PRESEK
List za mlade matematike, fizike, astronome in računalnikarje

ISSN 0351-6652
Letnik **14** (1986/1987)
Številka 3
Strani 145–155

Andrej Vitek:

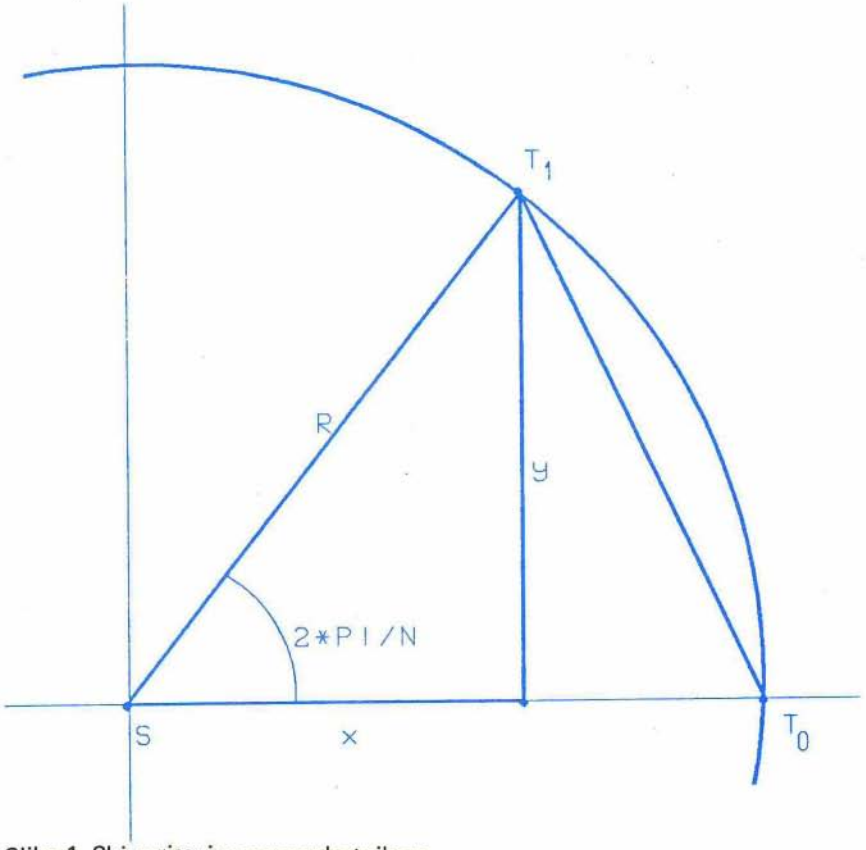
KAKO RAČUNALNIK NARIŠE KROŽNICO

Ključne besede: računalništvo, matematika.

Elektronska verzija: http://www.presek.si/14/831-Vitek.pdf

KAKO RAČUNALNIK NARIŠE KROŽNICO

Zadnjič (Presek **13** (1985/86)1) smo si ogledali računalnikovo ravnilo. Spoznali smo, kako računalnik nariše daljico. Danes si oglejmo še njegovo šestilo, poglejmo torej, kako računalnik riše krožni lok. Preden nadaljujemo, le ponovimo, kakšen je računalnikov papir: velika rešetka raznobarvnih pik, oštevilčenih z dvema celima številoma — številko stolpca in vrstice, v kateri pika stoji. Zato bo tudi krožnica, ki jo bomo narisali, rahlo žagasta. Toda to nas ne bo motilo, kot nas ni motilo zadnjič.



Slika 1. Skica risanja z mnogokotnikom

(Opomba: če trigonometrije ne obvladate, lahko ta odstavek mirno preskočite.)

Sedaj, ko znamo risati daljice, nam najprej pride na misel, da bi poskusili krožnico narisati z daljicami. Po obodu kroga razporedimo enakomerno razmaknjene točke, ki jih med seboj povežemo z daljicami. Tako namesto krožnice narišemo sicer mnogokotnik, ki pa krožnico ponazarja dovolj dobro, če smo le na obod razmetali dovolj točk. Poskusimo določiti koordinate teh točk! V ta namen privzemimo, da se središče kroga ujema z izhodiščem koordinatnega sistema, njegov polmer pa označimo z R. Na obod razporedimo N točk. Točko, v kateri bomo začeli risati, postavimo na os x in jo označimo s T_0, preostale pa naj bodo T_1, ..., T_{N-1}. Tako je središčni kot med zaporednima točkama enak ravno N-temu delu polnega kota, torej (izražen v radianih) $2*PI/N$ (glej sliko 1). Središčni kot med začetnim in k–tim naslednjim oglišcem (T_k) je zato k–krat tolikšen. Koordinati x_k, y_k točke T_k sta tako $x_k = R * \cos(k * 2 * PI/N)$ in $y_k = R * \sin(k * 2 * PI/N)$, kjer je $k = 0, 1, 2, ..., N - 1$. Če se središče krožnice in izhodišče koordinatnega sistema ne ujemata, moramo koordinatama x_k in y_k prišteti še koordinati središča x_S in y_S, tako da ima v splošnem primeru točka T_k koordinati (x_S+x_k, y_S+y_k).

Prepišimo te ugotovitve v postopek za risanje! V ta namen označimo z

R	– polmer kroga,
N	– število stranic mnogokotnika,
xsred, ysred	– koordinati središča krožnice,
xzač,yzač	– koordinati začetne točke trenutno risane stranice,
xkon,ykon	– koordinati končne točke te stranice,
kot	– središčni kot zadnje narisane točke,
korak	– središčni kot med zaporednima ogliščema.

Risanje gre potem takole:

(1) postavi se v začetno oglišče:

*xkon := R, ykon := 0, kot := 0, korak := 2*PI/N*

(2) izračunaj koordinati naslednje točke:

xzač := xkon, yzač := ykon,
kot := kot + korak,
*xkon := R * cos (kot), ykon := R * sin (kot)*

(3) prejšnjo točko poveži z njo:

Daljica (*xzač, yzač, xkon, ykon*)

(4) koraka (2) in (3) ponavljaj, dokler ni narisanih vseh *N* daljic.

Ustrezni podprogram prikazuje program 1. Podprogram Daljica v njem mora v resnici seveda povezati obe podani točki z ravno črto.

Program 1

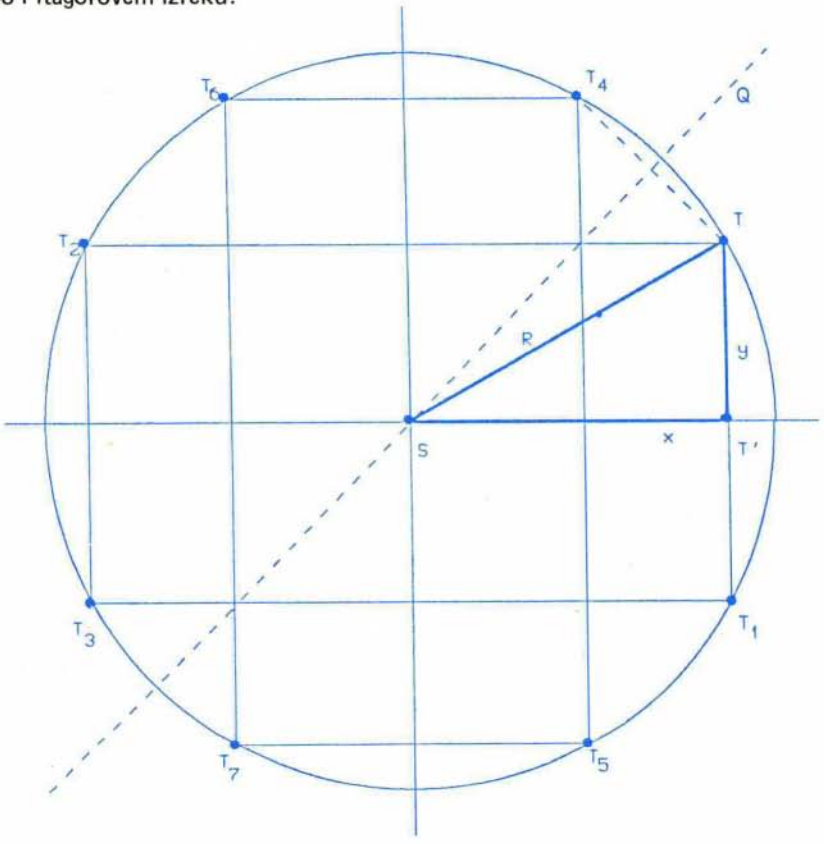
```
procedure Krog1 (xSred, ySred, R: real; N: integer);
  {Program 1: Risanje kroga s kotnimi funkcijami}
    const
        PI = 3.1415926;
    var
        xZac, yZac:  real;    {Zacetna tocka}
        xKon, yKon:real;      {Koncna tocka}
        kot, korak:  real;    {Srediscni kot trenutne tocke in korak}
        i:           integer;{Stevec}
    procedure Daljica (xZac, yZac, xKon, yKon: real);
    begin
      {Narise daljico od tocke xZac, yZac do tocke xKon, yKon}
    end   {Daljica};
begin  {Krog1}
    xKon := R; yKon := 0; kot := 0; korak := 2*PI/N;
    for i := 1 to N do
        begin
            xZac := xKon; yZac := yKon;
            kot := kot + korak;
            xKon := R * cos(kot); yKon := R * sin(kot);
            Daljica (xsred+xZac, ysred+yZac, xsred+xKon, ysred+yKon)
        end
end   {Krog1}.
```

Nazaj k celim številom

Prvo vrsto računalnikovega šestila smo tako spoznali. Pa nam ni nič preveč všeč: kotne funkcije nastopajo v njem, te pa takoj zahtevajo računanje z decimalnimi števili. Tako računanje pa je potrebno na enostavnih računalnikih, kot je na primer Spectrum, posebej sprogramirati, tako da je običajno

znatno počasnejše od računanja s celimi števili. Poskusimo zato poiskati kakšen postopek, kjer nam bo računanje s celimi števili povsem zadostovalo. Kot se spomnimo, nam je pri risanju daljice to uspelo.

Ker smo mojstri, nam bo uspelo tudi pri krogu. Seveda pa bo treba pred končnim programom streti še nekaj orehov. Zato si oglejmo krožnico podrobneje. Najprej si jo zato narišimo (slika 2). Njeno središče za začetek postavimo v izhodišče. Izberimo na krožnici poljubno točko T in njeni koordinati označimo z x in y. Označimo z S središče kroga, s T' pa projekcijo točke T na os x. Kot vemo, je krožnica s polmerom R ravno množica tistih točk, ki so od središča oddaljene za R. Tako je dolžina daljice ST ravno R, dolžina daljice ST' je x, dolžina daljice $T'T$ pa je y. Ker je trikotnik $ST'T$ pravokoten, velja po Pitagorovem izreku:



Slika 2. Skica naše naloge — zrcaljenja

$$x^2 + y^2 = R^2 \qquad (1)$$

Ker smo točko T izbrali povsem poljubno, velja ta zveza za vse točke krožnice. Od tod pa hitro vidimo, da so na krožnici tudi zrcalne slike točke T glede na os x (T_1 na sliki 2), glede na os y (T_2) ter glede na središče (T_3). (O tem se lahko sami kaj hitro prepričate.) Pa še eno zrcaljenje upoštevajmo: zamenjajmo obe koordinati (T_4 v sliki 2). Ta zamenjava se na sliki odraža kot zrcaljenje na premico Q, ki razpolavlja kot med osema x in y. Povzemimo ta razmislek: če je na krožnici točka (x, y), so na njej tudi točke $(-x, y)$, $(x, -y)$, $(-x, -y)$, (y, x), $(-y, x)$, $(y, -x)$ ter $(-y, -x)$. Krožnico bomo tako lahko risali na osmih krajih hkrati: kakor hitro bomo našli koordinate ene točke na krožnici, bomo vedeli še za zgornjih sedem zrcalnih točk. (Nekaj izjem je, ko je točk nekaj manj. Jih znate poiskati sami?) Zato bo dovolj, da narišemo le osmino krožnice (denimo od osi x pa do premice Q), preostale dele pa narišemo s pomočjo zgornjih zrcaljenj.

Še nekaj si na hitro oglejmo. Če točka $T(x, y)$ ni povsem na krožnici, izraz

$$N(x, y) = x^2 + y^2 - R^2 \qquad (2)$$

meri razdaljo točke T od krožnice in mu bomo rekli **odmik** točke od krožnice. Bliže ko je točka krožnici, bližji je odmik 0; $N(x, y)$ je enak nič le pri točkah krožnice. Znotraj kroga je odmik negativen, zunaj njega pa pozitiven. Zato nam bo odmik dragoceno sodilo, ki nam bo pomagalo izbirati točke.

Sedaj pa se lotimo iskanja točk na krožnici. Od zadnjič se še spomnimo, da pri izbiri točk na računalnikovem papirju nimamo povsem prostih rok: počrnimo lahko le točke s celoštevilskimi koordinatami. Privzemimo zato, da je polmer kroga celo število. Potem za eno točko na krožnici takoj vemo: točka $(R,0)$ je to,v kateri smo začeli risati krog pri prejšnjem postopku. Njen odmik je seveda enak 0, kot se lahko sami hitro prepričate. Postavimo se torej v to točko in se iz nje ozrimo naokrog.

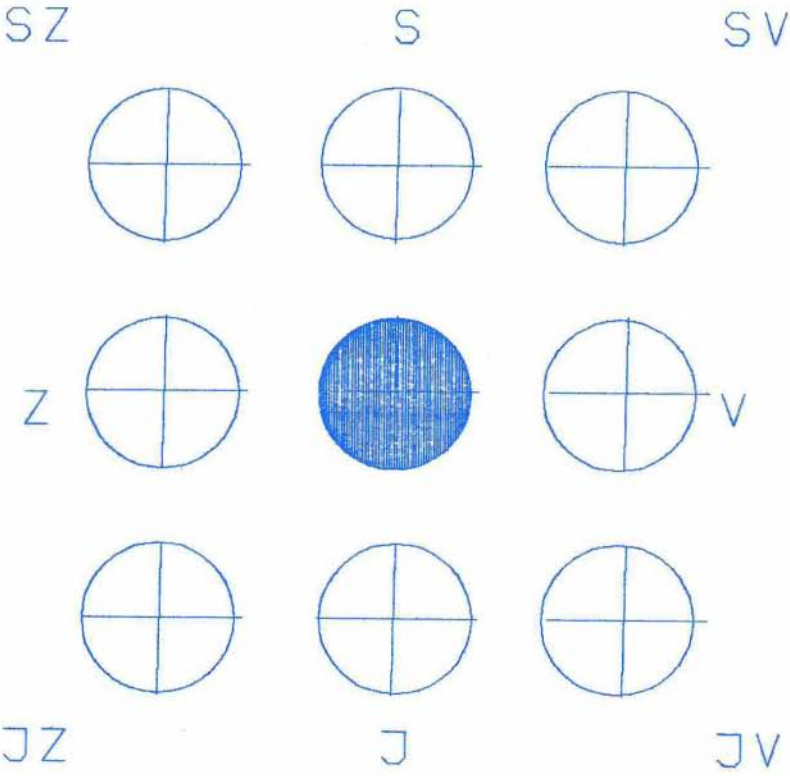
Ko izberemo eno od točk, lahko pot iz nje nadaljujemo le v eno od njenih osmih sosed (slika 3). (Po analogiji z zemljepisnimi kartami smo sliko označili s stranmi neba tako, da je sever S zgoraj, ostale strani pa si slede na običajen način.) Podobno kot pri daljici lahko krog teh osmih sosed takoj zožimo na dve, na tisti dve pač, za kateri vemo, da med njima vodi krožnica. Denimo, da iz izbrane začetne točke začnemo krožnico risati v protiurnem smislu. Potem vemo, da krožnica vodi navzgor in v levo. Ko tako korakamo od točke do točke, vemo, da vse do presečišča krožnice s premico Q krožnica vodi med severno in severozahodno sosedo. Pa nista obe sosedi enako dobri: ena je dlje

od krožnice kot druga. Boljši od obeh možnih korakov je seveda tisti, ki nas privede v točko, katere odmik je kar najmanjši.

Severni korak nas iz točke $T(x, y)$ privede v točko $T_S(x, y+1)$, severozahodni pa v točko $T_{SZ}(x-1, y+1)$. Poglejmo, za koliko se v posameznem primeru spremeni odmik.

$$N(x, y+1) - N(x,y) = x^2+(y+1)^2-R^2-(x^2+y^2-R^2) = 2*y+1$$
$$N(x-1, y+1) - N(x, y) = (x-1)^2+(y+1)^2-R^2-(x^2+y^2-R^2) = 2*(y-x+1)$$

Tako, pa smo pri postopku! Začetno točko poznamo, njen odmik ravno tako, vemo pa tudi, kako izbrati naslednjo točko. Zapišimo torej postopek:



Slika 3. Sosede točke

(1) postavi se v začetno točko:

$x := R, y := 0, odmik := 0$

(2) počrni točke:

$(x,y), (x,-y), (-x,y), (-x,-y),$
$(y,x), (y,-x), (-y,x), (-y,-x)$

(3) izračunaj odmik severne in severozahodne sosede:

$odmikS := odmik + 2 * y + 1, odmik\ SZ := odmik + 2 * (y-x+1)$

(4) če je ABS(*odmikS*) manjše od ABS(*odmikSZ*),
potem $y := y + 1, odmik := odmikS$
sicer $x := x - 1, y := y + 1, odmik := odmikSZ$

(5) korake (2), (3), (4) in (5) ponavljaj, dokler ni x manjši od y.

Program 2

```
procedure Krog2 (xSred, ySred, R : integer);
  {Program 2 : Risanje kroga v celih stevilih }
    var
        x,y:      integer;    { Trenutna tocka }
        odmik,                { Odmik trenutne tocke }
        odmikS,odmikSZ: integer; {ter njene severne in severozahodne sosede }
    procedure Pika (x, y: integer);
    begin
      {Pocrni tocko x, y }
    end    { Pika};
begin  {Krog 2 }
    x := R; y := 0; odmik := 0;
    while x > y do
        begin
            Pika (xSred+x, ySred+y);      Pika (xSred-x, ySred+y);
            Pika (xSred+x, ySred-y);      Pika (xSred-x, ySred-y);
            Pika (xSred+y, ySred+x);      Pika (xSred-y, ySred+x);
            Pika (xSred+y, ySred-x);      Pika (xSred-y, ySred-x);
            odmikS := odmik + 2 * y + 1;  odmik SZ := odmik + 2*(y-x+1);
            if ABS (odmikS) < ABS (odmik SZ) then
                begin y := y + 1; odmik := odmikS end
            else
                begin x := x - 1; y := y + 1; odmik := odmikSZ end
        end
end    {Krog2}.
```

Preden zapišemo postopku ustrezen program, opravimo še drobno izboljšavo. Pri računu odmikov v koraku (3) vidimo, da se odmika spreminjata vzporedno s spreminjanjem x in y, zato ju lahko le enostavno popravljamo in ne vedno znova računamo. Tako izboljšanemu postopku ustreza program 2. Program seveda še ni popoln: včasih počrni kakšno točko preveč (če ste odgovorili na eno od gornjih vprašanj, tudi veste, kdaj). Podprogram Pika v njem počrni točko s podanima koordinatama.

Krožni lok

Končno se lotimo še krožnega loka. To je del krožnice. Zaradi enostavnosti bomo vzeli, da ga podamo z njegovim središčem, polmerom ter začetno in končno točko. Tako začetna kot končna točka lahko ležita kjerkoli na krožnici, dogovorimo se le, da bomo lok risali v protiurnem smislu. Risali ga bomo po korakih od začetne točke proti končni. Podobne bližnjice, kot smo jo ubrali pri krogu, ki smo ga risali na osmih koncih hkrati, si tu torej ne moremo privoščiti.

Način risanja pa bo podoben risanju kroga. Postavili se bomo v začetno točko, izračunali njen odmik od krožnice ter korakali proti končni. Na vsakem koraku se bomo premaknili v eno od točkinih sosed in to počeli, vse dokler ne bomo prikorakali do končne točke. Kot vemo, ima vsaka točka osem sosed. Pri daljici in krogu smo to število lahko hitro zožili na dve, ki se potem med risanjem nista več spreminjali, na tisti dve, med katerima je vodila daljica oziroma krožnica. Tu se bomo morali malo bolj potruditi: primerjati bomo morali odmik treh sosed in med njimi izbrati najugodnejšo. Te tri sosede bomo izbrali na vsakem koraku posebej.

Najprej si oglejmo, kako bomo izbrali prave tri sosede. Spet nam bo pomagalo naše znanje o krogu. Postavimo njegovo središče v izhodišče in si oglejmo ponovno sliko 1. Vemo, da je v vsaki točki T na obodu kroga krožnica pravokotna na polmer ST, ki jo veže s središčem. V točki $(R, 0)$ je krožnica navpična, v točki $(0,R)$ pa vodoravna. Poskusimo sedaj določiti daljico, pravokotno na daljico ST. Daljica ST veže točko (0,0) s točko (x, y). Njeno smer določata x in y: večji je y pri izbranem x, bolj strma je daljica, večji je x pri izbranem y, bolj položna je. Pravimo, da sta x in y parametra smeri. Parametra pravokotne smeri pa sta $-y$ in x, pravokotna daljica iste dolžine torej veže točko (0,0) s točko $(-y, x)$. (To je le ena od pravokotnih daljic te dolžine. Znate poiskati še drugo?) Iz točke T bomo torej pogledali proti točki $(x-y, y+x)$ ter pri risanju kroga upoštevali tri sosede, med katerimi bomo pogledali, in sicer eno navpično, eno vodoravno ter poševno med njima. Kako jih določimo? Vo-

doravno sosedo določa predznak $-y$, navpično predznak x, natančneje pa ju določimo tako kot pri daljici. Poševna povzroča še najmanj preglavic, zakaj, lahko že sami razberete iz slike 3. Tudi postopka podrobneje ne bomo opisovali, gre pa takole:

Program 3

```
procedure KrozniLok (xSred, ySred, R: integer;
                     xZac, yZac, xKon, yKon: integer);
  { Program 3: Risanje kroznega loka s celimi stevili }
    var
        x, y:     integer;    { trenutna tocka }
        dx, dy:   integer;    { vodoravni in navpicni korak }
        odmik:    integer;    { odmik trenutne tocke }
        odmvod:integer;       { odmik vodoravne sosede }
        odmnav: integer;      { odmik navpicne sosede }
        odmpos: integer;      { odmik posevne sosede }
    procedure Pika (x, y: integer);
    begin
      { Pocrni tocko x, y }
    end   { Pika };
begin  { KrozniLok }
    x := xZac–xSred; y := yZac–ySred; odmik := x*x + y*y – R*R;
    repeat Pika (xSred+x, ySred+y);
        if y > Ø then dx := –1 else dx := 1;
        if x > Ø then dy := 1 else dy := –1;
        odmvod := odmik + 2 * x * dx + 1;
        odmnav := odmik + 2 * y * dy + 1;
        odmpos := odmvod + odmnav – odmik;
        if abs (x) < abs (y) then
            if abs (odmvod) < abs (odmpos) then
                begin x := x + dx; odmik := odmvod end
            else
                begin x := x + dx; y := y + dy; odmik := odmpos end
        else
            if abs (odmnav) < abs (odmpos) then
                begin y := y + dy; odmik := odmnav end
            else
                begin x := x + dx; y := y + dy; odmik := odmpos end
    until (x+xSred = xKon) and (y+ySred = yKon)
end   { KrozniLok }.
```

(1) postavi se v začetno točko in izračunaj njen odmik

(2) počrni trenutno točko

(3) s pomočjo predznakov $-y$ in x določi vse tri sosede

(4) izračunaj odmike vseh treh sosed

(5) izmed sosed izberi tisto, katere odmik je najmanjši ter se preseli vanjo

(6) korake (2), (3), (4) in (5) ponavljaj, dokler ne prispeš v končno točko

Ustrezen program prikazuje program 3. Program se seveda zanaša, da smo končno točko točno podali. V nasprotnem primeru se program nikoli ne ustavi.

Ob koncu velja tako pristaviti še nekaj besed o risanju splošnejših krogov in lokov. Pri njih polmer ni ravno celo število, pa tudi središče nima celoštevilskih koordinat. Običajno pri tem postopamo tako, da središče postavimo v najbližjo točko s celoštevilskimi koordinatami ter polmer zaokrožimo do najbližjega celega števila. Pri risanju loka se pri tem znajdemo pred dodatnim problemom: prestaviti moramo tudi začetno in končno točko loka. Enostavno, boste rekli: tako kot pri središču zaokrožimo njene koordinate. Pa ni čisto tako! Poglejmo, zakaj. V programu 3 smo se pri risanju zanašali na to, da sta obe točki točno na krožnici. Pri prestavljanju pa se lahko zgodi, da točko malo narobe prestavimo, da torej narisana krožnica prestavljeno točko malo zgreši. Pri začetni točki to še ni hud problem, huje je to pri končni, saj se program ustavi le, če krožnica točko natančno zadene.

Pa premislimo, kako bi tudi ta problem rešili. Zato še enkrat poglejmo, kako izbiramo točke pri risanju krožnice. Ko eno točko poznamo, za naslednjo izberemo tisto med sosedami, pri kateri je odmik $N(x, y)$ najmanjši. Kot se spomnimo, nam $N(x, y)$ meri oddaljenost narisane točke od prave krožnice. Pri risanju narišemo tiste točke, ki so krožnici najbližje. Pri prestavljanju končne točke moramo zato z uporabo odmika pogledati, ali ni morda katera od sosed nove točke bliže krožnici kot ta točka. Za končno točko moramo tako vzeti tisto, pri kateri je odmik $N(x, y)$ najmanjši. Poskusite program tako sami dopolniti!

Za konec pa še nekaj nalog. Vse so povezane z risanjem loka. Lok namreč v praksi podajamo na več različnih načinov. Prvega smo že povedali: podamo središče, polmer ter začetno in končno točko. Tu je podatkov preveč: pri obeh točkah moramo paziti, da sta od središča oddaljeni natanko za polmer. Pa sprostimo ta pogoj: podajmo lok s središčem, polmerom ter dvema točkama. Prva določa začetek loka tako, da je začetna točka loka presečišče krožnice in

poltraka od središča skozi podano točko. Druga točka na podoben način določa končno točko loka. Poskusite iz teh podatkov določiti podatke za prvotni način podajanja loka. Namesto točk si lahko pomagamo tudi s koti: smer poltraka lahko določimo kar s kotom, ki ga oklepa s poltrakom od središča v desno. Bi znali tovrstne podatke predelati v prvotne? Brez kotnih funkcij ne bo šlo!

Andrej Vitek