PROGRAMMERING A

C++

DEL 2 av 2.

GOTOXY, TEXTCOLOR och GRAPHICS.h

Kompendium med lektionsanteckningar och övningsuppgifter

Stefan Sundin

Torsbergsgymnasiet Bollnäs

GOTOXY och TEXTCOLOR

Att styra färg och position i utskrift till skärmen

Hittills har utskriften skett vänsterjusterad och med vit färg. Det går emellertid att själv välja både läge och färg på texten.
Utskriftspositionen flyttas med anropet:

```
gotoxy(x,y);
```

Där x och y är skärmkoordinaterna.

> Det horisontella x har värden mellan 1 och …. och det vertikala y mellan 1 och …..
> Var hamnar anropet gotoxy(1,1)? Svar: I övre vänstra hörnet!
> Fråga a: Var hamnar anropet gotoxy(80,24)? Svar:……
> Fråga b: Var hamnar anropet gotoxy(40,12)? Svar:……

Färgen ändras med anropet:

```
textcolor(n);
```

Där n är ett heltal mellan 1 och 16.
Varje siffra motsvarar en färg, t ex 1 är blått och 4 rött.

Vill vi t ex skriva ut *Hej* med röd färg i positionen (45,12) görs detta med följande anrop:

```
gotoxy(45,12);
textcolor(4);
cout<<"Hej";
```

OBS: För att använda gotoxy (int, int); måste du inkludera filen conio.h Samt ev. lägga till –lconio i rutan länkare unde projektalternativ

Uppgift 11:6

a. Gör ett program där du skriver ut ditt namn på 5 olika platser på skärmen.
b. Modifiera programmet så att namnet skrivs ut med olika färger.
c. Vilken siffra hör ihop med respektive färg? Ta hjälp av en for-loop som skriver ut siffran i rätt färg: 1: 2: 3: 4: 5: 6: 7: 8: 9:
10: 11: 12: 13: 14: 15: 16:

Uppgift 11:7

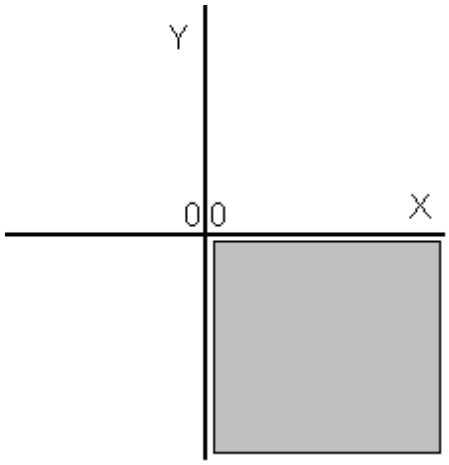
a. Skapa ett program som skriver ut asterisker (*) horisontellt över skärmen. For-loop!
b. Skapa ett program som skriver ut streck (_) vertikalt över skärmen. For-loop!
c. En utmaning: skriv ut o:n (o) *diagonalt* över skärmen. For-loop!

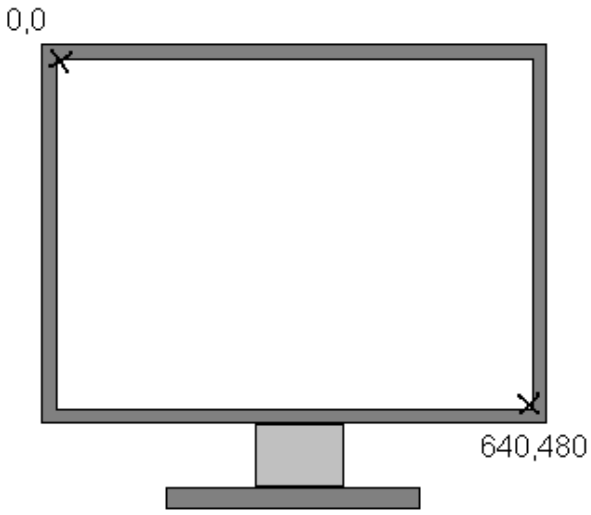
Grafik med winbgim och Dev C++

En ”console applikation” skriven i C++ kan använda bildskärmen i antingen *textmode* eller *grafikmode (text- eller grafikläge*. I textläge, så som vi använt skärmen hittills, kan man bara, som namnet säger, skriva text. Till sitt förfogande har man då de 256 asciitecknen. Några av dessa kallas semigrafiska och har utseendet av hörn och streck men det är fortfarande asciitecken och ingen egentlig grafik
I grafisk mode delas skärmen in i ett antal *pixlar*. Vid användande av VGA indelas skärmen i 480 rader med 640 pixlar i varje rad.

640x480 (**VGA**) - 800x600 (**SVGA**) - 1024x768 (**XVGA**)

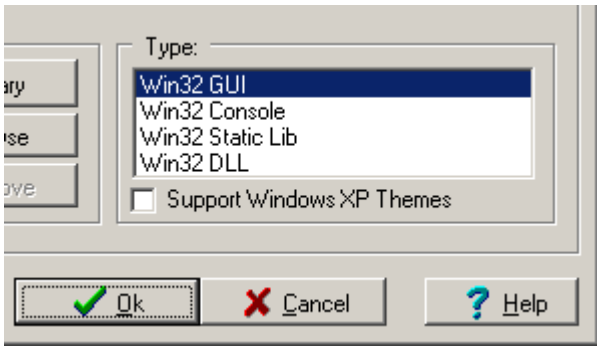
En pixel pekas ut på samma sätt som en punkt i ett koordinatsystem. Med den skillnaden att origo ligger i skärmens övre högra hörn och att y-axeln är riktat nedåt.



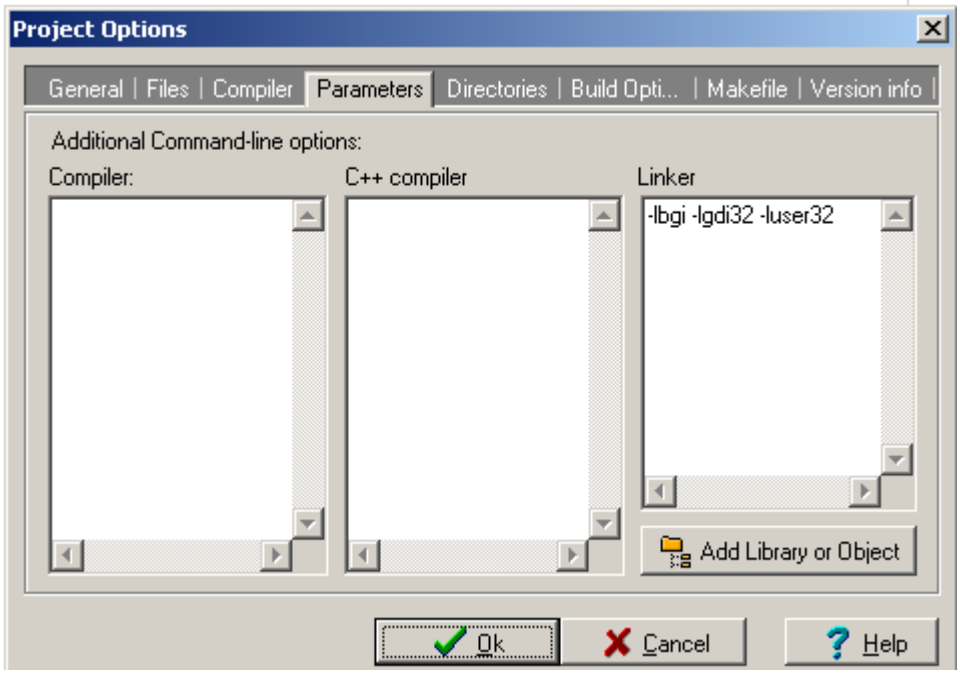


Ett första exempel

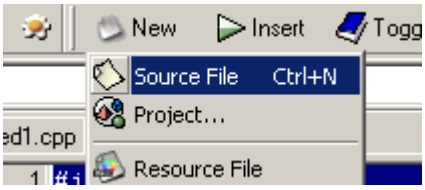
Skapa ett nytt tomt projekt : File – New - Projekt
OBS! inga mellanslag, eller ”konstiga” tecken som å ä eller ö i filnamnet!!!!
project - project options -win32 GUI



Observera att du måste ange att biblioteken –lbgi –lgdi32 –luser32 skall skickas med till länkaren.



Skapa sedan en ny källfil i ditt projekt och lägg till koden som följer på nästa sida:



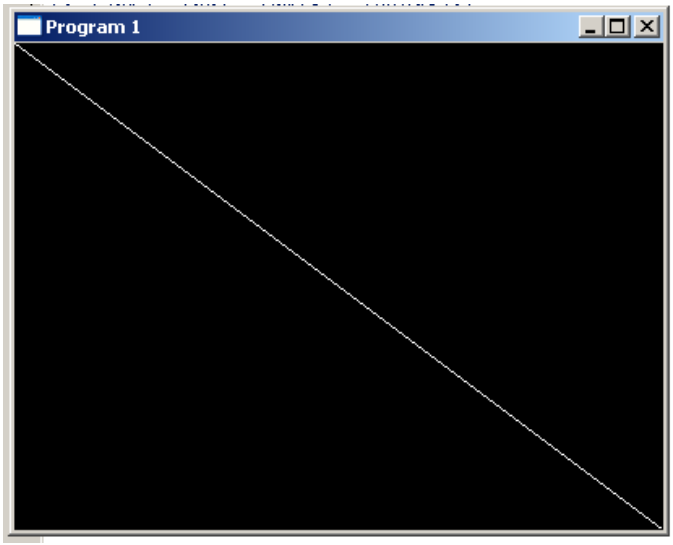

```
#include <graphics.h>
```

```
int main(void)
{
   /* initierar ett grafiskt fönster strlk 400*300 */
   initwindow(400, 300);
   setwindowtitle("Program 1");

/* Rita en linje  från punkt 0.0 till maximalt x och y värde*/
   line(0, 0, getmaxx(), getmaxy());

   /* stäng grafik återgå till text mode */
   getch();
   closegraph();
   return 0;
}
```

Om inget annat bestäms kommer ett program att från början hamna i textmode. För att få programmet att arbeta i grafisk mode, måste man först **initiera grafiken**. Detta görs med hjälp av funktionen **initwindow(int,int);.** Nu är det bara att sätta igång och rita!
Med **funktionen *line*** drar man en rät linje mellan två pixlar på skärmen. I detta exempel det längsta streck som går att dra. Från (0; 0), uppe i vänstra hörnet, till (399; 299), nere i det högra. Bilden ligga kvar tills någon tangent trycks ned. Grafiken släcks ned och systemet övergår till textmode, med funktionen *closegraph*, innan program-exekveringen avbryts.



Resultatet av ovanstående kod.

Exempel 2.

```
#include "winbgim.h"
int main()
{
      initwindow(400,400);
      setwindowtitle("Rektangel med diagonal");
      rectangle(100,50,300,300);
      line(100,50,300,300);
      outtextxy(100,320,"En rektangel med diagonal");
      outtextxy(100,350,"Valfri tangent avslutar programmet!");
      getch();
}
```

Funktionen **`rectangle(int,int,int,int)`** ritar förstås ut en rektangel parametrar till funktionen är vänstra hornet (100,50) nedre högra hörnet (300,300).

`outtextxy(100,320,"En rektangel med diagonal");`
med början i punkten 100,320 skriver funktionen **`outtextxy`** ut den text du anger som parameter.

Vi har hittills tittat på följande funktioner.

```
void initwindow (int, int);

void setwindowtitle(const char*title);

void rectangle(int,int,int,int);

void line((int,int,int,int);

void outtextxy (int, int, char const *);
```

Uppgift 1: Testa nu dessa funktioner och bekanta dig med dem, i några egna program! Prova till exempel att arbeta med större fönster: använd den upplösning du för närvarande har på skärmen.

Winbgim dokumentation på Internet

En komplett dokumentation för funktionerna i winbgim hittar du på på adressen
http://www.cs.colorado.edu/~main/bgi/doc/

Använd denna som en referens.

Färger

Om inget annat anges är ritfärgen till cirklar, rektanglar och text vit (WHITE);
För att ändra ritfärg använder man kommandot setcolor( ).
Du kan välja färg från denna lista med de ursprungliga BGI-färgerna,

nr	färg	nr	färg
0	BLACK	8	DARKGRAY
1	BLUE	9	LIGHTBLUE
2	GREEN	10	LIGHTGREEN
3	CYAN	11	LIGHTCYAN
4	RED	12	LIGHTRED
5	MAGENTA	13	LIGHTMAGENTA
6	BROWN	14	YELLOW
7	LIGHTGRAY	15	WHITE

setcolor(LIGHTRED);

men också använda en färg som du specificerar själv med utgångspunkt I **RGB** färgerna.
En färg bestäms då genom av blandning av färgerna röd, blå och grön. Var och en av färgerna ges ett värde mellan 0 och 255.
Exempel: **setcolor(COLOR(255,100,0));**

Här följer koden till ett program som ritar ut en rektangel med en diagonal, väntar på en tangenttryckning från användaren och sedan ritar ut en röd cirkel.
Funktionen **circle(int,int,int)** ritar en cirkel med radien 50. Cirkelns centrum bestäms till punkten (x=350, Y=200).

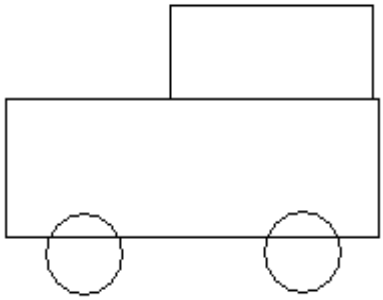
```
//Exempel 3
#include <winbgim.h>
main()
{
initwindow(500,400);
setwindowtitle("Rektangel med diagonal och en cirkel");

rectangle(100,50,300,300); // rita en rektangel
line(100,50,300,300); // rita linje
outtextxy(100,320,"En rektangel med diagonal");// text
outtextxy(100,400,"Tryck tangent!");
getch();
setcolor(RED);
circle(350,200,50);          // rita en cirkel
setcolor(COLOR(0,255,255));          //välj  färg
outtextxy(305,320,"och en cirkel");   // skriv text
getch();               // vänta på tangenttryck
}
```

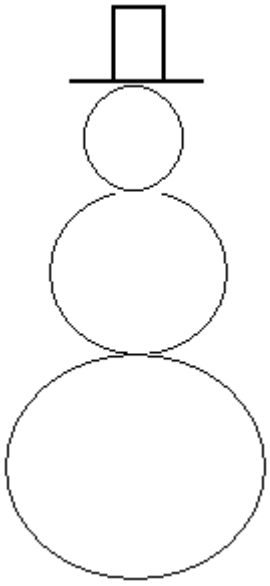
Uppgift 2: Gör nu ett program som ritar upp en kvadrat och i denna placerar en cirkel.
Försök få kvadraten att precis innesluta cirkeln.

Uppgift 2b: Ange valfri storlek på skärmen (initwindow(int,int) och undersök sedan vilka koordinater som är maxvärden för x och y genom att rita valfri figur på olika ställen på skärmen.

Uppgift 2c: Rita en bil med ungefär det utseende som bilden visar och placera den mitt på skärmen.



Uppgift 2d: Prova också att göra en snögubbe.



Floodfill

```
void floodfill(int x, int y, int border);
```

floodfill fyller ett område med färg/mönster. Utgångspunkten för fyllningen är förstås x och y. Om koordinaten x,y befinner sig inom ett inhägnat område fylls detta. Befinner sig x,y utanför ”inhägnaden” kommer området utanför att fyllas. Den tredje parametern anger den färg som inhägnar området.
För att använda floodfill måste du också ange **fillstyle.** Vi behöver inte gå så djupt in på fillstyle just nu det räcker med att använda koden här under:

```
setfillstyle(SOLID_FILL, YELLOW);//anger fyllningsstil och färg
```

SOLID_FILL fyller området med färg. Den andra parametern anger vald färg eller nyans.
Kompilera och provkör exemplet.

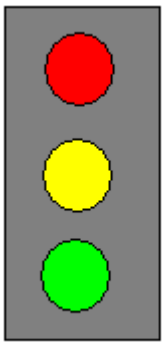
Exempel 4:

```
#include <iostream>
#include <winbgim.h>

int main()
{
initwindow(500,400);
setwindowtitle("Rektangel med diagonal och en boll");

rectangle(100,50,300,300);          // rita en rektangel
line(100,50,300,300);                   // rita linje
outtextxy(100,320,"En rektangel med diagonal");  // skriv text
outtextxy(100,400,"Tryck tangent!");
getch();
setcolor(RED);                           //en av 16 färger
circle(350,200,50);                       // rita en cirkel
setcolor(COLOR(0,255,255));            //välj RGB färg
outtextxy(305,320,"och en boll")      // skriv text
 setfillstyle(SOLID_FILL, YELLOW);  //anger fyllningsstil
floodfill(355,205,RED);             //fyller röd inhägnad med gult
getch();                            // vänta på tangenttryck
}
```

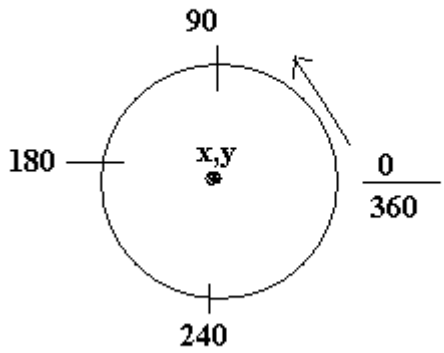
Uppgift 3: Skapa ett trafikljus som ser ut ungefär så här.



ARC

void arc(int x, int y, int stangle, int endangle, int radius);
arc draws a circular arc in the current drawing color centered at (x,y) with a radius given
Med metoden **arc** gör du en båge –en del av en cirkel med radien angiven i radius, startpunkten i stangle och slutpunkten i endangle. Int x och y anger cirkelns centrum.

ARC (x, y, start, stop ,radie)



setlinestyle

setlinestyle(int linestyle, unsigned upattern, int thickness);
setlinestyle bestämmer med vilken stil efterföljande linjer kommer att ritas ut.

Name	Value	Description
SOLID_LINE	0	Solid line
DOTTED_LINE	1	Dotted line
CENTER_LINE	2	Centered line
DASHED_LINE	3	Dashed line
USERBIT_LINE	4	User-defined line style

Med thickness bestämmer du förstås linjens tjocklek.

Name	Value	Description
NORM_WIDTH	1	1 pixel wide
THICK_WIDTH	3	3 pixels wide

För att få en tjockare linje, normalvärdet är 1 pixel räcker det om du anger värde för thicknes
t.ex. setlinestyle(0, 0, 5);
Mer info om setlinestyle hittar du på:
http://www.cs.colorado.edu/~main/bgi/doc/

Uppgift 4: Skapa en smile-gubbe med ungefär detta utseende.

Grafik: Enkel animering

Rita cirklar:
Skriv in, kompilera och kör detta program:

```
#include<winbgim.h>

int main()
{
    initwindow(600,400);

    for(int i=0;i<500; i++)
    {
    circle(50+i, 200, 50);
    getch();
    outtextxy(250,320, "TRYCK ENTER !");
    }
    closegraph();
return 0;
}
```

I for-satsen ritas det en cirkel med medelpunkten 50+i, 200. Cirkeln har radien 50 pixlar.
Ingen av cirklarna som ritas tas bort.
För att skapa illusionen av en cirkel som rör sig från vänster till höger måste varje cirkel tas bort innan en ny ritas ut, detta åstadkommer man genom att rita cirkeln en gång till, men denna gång med bakgrundsfärgen.

Uppgift 16:1 Gör nu om programmet så att cirkeln ser ut att förflytta sig:
Pseudokod:

- Starta grafiken.
- Loopa: Rita cirkeln..
  Rita cirkeln igen med bakgrundsfärgen
  Gör en kort paus //delay() eller tangenttryckning.
  Flytta fram medelpunkten
- Avsluta

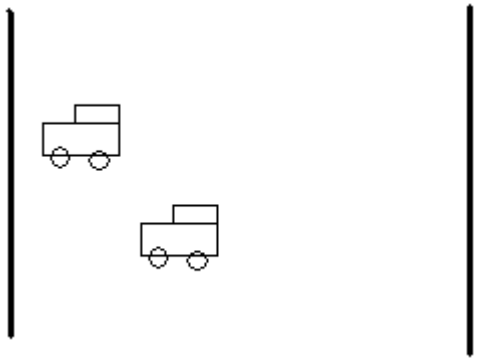
Bilrally

Mer grafik.. En litet större uppgift. Arbeta gärna två och två.

Uppgift 16:2
Ni ska nu skriva ett bilrallyprogram.
På skärmen ska två bilar ritas ut kanske kan den bil du ritat tidigare återanvändas? Dessutom skall man se mållinjen.
När användaren trycker ENTER skall loppet starta. Bilarna hastighet skall styras slumpmässigt. När en av bilarna når mållinjen avbryts loppet och resultatet skrivs ut på skärmen.

Pseudokod:

- Förbered grafik
- Förbered bana
- Rita bilar i startposition
- Avvakta tangenttryckning
- Starta loppet
- Så länge ingen bil nått målet:
  Sudda bilarna, bestäm position i x-led, rita bilarna
  Rita bilarna
  Gör en kort paus
- Skriv ut resultatet
- Invänta tangent tryckning innan avbrott



En, minst sagt, fartfylld bild! Vi ser här hur den nedre, av de båda rallybilarna, har ryckt åt sig, en som det kan tyckas ointaglig ledning. Men än återstår mer än halva loppet …… ……..
Spänning, fart och dramatik!!!!!