xxxUtf8

```
DEFINITION xxxUtf8;

   IMPORT Files;

   CONST
      sequenceTooLong = 4;
      unexpectedEnd = 2;
      wrongFollowingOctet = 3;
      wrongLeadingOctet = 1;

   TYPE
      CharDecoder = RECORD
         state-: INTEGER;
         ch-: CHAR;
         (VAR d: CharDecoder) HandleOctet (b: SHORTCHAR), NEW;
         (VAR d: CharDecoder) Start, NEW;
         (VAR d: CharDecoder) Stop, NEW
      END;

   PROCEDURE DecodeArrayToChars (IN utf: ARRAY OF SHORTCHAR; beg, end: INTEGER; VAR
chars: ARRAY OF CHAR; VAR charsPos: INTEGER; OUT error, errPos: INTEGER);
   PROCEDURE DecodeFileToChars (utf: Files.Reader; end: INTEGER; VAR chars: ARRAY OF
CHAR; VAR charsPos: INTEGER; OUT error: INTEGER; OUT errPos: INTEGER);
   PROCEDURE EncodeCharsToArray (IN chars: ARRAY OF CHAR; beg, end: INTEGER; VAR utf:
ARRAY OF SHORTCHAR; VAR pos: INTEGER);
   PROCEDURE EncodeStringToFile (wr: Files.Writer; IN str: ARRAY OF CHAR);

END xxxUtf8.
```

Работа с utf-8.

CONST **sequenceTooLong**, **unexpectedEnd**, **wrongFollowingOctet**, **wrongLeadingOctet**
Ошибочные состояния декодера, соответственно:
- последовательность октетов слишком длинная (чтобы результат декодирования уместился в CHAR)
- неожиданный конец последовательности
- неверный последующий октет
- неверный ведущий октет

TYPE **CharDecoder**
Декодер последовательности utf-8 октетов в литеры КП (CHAR).

Общая схема использования:

```
VAR   d: CharDecoder; rd: ByteSequenceReader; b: SHORTCHAR;
BEGIN
   d.Start; rd.ConnectTo(myByteSequnce); rd.ReadSChar(b);
   WHILE ~rd.eos & (d.state = 0) DO
      REPEAT
         d.HandleOctet(b); rd.ReadSChar(b)
      UNTIL (d.state >= 0) OR rd.eos;
      IF d.state = 0 THEN
         HandleChar(d.ch)
      END
   END;
   d.Stop; IF d.state > 0 THEN HandleError(d.state) END
```

END

Здесь ByteSequenceReader - гипотетический считыватель байт, выставляющий rd.eos после попытки читать из окончившейся последовательности.

state-: INTEGER
Состояние декодера.
  state = 0 - готов к обработке следующего октета
  state < 0 - ожидаются последующие октеты
  state > 0 - ошибка

ch-: CHAR
Декодированная литера.

PROCEDURE (VAR d: CharDecoder) **Start**
NEW
Запуск декодера.

Постусловия
d.state = 0
d.ch = 0X

PROCEDURE (VAR d: CharDecoder) **Stop**
NEW
Остановка декодера.

Постусловия
d.state' < 0
  d.state = unexpectedEnd

PROCEDURE (VAR d: CharDecoder) **HandleOctet** (b: SHORTCHAR)
NEW
Декодирование очередного октета *b*.

Предусловия
d.state <= 0 20

Постусловия
d.state = 0
  b оказался последним октетом
  d.ch - декодированная литера
d.state < 0
  b обработан и это не последний октет
d.state > 0
  b - недопустимый октет

PROCEDURE **DecodeArrayToChars** (IN utf: ARRAY OF SHORTCHAR; beg, end: INTEGER; VAR chars: ARRAY OF CHAR; VAR charsPos: INTEGER; OUT error, errPos: INTEGER)
Декодирует последовательность литер из последовательности utf-8 октетов [*beg*, *end*) в *utf*, сохраняя её в *chars* начиная с *charPos*.

Постусловия
error = 0
  успешно
error > 0
  ошибка - см. ошибочные состояния декодера
errPos индекс октета, следующего за последним обработанным
charsPos конец результирующей последовательности [charsPos', charsPos) литер

PROCEDURE **EncodeCharsToArray** (IN chars: ARRAY OF CHAR; beg, end: INTEGER; VAR utf: ARRAY OF SHORTCHAR; VAR pos: INTEGER)
Кодирует последовательность литер [*beg*, *end*) из *chars* в последовательность utf-8 октетов в *utf*, начинающуюся с *pos*.

Постусловия
pos конец результирующей последовательности [pos', pos) октетов

PROCEDURE **DecodeFileToChars** (utf: Files.Reader; end: INTEGER; VAR chars: ARRAY OF CHAR; VAR charsPos: INTEGER; OUT error: INTEGER; OUT errPos: INTEGER)
Декодирует последовательность литер из последовательности utf-8 октетов [*utf.Pos()*, *end*) в *utf.Base()*, сохраняя её в *chars* начиная с *charPos*.

Постусловия
error = 0
  успешно
error > 0
  ошибка - см. ошибочные состояния декодера
errPos положение октета, следующего за последним обработанным
charsPos конец результирующей последовательности [charsPos', charsPos) литер

PROCEDURE **EncodeStringToFile** (wr: Files.Writer; IN str: ARRAY OF CHAR)
Кодирует цепочку литер *str* в последовательность utf-8 октетов, сохраняемую в *wr.Base(),* начиная с текущего положения *wr*.