Решени примерни задачи за контролни и изпит по ГС

ТЕМА 3
2D графика. Представяне на експериментални реални данни в зададен графичен прозорец

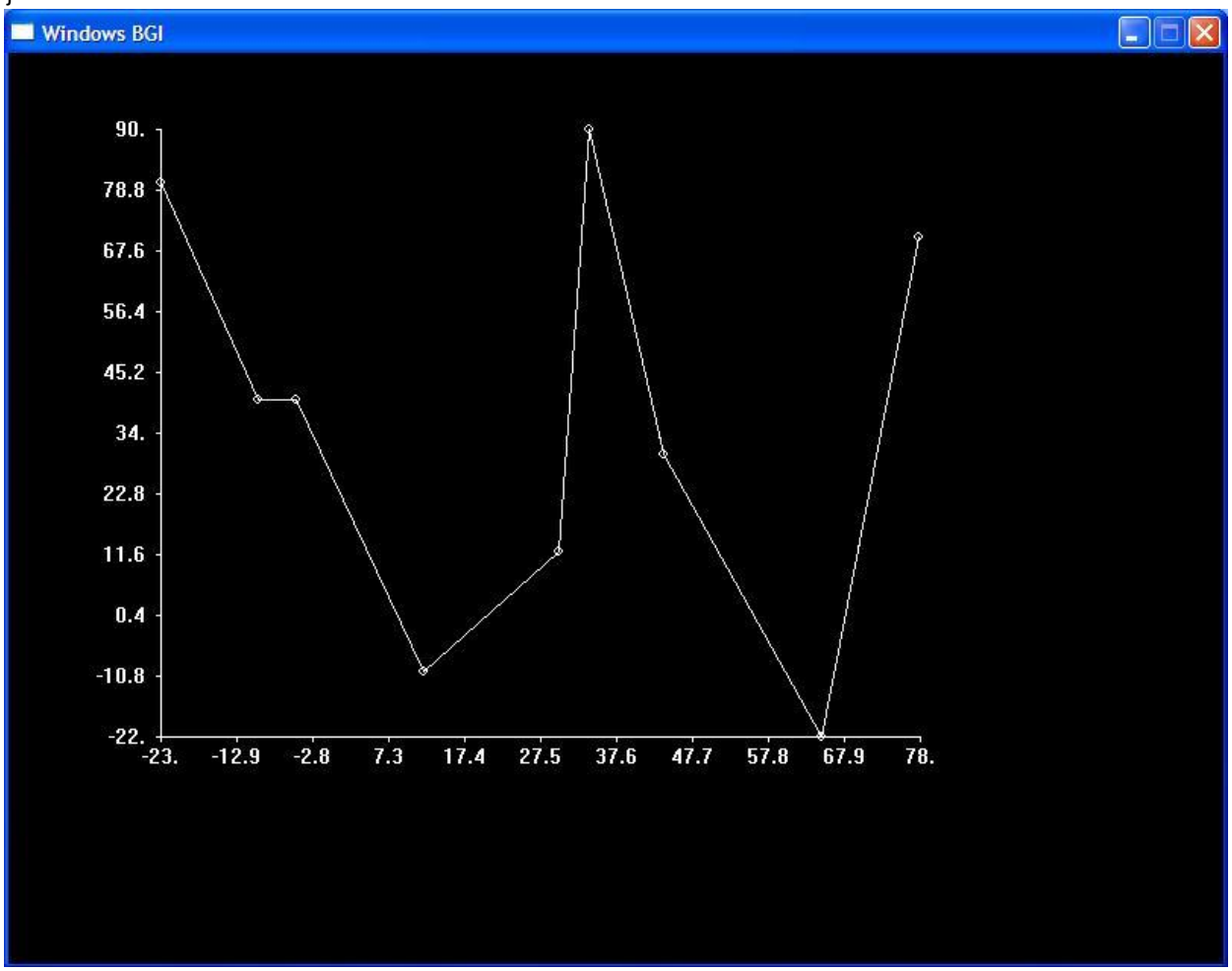
Задача 3.1

Входните данни са двойки реални стойности, които могат да бъдат прочетени от файл, клавиатура , или са зададени в два масива. Целта е да се преобразуват в координати на пиксели, които се представят като центрове на окръжности с малък радиус и се съединяват с отсечки в зададен от потребителя графичен прозорец (фиг.3).

```
#include <graphics.h>
#include <iostream>
using namespace std;
int main()
{
//задаване на входните данни - двойки стойности реални числа, които ще преобразуваме в
координати на пиксели
// тези входни данни могат да постъпват от файл или клавиатура или в масив
  float x[] = {-5, 12, 78, -23, 34, -10, 65, 30, 44},temp;
  float y[] = {40, -10, 70, 80, 90, 40, -22, 12, 30};
  int n=sizeof(x)/sizeof(x[0]); // определяне на броя на входните данни
  int i,j;
  int winwidth=800,winheight=600; // параметри на прозореца на графичната система
  int Px=500,Py=400,Dx=50,Dy=40,x0=100,y0=450 ; //параметри на графичния прозорец, в който ще
//се изобразят данните (вътре в прозореца на графичната система)

// сортиране на входните данни, в случай, че те са експериментални
  for(i = 0; i < n; i++)
  {
     for(j = 0; j < n-i-1; j++)
     {
        if(x[j] > x[j+1])
        {
           temp = x[j]; x[j]=x[j+1]; x[j+1]=temp;
           temp = y[j]; y[j]=y[j+1]; y[j+1]=temp;
        }
      }
   }
//намиране на диапазона на изменение на входните данни - xmin, xmax, ymin, ymax
// след сортировката xmin и xmax са съответно първия и последен елемент от масива x
  float xmin = x[0];
  float xmax = x[n-1];

// намиране на ymin и ymax
  float ymin = y[0];
  float ymax = y[0];
  for(int i = 1; i< n; i++) {
```

```
        if(y[i] < ymin) ymin = y[i];
        if(y[i] > ymax) ymax = y[i];
  }

//определяне на скалните коефициенти
  float sx = (xmax - xmin)/Px;
  float sy = (ymax - ymin)/Py;

// инициализация на графичната система чрез отваряне на графичен прозорец със зададен размер
  initwindow(winwidth,winheight);

//изчертаване на графичния прозорец
  line(x0,y0,x0+Px,y0);  //хоризонтална ос
  line(x0,y0,x0,y0-Py);  //вертикална ос
  int Ip = Px/Dx; int Jp = Py/Dy; //брой деления по хоризонталната и вертикалната ос

//изчертаване и надписване на деленията по хоризонталната ос
    char text[10];
    for(i = 0; i <= Ip; i++)
    {
    line(x0 + i*Dx, y0, x0+i*Dx,y0+3);   //изчертаване на деленията
    gcvt(xmin + i*Dx*sx, 5.2,text);      //преобразуване на реалната стойност, съответстваща на
//делението в символен низ
    settextjustify(1,2);
    outtextxy(x0 + i*Dx, y0+5, text);// извеждане на стойността, съответстваща на делението
    }

//изчертаване и надписване на деленията по вертикалната ос
    for(i = 0; i <= Jp; i++)
    {
    line(x0, y0-i*Dy, x0-3, y0- i * Dy);  //изчертаване на деленията
    gcvt(ymin+i*Dy*sy, 5.2,text);        //преобразуване на реалната стойност, съответстваща на
//делението в символен низ
    settextjustify(2,1);
    outtextxy(x0-10, y0 - Dy*i+5, text); // извеждане на стойността, съответстваща на делението
          }

//преобразуване на входните дании в координати на пиксели, които се използват за центрове на
//окръжности с радиус 3 пиксела
  for(i = 0 ; i < n; i ++)
  {
  int xprim=x0 + (x[i]-xmin)/sx;
  int yprim=y0 - (y[i]-ymin)/sy;
  circle(xprim,yprim,3);
  }

//свързване на окръжностите с отсечки и получаване на 2D графика, съответстваща на входните
//данни x,y
  for(i = 0; i < n-1; i++) {
```

```
            int xa = x0 + (x[i] - xmin)/sx;
            int ya = y0 - (y[i] - ymin)/sy;
            int xb = x0 + (x[i+1] - xmin)/sx;
            int yb = y0 - (y[i+1] - ymin)/sy;
            line(xa, ya, xb, yb);
    }
    getch();
    return 0;
}
```



Фиг.3. Представяне на експериментални данни в графичен прозорец.

ТЕМА 4
2D графика. Графично представяне на зависимости зададени чрез функции в зададен графичен прозорец

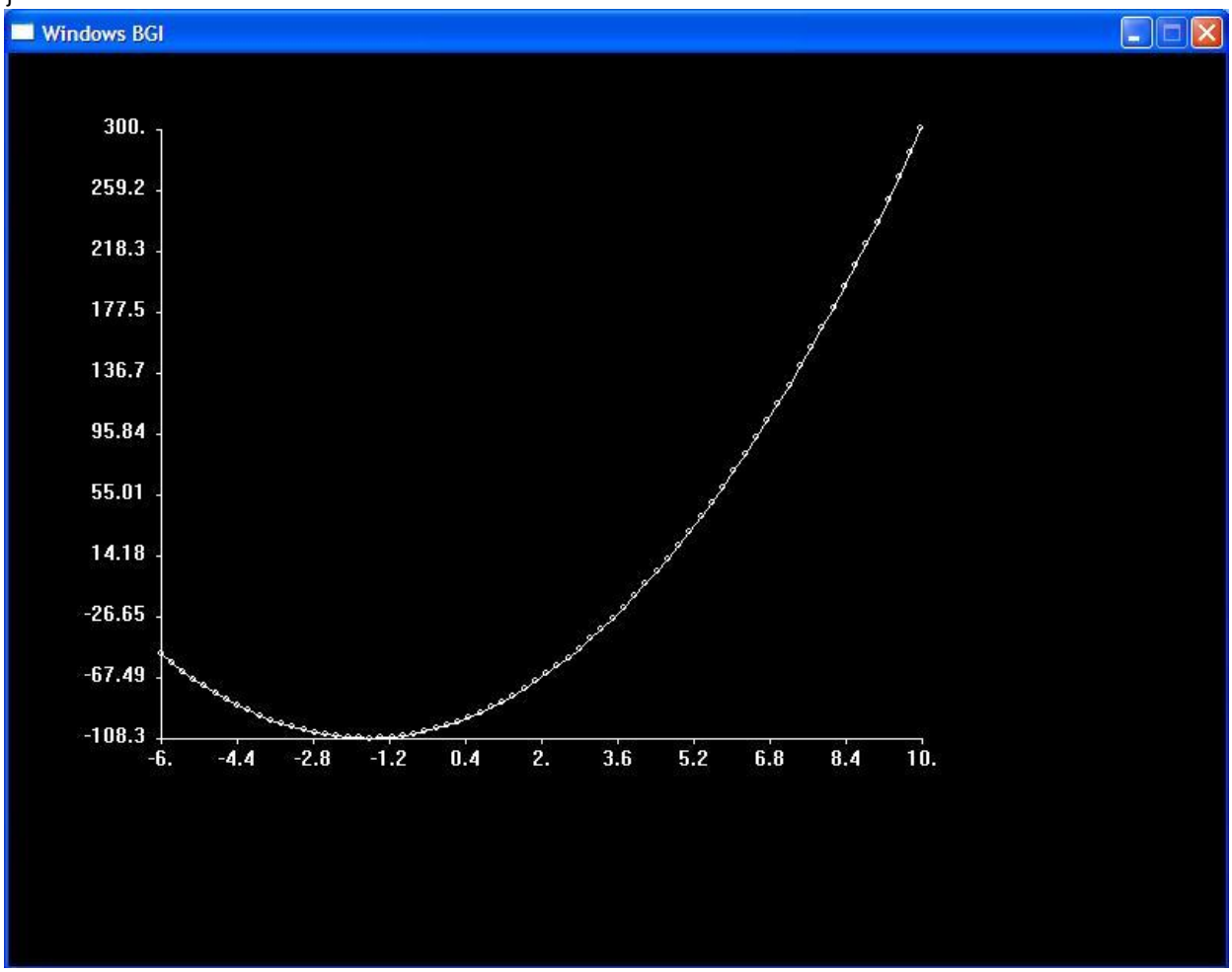
Задача 4.1

Входните данни са двойки реални стойности, които могат да бъдат изчислени по зададена функция y =3*x^2+10*x-100 ***в зададен диапазон*** -6<=x<=10 ***и зададен брой стойности-40, които трябва да се изчислят. Изчислените данни запълват два масива и алгоритъма продължава като в задача 3.1. Целта е да се преобразуват в координати на пиксели, които се представят като центрове на окръжности с малък радиус и се съединяват с отсечки в зададен от потребителя графичен прозорец.С червен цвят са отбелязани разликите със задача 3.***

```
#include <graphics.h>
#include <iostream>
using namespace std;

int main()
{
//в задачата се изобразява функцията 3*x^2+10*x-100 за стойности на x в диапазона -6<=x<=10.
//Изобразяват се 40 стойности на функцията в зададения диапазон (n=40)
//входните данни се получават от известната функция при зададен диапазон на изменение на x  от
//xmin до xmax и зададен брой стойности n, които трябва да получим
  float xmin = -6;
  float xmax = 10;
  int n=40;
  int i;
  double x[n];
  double y[n];
  int winwidth=800,winheight=600; // параметри на прозореца на графичната система
  int Px=500,Py=400,Dx=50,Dy=40,x0=100,y0=450 ; //параметри на графичния прозорец, в който ще
//се изобразят данните (вътре в прозореца на графичната система)

//изчисляване на стъпката на изменение на входните данни, необходима за получаване на n броя
//стойности в масива x
  double dx = (xmax-xmin) / (n-1);

// попълване на входните масиви с изчислените данни
    for (i = 0; i < n; i ++)
    {
          x[i] = xmin + i * dx ;
          y[i] = 3*x[i]*x[i] + 10*x[i] -100;  //3*x^2+10*x-100
    }

// намиране на ymin и ymax
  double ymin = y[0];
  double ymax = y[0];
```

```
  for(i = 0; i< n; i++) {
          if(y[i] < ymin) ymin = y[i];
          if(y[i] > ymax) ymax = y[i];
  }

//определяне на скалните коефициенти
  float sx = (xmax - xmin)/Px;
  float sy = (ymax - ymin)/Py;

// инициализация на графичната система чрез отваряне на графичен прозорец със зададен размер
  initwindow(winwidth,winheight);

//изчертаване на графичния прозорец
  line(x0,y0,x0+Px,y0);//хоризонтална ос
  line(x0,y0,x0,y0-Py);//вертикална ос

  int Ip = Px/Dx; int Jp = Py/Dy;//брой деления по хоризонталната и вертикалната ос

//изчертаване и надписване на делениятa по хоризонталната ос
    char text[10];
    for(i = 0; i <= Ip; i++)
    {
    line(x0 + i*Dx, y0, x0+i*Dx,y0+3);   //изчертаване на делениятa
    gcvt(xmin + i*Dx*sx, 5.2,text);      //преобразуване на реалната стойност, съответстваща на
//делението в символен низ
     settextjustify(1,2);
    outtextxy(x0 + i*Dx, y0+5, text);// извеждане на стойността, съответстваща на делението
    }

//изчертаване и надписване на делениятa по вертикалната ос
    for(i = 0; i <= Jp; i++)
    {
    line(x0, y0-i*Dy, x0-3, y0- i * Dy);  //изчертаване на делениятa
    gcvt(ymin+i*Dy*sy, 4.2,text);        //преобразуване на реалната стойност, съответстваща на
//делението в символен низ
    settextjustify(2,1);
    outtextxy(x0-10, y0 - Dy*i+5, text); // извеждане на стойността, съответстваща на делението
          }

//преобразуване на входните дании в координати на пиксели, които се използват за центрове на
//окръжности с радиус 2 пиксела
  for(i = 0 ; i < n; i ++)
  {
  int xprim=x0 + (x[i]-xmin)/sx;
  int yprim=y0 - (y[i]-ymin)/sy;
  circle(xprim,yprim,2);
  }
```

```
 //свързване на окръжностите с отсечки и получаване на 2D графика, съответстваща на входните
//данни x,y
  for(i = 0; i < n-1; i++) {
          int xa = x0 + (x[i] - xmin)/sx;
          int ya = y0 - (y[i] - ymin)/sy;
          int xb = x0 + (x[i+1] - xmin)/sx;
          int yb = y0 - (y[i+1] - ymin)/sy;
          line(xa, ya, xb, yb);
  }
  getch();
  return 0;
}
```



Фиг.4.1 ***Графика на функцията*** **y =3*x^2+10*x-100** ***в зададен диапазон*** **-6<=x<=10** ***и зададен брой стойности 40,***

Задача 4.2

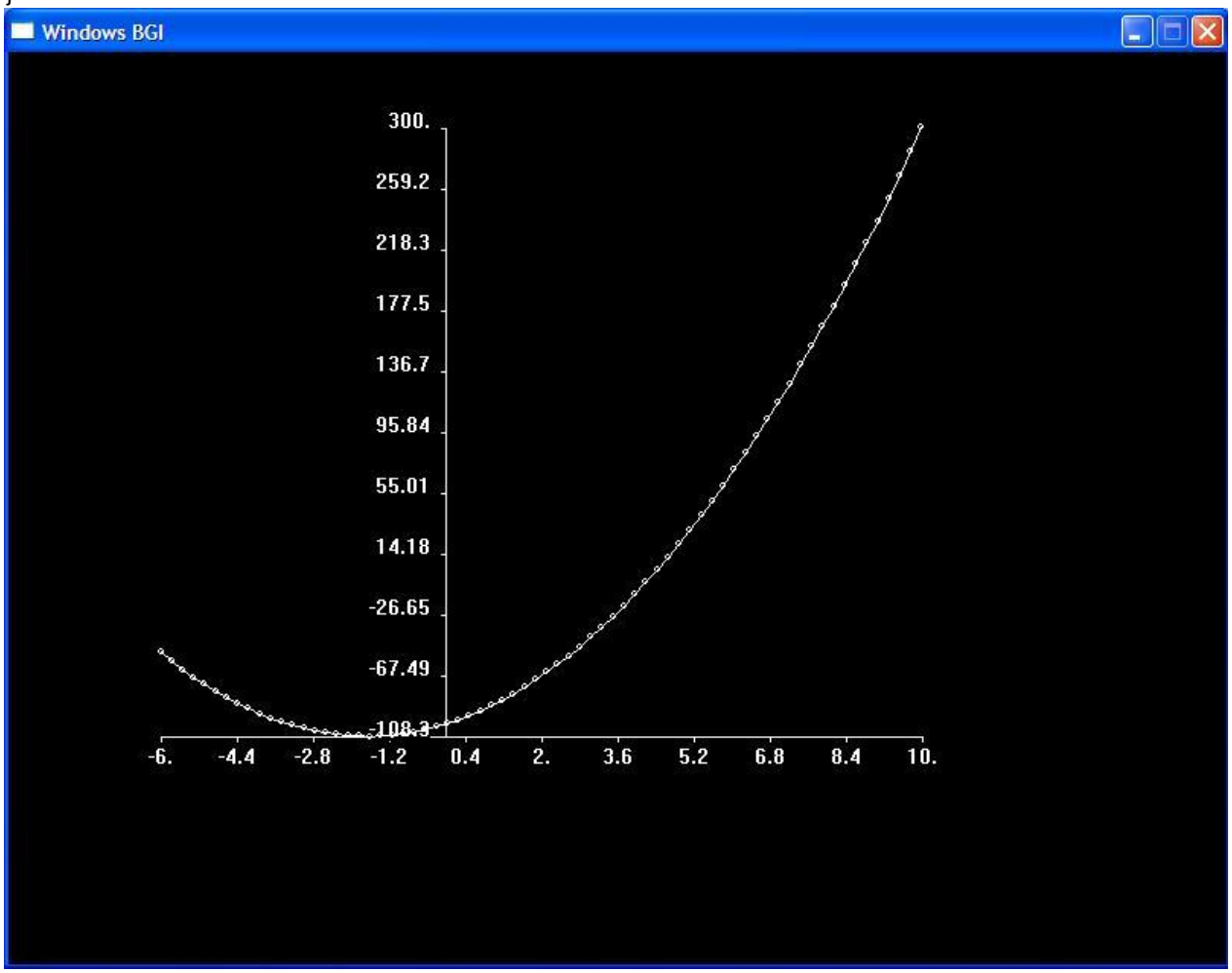
Вариант на задача 4.1 . Задачата включва текста на задача 4.1 и допълнителното изискване осите на изображението да преминават през нулевите значения на данните. Пример с вертикална ос минаваща през стойността 0 на абцисата на фиг. 4.2

За целта е необходимо да се изчисли новото начало на осите, така че то да отговаря на нулеви значения. В червен цвят са отбелязани разликите със задача 4.1

```
#include <graphics.h>
#include <iostream>
using namespace std;

int main()
{
//вариант на задачата от lab4 с изместване на вертикалната ос през x=0
//в задачата се изобразява функцията 3*x^2+10*x-100 за стойности на x в диапазона -6<=x<=10.
//Изобразяват се 40 стойности на функцията в зададения диапазон (n=40)
//входните данни се получават от известната функция при зададен диапазон на изменение на x  от
//xmin до xmax и брой стойности n, които трябва да получим
  float xmin = -6;
  float xmax = 10;
  int n=70;
  int i;
  double x[n];
  double y[n];
  int winwidth=800,winheight=600; // параметри на прозореца на графичната система
  int Px=500,Py=400,Dx=50,Dy=40,x0=100,y0=450 ; //параметри на графичния прозорец, в който ще
//се изобразят данните (вътре в прозореца на графичната система)

//изчисляване на стъпката на изменение на входните данни, необходима за получаване на n броя
//стойности в масива x
  double dx = (xmax-xmin) / (n-1);

// попълване навходните масиви с изчислените данни
  for (i = 0; i < n; i ++)
   {
          x[i] = xmin + i * dx ;
          y[i] = 3*x[i]*x[i] + 10*x[i] -100; //3*x^2+10*x-100
   }

// намиране на ymin и ymax
  double ymin = y[0];
  double ymax = y[0];

  for(i = 0; i< n; i++) {
          if(y[i] < ymin) ymin = y[i];
          if(y[i] > ymax) ymax = y[i];
  }

//определяне на скалните коефициенти
  float sx = (xmax - xmin)/Px;
  float sy = (ymax - ymin)/Py;

// инициализация на графичната система чрез отваряне на графичен прозорец със зададен размер
  initwindow(winwidth,winheight);

//изчертаване на графичния прозорец
```

```
  int x0n=x0;
  int y0n=y0;
  if (xmin<0) x0n=x0+(0-xmin)/sx;
   if (ymin<0) y0n=y0-(0-ymin)/sy;
  line(x0,y0n,x0+Px,y0n);//хоризонтзлнз ос
  line(x0n,y0,x0n,y0-Py);//вертикална ос

  int Ip = Px/Dx; int Jp = Py/Dy;//брой деления по хоризонталната и вертикалната ос

//изчертаване и надписване на деленията по хоризонталната ос
    char text[10];
    for(i = 0; i <= Ip; i++)
    {
    line(x0 + i*Dx, y0n, x0+i*Dx,y0n+3);   //изчертаване на деленията
    gcvt(xmin + i*Dx*sx, 5.2,text);     //преобразуване на реалната стойност, съответстваща на
//делението в символен низ
    settextjustify(1,2);
    outtextxy(x0 + i*Dx, y0n+5, text);// извеждане на стойността, съответстваща на делението
    }

//изчертаване и надписване на деленията по вертикалната ос
    for(i = 0; i <= Jp; i++)
    {
    line(x0n, y0-i*Dy, x0n-3, y0- i * Dy);  //изчертаване на деленията
    gcvt(ymin+i*Dy*sy, 5.2,text);       //преобразуване на реалната стойност, съответстваща на
//делението в символен низ
    settextjustify(2,1);
    outtextxy(x0n-10, y0 - Dy*i+5, text); // извеждане на стойността, съответстваща на делението
            }

//преобразуване на входните дании в координати на пиксели, които се използват за центрове на
//окръжности с радиус 2 пиксела
  for(i = 0 ; i < n; i ++)
  {
  int xprim=x0 + (x[i]-xmin)/sx;
  int yprim=y0 - (y[i]-ymin)/sy;
  circle(xprim,yprim,2);
  }

  //свързване на окръжностите с отсечки и получаване на 2D графика, съответстваща на входните
//данни x,y
  for(i = 0; i < n-1; i++) {
            int xa = x0 + (x[i] - xmin)/sx;
            int ya = y0 - (y[i] - ymin)/sy;
            int xb = x0 + (x[i+1] - xmin)/sx;
            int yb = y0 - (y[i+1] - ymin)/sy;
            line(xa, ya, xb, yb);
  }
```

```
  getch();
  return 0;
}
```



Фиг.4.2 ***Графика на функцията*** **y =3*x^2+10*x-100** ***в зададен диапазон*** **-6<=x<=10** ***и зададен брой стойности 40,с изместване на вертикалната ос до положение x0n, отговарящо на величина x=0.***

Задача 4.3

Друг вариант на задачад 4.2 с хоризонтална ос, минаваща през нулевите значения на ординатата.

Разликата със задача 4.2 е тази, че се премества и хоризонталната ос на положение y0n, отговарящо на величина y =0 (фиг.4.3) .

За целта е необходимо да се изчисли колко пиксела отговарят на изменение на входната величина от ymin до 0 (в случай, че ymin <0)

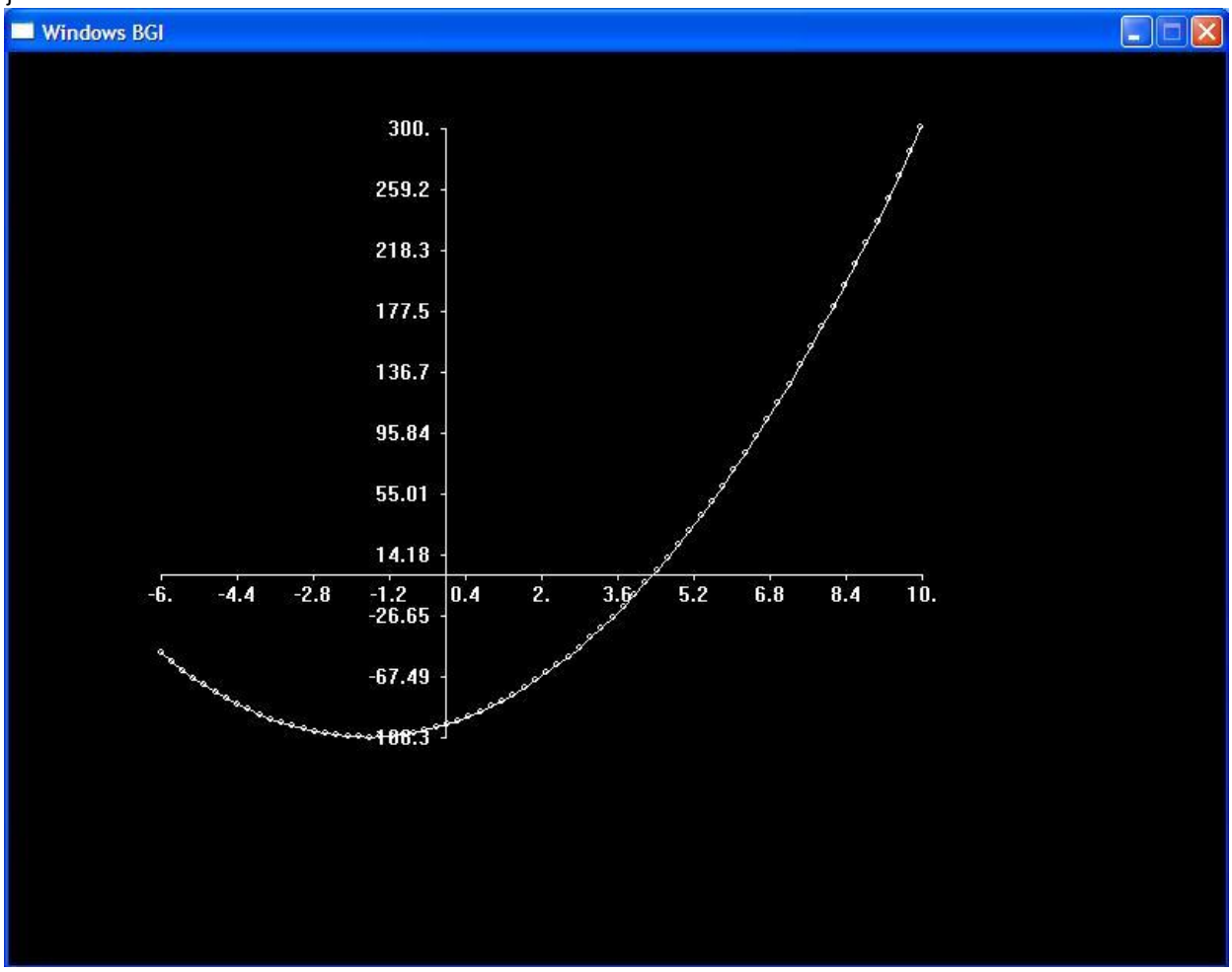
В червен цвят са отбелязани разликите със задача 4.2

```
#include <graphics.h>
#include <iostream>
using namespace std;
```

```
int main()
{
//вариант на задача 4.1 с изместване и на вертикалната ос през x=0 и хоризонталната ос през y=0
//в задачата се изобразява функцията 3*x^2+10*x-100 за стойности на x в диапазона -6<=x<=10.
//Изобразяват се 40 стойности на функцията в зададения диапазон (n=40)
  float xmin = -6;
  float xmax = 10;
  int n=40;
  int i;
  double x[n];
  double y[n];
  int winwidth=800,winheight=600; // параметри на прозореца на графичната система
  int Px=500,Py=400,Dx=50,Dy=40,x0=100,y0=450 ; //параметри на графичния прозорец, в който ще
//се изобразят данните (вътре в прозореца на графичната система)

//изчисляване на стъпката на изменение на входните данни, необходима за получаване на n броя
//стойности в масива x
  double dx = (xmax-xmin) / (n-1);

// попълване навходните масиви с изчислените данни
 for (i = 0; i < n; i ++)
  {
          x[i] = xmin + i * dx ;
          y[i] = 3*x[i]*x[i] + 10*x[i] -100; //3*x^2+10*x-100
  }

// намиране на ymin и ymax
  double ymin = y[0];
  double ymax = y[0];

  for(i = 0; i< n; i++) {
          if(y[i] < ymin) ymin = y[i];
          if(y[i] > ymax) ymax = y[i];
  }

//определяне на скалните коефициенти
  float sx = (xmax - xmin)/Px;
  float sy = (ymax - ymin)/Py;

// инициализация на графичната система чрез отваряне на графичен прозорец със зададен размер
  initwindow(winwidth,winheight);

//изчертаване на графичния прозорец
//намиране на x-координатата, съответстваща на стойността x=0 в случай, че xmin<0
  int x0n=x0;
  if (xmin<0) x0n=x0+(0-xmin)/sx;

//намиране на y-координатата, съответстваща на стойността y=0 в случай, че ymin<0
```

```
  int y0n=y0;
  if (ymin<0) y0n=y0-(0-ymin)/sy;

  line(x0,y0n,x0+Px,y0n);//хоризонтална ос
  line(x0n,y0,x0n,y0-Py);//вертикална ос

  int Ip = Px/Dx; int Jp = Py/Dy;//брой деления по хоризонталната и вертикалната ос

//изчертаване и надписване на деленията по хоризонталната ос
    char text[10];
    for(i = 0; i <= Ip; i++)
    {
    line(x0 + i*Dx, y0n, x0+i*Dx,y0n+3);   //изчертаване на деленията
    gcvt(xmin + i*Dx*sx, 5.2,text);      //преобразуване на реалната стойност, съответстваща на
//делението в символен низ
    settextjustify(1,2);
    outtextxy(x0 + i*Dx-10, y0n+5, text);// извеждане на стойността, съответстваща на делението
    }

//изчертаване и надписване на деленията по вертикалната ос
    for(i = 0; i <= Jp; i++)
    {
    line(x0n, y0-i*Dy, x0n-3, y0- i * Dy);  //изчертаване на деленията
    gcvt(ymin+i*Dy*sy, 4.2,text);        //преобразуване на реалната стойност, съответстваща на
//делението в символен низ
    settextjustify(2,1);
    outtextxy(x0n-10, y0 - Dy*i+5, text); // извеждане на стойността, съответстваща на делението
            }

//преобразуване на входните дании в координати на пиксели, които се използват за центрове на
//окръжности с радиус 2 пиксела
  for(i = 0 ; i < n; i ++)
  {
  int xprim=x0 + (x[i]-xmin)/sx;
  int yprim=y0 - (y[i]-ymin)/sy;
  circle(xprim,yprim,2);
  }

  //свързване на окръжностите с отсечки и получаване на 2D графика, съответстваща на входните
//данни x,y
  for(i = 0; i < n-1; i++) {
            int xa = x0 + (x[i] - xmin)/sx;
            int ya = y0 - (y[i] - ymin)/sy;
            int xb = x0 + (x[i+1] - xmin)/sx;
            int yb = y0 - (y[i+1] - ymin)/sy;
            line(xa, ya, xb, yb);
  }
  getch();
  return 0;
```

}



Фиг.4.3 ***Графика на функцията*** **y =3*x^2+10*x-100** ***в зададен диапазон*** **-6<=x<=10** ***и зададен брой стойности 40,с изместване на вертикалната ос до положение x0n, отговарящо на величина x=0 иизместване на хоризонталната ос на положение y0n, отговарящо на величина y =0***

Задача 4.4

Пример с представяне на две функции в един графичен прозорец.

Да се предтавят функциите: y =3*x^2+10*x-100 ***в зададен диапазон*** -6<=x<=10 ***и чрез 40 точки и функцията*** y2=2*x2^2-8*x2-30 ***в зададен диапазон*** -2<=x2<=20 *чрез 50 точки,* ***които трябва да се изчислят (фиг.4.4) Изчислените данни запълват нови два масива. Пресмятат се общи скални коефициенти за двете функции. В червен цвят са отбелязани разликите със задача 4.1***

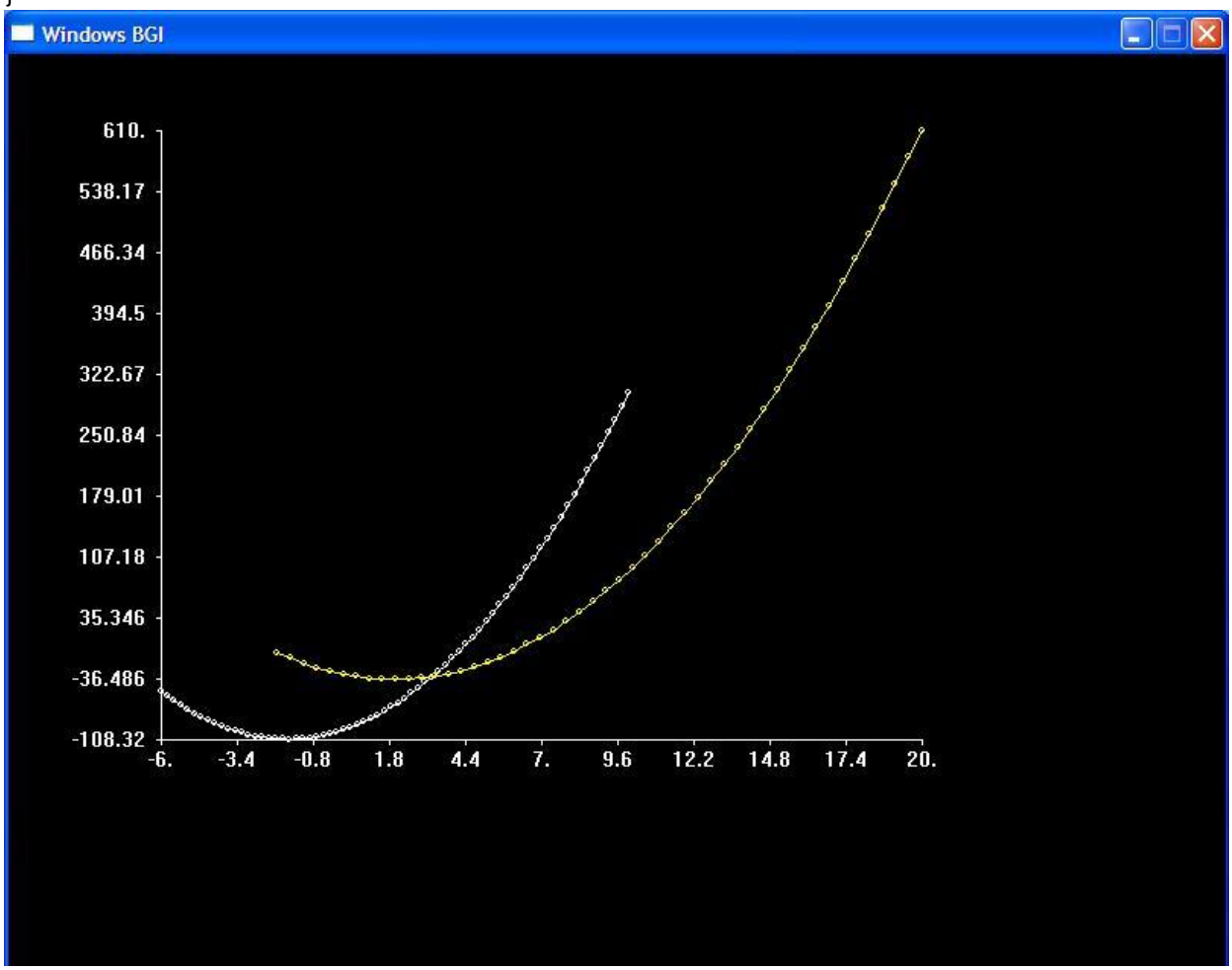
```
#include <graphics.h>
#include <iostream>
using namespace std;

int main()
{
```

```
//в задачата се изобразяват две функции в един графичен прозорец с общи скални коефициенти
// Изобразява се първата  функция y1=3*x1^2+10*x1-100 за стойности на x в диапазона -6<=x1<=10.
//Изобразяват се 40 стойности на функцията в зададения диапазон (n1=40)
// Добавчме още една функция y2=2*x2^2-8*x2-30 за стойности на x2 в диапазона -2<=x2<=20.
//Изобразяват се 50 стойности на функцията в зададения диапазон (n2=50)
//входните данни се получават от известните функции при зададени диапазони на изменение на
//x1  от x1min до x1max и брой стойности n1,и на x2  от x2min до x2max
//и брой стойности n2, които трябва да получим от функциите съответно y1 и y2
  double x1min = -6;
  double x1max = 10;
  int n1=70;
  int i;
  double x1[n1];
  double y1[n1];
  double x2min = -2;
  double x2max = 20;
  int n2=50;
  double x2[n2];
  double y2[n2];
  double xmin;
  double xmax;
  double ymin;
  double ymax;
  int winwidth=800,winheight=600; // параметри на прозореца на графичната система
  int Px=500,Py=400,Dx=50,Dy=40,x0=100,y0=450 ; //параметри на графичния прозорец, в който ще
//се изобразят данните (вътре в прозореца на графичната система)

//изчисляване на стъпката на изменение на входните данни, необходима за получаване на n1 броя
//стойности в масива x1
  double dx1 = (x1max-x1min) / (n1-1);

//изчисляване на стъпката на изменение на входните данни, необходима за получаване на n2 броя
//стойности в масива x2
  double dx2 = (x2max-x2min) / (n2-1);


// попълване на входните масиви x1 и y1 с изчислените данни
    for (i = 0; i < n1; i ++)
{
          x1[i] = x1min + i * dx1 ;
          y1[i] = 3*x1[i]*x1[i] + 10*x1[i] -100; //y1=3*x1^2+10*x1-100
  }

// попълване на входните масиви x2 и y2 с изчислените данни
    for (i = 0; i < n2; i ++)
 {
          x2[i] = x2min + i * dx2 ;
          y2[i] = 2*x2[i]*x2[i] - 8*x2[i] -30;    //y2=2*x2^2-8*x2-30
  }
```

```
// намиране на y1min и y1max
  double y1min = y1[0];
  double y1max = y1[0];

  for(i = 0; i< n1; i++)
{
        if(y1[i] < y1min) y1min = y1[i];
        if(y1[i] > y1max) y1max = y1[i];
  }

// намиране на y2min и y2max
  double y2min = y2[0];
  double y2max = y2[0];

  for(i = 0; i< n2; i++)
{
        if(y2[i] < y2min) y2min = y2[i];
        if(y2[i] > y2max) y2max = y2[i];
  }

// намиране на общи минимуми и максимуми
 xmin=x1min;
 if(x2min<xmin) xmin=x2min;
  xmax=x1max;
 if(x2max>xmax) xmax=x2max;

 ymin=y1min;
 if(y2min<ymin) ymin=y2min;
  ymax=y1max;
 if(y2max>ymax) ymax=y2max;

//определяне на скалните коефициенти, които са еднакви за двете функции
  double sx = (xmax - xmin)/Px;
  double sy = (ymax - ymin)/Py;

// инициализация на графичната система чрез отваряне на графичен прозорец със зададен размер
  initwindow(winwidth,winheight);

//изчертаване на графичния прозорец
  line(x0,y0,x0+Px,y0);//хоризонтзлнз ос
  line(x0,y0,x0,y0-Py);//вертикална ос

  int Ip = Px/Dx; int Jp = Py/Dy;//брой деления по хоризонталната и вертикалната ос

//изчертаване и надписване на деленията по хоризонталната ос
    char text[10];
    for(i = 0; i <= Ip; i++)
    {
```

```
    line(x0 + i*Dx, y0, x0+i*Dx,y0+3);   //изчертаване на деленията
    gcvt(xmin + i*Dx*sx, 5.2,text);      //преобразуване на реалната стойност, съответстваща на
//делението в символен низ
    settextjustify(1,2);
    outtextxy(x0 + i*Dx, y0+5, text);// извеждане на стойността, съответстваща на делението
    }

//изчертаване и надписване на деленията по вертикалната ос
    for(i = 0; i <= Jp; i++)
    {
    line(x0, y0-i*Dy, x0-3, y0- i * Dy);  //изчертаване на деленията
    gcvt(ymin+i*Dy*sy, 5.2,text);        //преобразуване на реалната стойност, съответстваща на
//делението в символен низ
    settextjustify(2,1);
    outtextxy(x0-10, y0 - Dy*i+5, text); // извеждане на стойността, съответстваща на делението
            }

//преобразуване на входните дании x1,y1  в координати на пиксели, които се използват за
//центрове на  окръжности с радиус 2 пиксела
  for(i = 0 ; i < n1; i ++)
  {
  int x1prim=x0 + (x1[i]-xmin)/sx;
  int y1prim=y0 - (y1[i]-ymin)/sy;
  circle(x1prim,y1prim,2);
  }

//преобразуване на входните дании x2,y2  в координати на пиксели, които се използват за
//центрове на  окръжности с радиус 2 пиксела
  setcolor(YELLOW);
   for(i = 0 ; i < n2; i ++)
  {
  int x2prim=x0 + (x2[i]-xmin)/sx;
  int y2prim=y0 - (y2[i]-ymin)/sy;
  circle(x2prim,y2prim,2);
  }
 //свързване на окръжностите с отсечки и получаване на 2D графика, съответстваща на входните
//данни x1,y1
 setcolor(WHITE);
  for(i = 0; i < n1-1; i++) {
            int xa = x0 + (x1[i] - xmin)/sx;
            int ya = y0 - (y1[i] - ymin)/sy;
            int xb = x0 + (x1[i+1] - xmin)/sx;
            int yb = y0 - (y1[i+1] - ymin)/sy;
            line(xa, ya, xb, yb);
  }
 //свързване на окръжностите с отсечки и получаване на 2D графика, съответстваща на входните
//данни x2,y2

  setcolor(YELLOW);
```

```
 for(i = 0; i < n2-1; i++) {
        int xa = x0 + (x2[i] - xmin)/sx;
        int ya = y0 - (y2[i] - ymin)/sy;
        int xb = x0 + (x2[i+1] - xmin)/sx;
        int yb = y0 - (y2[i+1] - ymin)/sy;
        line(xa, ya, xb, yb);
 }
 getch();
 return 0;
}
```



Фиг.4.4. Две функции в един графичен прозорец с общи мащабни коефициенти

ТЕМА 5
Бизнес графика. Хистограми – част 1

Задача 5.1

Информацията, която визуализираме е икономическа, статистическа или социологическа. Можем да използваме представяне във вид на 2D графика, но по-подходящо е да използваме типичните за този вид информация изображения, каквито са хистограмите и кръговите диаграми.При хистограмите данните се представят като вертикални или хоризонтални стълбчета (правоъгълници или паралелепипеди или цилиндри).

Да се представят чрез вертикални хистограми данните за финансовите резултати на една фирма за последните 8 месеца.

Входните данни са двойки стойности, които могат да бъдат прочетени от файл, клавиатура , или са зададени в два масива, както в задача 3. Първият масив съдържа реалните данни, които се изобразяват като стълбчета, успоредни на скалата със стойностите, а втория масив съдържа надписите, които се изобразяват по скалата с надписи.

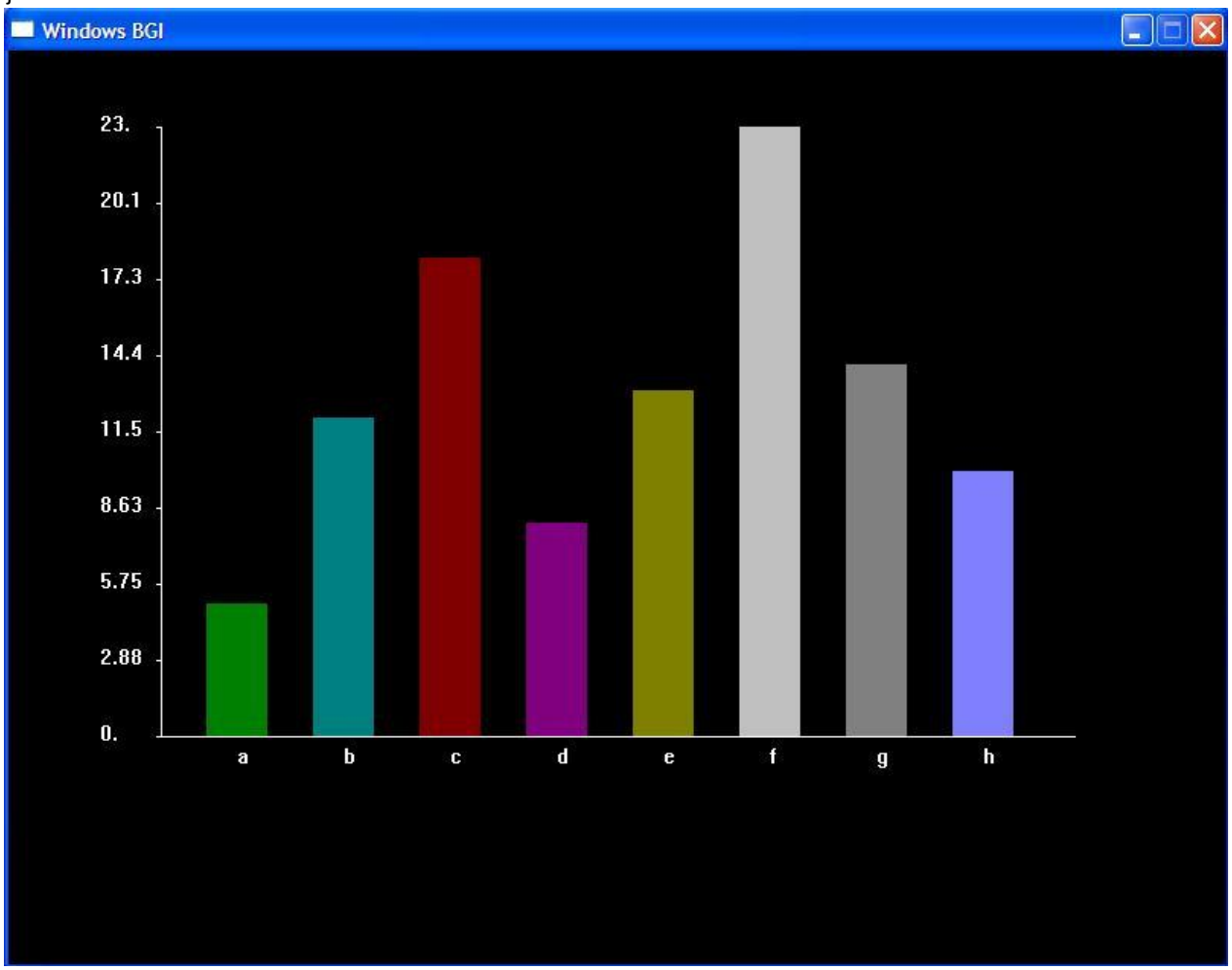
В задачата входните данни се изобразяват като вертикални стълбчета.

Визуализирани са три варианта – само с положителни данни(фиг.5.1а), с положителни и отрицателни данни(фиг.5.1в) и само с отрицателни данни(фиг.5.1с).

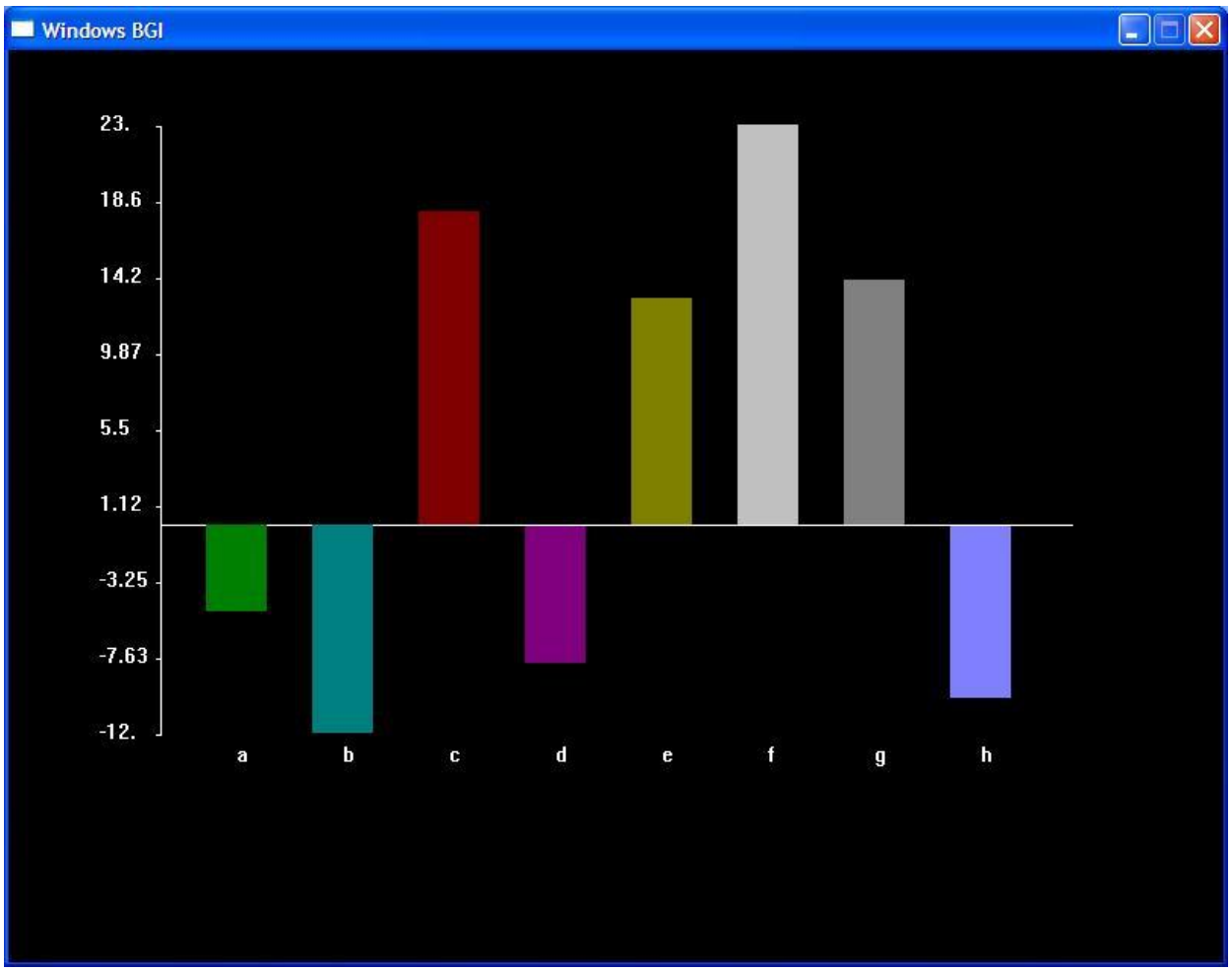
```
#include <graphics.h>
#include <iostream>
using namespace std;
int main()
{
 // реалните данни, които се изобразяват като вертикални стълбчета, се четат от файл или
//клавиатура или масив
  float a[] = {5, 12, 18, 8, 13, 23, 14, 10};
 //надписите, които ще са текст под всяко стълбче, се четат от файл или клавиатура или масив
  char labels[][20]={"a", "b", "c", "d", "e", "f", "g", "h"};
  int n=sizeof(a)/sizeof(a[0]);
  int winwidth=800,winheight=600; // параметри на прозореца на графичната система
  int Px=600,Py=400,D=50,Ds=40,Dc=30,x0=100,y0=450 ; //параметри на графичния прозорец, в който
//ще се изобразят данните (вътре в прозореца на графичната система)
  int i,x1,y1,x2,y2;

//намиране на диапазона на изменение на данните
   float amin = a[0];
   float amax = a[0];

  for(int i = 1; i< n; i++) {
        if(a[i] < amin) amin = a[i];
        if(a[i] > amax) amax = a[i];
  }
```

```
  if (amin>0) amin=0;// всички стойност са положителни, разширяваме диапазона до amin=0, за да
//може най-малкото стълбче да се види като стълбче, а не като отсечка върху скалата с надписите
  if (amax<0) amax=0;// всички стойност са отрицателни, разширяваме диапазона до amax=0, за да
//може най-малкото стълбче да се види като стълбче, а не като отсечка върху оста с надписите

//определяне на скалния коефициент за скалата със стойностите(вертикалната)
  float s = (amax - amin)/Py;

 // определяне на новото положение на хоризонталната ос,за да може да се изобразяват само
//положителни, само отрицателни или смесени данни
  int y0n=y0+amin/s;

 // инициализация на графичната система чрез отваряне на графичен прозорец със зададен размер
  initwindow(winwidth,winheight);

//изчертаване на графичния прозорец
  line(x0,y0n,x0+Px,y0n);//хоризонтална ос
  line(x0,y0,x0,y0-Py);//вертикална ос

  int I = Py/D; // брой деления по скалата със стойностите
  char text[10];

 //изчертаване и надписване на деленията по оста със стойностите
  for(i=0; i<=I; i++)
{
 line(x0, y0-D*i, x0-3, y0-D*i);//изчертаване на деленията
 gcvt(amin + i*D*s,3.2, text);//преобразуване на реалната стойност, съответстваща на  делението в
//символен низ
  settextjustify(2,1);
  outtextxy(x0-10,y0-D*i+5, text);  // извеждане на стойността, съответстваща на делението
 }

 //надписване на оста с надписите
  for(i=1; i<=n; i++)
  {
   settextjustify(1,2);
   outtextxy(x0+i*(Ds+Dc)-Ds/2,y0+5, labels[i-1]);  // извеждане на надписите, съответстващи на
//данните
       }

// изобразяване на стълбчетата, съответстващи на данните
          for(i=1; i<n+1; i++)
          {
             x1=x0+i*(Ds+Dc)-Ds;// x координата на горен ляв ъгъл
             y1=y0n-(a[i-1])/s;      //y координата на горен ляв ъгъл
             x2=x0+i*(Ds+Dc);      // x координата на долен десен ъгъл
             y2=y0n;                  // y координата на долен десен ъгъл
             setfillstyle(1, i+1);
             bar(x1,y1,x2,y2);
```

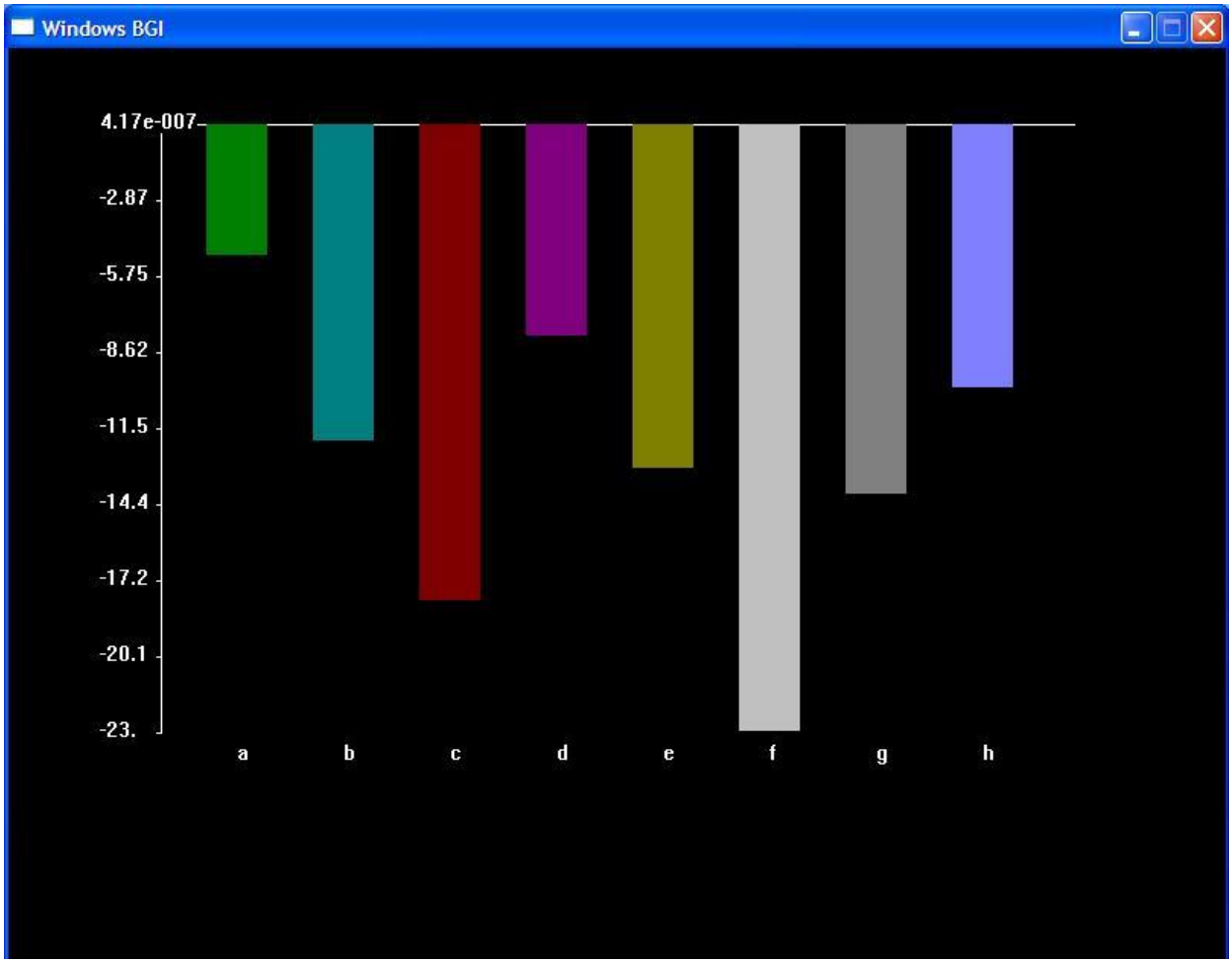
```
        }
  getch();
  return 0;
}
```



Фиг.5.1а. Само положителни входни данни float a[] = {5, 12, 18, 8, 13, 23, 14, 10};



Фиг.5.1в. Положителни и отрицателни входни данни float a[] = {-5, -12, 18, -8, 13, 23, 14, -10};



Фиг.5.1c. Само отрицателни входни данни float a[] = {-5, -12, -18, -8, -13, -23, -14, -10};

Задача 5.2

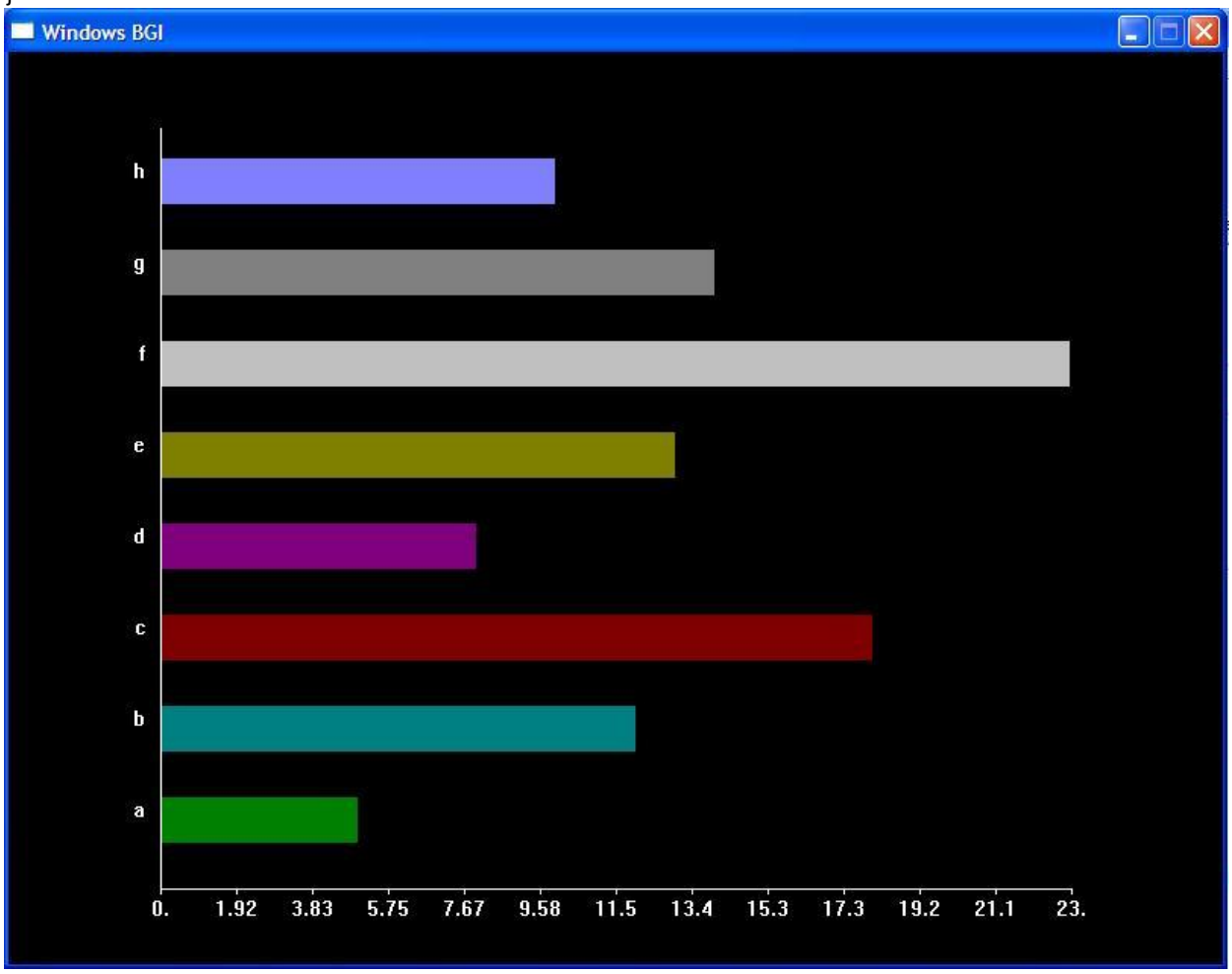
Задача за представяне на финансовите резултати на една фирма във вид на <u>хоризонтални хистограми</u>.

Разликата със задача 5.1 е, че същите входни данни се изобразяват като хоризонтални стълбчета. Визуализирани са три варианта – само с положителни данни (фиг.5.2а), с положителни и отрицателни данни (фиг.5.2в) и само с отрицателни данни (фиг.5.2с).

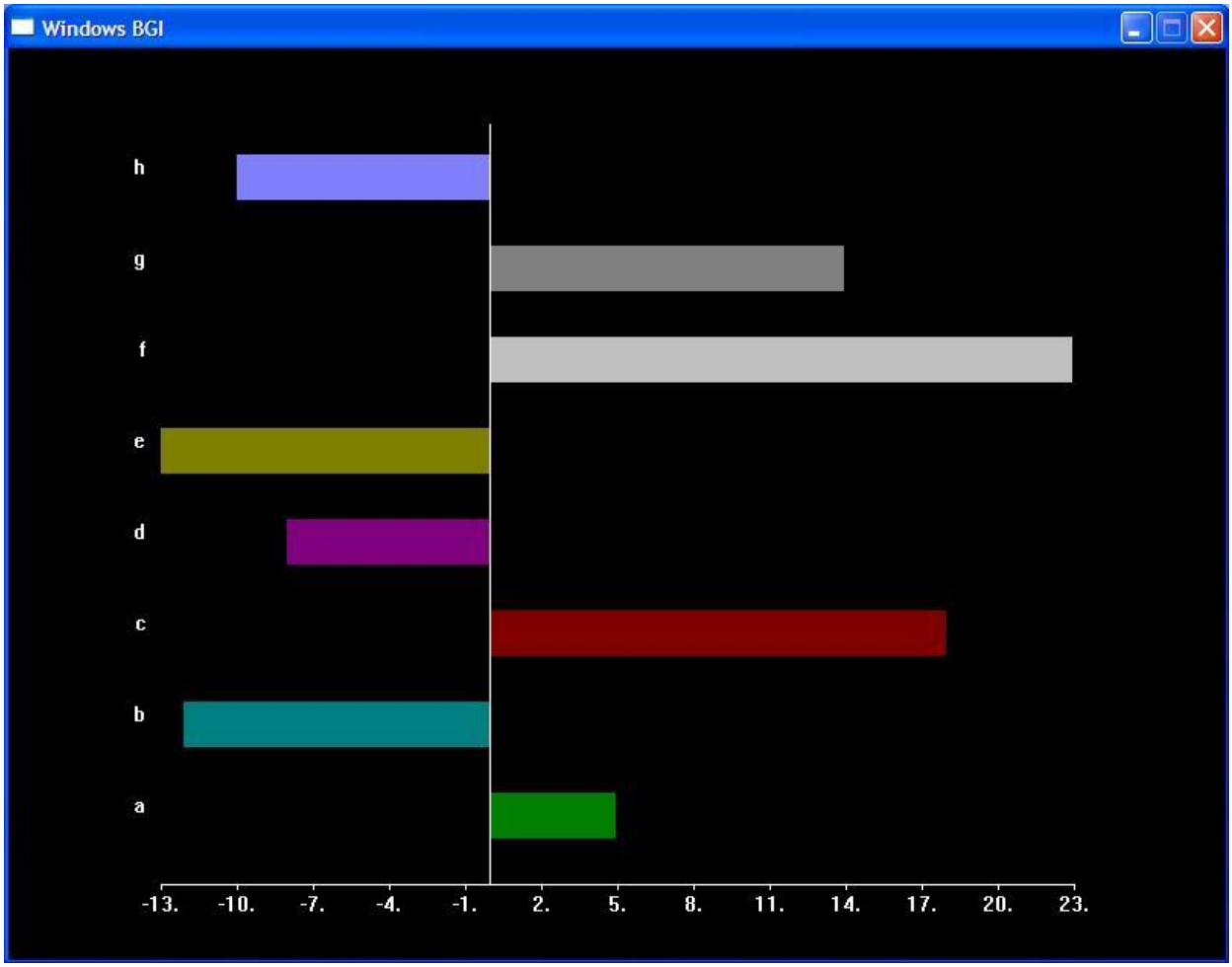
```
#include <graphics.h>
#include <iostream>
using namespace std;
int main()
{
 // реалните данни, които се изобразяват като вертикални стълбчета, се четат от файл или
//клавиатура или масив
  float a[] = {5, 12, 18, 8, 13, 23, 14, 10};
 //надписите, които ще са текст под всяко стълбче, се четат от файл или клавиатура или масив
  char labels[][20]={"a", "b", "c", "d", "e", "f", "g", "h"};
  int n=sizeof(a)/sizeof(a[0]);
```

```
  int winwidth=800,winheight=600; // параметри на прозореца на графичната система
  int Px=600,Py=500,D=50,Ds=30,Dc=30,x0=100,y0=550 ; //параметри на графичния прозорец, в който
//ще се изобразят данните (вътре в прозореца на графичната система)
  int i,x1,y1,x2,y2;

//намиране на диапазона на изменение на данните
   float amin = a[0];
   float amax = a[0];

  for(int i = 1; i< n; i++) {
          if(a[i] < amin) amin = a[i];
          if(a[i] > amax) amax = a[i];
  }
  if (amin>0) amin=0;// всички стойност са положителни, разширяваме диапазона до amin=0, за да
//може най-малкото стълбче да се види като стълбче, а не като отсечка върху оста с надписите
  if (amax<0) amax=0;// всички стойност са отрицателни, разширяваме диапазона до amax=0, за да
//може най-малкото стълбче да се види като стълбче, а не като отсечка върху оста с надписите

//определяне на скалния коефициент за скалата със стойностите (хоризонталната)
  float s = (amax - amin)/Px;

 // определяне на новото положение на хоризонталната ос,за да може да се изобразяват само
положителни, само отрицателни и смесени данни
  int x0n=x0-amin/s;

  // инициализация на графичната система чрез отваряне на графичен прозорец със зададен
//размер
  initwindow(winwidth,winheight);

//изчертаване на графичния прозорец
  line(x0,y0,x0+Px,y0);//хоризонтална ос
  line(x0n,y0,x0n,y0-Py);//вертикална ос

  int I = Px/D; // брой деления по скалата със стойностите
  char text[10];

 //изчертаване и надписване на деленията по оста със стойностите
  for(i=0; i<=I; i++)
 {
   line(x0+i*D, y0, x0+i*D, y0+3);//изчертаване на деленията
   gcvt(amin + i*D*s,3.2, text);//преобразуване на реалната стойност, съответстваща на  делението в
//символен низ
   settextjustify(1,2);
   outtextxy(x0+i*D,y0+5, text);  // извеждане на стойността, съответстваща на делението
  }

 //надписване на оста с надписите
  for(i=1; i<=n; i++)
  {
```

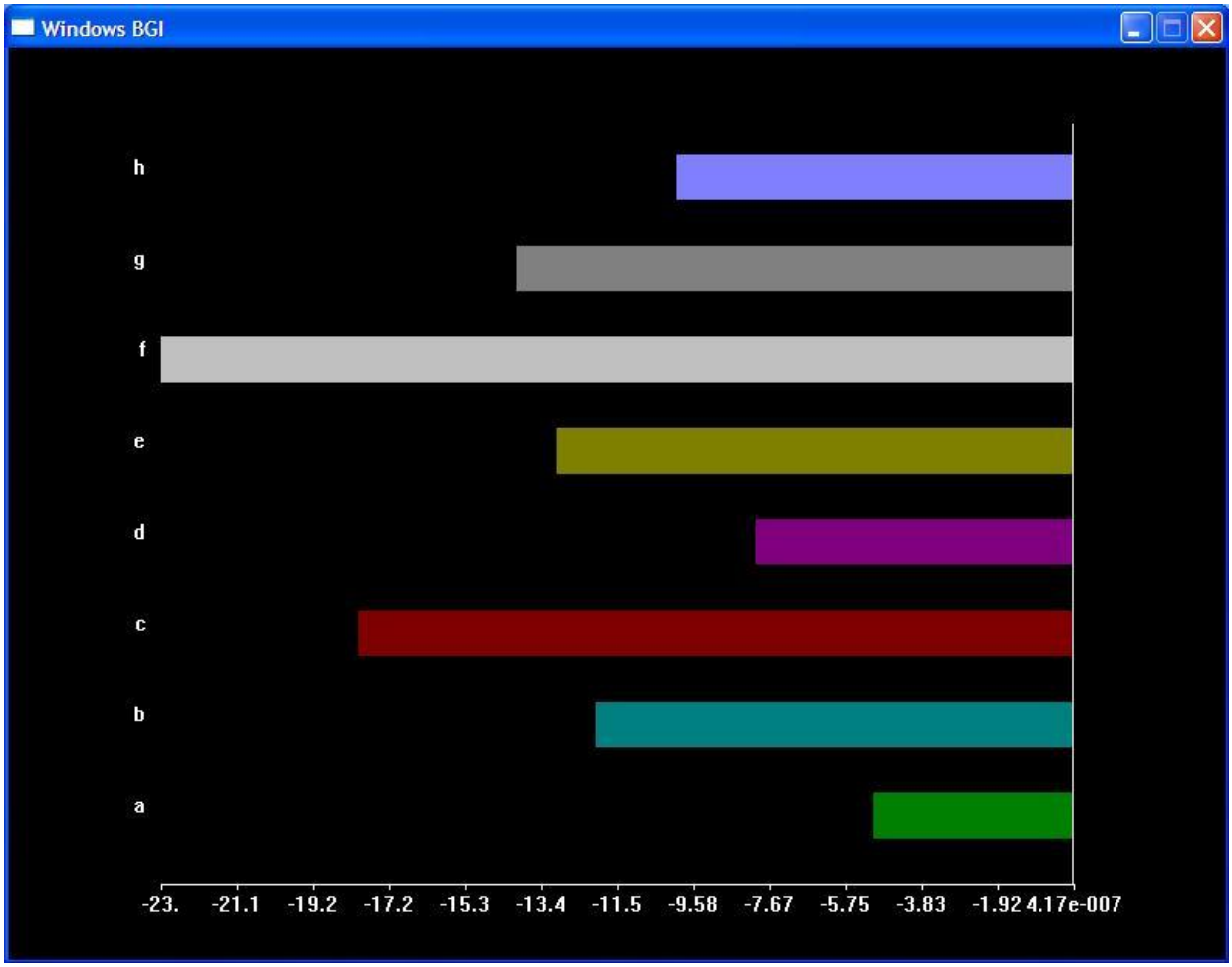
```
    settextjustify(2,2);
    outtextxy(x0-10,y0-i*(Ds+Dc),labels[i-1]);  // извеждане на надписите, съответстващи на данните
   }
// изобразяване на стълбчетата, съответстващи на данните
          for(i=1; i<n+1; i++)
          {
             x1=x0n;                  // x координата на горен ляв ъгъл
             y1=y0-i*(Ds+Dc);         //y координата на горен ляв ъгъл
             x2=x0n+a[i-1]/s;         // x координата на долен десен ъгъл
             y2=y0-i*(Ds+Dc)+Ds; // y координата на долен десен ъгъл
             setfillstyle(1, i+1);
             bar(x1,y1,x2,y2);
          }
  getch();
  return 0;
}
```



Фиг.5.2а. Само положителни входни данни float a[] = {5, 12, 18, 8, 13, 23, 14, 10};



Фиг.5.2в. Положителни и отрицателни входни данни float a[] = {-5, -12, 18, -8, 13, 23, 14, -10};



Фиг.5.2c. Само отрицателни входни данни float a[] = {-5, -12, -18, -8, -13, -23, -14, -10};

ТЕМА 6
Бизнес графика. Хистограми – част 2

Задача 6.1

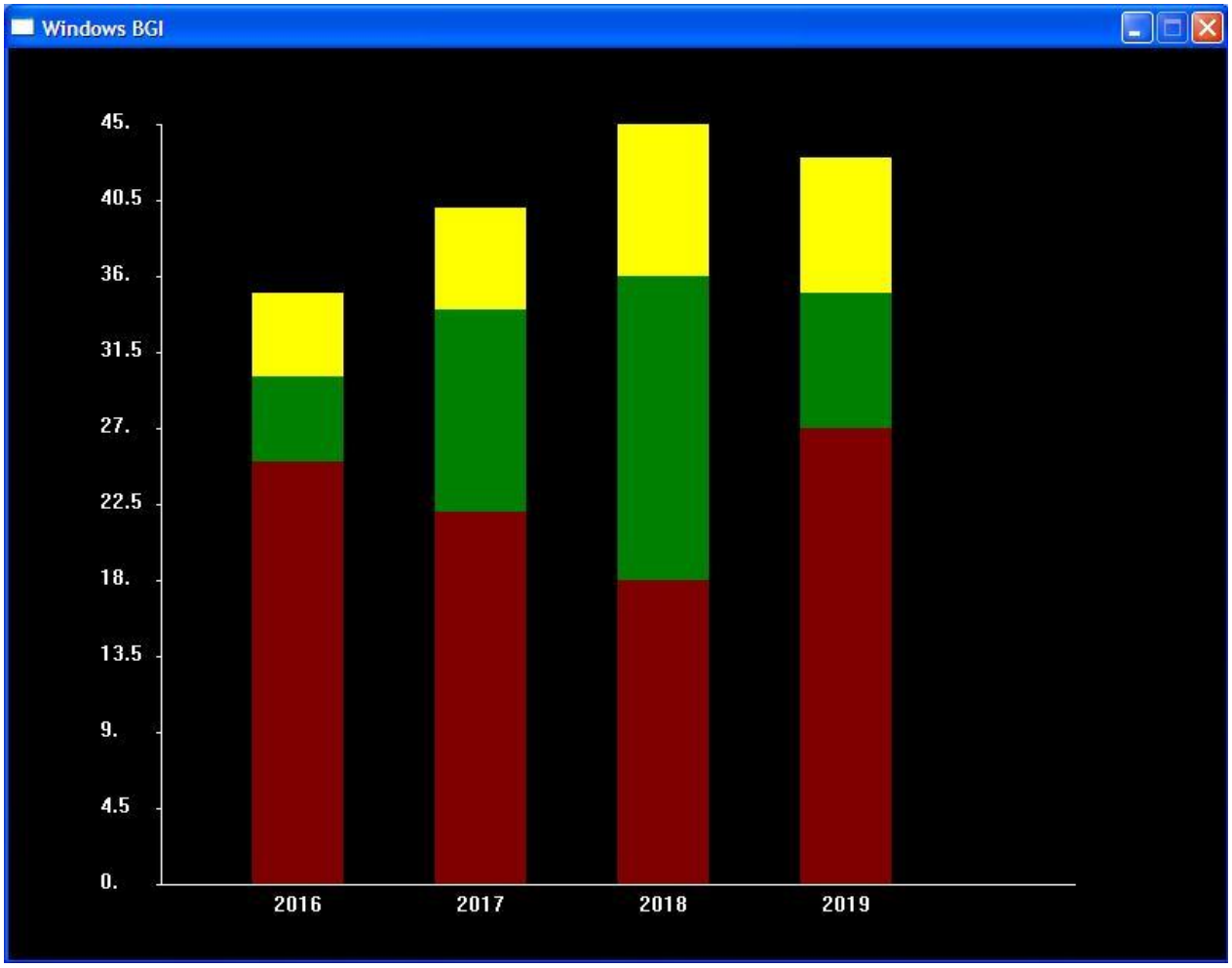
Да се представят във вид на съпоставяща вертикална хистограма (фиг.6.1) данните за приетите студенти последните 4 години в три категории – бакалаври, магистри и доктори в задаен графичен прозорец.

Всяко стълбче се образува от наслагване на три данни, които се отнасят към една категория, например прием на бакалаври, магистри и англоезично обучение в една учебна година. Другите стълбчета са аналогични данни за други учебни години. Мащабния коефициент се определя от най-големия брой студенти (бакалаври+магистри+англоезично обучение), приети през четирите наблюдавани години.

В случая данните са организирани в три едномерни масива с по 4 елемента. Могат да се представят и като 4 едномерни масива с по 3 елемента или като двумерен масив, или да се четат от файл или клавиатура.

```
#include <graphics.h>
#include <iostream>
using namespase std;
int main()
{
//съпоставяща хистограма
//пример: прием на бакалаври, магистри, аео за последните 4 учебни години
//всяко стълбче представлява приема за съответната година по трите компонента, оцветени с
//различен цвят

   float b[4] = {25, 22, 18, 27};//прием бакалаври за 4 учебни години -стълбчетата с червен цвят на
//фиг.6.1
   float m[4] = {5, 12, 18, 8};  //прием магистри за 4 учебни години - стълбчетата със зелен цвят на
//фиг.6.1
   float a[4] = {5, 6,9, 8};       //прием АЕО за 4 учебни години - стълбчетата с жълт цвят на фиг.6.1
//надписите, които ще са текст под всяко стълбче, се четат от файл или клавиатура или масив
  char labels[][20]={"2016", "2017", "2018", "2019"};
  int winwidth=800,winheight=600; // параметри на прозореца на графичната система
  int Px=600,Py=500,D=50,Ds=60,Dc=60,x0=100,y0=550 ; //параметри на графичния прозорец, в който
//ще се изобразят данните (вътре в прозореца на графичната система)
  int n=4,i;
//скалният коефициент се определя от максималния прием по трите пера за година
float s[n] ;
for(int i = 1; i< n; i++) {
        s[i] =b[i]+m[i]+a[i];
  }
float smax=0;
  for(int i = 1; i< n; i++) {
        if(s[i] > smax) smax = s[i];
  }
```

```
  float ss = smax/Py;   //скален коефициент

 // инициализация на графичната система чрез отваряне на графичен прозорец със зададен размер
  initwindow(winwidth,winheight);

//изчертаване на графичния прозорец
  line(x0,y0,x0+Px,y0);//хоризонтзлнз ос
  line(x0,y0,x0,y0-Py);//вертикална ос

  int I = Py/D; // брой деления по скалата със стойностите
  char text[10];

 //изчертаване и надписване на деленията по оста със стойностите
  for(i=0; i<=I; i++)
 {
 line(x0, y0-D*i, x0-3, y0-D*i);//изчертаване на деленията
 gcvt(i*D*ss,3.2, text);//преобразуване на реалната стойност, съответстваща на  делението в
//символен низ
   settextjustify(2,1);
  outtextxy(x0-10,y0-D*i+5, text);  // извеждане на стойността, съответстваща на делението
  }
 //надписване на оста с надписите
  for(i=1; i<=n; i++)
  {
      settextjustify(1,2);
      outtextxy(x0+i*(Ds+Dc)-Ds/2,y0+5,labels[i-1]);  // извеждане на надписите, съответстващи на
//данните
  }
// изобразяване на стълбчетата, съответстващи на данните

          for(i=1; i<n+1; i++)
          {  setfillstyle(1, RED);
                   bar(x0+i*(Ds+Dc)-Ds, y0-(b[i-1])/ss, x0+i*(Ds+Dc), y0);
             setfillstyle(1, GREEN);
                   bar(x0+i*(Ds+Dc)-Ds, y0-(b[i-1]+m[i-1])/ss, x0+i*(Ds+Dc), y0-(b[i-1])/ss);
             setfillstyle(1, YELLOW);
                   bar(x0+i*(Ds+Dc)-Ds, y0-(b[i-1]+m[i-1]+a[i-1])/ss, x0+i*(Ds+Dc), y0-(b[i-1]+m[i-1])/ss);
          }

  getch();
  return 0;
}
```



фиг.6.1 Съпоставяща хистограма с вертикални стълбчета

Задача 6.2

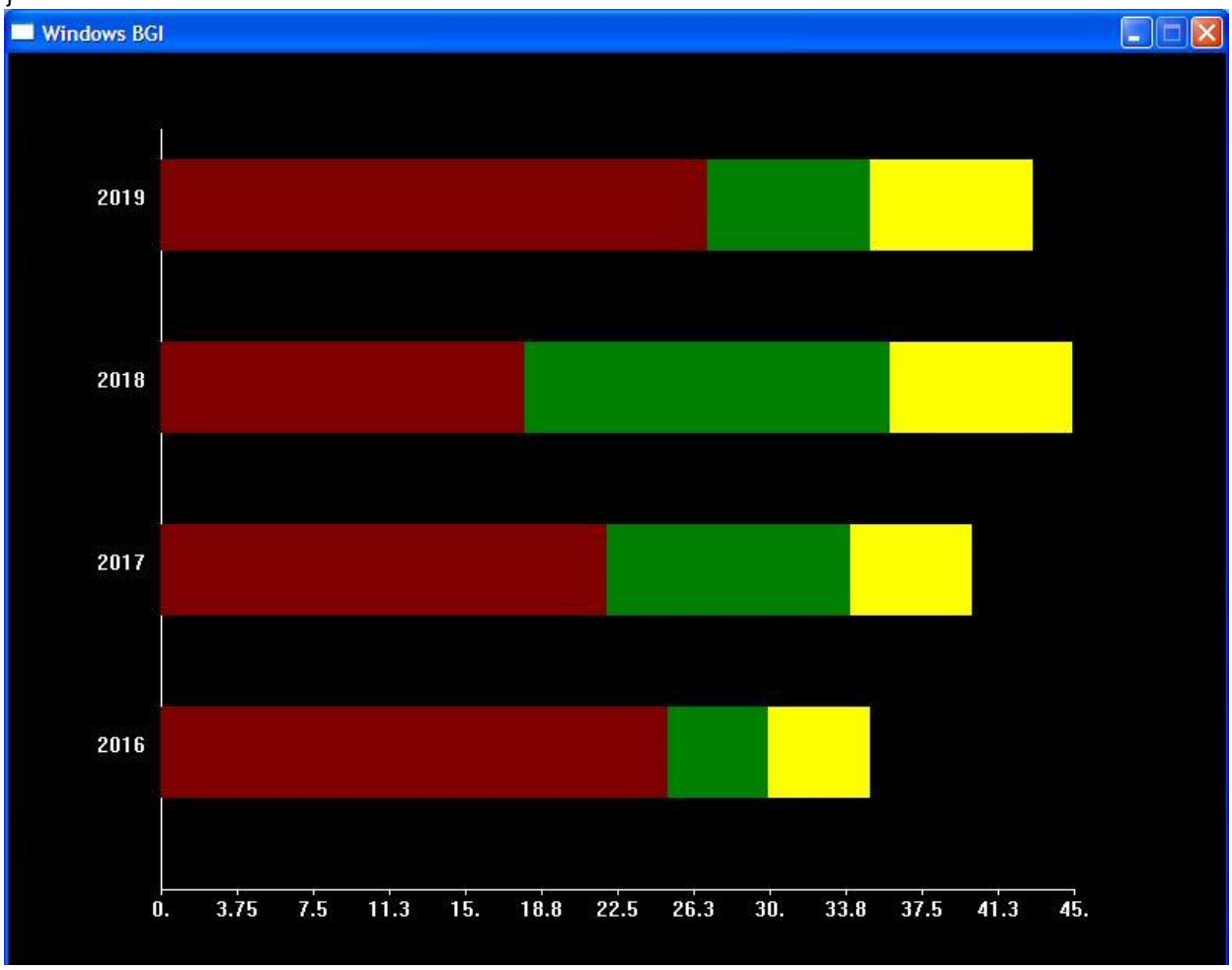
Да се представят във вид на съпоставяща <u>хоризонтална</u> хистограма (фиг.6.2) данните за приетите студенти последните 4 години в три категории – бакалаври, магистри и доктори в задаен графичен прозорец.

Условието на задачата е като в задача 6.1, но визуализираните стълбчета са хоризонтални.

```
#include <graphics.h>
#include <iostream>
using namespase std;

int main()
{
//съпоставяща хистограма
//пример: прием на бакалаври,магистри,аео за последните 4 години
//всяко стълбче представлява приема за съответната година по трите компонента, оцветени с
//различен цвят

   float b[4] = {25, 22, 18, 27}; //прием бакалаври за 4 гогини
```

```
   float m[4] = {5, 12, 18, 8}; //прием магистри за 4 години
   float a[4] = {5, 6,9, 8}; //прием АЕО за 4 гогини
//надписите, които ще са текст под всяко стълбче, се четат от файл или клавиатура или масив
  char labels[][20]={"2016", "2017", "2018", "2019"};
  int winwidth=800,winheight=600; // параметри на прозореца на графичната система
  int Px=600,Py=500,D=50,Ds=60,Dc=60,x0=100,y0=550 ; //параметри на графичния прозорец, в който
//ще се изобразят данните (вътре в прозореца на графичната система)
  int n=4,i;
//скалният коефициент се определя от максималния прием по трите пера за година
float s[n] ;
for(int i = 1; i< n; i++) {
          s[i] =b[i]+m[i]+a[i];
  }
float smax=0;
  for(int i = 1; i< n; i++) {
          if(s[i] > smax) smax = s[i];
  }

  float ss = smax/Px;  //скален коефициент

 // инициализация на графичната система чрез отваряне на графичен прозорец със зададен размер
  initwindow(winwidth,winheight);

//изчертаване на графичния прозорец
  line(x0,y0,x0+Px,y0);//хоризонтална ос
  line(x0,y0,x0,y0-Py);//вертикална ос


  int I = Px/D;   // брой деления по скалата със стойностите
  char text[10];

 //изчертаване и надписване на деленията по оста със стойностите
  for(i=0; i<=I; i++)
{
  line(x0+i*D, y0, x0+i*D, y0+3);//изчертаване на деленията
  gcvt(i*D*ss,3.2, text);//преобразуване на реалната стойност, съответстваща на //делението в
//символен низ
  settextjustify(1,2);
   outtextxy(x0+i*D,y0+5, text);  // извеждане на стойността, съответстваща на делението
  }

 //надписване на оста с надписите
  for(i=1; i<=n; i++)
 {
   settextjustify(2,1);
   outtextxy(x0-10,y0-i*(Ds+Dc)+Ds/2,labels[i-1]);  // извеждане на надписите, съответстващи на
//данните
  }
```

```
// изобразяване на стълбчетата, съответстващи на данните
        for(i=1; i<n+1; i++)
        {  setfillstyle(1, RED);
              bar(x0, y0-i*(Ds+Dc), x0+(b[i-1])/ss, y0-i*(Ds+Dc)+Ds);
           setfillstyle(1, GREEN);
                    bar(x0+(b[i-1])/ss, y0-i*(Ds+Dc), x0+(b[i-1]+m[i-1])/ss, y0-i*(Ds+Dc)+Ds);
           setfillstyle(1, YELLOW);
              bar(x0+(b[i-1]+m[i-1])/ss, y0-i*(Ds+Dc), x0+(b[i-1]+m[i-1]+a[i-1])/ss, y0-i*(Ds+Dc)+Ds);
        }
  getch();
  return 0;
}
```



фиг.6.2 Съпоставяща хистограма с хоризонтални стълбчета

Задача 6.3

Да се представят във вид на съпоставяща <u>вертикална</u> хистограма (фиг.6.3) данните за приориходи и разходи по 8 пера на една фирма в задаен графичен прозорец.

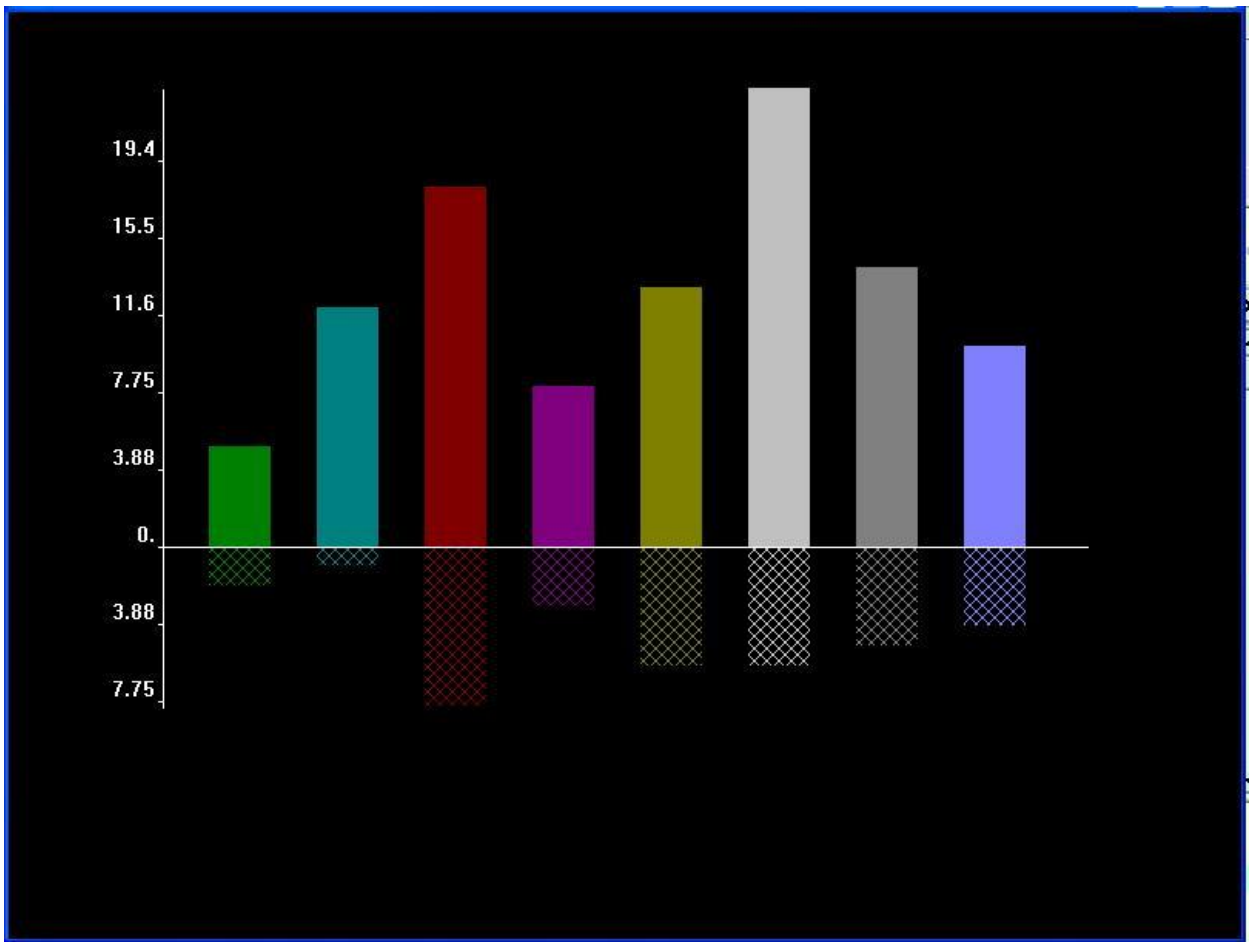
Хистограма приход/разход с вертикални стълбчета – фиг.6.3

Данните са организирани в два масива, съответно по един за приходите и за разходите –положителни реални стойности. Скалния коефициент се определя от сумата от максималния приход и максималния разход. Хоризонталната ос се премества на ново положение, съответстващо на максималния разход, преобразуван в пиксели. Всеки приход се

изобразява като вертикално стълбче нагоре от новото положение на хоризонталната ос, а съответстващия му разход – надолу от оста. Стойностите по деленията по скалата със стойностите започват от новото положение на хоризонталната ос – на пиксел с координати (x0,y0n) съответства стойност нула. В долния край на вертикалната ос. (координати x0,y0) съответства стойността на максималния разход, а в горния край (координати x0,y0-Py) съответства стойността на максималния приход.

```
#include <graphics.h>
#include <iostream>
using namespace std;

int main()
{
 //хистограма приход-разход
 // реалните данни, които се изобразяват като вертикални стълбчета, се четат от файл или
//клавиатура или масив
  float prihodi[] = {5, 12, 18, 8, 13, 23, 14, 10}; // масив с приходи
  float razhodi[] = {2, 1, 8, 3, 6, 6, 5, 4};  // масив с разходи
  int n=sizeof(prihodi)/sizeof(prihodi[0]);
  int winwidth=800,winheight=600; // параметри на прозореца на графичната система
  int Px=600,Py=400,D=50,Ds=40,Dc=30,x0=100,y0=450 ; //параметри на графичния прозорец, в който
//ще се изобразят данните (вътре в прозореца на графичната система)
  int i,x1,y1,x2,y2;

//намиране на диапазона на изменение на данните
   float prihodmax = prihodi[0];
   float razhodmax = razhodi[0];

  for(int i = 1; i< n; i++) {
          if(prihodi[i] > prihodmax) prihodmax= prihodi[i];
          if(razhodi[i] > razhodmax) razhodmax= razhodi[i];
  }

//определяне на скалния коефициент за скалата със стойностите (вертикалната)
  float s = (prihodmax +razhodmax)/Py;

 // определяне на новото положение на хоризонталната ос,съответстващо на максималния разход
  int y0n=y0-razhodmax/s;

 // инициализация на графичната система чрез отваряне на графичен прозорец със зададен
//размер
  initwindow(winwidth,winheight);

//изчертаване на графичния прозорец
  line(x0,y0n,x0+Px,y0n);//хоризонтална ос на новото положение (преместена със сттойността на
//макс.разход
  line(x0,y0,x0,y0-Py);//вертикална ос

  int h = prihodmax/s; //частта от вертикалната ос, върху която се разполагат приходите
```

```
  int I = h/D;  // брой деления по тази част от скалата

  char text[10];
//изчертаване и надписване на деленията по оста на приходите
  for(i=0; i<=I; i++)
 {
  line(x0, y0n-D*i, x0-3, y0n-D*i);//изчертаване на деленията
  gcvt(i*D*s,3.2, text);//преобразуване на реалната стойност, съответстваща на //делението в
символен низ
  settextjustify(2,0);
  outtextxy(x0-5,y0n-D*i, text);  // извеждане на стойността, съответстваща на делението
   }
  int h1 = razhodmax/s;//частта от вертикалната ос, върху която се разполагат разходите
  int J = h1/D;  // брой деления по тази част от скалата

//изчертаване и надписване на деленията по оста на разходите

  for(i=0; i<=J; i++)
   {
   line(x0, y0n+D*i, x0-3, y0n+D*i);//изчертаване на деленията
   gcvt(i*D*s,3.2, text);//преобразуване на реалната стойност, съответстваща на //делението в
//символен низ
   settextjustify(2,0);
   outtextxy(x0-5,y0n+D*i, text);  // извеждане на стойността, съответстваща на делението
   }

// изобразяване на стълбчетата, съответстващи на приходите
          for(i=1; i<n+1; i++)
          {
 // изобразяване на стълбчетата, съответстващи на приходите
             x1=x0+i*(Ds+Dc)-Ds;        // x координата на горен ляв ъгъл
             y1=y0n-(prihodi[i-1])/s;   // y координата на горен ляв ъгъл
             x2=x0+i*(Ds+Dc);           //x координата на долен ляв ъгъл
             y2=y0n;                    // y координата на долен десен ъгъл
             setfillstyle(1, i+1);
             bar(x1,y1,x2,y2);
// изобразяване на стълбчетата, съответстващи на разходите

          x1=x0+i*(Ds+Dc)-Ds;          //x координата на горен ляв ъгъл
          y1=y0n;                      // y координата на горен ляв ъгъл
           x2=x0+i*(Ds+Dc);            //x координата на долен десен ъгъл
           y1=y0n+(razhodi[i-1])/s;    // y координата на долен десен ъгъл
           setfillstyle(8, i+1);
          bar(x1,y1,x2,y2);
   }
  getch();
  return 0;
}
```



фиг.6.3 Хистограма приход/разходс вертикални стълбчета

Задача 6.4

Да се представят във вид на съпоставяща <u>хоризонтална</u> хистограма (фиг.6.3) данните за приориходи и разходи по 8 пера на една фирма в задаен графичен прозорец.

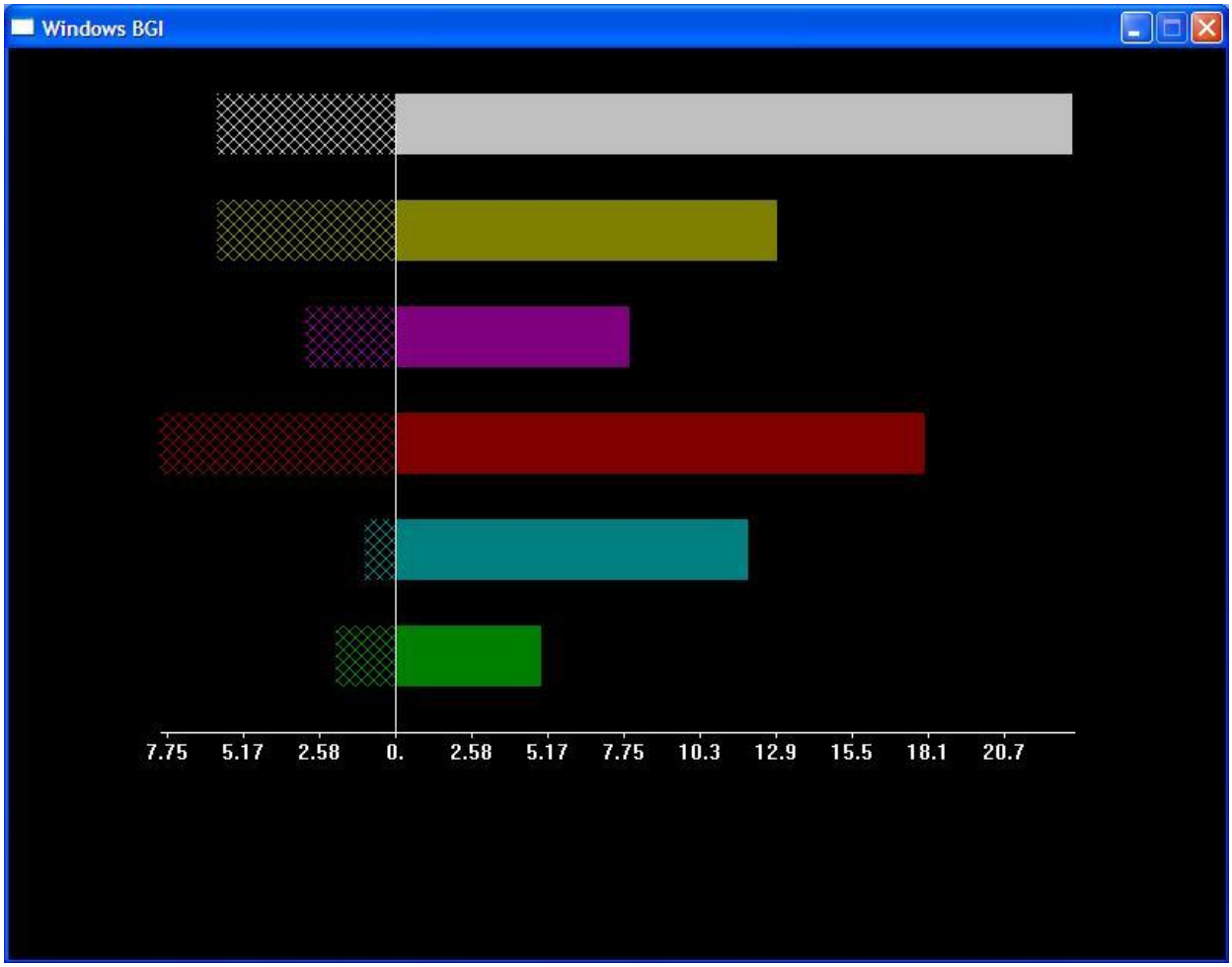
Хистограма приход/разход с хоризонтални стълбчета – фиг.6.4

Данните са организирани в два масива, съответно по един за приходите и за разходите –положителни реални стойности. Скалния коефициент се определя от сумата от максималния приход и максималния разход. Вертикалната ос се премества на ново положение, съответстващо на максималния разход, преобразуван в пиксели. Всеки приход се изобразяват като хоризонтално стълбче надясно от новото положение на вертикалната ос, а съответстващия му разход – наляво от оста.Стойностите по деленията по скалата със стойностите започват от новото положение на вертикалната ос – на пиксел с координати (x0n,y0) съответства стойност нула. В левия край на хоризонталната ос (координати x0,y0) съответства стойността на максималния разход, а в десния край (координати x0+Px,y0) съответства стойността на максималния приход.

```
#include <graphics.h>
#include <iostream>
using namespace std;
```

```
int main()
{
 //хистограма приход-разход
 // реалните данни, които се изобразяват като вертикални стълбчета, се четат от файл или
//клавиатура или масив
  float prihodi[] = {5, 12, 18, 8, 13, 23, 14, 10};
  float razhodi[] = {2, 1, 8, 3, 6, 6, 5, 4};
  int n=sizeof(prihodi)/sizeof(prihodi[0]);
  int winwidth=800,winheight=600; // параметри на прозореца на графичната система
  int Px=600,Py=400,D=50,Ds=40,Dc=30,x0=100,y0=450 ; //параметри на графичния прозорец, в който
//ще се изобразят данните (вътре в прозореца на графичната система)
  int i,x1,y1,x2,y2;

//намиране на диапазона на изменение на данните
   float prihodmax = prihodi[0];
   float razhodmax = razhodi[0];

  for(int i = 1; i< n; i++) {
          if(prihodi[i] > prihodmax) prihodmax= prihodi[i];
          if(razhodi[i] > razhodmax) razhodmax= razhodi[i];
  }

//определяне на скалния коефициент за скалата със стойностите(хоризонталната)
  float s = (prihodmax +razhodmax)/Px;

 // определяне на новото положение на вертикалната ос,съответстващо на максималния разход
  int x0n=x0+razhodmax/s;

  // инициализация на графичната система чрез отваряне на графичен прозорец със зададен
размер
  initwindow(winwidth,winheight);

//изчертаване на графичния прозорец
  line(x0,y0,x0+Px,y0);//хоризонтална ос
  line(x0n,y0,x0n,y0-Py);//вертикална ос , изместена със стойността на макс. Разход, преобразуван в
//пиксели

  int h = prihodmax/s;//частта от хоризонталната ос, върху която се разполагат приходите
  int I = h/D;  // брой деления по тази част от скалата

  char text[10];
//изчертаване и надписване на деленията по оста на приходите
  for(i=0; i<=I; i++)
 {
   line(x0n+i*D, y0, x0n+i*D, y0+3);//изчертаване на деленията
   gcvt(i*D*s,3.2, text);//преобразуване на реалната стойност, съответстваща на делението в
//символен низ
    settextjustify(1,2);
    outtextxy(x0n+i*D,y0+5, text);  // извеждане на стойността, съответстваща на делението
```

```
  }

  int h1 = razhodmax/s;        //частта от вертикалната ос, върху която се разполагат разходите
  int J = h1/D;       // брой деления по тази част от скалата

//изчертаване и надписване на деленията по оста на разходите
    for(i=0; i<=J; i++)
 {
    line(x0n-i*D, y0, x0n-i*D, y0+3);//изчертаване на деленията
    gcvt(i*D*s,3.2, text);//преобразуване на реалната стойност, съответстваща на  делението в
//символен низ
     settextjustify(1,2);
     outtextxy(x0n-i*D,y0+5, text);  // извеждане на стойността, съответстваща на делението
  }

// изобразяване на стълбчетата, съответстващи на приходите
          for(i=1; i<n+1; i++)
          {
 // изобразяване на стълбчетата, съответстващи на приходите
              x1=x0n;                          //x координата на горен ляв ъгъл
              y1=y0-i*(Ds+Dc);                 // y координата на горен ляв ъгъл
              x2=x0n+(prihodi[i-1])/s;     //x координата на долен ляв ъгъл
              y2=y0-i*(Ds+Dc)+Ds;          // y координата на долен десен ъгъл
              setfillstyle(1, i+1);
              bar(x1,y1,x2,y2);
// изобразяване на стълбчетата, съответстващи на разходите
             x1=x0n-(razhodi[i-1])/s;       //x координата на горен ляв ъгъл
              y1=y0-i*(Ds+Dc);                 // y координата на горен ляв ъгъл
              x2=x0n;                          //x координата на долен ляв ъгъл
              y2=y0-i*(Ds+Dc)+Ds;          // y координата на долен десен ъгъл
              setfillstyle(8, i+1);
              bar(x1,y1,x2,y2);
    }
  getch();
  return 0;
}
```



фиг.6.4 Хистограма приход/разход с хоризонтални стълбчета

ТЕМА 7
Бизнес графика. Кръгови диаграми

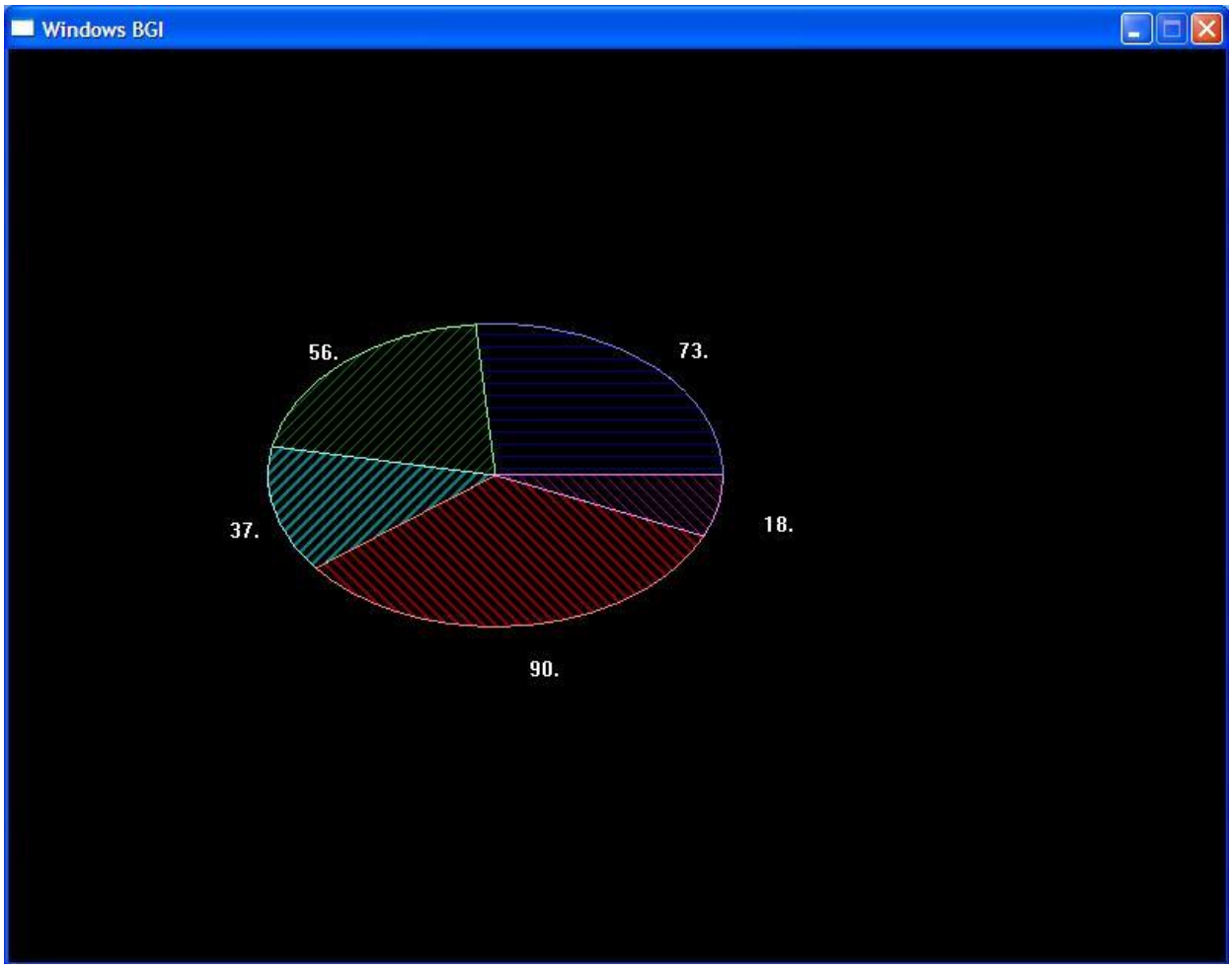
Задача 7.1

Кръговите диаграми представят данните като части от едно цяло, което може да бъде окръжност или елипса или цилиндър. Подходящи са при представяне на процентни съотношения, например резултати от избори

Входните данни са реални стойности, които могат да бъдат прочетени от файл, клавиатура , или са зададени в масив, както в задачата. Всяка реална стойност трябва да се преобразува в ъгъл на парчето от кръга или елипсата. Надписите са изнесени по ъглополовящата на всеки сектор извън елипсата- фиг.7.1, а може да се изведат и в легенда.

```
#include <math.h>
#include <graphics.h>
#include <iostream>
using namespace std;
int main()
{
//кръгова диаграма
  double a[]={73,56,37,90,18}, A=0;
  int n=sizeof(a)/sizeof(a[0]);
  int winwidth=800,winheight=600; // параметри на прозореца на графичната система
  int x0=320, y0=280, rx=150, ry=100;//координати на центъра и радиуси на елипсата
  int xi, yi,i,j;
  double na4, krai, gama;
  char text[10];

//намиране на сумата от данните
// ъгълът, съответстващ на данните се определя като gama =a[i]/A*360- в градуси или
//gama =a[i]/A*2* M_PI – в радиани
  for (i=0; i<n ; i++)
  A+=a[i];

// инициализация на графичната система чрез отваряне на графичен прозорец със зададен размер
  initwindow(winwidth,winheight);

   na4=0;         //начален ъгъл в градуси за първия сектор от елипсата

//изобразяванв на кръговата диаграма като елипса
for(i=0;i<n;i++)
{
  gama =a[i]/A*360;    //определяне на ъгъла на сектора от елипсата,съответстващ на данните
  krai = na4+gama;      //определяне на края на сектора в градуси
  setcolor(i+9);
  setfillstyle(2+i,i+1);
  sector(x0,y0,na4,krai,rx,ry);      //изчертаване на текущия сектор от елипсата
//определяне на мястото на надписа, изнесен извън сектора по ъглополовящата на текущия ъгъл
  xi=x0+1.2*rx*cos((na4+gama/2.0)/360.0*2*M_PI);
   yi=y0-1.2*ry*sin((na4+gama/2.0)/360.0*2*M_PI);
   setcolor(WHITE);
```

```
    gcvt(a[i],5, text);
    outtextxy(xi, yi, text);      //извеждане на стойността
    na4=krai;//начален ъгъл за следващия сектор
}
  getch();
  return 0;
}
```



Фиг.7.1 Кръгова диаграма

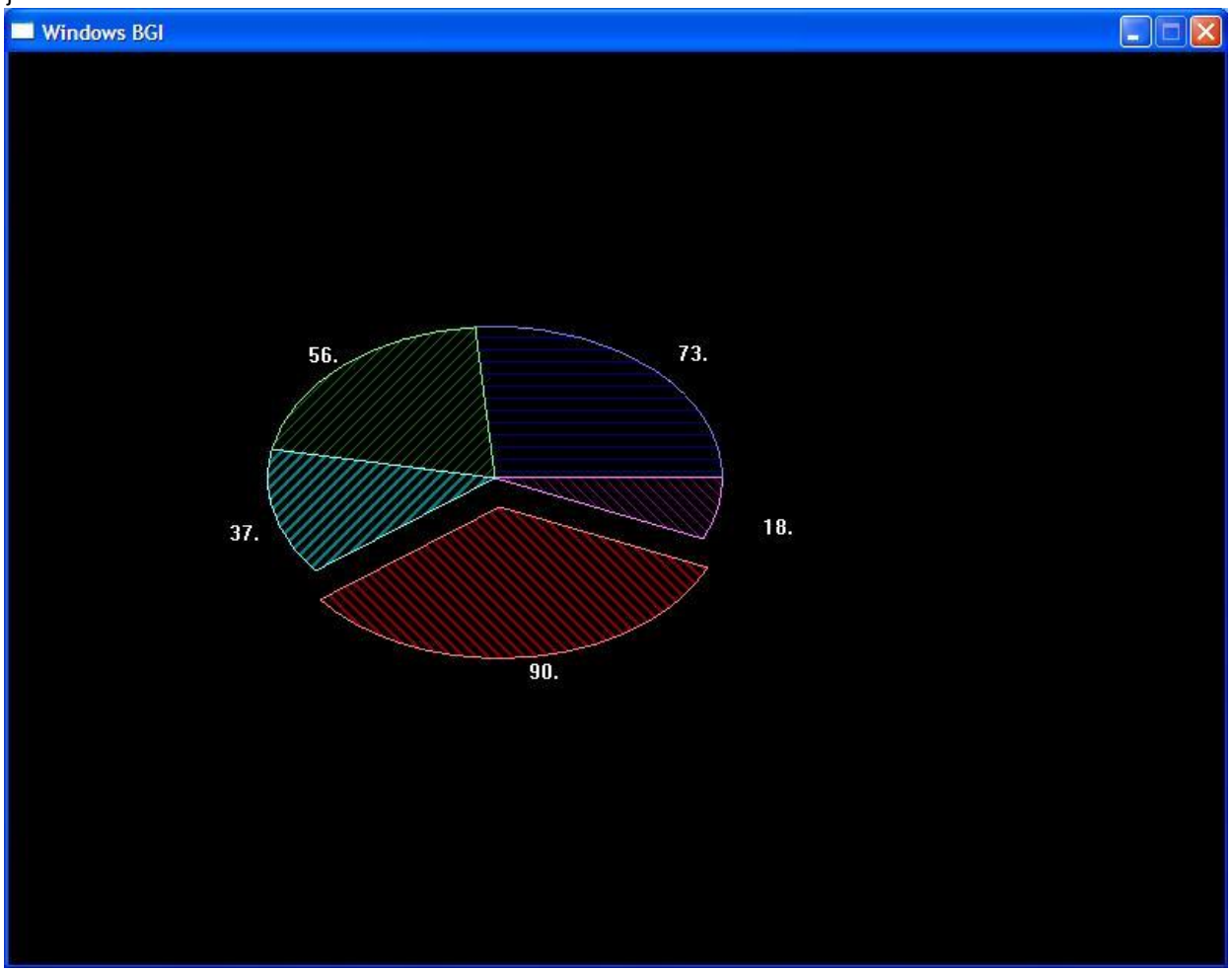
Задача 7.2

Модифициран вариант на задача 7.1. Намерен е най-големия сектор и той е изнесен по ъглополовящата с цел акцентиране върху него(фиг.7.2). Промените спрямо задача 7.1 са отбелязани в червен цвят.

```
#include <math.h>
#include <graphics.h>
#include <iostream>
using namespace std;
int main()
{
//кръгова диаграма
```

```
  double a[]={73,56,37,90,18}, A=0,max;
  int n=sizeof(a)/sizeof(a[0]);
  int winwidth=800,winheight=600; // параметри на прозореца на графичната система
  int x0=320, y0=280, rx=150, ry=100;//координати на центъра и радиуси на елипсата
  int xi, yi,i,j;
  double na4, krai, gama;
  char text[10];

//намиране на сумата от данните
  for (i=0; i<n ; i++)
  A+=a[i];

//намиране на максималния елемент
  max=a[0];
  for(i=0;i<n;i++)
  if(a[i]>max) max=a[i];

// инициализация на графичната система чрез отваряне на графичен прозорец със зададен размер
  initwindow(winwidth,winheight);

   na4=0;    //начален ъгъл в градуси за първия сектор от елипсата
   for(i=0;i<n;i++)
   {
    gama =a[i]/A*360;//определяне на ъгъла на сектора от елипсата,съответстващ на данните
    krai = na4+gama; //определяне на края на сектора в градуси
    setcolor(i+9);
    setfillstyle(2+i,i+1);

//изнасяне на максималния елемент извън центъра на елипсата по ъглополовящата с малък
//радиус
                xi=x0+0.2*rx*cos((na4+gama/2.0)/360.0*2*M_PI);
                yi=y0-0.2*ry*sin((na4+gama/2.0)/360.0*2*M_PI);
                if(a[i]==max)
                sector(xi,yi,na4,krai,rx,ry);//изчертаване на най-големия сектор,изнесен по
//ъглополовящата
                else
                sector(x0,y0,na4,krai,rx,ry);//изчертаване на текущия сектор от елипсата

                //определяне на мястото на надписа, изнесен извън сектора по ъглополовящата на
//текущия ъгъл
      xi=x0+1.2*rx*cos((na4+gama/2.0)/360.0*2*M_PI);
      yi=y0-1.2*ry*sin((na4+gama/2.0)/360.0*2*M_PI);
      setcolor(WHITE);
      gcvt(a[i],5, text);
      outtextxy(xi, yi, text);    //извеждане на стойността
      na4=krai;        //начален ъгъл за следващия сектор
           }
  getch();
```

```
  return 0;
}
```



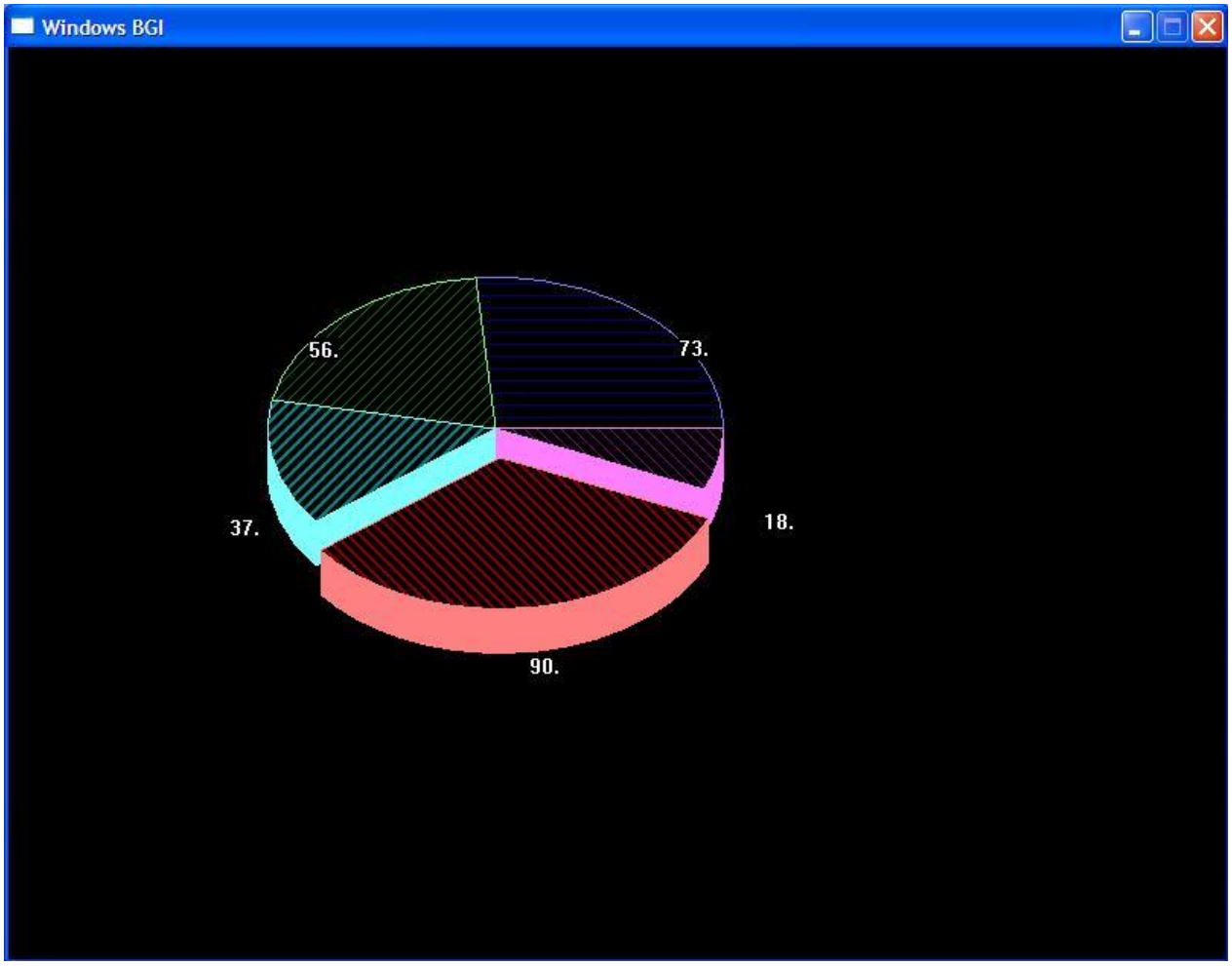
Фиг.7.2 Кръгова диаграма с форма на елипса с акцент върху най-големия сектор

Задача 7.3

Модифициран вариант на задача 7.2. Кръговата диаграма е показана като цилиндър (фиг.7.3). За целта тя се изчертава 30 пъти с намаляване на y координатата на центъра с по един пиксел. Илюзията за триизмерност се постига с използване на основни и допълнителни цветове или с цвят и щриховка. Промените спрямо задача 7.2 са отбелязани в червен цвят.

```
#include <math.h>
#include <graphics.h>
#include <iostream>
using namespace std;
int main()
{
//кръгова диаграма

  double a[]={73,56,37,90,18}, A=0,max;
  int n=sizeof(a)/sizeof(a[0]);
  int winwidth=800,winheight=600; // параметри на прозореца на графичната система
```

```
  int x0=320, y0=280, rx=150, ry=100;//координати на центъра и радиуси на елипсата
  int xi, yi,i,j;
  double na4, krai, gama;
  char text[10];

//намиране на сумата от данните
  for (i=0; i<n ; i++)
  A+=a[i];

//намиране на максималния елемент
  max=a[0];
  for(i=0;i<n;i++)
  if(a[i]>max) max=a[i];
 // инициализация на графичната система чрез отваряне на графичен прозорец със зададен размер
  initwindow(winwidth,winheight);
 //изчертаване като цилиндър
  for (j=0; j<30; j++)
  {
   na4=0;//начален ъгъл в градуси за първия сектор от елипсата
   for(i=0;i<n;i++)
   {
    gama =a[i]/A*360;//определяне на ъгъла на сектора от елипсата,съответстващ на данните
    krai = na4+gama; //определяне на края на сектора в градуси
    setcolor(i+9);
    setfillstyle(2+i,i+1);

//изнасяне на максималния елемент извън центъра на елипсата по ъглополовящата с малък
//радиус
                xi=x0+0.2*rx*cos((na4+gama/2.0)/360.0*2*M_PI);
                yi=y0-0.2*ry*sin((na4+gama/2.0)/360.0*2*M_PI);
                if(a[i]==max)
                sector(xi,yi-j,na4,krai,rx,ry);//изчертаване на най-големия сектор, изнесен по
//ъглополовящата
                else
                sector(x0,y0-j,na4,krai,rx,ry);//изчертаване на текущия сектор от елипсата

                //определяне на мястото на надписа, изнесен извън сектора по ъглополовящата на
//текущия ъгъл
      xi=x0+1.2*rx*cos((na4+gama/2.0)/360.0*2*M_PI);
      yi=y0-1.2*ry*sin((na4+gama/2.0)/360.0*2*M_PI);
     setcolor(WHITE);
      gcvt(a[i],5, text);
      outtextxy(xi, yi, text);//извеждане на стойността
      na4=krai;//начален ъгъл за следващия сектор
          }
          }
  getch();
  return 0;
}
```



Фиг.7.3 Кръгова диаграма с форма на цилиндър с акцент върху най-големия сектор

ТЕМА 8

Двумерни преобразувания

Задача 8.1

Задачата илюстрира движение на графичен обект- окръжност, като първоначално тя се придвижва хоризонтално до центаъра на графичния прозорец, после се мащабира спрямо точката с най-малката координата y. Следва ротация на графичния обект, около центъра на най-голямата окръжност, получена след мащабирането.

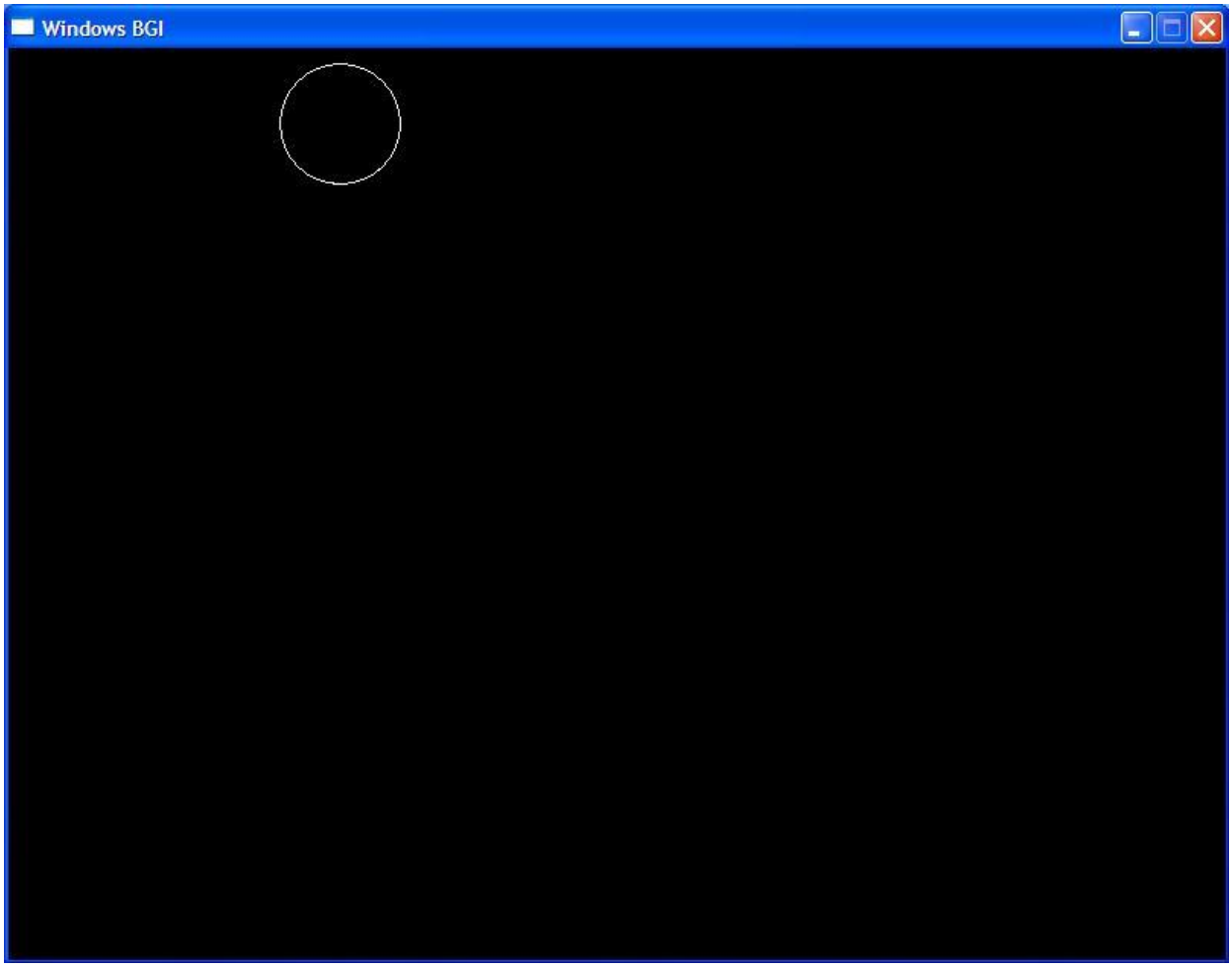
Използва се прост обект с център на симетрия – окръжност с радиус 40 пиксела, изчертана първоначално в горния ляв ъгъл на прозореца. Следва преместване на окръжността до средата на прозореца хоризонтално, като се показва движението (преместване с по един пиксел, последвано от изтриване (фиг.8.1-а).

След това се изпълнява няколкократно мащабиране само вертикално спрямо точка на мащабиране с най-малката y-координата на придвижената до центъра окръжност (фиг.8.1-в). Коефициент на мащабиране Sy=1.2, приложен многократно.

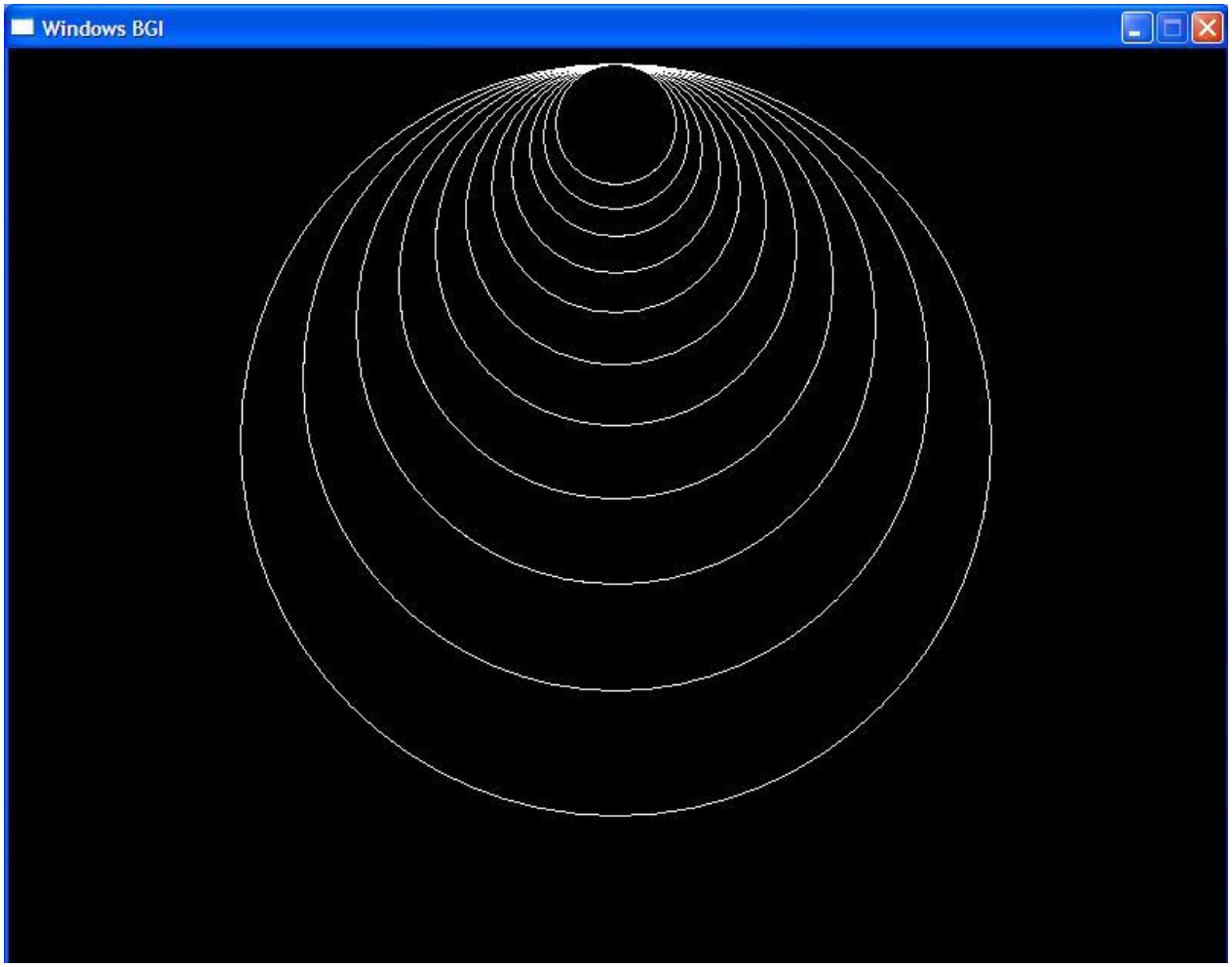
Третата част (фиг.8.1-с) реализира въртене на точка – центърът на малката окръжност се върти около центъра на голямата окръжност и малката окръжност се изчертава – получава се въртене на малката окръжност около голямата на 360 градуса с по ъгъл от един градус и показване на движението.

```
#include <graphics.h>
#include <iostream>
#include <dos.h>
#include <math.h>
using namespace std;
int main()
{
  int winwidth=800,winheight=600; // параметри на прозореца на графичната система
  double xc=50, yc=50, r = 40;//център и радиус на окръжността
  int i;
// инициализация на графичната система чрез отваряне на графичен прозорец със зададен размер
  initwindow(winwidth,winheight);

//преместване на окръжността до средата на прозореца хоризонално с показване на движението
//(движение по x)
        for(i = 0; i < (getmaxx()/2 - 50); i++)
   {
                delay(5);
                setcolor(0);
                circle(xc, yc, r);
                setcolor(15);
                xc++;
                circle(xc, yc, r);
    }

//мащабиране на окръжността спрямо точката с минимална стойност на y - yf и коефициент на
//мащабиране Sy=1.2 (мащабиране по y)
        double xf, yf;
        xf = xc; yf = yc - r;
        double Sy=1.2,nr;
```

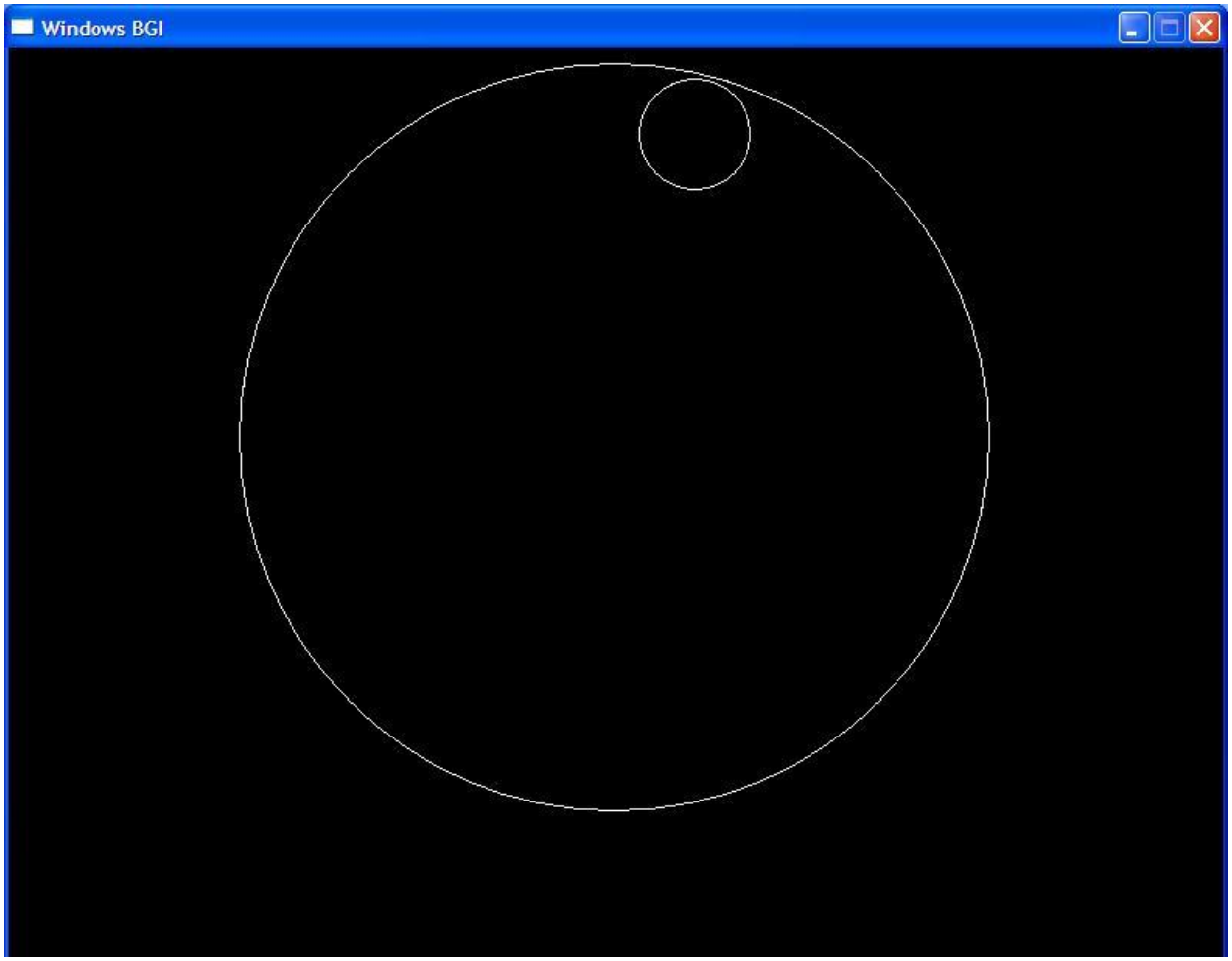
```
        while((yc+nr) < winheight-100)
   {
                delay(400);
                nr = (yc-yf) * Sy;
                yc=yf+(yc-yf)*Sy;
                circle(xc, yc, nr);
   }
//въртене на точка около точка - центъра на първата окръжност около центъра на последната с
//изчертаване на окръжностите
//и показване на движението
        delay(200);
        cleardevice();
        circle(xc, yc, nr);
        int yc1=yf+r;
        double x, y;
        for(i = 0; i < 360; i ++)
  {
         x = xc + (xc - xc)*cos(i*3.14/180) - (yc1 -yc)*sin(i*3.14/180);
         y = yc + (xc - xc)*sin(i*3.14/180) + (yc1 -yc)*cos(i*3.14/180);
         setcolor(WHITE);
        circle(x, y, r-3);
        delay(50);
        setcolor(BLACK);
        circle(x, y, r-3);
        }
        getch();
        return 0;
}
```



Фиг.8.1-а Движение по хоризонтала на обект



Фиг.8.1-в Мащабиране по вертикала на обект(центъра на малката окруцност) спрямо зададена точка на мащабиране и мащабен коефициент



Фиг.8.1-с Въртене на точка – центъра на малката окръжност около точка – центъра на голямата окръжност с ъгъл на въртене от 1 до 360 градуса и показване на движението

Задача 8.2

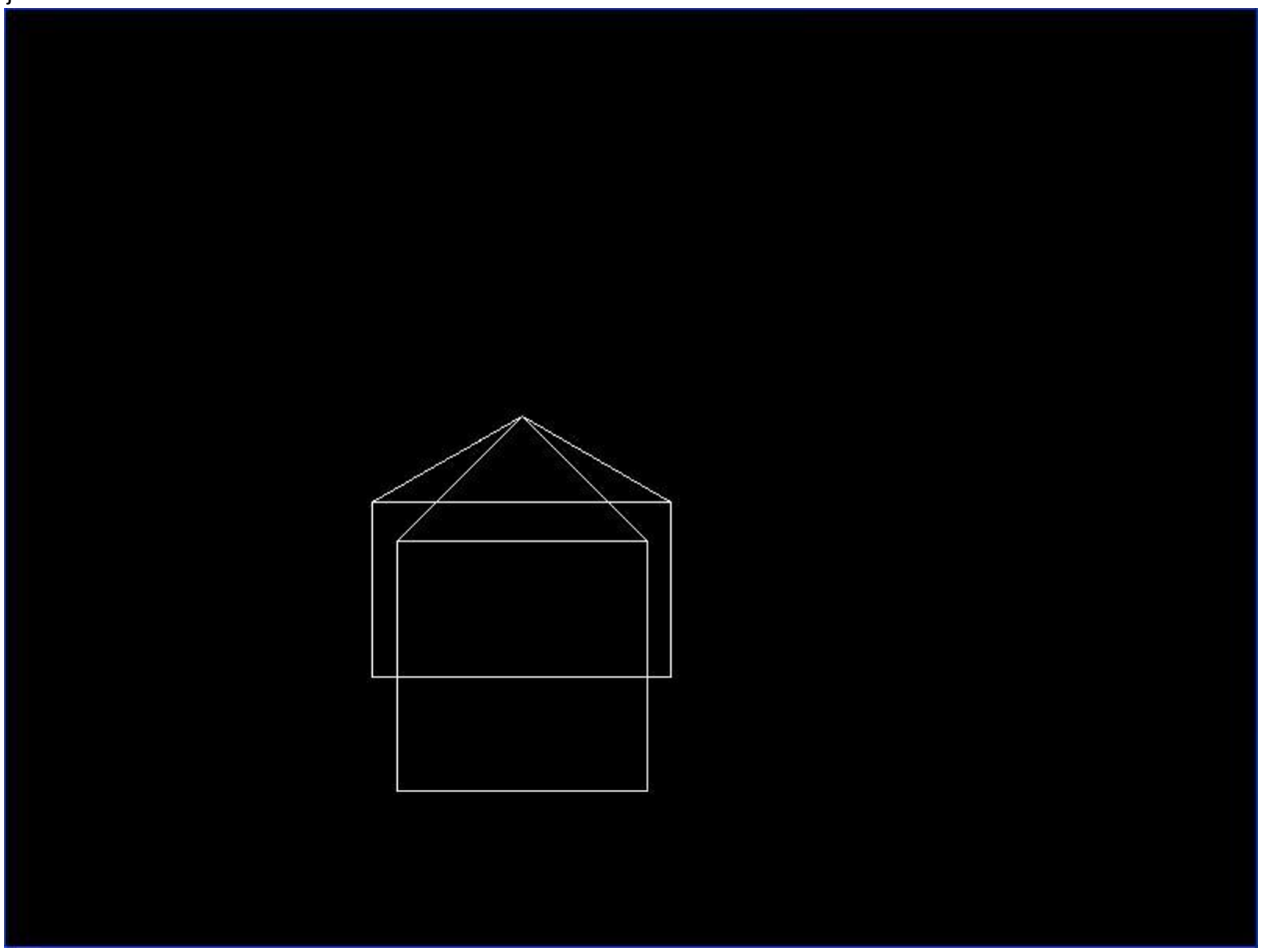
По зададени координати на връх А и дължина на страната на квадрат се изчертава първоначална фигура, която се мащабира с различни коефициенти на мащабиране спрямо т.М (фиг8.2-а).

След това е реализирано въртене на цялата фигура около точка М от един до 360 градуса през 60 градуса като се показва движението (фиг.8.2-в).

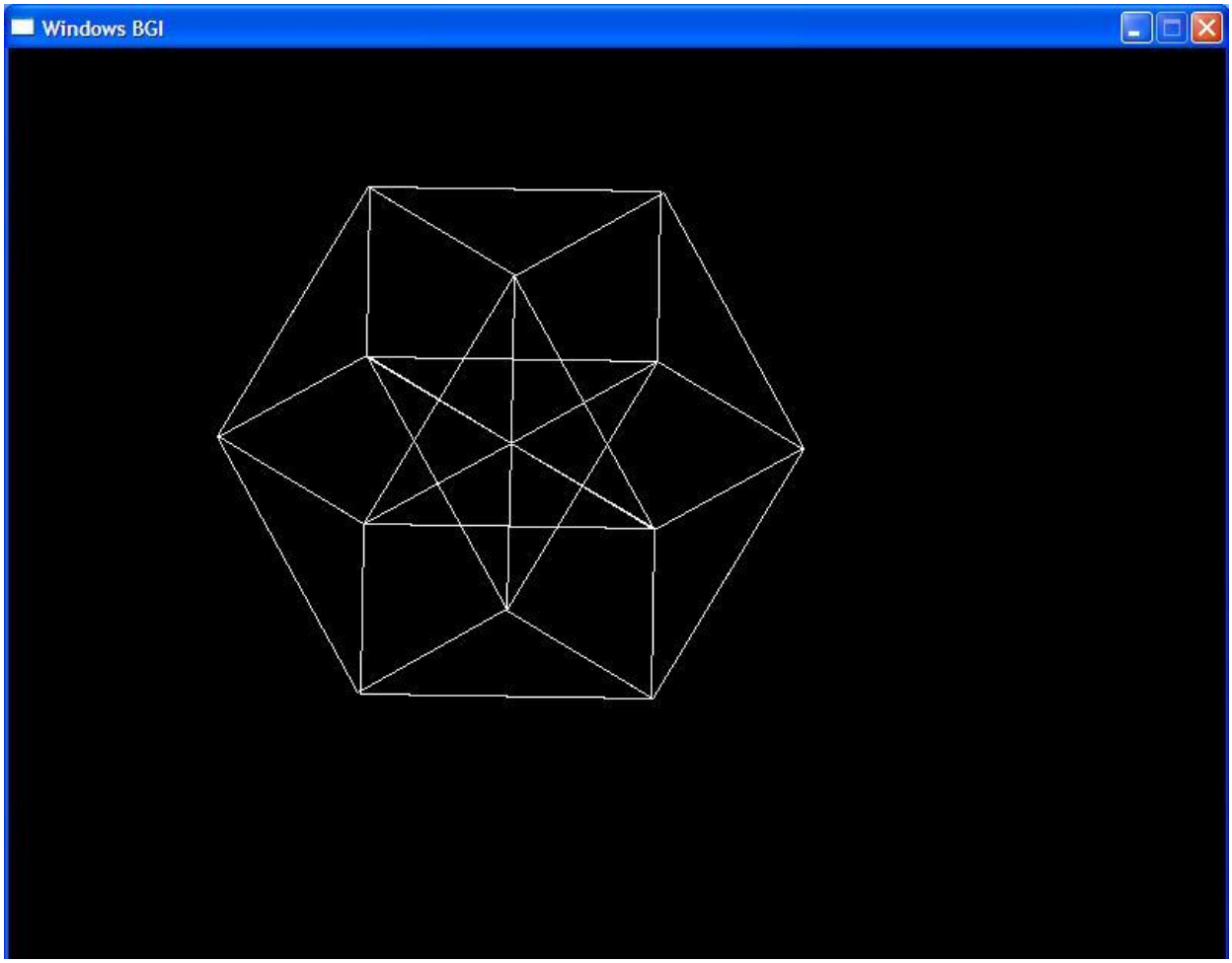
```
#include <graphics.h>
#include <iostream>
#include <dos.h>
#include <math.h>
using namespace std;
int main()
{
  int winwidth=800,winheight=600; // параметри на прозореца на графичната система
// инициализация на графичната система чрез отваряне на графичен прозорец със зададен размер
  initwindow(winwidth,winheight);
```

```
int ax = 250, ay = 500, a = 160, i; //зададени данни, необходими за изчертаване на фигурата

//определяне координатите на останалите върхове
   int bx=ax+a;
   int by=ay;
   int cx=ax+a;
   int cy=ay-a;
   int dx=ax;
   int dy=ay-a;
   int mx=ax+a/2;
   int my=ay-3./2*a;

//изчертаване
   rectangle(dx,dy,bx,by);
   line(dx,dy,mx,my);
   line(mx,my,cx,cy);
   delay(500);
// при изчертаването може да се използва и drawpoly

//мащабиране спямо т.М и зададени мащабни коефициенти sx = 1.2, sy = 0.7;
   double sx = 1.2, sy = 0.7;  //  мащабни коефициенти
//горен ляв
   int dxm = mx+ (dx-mx)*sx;
   int dym = my+ (dy-my)*sy;
//долен десен
   int bxm = mx+ (bx-mx)*sx;
   int bym = my+ (by-my)*sy;
//тези мащабирани точки са ни необходими за въртенето
   int axm = mx+ (ax-mx)*sx;
   int aym = my+ (ay-my)*sy;
   int cxm = mx+ (cx-mx)*sx;
   int cym = my+ (cy-my)*sy;

   rectangle(dxm,dym,bxm,bym);
   line(dxm,dym,mx,my);
   line(mx,my,cxm,cym);
   delay(500);
   cleardevice();

//завъртане около т.М на 360 градуса. Показва се движението като се изчертава завъртяната
фигура през 1 градус

for(i=1;i<=360;i+=360)
   {
                  int axr = mx + (axm - mx)*cos(i*M_PI/180) - (aym - my)*sin(i*M_PI/180);
                  int ayr = my + (axm - mx)*sin(i*M_PI/180) + (aym - my)*cos(i*M_PI/180);

                  int bxr = mx + (bxm - mx)*cos(i*M_PI/180) - (bym - my)*sin(i*M_PI/180);
                  int byr = my + (bxm - mx)*sin(i*M_PI/180) + (bym - my)*cos(i*M_PI/180);
```

```
                int cxr = mx + (cxm - mx)*cos(i*M_PI/180) - (cym - my)*sin(i*M_PI/180);
                int cyr = my + (cxm - mx)*sin(i*M_PI/180) + (cym - my)*cos(i*M_PI/180);

                int dxr = mx + (dxm - mx)*cos(i*M_PI/180) - (dym - my)*sin(i*M_PI/180);
                int dyr = my + (dxm - mx)*sin(i*M_PI/180) + (dym- my)*cos(i*M_PI/180);

                setcolor(15);
                moveto(axr,ayr);
                lineto(bxr, byr);
                lineto(cxr,cyr);
                lineto(dxr,dyr);
                lineto(axr,ayr);
                line(mx, my, cxr, cyr);
                line(mx, my, dxr, dyr);
                delay(40);
        }
  getch();
  return 0;
}
```



Фиг.8.2-а. Мащабиране на квадрата спрямо т.М с мащабни коефициенти sx = 1.2, sy = 0.7



Фиг.8.2-в. Въртене на мащабирания правоъгълник през 60 градуса.

ТЕМА 9
Интерполация и апроксимация – Многочлен на Безие

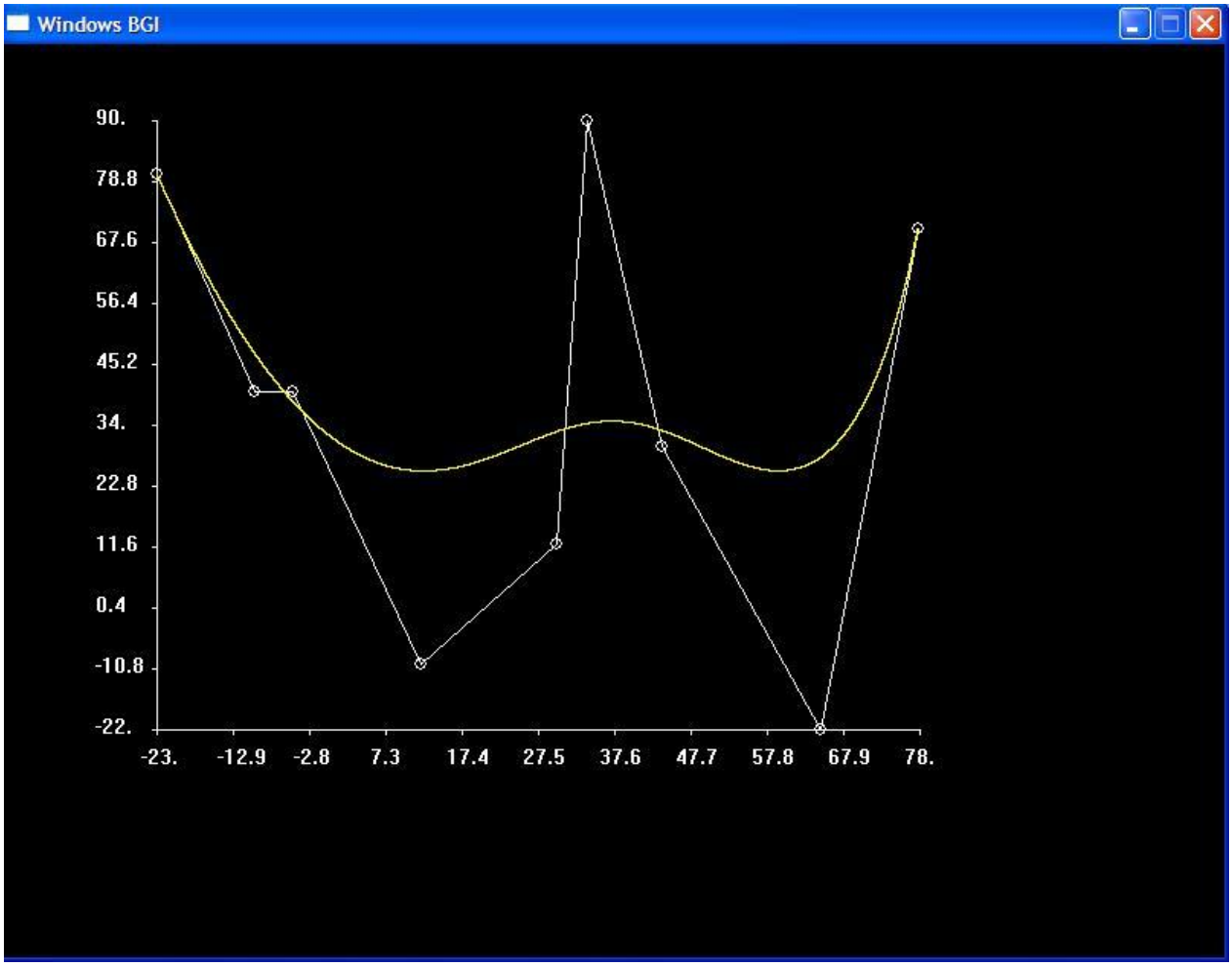
Задача 9.1

Използвайки алгоритъма за двумерно представяне освен на зададените данни да се изобрази и кривата на Безие построена по тези данни. Геометричният алгоритъм за построяване на многочлен на Безие (фиг.9.1) е добавен към текста на зад.3 са оцветени в червено. Реалните данни са преобразувани в зад.3 в координати на пиксели и те служат за първоначални точки ориентири за реализиране на алгоритъма. Параметърът t се изменя със стъпка 0.00001, тоест се получават 100000 пиксела от кривата на Безие.

```
#include <graphics.h>
#include <iostream>
using namespace std;
int main()
{
//задаване на входните данни - двойки стойности реални числа, които ще преобразуваме в
//координати на пиксели
// тези входни данни могат да постъпват от файл или клавиатура или в масив
  float x[] = {-5, 12, 78, -23, 34, -10, 65, 30, 44},temp;
  float y[] = {40, -10, 70, 80, 90, 40, -22, 12, 30};
  int n=sizeof(x)/sizeof(x[0]); // определяне на броя на входните данни
  int i,j;
  int winwidth=800,winheight=600; // параметри на прозореца на графичната система
  int px=500,py=400,Dx=50,Dy=40,x0=100,y0=450 ; //параметри на графичния прозорец, в който ще
//се изобразят данните (вътре в прозореца на графичната система)
  double Px[n], Py[n];//първоначални точки ориентири за построяване на многочлена на //Безие
   double Rx[n], Ry[n];//стари точки ориентири за построяване на многочлена на Безие
   double Qx[n], Qy[n];//нови точки ориентири за построяване на многочлена на Безие
// сортиране на входните данни, в случай, че те са експериментални
          for(i = 0; i < n; i++)
  {
            for(j = 0; j < n-i-1; j++)
         {
              if(x[j] > x[j+1])
              {
               temp = x[j]; x[j]=x[j+1]; x[j+1]=temp;
              temp = y[j]; y[j]=y[j+1]; y[j+1]=temp;
             }
         }
   }
//намиране на диапазона на изменение на входните данни - xmin,xmax,ymin,ymax
// след сортировката xmin и xmax са съответно първия и последен елемент от масива x
  float xmin = x[0];
  float xmax = x[n-1];

// намиране на ymin и ymax
  float ymin = y[0];
  float ymax = y[0];
```

```
  for(int i = 1; i< n; i++) {
          if(y[i] < ymin) ymin = y[i];
          if(y[i] > ymax) ymax = y[i];
  }

//определяне на скалните коефициенти
  float sx = (xmax - xmin)/px;
  float sy = (ymax - ymin)/py;

// инициализация на графичната система чрез отваряне на графичен прозорец със зададен размер
  initwindow(winwidth,winheight);

//изчертаване на графичния прозорец
  line(x0,y0,x0+px,y0);//хоризонтзлнз ос
  line(x0,y0,x0,y0-py);//вертикална ос

  int Ip = px/Dx; int Jp = py/Dy;//брой деления по хоризонталната и вертикалната ос

//изчертаване и надписване на деленията по хоризонталната ос
    char text[10];
    for(i = 0; i <= Ip; i++)
    {
    line(x0 + i*Dx, y0, x0+i*Dx,y0+3);   //изчертаване на деленията
    gcvt(xmin + i*Dx*sx, 5.2,text);      //преобразуване на реалната стойност, съответстваща на
//делението в символен низ
    outtextxy(x0 + i*Dx-10, y0+10, text);// извеждане на стойността, съответстваща на делението
    }

//изчертаване и надписване на деленията по вертикалната ос
    for(i = 0; i <= Jp; i++)
    {
    line(x0, y0-i*Dy, x0-3, y0- i * Dy);  //изчертаване на деленията
    gcvt(ymin+i*Dy*sy, 5.2,text);        //преобразуване на реалната стойност, съответстваща на
//делението в символен низ
    outtextxy(x0-40, y0 - Dy*i-10, text); // извеждане на стойността, съответстваща на делението
          }

//преобразуване на входните дании в координати на пиксели, които се използват за центрове на
//окръжности с радиус 4 пиксела
  for(i = 0 ; i < n; i ++)
  {
  int xprim=x0 + (x[i]-xmin)/sx;
  int yprim=y0 - (y[i]-ymin)/sy;
  circle(xprim,yprim,4);
  Px[i] = xprim; Py[i] = yprim; // подготовка на масивите, съдържащи координатите на
//първоначалните точки ориентири за кривата на Безие
  }
```

```
//свързване на окръжностите с отсечки и получаване на 2D графика, съответстваща на входните
//данни x,y
  for(i = 0; i < n-1; i++) {
          int xa = x0 + (x[i] - xmin)/sx;
          int ya = y0 - (y[i] - ymin)/sy;
          int xb = x0 + (x[i+1] - xmin)/sx;
          int yb = y0 - (y[i+1] - ymin)/sy;
          line(xa, ya, xb, yb);
  }
 //геометричен алгоритъм за построяване на многочлен на Безие
   int m = n;
          for(double t = 0; t <= 1; t+=0.0001)
      {
                  for(i = 0; i < m; i++)
                 {
                          Rx[i] = Px[i];
                          Ry[i] = Py[i];
                  }
                  n = m;
                  while(n > 0)
               {
                          for(i=0; i < n-1; i++)
                          {
                                  Qx[i] = Rx[i] + t*(Rx[i+1] - Rx[i]);
                                  Qy[i] = Ry[i] + t*(Ry[i+1] - Ry[i]);
                          }
                          n--;
                          for(i = 0; i < n; i++)
                         {
                                  Rx[i] = Qx[i];
                                  Ry[i] = Qy[i];
                          }
                  }
                  putpixel(Rx[0], Ry[0], 14);
          }
  getch();
  return 0;
}
```



Фиг.9.1 Реализация на геометричния алгоритъм за построяване на многочлен на Безие по точки ориентири

Задача 9.2

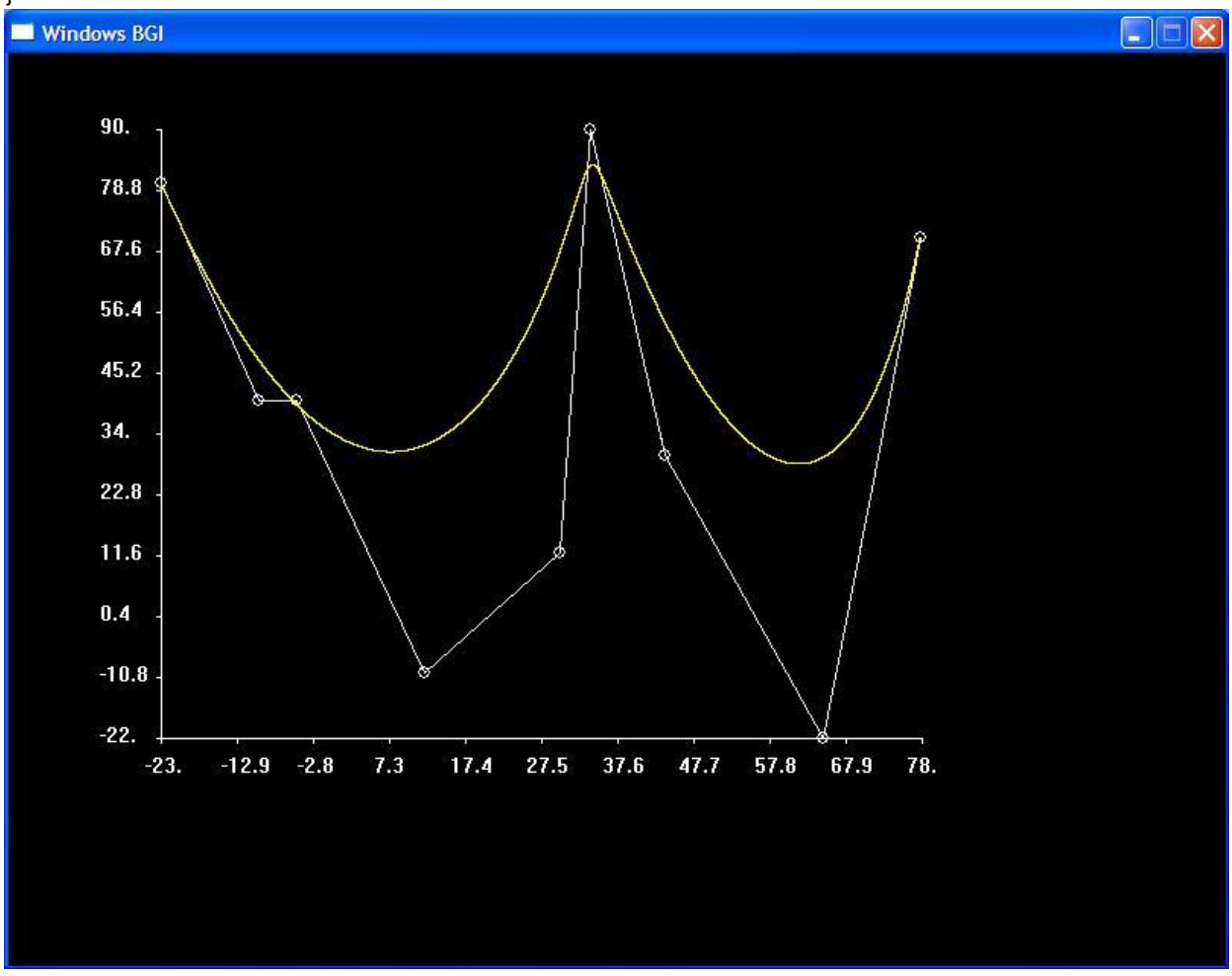
Използва се задача 9.1, за да се покаже как установяването на точки *многократно използвани* променят вида на кривата (фиг.9.2). Кратните точки са с еднакви стойности за x и съответно за y (много точки, разположени на едно и също място в прозореца). Колкото повече са точките на едно и също място, т.е колкото по-кратна е точката, толкова по-силно привлича тази точка кривата към себе си. Установена е една кратна точка – точка номер 6 с кратност 6 (използват се номерата на точките след сортировката). Допълненията са оцветени в червено.

```
#include <graphics.h>
#include <iostream>
using namespace std;
int main()
{
//задаване на входните данни - двойки стойности реални числа, които ще преобразуваме в
//координати на пиксели
// тези входни данни могат да постъпват от файл или клавиатура или в масив
  float x[] = {-5, 12, 78, -23, 34,34,34,34,34,34, -10, 65, 30, 44},temp;
  float y[] = {40, -10, 70, 80, 90,90,90,90,90,90, 40, -22, 12, 30};
  int n=sizeof(x)/sizeof(x[0]); // определяне на броя на входните данни
```

```
  int i,j;
  int winwidth=800,winheight=600; // параметри на прозореца на графичната система
  int px=500,py=400,Dx=50,Dy=40,x0=100,y0=450 ; //параметри на графичния прозорец, в който ще
//се изобразят данните (вътре в прозореца на графичната система)
   double Px[n], Py[n];//първоначални точки ориентири за построяване на многочлена на Безие
   double Rx[n], Ry[n];//стари точки ориентири за построяване на многочлена на Безие
   double Qx[n], Qy[n];//нови точки ориентири за построяване на многочлена на Безие
// сортиране на входните данни, в случай, че те са експериментални
            for(i = 0; i < n; i++)
            {
              for(j = 0; j < n-i-1; j++)
              {
                if(x[j] > x[j+1])
                {
                   temp = x[j]; x[j]=x[j+1]; x[j+1]=temp;
                   temp = y[j]; y[j]=y[j+1]; y[j+1]=temp;
                }
              }
            }
//намиране на диапазона на изменение на входните данни - xmin,xmax,ymin,ymax
// след сортировката xmin и xmax са съответно първия и последен елемент от масива x
  float xmin = x[0];
  float xmax = x[n-1];

// намиране на ymin и ymax
  float ymin = y[0];
  float ymax = y[0];
  for(int i = 1; i< n; i++) {
          if(y[i] < ymin) ymin = y[i];
          if(y[i] > ymax) ymax = y[i];
  }

//определяне на скалните коефициенти
  float sx = (xmax - xmin)/px;
  float sy = (ymax - ymin)/py;

// инициализация на графичната система чрез отваряне на графичен прозорец със зададен размер
  initwindow(winwidth,winheight);

//изчертаване на графичния прозорец
  line(x0,y0,x0+px,y0);//хоризонтзлнз ос
  line(x0,y0,x0,y0-py);//вертикална ос

  int Ip = px/Dx; int Jp = py/Dy;//брой деления по хоризонталната и вертикалната ос

//изчертаване и надписване на деленията по хоризонталната ос
    char text[10];
    for(i = 0; i <= Ip; i++)
    {
```

```
   line(x0 + i*Dx, y0, x0+i*Dx,y0+3);   //изчертаване на делениятa
    gcvt(xmin + i*Dx*sx, 5.2,text);      //преобразуване на реалната стойност, съответстваща на
//делението в символен низ
    outtextxy(x0 + i*Dx-10, y0+10, text);// извеждане на стойността, съответстваща на делението
    }

//изчертаване и надписване на делениятa по вертикалната ос
    for(i = 0; i <= Jp; i++)
    {
    line(x0, y0-i*Dy, x0-3, y0- i * Dy);  //изчертаване на делениятa
    gcvt(ymin+i*Dy*sy, 5.2,text);        //преобразуване на реалната стойност, съответстваща на
//делението в символен низ
    outtextxy(x0-40, y0 - Dy*i-10, text); // извеждане на стойността, съответстваща на делението
    }

//преобразуване на входните дании в координати на пиксели, които се използват за центрове на
//окръжности с радиус 4 пиксела
  for(i = 0 ; i < n; i ++)
  {
  int xprim=x0 + (x[i]-xmin)/sx;
  int yprim=y0 - (y[i]-ymin)/sy;
  circle(xprim,yprim,4);
  Px[i] = xprim; Py[i] = yprim; // подготовка на масивите, съдържащи първоначалните точки
//ориентири за кривата на Безие
  }

//свързване на окръжностите с отсечки и получаване на 2D графика, съответстваща на входните
//данни x,y
  for(i = 0; i < n-1; i++) {
            int xa = x0 + (x[i] - xmin)/sx;
            int ya = y0 - (y[i] - ymin)/sy;
            int xb = x0 + (x[i+1] - xmin)/sx;
            int yb = y0 - (y[i+1] - ymin)/sy;
            line(xa, ya, xb, yb);
  }
 //геометричен алгоритъм за построяване на многочлен на Безие
   int m = n;
            for(double t = 0; t <= 1; t+=0.00001)
 {
                        for(i = 0; i < m; i++)
                       {
                                    Rx[i] = Px[i];
                                    Ry[i] = Py[i];
                        }
                        n = m;
                        while(n > 0)
                        {
                                    for(i=0; i < n-1; i++)
                                   {
```

```
                        Qx[i] = Rx[i] + t*(Rx[i+1] - Rx[i]);
                        Qy[i] = Ry[i] + t*(Ry[i+1] - Ry[i]);
                    }
                n--;
                for(i = 0; i < n; i++)
                {
                        Rx[i] = Qx[i];
                        Ry[i] = Qy[i];
                }
            }
            putpixel(Rx[0], Ry[0], 14);
        }
  getch();
  return 0;
}
```



Фиг.9.2 Реализация на геометричния алгоритъм за построяване на многочлен на Безие по точки ориентири с установяване на кратна точка – точка номер 6 с кратност 6. Вижда се как кратната точка придърпва кривата към себе си.

Задача 9.3

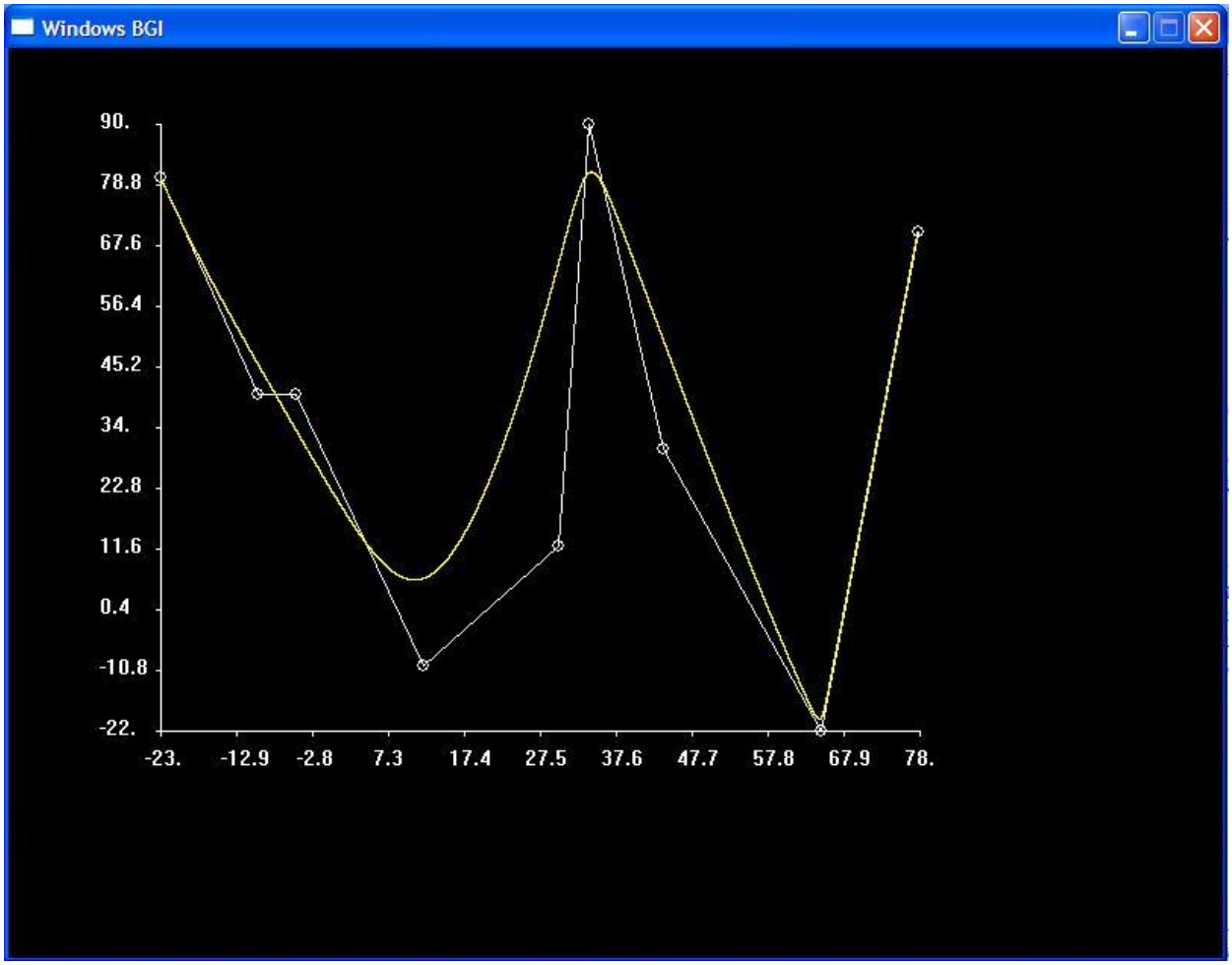
Използва се задача 9.1, за да се покаже как потребителят може да променя вида на кривата на Безие, като разполага кратни точки на различни места. Установени са три кратни

точки – точка номер 4 с кратност 5, точка номер 6 с кратност 8 и точка номер 8 с кратност 8 (използват се номерата на точките след сортировката). Допълненията са оцветени в червено.

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

using namespace std;
int main()
{
//задаване на входните данни - двойки стойности реални числа, които ще преобразуваме в
//координати на пиксели
// тези входни данни могат да постъпват от файл или клавиатура или в масив
  float x[] = {-5, 12,12,12,12,12 78, -23, 34,34,34,34,34,34,34,34, -10, 65,65,65,65,65,65,65,65, 30,
44},temp;
  float y[] = {40, -10,-10,-10,-10,-10, 70, 80, 90,90,90,90,90,90,90,90, 40, -22,-22,-22,-22,-22,-22,-22,-22,
12, 30};
  int n=sizeof(x)/sizeof(x[0]); // определяне на броя на входните данни
  int i,j;
  int winwidth=800,winheight=600; // параметри на прозореца на графичната система
  int px=500,py=400,Dx=50,Dy=40,x0=100,y0=450 ; //параметри на графичния прозорец, в който ще
//се изобразят данните (вътре в прозореца на графичната система)
          double Px[n], Py[n];//първоначални точки ориентири за построяване на многочлена на
Безие
          double Rx[n], Ry[n];//стари точки ориентири за построяване на многочлена на Безие
    double Qx[n], Qy[n];//нови точки ориентири за построяване на многочлена на Безие
// сортиране на входните данни, в случай, че те са експериментални
            for(i = 0; i < n; i++)
            {
              for(j = 0; j < n-i-1; j++)
               {
                 if(x[j] > x[j+1])
                 {
                   temp = x[j]; x[j]=x[j+1]; x[j+1]=temp;
                   temp = y[j]; y[j]=y[j+1]; y[j+1]=temp;
                 }
                }
              }
//намиране на диапазона на изменение на входните данни - xmin,xmax,ymin,ymax
// след сортировката xmin и xmax са съответно първия и последен елемент от масива x
  float xmin = x[0];
  float xmax = x[n-1];

// намиране на ymin и ymax
  float ymin = y[0];
  float ymax = y[0];
  for(int i = 1; i< n; i++) {
          if(y[i] < ymin) ymin = y[i];
          if(y[i] > ymax) ymax = y[i];
  }
```

```
//определяне на скалните коефициенти
  float sx = (xmax - xmin)/px;
  float sy = (ymax - ymin)/py;
// инициализация на графичната система чрез отваряне на графичен прозорец със зададен размер
  initwindow(winwidth,winheight);
//изчертаване на графичния прозорец
  line(x0,y0,x0+px,y0);//хоризонтзлнз ос
  line(x0,y0,x0,y0-py);//вертикална ос

  int Ip = px/Dx; int Jp = py/Dy;//брой деления по хоризонталната и вертикалната ос

//изчертаване и надписване на деленията по хоризонталната ос
    char text[10];
    for(i = 0; i <= Ip; i++)
    {
    line(x0 + i*Dx, y0, x0+i*Dx,y0+3);   //изчертаване на деленията
    gcvt(xmin + i*Dx*sx, 5.2,text);      //преобразуване на реалната стойност, съответстваща на
//делението в символен низ
     outtextxy(x0 + i*Dx-10, y0+10, text);// извеждане на стойността, съответстваща на  делението
    }
//изчертаване и надписване на деленията по вертикалната ос
    for(i = 0; i <= Jp; i++)
    {
    line(x0, y0-i*Dy, x0-3, y0- i * Dy);  //изчертаване на деленията
    gcvt(ymin+i*Dy*sy, 5.2,text);        //преобразуване на реалната стойност, съответстваща на
//делението в символен низ
    outtextxy(x0-40, y0 - Dy*i-10, text); // извеждане на стойността, съответстваща на делението
          }
//преобразуване на входните дании в координати на пиксели, които се използват за центрове на
//окръжности с радиус 2 пиксела
  for(i = 0 ; i < n; i ++)
  {
  int xprim=x0 + (x[i]-xmin)/sx;
  int yprim=y0 - (y[i]-ymin)/sy;
  circle(xprim,yprim,4);
  Px[i] = xprim; Py[i] = yprim; // подготовка на масивите, съдържащи първоначалните точки
//ориентири за кривата на Безие
  }
//свързване на окръжностите с отсечки и получаване на 2D графика, съответстваща на входните
//данни x,y
  for(i = 0; i < n-1; i++) {
          int xa = x0 + (x[i] - xmin)/sx;
          int ya = y0 - (y[i] - ymin)/sy;
          int xb = x0 + (x[i+1] - xmin)/sx;
          int yb = y0 - (y[i+1] - ymin)/sy;
          line(xa, ya, xb, yb);
  }
 //геометричен алгоритъм за построяване на многочлен на Безие
```

```
   int m = n;
        for(double t = 0; t <= 1; t+=0.00001)
{
                for(i = 0; i < m; i++)
                {
                        Rx[i] = Px[i];
                        Ry[i] = Py[i];
                }
                n = m;
                while(n > 0) {
                        for(i=0; i < n-1; i++) {
                                Qx[i] = Rx[i] + t*(Rx[i+1] - Rx[i]);
                                Qy[i] = Ry[i] + t*(Ry[i+1] - Ry[i]);
                        }
                        n--;
                        for(i = 0; i < n; i++)
                        {
                                Rx[i] = Qx[i];
                                Ry[i] = Qy[i];
                        }
                }
                putpixel(Rx[0], Ry[0], 14);
        }
  getch();
  return 0;
}
```



Фиг.9.3 Реализация на геометричния алгоритъм за построяване на многочлен на Безие по точки ориентири с установяване на кратни точки – точка номер 4 с кратност 5, точка номер 6 с кратност 8 и точка номер 8 с кратност 8. По този начин може да се управлява вида на кривата.

ТЕМА 10

Интерполация и апроксимация – Многочлен на Лагранж

Задача 10.1

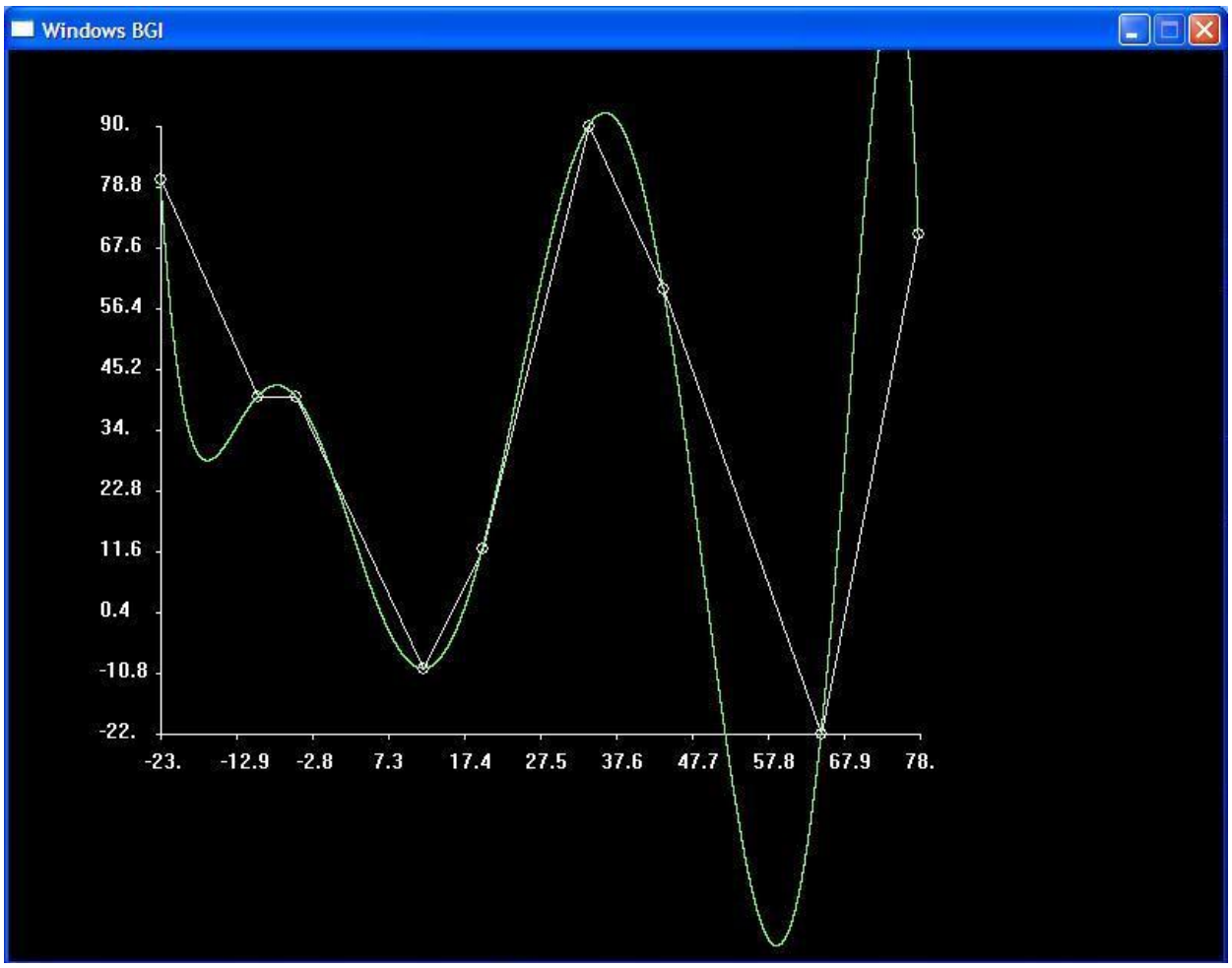
Използвайки алгоритъма за двумерна графика да се изобразят както входните данни така и построената по тях крива на Лагранж.

Върху зад.3 е реализирано построяване на многочлен на Лагранж (фиг.10.1). Допълненията са оцветени в червено. Реалните данни са преобразувани в зад.3 в координати на пиксели и те служат за реализиране на алгоритъма. Параметърът xx се изменя със стъпка 0.001, тоест се получават 1000 пиксела от интерполационната крива на Лагранж. Ясно се виждат колебанията, които претърпява интерполационната крива.

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

using namespace std;
int main()
{
//задаване на входните данни - двойки стойности реални числа, които ще преобразуваме в координати на пиксели
// тези входни данни могат да постъпват от файл или клавиатура или в масив
  float x[] = {-5, 12, 78, -23, 34,-10, 65, 20, 44},temp;
  float y[] = {40, -10,70, 80, 90, 40, -22, 12,60};
  int n=sizeof(x)/sizeof(x[0]); // определяне на броя на входните данни
  int i,j;
  int winwidth=800,winheight=600; // параметри на прозореца на графичната система
  int px=500,py=400,Dx=50,Dy=40,x0=100,y0=450 ; //параметри на графичния прозорец, в който ще се изобразят данните (вътре в прозореца на графичната система)
  double Px[n], Py[n];  // масивите ще съдържат координатите на точките, по които се  построява //многочлена на Лагранж

// сортиране на входните данни, в случай, че те са експериментални
            for(i = 0; i < n; i++)
            {
               for(j = 0; j < n-i-1; j++)
               {
                  if(x[j] > x[j+1])
                  {
                    temp = x[j]; x[j]=x[j+1]; x[j+1]=temp;
                    temp = y[j]; y[j]=y[j+1]; y[j+1]=temp;
                  }
               }
            }
//намиране на диапазона на изменение на входните данни - xmin,xmax,ymin,ymax
// след сортировката xmin и xmax са съответно първия и последен елемент от масива x
  float xmin = x[0];
  float xmax = x[n-1];

// намиране на ymin и ymax
  float ymin = y[0];
```

```
  float ymax = y[0];
  for(int i = 1; i< n; i++) {
          if(y[i] < ymin) ymin = y[i];
          if(y[i] > ymax) ymax = y[i];
  }

//определяне на скалните коефициенти
  float sx = (xmax - xmin)/px;
  float sy = (ymax - ymin)/py;

// инициализация на графичната система чрез отваряне на графичен прозорец със зададен размер
  initwindow(winwidth,winheight);

//изчертаване на графичния прозорец
  line(x0,y0,x0+px,y0);//хоризонтзлнз ос
  line(x0,y0,x0,y0-py);//вертикална ос

  int Ip = px/Dx; int Jp = py/Dy;//брой деления по хоризонталната и вертикалната ос

//изчертаване и надписване на деленията по хоризонталната ос
    char text[10];
    for(i = 0; i <= Ip; i++)
    {
    line(x0 + i*Dx, y0, x0+i*Dx,y0+3);   //изчертаване на деленията
    gcvt(xmin + i*Dx*sx, 5.2,text);      //преобразуване на реалната стойност, съответстваща на
//делението в символен низ
     outtextxy(x0 + i*Dx-10, y0+10, text);// извеждане на стойността, съответстваща на делението
    }

//изчертаване и надписване на деленията по вертикалната ос
    for(i = 0; i <= Jp; i++)
    {
    line(x0, y0-i*Dy, x0-3, y0- i * Dy);  //изчертаване на деленията
     gcvt(ymin+i*Dy*sy, 5.2,text);        //преобразуване на реалната стойност, съответстваща на
//делението в символен низ
    outtextxy(x0-40, y0 - Dy*i-10, text); // извеждане на стойността, съответстваща на делението
     }

//преобразуване на входните дании в координати на пиксели, които се използват за центрове на
окръжности с радиус 4 пиксела
  for(i = 0 ; i < n; i ++)
  {
  int xprim=x0 + (x[i]-xmin)/sx;
  int yprim=y0 - (y[i]-ymin)/sy;
  circle(xprim,yprim,4);
  Px[i] = xprim; Py[i] = yprim; // подготовка на масивите, съдържащи координатите на точките, по
//които се строи многочлена на Лагранж
```

```
//свързване на окръжностите с отсечки и получаване на 2D графика, съответстваща на входните данни x,y
  for(i = 0; i < n-1; i++) {
            int xa = x0 + (x[i] - xmin)/sx;
            int ya = y0 - (y[i] - ymin)/sy;
            int xb = x0 + (x[i+1] - xmin)/sx;
            int yb = y0 - (y[i+1] - ymin)/sy;
            line(xa, ya, xb, yb);
  }

//многочлен на Лагранж
double yy,pr,xx;
    for(double xx=Px[0];xx<=Px[n-1];xx+=0.001)
    {
       yy=0;
       for (i=0;i<n;i++)
       {
         pr=1;
         for(j=0;j<n;j++)
         {
         if(i!=j)
         pr=pr*(xx-Px[j])/(Px[i]-Px[j]);
         }
       yy=yy+Py[i]*pr;
       }
    putpixel(xx,yy,LIGHTGREEN);
    }
  getch();
  return 0;
}
```



Фиг.10.1. Реализация на интерполационния многочлен на Лагранж по точките, получени в задача 3. Виждат се колебанията, които претърпява интерполационната крива.

Задача 10.2

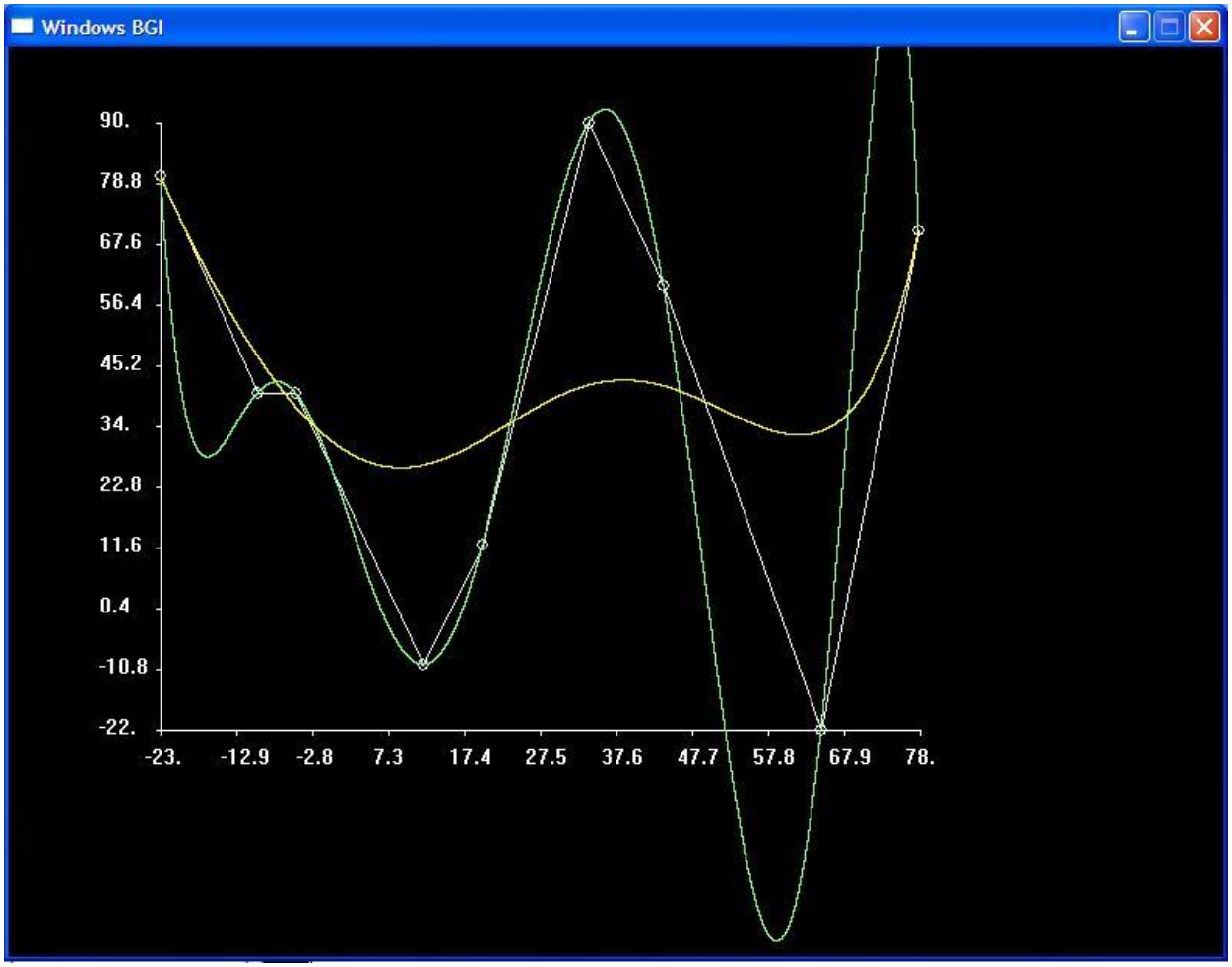
Върху зад.3 са реализирани интерполационния *многочлен на Лагранж и геометричния алгоритъм за построяване на многочлена на Безие.* Допълненията са оцветени в червено. Реалните данни са преобразувани в зад.3 в координати на пиксели и те служат за реализиране на алгоритите. Може да се направи съпоставка на двете криви.

```
#include <graphics.h>
#include <iostream>
using namespace std;
int main()
{
//задаване на входните данни - двойки стойности реални числа, които ще преобразуваме в
координати на пиксели
// тези входни данни могат да постъпват от файл или клавиатура или в масив
  float x[] = {-5, 12, 78, -23, 34,-10, 65, 20, 44},temp;
  float y[] = {40, -10,70, 80, 90, 40, -22, 12,60};
  int n=sizeof(x)/sizeof(x[0]); // определяне на броя на входните данни
  int i,j;
```

```
  int winwidth=800,winheight=600; // параметри на прозореца на графичната система
  int px=500,py=400,Dx=50,Dy=40,x0=100,y0=450 ; //параметри на графичния прозорец, в който ще
се изобразят данните (вътре в прозореца на графичната система)
  double Px[n], Py[n];//първоначални точки ориентири за построяване на многочлена на Безие, а
//също използвани и за многочлена на Лаграж
  double Rx[n], Ry[n];//стари точки ориентири за построяване на многочлена на Безие
   double Qx[n], Qy[n];//нови точки ориентири за построяване на многочлена на Безие
// сортиране на входните данни, в случай, че те са експериментални
            for(i = 0; i < n; i++)
            {
              for(j = 0; j < n-i-1; j++)
              {
                 if(x[j] > x[j+1])
                 {
                  temp = x[j]; x[j]=x[j+1]; x[j+1]=temp;
                  temp = y[j]; y[j]=y[j+1]; y[j+1]=temp;
                 }
              }
             }
//намиране на диапазона на изменение на входните данни - xmin,xmax,ymin,ymax
// след сортировката xmin и xmax са съответно първия и последен елемент от масива x
  float xmin = x[0];
  float xmax = x[n-1];

// намиране на ymin и ymax
  float ymin = y[0];
  float ymax = y[0];
  for(int i = 1; i< n; i++) {
          if(y[i] < ymin) ymin = y[i];
          if(y[i] > ymax) ymax = y[i];
  }

//определяне на скалните коефициенти
  float sx = (xmax - xmin)/px;
  float sy = (ymax - ymin)/py;

// инициализация на графичната система чрез отваряне на графичен прозорец със зададен размер
  initwindow(winwidth,winheight);

//изчертаване на графичния прозорец
  line(x0,y0,x0+px,y0);//хоризонтзлнз ос
  line(x0,y0,x0,y0-py);//вертикална ос

  int Ip = px/Dx; int Jp = py/Dy;//брой деления по хоризонталната и вертикалната ос

//изчертаване и надписване на деленията по хоризонталната ос
   char text[10];
   for(i = 0; i <= Ip; i++)
   {
```

```
    line(x0 + i*Dx, y0, x0+i*Dx,y0+3);   //изчертаване на деленията
    gcvt(xmin + i*Dx*sx, 5.2,text);      //преобразуване на реалната стойност, съответстваща на
//делението в символен низ
    outtextxy(x0 + i*Dx-10, y0+10, text);// извеждане на стойността, съответстваща на делението
    }

//изчертаване и надписване на деленията по вертикалната ос
    for(i = 0; i <= Jp; i++)
    {
    line(x0, y0-i*Dy, x0-3, y0- i * Dy);  //изчертаване на деленията
    gcvt(ymin+i*Dy*sy, 5.2,text);        //преобразуване на реалната стойност, съответстваща на
//делението в символен низ
    outtextxy(x0-40, y0 - Dy*i-10, text); // извеждане на стойността, съответстваща на делението
    }

//преобразуване на входните дании в координати на пиксели, които се използват за центрове на
окръжности с радиус 4 пиксела
  for(i = 0 ; i < n; i ++)
  {
  int xprim=x0 + (x[i]-xmin)/sx;
  int yprim=y0 - (y[i]-ymin)/sy;
  circle(xprim,yprim,4);
  Px[i] = xprim; Py[i] = yprim; // подготовка на масивите, съдържащи първоначалните точки
//ориентири за кривата на Безие и многочлена  на Лагранж
  }

//свързване на окръжностите с отсечки и получаване на 2D графика, съответстваща на входните
данни x,y
  for(i = 0; i < n-1; i++) {
            int xa = x0 + (x[i] - xmin)/sx;
            int ya = y0 - (y[i] - ymin)/sy;
            int xb = x0 + (x[i+1] - xmin)/sx;
            int yb = y0 - (y[i+1] - ymin)/sy;
            line(xa, ya, xb, yb);
  }

//многочлен на Лагранж
double yy,pr,xx;
    for(double xx=Px[0];xx<=Px[n-1];xx+=0.001)
    {
       yy=0;
       for (i=0;i<n;i++)
       {
         pr=1;
         for(j=0;j<n;j++)
         {
         if(i!=j)
         pr=pr*(xx-Px[j])/(Px[i]-Px[j]);
         }
```

```
        yy=yy+Py[i]*pr;
        }
    putpixel(xx,yy,LIGHTGREEN);
    }

//геометричен алгоритъм за построяване на многочлен на Безие
  int m = n;
            for(double t = 0; t <= 1; t+=0.00001)
            {
                        for(i = 0; i < m; i++)
                        {
                                    Rx[i] = Px[i];
                                    Ry[i] = Py[i];
                        }
                        n = m;
                        while(n > 0) {
                                    for(i=0; i < n-1; i++)
                                    {
                                                Qx[i] = Rx[i] + t*(Rx[i+1] - Rx[i]);
                                                Qy[i] = Ry[i] + t*(Ry[i+1] - Ry[i]);
                                    }
                                    n--;
                                    for(i = 0; i < n; i++)
                                    {
                                                Rx[i] = Qx[i];
                                                Ry[i] = Qy[i];
                                    }
                        }
                        putpixel(Rx[0], Ry[0], 14);
            }
   getch();
   return 0;
}
```



Фиг.10.2. Реализирани са интерполационния многочлен на Лагранж(в зелен цвят) и геометричния алгоритъм за построяване на многочлена на Безие (в жълт цвят). Коментарите за двете криви оставяме на читателя.