14 ОҚИҒАЛАР

14.1 Оқиға ұғымы

Windows операциялық жүйесі жұмысының негізінде оқиғаны басқару принципі жатады. Windows жүйесіне арналған барлық қосымшалар іске қосылғаннан кейін пайдаланушының әрекеттерін немесе операциялық жүйенің хабарламаларын күтеді және оларға белгілі бір тәртіпте жауап береді - қосымшаның хабарламалар кезегі арқылы оқиғалар құрылады.

Әрбір басқару элементінің (батырма немесе меню жолы) өз идентификаторы болады. Батырманы басқан кезде немесе меню жолын таңдаған кезде Windows қосымшаның хабарламалар кезегіне хабарламаны кіргізеді. Ол хабарламада қолданылған басқару элементінің идентификаторы болады. Windows операциялық жүйесі басқару элементінен келген хабарламаны өзітиісті қосымшаның (басқару элементі осы қосымшаға тиісті) кезегіне бағыттайды.

Егер форма терезесінде батырма басылған болса, онда батырманы басу факті (хабарламасы) Windows операциялық жүйесі арқылы форма терезесінің хабрламалар кезегіне қайтып келеді ады және осыдан кейін батырманың Click оқиғасы пайда болады.

Сонымен, оқиғаның пайда болу механизмі түсінікті, бірақ оқиға деген не?

Біз 21-бөлімде (класс құрылымы) класс оқиғаларын арнайы әрекеттер ретінде анықтадық, ол оқиғалар кластың пайдаланушы әрекеттеріне немесе бағдарламаның белгілі өзгерістеріне жауап беруіне мүмкіндік береді.

Осы анықтамадан кейін оқиғалар мен әдістер – оқиғалар өңдеуішіне синоним болып табылады.

Осы ұғымдарды анықтайық. Әрбір объект белгілі бір кластың данасы болып келеді. Класс қасиеттері оның жағдайын, ал класс әдістері оның әрекеттерін анықтайды.

Барлық объекттерде қасиеттер жиыны бірдей, бірақ объекттер қасиеттерінің мәні әртүрлі, сондықтан бір кластың объекттері әртүрлі жағдайларда бола алады. Мысалы, «Қызметкер» класының объекттерінде «кіріс» қасиетінің мәні әртүрлі болуы мүмкін және осы объекттер әртүрлі күйде болуы мүмкін. Мысалы, «автомобильді сатып алғым келеді, бірақ оған жағдайым жоқ …».

Барлық объекттердің әдістері бірдей және бір кластың барлық объекттерінің оқиғалар жиыны бірдей болады. Бірақ пайда болған оқиғаларды өңдейтін әдістер әртүрлі болуы мүмкін. Мысалы, форманың екі батырмасының («Деректерді енгізу», «2-формаға көшу») Click оқиғалары бірдей, бірақ оқиғалар өңдеуіштері (олардың кодтары) әр түрлі.

Сонымен, оқиға класс қасиеті ретінде барлық объекттерге ортақ болады және кластың әрбір нақты данасы үшін оның жүзеге асырылуы әртүрлі болады.

Оқиғалар кейбір хабарлама пайда болған кезде объекттің жекеленген әрекетін анықтауға мүмкіндік береді.

Сонымен, класс оқиғалары дегеніміз –пайдаланушы әрекеттеріне немесе бағдарламадағы өзгерістерге класқа жауап беруіне мүмкіндік беретін арнайы әдістер.

Пайдаланушының көптеген әрекеттеріне оқиғалар өңдеуіштерінің үлгілері құрылып қойылған (Properties ->Events).

14.2 Visual Studio.NET ортасының жиі қолданылатын кейбір оқиғалары

Windows–қосымшалардың әрбір басқару элементі үшін оқиғалардың өз жинағы анықталған. Оқиғалар өңдеуіштері бойынша үлгінің дайындамасы мына жолмен құрылады: керекті басқару элементінің Properties терезесіндегі, Events бетіндегі сәйкес оқиға атауының оң жағында орналасқан өрісті екі рет шерту.

Басқару элементтерінде жиі қолданылатын оқиғалар:

Click, DoubleClick – тышқанды бір немесе екі рет шерту;

KeyDown, KeyUp – кез келген пернені (пернелер тіркестері) басу немесе оны жіберу;

KeyPress – пернені басу (ASCII-коды бар);

MouseDown, MouseUp – тышқан батырмасын басу немесе оны жіберу;

MouseMove – тышқан орнын ауыстыру;

Paint – форма кескінін салу керек болған кезде пайда болады.

Мысалы, формада орналасқан «Басу» батырмасын екі рет шертсе, онда осы оқиғаның өңдеуіші құрылады, өңдеуіш тақырыбының жазылуы төменде көрсетілген:

```
    private void button1_Click(object sender,
EventArgs e).
```

Егер бізге батырманы екі рет басу нәтижесінде пайда болған оқиғаны өңдеу керек болса, онда форманың оқиғаларында «KeyDown» оқиғасының оң жағында өрісті басу керек, нәтижесінде осы оқиғаның өңдеуіш үлгісі пайда болады, өңдеуіш тақырыбының жазылуы мына түрде болады: private void Form1_MouseDoubleClick(object sender, MouseEventArgs e).

Мысал ретінде тағы екі оқиға өңдеуішінің үлгілерін көрсетейік:

```
    private void textBox1_KeyDown(object sender,
KeyEventArgs e)
    private void Form1_MouseClick(object sender,
MouseEventArgs e)
```

Олардың қашан құрылатынын, өзіңіз анықтап көріңіз?

Барлық оқиғаның өңдеуішінің үлгілерінде екі формалды параметр бар, сонымен қатар бірінші параметр (object sender) барлығында бірдей.

C# тілінде бағдарламалау бойынша көптеген оқулықтарда оқиға жұмысының механизмін түсіндірген кезде «жариялау-жазылу» үлгісі қолданылады. Осы үлгі бойынша хабарламаны (sender) жіберуші бір класс хабарламаны жариялайды, ал хабарламаны алушы басқа кластар осы хабарламаларды алуға жазылады.

Жоғарыда жазылған терминологияға сәйкес барлық хабарламалар өңдеуіштерінің object (C# тілінде object класы кез келген класқа базалық класс болып келеді) типіндегі sender бірінші формалды параметрі хабарлама жіберуші болып табылады (хабарлама жіберуші).

Оқиғалар өңдеуішінің үлгілеріндегі екінші формалды параметр объект типіндегі айнымалы (XXXEventArgs класының объектіне сілтеме) болып келеді. Мысалы, тышқан өңдеуіші үшін тышқан меңзерінің e.X e.Y координаттарын алуға болады.

XXXEventArgs класы (XXX – оқиға атауы) барлық стандартты оқиғаларда EventArgs жүйелі класының мұрагері болып табылады.

«Жариялау-жазылу» үлгісін сипаттағанда хабарларды алушылар (receivers) осы хабарларды алуға жазылатынын ескерткен болатынмыз. Яғни басқару элементтері мен форма үшін хабарларды көрсету (жазу) керек. Оқиғалар өңдеуіштерінің әдістері бірыңғай жазылу пішімінде болғандықтан, оларды көпадресті делегаттар – функциялардың кез келген санын көрсете алатын делегаттар арқылы біріктіруге болады. Form1.Designer.cs файлында, инициализация бөлімінде әрбір элемент үшін қасиеттерді анықтаумен қатар, оқиғалардың тізімі анықталады. Мысалы, оқиғалар тізімі анықталған форманың инициализациялау үзіндісі келесі түрде болады:

```
     private void InitializeComponent()
     {
      . . .
     this.MouseDoubleClick += new
System.Windows.Forms.MouseEventHandler
      (this.Form1_MouseDoubleClick);
     this.Paint += new
System.Windows.Forms.PaintEventHandler(this.Form1_Paint);
     this.MouseClick += new
System.Windows.Forms.MouseEventHandler
      (this.Form1_MouseClick);
     this.MouseMove += new System.Windows.Forms.MouseEventHandler
     (this.Form1_MouseMove);
          . . ..
     }
```

Екскерту, барлық визуалды басқару элементтер Control класынан таралған.

14.3 Кластардың стандартты оқиғаларын қолдану мысалы

Кластардың «стандартты» оқиғаларын қолданатын бағдарламаны қарастырайық

Қосымшада формадағы тышқанның сол жақ батырмасын бір рет басу бойынша пайда болатын оқиға өңдеуіші қолданылған, сонымен қатар қосымшада меңзер тек форманың сол жақ бөлігінде орналасқан жағдайда ғана тышқан батырмасын басу оқиғасына жауап қайтарады:

```
    private void Form1_MouseClick(object sender,
MouseEventArgs e)
     {
    if ((e.X < 20 && e.X > 0 && e.Y < 20 & e.Y > 0) &&
p == 0)
```

Осы тәсілді бағдарламашылар экранның белгілі орнында тышқанның шертілуіне жауап қайтару үшін жиі қолданады. Мысалы, қала картасында, топтағы студентердің фотосуреттерінде – форманың белгілі бөлігіне тышқан меңзерін қойған кезде керекті түсініктемелерді көрсетуге болады.

Форма бойымен тышқанның орын ауыстыру оқиғасының өңдеуіші тышқан меңзерінің орнын 100 нүктеден тұратын массивке жаза алады (қосымша шарт p==1).

Графикалық режімде массивті қарап шығу тышқанның сол жақ батырмасын екі рет басу нәтижесінде орындалады.

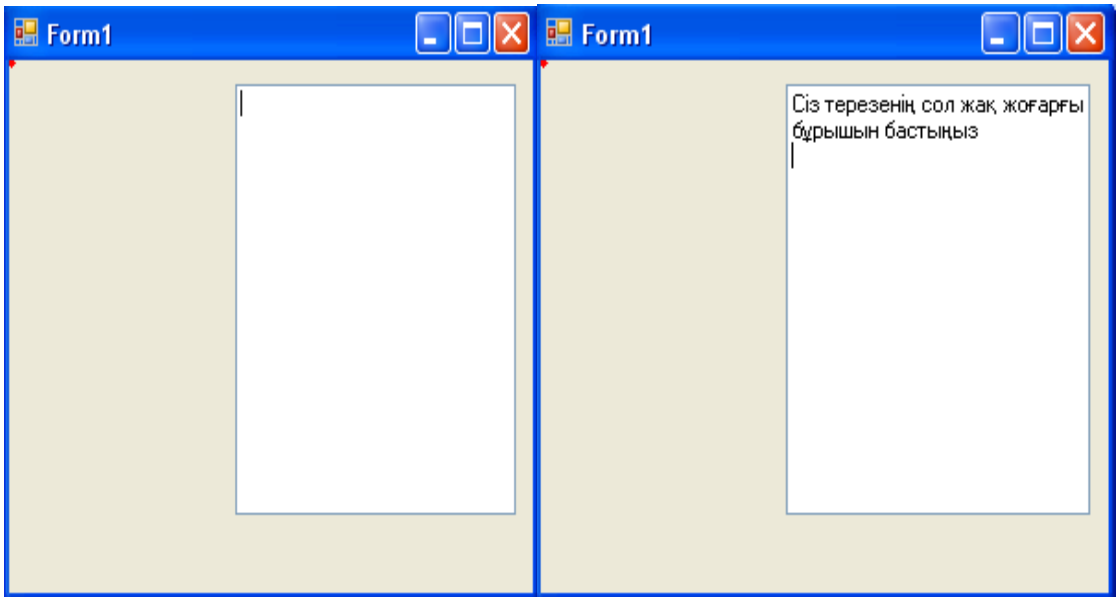
Батырманы басу өңдеуішінің көмегімен редактор терезесінде бағдарламаның алғашқы нұсқауы жүзеге асады.

Form1.cs файлының коды:

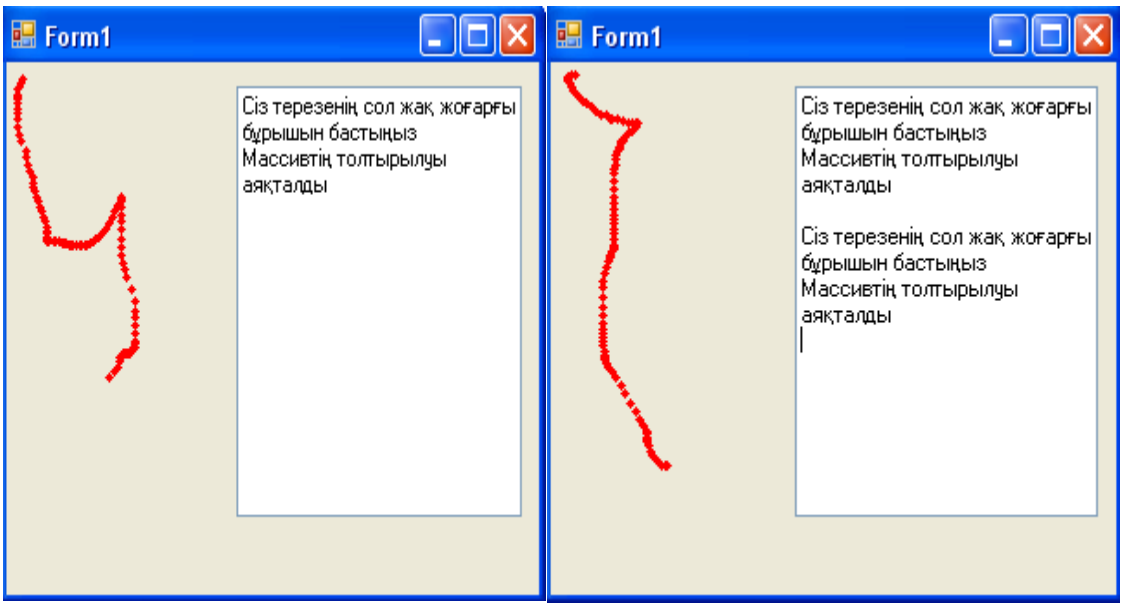
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
namespace WindowsApplication1
{
 public partial class Form1 : Form
 {
 public int p,i,j;
 int[,] tk = new int[100, 2];
 public Form1()
 {
 InitializeComponent();
 textBox1.Text = "";
 p = 0; i = 0; j = 0;
 }
 private void Form1_MouseClick(object sender, MouseEventArgs
e)
 {
 if ((e.X < 20 && e.X > 0 && e.Y < 20 & e.Y > 0) && p == 0)
 {
 textBox1.AppendText("Сіз терезенің сол жақ жоғарғы бұрышын
бастыңыз \r\n");
```

```
        p++;
        }
        }
        private void Form1_MouseMove(object sender, MouseEventArgs
e)
        {
        if (p == 1)
        if (i < 100) { tk[i, 0] = e.X; tk[i, 1] = e.Y; i++; }
        else
        {
        textBox1.AppendText("Массивтің толтырылуы аяқталды \r\n");
        p++;
        }
        }
        private void Form1_MouseDoubleClick(object sender,
      MouseEventArgs e)
        {
        this.Invalidate();
        }
        private void Form1_Paint(object sender, PaintEventArgs e)
        {
        Pen myPen = new Pen(Color.Red, 2);
        Graphics g = e.Graphics;
        for (int n = 0; n < 100; n++)
        g.DrawEllipse(myPen, tk[n, 0], tk[n, 1], 2, 2);
        }
        private void textBox1_KeyDown(object sender, KeyEventArgs
e)
        {
        p = 0; i = 0;
        }
        }
      }
```

Қосымша жұмысы 14.1 және 14.2 суреттерінде көрсетілген.



14.1-сурет – Қосымшаны іске қосу және массивті толтыру



14.2-сурет – Массивті көрсету, қосымшаны босату және оны қайтадан іске қосу

14.4 Кластардың стандартты емес оқиғалары

Пайдаланушы әрекеттеріне қосымшаның жауап қайтаруы ретіндегі стандартты оқиғалармен қатар Windows-қосымшаның ішінде арнайы әдістер арқылы оқиғалар құрылуы мүмкін.

Қосымшаның өзі құратын оқиғалар түрлі объекттер арасында динамикалық байланысты ұйымдастыруға мүмкіндік береді, мысалы, дүкенде белгілі бір тауардың сатылуы көптеген құжаттардың ішіндегісін

автоматты түрде өзгертуі ерек (Сатудан түсетін жалпы кіріс, нақты тауарды сату бойынша құжат, осы тауардың қоймада болуы, т.б.).

Сонымен, хабарды шығару көзінен пайда болған оқиға туралы қосымшаның кейбір объекттеріне хабар беру қажеттілігі туындайды.

Оқиға көзі және оқиғану алушы (кейде оларды клиент деп атайды) арасындағы өзара әрекеттерінің механизмі делегатты пайдалануға негізделген.

Қосымшада делегат данасы жарияланады, ол оқиғалар өңдеуіштерінің стандартты әдістеріне сәйкес болып келеді.

Одан кейін оқиға көзі (sender) болып келетін класты анықтау керек және сол жерде оқиғаны сипаттайтын әдісті, оқиғаны іске асыратын (инициирующий) әдісті анықтау керек.

Қосымшаның жұмысы барысында оқиғалар өңдеуіштерінің объекттерін (клиенттері) делегат арқылы өңделетін әдістер тізіміне қосу керек. Осы процесс оқиғалар өңдеуіштерін тіркеу деп аталады.

Оқиға пайда болғанда барлық тіркелген әдістер делегат арқылы кезек бойынша орындалуға шақырылады.

Оқиға көзі мен пайдаланушы арасындағы өзара әрекеттесу механизмінің жұмысы келесі мысалда қарастырылған.

14.1-есеп. Минус 5-тен 10-ға дейінгі аралықта болатын 10 кездейсоқ бүтін сандардан тұратын массивті құру керек. Теріс сандарды қолдануға болмайды деп жорамалдайық және олар үшін оқиғаларды құрайық. массивті экранға басып шығару, оның қосындысын және мәндердің өзгеру графигінің суретін салу керек.

Егер массив мәні теріс сан болатын болса, онда оқиғаны құру керек және осы оқиғаға екі өңдеуіш жауап беруі керек.

Біріншісі теріс санның таңбасын ауыстыру керек, екіншісі – массив элементтерінің жаңа мәндеріне сәйкес сандардың қосындысын өзгерту керек.

Көрнекілік үшін массивтің өзгерген мәндерінің графигі шығарылсын.

Сонымен, оқиғаны құру және қолдану үшін алдымен делегатты жариялау керек (делегат арқылы клиент пен оқиға көзі арасында байланыс орнайды).

.NET кітапханасында стандартты делегаттардың көптеген түрі сипатталған, олар оқиғаларды өңдеу механизмін жүзеге асыру үшін арналған. Осы кластардың көпшілігі бір ереже бойынша жазылған:

– делегат атауы оқиға атауымен басталып EventHandler жұрнағымен аяқталады;

– делегаттың екі формалды параметрі бар. Бірінші параметр оқиға көзін анықтайды және object типінде болады. Екінші параметр оқиға аргументін анықтайды және EventArgs типінде немесе одан туындайтын типте болады.

Делегатты жариялағанда осы ережелерді ұстанған жөн, мысалы:

```
public delegate void ZamenaEventHandler(object
sender, ZamenaEventArgs arge);
```

Осы мысалда біз Zamena оқиғасын өңдеу үшін делегатты құрамыз, оқиға объектте – оқиға көзінде пайда болатын өзгерістермен байланысты. Делегат сипаттамасында екі аргумент көрсетілген: оқиғаны туындатқан sender объекті және оқиғаға байланысты параметрлері бар ZamenaEventArgs типіндегі arge объекті.

Бізге массив индексін ғана беру керек болғандықтан, ZamenaEventArgs класын мына тәртіпте анықтау дұрыс болады:

```
public class ZamenaEventArgs : EventArgs
 {
 public int item;
 }
```

Оқиғаларды өңдеу механизмін жүзеге асырудың келесі кезеңі - оқиға көзі (sender) болатын класты анықтау. Осы әдіс арнайы жазылу пішімінде болады және көбінесе оған сәйкес делегаттың жазылу пішімімен анықталады. Әдіске қол жеткізу спецификаторын анықтағаннан кейін (эдетте public) event қызметтік сөзі жазылады, одан кейін делегат анықтайтын тип және оқиға атауы көрсетіледі.

```
public event ZamenaEventHandler Zamena;
```

Кластың толық сипаттмасы мана түрде болады:

```
class sobit
 {
 public event ZamenaEventHandler Zamena;
 public void prov(ZamenaEventArgs arge)
 {
  if (masi[arge.item] < 0)
  {
  Zamena(this,arge);
  }
 }
 }
```

Оқиғаларды өңдеу механизмін жүзеге асырудың келесі кезеңінде оқиғаларды қабылдаушы кластарды анықтау керек. Осы кластардың ерекшелігі – оларда оқиғалар өңдеуіштерінің әдістері болуы керек. Әдістердің жазылу пішімі сәйкес делегаттың жазылу пішіміне сай болуы керек. Мысалы:

```
 class zam1
 {
 public void OnZam1(object sender, ZamenaEventArgs
e)
 {
  masi[e.item] = masi[e.item] * (-1);
 }
 }
```

Оқиғалар өңдеуіштерін тіркеу оқиғалар өңдеуіштерінің кластарының (кластар емес) объекттері үшін жүргізу керек. Ол тек қана қосымшаның жұмысы кезінде ғана мүмкін – объекттер (айнымалылыр) қосымшаның жұмысы кезінде ғана құрылады, мысалы, форманың инициализациясы барысында:

```
public Form1()
 {
 InitializeComponent();
 zam1 z1 = new zam1();
 zam2 z2 = new zam2();
 so.Zamena += z1.OnZam1;
 so.Zamena += z2.OnZam2;
 }
```

Қосымша жұмысын көрсету үшін екі батырма қолданылады – «Массивті құрк» және «Массивті тексеру және оқиғаларды іске қосу».

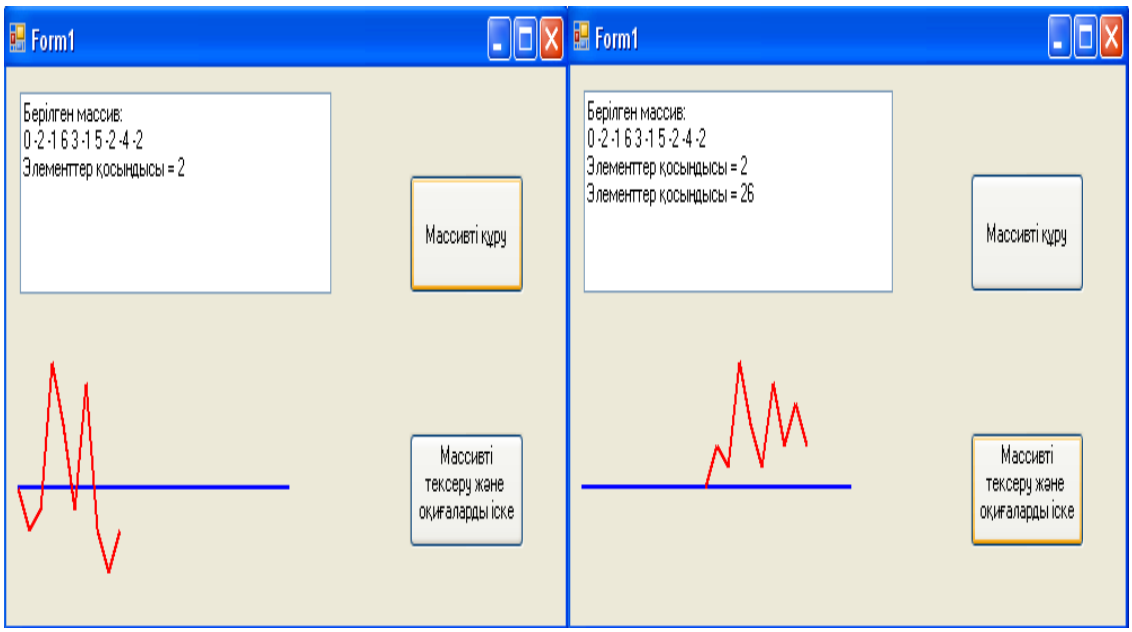
Бағдарлама коды:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
namespace WindowsFormsApplication1
{

 public partial class Form1 : Form
 {
 public delegate void ZamenaEventHandler(object sender,
ZamenaEventArgs arge);
 public int cob=0;
 public static int sum = 0;
 public static int[] masi = new int[10];
 public class ZamenaEventArgs : EventArgs
 {
  public int item;

 }
 class sobit
 {
  public event ZamenaEventHandler Zamena;
  public void prov(ZamenaEventArgs arge)
  {
   if (masi[arge.item] < 0)
   {
   Zamena(this,arge);
   }
  }
 }
 sobit so = new sobit();
 class zam1
```

```
      {
       public void OnZam1(object sender, ZamenaEventArgs e)
       {
       masi[e.item] = masi[e.item] * (-1);
       }
      }
      class zam2
      {
       public void OnZam2(object sender, ZamenaEventArgs e)
       {
       //if (masi[e.item] > 0) //оқиғаларды өңдеу кезектілігін
тексеру
        sum = sum + 2 * masi[e.item];
       }
      }
      public Form1()
      {
       InitializeComponent();
       zam1 z1 = new zam1();
       zam2 z2 = new zam2();
       so.Zamena += z1.OnZam1;
       so.Zamena += z2.OnZam2;
      }
      private void button1_Click(object sender, EventArgs e)
      {
       cob = 0; sum = 0;
       string ss="";
       Random rnd = new Random();
       for(int i=0;i<10;i++)
       {
       masi[i] = rnd.Next() % 15 - 5;
       sum = sum + masi[i];
       ss = ss + masi[i].ToString() + " ";
       }
       textBox1.AppendText("Берілген массив: \r\n");
       textBox1.AppendText(ss + "\r\n");
       textBox1.AppendText("Элементтер қосындысы = " +
sum.ToString() + "\r\n");
       this.Invalidate();
      }
      private void button2_Click(object sender, EventArgs e)
      {
       cob=1;
       ZamenaEventArgs zz = new ZamenaEventArgs();
       for (int i = 0; i < 10; i++)
       {
       zz.item = i;
       so.prov(zz);
       }
       textBox1.AppendText("Элементтер қосындысы = " +
sum.ToString() + "\r\n");
       this.Invalidate();
      }
```

```
        private void Form1_Paint(object sender, PaintEventArgs e)
        { Graphics g = e.Graphics;
         g.DrawLine(new Pen(Brushes.Blue, 2), 10, 200, 250, 200);
         Pen myPen = new Pen(Color.Red, 2);
         if (cob == 0)
         {
         for (int i = 1; i < 10; i++)
          g.DrawLine(myPen, i * 10, (200 - masi[i - 1] * 10), (i +
1) * 10, (200 - masi[i] * 10));

         }
         if (cob == 1)
         { for (int i = 1; i < 10; i++)
          g.DrawLine(myPen, i * 10 + 110, (200 - masi[i - 1] * 10),
(i + 1) * 10 + 110, (200 - masi[i] * 10));
         }
        }
        }
       }
```

Қосымшаның жұмысы 14.3-суретте көрсетілген.



14.3-сурет – Оқиғаларды өңдеу бойынша қосымша жұмысы

14.5 Өзін-өзі тексеру сұрақтары

1 Оқиға ұғымы.

2 Пайдаланушы әрекеттеріне арналған, басқару элементтерінің оқиғалар өңдеуіштерінің дайындамалары қай жерде орналасқан?

3 Кластың оқиғалары әдетте қалай аталады?

4 Формадағы батырманы екі рет шерткенде не орындалады?

5 Барлық хабарлама өңдеуіштерінің бірінші формалды параметрі нені анықтайды?

6 Хабарлама өңдеуішінің екінші формалды параметрі нені анықтайды?

7 Формада орналасқан басқару элементтерінің оқиғалары қай жерде көрсетіледі?

8 Кластың барлық оқиғаларын қалай біріктіруге болады?

9 Форманың көпадресті делегатында оқиғаларды қосу қай жерде орындалады?

10 Форманың көпадресті делегаты қай жерде жарияланады?