Lösung zur Klausur zu Programmieren I

WS 93 / 94

9. Februar 1994

Aufgabe		Punkte
1	a)	2
	b)	2
	c)	2
	d)	2
	e)	2
	f)	2
2		25
3		8
4	a)	2
	b)	5
	c)	8
5		10
6		10
7		20
Summe		**100**

Aufgabe 1

(*a*)

```
$ cd
$ ls -l | grep '^d' | wc -l
```

ist OK, aber besser:

```
$ ls -l $HOME | grep '^d' | wc -l
```

(*b*) (i)

```
$ cat kapitel.2 >> kapitel.1
```

(ii)

```
$ vi kapitel.1
G
:r kapitel.2
ZZ
```

und noch *x* andere Lösungen. Trotzdem noch eine andere:

```
$ vi + kapitel.1
:r kapitel.2
ZZ
```

(*c*)

```
$ grep -c '^$' kapitel.1
```

oder

```
$ grep '^$' kapitel.1 | wc -l
```

(*d*) (i)

```
$ m2c -o doit doit.m2
```

(ii)

```
$ m2c doit.m2
$ mv a.out doit
```

besser:

```
$ m2c doit.m2 && mv a.out doit
```

(*e*)

```
$ echo $?
```

(*f*)

```
:1,$s/PROC/PROCEDURE/
```

Für Schreibfaule:

```
:%s/PROC/&EDURE/
```

Aufgabe 2

```
MODULE head;
   IMPORT Arguments;
   FROM StdIO   IMPORT FILE, MODE, Fopen, Fclose, stdout, stderr;
   FROM FtdIO   IMPORT FwriteString, FwriteLn;
   FROM Gegeben IMPORT FreadLine;

   CONST LineLen = 256;
         FileLen = 14;

   VAR   line      : ARRAY [0..LineLen] OF CHAR;
         filename  : ARRAY [0..FileLen] OF CHAR;
         number, i : CARDINAL;
         flag      : CHAR;
         f         : FILE;

BEGIN
   number := 5;

   Arguments.InitArgs("[-nnn] file { file }");

   WHILE Arguments.GetFlag(flag) DO
      CASE flag OF
      | "0".."9":
         Arguments.UngetOpt;
         Arguments.FetchCard(number)
      ELSE
         Arguments.Usage
      END
   END;

   IF NOT Arguments.GetArg(filename) THEN
      Arguments.Usage
   END;

   REPEAT
      i := 0;
      IF Fopen(f, filename, read, TRUE) THEN
         FwriteString(stdout, filename); FwriteLn(stdout);

         WHILE (i < number) AND FreadLine(f, line) DO
            FwriteString(stdout, line); FwriteLn(stdout);
            INC(i)
         END;

         IF NOT Fclose(f) THEN
            FwriteString(stderr, filename);
            FwriteString(stderr, ": cannot close"); FwriteLn(stderr)
         END
      ELSE
         FwriteString(stderr, filename);
         FwriteString(stderr, ": cannot open"); FwriteLn(stderr)
      END
   UNTIL NOT Arguments.GetArg(filename)
END head.
```

Aufgabe 3

```
Initialisierung:    x = 4, y = 2

IF-Statement, 1. Bedingung

Obscure(4, 2):      x = 5, y = 2, Resultat = 7
Obscure(2, 5):      x = 5, y = 3, Resultat = 8

IF-Statement, 1. Bedingung ist TRUE (7 < 8),
d.h. 2. Bedingung (OR-Teil) wird nicht betrachtet.
Weiter im IF-Statement, THEN-Teil

Obscure(5, 3):      x = 6, y = 3, Resultat = 9
Zuweisung an x:     x = 9, y = 3

Obscure(3, 9):      x = 9, y = 4, Resultat = 13
Zuweisung an y:     x = 9, y = 13

Ausgabe             x = 9, y = 13
```

Aufgabe 4

(*a*)

```
TYPE Farbe        = (rot, gruen, blau);
     LinienBreite = [1..5];
     FuellModus   = BOOLEAN;
```

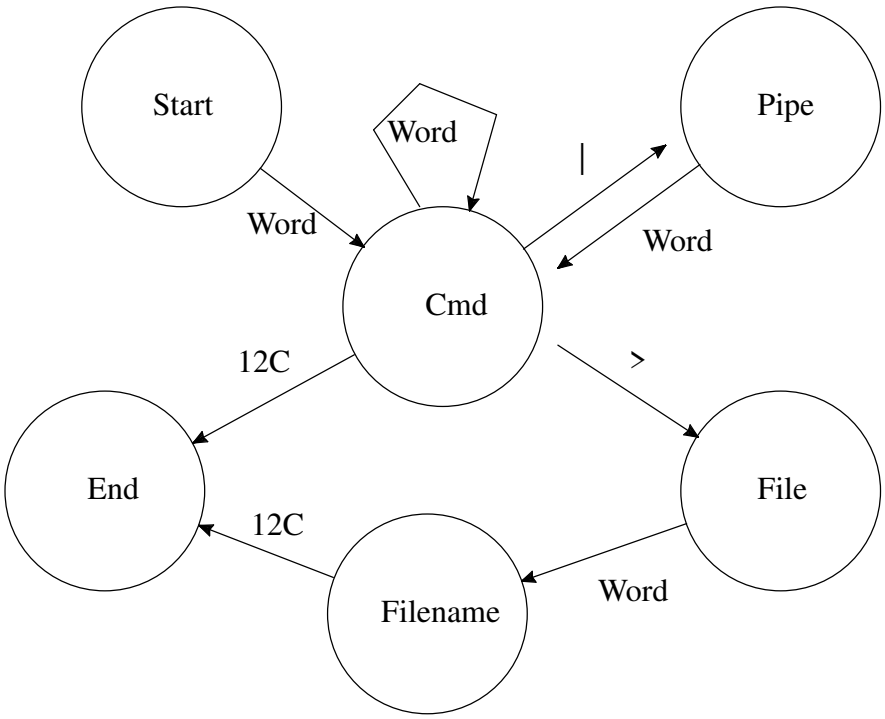
(*b*)

```
TYPE KoordinatenTyp = (kartesisch, polar);
     Koordinaten = RECORD
        CASE typ : KoordinatenTyp OF
        | kartesisch:
           x, y   : REAL
        | polar:
           r, phi : REAL
        END
     END;
     Radius = REAL;
```

(*c*)

```
TYPE ObjektTyp = (Punkt, Linie, Rechteck, Kreis);
     Objekt = RECORD
        farbe : Farbe;
        CASE typ : ObjektTyp OF
        | Punkt:
           p            : Koordinaten
        | Linie:
           l1, l2       : Koordinaten;
           linienBreite : LinienBreite
        | Rechteck:
           r1, r2       : Koordinaten;
           linienBreite : LinienBreite;
           fuellModus   : Fuellmodus
        | Kreis:
           m            : Koordinaten;
           r            : Radius;
           linienBreite : LinienBreite;
           fuellModus   : Fuellmodus
        END
     END;
```

Aufgabe 5



Aufgabe 6

```
Cmd ::= { Word | '"' { Word } '"' | '$' Word | '<' Word } '12C'
```

Aufgabe 7

```
FROM StdIO IMPORT stderr;
FROM FtdIO IMPORT FwriteString, FwriteLn;

PROCEDURE Insert(VAR t : Table; w : Word): BOOLEAN;
(*liefert TRUE, falls eingefuegt; FALSE, falls bereits besetzt*)

   PROCEDURE GetIndex(str : ARRAY OF CHAR): CARDINAL;
   (*berechnet den Index von str*)

      VAR base, index, i : CARDINAL;

   BEGIN
      base := 10; index := 0; i := 0;
      WHILE (i <= HIGH(str)) AND (str[i] # 0C) DO
         INC(index, ORD(str[i]) * base);
         base := base * 10;
         INC(i)
      END;
      RETURN ((index MOD Size) + 1)
   END GetIndex;

   VAR index : CARDINAL;

BEGIN
   index := GetIndex(w);
   IF t[index].occupied THEN
      FwriteString(stderr, "Entry occupied"); FwriteLn(stderr);
      RETURN FALSE
   ELSE
      t[index].occupied := TRUE;
      t[index].element  := w;
      RETURN TRUE
   END
END Insert;
```