

Namangan Muxandislik Pedagogika Instituti

“Informatika” fakulteti

“ **ALGORITMIK TILLAR** ”

fanidan yozgan

# KURS ISHI

Topshirdi:

12-KTIAT 07

guruh talabasi

A. Mamadaliyev.

Qabul qildi:

N. Qurbonov.

Namangan 2010

## **3-Tajriba ishi.**

Mavzu: Dasturlash tillarining  
grafik rejimida ishlash.

Ishning maqsadi: Turli xil funksiyalarning grafiklarini, statik  
va dinamik tasvirlarni  
dasturlash tillaridan foydalanib hosil  
qilishni organish.

### ***Ishning rejasi:***

1. Pascal dasturlash tilining grafik rejimi va unda ishlash.
2. 5-jadvalda berilgan funksiyalarning grafiklarini qurish algoritmini ishlab chiqing va uni blok-sxemalar orqali ifodalang.
3. Berilgan xarakatsiz va xarakatli jismlar tasvirini yaratish algoritmlarini ishlab chiqing va uni blok-sxemalarda ifodalang,
4. Algoritmning Paskal tilidagi dasturini tuzing.
5. Dastur bo'yicha natijalar oling va hisobot tayyorlang.

# 1. Pascal dasturlash tilining grafik rejimi va unda ishlash.

## a) Turbo -paskalning grafika rejimida ekran xolati.

Malumki, ekran to'rtburchak maydon bo'lib, juda ko'p nuqta(Pixel)lardan tashkil topgan. Grafik ekran ishchi maydon va bordyur (ekranning chetki qismi) dan iborat boladi.

Grafik rejimda ekrandagi barcha nuqtalarning ranglarini o'zgartirib chiqish mumkin (eslatib otamiz, matnli rejimda bu mumkin emas). Turli rangga bo'yalgan nuqtalar yordamida chiziqlar, matnlar va boshqa xar hil tasvirlar hosil qilinadi. Ekran turiga qarab ranglar soni turli hil bolishi mumkin, eng kami bilan esa 2 hil rang boladi.

Kompyuter ekrani yoki matnli rejim yoki grafik rejim holatida bo'ladi. Bir vaqtning o'zida ekranning bir qismi grafik rejimda, bir qismi matnli rejimda bo'lishi mumkin emas. Chunki ekranda ko'rinayotgan barcha tasvirlar video xotiradagi malumotlarning aksi - tasviridir.

Ekranning grafik yoki matnli rejimlariga qarab, videoxotiradagi ma'lumotlar turlicha boladi.

Videoxotira, yorug'lik trubkasining kontrolleri, kirish-chiqish portlari va h.k. lar bitta platada joylashadi va displey adapteri deb ataladi. Amalda bir necha xil displey adapterlari mayjud bolib, ular quyidagi ko'satkichlari bilan bir-biridan farq qiladi:

- ekranning tasvirni ko'rsatish sifati;
- ekranda bir vaqtda korsatish mumkin bo'lgan ranglar soni.

Quyida amalda keng tarqalgan videoadapterlar sanab o'tilgan:

1. CGA(Color Graphics Adapter);
2. MCGA(Multi Color Graphics Array);
3. EGA(Enhanced Graphic Adapter);
4. VGA(Video Graphics Array).

Kompyuterda qanday adapterning o'rnatilishiga qaramay Turbo – Paskalning prosedura va funksiyalaridan to'liq foydalanish mumkin. Ularni o'zaro sozlashuvi avtomatik tarzda kechadi. Bu sozlashlarni grafik drayverlar deb ataluvchi maxsus programmalar bajaradi. Drayverlar \*.BGI kengaytmasi bor bo'lgan fayllarda joylashgan.

Misol uchun, EGA va VGA adapterlari bilan ishlash uchun zarur drayverlar EGAVGA.BGI faylida,CGA va MCGA adapterlariga mos drayverlar esa CGA.BGI faylida joylashgan.

Turli xil drayverlarni ko'rsatish uchun **Graph** modulida quyidagi o'zgarmlar aniqlangan:

const

*Detectr=0;{Drayverni avtomatik tarzda aniqlash}*

CGA=1;

MCGA=2;

EGA=3;

EGA64=4;

EGAMONO=5;

IBM8514=6;

HERCMONO=7;

ATT400=8;

VGA=9;

PC327=10.

Graph modulida grafik rejimlarni korsatish uchun esa quyidagi o'zgarmaslar aniqlangan (CGA, EGA va VGA adapterlari uchun);

const

CGAC0=0;

CGAC1=1;

CGAC2=2;

CGAe3=3;

CGAN<sub>2</sub>=4;

EGALo=0;

EGAHi=1;

EGA64Lo=0;

EGA64Hi=1;

VGALo=0;

VGAMed=1;

VGAHi=2.

### **b) Displeyni grafik rejimga otkazish.**

Displeyni doimiy rejimi, matnli rejim hisoblanadi. Displeyni grafik rejimga o'tkazish uchun *Graph* modulining *InitGraph* prosedurasidan foydalaniladi:

*InitGraph(GD,GM,Path)*

bu yerda      *GD*-drayver nomeri ;

*GM*-rejim nomeri ;

*Path*-kerakli drayver joylashgan fayl yoli.

*GD* va *GM* nomerlarni qanday aniqlashni oldingi mavzuda ko'rib chiqdik. Agar *Path* ozgaruvchisi bo'sh satr qatorini tashkil qilsa (*Path=''*) drayverni ishchi katalogdan qidiriladi.

*GD* va *GM* o'zgaruvchi parametrlar hisoblanadi. Agar *InitGraph* prosedurasini ishga tushirishda *GD=0* bolsa, zarur drayver va bu drayver uchun eng yaxshi rejim avtomatik tarzda aniqlanadi.

*Graph* modulida qulaylik uchun *Detect* o'zgarmasi (*Detect=0*) kiritilgan. Misollarda undan qanday foydalanganligiga ahamiyat bering.

*InitGraph* prosedurasiga simmetrik prosedura *CloseGraph* hisoblanadi. Bu prosedura drayverni xotiradan chiqarib tashlaydi va oldingi videorejimni tiklaydi.

Quyidagi programma displeyni grafik rejimga otkazadi va shu zahotiyiq uni asli holiga qaytaradi:

```
uses Graph;
```

```
var
```

```
    Gd,Gm: Integer;
```

```
begin
```

```
GD:=detect; {Drayverni avtomatik tarzda aniqlaydi, chunki DetectqO}
```

```
    InitGraph(GD,GM, 'd:\tp'); {Grafik rejimni o'rnatish}
```

```
ReadLn;                            {iEnter bosqichini bosilishini kutish}
```

```
CloseGraph;
```

```
end.
```

Ba'zi hollarda, masalan .BGI kengaytmali fayl yolini xato ko'rsatilsa yoki grafik rejimni o'rnatish uchun tezkor xotira yetishmasa *InitGraph* prosedurasini o'z ishini muvaffaqiyatli yakunlay olmaydi. Bunday hollarda yol qo'yilgan xatoliklarni aniqlash uchun *GraphResult* funksiyasidan foydalanish mumkin. Bu funksiya grafik rejimni ishga tushirishga harakat qilganda quyidagi qiymatlardan birini beradi:

0 - xatolar yo q;

2 - grafik plati o'rnatilmagan;

3 - drayver fayli topilmadi;

4 - noto'g'ri drayver o'rnatilgan;

5 - grafik rejimni o'rnatish uchun tezkor xotira yetishmagan;

*GraphResult* funksiyasidan foydalanishga doir quyidagi programmani ko'rib chiqaylik:

```
uses Graph;
```

```
var
```

```
    GA GM, ErrCode: integer;
```

```
begin
```

```
    GD:-Detect;    {Drayverni avtomatik tarzda aniqlash}
```

```
    InitGraph(GD,GM, ' ');    {Grafik rejimni ishga tushirish}
```

```
    ErrCode:=GraphResult;
```

```
    if ErrCode<>0 then WriteLn(ErrCode,'raqamli xato)
```

```
        {Xatoga yol qo'yilgan}
```

```
    else CloseGraph;
```

```
        {Xato topilmagan va grafik rejimni ishi yakunlandi}
```

```
End.
```

### **c) Nuqta, chiziq va ranglar.**

Biz ekranda grafik rejimni qanday o'rnatishni ko'rib chiqdik. Endi ekranda turli xil nuqtalar, chiziqlar va shakllar chiqarishni tashkil qilaylik. *Graph* modulida 80 taga yaqin funksiya va proseduralar mavjud bo'lib, ulardan foydalanib quyidagi ishlarni qilish mumkin

- nuqtalar qurish;
- kesmalar chizish;
- ellips va aylanalar chizish;
- to'g'ri to'rtburchaklar va ko'pburchaklar chizish;
- yopiq shakllarni turli ranglarga bo'yash, hamda bir necha hil standart va keragicha nostandart usullar bilan sohalarni shtrixlash;
- ekranga turli shriftdagi matnlarni chiqarish;
- ekran sohasini eslab qolish va ularni surish.,

Grafik rejimda ishlash xuddi matematika fanidagi Dekart koordinatalar sistemasida nuqtalar orqali turli hil shakllar yasashga o'xshab ketadi. Ekrandagi har bir nuqta o'zining koordinatalariga ega. Ekraning chapdan eng tepadagi nuqtasining koordinatasi (0,0) ga teng. X koordinatasi chapdan o'ngga o'sib borsa, Y koordinatasi yuqoridan pastga qarab o'sib boradi. Nuqta koordinatasini aniqlovchi (x,y) juftlikdagi birinchi qiymat OX o'qidan, ikkinchi qiymat esa OY o'qidan aniqlanadi. Ekraning o'ngdan eng pastidagi nuqtasi oxirgi nuqta hisoblanadi va uning koordinatasi grafik

rejimning turiga bog'liq. Masalan, VGAHi rejimida ekrandagi nuqtalar soni 640x480ga teng. O'ngdan eng pastki nuqta koordinatasi esa (639,479) ga teng boladi.

Endi *Graph* modulining eng ko'p ishlatiladigan funksiya va proseduralari bilan tanishib qhiqaylik.

1. ***PutPixel*** (x,y, color) prosedurasi -(x,y) koordinatali nuqtani *Color* parametri bilan aniqlangan rangga bo'yab beradi;

Misol: PutPixel(100,120,Red) - ekranda (100,120) koordinatali qizil nuqta paydo boladi.

2. ***GetPixel*** (x,y) funksiyasi - (x,y) koordinatali nuqtaning rangini aniqlab beradi;

Misol: Color!=GetPixel(100,120); (100,120) koordinatadagi nuqtaga qo'yilgan rang qiymatini aniqlaydi.

3. ***Line*** (x1,y1,x2,y2) prosedurasi -(x1,y1) va (x2,y2) koordinatali nuqtalarni tutashtiruvchi kesma chizadi;

4. ***Circle***(x,y, Radius) prosedurasi - markazi (x,y) nuqtada joylashgan radiusi Radius ga teng aylana chizadi;

5. ***Rectangle*** (x1,y1,x2,y2) prosedurasi-chap yuqoridagi burchagi (x1,y1) va ongdan pastki burchagi (x2,y2) koordinatali nuqtalar orqali otkazilagan tog'ri to'rtburchak chizadi;

6. ***SetColor*** (Color) prosedurasi - chizish rangini o'rnatadi;

Grafik rejimda ranglarni belgilash uchun xuddi matnli rejimdagi kabi quyidagi o'zgarmaslar ishlatiladi!

const

Black=0;{Qora}

Blue=1;{Ko'k}

Green=2;{Yashil}

Cyan=3;{Zangori rang}

Red=4;{Qizil}

Magenta=5; {Pushtirang}

Brown=6; {Jigarrang}

LightGray=7;{Och kulrang}

DarkGray=8;{To'q kulrang}

LightBlue=9;{Och havorang}

LightGreen=10;{Och ko'k}

```

LightCyan=11;{Och zangorirang}
LightRed=12;{Och qizil}
LigthMagenta=13;{Och pushtirang}
Yellow=14;{Sariq}
White=15;{Oq}

```

Misol:

*uses Graph;*

*var*

*GD,GM integer;*

*Rang, Radius: word;*

*Begin*

*GD:=Detect; {Drayvemi avtomatik tarzda aniqlash}*

*InitGraph(GD,GM,' '); {Graftk rejimni ishga tushirish} .*

*for Rang:=15 downto 0 do*

*begin*

*setcolor(Rang); {CHizish rangini o'rnatish}*

*Radius:=Rang\*10;*

*Circle(GetMaxX div 2, GetMaxYdiv 2,Radius);*

*{Markazi ekran markazida joylashgan aylana chizish}*

*end;*

*Readln;*

*CloseGraph;*

*end.*

#### **d) Turli xil figuralar.**

Graph modulining prosedura va funksiyalari yuqoridagi mavzuda korib chiqilgan funksiya va proseduralar yordamida faqat chiziqlar chizish ni mumkin. Endi boshqa turli xil ranglar bilan tildirilgan figuralar chizishni tashkil etishga yordam beruvchi yana bir nechta prosedura va funksiyalar bilan tanishib chiqamiz.

1. SetFillStyle(Style,Color) prosedurasi - Color rangi bilan sohalarni tildirish va ularni korsatilgan uslubda tildirish (shtrixovka qilish) uchun ishlatiladi; Sohani turli ranglar bilan tildirish o'zgarmlari:



const

EmptyFill=0; {sohani ekran fonining rangiga bo'yaydi}

SolidFill=1; {sohani belgilangan rangda uzluksiz toldirish}

LineFill=2; {sohani qalin gorizontal ( ..... ) chiziqlar bilan toldiradi}

LtSlashFill=3; {sohani ingichka "/// " belgilari bilan toldiradi}

SlashFill=4; {sohani qalin "/// " belgilari bilan toldiradi}

BkSlashFill=5; {sohani qalin "\\\" belgilari bilan toldiradi}

LtBkSlashFill=6; {sohani "\\\" qatlami bilan toldiradi}

HatChFi 11-7; {sohani to'r bilan toldiradi}

XhatChFill=8; {sohani egri to'r bilan toldiradi}

InterLeaveFill=9; {sohani zich egri shtrixovka bilan toldiradi}

WideDotFil 1=10; {sohani kam uchrovchi nuqtalar bilan toldiradi}

CloseDotFil 1=11; {sohani zich nuqtalar bilan toldiradi}

UserFill= 12; {sohani dasturchi aniqlagan shtrixovka bilan toldiradi}

2.Bar(x1,y1,x2,y2) prosedurasi ekrandagi rang va shtrixovka ustiga to'g'ri tort burchak quradi;

3.Bar3D(x1,y1,x2,y4,Depth,Top) prosedurasi ham shunday rang va shtrixovka bilan toldirilgan parallelepiped chizadi. Depth o'zgaruvchisi parallelepiped balandligini anglatadi. Top mantiqiy o'zgaruvchisi true qiymatli bolsa parallelepipedning yuqori asosi chiziladi aks xolda, chizilmay ochiq qoladi;

4. FillEllipse(x,y,XRadius,YRadius) prosedurasi oldin o'rnatilgan rangga toldirilgan ellips chizadi. Ellips o'qlari koordinata o'qlariga paralel deb olinadi. XRadius - ellips eni, YRadius - ellips balandligi.

### **e) Grafik rejimdagi matnlar.**

Grafik rejimda matnlarni yozish uchun ikki xil tipdagi shriftdan foydalanish mumkin: nuqtalar matrisasi va simvolni tashkil etuvchi vektorlar qatori orqali.

Shriftlar fayllari .CHR kengaytmasiga ega boladi va shriftni ishga sozlaganda kerakli fayllar ishchi katalogda yoki .BGI grafik drayveri joylashgan katalogda bolishi kerak.

Shriftni tanlash va masshtab o'rnatish SetTextStyle prosedurasi yordamida amalga oshiriladi:

SetTextStyle (Font, Direction, Size) - kerakli shriftni o'rnatadi, matnni chiqarish yonalishini aniqlaydi va belgilar olchovini belgilab beradi. Font- shriftni aniqlovchi o'zgaruvchi, Direction-matnni chop etish yo'nalishini ko'rsatuvchi o'zgaruvchi (chapdan o'ngga yoki pastdan yuqoriga), Size-shrift olchovini aniqlovchi o'zgaruvchi. Matrisali shriftda olchov Size=1, vektor shriftida esa Size=4 qiymatlarida yerishiladi.

Turli xil shriftlarni ko'rsatish va matnlarni chop etish yo'nalishlarini tanlash uchun quyidagi o'zgarmaslar aniqlangan:

```
const
{ shriftlar }
DefaultFont=0; { 8x8 nuqtali standart matrisali shrift }
TriplexFont=1; { vektorli shrift }
SmallFont=2; { vektorli shrift }
SansSerifFont=3; { vektorli shrift }
GothicFont=4; { vektorli shrift }
{ matn yo'nalishi }
HorizDir=0; { chapdan o'ngga }
VertDir=1 { pastdan yuqoriga }
```

*OutTextXY(x,y,TextString)* proseduralari oldindan aniqlangan shriftda, yo'nalishda va belgi olchovida *TextString* qatorini (x,y) nuqtadan boshlab chop etadi.

*SetTextJustify(Horiz,Vert)* proseduralari *OutTextXY* proseduralari chiqaradigan matnni avtomatik tarzda tekislab beradi. *Horiz* - gorizontal, *Vert* - vertikal tekislashlar.

Matnlarni tekislash uchun quyidagi o'zgarmaslar aniqlangan:

```
const
{ gorizontal tekislash uchun }
LeftText=0; { chap tomonga nisbatan tekislash }
CenterText=1; { markazga nisbatan tekislash }
RightText=2; { o'ng tomonga nisbatan tekislash }
{ vertikal tekislash uchun }
BottomText=0; { pastgi tomonga nisbatan tekislash }
CenterText=1; { markazga nisbatan tekislash }
TopText=2; { yuqori tomonga nisbatan tekislash }
```

## **f) Ekran sohalari**

*GetImage*, *Put Image* proseduralari va *ImageSize* funksiyasi yordamida tasvirlarning to'g'ri tortburchakli sohalarni hotirada eslab qolish va ularni ekranga chiqarishimiz mumkin.

1. *ImageSize(x1,y1,x2,y2)* funksiyasi — ekranning to'g'ri to'rtburchakli sohasini saqlash uchun zarur bo'lgan xotira olchovini (baytlarda) beradi. (x1,y1) to'g'ri to'rtburchakli ko'rinishning chapdan

yuqoridagi,  $(x_2, y_2)$  - esa, pastdan o'ngdagi burchak nuqtalari uchun koordinatalar.

2. *Getimage*( $x_1, y_1, x_2, y_2, Area$ ) prosedurasi xotiraning *Area* sohasida to'g'ri to'rtburchakli ekran tasvirini saqlaydi.  $(x_1, y_1)$  va  $(x_2, y_2)$  lar yuqoridagi ma'noda qayta ishlatilmoqda.

3. *PutLnage*( $x, y, Areti, Mode$ ) prosedurasi ekranning ko'rsatilgan joyiga tasvir ko'rinishini chop etadi.  $(x, y)$  — xotiraning *Area* sohasidagi tasvir ko'rinishi nusxasini chop etiladigan, ekranning chapdan yuqoridagi nuqtasining koordinatasi. *Mode*-tasvirni ekranga chiqarish rejimi.

Tasvirlarni ekranga chiqarish rejimini aniqlash uchun foydalaniladigan o'zgarmaslar:

const

{PutImage prosedurasi uchun o'zgarmaslar}

NormalPut=0; { mavjud tasvirni almashtirish }

XorPut=1; { XOR mantiqiy amali }

OrPut=2; { OR mantiqiy amali }

AndPut=3; { AND mantiqiy ko'paytirish amali }

NotPut=4 { NOT mantiqiy rad etmoq amali }

### **g) Xatolar tahlili**

Grafik rejimni o'rnatish mavzusida yol qo'yilgan xatolar diagnostikasi uchun *GraphResult* funksiyasidan foydalangan edik. Hozir shu funksiya beradigan xatolar kodi bilan toliqroq tanishib chiqaylik.

Quyida *GraphResult* funksiyasi beradigan kodlarga mos o'zgarmaslar ro'yhati keltirilgan:

const

grOk=0; { xatolar yo'q }

grNoInitGraph=-1; { Grafik rejim o'rnatilmagan (InitGraph prosedurasini ishga tushiring ) }

grNotDetected=-2; { Grafik plata o'rnatilmagan }

-6 - sohalarni ko'chirishda xotira chegarasidan chiqish;

-7 - zarur sohani bo'yash paytida xotira chegarasidan chiqish;

-8 - shrift fayli topilmagan;

-9 - shrift faylini ishga tushirish uchun xotira yetishmayapti;

-10 - tanlangan drayver uchun noto'g'ri grafik rejim.

### **h) Graph modulining prosedura va funksiyalar**

1. Arc prosedurasi - aylana yoyini chizadi.

Aniqlanishi : Arc(x,y : integer; StAng, EndAng, Radius: Word);

X,Y - aylana markazining koordinatasi;

StAng, EndAng - mos ravishda yoyning boshlang'ich va oxirgi burchaklari;

Radius-aylana radiusi.

2.Bar prosedurası - rangga bo'yalgan to'g'ri to'rtburchak chizadi.

Aniqlanish: Bar(x1 ,y1 ,x2,y2:integer);

(x1,y1) va (x2,y2) mos ravishda to'g'i tortburchakning chetki nuqtalari koordinatalari:

3.Bar3D prosedurası rangga boyalgan parallelipiped chizadi.

Aniqlanishi :Bar3D(x1,y1,x2,y2:integer;Depth:word;Top:boolean);

(x1,y1) va (x2,y2) asosni tashkil etuvchi to'g'ri to'rtburchak uchlarining koordinatalari;

Depth -parallelipiped chuqurligi;

Top- mantiqiy o'zgaruvchi.

4. Circle prosedurası - aylana chizadi;

Aniqlanishi: Circle (x,y:integer; Radius: word)x

(x,y) aylana markazining koordinatasi; Radius-aylana radiusi.

5. CloseGraph prosedurası grafik rejimini uzadi.

Aniqlanishi :Closegraph;(parametrsiz prosedura)

6.DrawPoly prosedurası - ko'p burchak chizadi.

Aniqlanishi :DrawPoly(NumPoints:word; var PolyPoints);

NumPoints - ko'pburchak lomonlari soni;

PolyPoints - ko'pburchak uchlarining koordinatalaridan tuzilgan massiv.

7.Ellipse prosedurası - ellips yoyini chizadi.

Aniqlanishi: Ellipse(x,y:integer;StAng,EndAng:word;XRadius,YRadius:word);

(x,y) — ellips markaznirig koordinatasi;

St Ang va EndAng - yoyning boshlang'ich va oxirgi burchaklari;

Xradiues va Yradius nios ravishda ellips balandligi va eni.

8.FillPoly prosedurası - rangli ko'pburchak chizadi.

Aniqlanishi: FillPoly (NumPointsrword; var PolyPoints);

NumPoints - ko'pburchakning uchlari soni;

PolyPoints - ko'pburchak uchlari koordinatalaridan tuzilgan massiv.

9. GetArcCoords prosedurası - oxirgi niarta ishlatilgan Arc prosedurasining koordinatalarini aniqlaydi.

Aniqlanishi: GetArcCoords (var ArcCoords:ArcCoords Type);

10. *GctColor* funksiyasi - ekran rangini aniqlaydi.

Aniqlanishi: `GetColor:word;`

11. *GetGraphMode* funksiyasi - grafik ekranni qaytaradi. Aniqlanishi: `GetGraphMode:integer:`

12. *Get Image* prosedurasi - ekranning berilgan sohasini Area da saqlaydi.

Aniqlanishi: `GetImage(x1,y1,x2,y2:integer;var Area):`

13. *GetMaxColor* funksiyasi - rangning eng katta qiymatini hisoblaydi.

Aniqlanishi: `GetMaxColor:word;`

14. *GetPixel* funksiyasi - berilgan nuqta rangini aniqlaydi.

Aniqlanishi: `GetPixel(x,y:integer):word;`

15. *GraphErrorMsg* funksiyasi - berilgan kod bo'yicha xato haqida satr malumot beradi.

Aniqlanishi: `GraphErrorMsg(Code:integer):string;`

16. *LineTo* prosedurasi - oldingi aniqlangan nuqtadan berilgan nuqtagacha kesma chizadi.

Aniqlanishi: `LineTo(x,y:integer);`

17. *PieSUce* prosedurasi sektor chizadi.

Aniqlanishi: `PieSlice(x,y:integer;StAng,EndAng,Raduis:word);`