4.2.2.3 Mise en œuvre par curseur

Exemple : J U S S I E U

Tête		Val		Sv
5	1	U	1	9
Libre	2	U	2	0
4	3	I	3	7
	4		4	6
	5	J	5	1
	6		6	8
	7	E	7	2
	8		8	0
	9	S	9	10
	10	S	10	3

N.B. : on peut manipuler des ***cellules***, identifiées par un entier, des ***valeurs***, correspondant à l'informations stockée dans la liste (ici caractère), et des ***positions***, c'est-à-dire le numéro d'ordre d'un élément dans la liste.

Primitives nécessaires

P/F	*Nom*	*Arguments*	*Résultat*
P	`InitListe`	(var) l	/
F	`LgrListe`	l	position
F	`EltListe`	l, position	E
P	`InserListe`	(var) l, e, p	/
P	`SupprListe`	(var) l, p	/
	Gestion des places libres		
P	`InitLibre`	(var) l	/
F	`newLibre`	(var) l	cellule
P	`freeLibre`	(var) l	/

Définitions générale

```
const Max = 10 ;                    (* limité à 10 dans notre exemple *)
const Nil =  0 ;                    (* pas de suivant *)
type cellule = 0..Max ;
type position = 1..Max ;
type valeur = char ;                (* Correspond au type E *)

type Liste = record
      tete : cellule ;
      val : array[1..Max] of valeur ;
      sv : array[1..Max] of cellule ;
      libre : cellule ;
     end ;
```

Code

```
(*****************************************)
(*      procédures d'initialisation      *)
(*****************************************)
(* Au départ la liste est vide *)
procedure InitListe(var l : Liste) ;
begin
  l.tete := Nil ;
  InitLibre(l) ;
end ;
```

```
(* Au départ, toutes les cases sont     *)
(* libres : on les chaîne toutes dans   *)
(* l'ordre                              *)
procedure InitLibre(var l : Liste) ;
var i : integer ;
begin
  l.libre := 1 ;
  for i:=1 to Max-1 do
    l.sv[i] := i+1 ;
  l.sv[Max] := Nil ;
end ;
```



```
(*****************************************)
(*            longueur                   *)
(*****************************************)
(* Parcours nécessaire                   *)
function LgrListe(l : Liste) : integer ;
var i,c : integer ;
begin
  i := l.tete ; c := 0 ;
  while i <> Nil do begin
    c := c+1 ;
    i := l.sv[i] ;
  end ;
  longueur := c ;
end ;

(*****************************************)
(*       insert, delete, lookup          *)
(*****************************************)
function EltListe(l : Liste,
                  p : position) : valeur ;
var i, n : integer ;
begin
  i := l.tete ; n := 0 ;
  while ((i <> Nil) and (n < p)) do begin
    i := l.sv[i] ;
    n := n + 1 ;
  end ;
  if (n < p) then ERREUR ;
  else EltListe := l.val[c] ;
end ;

procedure InserListe(var l : Liste,
                     p : position,
                     e : valeur) ;
var i, n : integer ;
begin
(* cas particulier de l'insertion en tête *)
  if p = 1 then begin
    c := newLibre(l) ;
    c.val := e ;
    c.sv := l.tete ;
    l.tete := c ;
  end
  else begin (* cas général : parcours *)
    i := l.tete ;
    n := 0 ;
    while ((i <> Nil) and (n < p-1)) do begin
      i := l.sv[i] ;
      n := n+1 ;
    end ;
    if (n<p) then ERREUR
    else begin
      c := newLibre(l) ;
      c.val := e ;
      c.sv := l.sv[i] ;
      l.sv[i] := c ;
    end ;
  end ;
end ;
```

```
procedure SupprListe(var l : Liste,
                     p : position) ;
var i, n : integer ;
    tmp : cellule ;
begin
(* suppression en tête *)
  if p = 1 then begin
    tmp := l.tete ;
    l.tete := l.sv[l.tete] ;
    freeLibre(tmp) ;
  end
  else begin (* cas général : parcours *)
    i := l.tete ;
    n := 0 ;
    while ((i <> Nil) and (n < p-1)) do begin
      i := l.sv[i] ;
      n := n+1 ;
    end ;
    if (n<p) then ERREUR
    else begin
      tmp := l.tete ;
      l.sv[i] := l.sv[l.sv[i]] ;
      freeLibre(l,tmp) ;
    end ;
  end ;
end ;

(*****************************************)
(*   Gestion des places libres           *)
(*****************************************)
(* Il s'agit simplement d'insertions et  *)
(* de suppressions en tête               *)

function newLibre(var l: Liste) : cellule ;
var tmp : cellule ;
begin
  if l.libre = 0 then ERREUR
  else begin
    tmp := l.libre ;
    l.libre := l.sv[l.libre] ;
    newLibre := tmp ;
  end ;
end ;

procedure freeLibre(var l: Liste, c:cellule) ;
begin
  l.sv[c] := l.libre ;
  l.libre := c ;
end ;
```