```
{
Autor : Slawek Kolasinski
Data  : 04.2009
e-mail: skola@mimuw.edu.pl
}
Program Graph;
const
   N = 26;
type
   { typ danych w grafie }
   gtyp    = record
                odwiedzony : boolean;
                numer      : integer;
                odl        : 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);
```

```
120      l_znajdz := not l_pusta(l);
121   end; { l_znajdz }

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

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

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

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

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

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

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

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

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

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



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

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

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

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

206   (********** OBSLUGA OPERACJI NA GRAFACH  **********)
207   procedure g_inicjuj(var g : graf);
208   var
209      i : integer;
210   begin
211      for i := 1 to N do begin
212         l_inicjuj(g[i].sasiedzi);
213         g[i].dane.odwiedzony := false;
214         g[i].dane.odl := 0;
215         g[i].dane.numer := i;
216      end;
217   end; { g_inicjuj }

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

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

237   function g_jest_krawedz(var g : graf; u,v : integer) : boolean;
238   begin
239      g_jest_krawedz := l_znajdz(g[u].sasiedzi, v);
```

```
end; { g_jest_krawedz }

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

procedure g_dodaj_krawedz2(var g : graf; u,v : integer);
begin
   g_dodaj_krawedz(g,u,v);
   if u <> v then g_dodaj_krawedz(g,v,u);
end; { g_dodaj_krawedz2 }

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

procedure g_drukuj(var g : graf);
var
   u : integer;
begin
   for u := 1 to N do begin
      write(u, '(' ,g[u].dane.odl, '): ');
      l_drukuj(g[u].sasiedzi);
      writeln;
   end;
end; { g_drukuj }

procedure g_z_pliku(nazwa : string; var g : graf);
var
   plik : text;
   u,v  : integer;
begin
   assign(plik,nazwa);
   reset(plik);
   g_inicjuj(g);
   while not eof(plik) do begin
      readln(plik, u, v);
      if (1 <= u) and (1 <= v) and (u <= N) and (
                                                v <= N) then begin
         if not g_jest_krawedz(g,u,v) then g_dodaj_krawedz(
                                                   g,u,v);
      end else begin
         writeln('g_z_pliku: Indeks wierzcholka poza zakresem!');
         halt;
      end;
   end;
   close(plik);
end; { g_z_pliku }

procedure g_odleglosci_z(var g : graf; start : integer);
var
   { qa - aktualny, qp - poprzedni }
   qa,qp      : kolejka;
   pop,akt,nr : integer;
   l          : lista;
begin
   q_inicjuj(qa);
   q_inicjuj(qp);
```



```
   g_zeruj_odwiedzone(g);
   q_enqueue(qa,start);
   q_enqueue(qp,0);
   g[start].dane.odwiedzony := true;
   g[start].dane.odl := 0;
   while not q_pusta(qa) do begin
      akt := q_dequeue(qa);
      pop := q_dequeue(qp);
      l := g[akt].sasiedzi;
      while not l_pusta(l) do begin
         nr := l_glowa(l);
         if not g[nr].dane.odwiedzony then begin
            q_enqueue(qa,nr);
            q_enqueue(qp,akt);
            g[nr].dane.odwiedzony := true;
         end;
         l := l_ogon(l);
      end;
      if pop <> 0 then begin
         g[akt].dane.odl := g[pop].dane.odl + 1;
      end;
   end;
end; { g_odleglosci_z }

var
   g : graf;
begin
   g_z_pliku('graf.txt',g);
   g_odleglosci_z(g,1);
   g_drukuj(g);
   g_kasuj(g);
end.
```