```
{
Autor : Slawek Kolasinski
Data  : 04.2009
e-mail: skola@mimuw.edu.pl
}
Program Graph;
const
   N = 10;
type
   { typ danych w grafie }
   gtyp    = record
                odwiedzony : boolean;
                numer      : integer;
             end;
   { typ danych na liscie }
   ltyp    = integer;
   lista   = ^element;
   wezel   = record
                dane     : gtyp;
                sasiedzi : lista;
             end;
   element = record
                glowa : ltyp;
                ogon  : lista;
             end;
   graf    = array[1..N] of wezel;
   stos    = lista;
   kolejka = record
                pierwszy : lista;
                ostatni  : lista;
             end;

(********** OBSLUGA OPERACJI NA LISTACH  **********)
procedure l_inicjuj(var l : lista);
begin
   l := nil;
end; { l_inicjuj }

function l_pusta(l : lista) : boolean;
begin
   l_pusta := (l = nil);
end; { l_pusta }

function l_glowa(l : lista) : ltyp;
begin
   if not l_pusta(l) then
      l_glowa := l^.glowa
   else begin
      writeln('glowa: pusta lista!');
      halt;
   end;
end; { l_glowa }

function l_ogon(l : lista) : lista;
begin
   if not l_pusta(l) then
      l_ogon := l^.ogon
   else begin
      writeln('ogon: pusta lista!');
      halt;
```



```
   end;
end; { l_ogon }

function l_usun(var l : lista) : ltyp;
var
   czubek : lista;
begin
   l_usun := l_glowa(l);
   czubek := l;
   l := l_ogon(l);
   dispose(czubek);
end; { l_usun }

procedure l_kasuj(var l : lista);
var
   ignoruj : ltyp;
begin
   while not l_pusta(l) do ignoruj := l_usun(l);
end; { l_kasuj }

procedure l_drukuj(l : lista);
begin
   if not l_pusta(l) then begin
      write(l_glowa(l));
      l := l_ogon(l);
   end;
   while not l_pusta(l) do begin
      write(' ', l_glowa(l));
      l := l_ogon(l);
   end;
end; { l_drukuj }

procedure l_dodaj_przed(var l : lista; x : ltyp);
var
   elt : lista;
begin
   new(elt);
   elt^.glowa := x;
   elt^.ogon := l;
   l := elt;
end; { l_dodaj_przed }

procedure l_dodaj_za(var l : lista; x : ltyp);
var
   elt : lista;
begin
   if l_pusta(l) then l_dodaj_przed(l,x)
   else begin
      new(elt);
      elt^.glowa := x;
      elt^.ogon := l^.ogon;
      l^.ogon := elt;
   end;
end; { l_dodaj_za }

function l_znajdz(l : lista; x : ltyp) : boolean;
begin
   while (not l_pusta(l)) and (l_glowa(l) <> x) do l := l_ogon(
                                                          l);
   l_znajdz := not l_pusta(l);
```

```
120 end; { l_znajdz }

122 function l_dlugosc(l : lista) : integer;
123 var
124    licznik : integer;
125 begin
126    licznik := 0;
127    while not l_pusta(l) do begin
128       licznik := licznik + 1;
129       l := l_ogon(l);
130    end;
131    l_dlugosc := licznik;
132 end; { l_dlugosc }

134 (********** OBSLUGA OPERACJI NA STOSIE  **********)
135 procedure s_inicjuj(var s : stos);
136 begin
137    l_inicjuj(s);
138 end; { s_inicjuj }

140 procedure s_kasuj(var s :stos);
141 begin
142    l_kasuj(s);
143 end; { s_kasuj }

145 procedure s_push(var s : stos; x : ltyp);
146 begin
147    l_dodaj_przed(s,x);
148 end; { s_push }

150 function s_pop(var s : stos) : ltyp;
151 begin
152    s_pop := l_usun(s);
153 end; { s_pop }

155 function s_top(s : stos) : ltyp;
156 begin
157    s_top := l_glowa(s);
158 end; { s_top }

160 function s_pusty(s : stos) : boolean;
161 begin
162    s_pusty := l_pusta(s);
163 end; { s_pusty }

165 (********** OBSLUGA OPERACJI NA KOLEJCE  **********)
166 procedure q_inicjuj(var q : kolejka);
167 begin
168    l_inicjuj(q.pierwszy);
169    q.ostatni := q.pierwszy;
170 end; { q_inicjuj }

172 procedure q_kasuj(var q : kolejka);
173 begin
174    l_kasuj(q.pierwszy);
175    q.ostatni := q.pierwszy;
176 end; { q_kasuj }

178 function q_pusta(q : kolejka) : boolean;
179 begin
```



```
180    q_pusta := l_pusta(q.pierwszy) and l_pusta(q.ostatni);
181 end; { q_pusta }

183 procedure q_enqueue(var q : kolejka; x : ltyp);
184 begin
185    if q_pusta(q) then begin
186       l_dodaj_przed(q.pierwszy,x);
187       q.ostatni := q.pierwszy;
188    end else begin
189       l_dodaj_za(q.ostatni,x);
190       q.ostatni := l_ogon(q.ostatni);
191    end;
192 end; { q_push }

194 function q_dequeue(var q : kolejka) : ltyp;
195 begin
196    q_dequeue := l_usun(q.pierwszy);
197    if l_pusta(q.pierwszy) then q.ostatni := q.pierwszy;
198 end; { q_pop }

200 function q_top(q : kolejka) : ltyp;
201 begin
202    q_top := l_glowa(q.pierwszy);
203 end; { q_top }

205 (********** OBSLUGA OPERACJI NA GRAFACH  **********)
206 procedure g_inicjuj(var g : graf);
207 var
208    i : integer;
209 begin
210    for i := 1 to N do begin
211       l_inicjuj(g[i].sasiedzi);
212       g[i].dane.odwiedzony := false;
213       g[i].dane.numer := i;
214    end;
215 end; { g_inicjuj }

217 procedure g_zeruj_odwiedzone(var g : graf);
218 var
219    i : integer;
220 begin
221    for i := 1 to N do begin
222       g[i].dane.odwiedzony := false;
223    end;
224 end; { g_zeruj_odwiedzone }

226 procedure g_kasuj(var g : graf);
227 var
228    u : integer;
229 begin
230    for u := 1 to N do begin
231       l_kasuj(g[u].sasiedzi);
232    end;
233 end; { g_kasuj }

235 procedure g_dodaj_krawedz(var g : graf; u,v : integer);
236 begin
237    l_dodaj_przed(g[u].sasiedzi, v);
238 end; { g_dodaj_krawedz }
```

```
240 function g_stopien(var g : graf; u : integer) : integer;
241 begin
242    g_stopien := l_dlugosc(g[u].sasiedzi);
243 end; { g_stopien }

245 procedure g_drukuj(var g : graf);
246 var
247    u : integer;
248 begin
249    for u := 1 to N do begin
250       write(u, '(' ,g_stopien(g,u), '): ');
251       l_drukuj(g[u].sasiedzi);
252       writeln;
253    end;
254 end; { g_drukuj }

256 procedure g_z_pliku(nazwa : string; var g : graf);
257 var
258    plik : text;
259    u,v  : integer;
260 begin
261    assign(plik,nazwa);
262    reset(plik);
263    g_inicjuj(g);
264    while not eof(plik) do begin
265       readln(plik, u, v);
266       if (1 <= u) and (1 <= v) and (u <= N) and (v <= N) then
267          g_dodaj_krawedz(g,u,v)
268       else begin
269          writeln('g_z_pliku: Indeks wierzcholka poza zakresem!');
270          halt;
271       end;
272    end;
273    close(plik);
274 end; { g_z_pliku }

276 { Szkielet przeszukiwania w szerz }
277 procedure g_bfs(var g : graf; start : integer);
278 var
279    q      : kolejka;
280    akt,nr : integer;
281    l      : lista;
282 begin
283    q_inicjuj(q);
284    { oznacz wszystkie wezly jako nieodwiedzone }
285    g_zeruj_odwiedzone(g);
286    { wrzuc do kolejki numer wezla startowego }
287    q_enqueue(q,start);
288    g[start].dane.odwiedzony := true;
289    while not q_pusta(q) do begin
290       { wyjmij z kolejki wezel oczekujacy na odwiedzenie }
291       akt := q_dequeue(q);
292       { przejrzyj sasiadow akt i ... }
293       l := g[akt].sasiedzi;
294       while not l_pusta(l) do begin
295          nr := l_glowa(l);
296          { ... wstaw do kolejki tych,
                              ktorzy jeszcze nie byli odwiedzeni }
297          if not g[nr].dane.odwiedzony then begin
298             q_enqueue(q,nr);
```



```
299             g[nr].dane.odwiedzony := true;
300          end;
301          l := l_ogon(l);
302       end;
303       { ... }
304       { PRZETWORZ JAKOS WEZEL O NUMERZE akt }
305       write(akt, ' ');
306       { ... }
307    end;
308 end;

310 { Szkielet przeszukiwania w glab }
311 procedure g_dfs(var g : graf; start : integer);
312 var
313    s      : stos;
314    akt,nr : integer;
315    l      : lista;
316 begin
317    s_inicjuj(s);
318    { oznacz wszystkie wezly jako nieodwiedzone }
319    g_zeruj_odwiedzone(g);
320    { wrzuc na stos numer wezla startowego }
321    s_push(s,start);
322    g[start].dane.odwiedzony := true;
323    while not s_pusty(s) do begin
324       { zdejmij ze stosu wezel oczekujacy na odwiedzenie }
325       akt := s_pop(s);
326       { oznacz go jako juz odwiedzony }
327       g[akt].dane.odwiedzony := true;
328       { przejrzyj sasiadow akt i ... }
329       l := g[akt].sasiedzi;
330       while not l_pusta(l) do begin
331          nr := l_glowa(l);
332          { ... wrzuc na stos tych,
                              ktorzy jeszcze nie byli odwiedzeni }
333          if not g[nr].dane.odwiedzony then begin
334             s_push(s,nr);
335             g[nr].dane.odwiedzony := true;
336          end;
337          l := l_ogon(l);
338       end;
339       { ... }
340       { PRZETWORZ JAKOS WEZEL O NUMERZE akt }
341       write(akt, ' ');
342       { ... }
343    end;
344 end;

346 var
347    g : graf;
348 begin
349    g_z_pliku('graf.txt',g);
350    writeln('graf');
351    g_drukuj(g);

353    write('dfs(1): '); g_dfs(g,1); writeln;
354    write('bfs(1): '); g_bfs(g,1); writeln;

356    g_kasuj(g);
357 end.
```