PRESEK
List za mlade matematike, fizike, astronome in računalnikarje

ISSN 0351-6652
Letnik **17** (1989/1990)
Številka 6
Strani 338–343

Matija Lokar:

GRAFIKA IN DELO Z ZASLONOM V TURBO PASCALU

Ključne besede: računalništvo.

Elektronska verzija: http://www.presek.si/17/966-Lokar.pdf

GRAFIKA IN DELO Z ZASLONOM V TURBO PASCALU

Proizvajalka Turbo pascala, firma Borland, je s Turbo pascalom, verziji 4 in 5, vpeljala poleg obstoječih razširitev standardnega pascala, še vrsto novih posebnosti. Najpomembnejša je vpeljava modula, ali kot ga imenuje Turbo pascal, UNIT, nekakšne knjižnice podprogramov. Module lahko pišemo tudi sami. Vendar se bomo tu omejili na to, kako uporabljamo obstoječe.

Skupaj s Turbo pascalom dobimo osem standardnih modulov. To so *System*, *Overlay*, *Graph*, *Dos*, *Crt*, *Turbo3*, *Graph3* in *Printer*. Če želimo kak modul uporabljati, moramo to v programu tudi povedati. To storimo tako, da še pred vsemi ostalimi deklaracijami uporabimo novo rezervirano besedo **uses** in nato naštejemo uporabljene module. Kakor hitro je v našem programu ta deklaracija, že lahko uporabimo vse podprograme in funkcije, ki so napisani v posameznem modulu, kot bi bili del standardnega pascala.

```
program  izpis;
uses Crt;
begin
  GotoXY(10,10);
  writeln(' Izpis na mestu 10,10 ')
end.
```

Podprograma `GotoXY` standardni Turbo pascal ne pozna. Če ne bi navedli `uses Crt`, bi prevajalnik javil, da procedure ne pozna. Z navedenim stavkom pa smo si omogočili uporabo vseh podprogramov, ki so v modulu *Crt*. Seveda pa mora imeti Turbo pascal dostop do datotek, kjer ti moduli so. Če bo prevajalnik protestiral, češ., da določenega modula ne najde, povprašajte kakega izkušenejšega kolega, kako pokažemo Turbo pascalu, kje moduli so.

V nadaljevanju si bomo ogledali uporabo dveh modulov in sicer modula *Crt*, ki skrbi za delo z zaslonom in tipkovnico in modula *Graph*, ki nam nudi grafične podprograme. Našteli bomo le manjši del podprogramov, ki so na voljo v obeh modulih. Spisek vseh si lahko ogledate s pomočjo v Turbo pascal vgrajene pomoči. Poglejmo, kako. Kjerkoli v Turbo pascalu pritisnimo na F1. Ne glede na izpisano informacijo pritisnimo še enkrat na F1. Na zaslonu bomo dobili glavni jedilnik pomoči. Z nekaj zaporednimi pritiski na puščice osvetlimo napis `Units` in pritisnimo Enter. V oknu pomoči bomo dobili izpisan seznam vseh osmih modulov. Osvetlimo ime modula o katerem želimo informacije in pritisnemo Enter. S tipkama PgDn in PgUp se premikamo po dobljenem besedilu gor in dol. Tekst je seveda v angleščini,

pa bo že šlo. Več o načinu uporabe pomoči si lahko ogledate v članku *Turbo pascal na hitro* v četrti številki Preseka.

1. Modul Crt

S pomočjo tega modula opravljamo številne akcije v povezavi s tipkovnico in zaslonom. Lahko npr. zbrišemo zaslon, čakamo določen čas, se postavimo z utripačem na poljubno mesto na zaslonu, ... Vsebuje številne podprograme, ki so bili standardni v verziji 3.0. Navedimo nekaj najpogosteje uporabljanih. Pri vsakem podprogramu bomo navedli kako je deklarira, tako da bomo videli, koliko in kakšne parametre potrebuje. Namesto tipov `word` in `byte` si mirno lahko mislite parameter tipa `integer`, le v pravih mejah mora biti. Več o tem si preberite v prejšnji številki Preseka, v članku *Novi tipi v Turbo pascalu*.

ClrScr zbriše zaslon.

```
procedure ClrScr;
```

Delay `procedure Delay(ms : word);`

Prekine izvajanje programa za približno `ms` milisekund.

GotoXY `procedure GotoXY(X,Y : byte);`

Utripač postavi v vrstico `Y` in stolpec `X`. Zgornji levi kot zaslona ima koordinati (1,1), koordinati spodnjega desnega kota pa sta odvisni od zaslona. Največkrat sta (80,25). Če sta kordinati napačni, utripač ostane na svojem mestu, program pa teče dalje.

KeyPressed vrne vrednost true, če je bila na tipkovnici pritisnjena tipka, sicer false.

```
function KeyPressed : Boolean;
```

Pritisnjena tipka ostane v vmesnem pomnilniku tipkovnice. Na to moramo paziti, saj če s tipkovnico ne počnemo ničesar, ostane vrednost funkcije resnično. Npr. pogosto srečamo zanko oblike `repeat until KeyPressed`. Z zanko želimo doseči, da se izvajanje programa nadaljuje šele po pritisku na poljubno tipko. Vendar, če je ta zanka del neke druge zanke, v kateri ne beremo s tipkovnice, se bo izvajanje prekinilo le pri prvem prehodu zunanje zanke. Pri ostalih bo `KeyPressed` ostalo `true` in navedena zanka se bo takoj iztekla. Npr.

```
program a;
uses crt;
begin
  repeat
    writeln('==============');
    repeat until KeyPressed
  until false
end.
```

Program bo na pritisk na tipko čakal le po prvem izpisu, kasneje pa ne več. Zato je varneje take prekinitve izpeljati s funkcijo ReadKey.

ReadKey prebere znak s tipkovnice.

```
function  ReadKey : char;
```

Pritisnjena tipka se ne pozna na zaslonu. Čakanje na uporabnika izpeljemo takole

```
program a;
uses crt;
var c : char;
begin
  repeat
    writeln('==============');
    c := ReadKey
  until false
end.
```

2. Modul Graph

Modul *Graph* vsebuje konstante, tipe in podprograme za delo z grafiko. Omogoča delo z večino najbolj razširjenih grafičnih standardov za PC. Skupaj s Turbo pascalom dobimo gonilnike za naslednje grafične adapterje: CGA, EGA, Hercules, MCGA, VGA, AT&T 400, 3270 PC in IBM-8514. Ti so na datotekah s podaljškom BGI. Če bomo program izvajali na računalniku z grafično kartico Hercules, moramo imeti datoteko HERC.BGI.

Pa poglejmo, kaj moramo storiti, če želimo uporabljati grafiko. Način, ki bo opisan, morda res ni najelegentnejši, vendar je dokaj zanesljiv ne glede na

organizacijo našega diska. Okostje vsakega našega grafičnega programa bo približno takšno

```
program grafika;
uses Graph;
const Pot = 'C:\TP';        { kje so datoteke *.BGI }
var GD,GM : integer;
     ...
begin
  .
  { zacetek dela v graficnem nacinu }
  DetectGraph(GD,GM);
  InitGraph(GD,GM,Pot);
  .
  .            { delo v graficnem nacinu }
  .
  Readln;
  CloseGraph;  { konec dela v graficnem nacinu }
  ...
```

Kot vidimo, smo uporabili konstanto `Pot`, s katero smo opisali, kje na disku so ustrezni gonilniki in dve spremenljivki, s katerima se opiše grafična kartica in način dela. Nato smo s proceduro `DetectGraph` ugotovili, katera grafična kartica je v našem računalniku in z `InitGraph` prešli v grafični način. Le-tega smo zapustili s pomočjo podprograma `CloseGraph`. Še prej pa smo uporabili trik s stavkom `readln`, ki prepreči, da bi rezultati prehitro izginili iz zaslona, saj `CloseGraph` preklopi spet nazaj na tekstovni način in s tem grafična slika izgine z zaslona. Tako pa bo program čakal, dokler ne pritisnemo na Enter.

Koordinatno izhodišče (točka (0,0)) je v zgornjem levem kotu. Koordinata x narašča proti desni. Vrednosti koordinate y naraščajo navzdol. Koordinati spodnjega desnega kota sta odvisni od grafične kartice, ki jo uporabljamo. Pri nas je najbolj razširjena kartica Hercules. Pri tej sta koordinati spodnjega desnega kota (719,347).

Omeniti velja še to, da v grafičnem načinu ne moremo uporabljati standardnih podprogramov `write` in `writeln`, ampak si moramo pomagati s posebnimi podprogrami za izpis, ki so na voljo v modulu *Graph*.

Navedimo sedaj nekatere izmed več kot 50 podprogramov in funkcij.

Circle `procedure Circle(x,y: integer; Radius: word);`

Nariše krog s središčem v točki (x, y) in polmerom `Radius`.

Line `procedure Line(x1,y1,x2,y2: integer);`

Nariše daljico z začetkom v točki $(x1, y1)$ in koncem v točki $(x2, y2)$.

LineRel `procedure LineRel(Dx,Dy: integer);`

Nariše daljico z začetkom v točki, kjer je trenutno grafični kazalec (recimo (x, y)) in koncem v točki $(x + Dx, y + Dy)$. Grafični kazalec je po tem podprogramu v končni točki.

LineTo `procedure LineTo(X,Y: integer);`

Nariše daljico z začetkom v točki, kjer je trenutno grafični kazalec, in koncem v točki (x, y). Grafični kazalec je potem v končni točki.

MoveTo premik grafičnega kazalca v določeno točko

```
procedure MoveTo(X,Y: integer);
```

Premakne grafični kazalec v točko (X, Y). Grafični kazalec na zaslonu ni viden.

OutTextXY na določenem mestu izpiše niz

```
procedure OutTextXY(X,Y: integer;
                    TextString: string);
```

Izpiše niz `TextString` z začetkom na mestu (X, Y). Če je niz predolg, se ne izpiše (pri standardni nastavitvi). Če želimo izpisati števila, jih moramo prej spremeniti v niz (npr. s podprogramom `Str` - glej članek v prejšnjem Preseku).

PutPixel na določenem mestu nariše točko

```
procedure PutPixel(X,Y: integer; Color: word);
```

Na mestu (X, Y) nariše točko v barvi `Color` (črna = 0, bela = 15).

Naštete niso številne procedure za razne nastavitve, izpopolnjevanje likov, itd. Zelo pogosto vprašanje je tudi to, kako sliko z zaslona spraviti na papir. Žal ustreznega podprograma v modulu *Graph* ni. Če bo za to več zanimanja, bomo v Preseku objavili ustrezen program, ki ga najdete tudi v knjigi *Matija Lokar, Turbo pascal.* Tam si lahko preberete več o uporabi modulov in Turbo pascalu nasploh.

Matija Lokar