```
 1: unit ConjuntoDinamico;
 2: interface
 3:     type
 4:        tInfo=char;
 5:        tConjunto=^tNodo;
 6:        tNodo=record
 7:          info:tInfo;
 8:          sig:tConjunto;
 9:        end;
10:
11:     procedure conjuntoVacio(var conjunto:tConjunto);
12:     function esConjuntoVacio(conjunto:tConjunto):boolean;
13:     function pertenece(info:tInfo; conjunto:tConjunto):boolean;
14:     procedure insertar(info:tInfo; var conjunto:tConjunto);
15:     function cardinal(conjunto:tConjunto):integer;
16:     procedure interseccion(conjunto1, conjunto2:tConjunto; var
    conjuntonuevo:tConjunto);
17:     procedure union(conjunto1, conjunto2:tConjunto; var
    conjuntonuevo:tConjunto);
18:     function conjuntoToStr(conjunto: tConjunto):string;
19:
20:
21: implementation
22:
23:     const
24:       nulo=nil;
25:     type
26:       tPos=tConjunto;
27:
28:     function CreaNodo(info:tInfo):tConjunto;
29:     {Objetivo: Crea un nuevo nodo en el conjunto con el dato info
30:      Entrada: Info
31:      Salida : un nuevo conjunto
32:     }
33:     var
34:       nuevo:tPos;
35:     begin
36:       new(nuevo);
37:       nuevo^.info:=info;
38:       nuevo^.sig:=nulo;
39:       CreaNodo:=nuevo;
40:     end;
41:
42:     function primerElemento(conjunto:tConjunto):tPos;
43:     {Objetivo: Determina la posicion del primer elemento en el conjunto
44:      Entrada: Un conjunto
45:      Salida:  La posicion del elemento en el conjunto
46:     }
47:     begin
48:       primerElemento:=conjunto;
49:     end;
50:
51:     function siguienteElemento(conjunto:tConjunto;p: tPos):tPos;
52:     {Objetivo: Devuelve la posicion del elemneto siguiente al que se pasa como
    argumento
53:      Entrada:  conjunto,p
54:      Salida:   siguiente elemento a p
55:     }
```

```
56:     begin
57:        if esConjuntoVacio(conjunto) then
58:           siguienteElemento:=nulo
59:        else
60:           siguienteElemento:=p^.sig;
61:     end;
62:
63:
64:     procedure conjuntoVacio(var conjunto:tConjunto);
65:     {Objetivo: Crear un conjunto vacio
66:      Entrada:
67:      Salida:  Un conjunto vacio
68:     }
69:     begin
70:       conjunto:=nulo;
71:     end;
72:
73:
74:     function esConjuntoVacio(conjunto:tConjunto):boolean;
75:     {Objetivo: Determinar si un conjunto está vacio
76:      Entrada:  conjunto
77:      Salida:   True si el conjunto esta vacio, false en caso contrario
78:      Precond:  El conjunto debe estar inicializado
79:     }
80:     begin
81:       esConjuntoVacio:=(conjunto=nulo);
82:     end;
83:
84:
85:     function pertenece(info:tInfo; conjunto:tConjunto):boolean;
86:     {Objetivo: Determina la pertenencia de un elemento a un conjunto
87:      Entrada:  Un elemento y un conjunto
88:      Salida:   True si el elemento pertenece a Conjunto, false en caso contrario
89:      Precond:  El conjunto debe estar inicializado
90:     }
91:     var
92:       auxc:tPos;
93:     begin
94:       auxc:=primerElemento(conjunto);
95:       while (auxc<>nulo) and (auxc^.info<>info) do
96:          auxc:=siguienteElemento(conjunto,auxc);
97:
98:        pertenece:=(auxc<>nulo);
99:     end;
100:
101:    procedure insertar(info:tInfo; var conjunto:tConjunto);
102:    {Objetivo: Añade un elemento a un conjunto
103:     Entrada:  El elemento a insertar y el conjunto
104:     Salida:   Conjunto con el elemento insertado
105:    }
106:    var
107:      temp: tPos;
108:    begin
109:       if not pertenece(info, conjunto) then
110:       begin
111:          temp:=CreaNodo(info);
112:          temp^.sig:=conjunto;
113:          conjunto:=temp;
```

```
114:        end;
115:     end;
116:
117:
118:     function cardinal(conjunto:tConjunto):integer;
119:     {Objetivo: Cuenta el número de elementos del conjunto
120:      Entrada:  Conjunto
121:      Salida:   Número de elementos
122:     }
123:     var
124:       auxc:tPos;
125:       i:integer;
126:     begin
127:       i:=0;
128:       auxc:=primerElemento(conjunto);
129:       while (auxc<>nulo) do
130:       begin
131:          i:=i+1;
132:          auxc:=siguienteElemento(conjunto,auxc);
133:       end;
134:       cardinal:=i;
135:     end;
136:
137:
138:     procedure interseccion(conjunto1, conjunto2:tConjunto; var
    conjuntonuevo:tConjunto);
139:     {Objetivo: Realiza la interseccion de dos conjuntos
140:      Entrada:  Dos Conjuntos
141:      Salida:   Un nuevo conjunto resultado de la interseccion de los conjuntos
    de entrada
142:      Precond:  Los conjuntos deben estar inicializados y se supone memoria
    suficiente
143:     }
144:     var
145:       aux1: tPos;
146:     begin
147:       conjuntoVacio(conjuntonuevo);
148:       if not esConjuntoVacio(conjunto1) and
149:          not esConjuntoVacio(conjunto2) then
150:       begin
151:         aux1:=primerElemento(conjunto1);
152:         while (aux1<>nulo) do
153:         begin
154:           if pertenece(aux1^.info, conjunto2) then
155:             insertar(aux1^.info, conjuntonuevo);
156:           aux1:=siguienteElemento(conjunto1, aux1);
157:         end;
158:       end;
159:     end;
160:
161:
162:     procedure union(conjunto1, conjunto2:tConjunto; var
    conjuntonuevo:tConjunto);
163:     {Objetivo: Realiza la union de dos conjuntos
164:      Entrada:  Dos Conjuntos
165:      Salida:   Un nuevo conjunto resultado de la union de los conjuntos de
    entrada
166:      Precond:  Los conjuntos deben estar inicializados y se supone memoria
    suficiente
```

```
167:     }
168:     var
169:       auxc:tPos;
170:     begin
171:       conjuntoVacio(conjuntonuevo);
172:       auxc:=primerElemento(conjunto1);
173:       while (auxc<>nulo) do
174:       begin
175:          insertar(auxc^.info, conjuntonuevo);
176:          auxc:=auxc^.sig;
177:       end;
178:       auxc:=primerElemento(conjunto2);
179:       while auxc<>nulo do
180:       begin
181:            insertar(auxc^.info, conjuntonuevo);
182:       end;
183:     end;
184:
185:     function conjuntoToStr(conjunto: tConjunto):string;
186:     {Objetivo: Devuelve el contenido de un conjunto
187:      Entrada:  Conjunto
188:      Salida:   String, con los elementos del conjunto separados por un espacio
189:     }
190:     var
191:       auxc:tPos;
192:       cadena:string;
193:     begin
194:       cadena:='';
195:       auxc:=primerElemento(conjunto);
196:       while (auxc<>nulo) do
197:       begin
198:         cadena:=cadena + auxc^.info;
199:         auxc:=siguienteElemento(conjunto, auxc);
200:         if auxc<>nulo then
201:            cadena := cadena + ' ';
202:       end;
203:       conjuntoToStr:=cadena;
204:     end;
205:
206:
207: end.
```