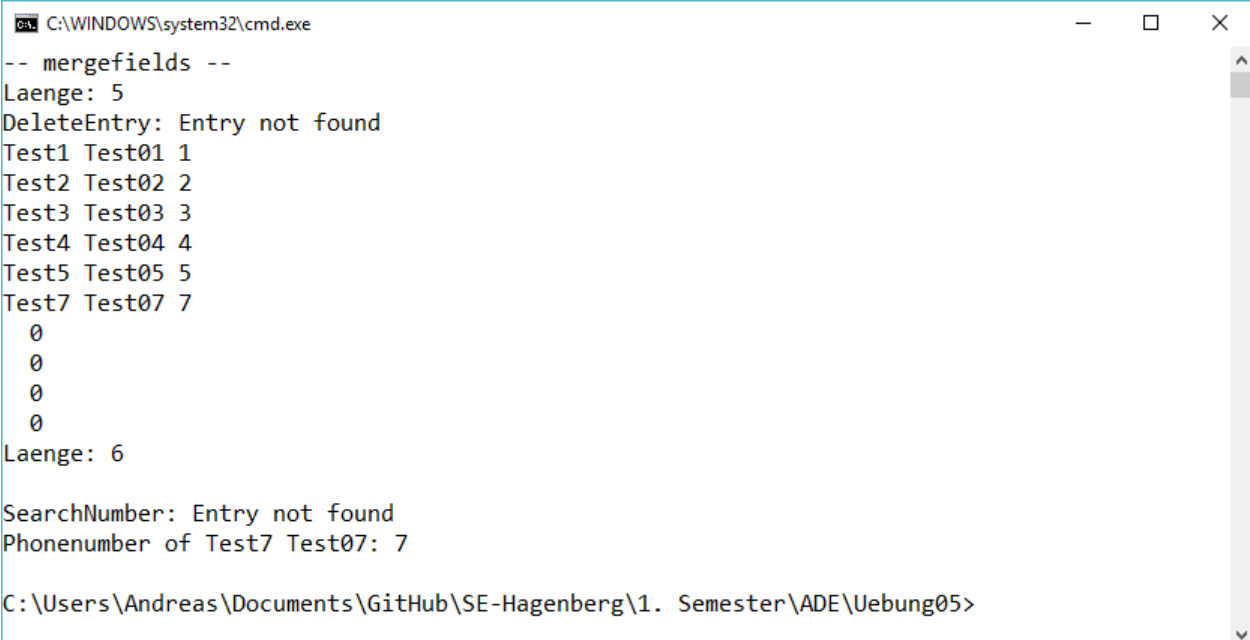
    (*procedure for printing out the whole dictionary*)
87 PROCEDURE PrintDictionary(dictionary : Phonebook);
    VAR i : Integer;
89 BEGIN
    FOR i := 1 TO length(dictionary) DO
91         WriteLn(dictionary[i].firstName, ' ', dictionary[i].lastName, ' ',
            dictionary[i].phoneNumber);
    END;
93

    VAR dictionary : Phonebook;
95 BEGIN
    WriteLn('— mergefields —');
97     AddEntry('Test1', 'Test01', 1, dictionary);
        AddEntry('Test2', 'Test02', 2, dictionary);
99     AddEntry('Test3', 'Test03', 3, dictionary);
        AddEntry('Test4', 'Test04', 4, dictionary);

```

```
101 AddEntry( 'Test5 ', 'Test05 ',5,dictionary);
103 WriteLn( 'Laenge: ',NrofEntries(dictionary));
105 DeleteEntry( 'Test6 ', 'Test06 ',dictionary);
107 AddEntry( 'Test6 ', 'Test06 ',6,dictionary);
AddEntry( 'Test7 ', 'Test07 ',7,dictionary);
109
DeleteEntry( 'Test6 ', 'Test06 ',dictionary);
111 PrintDictionary(dictionary);
WriteLn( 'Laenge: ',NrofEntries(dictionary),#13#10);
113
SearchNumber( 'Test7 ', 'Test7 ',dictionary);
115 SearchNumber( 'Test7 ', 'Test07 ',dictionary);
END.
```

phonedictionary.pas



```
C:\WINDOWS\system32\cmd.exe
-- mergefields --
Laenge: 5
DeleteEntry: Entry not found
Test1 Test01 1
Test2 Test02 2
Test3 Test03 3
Test4 Test04 4
Test5 Test05 5
Test7 Test07 7
0
0
0
0
Laenge: 6

SearchNumber: Entry not found
Phonenummer of Test7 Test07: 7

C:\Users\Andreas\Documents\GitHub\SE-Hagenberg\1. Semester\ADE\Uebung05>
```

Abbildung 1: Testfälle Phonedictionary

## Testfälle

Es werden alle Funktionen und Procedures getestet, auch auf falsche Eingaben.

## Aufgabe 2

### Lösungsidee

Zuerst wird das erste Array durchsucht und alle Zahlen die nicht in dem zweiten Array vorkommen werden in das dritte Array geschrieben. Dadurch das die beiden ersten Arrays sortiert sind ist das dritte Array nah dem einfügen auch sortiert. Danach wird beim Durchlauf des zweiten Arrays alle Zahlen die nicht im ersten Array vorkommen geordnet in das dritte Array eingefügt.

```

1 program mergefields;
2
3 (*gibt ein array auf der konsole aus*)
4 procedure printArray(a : ARRAY OF INTEGER);
5   var i : INTEGER;
6   begin
7     for i := 0 to length(a)-1 do
8       Write(a[i], ' ');
9       WriteLn();
10    end;
11
12 (*setzt alle elemente in einem array auf 0*)
13 procedure clearArray(var a : ARRAY OF INTEGER);
14   var i : INTEGER;
15   begin
16     for i := 0 to length(a)-1 do
17       a[i] := 0;
18    end;
19
20 PROCEDURE Merge(a1, a2: ARRAY OF INTEGER; VAR a3: ARRAY OF INTEGER; VAR n3:
21   INTEGER);
22   var i,i2,i3,i4,count : INTEGER;
23   var found : Boolean;
24   BEGIN
25     count := 0;
26     found := False;
27     n3 := 0;
28
29     (*schreibt alle unterschiedlichen elemente von a1 und a2 aus a1 in a3*)
30     FOR i := 0 TO length(a1)-1 DO BEGIN
31       FOR i2 := 0 TO length(a2)-1 DO BEGIN
32         IF a2[i2] = a1[i] THEN
33           found := True;
34         END;
35         IF found = False then
36           BEGIN
37             a3[count] := a1[i];
38             count := count + 1;
39             n3 := n3 + 1;
40           END;
41         found := False;
42       END;
43
44     (*schreibt alle unterschiedlichen elemente aus a1 und a2 aus a2 geordnet in
45     a3*)
46     FOR i := 0 TO length(a2)-1 DO BEGIN
47       FOR i2 := 0 TO length(a1)-1 DO BEGIN

```

```

46      IF a1[i2] = a2[i] THEN
47          found := True;
48      END;

50      IF found = False THEN
51          FOR i3 := 0 TO length(a3)-1 DO
52              IF (a2[i] <= a3[i3]) OR ((a2[i] >= a3[i3]) AND (a3[i3] = 0)) THEN
53                  BEGIN
54                      FOR i4 := length(a3)-1 DOWNTO i3+1 DO
55                          a3[i4] := a3[i4-1];
56
57                          a3[i3] := a2[i];
58                          n3 := n3 + 1;
59                          break;
60                      END;
61                  found := false;
62              END;
63          END;

64      var a1 : ARRAY [1 .. 6] OF INTEGER;
65      var a2 : ARRAY [1 .. 4] OF INTEGER;
66      var a3 : ARRAY [1 .. (length(a1) + length(a2))] OF INTEGER;
67      var n3 : INTEGER;

68      BEGIN
69          WriteLn('— mergefields —');
70          a1[1] := 1;
71          a1[2] := 2;
72          a1[3] := 3;
73          a1[4] := 4;
74          a1[5] := 5;
75          a1[6] := 6;
76
77          a2[1] := 7;
78          a2[2] := 8;
79          a2[3] := 9;
80          a2[4] := 10;

81      Merge(a1, a2, a3, n3);

82      printArray(a1);
83      printArray(a2);
84      printArray(a3);
85      WriteLn('n3: ', n3, #13#10);
86
87      clearArray(a3);

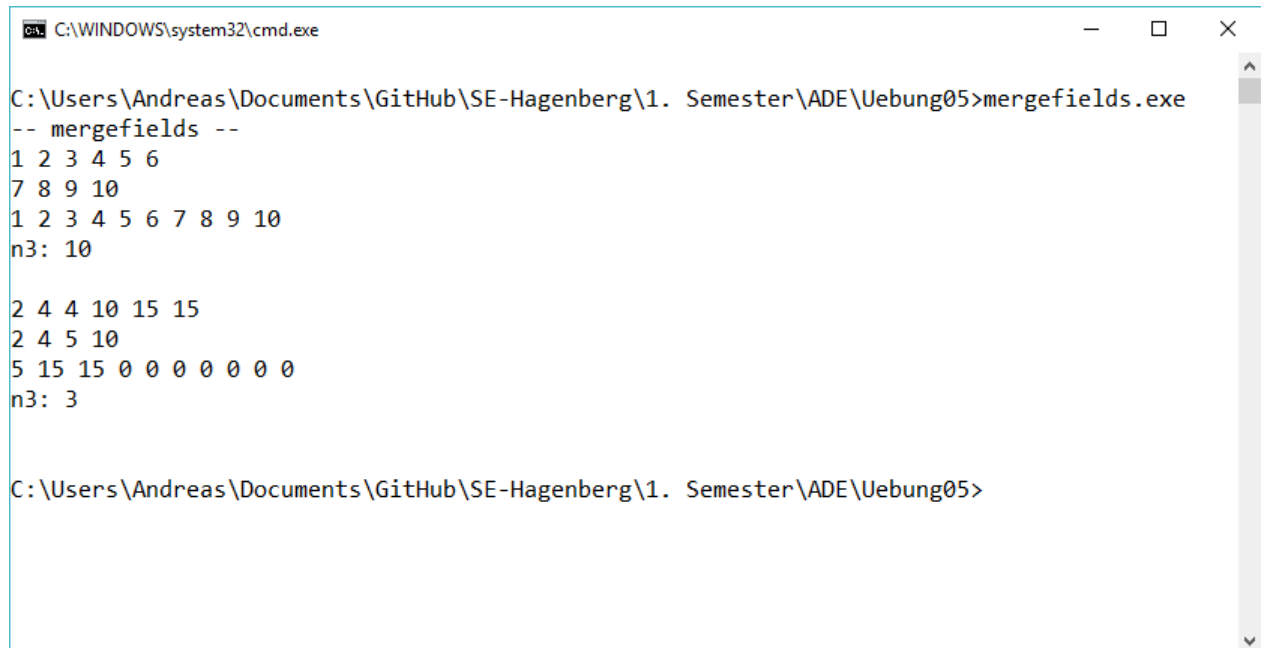
88      a1[1] := 2;
89      a1[2] := 4;
90      a1[3] := 4;
91      a1[4] := 10;
92      a1[5] := 15;
93      a1[6] := 15;

94      a2[1] := 2;
95      a2[2] := 4;
96      a2[3] := 5;
97      a2[4] := 10;

```

```
104 Merge(a1, a2, a3, n3);  
106  
108 printArray(a1);  
108 printArray(a2);  
108 printArray(a3);  
110 WriteLn('n3: ', n3, #13#10);  
END.
```

mergefields.pas



```
C:\WINDOWS\system32\cmd.exe  
C:\Users\Andreas\Documents\GitHub\SE-Hagenberg\1. Semester\ADE\Uebung05>mergefields.exe  
-- mergefields --  
1 2 3 4 5 6  
7 8 9 10  
1 2 3 4 5 6 7 8 9 10  
n3: 10  
  
2 4 4 10 15 15  
2 4 5 10  
5 15 15 0 0 0 0 0 0 0  
n3: 3  
  
C:\Users\Andreas\Documents\GitHub\SE-Hagenberg\1. Semester\ADE\Uebung05>
```

Abbildung 2: Testfälle Mergefields

## Testfälle

Zwei verschiedene Arrays, mit unterschiedlichen Zahlen bei denen Mergesort durchgeführt wird.