Allg. Informatik für WiWi (WS 2000)

Abgabetermin: 11. Januar 2001

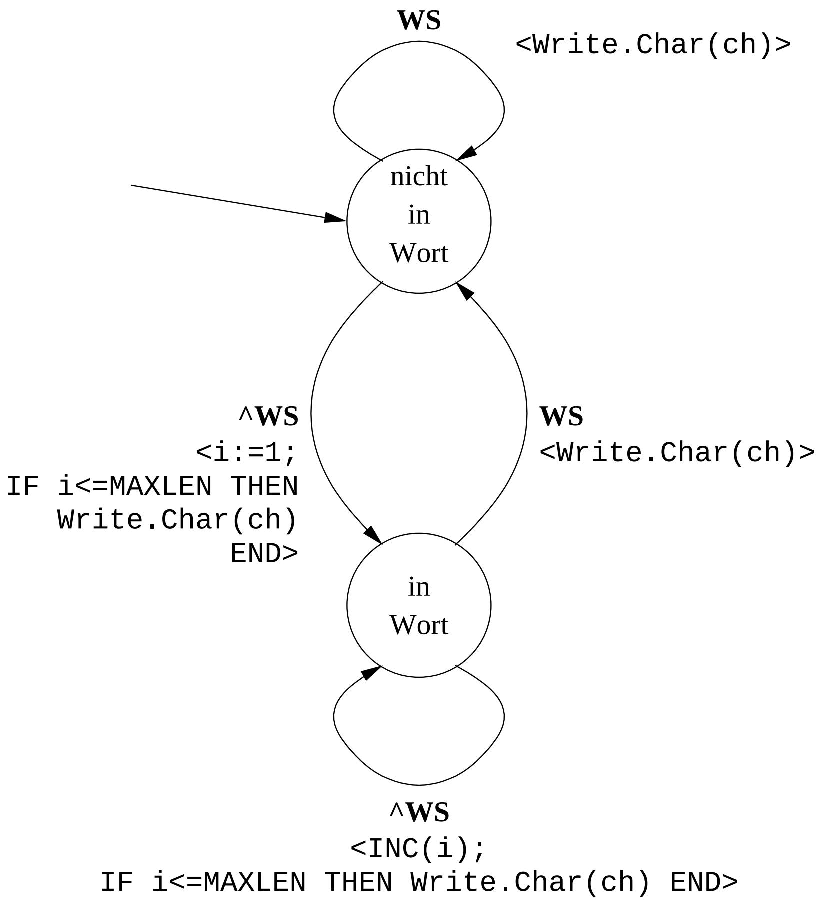
Beispiellösung

1 Bruchstücke (10 Punkte)

Zunächst sollen Sie ein Oberon-Modul `WordCut` schreiben, das Zeichen von der Standardeingabe liest. Whitespaces, d. h. Leerzeichen, Tabulatoren und Zeilenumbrüche, sollen dabei sofort ausgegeben werden. Alle anderen Zeichen sind jeweils Bestandteil eines Wortes, d. h. Worte sind durch Whitespaces voneinander getrennte Folgen von Nicht-Whitespaces. Von jedem Wort sollen nur die ersten `MAXLEN` Buchstaben ausgegeben werden, wobei `MAXLEN` in Ihrem Programm als ganzzahlige Konstante (z. B. mit Wert 3) zu definieren ist. Die restlichen Zeichen eines Wortes werden ignoriert. Sie können (müssen aber nicht!) zunächst einen Automaten konstruieren (vgl. Übungen) und diesen dann in ein Oberon-Programm transformieren.

Lösung:

Zunächst können wir den folgenden Automaten konstruieren. Bei jedem Übergang ist in spitzen Klammern das Oberon-Programmchen angegeben, das in diesem Fall ausgeführt werden soll.



Abk.: **WS = [<space><tab><newline>]**
^WS = [^<space><tab><newline>]

```
DEFINITION WordCut;
END WordCut.
```

```
MODULE WordCut;

  IMPORT Read, Write, Streams, ASCII;

  CONST NichtInWort = 1;     (* Zustand *)
        InWort = 2;          (* Zustand *)
        MAXLEN = 3;          (* max. Anzahl auszugeb. Buchst. pro Wort *)

  VAR ch : CHAR;           (* gelesenes Zeichen *)
      zustand : INTEGER;   (* aktueller Zustand *)
      i : INTEGER;         (* Anz. der geles. Buchst. des akt. Wortes *)

BEGIN
```

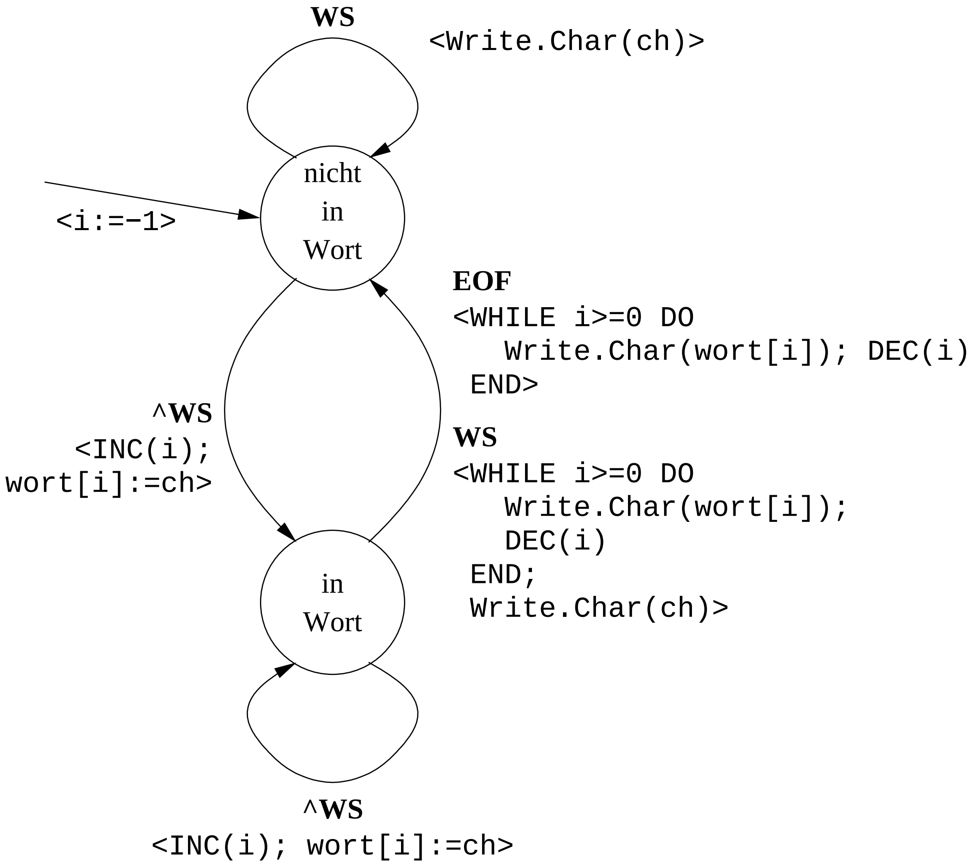
```
  (* am Anfang Zustand NichtInWort, da bisher leeres Wort gelesen *)
  zustand := NichtInWort;

  Read.Char(ch);
  WHILE ~ Streams.stdin.eof DO

    IF zustand = NichtInWort THEN

      IF (ch = ASCII.sp) OR (ch = ASCII.tab) OR (ch = ASCII.nl) THEN
        zustand := NichtInWort; (* ueberfluessig *)
        Write.Char(ch);
      ELSE
        zustand := InWort;
        i := 1;
        IF i <= MAXLEN THEN Write.Char(ch) END
      END;

    ELSIF zustand = InWort THEN

      IF (ch = ASCII.sp) OR (ch = ASCII.tab) OR (ch = ASCII.nl) THEN
        zustand := NichtInWort;
        Write.Char(ch);
      ELSE
        zustand := InWort; (* ueberfluessig *)
        INC(i);
        IF i <= MAXLEN THEN Write.Char(ch) END
      END;

    ELSE
      Write.StringS(Streams.stderr, "Illegaler Zustand");
      Write.LnS(Streams.stderr);
    END;

    Read.Char(ch);
  END;
END WordCut.
```

2 Wortverdreher (10 Punkte)

Nun dürfen Sie ein Oberon-Programm `Invert` schreiben, das Zeichen für Zeichen von der Standardeingabe liest. Whitespaces sollen unverändert ausgegeben werden. Worte sollen umgedreht ausgegeben werden, d. h. zunächst das letzte Zeichen des Wortes ausgeben, dann das vorletzte und zuletzt das erste Zeichen des Wortes. Sie können annehmen, dass Worte die maximale Länge `MAXLEN` besitzen. Diese Konstante sollten Sie in Ihrem Programm definieren (z. B. mit dem Wert 80). (Hinweis: Während Sie ein Wort lesen, sollten Sie die einzelnen Zeichen in einem Array speichern. Dieses Array kann die feste Länge `MAXLEN` haben. Sollte ein Wort größerer Länge vorkommen, so geben Sie eine Fehlermeldung aus. Die Grundstruktur dieses Programmes ist gleich der Grundstruktur des Programmes in der vorigen Aufgabe.)

Lösung:

Zunächst können wir den folgenden Automaten konstruieren. Bei jedem Übergang ist in spitzen Klammern das Oberon-Progrämmchen angegeben, das in diesem Fall ausgeführt werden soll.



Abk.:	**WS = [<space><tab><newline>]**
	^WS = [^<space><tab><newline>]
	EOF = Dateiende

```
DEFINITION Invert;
END Invert.
```

```
MODULE Invert;

  IMPORT Read, Write, Streams, ASCII;

  CONST NichtInWort = 1;     (* Zustand *)
```

```
      InWort = 2;          (* Zustand *)
      MAXLEN = 80;         (* maximale Wortlaenge (fuers Array) *)

  VAR ch : CHAR;                    (* gelesenes Zeichen *)
      zustand : INTEGER;            (* aktueller Zustand *)
      wort : ARRAY MAXLEN OF CHAR;  (* bisher gelesenes Wort *)
      i : INTEGER;                  (* Index des letzten Zeichens im Wort *)

BEGIN
  (* am Anfang Zustand NichtInWort, da bisher leeres Wort gelesen
   * und Index i zeigt noch auf kein Zeichen;
   * also -1, da erstes Zeichen bei 0 *)
  zustand := NichtInWort;
  i := -1;

  Read.Char(ch);
  WHILE ~ Streams.stdin.eof DO

    IF zustand = NichtInWort THEN

      IF (ch = ASCII.sp) OR (ch = ASCII.tab) OR (ch = ASCII.nl) THEN
        zustand := NichtInWort; (* ueberfluessig *)
        Write.Char(ch);
      ELSE
        zustand := InWort;
        INC(i);
        wort[i] := ch;
      END;

    ELSIF zustand = InWort THEN

      IF (ch = ASCII.sp) OR (ch = ASCII.tab) OR (ch = ASCII.nl) THEN
        zustand := NichtInWort;
        WHILE i >= 0 DO
          Write.Char(wort[i]);
          DEC(i);
        END;
        Write.Char(ch);
      ELSE
        zustand := InWort; (* ueberfluessig *)
        INC(i);
        IF i = MAXLEN THEN
          Write.StringS(Streams.stderr, "Zu langes Wort");
          Write.LnS(Streams.stderr);
          DEC(i);
        END;
```

```
        wort[i] := ch;
      END;

    ELSE
      Write.StringS(Streams.stderr, "Illegaler Zustand");
      Write.LnS(Streams.stderr);
    END;

    Read.Char(ch);
  END;

  (* zum Schluss alle Zeichen ausgeben, die sich noch im Array befinden *)
  IF zustand = InWort THEN (* IF ist ueberfluessig; WHILE genuegt *)
    WHILE i >= 0 DO
      Write.Char(wort[i]);
      DEC(i);
    END;
  END;
END Invert.
```