Burt Rosenberg

Answer Set 3

Out: 21 September, 1993

```
program linkedList( input, output ) ;
{
  An answer to homework 3, Math 220/317
  18 Sep 1993
  University of Miami
  Prof. Burt Rosenberg
}

{ Notes:
    Part of this file deals with MyStringType, an array of
      characters where the end-of-string is signaled by
      an EOS character. A predicate testing a string to be empty,
      a procedure to write a string to the console,
      and a group of three subroutines to get a word from
      a given file are given.

      Because getting a word is complicated, it is broken in
      two levels of detail. The function oneWordAux does a lot
      of work but gets to assume (1) there will be no eof's in
      the way and (2) that multiple delimiters will be treated
      as a series of zero-length words. The wrapper function called
      oneWord is responsible for checking eofs before calling
      oneWordAux and for throwing out the zero-length words.

    The list stuff is relatively straight forward. Using a dummy
      header means that ListAddAtHead can be a procedure, rather
      than a function.
}

const
  STRLEN = 50 ;
  EOS = chr(0) ;
  InFile = 'test.txt' ;

type

  MyStringType = array [0..STRLEN-1] of char ;
  { A string is an array terminated by an EOS }
```

```
  ListData = MyStringType ;
  List = ^ListNode ;
  ListNode = record
      data : ListData ;
      next : List ;
      end ;

{============== string stuff ============}

function EmptyString( s : MyStringType ) : boolean ;
{
   Test if s is the empty string ;
}
begin
  EmptyString := (s[0]=EOS) ;
end ;

procedure StringPrint( s : MyStringType ) ;
{
   Print the given string.
}
var i : integer ;
begin
  i := 0 ;
  while ((s[i]<>EOS) AND (i<STRLEN)) do begin
    write(s[i]) ;
    i := i + 1 ;
  end ;
end ;

function delimiter( ch : char ) : boolean ;
{
  returns true if character ch is a delimiter character.
}
var b : boolean ;
begin
  b := true ;
  if ((ch>='a') AND (ch<='z'))
     then b := false ;
  if ((ch>='A') AND (ch<='Z'))
     then b := false ;
  delimiter := b ;
end ;
```

```
function oneWordAux( var f : TEXT ) : MyStringType ;
{
  Get characters from file f up until the next delimiter,
  or end-of-line, where the delimiters are characters
  for which the function delimiter(ch) returns TRUE.

  Depends on eof(f) being false on entry, and that
  a delimiter separates characters from the eof.

  If the first character on entry is a delimiter,
  return the empty string.
}
var
  a : MyStringType ;
  flag : boolean ;
  i : integer ;
  ch : char ;
begin
  flag := true ;
  i := 0 ;
  { i is the next place in the array to fill.}
  while flag do
    if eoln(f) then begin         {CASE 1: an end of line}
       readln(f) ;
       flag := false
    end else begin
      read( f, ch) ;
      if delimiter(ch) then begin {CASE 2: a delimiter}
        flag := false
      end else begin              {CASE 3: a valid character}
        a[i] := ch ;
        i := i + 1 ;
        if (i=(STRLEN-1))
          then flag := false ;    {avoid overflow of strings}
      end
    end ;
  a[i] := EOS ;
  oneWordAux := a ;
end ;
```

```
function oneWord( var f : TEXT ) : MyStringType ;
{
  wrapper around oneWordAux, returns the next word in
  the file f or an empty string if there is no next
  word.
}
var a : MyStringType ;
begin
  if eof(f) then a[0] := EOS
  else repeat
    a := oneWordAux( f ) ;
    until ((a[0] <> EOS) OR eof(f)) ;
  oneWord := a ;
end ;

{=========== list stuff ============}

function ListCreate : List ;
var p : List ;
begin
  new(p) ;
  p^.data[0] := EOS ; { this is a bad programming practice! }
  p^.next := NIL ;
  ListCreate := p ;
end ;

procedure ListAddAtHead( l : List ; d : ListData ) ;
var p : List ;
begin
  new(p) ;
  p^.data := d ;
  p^.next := l^.next ;
  l^.next := p ;
end ;

procedure ListPrint( l : List ) ;
begin
  while (l^.next<>NIL) do begin
    StringPrint( l^.next^.data ) ;
    writeln ;
    l := l^.next ;
   end ;
end ;
```

```
procedure ListDestroy( l : List ) ;
var p : List ;
begin
  while ( l<> NIL ) do begin
    p := l^.next ;
    Dispose(l) ;
    l := p ;
  end ;
end ;


{============ main ===============}

var
  fin : TEXT ;
  s : MyStringType ;
  l : List ;

begin

  l := ListCreate ;

  {READ }
  reset( fin, InFile ) ;
  s := oneWord( fin ) ;
  while (NOT EmptyString(s)) do begin
    ListAddAtHead( l, s ) ;
    s := oneWord( fin ) ;
  end ;
  close( fin ) ;

  {WRITE }
  ListPrint( l ) ;
  ListDestroy( l ) ;

end.
```

A sample run:

```
impala> cat test.txt
Play: ROMEO AND JULIET.
 Act: ACT II.
Scen: SCENE II.

    ...

      But, soft! what light through yonder window breaks? 2/2/2
      It is the east, and Juliet is the sun!-             2/2/3

    ...
```

```
impala> a.out
sun
the
is
Juliet
and
east
the
is
It
breaks
window
yonder
through
light
```

```
what
soft
But
II
SCENE
Scen
II
ACT
Act
JULIET
AND
ROMEO
Play
impala>
```