Programmierung **WS 2000/2001**

Prof. H. Lichter
M. Schnizler

Übung 9
Musterlösung

Aufgabe 9.1

Aufwand der Funktion EMPotenz:

$A_{EMP}(n) = n * 1 = n$

Begründung: Die FOR-Schleife wird genau n-mal durchlaufen, wobei jedes Mal eine Multiplikation durchgeführt wird.

Aufwand der Funktion SQPotenz:

Zunächst Berechnung einiger Beispielwerte:

$A_{SQP}(0) = 0$ => Keine Multiplikation!
$A_{SQP}(1) = A_{SQP}(0) + 1$ => Rekursion mit 0 + 1 Multiplikation

$A_{SQP}(2) = A_{SQP}(2 \text{ DIV } 2) + 1 = A_{SQP}(1) + 1 = 2$
$A_{SQP}(3) = A_{SQP}(3 - 1) + 1 = A_{SQP}(2) + 1 = A_{SQP}(2 \text{ DIV } 2) + 1 + 1 = A_{SQP}(1) + 2 = 3$
$A_{SQP}(4) = A_{SQP}(4 \text{ DIV } 2) + 1 = A_{SQP}(2) + 1 = A_{SQP}(2 \text{ DIV } 2) + 1 + 1 = A_{SQP}(1) + 2 = 3$
$A_{SQP}(5) = A_{SQP}(4) + 1 = A_{SQP}(2) + 2 = A_{SQP}(1) + 3 = 4$
$A_{SQP}(6) = A_{SQP}(3) + 1 = A_{SQP}(2) + 2 = A_{SQP}(1) + 3 = 4$
$A_{SQP}(7) = A_{SQP}(6) + 1 = A_{SQP}(3) + 2 = A_{SQP}(2) + 3 = A_{SQP}(1) + 4 = 5$
$A_{SQP}(8) = A_{SQP}(4) + 1 = A_{SQP}(2) + 2 = A_{SQP}(1) + 3 = 4$

Im folgenden sei $ld\ n = \log_2 n$ der ganzzahlige Wert des Logarithmus zur Basis 2.

Untere Schranke:
Betrachtet man die Beispiele, so zeigt sich, dass der Aufwand für n = 8 den günstigsten Fall darstellt. Der Exponent n ist eine ganzzahlige Potenz von 2, nämlich $n = 2^3 = 8$ und da der Exponent in jedem Rekursionsschritt halbiert wird, ergibt sich folgender Aufwand:

$A_{SQP}(n) = ld\ n + 1$ (für $n = 2^m$ und $m \geq 0$)

Obere Schranke:
Im ungünstigsten Fall ist der Exponent ungerade und es entsteht bei jeder darauffolgenden Division durch 2 wiederum eine ungerade Zahl. Das ist beispielsweise der Fall bei n = 7, zu Beginn und nach jeder Division ist eine zusätzliche Multiplikation für die Rekursion mit dem um eins verringerten Wert notwendig:

$A_{SQP}(n) = 2 * ld\ n + 1$ (für $n = 2^m - 1$ und $m \geq 0$)

Vergleich der Ergebnisse:

$A_{EMP}(1024) = 1024$ $A_{SQP}(1024) = ld\ 1024 + 1 = 10 + 1 = 11$ Faktor: $1024 / 11 \approx 93$
$A_{EMP}(1023) = 1023$ $A_{SQP}(1023) = 2 * ld\ 1023 + 1 = 18 + 1 = 19$ Faktor: $1023 / 19 \approx 54$

Aufgabe 9.2

```
MODULE TerminDatei EXPORTS Main;

(* Dieses Programm realisiert eine simple Terminverwaltung, die
   zuvor eingegebene Termine sortiert ausgibt. Die Termine koennen
   ausserdem in einer Datei gespeichert und von dort wieder gelesen
   werden.

   Autor           : Moritz Schnizler, RWTH Aachen
   Umgebung        : PM 3, Windows NT 4.0
   Erstellt        : 23.11.00
   Letzte Aenderung: 19.01.01 *)

IMPORT SIO;    (* Importiere notwendige Ein-/Ausgabeoperationen *)
IMPORT IO;     (* Importiere Operationen um Dateien zu oeffnen*)
IMPORT Rd, Wr; (* Importiere Op. um Dateien zu lesen/schreiben *)

CONST MAXTERMIN = 10;         (* Maximale Anzahl der Termine *)
      KALENDER  = "Kalender"; (* Name der Ausgabedatei *)

TYPE (* Unterbereichtstypen und Typ Uhrzeit fuer die Zeitangabe *)

     Stunden        = [0..23];

     Minuten        = [0..59];

     Uhrzeit        = RECORD
                        stunde: Stunden;
                        minute: Minuten;
                      END;


     (* Typ Termin fuer einzelnen Termin mit Uhrzeit/Beschreibung *)

     Termin         = RECORD
                        beschreibung: TEXT;
                        zeit        : Uhrzeit;
                      END;


     (* Typ Terminkalender fuer maximal MAXTERMIN Termine *)

     Terminkalender = ARRAY [1..MAXTERMIN] OF Termin;


(* ... unveränderte Prozeduren und Funktionen aus Aufg. 6.2 werden
       im weiteren nicht aufgeführt ... *)
```

```
(* Prozeduren fuer das Schreiben und Lesen der Termindatei *)

PROCEDURE TerminLesen(datei: Rd.T): Termin =
(* Liest die Daten des Termins aus einer gegebenen Datei *)
VAR ergTermin   : Termin;
    trennzeichen: CHAR;

BEGIN
  ergTermin.zeit.stunde := SIO.GetInt(datei);
  trennzeichen := SIO.GetChar(datei);
  ergTermin.zeit.minute := SIO.GetInt(datei);
  trennzeichen := SIO.GetChar(datei);
  ergTermin.beschreibung := SIO.GetLine(datei);

  RETURN ergTermin;
END TerminLesen;


PROCEDURE TerminSchreiben(datei: Wr.T; termin: Termin) =
(* Schreibt die Daten des Termins in die gegebene Datei im Format:
   Stunde:MinuteBeschreibungRETURN *)
BEGIN
  SIO.PutInt(termin.zeit.stunde, 10, datei);
  SIO.PutChar(':', datei);  (* Als Trennzeichen *)
  SIO.PutInt(termin.zeit.minute, 10, datei);
  SIO.PutChar(' ', datei);         (* Als Trennzeichen *)
  SIO.PutLine(termin.beschreibung, datei);
END TerminSchreiben;


PROCEDURE TerminkalenderLesen(datei: Rd.T; VAR termine:
Terminkalender; VAR maxIndex: [0..MAXTERMIN]) =
(* Liest alle (maximal MAXTERMIN viele) Termine eines
Terminkalenders aus der gegebenen Datei.
   Der gefüllte Terminkalender und der maximal belegte Index werden
zurueckgegeben. *)
VAR i: INTEGER := 0;
BEGIN

  WHILE NOT SIO.EOF(datei) AND (i <= MAXTERMIN) DO
    INC(i);
    termine[i] := TerminLesen(datei);
  END;

  maxIndex := i;
END TerminkalenderLesen;


PROCEDURE TerminkalenderSchreiben(datei: Wr.T; termine:
Terminkalender; maxIndex: INTEGER) =
(* Schreibt den bis maxIndex gefuellten Terminkalender in die
gegebene Datei *)
BEGIN
  FOR i := FIRST(termine) TO maxIndex DO
    TerminSchreiben(datei, termine[i]);
  END;
END TerminkalenderSchreiben;
```

```
VAR tKalender  : Terminkalender;
    terminIndex: [0..MAXTERMIN] := 0;

    (* Variablen fuer zum Schreiben bzw. Lesen geoeffnete Datei *)
    schreibDatei: Wr.T;
    leseDatei   : Rd.T;

    (* Notwendig fuer die Benutzerinteraktion *)
    auswahl    : CHAR;
    restzeile  : TEXT;

BEGIN
  (* Lesen der Terminkalenderdatei *)
  leseDatei := IO.OpenRead(KALENDER);
  IF (leseDatei # NIL) THEN
    TerminkalenderLesen(leseDatei, tKalender, terminIndex);
    Rd.Close(leseDatei);
  END;

  (* Benutzerinteraktion mittels REPEAT-Schleife *)
  REPEAT

    (* Auswahlmenue ausgeben *)
    SIO.Nl();
    SIO.PutLine("***  Terminkalender  ***");
    SIO.PutLine("(e) Neuen Termin eingeben");
    SIO.PutLine("(+) Aufsteigend sortiert ausgeben");
    SIO.PutLine("(-) Absteigend sortiert ausgeben");
    SIO.PutLine("(q) Beendet das Programm");
    SIO.Nl();

    (* Eingabe der ausgewaehlten Operation *)
    SIO.PutText("Auswahl: ");
    auswahl   := SIO.GetChar();
    restzeile := SIO.GetLine();
    SIO.Nl();

    (* Ausgewaehlte Operation ausfuehren *)
    CASE auswahl OF
       'e', 'E' => IF terminIndex < MAXTERMIN THEN
                     (* Solange noch Platz im Terminkalender *)
                     INC(terminIndex);
                     tKalender[terminIndex] :=TerminEingeben();

                     (* Terminkalender sofort sortieren *)
                     TerminkalenderSortieren(tKalender,terminIndex);
                   ELSE
                     SIO.PutLine("Terminkalender leider voll!");
                   END;

    |  '+'      => (* Terminkalender aufsteigend ausgeben *)
                   TerminkalenderAusgeben(tKalender, terminIndex,
                                          TRUE);

    |  '-'      => (* Terminkalender absteigend ausgeben *)
                   TerminkalenderAusgeben(tKalender, terminIndex,
                                          FALSE);
```

```
    |  'q', 'Q' => (* Kalenderdaten in Datei schreiben *)
                   schreibDatei := IO.OpenWrite(KALENDER);

                   IF (schreibDatei # NIL) THEN
                     TerminkalenderSchreiben(schreibDatei,tKalender,
                                             terminIndex);
                     Wr.Close(schreibDatei);
                   ELSE
                     SIO.PutLine("Kann Datei nicht schreiben!");
                   END;

                   SIO.PutLine("Programmende!");
    ELSE
      SIO.PutLine("Ungueltige Eingabe!");
    END;
  UNTIL (auswahl = 'q') OR (auswahl = 'Q');
END TerminDatei.
```

Probelauf des Programms, um drei Termine einzugeben:

```
***  Terminkalender  ***
(e) Neuen Termin eingeben
(+) Aufsteigend sortiert ausgeben
(-) Absteigend sortiert ausgeben
(q) Beendet das Programm

Auswahl: e

Beschreibung des Termins: Frühstücken
Uhrzeit (Stunden): 6
Uhrzeit (Minuten): 00

***  Terminkalender  ***
...

Auswahl: e

Beschreibung des Termins: Abendessen
Uhrzeit (Stunden): 20
Uhrzeit (Minuten): 30

***  Terminkalender  ***
...

Auswahl: e

Beschreibung des Termins: Mittagessen
Uhrzeit (Stunden): 13
Uhrzeit (Minuten): 00
```

Inhalt der erzeugten Beispieldatei:

```
6:0Fr hst cken
13:0Mittagessen
20:30Abendessen
```

Aufgabe 9.3

a) Geeignete Äquivalenzklassen zu den gegebenen Eingabebedingungen:

Eingabebedingung	Äquivalenzklassen	
	Gültig	Ungültig
Familienname notwendig	ein oder mehr Zeichen (1)	leerer Text (2)
Rufname notwendig	ein oder mehr Zeichen (3)	leerer Text (4)
maximal fünf Vornamen	keiner oder maximal vier weitere Vornamen (5)	fünf oder mehr weitere Vornamen (6)
nach greg. Kalender gültiges Geburtsdatum notwendig	gültiges Datum von 1800 - 3000 außer Schalttage (7), Schalttag von 1800 - 3000 (8)	ungültiges Datum von 1800 bis 3000 (9), ungültiges Datum vor 1800 oder nach 3000 (10), gültiges Datum vor 1800 (11), gültiges Datum nach 3000 (12)
Geburtsort notwendig	ein oder mehr Zeichen (13)	leerer Text (14)
Geschlecht notwendig	{männlich, weiblich} (15)	sonstige Eingabe (16)
Strassenangabe notwendig	ein oder mehr Zeichen (17)	leerer Text (18)
Hausnummer von 1 bis 1000	Wert von 1 bis 1000 (19)	Wert < 1 (20), Wert > 1000 (21)
Adresszusatz freiwillig	beliebige Eingabe (22)	
Postleitzahl von 0 bis 99999	Wert von 0 bis 99999 (23)	Wert < 0 (24), Wert > 99999 (25)
Ortsangabe notwendig	ein oder mehr Zeichen (26)	leerer Text (27)
Familienstand notwendig	{verh., ledig, gesch., verw.}	sonstige Eingabe (28)
falls verheiratet, Name des Ehepartners erforderlich	{verh.} (29), {ledig, gesch., verw.} (30)	sonstige Eingabe (28)
falls verheiratet, Familien-name des EP notwendig	ein oder mehr Zeichen (31)	leerer Text (32)
falls verheiratet, Rufname des EP notwendig	ein oder mehr Zeichen (33)	leerer Text (34)
falls verheiratet, max. fünf Vornamen des EP	kein oder maximal vier weitere Vornamen (35)	fünf oder mehr weitere Vornamen (36)
nach greg. Kalender gültiges Anmeldedatum notwendig	gültiges Datum von 1800 - 3000 außer Schalttage (37), Schalttag von 1800 - 3000 (38)	ungültiges Datum von 1800 bis 3000 (39), ungültiges Datum vor 1800 oder nach 3000 (40), gültiges Datum vor 1800 (41), gültiges Datum nach 3000 (42)

Testfälle, welche die oben angegebenen Äquivalenzklassen überdecken:

Testfallnummer	1	2	3	4	5	6	7	8	9
Familienname	M	Meier	Meier		Meier	Testfall	Meier	Meier	Meier
Rufname	W	Wilhelm	Wilhelm	Wilhelm		nicht	Wilhelm	Wilhelm	Wilhelm
2. Vorname		Albert	Albert	Albert	Albert	möglich!	Albert	Albert	Albert
3. Vorname		Friedrich							
4. Vorname		Franz							
5. Vorname		Ludwig							
Geb.datum	01.01.1800	31.12.3000	29.02.1980	20.01.2000	20.01.2000		30.02.1980	32.03.1799	20.03.1799
Geburtsort	B	Bonn	Bonn	Bonn	Bonn		Bonn	Bonn	Bonn
Geschlecht	männlich	männlich	männlich	männlich	männlich		männlich	männlich	männlich
Strasse	S	Spreeweg	Spreeweg	Spreeweg	Spreeweg		Spreeweg	Spreeweg	Spreeweg
Hausnummer	1	1000	500	500	500		500	500	500
Adresszusatz		A	A	A	A		A	A	A
Postleitzahl	0	99999	50000	50000	50000		50000	50000	50000
Ort	B	Bonn	Bonn	Bonn	Bonn		Bonn	Bonn	Bonn
Familienstand	verheiratet	verheiratet	ledig	ledig	ledig		ledig	ledig	ledig
Familienname EP	W	Walter							
Rufname EP	O	Olga							
2. Vorname EP		Erika							
3. Vorname EP		Helga							
4. Vorname EP		Erna							
5. Vorname EP		Lora							
Anmeldedatum	01.01.1800	31.12.3000	29.02.1904	20.03.1880	20.03.1880		20.03.1880	20.03.1980	20.03.1980
Sollresultat	OK	OK	OK	Familien-name fehlt!	Rufname fehlt!	Zuviele Vornamen!	Ungültiges Geb.datum!	Ungültiges Geb.datum!	Geb.datum zu früh!

Testfallnummer	10	11	12	13	14	15	16	17	18
Familienname	Meier	Meier	Meier	Meier	Meier	Meier	Meier	Meier	Meier
Rufname	Wilhelm	Wilhelm	Wilhelm	Wilhelm	Wilhelm	Wilhelm	Wilhelm	Wilhelm	Wilhelm
2. Vorname	Albert	Albert	Albert	Albert	Albert	Albert	Albert	Albert	Albert
3. Vorname									
4. Vorname									
5. Vorname									
Geb.datum	01.01.3001	20.03.1988	20.03.1988	20.03.1980	20.03.1980	20.03.1980	20.03.1980	20.03.1980	20.03.1980
Geburtsort	Bonn		Bonn	Bonn	Bonn	Bonn	Bonn	Bonn	Bonn
Geschlecht	männlich	männlich	xy	männlich	männlich	männlich	männlich	männlich	männlich
Strasse	Spreeweg	Spreeweg	Spreeweg		Spreeweg	Spreeweg	Spreeweg	Spreeweg	Spreeweg
Hausnummer	500	500	500	500	0	1001	500	500	500
Adresszusatz	A	A	A	A	A	A	A	A	A
Postleitzahl	50000	50000	50000	50000	50000	50000	-1	100000	50000
Ort	Bonn	Bonn	Bonn	Bonn	Bonn	Bonn	Bonn	Bonn	
Familienstand	ledig	ledig	ledig	ledig	ledig	ledig	ledig	ledig	ledig
Familienname EP									
Rufname EP									
2. Vorname EP									
3. Vorname EP									
4. Vorname EP									
5. Vorname EP									
Anmeldedatum	20.03.1980	20.03.1980	20.03.1980	20.03.1980	20.03.1980	20.03.1980	20.03.1980	20.03.1980	20.03.1980
Sollresultat	Geb.datum zu spät!	Geburtsort fehlt!	Geschlecht falsch!	Strasse fehlt!	Nummer falsch!	Nummer falsch!	PLZ falsch!	PLZ falsch!	Wohnort fehlt!

Testfallnummer	19	20	21	22	23	24	25	26	27 (Zusatz)
Familienname	Meier	Meier	Meier	Testfall	Meier	Meier	Meier	Meier	Meier
Rufname	Wilhelm	Wilhelm	Wilhelm	nicht	Wilhelm	Wilhelm	Wilhelm	Wilhelm	Wilhelm
2. Vorname	Albert	Albert	Albert	möglich!	Albert	Albert	Albert	Albert	Albert
3. Vorname									
4. Vorname									
5. Vorname									
Geb.datum	20.03.1988	20.03.1980	20.03.1988		20.03.1980	20.03.1980	20.03.2000	20.03.2000	20.03.2000
Geburtsort	Bonn	Bonn	Bonn		Bonn	Bonn	Bonn	Bonn	Bonn
Geschlecht	männlich	männlich	männlich		männlich	männlich	männlich	männlich	männlich
Strasse	Spreeweg	Spreeweg	Spreeweg		Spreeweg	Spreeweg	Spreeweg	Spreeweg	Spreeweg
Hausnummer	500	500	500		500	500	500	500	500
Adresszusatz	A	A	A		A	A	A	A	A
Postleitzahl	50000	50000	50000		50000	50000	50000	50000	50000
Ort	Bonn	Bonn	Bonn		Bonn	Bonn	Bonn	Bonn	Bonn
Familienstand	unbekannt	verheiratet	verheiratet		ledig	ledig	ledig	ledig	ledig
Familienname EP			Walter						
Rufname EP		Olga							
2. Vorname EP		Erika	Erika						
3. Vorname EP									
4. Vorname EP									
5. Vorname EP									
Anmeldedatum	20.03.1980	20.03.1980	20.03.1980		4.13.1960	43.12.3001	31.12.1799	01.01.3001	29.02.1983
Sollresultat	Familienstand falsch!	Fam.name EP fehlt!	Rufname EP fehlt!		Ungültiges Anmdatum!	Ungültiges Anmdatum!	Ungültiges Anmdatum!	Ungültiges Anmdatum!	Ungültiges Anmdatum!

Tabelle der Überdeckung Testfall/Äquivalenzklassen:

Testfall	Äquivalenzklassen																																									
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42
1	U		U		U		U						U		X		U		U			X	U			U			X		U		U		U		U					
2	X		X		O		O						X		X		X		O			X	O			X			X		X		X		O		O					
3	X		X		X			X					X		X		X		X			X	X			X				X								X				
4		X	X		X		X						X		X		X		X			X	X			X				X							X					
5	X			X	X		X						X		X		X		X			X	X			X				X							X					
6	X		X			X	X						X		X		X		X			X	X			X				X							X					
7	X		X		X				X				X		X		X		X			X	X			X				X							X					
8	X		X		X					X			X		X		X		X			X	X			X				X							X					
9	X		X		X						X		X		X		X		X			X	X			X				X							X					
10	X		X		X							X	X		X		X		X			X	X			X				X							X					
11	X		X		X		X							X	X		X		X			X	X			X				X							X					
12	X		X		X		X						X			X	X		X			X	X			X				X							X					
13	X		X		X		X						X		X			X	X			X	X			X				X							X					
14	X		X		X		X						X		X		X			X		X	X			X				X							X					
15	X		X		X		X						X		X		X				X	X	X			X				X							X					
16	X		X		X		X						X		X		X		X			X		X		X				X							X					
17	X		X		X		X						X		X		X		X			X			X	X				X							X					
18	X		X		X		X						X		X		X		X			X	X				X			X							X					
19	X		X		X		X						X		X		X		X			X	X			X		X									X					
20	X		X		X		X						X		X		X		X			X	X			X			X			X	X		X		X					
21	X		X		X		X						X		X		X		X			X	X			X			X		X			X	X		X					
22	X		X		X		X						X		X		X		X			X	X			X			X		X		X			X	X					
23	X		X		X		X						X		X		X		X			X	X			X				X									X			
24	X		X		X		X						X		X		X		X			X	X			X				X										X		
25	X		X		X		X						X		X		X		X			X	X			X				X											X	
26	X		X		X		X						X		X		X		X			X	X			X				X												X

U = Untergrenze Äquivalenzklasse
O = Obergrenze Äquivalenzklasse
X = beliebiger Wert aus Äquivalenzklasse

HINWEIS: Bei der Bestimmung der Äquivalenzklassen und der Grenzwertanalyse werden keine Abhängigkeiten zwischen den Eingaben berücksichtigt, sondern nur die einzelnen Datenwerte betrachtet. Beispielsweise wird bei der Testfallauswahl nicht erzwungen, dass das Anmeldedatum später als das Geburtsdatum liegt. Auch lassen sich durch eine andere Wahl der Äquivalenzklassen (vor allem bezüglich des Datums) noch wesentlich mehr Testfälle ermitteln, die mit Sicherheit weitere Fehler finden. Dies erfordert deutlich mehr Aufwand, der sich allerdings lohnt, wie sich allein an dem zusätzlichen Testfall 27 zeigt.

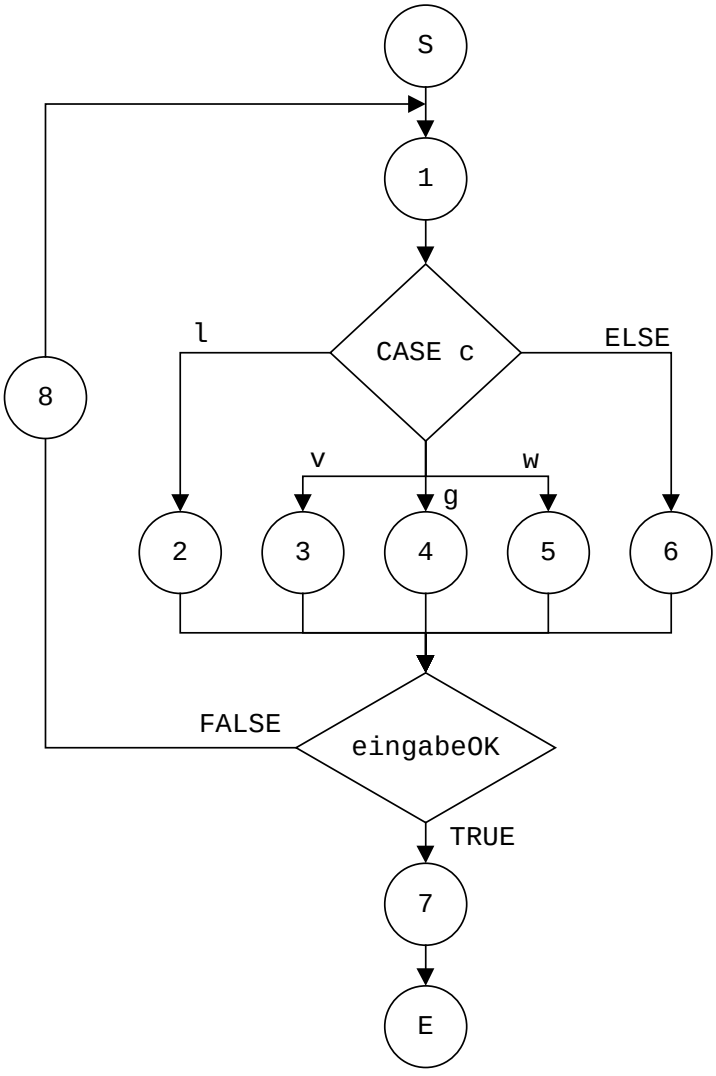
b) Festgestellte Abweichungen zwischen Soll- und Ist-Resultat:

Testfall	Abweichung
4	Keine Meldung, wenn Familienname fehlt!
5	Keine Meldung, wenn Rufname fehlt!
6	Ungültiges Datum wird akzeptiert!
11	Keine Meldung, wenn Geburtsort fehlt!
13	Keine Meldung, wenn Strasse fehlt!
18	Keine Meldung, wenn Wohnort fehlt!
20	Keine Meldung, wenn Familienname Ehepartner fehlt!
21	Keine Meldung, wenn Rufname Ehepartner fehlt!
27	Falscher Schalttag wird akzeptiert!

Erzielte Zweigüberdeckung: 89,6 %

(beispielsweise wird in keinem Testfall als Familienstand "geschieden" oder "verwitwet" verwendet)

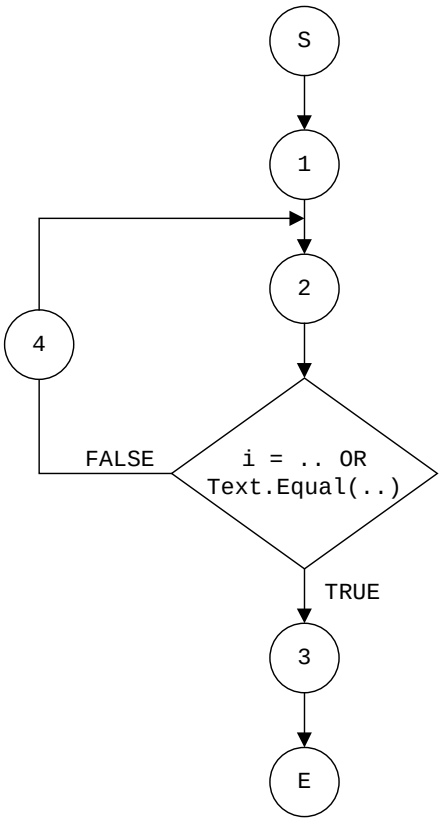
c) Flussdiagramm für Prozedur `FamilienstandEingeben()`:



Testfälle für Zweigüberdeckung:

Testfall	Eingabe für c	Soll-Rückgabewert	Überdeckte Zweige
1	'l'	Familienstand.ledig	1, 2, 7
2	'v'	Familienstand.verheiratet	1, 3, 7
3	'g'	Familienstand.geschieden	1, 4, 7
4	'w'	Familienstand.verwitwet	1, 5, 7
5	'x' 'l'	Familienstand.ledig	1, 6, 8, 1, 2, 7

c) Flussdiagramm für Prozedur `PersonEingeben()`:



Testfälle für eingeschränkte Pfadüberdeckung:

Testfall	Eingabe für Person	Soll-Rückgabewert für Person	Überdeckter Pfad
1	"Müller" "Werner" ""	Müller, Werner, , , ,	Keine Iteration
2	"Müller" "Werner" "Albert" ""	Müller, Werner, Albert, , ,	1 Iteration
3	"Müller" "Werner" "Albert" "Franz" "Ludwig" "Wilhelm"	Müller, Werner, Albert, Franz, Ludwig, Wilhelm	3 Iterationen