

```

1 {
2 Autor : Slawek Kolasinski
3 Data : 02.2009
4 e-mail: skola@mimuw.edu.pl
5 }
6
7 Program SLLists;
8 type
9     typ      = integer;
10    lista   = ^element;
11    element = record
12        dane : typ;
13        ogon : lista;
14    end;
15
16 { Inicjuje liste }
17 procedure inicjuj(var l : lista);
18 begin
19     l := nil;
20 end; { inicjuj }
21
22 { Sprawdza czy lista jest pusta }
23 function pusta(l : lista) : boolean;
24 begin
25     pusta := (l = nil);
26 end; { pusta }
27
28 { Zwraca wartosc zapisana w pierwszym elemencie listy
29   ale nie modyfikuje listy }
30 function glowa(l : lista) : typ;
31 begin
32     if not pusta(l) then
33         glowa := l^.dane
34     else
35         begin
36             writeln('glowa: pusta lista!');
37             halt;
38         end;
39 end; { glowa }
40
41 { Zwraca ogon listy, tzn. cala liste bez pierwszego elementu.
42   Nie modyfikuje listy. }
43 function ogon(l : lista) : lista;
44 begin
45     if not pusta(l) then
46         ogon := l^.ogon
47     else
48         begin
49             writeln('ogon: pusta lista!');
50             halt;
51         end;
52 end; { ogon }
53
54 { Usuwa pierwszy element z listy }
55 function usun(var l : lista) : typ;
56 var
57     czubek : lista;
58 begin
59     usun := glowa(l);
60     czubek := l;
61     l := ogon(l);
62     dispose(czubek);
63 end; { usun }
64
65 { Dodaje na poczatek listy nowy element }
66 procedure dodaj(var l : lista; x : typ);
67 var
68     elt : lista;
69 begin
70     new(elt);
71     elt^.dane := x;
72     elt^.ogon := l;
73     l := elt;
74 end; { dodaj }
75
76 { Podlacza liste l2 na koncu listy l1.
77   Lista l1 jest wydłużana o elementy listy l2.
78   Lista l2 jest opróżniana. }
79 procedure polacz(var l1, l2 : lista);
80 var

```

```

81 aktualny, poprzedni : lista;
82 begin
83     if not pusta(l1) then
84         begin
85             poprzedni := nil;
86             aktualny := l1;
87             while not pusta(aktualny) do
88                 begin
89                     poprzedni := aktualny;
90                     aktualny := ogon(aktualny);
91                 end;
92             poprzedni^.ogon := l2;
93         end
94     else
95         l1 := l2;
96     inicjuj(l2);
97 end; { polacz }
98
99 { Scala dwie uporządkowane rosnaco listy.
100   W wyniku tworzy nowa liste. }
101 function scal(l1,l2 : lista) : lista;
102 var
103     wsk1,wsk2,wynik,rezsta : lista;
104     dane1,dane2           : typ;
105 begin
106     wsk1 := l1;
107     wsk2 := l2;
108     inicjuj(wynik);
109
110     while (not pusta(wsk1)) and (not pusta(wsk2)) do
111         begin
112             dane1 := glowa(wsk1);
113             dane2 := glowa(wsk2);
114             if dane1 < dane2 then
115                 begin
116                     dodaj(wynik, dane1);
117                     wsk1 := ogon(wsk1);
118                 end else begin
119                     dodaj(wynik, dane2);
120                     wsk2 := ogon(wsk2);
121                 end
122             end;
123
124         if not pusta(wsk1) then
125             rezsta := wsk1
126         else
127             rezsta := wsk2;
128
129         while not pusta(rezsta) do
130             begin
131                 dodaj(wynik, glowa(rezsta));
132                 rezsta := ogon(rezsta);
133             end;
134
135         scal := wynik;
136     end; { scal }
137
138 { Sortuje liste rosnaco algorymem sortowania przez wstawianie. }
139 procedure sortuj(var l : lista);
140 var
141     max, maxpop, akt, pop, wynik, rezsta : lista;
142
143     procedure znajdz_max;
144     begin
145         pop := nil;
146         akt := rezsta;
147
148         maxpop := pop;
149         max := akt;
150
151         while not pusta(akt) do
152             begin
153                 if glowa(akt) > glowa(max) then
154                     begin
155                         max := akt;
156                         maxpop := pop;
157                     end;
158                 pop := akt;
159                 akt := ogon(akt);
160             end;

```

```

161     end;
162
163 begin
164     if pusta(l) then exit;
165
166     reszta := l;
167     inicjuj(wynik);
168
169     while not pusta(reszta) do
170         begin
171             znajdz_max;
172
173             if pusta(maxpop) then
174                 reszta := ogon(reszta)
175             else
176                 maxpop^.ogon := ogon(max);
177
178             max^.ogon := wynik;
179             wynik := max;
180         end;
181
182     l := wynik;
183 end; { sortuj }
184
185 { Usuwa liste z pamieci. }
186 procedure kasuj(var l : lista);
187 var
188     ignoruj : typ;
189 begin
190     while not pusta(l) do ignoruj := usun(l);
191 end; { kasuj }
192
193 { Drukuje na ekranie cala liste. }
194 procedure drukuj(l : lista);
195 var
196     wsk : lista;
197 begin
198     wsk := l;
199     if not pusta(wsk) then
200         begin
201             write(wsk^.dane);
202             wsk := ogon(wsk);
203         end;
204     while not pusta(wsk) do
205         begin
206             write(' ', wsk^.dane);
207             wsk := ogon(wsk);
208         end;
209 end; { drukuj }
210
211 var
212     l1,l2,l3 : lista;
213
214 begin
215     inicjuj(l1);
216     inicjuj(l2);
217
218     dodaj(l1, 20);
219     dodaj(l2, -90);
220     dodaj(l1, 13);
221     dodaj(l2, 27);
222     dodaj(l1, 56);
223     dodaj(l2, 1003);
224     dodaj(l1, 46);
225     dodaj(l2, -23);
226     dodaj(l1, -6);
227     dodaj(l2, 67);
228
229     dodaj(l1, -75);
230     dodaj(l2, 44);
231     dodaj(l1, 6345);
232     dodaj(l2, 313);
233     dodaj(l1, 768);
234     dodaj(l2, 2213);
235     dodaj(l1, 868);
236     dodaj(l2, -156);
237     dodaj(l1, 9467);
238     dodaj(l2, -230);
239
240     write('l1 = '); drukuj(l1); writeln('|');

```

```

241     write('l2 = '); drukuj(l2); writeln('|');
242
243     sortuj(l1);
244     write('posortowana l1 = '); drukuj(l1); writeln('|');
245
246     sortuj(l2);
247     write('posortowana l2 = '); drukuj(l2); writeln('|');
248
249     l3 := scal(l1,l2);
250     write('po scaleniu l3 = '); drukuj(l3); writeln('|');
251
252     polacz(l1,l2);
253     write('po polaczeniu l1 = '); drukuj(l1); writeln('|');
254     write('po polaczeniu l2 = '); drukuj(l2); writeln('|');
255
256     kasuj(l1);
257     kasuj(l2);
258
259     write('po skasowaniu l1 = '); drukuj(l1); writeln('|');
260     write('po skasowaniu l2 = '); drukuj(l2); writeln('|');
261 end.
262
263 {
264
265 Wynik dzialania programu na ekranie:
266
267 l1 = |9467 868 768 6345 -75 -6 46 56 13 20|
268 l2 = |-230 -156 2213 313 44 67 -23 1003 27 -90|
269 posortowana l1 = |-75 -6 13 20 46 56 768 868 6345 9467|
270 posortowana l2 = |-230 -156 -90 -23 27 44 67 313 1003 2213|
271 po scaleniu l3 = |9467 6345 2213 1003 868 768 313 67 56 46 44 27 20 13 -6 -23 -75 -90 -156 -230|
272 po polaczeniu l1 = |-75 -6 13 20 46 56 768 868 6345 9467 -230 -156 -90 -23 27 44 67 313 1003
273 po polaczeniu l2 = ||
274 po skasowaniu l1 = ||
275 po skasowaniu l2 = ||
276
277 }

```