3-лекция

C# тілінің операторлары

Сұрақтар:

1. Жалпы мәліметтер
2. Тармақталу операторы
3. Ауыстырғыш оператор
4. Циклдік операторлар
5. C# тіліндегі жиымдар

5.1. Бір өлшемді жиымдар

5.2. Төртбұрышты жиымдар

5.3. Сатылы жиымдар

1. Жалпы мәліметтер

Кез келген программаны алгоритм блоктарының өзара байланысуына қарай үш түрлі басқару құрылымынан жасауға болады. Оларды құрылымдық программалаудың базалық құрастырғыштары (конструкциялары) деп атайды.

Бірнеше операторлардың тізбектей орындалуынан тұратын құрастырғыш *реттік* (сызықтық) деп аталады. Қандай да бір шарттың орындалуына тәуелді құрастырғышты *тармақталу* деп атаймыз. *Цикл* операторлар тізбегінің бірнеше рет қайталап орындалуын білдіреді.

Оператор – тілдің қарапайым сөйлемі, ол белгілі бір әрекет немесе амал орындап, **;** таңбасымен аяқталады.

Сызықтық (реттік) құрылым бірінен кейін бірі орындалып тізбектеле орналасқан бір-неше операторлардан тұрады.

Тармақты – шартқа байланысты екі оператордың бірінің орындалуы

Цикл – операторлар бөлігінің бірнеше рет қайталана орындалуы.

Кез келген нүктелі үтірмен аяқталатын өрнек оператор болып саналады. Ол көбінесе берілген өрнек бойынша есептеу ісін атқарады. Бос оператор да өрнектің бір түрі болып табылады.

Мысалдар:

```
i++;                  // инкремент операциясы
a *= b+c;             // көбейтіп меншіктеу амалы
fun(i, k);            // функцияны шақыру орындалады
while (true);         // бос операторлардан цикл (шексіз)
```

Блок немесе құрама оператор – бұл жүйелі жақшалармен шектелген сипаттамалар мен операторлар тізбегі {…}. Блок компилятор үшін бір оператор сияқты орындалады, ол синтаксис бойынша бір оператор болғанмен, алгоритм бойынша бірнеше операторлардан тұрады.

Басқару операторлары

Бұлар программадағы операторлардың орындалу реттілігін анықтайды да, алгоритмдерді жүзеге асырудың негізгі құралы болып табылады.

Басқару операторларының түрлері (категориялары – санаттары):

1. Таңдау операторлары, олар мынадай түйінді сөздер арқылы енгізіледі: if, if ... else ..., switch.
2. Циклдік (қадамдық – итеративтік) операторлар, while, do ... while, for, foreach түйінді сөздері арқылы енгізіледі.
3. Көшу (ауысу) операторлары, goto, break, continue түйінді сөздері арқылы енгізіледі.

2. ТАҢДАУ (ТАРМАҚТАЛУ) ОПЕРАТОРЫ

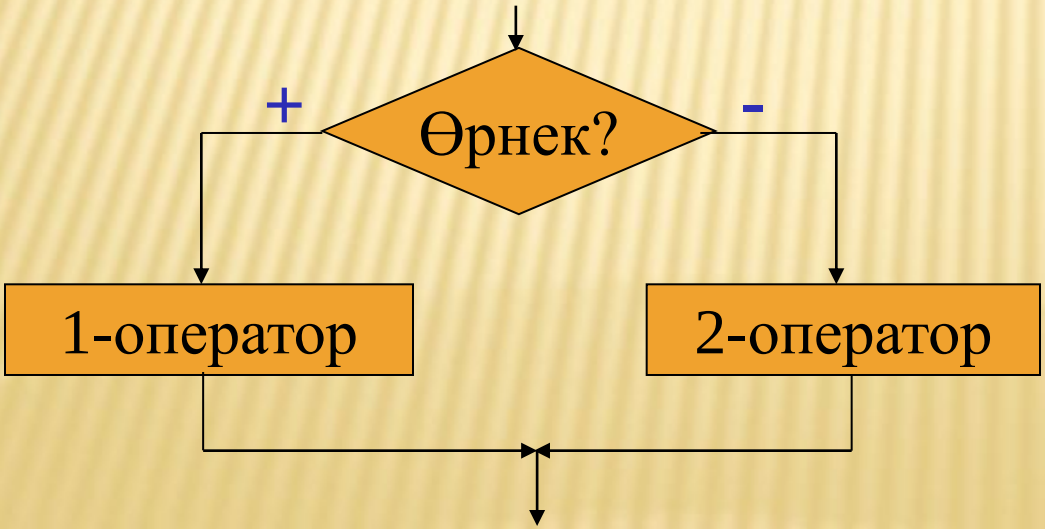
Тармақталу операторлары: **if, else**
және ауыстырғыш оператор: **switch**

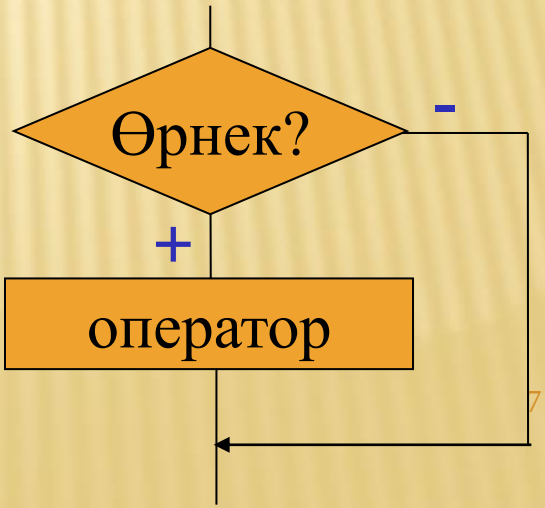
if шартты операторы есептеу жолының екі бағытта тармақталуын жүзеге асырады.

Оператор форматы:

if (өрнек) 1-оператор; [else 2-оператор;]

Шартты операторлардың құрылымдық схемасы:





if және else конструкциялары С тіліндегідей түрде қабатталып та қолданыла береді:

```
if (a == 5)
{
  …
}
else if (a < 5)
      {
        …
      }
      else
      {
        …
      }
```

```
if ( a < 0)   b = 1;                                    // 1

if ( a < b && ( a > d  || a == 0 ))   b++;
   else { b *= a; a = 0; }                              // 2

if ( a<b )
    if ( a < c ) m = a;
        else m = c;
    else
        if ( b < c ) m = b;
            else m = c;                                 // 3

 if ( b > a ) max = b;
    else max = a;                                       // 4
```

Санның тақ немесе жұп екенін анықтау

```
using System;
namespace  Chet_nechet
{
   class Class1
   {
      static void Main(string[] args)
      {   int k = Int32.Parse(Console.ReadLine());
           if  (k%2==0)
              {  Console.WriteLine("Четное число");    }
           else
              {  Console.WriteLine("Нечетное число"); }
          Console.ReadLine();
      }
   }
}
```

Қабатталған шартты операторлар

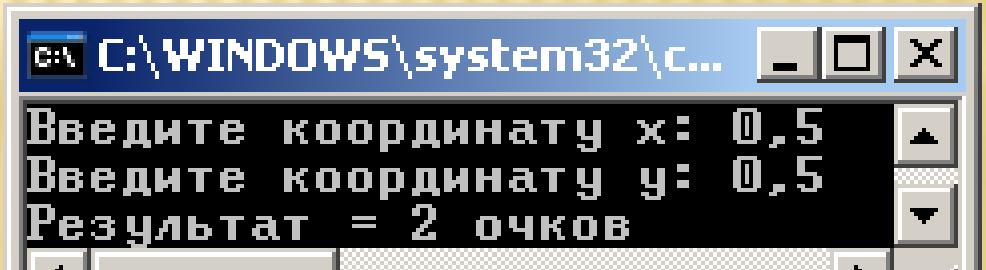
```
if  ((age > 0) && (age <7))  period = 1;
        else  if ((age >= 7)   && (age <17)) period =
2;
        else  if ((age >= 17) && (age <22)) period =
3;
        else  if ((age >= 22) && (age <27)) period =
4;
        else  if ((age >= 27) && (age <37)) period =
5;
    else  if ((age >= 37) && (age <57)) period = 6;
        else  period = 7;
```

1 : child
2 : schoolboy
3 : student
4 : junior researcher
5 : senior researcher
6 : professor
  : не определен

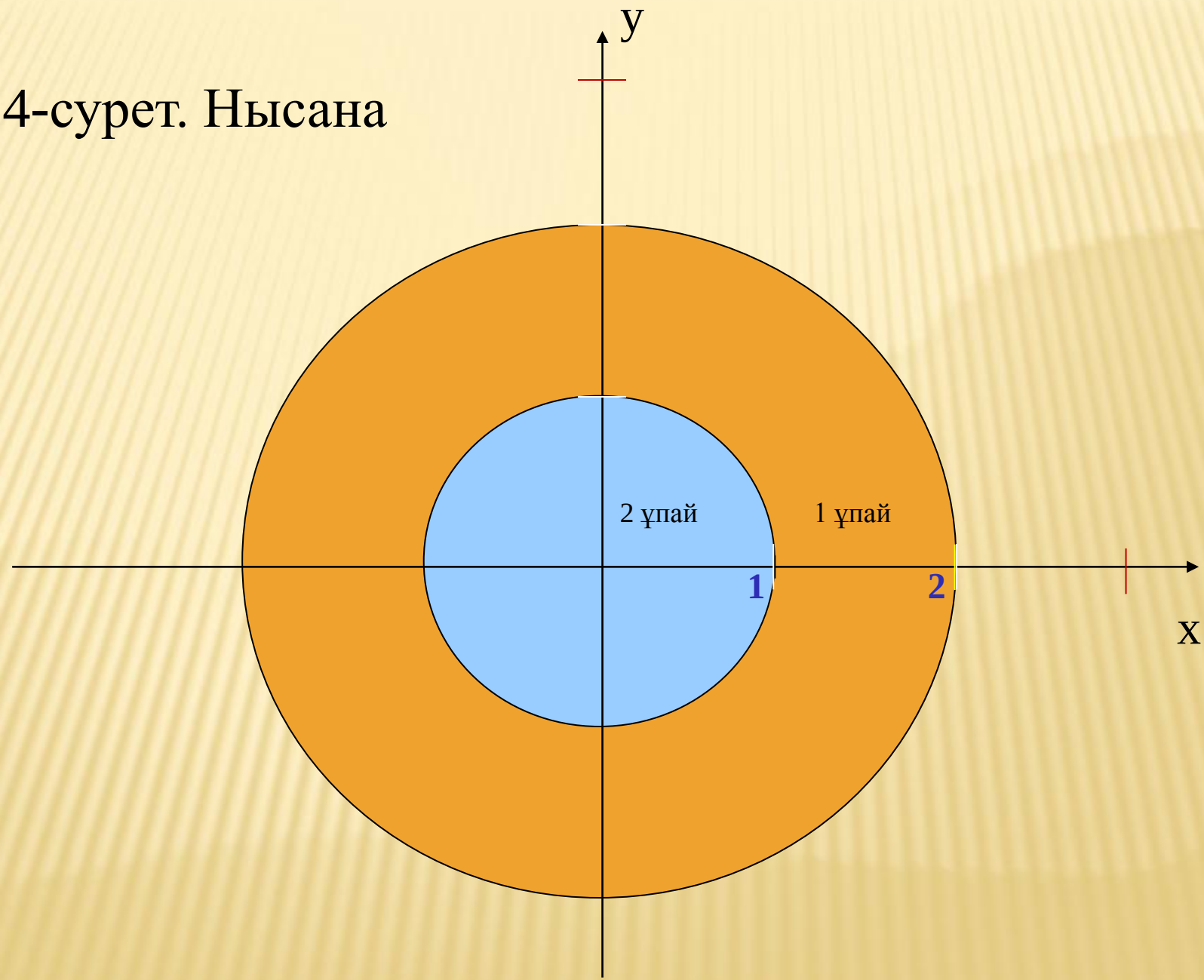
```
switch (period)
        {       case 1:
                        status = "child";
                        break;
                case 2:
                        status = "schoolboy";
                        break;
                case 3:
                        status = "student";
                        break;
                case 4:
                        status = "junior researcher";
                        break;
                case 5:
                        status = "senior researcher";
                        break;
                case 6:
                        status = "professor";
                        break;
                default :
                        status = "не определен";
                        break;
        }
        Console.WriteLine("Имя = {0}, Возраст = {1}, Статус =
{2}",
                name, age, status);
```

Мысалы:

```
using System;
namespace Listing4_1
{  class Class1
   {  static void Main( )
      {  Console.WriteLine( "Введите координату  x: " );
         string buf = Console.ReadLine();
         double x = Convert.ToDouble( buf );
         Console.WriteLine( "Введите координату  y: " );
         buf = Console.ReadLine();
         double y = double.Parse( buf );
         int kol = 0;
         if       ( x * x + y * y < 1 ) kol = 2;
         else   if ( x * x + y * y < 4 ) kol = 1;
         Console.WriteLine( "Результат = {0} очков", kol );
      }
   }
}
```



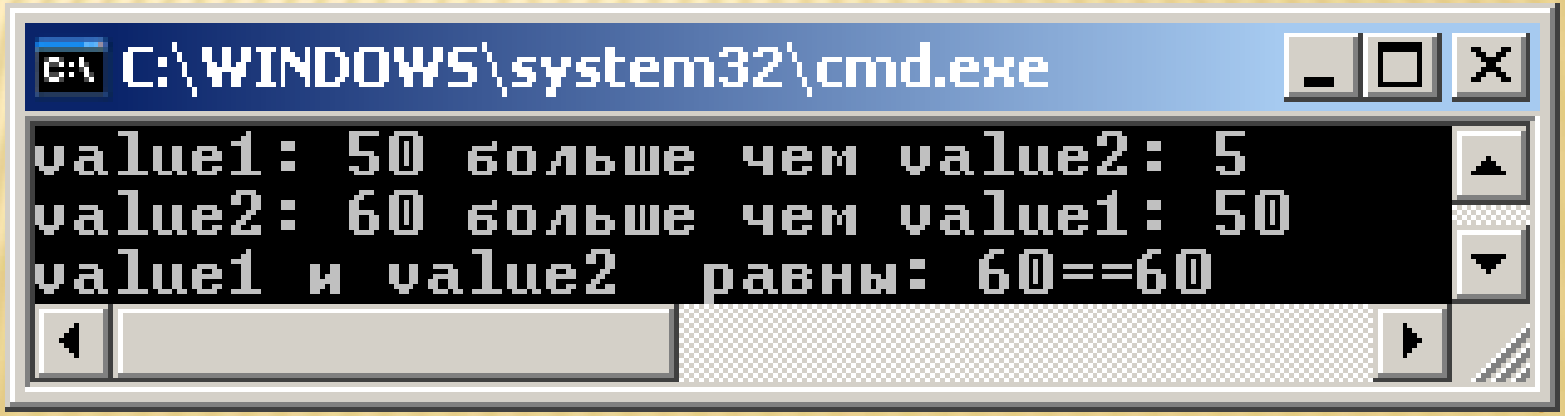
4-сурет. Нысана



Екінші мысал

```
using System;
namespace If_else1
{
  class Program
  { static void Main()
    { int value1 = 50;
      int value2 = 5;
      if ( value1 > value2 )
         Console.WriteLine ("value1: {0} больше чем
              value2: {1}",value1,value2);
      else
         Console.WriteLine("value2: {0} больше или
              равно value1: {1}",value2,value1);
```

```
        //устанавливаем второе значение больше первого
        value2 = 60;
        if ( value1 < value2 )
            Console.WriteLine("value2: {0}  больше чем
                    value1:  {1}", value2, value1);
    //делаем значения одинаковыми
        value1 = value2;
        if (value1 == value2)
            Console.WriteLine("value1 и
                    value2  равны: {0}=={1}", value1, value2);
    }
}
}
```



Қабатталған шартты операторлар

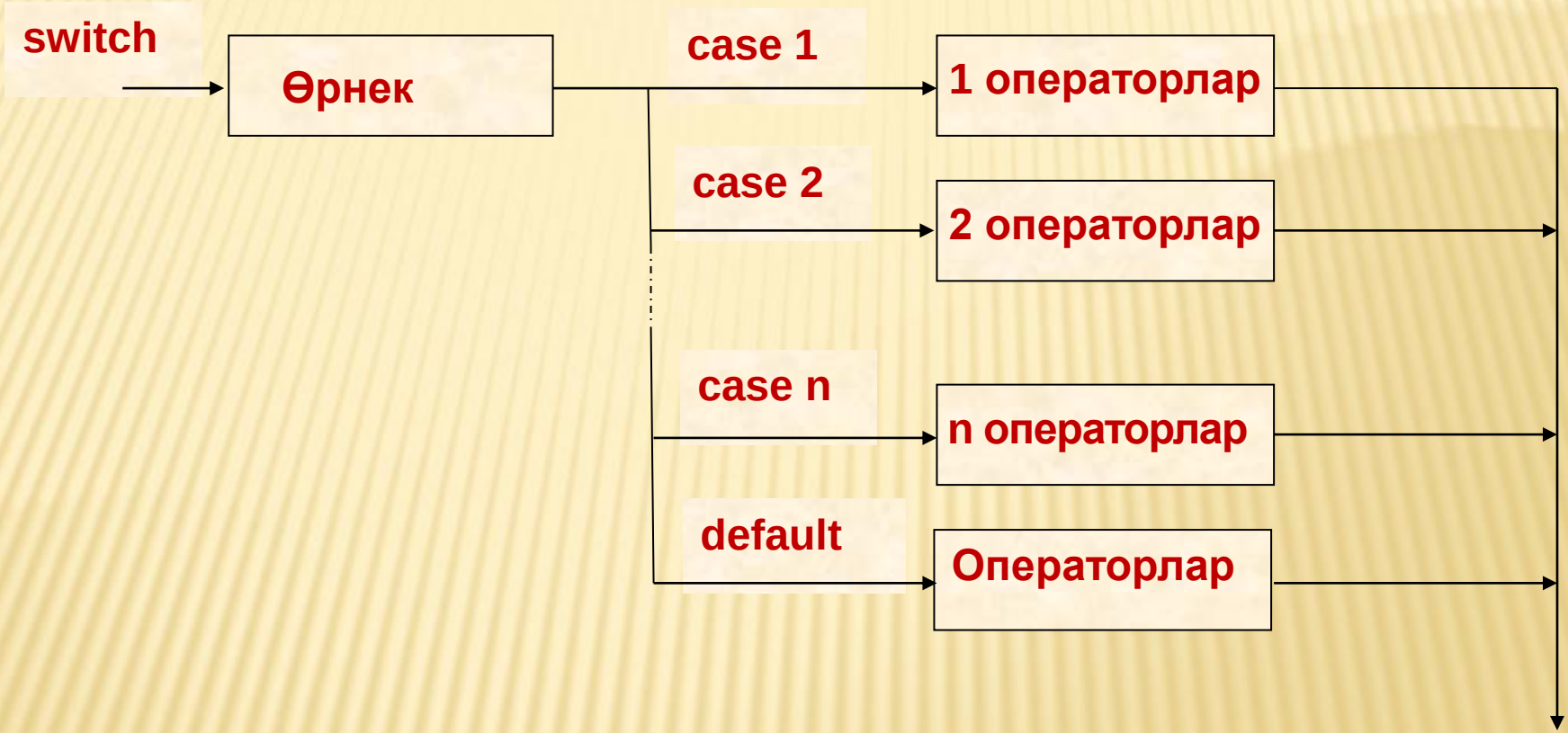
```
class Example4_3
{  public static void Main()
   {
      int reactorTemp = 1500;
      string emergencyValve = "закрыт";
      if (reactorTemp <= 1000)
         System.Console.WriteLine("Температура в реакторе
                                                нормальная");
      else
      {
         System.Console.WriteLine("Слишком высокая
                                       температура в реакторе!");
         if (emergencyValve == "закрыт")
            System.Console.WriteLine("Реактор в процессе
                                                плавления!");
      }
   }
}
```

3. Switch операторы

switch (ауыстырғыш) операторы есептеу процесін бірнеше тармаққа бөліп жібереді. Оның алгоритмдік схемасы келесі слайдта көрсетілген. Оператор форматы:

```
switch ( өрнек )
{
   case 1_тұрақты_өрнек: [1_операторлар_тізімі]
   case 2_тұрақты_өрнек: [2_операторлар_тізімі]
   ...
   case n_тұрақты_өрнек: [n_операторлар_тізімі]
   [default: операторлар ]
}
```

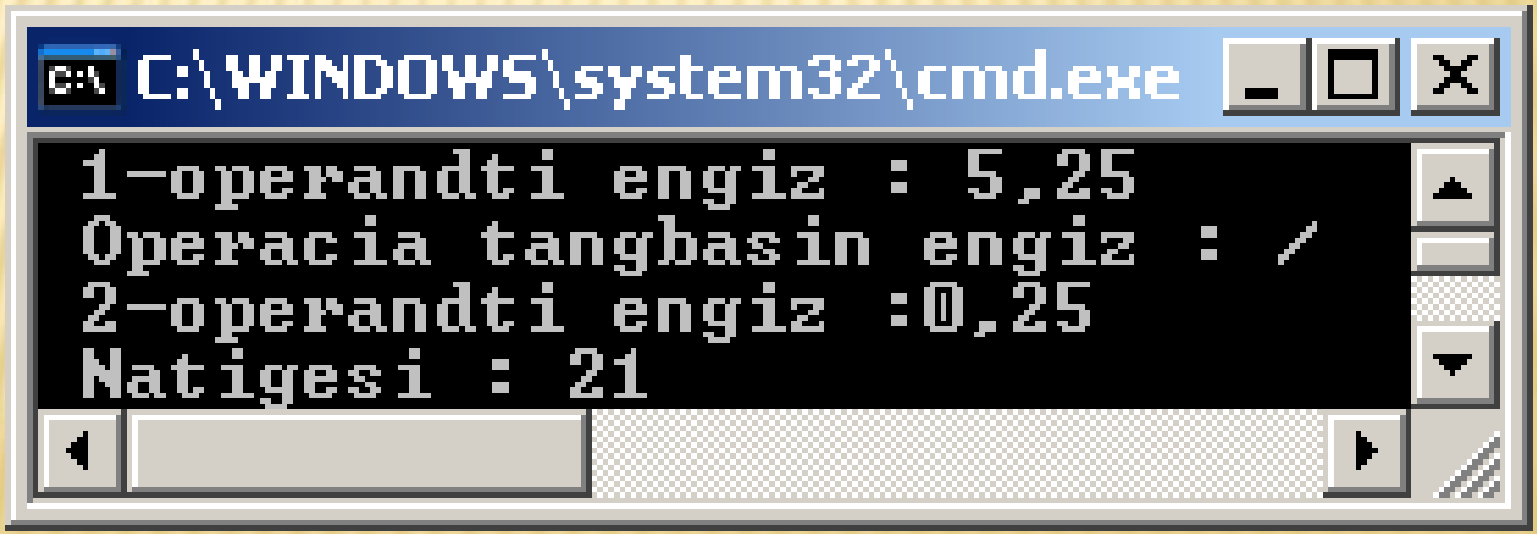
Оператордың құрылымдық схемасы суретте көрсетілген.



Қарапайым 4 амалды орындайтын калькулятор жұмысын атқаратын программа:

```
using System;
namespace ConsoleApplication1
{  class Class1
   {  static void Main( )
      {   string buf;
          double a, b, res;
          Console.WriteLine (" 1-operandti engiz : ")
          buf = Console.ReadLine( );
          a = double.Parse( buf );
          Console.WriteLine(  " Operacia tangbasin engiz : "  );
          char op = (char)Console.Read();
          Console.ReadLine():
          Console.WriteLine( " 2-operandti engiz :"  );
          buf = Console.ReadLine( );
          b = double.Parse( buf );
          bool  ok = true;
```

```
switch (op)
   { case '+': res = a + b; break;
case '-': res = a - b; break;
case '*': res = a * b; break;
case '/': res = a / b; break;
default : res = double.NAN; ok = false; break;
    }
if (ok) Console.WriteLine(" Natigesi : " + res); else
Console.WriteLine(" Belgisiz amal ");
}
 }
}
```



```
Console.WriteLine ("Введите оценку: ");
int k = Int32.Parse (Console.ReadLine());
Console.WriteLine (k.ToString());
switch (k)
{
   case 1:
   case 2:
          Console.WriteLine("Неудовлетворительно");
           break;
   case 3:
           Console.WriteLine("Удовлетворительно");
           break;
   case 4:
           Console.WriteLine("Хорошо");
           break;
   case 5:
           Console.WriteLine("Отлично");
            break;
   default:
            Console.WriteLine("Ошибка");
            break;
}
```

Case жолының бір тармағында немесе default операторлары соңында break міндетті түрде болуы тиіс. Келесі жолдарда қателер бар:

```
...
{
    case 1:
            Console.WriteLine("Совсем неудовлетворительно");
             //Ошибка! Тут пропущен break
    case 2:
             Console.WriteLine("Неудовлетворительно");
              break;
….
   default:
             Console.WriteLine("...");
             //Ошибка! Тут пропущен break
}
```

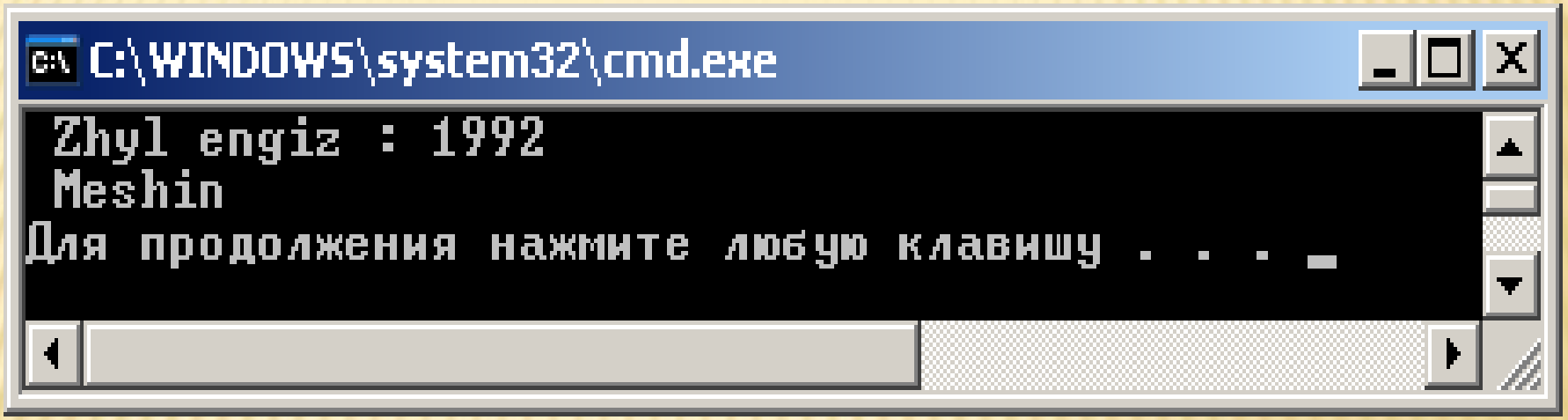
Қазақша жыл санау мысалы:

```
using System;
namespace Tyshkan
{
    class Class1
    {
        static  void  Main()
        {
            string  buf;
            int  gyl;
            Console.Write(" Zhyl engiz : ");
            buf = Console.ReadLine();
            gyl = int.Parse(buf);
```

```
switch (gyl % 12)
        {  case 0: Console.Write(" Meshin ");  break;
           case 1: Console.Write(" Tauyk ");    break;
           case 2: Console.Write(" It ");           break;
           case 3: Console.Write(" Dongyz ");  break;
           case 4: Console.Write(" Tushkan "); break;
           case 5: Console.Write(" Syir ");         break;
           case 6: Console.Write(" Barys ");     break;
           case 7: Console.Write(" Koian ");     break;
           case 8: Console.Write(" Ulu ");          break;
           case 9: Console.Write(" Zhylan ");    break;
           case 10: Console.Write(" Zhylky ");  break;
           case 11: Console.Write(" Koy ");       break;

           default: Console.Write(" Butin natural san
                                   engizu kerek ");      break;
      }   Console.WriteLine();
    } } }
```

Программа нәтижесі:



4. Циклдік операторлар

C# тілінде үш циклдік оператор бар, олардың жазылуы:

while (шарт-өрнек) {операторлар} шарты алдында

do {операторлар} while (шарт-өрнек); шарты соңында

for (инициализация; шарт-өрнек; модификация)
{ операторлар }

foreach (тип айнымалы-аты in шарт-өрнек)
{ операторлар }

C# тілінде 4 басқаруды беру операторы бар:

- ✓ goto шартсыз өту операторы;
- ✓ break циклдан шығу операторы;
- ✓ continue циклдың келесі итерациясына көшу операторы;
- ✓ return функциядан қайтару операторы.

goto шартсыз көшу операторының форматы:

goto <белгі>;

break циклдан шығу операторы цикл операторларының ішінде қолданылады немесе switch операторынан шығуды қамтамасыз етеді.

continue циклдың келесі итерациясына көшу операторы, цикл тұлғасының аяғына дейінгі операторларды бос жіберіп, басқаруды келесі итерацияға (қадамға) береді.

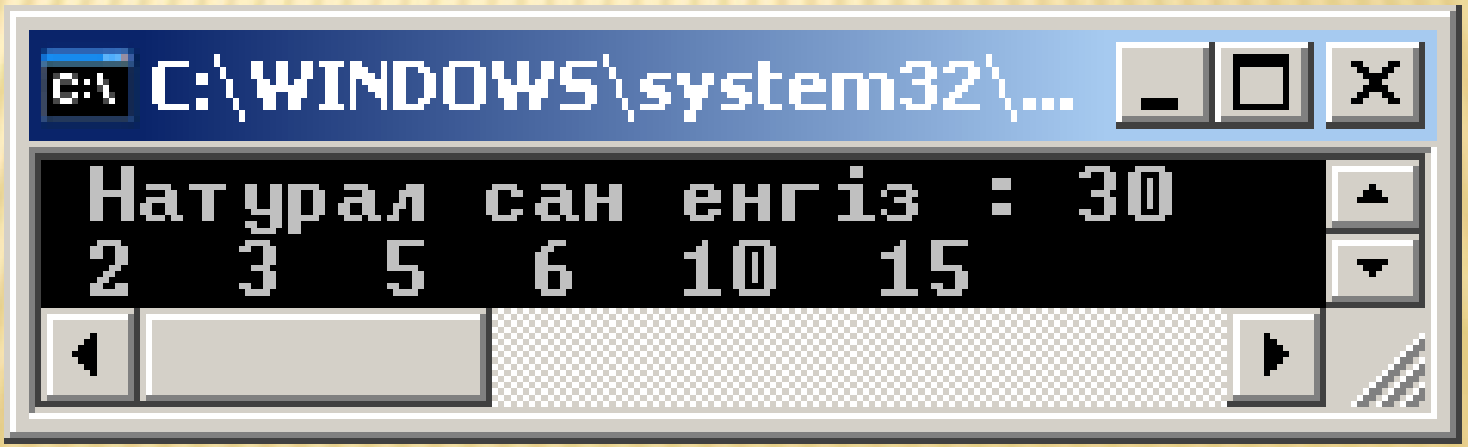
return функциядан қайтару операторы – ол функцияның орындалуын аяқтап, басқаруды шақыру нүктесіне береді.

While (әзірше) циклі
(алғы шартты цикл)

Келесі мысалда енгізілген **num** бүтін санының барлық бөлгіштері анықталады.

```
// Берілген оң бүтін санның бөлгіштерін табу
using System;
namespace While_delitely
{
    class Class1
    {
        static void Main()
        {  string buf;
            int  num, half, div;
            Console.Write(" Натурал сан енгіз : ");
            buf = Console.ReadLine(); // енгізу
            num = Convert.ToInt32(buf); // түрлендіру
```

```
        half = num / 2;
        div = 2;                // алғашқы бөлгіш
        while(div<=half)
        { if ((num % div)==0)
            Console.Write(" "+div);
          div++;
        } Console.WriteLine();    // күте тұру
      }
   }
}
```



Фибоначчи сандарын шығару (<50)

```
using System;
class Example4_9
{  public static void Main()
   {  int  oldNumber = 1;    // алғашқы екі сан 1
      int  currentNumber = 1;
      int  nextNumber;
      Console.Write(currentNumber + " ");
      while (currentNumber < 50)
      {  Console.Write(currentNumber + " ");
         nextNumber = currentNumber + oldNumber;
         oldNumber = currentNumber;
         currentNumber = nextNumber;
      }
   }
}
```

Do … while циклі (дейін)
(соңғы шартты цикл)

Мысал. Бұл программада латынның y (yes) әрпін енгізгенше сөз немесе әріп енгізуді сұрайтын программа келтірілген.

```
using System;
class Classl
 {  static void Main()
        {  char answer;
           do
           {  Console.WriteLine("Бес коясыз ба, агай?");
              answer = (char)Console.Read();
              Console.ReadLine();
           } while (answer != 'y');
        }
    }
}
```

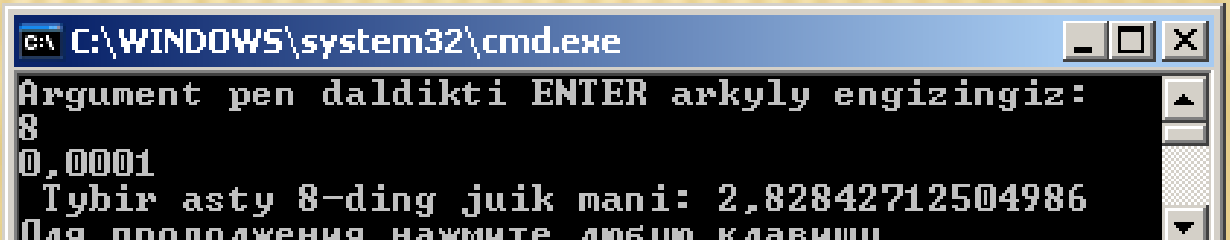
Мысал. Бұл программада енгізілген нақты аргумент – **x**-тің квадрат түбірін берілген дәлдікпен – eps жуық шамамен итерациялық формула арқылы анықтаймыз:

$$y_n = (y_{n-1} + x/y_{n-l}),$$

мұндағы y_{n-l} – түбірдің алдыңғы жуық мәні (есептеу алдында бұл мән кез келген оң сан ретінде таңдалады), y_n – түбірдің келесі табылған жуық мәні. Есептеу процесі түбірдің анықталған екі жуық мәндері айырмасының абсолюттік мәні берілген дәлдіктен төмен болған сәтте тоқталады.

Абсолюттік мәнді табу үшін стандартты Math.Abs() функциясы қолданылады.

```
using System;
namespace Tubir_tabu
{  class Program
    {  static void Main( )
        { double x, eps;          // аргумент пен далдік
          double Yp, Y = 1;  // тубірдін алдынгы, келесі жуык мані
          Console.WriteLine("Argument  pen daldikti ENTER
                                        arkyly engizingiz: ");
          x = Double.Parse(Console.ReadLine());
          eps = Double.Parse(Console.ReadLine());
          do
           { Yp = Y;
             Y = (Yp + x/Yp)/2;
            } while (Math.Abs(Y - Yp) >= eps);
          Console.WriteLine(" Tybir asty {0}-ding juik mani: {1}", x, Y);
        }
    }
}
```

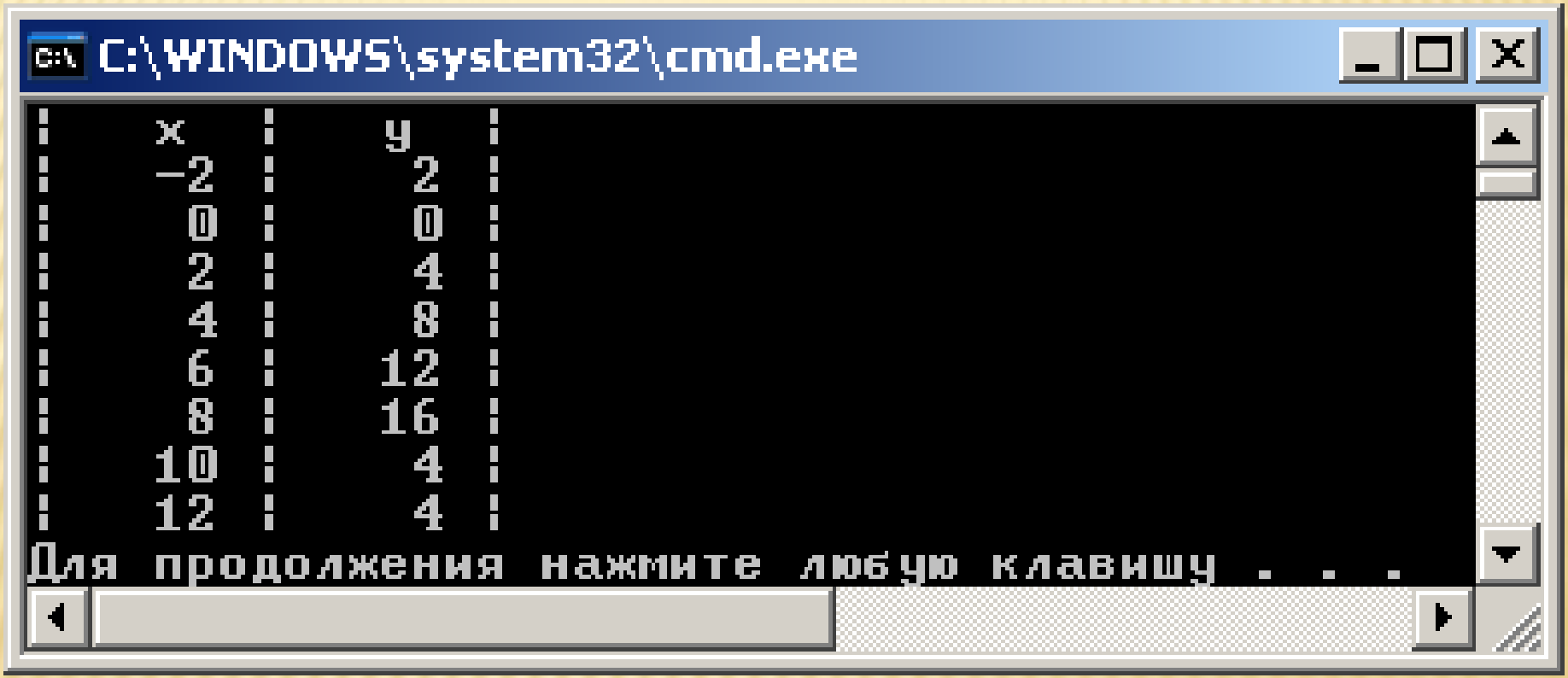


Параметрлі цикл for (үшін)

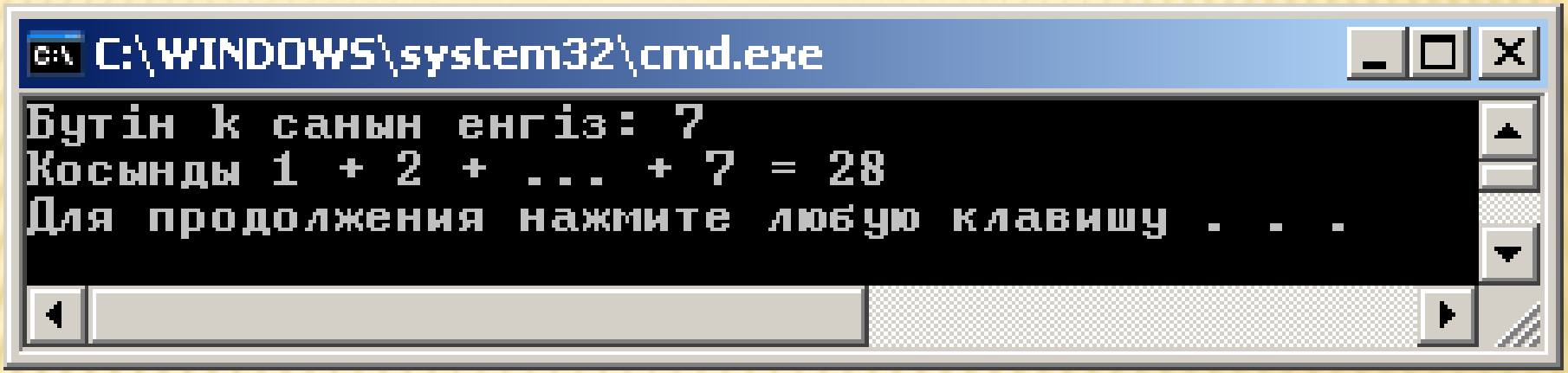
Мысал. Берілген y функциясының мәндерін оның аргументі Xn-нен Xk-ға дейін dX қадаммен өзгерген кезде анықтау программасы:

$$y = \begin{cases} t, & x < 0 \\ tx, & 0 \le x < 10 \\ 2t, & x \ge 10 \end{cases}$$

```
using System;
class Class1
   {  static void Main( )
      {
        double Xn = -2, Xk = 12, dX = 2, t = 2, y;
        Console.WriteLine( "|   x  |   y  |" );
        for ( double x = Xn; x <= Xk; x += dX )
        {  y = t;
           if ( x >= 0 && x < 10 ) y = t * x;
           if (x >= 10) y = 2 * t;
           Console.WriteLine( "| {0,4} | {1,4} |", x, y );
        }
      }
   }
```


C:\WINDOWS\system32\cmd.exe
x	y
-2	2
0	0
2	4
4	8
6	12
8	16
10	4
12	4
Для продолжения нажмите любую клавишу . . .

```
using System; // 1-ден k-ға дейінгі сандар қосындысы
namespace For1
{  class Program
    {
       static void Main(string [ ]  args)
        {
           Console.Write("Бутін k санын енгіз: ");
           int k = Int32.Parse(Console.ReadLine());
           int sum = 0; // k-га дейінгі сандар косындысы
           for (int i = 1; i <= k; i++)
           {
               sum += i;
           }
           Console.WriteLine("Косынды 1 + 2 + ...
                                        + {0} = {1}", k, sum);
       }
   }
}
```


C:\WINDOWS\system32\cmd.exe
Бутін k санын енгіз: 7
Косынды 1 + 2 + ... + 7 = 28
Для продолжения нажмите любую клавишу . . .

5. C# тіліндегі жиымдар (массивтер)

C# тіліндегі жиымдар C/C++ тілдерінен өзгешелеу болып келеді. Бірден мысалдар келтірейік.

```
int[ ] k;                  //k  жиымын анықтау
k=new int [3];    //3 бүтін саннан тұратын жиым
k[0]=-5; k[1]=4; k[2]=55;   //Жиым элементтері
        //Жиымның 3-элементін экранға шығару
Console.WriteLine(k[2].ToString());
```

Бұлар түсінікті шығар. Жиым былай анықталады:

```
int[ ] k;
```

Мынадай нұсқалар дұрыс емес:

```
int k[ ];    // Қате!
int k[3];  // Қате!
int [3] k;  // Қате!
```

Жиымды сипаттау үшін мәлімет типінен соң, бос тік жақшалар қойылады да, оның аты жазылады:

float[ ] ar;

ar айнымалысының типі — жылжымалы нүктелі сандар жиымы (нақты сан), ал негізінде **ar** — бұл нұсқауыш. C# тілінде жиым сілтемелік тип (reference type) болып табылады. Сөз тіркесі де осы типке жатады.

ar айнымалысының алғашқы анықталған мәні — **null**. Бұл жиымға компьютер жадынан орын бөлу үшін, **new** операторы арқылы жиымдағы элементтер санын көрсету керек:

ar = new float[3] ;

Жоғарыдағы екі операторды біріктіріп жазуға да болады:

float[ ] ar = new float[3];

Бұған қоса, жиымды сипаттау кезінде оны бірден инициалдауға болады:

float[ ] ar = new float[3] { 3.14, 2.17, 100 };

Инициалдау мәндерінің саны жиымда көрсетілген элементтер санымен сәйкес болуы тиіс.

Инициалдау кезінде жиым элементтері санын бермеуге де болады:

float[ ] ar = new float[ ] { 3.14, 2.17, 100 };

new сөзін жазбауға да рұхсат етілген:

float[ ] ar = { 3.14, 2.17, 100 };

Кейіннен программада **ar** айнымалысына басқа өлшемдегі **float** типін меншіктеуге де болады:

ar = new float[5];

Мұнда **float** типті 5 мәнді сақтауға арналған жады бөлінеді, алғашқыда олардың бәрінің де мәні 0-ге тең деп саналады.

Алдыңғы бөлінген **float** типті 3 мәнді сақтауға арналған жады қоқыс ретінде (garbage collection) өздігінен босатылады, өйткені C# тілінде delete операторы жоқ.

Сөз тіркестеріндегі сияқты жиым элементтерінің санын мынадай өрнекпен анықтауға болады:

ar.Length;

C# тілінде бір өлшемді жиымдар және одан өзге жиымдардың жиымы болып табылатын екі (одан да көп) өлшемді төртбұрышты және сатылы (тураланбаған - jagged) жиымдар құру мүмкіндік бар.

Бір өлшемді жиымдар программаларда жиі қолданылады. Оларды сипаттау нұсқалары:

типі[ ] аты;

типі[ ] аты = new тип [өлшемі];

типі[ ] аты = { инициалдау_тізімі };

типі[ ] аты = new типі [ ] { инициалдау_тізімі };

типі[ ] аты = new типі [өлшемі] { инициалдау_тізімі };

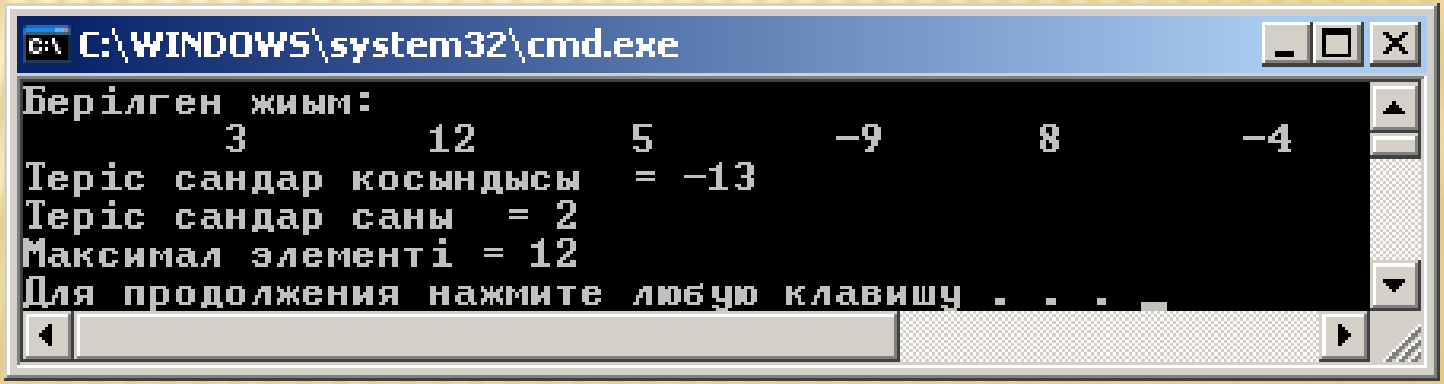
Бұларға бір-бір мысал:

```
int[ ]  a;        // 1   әзірше элементтері жоқ
int[ ] b = new int[4];  // 2  элементтері  0-ге тең
int[ ] c = { 61, 2,  5,  -9 };  // 3   new жазбаса да болады
int[ ] d = new int[ ] { 61, 2, 5,  -9 };  // 4      өлшемі
                                          //        есептеледі
int[ ] e = new int[4] { 61, 2, 5, -9 };  // 5      өлшемін
                                  // артығымен сипаттау
```

6 бүтін саннан тұратын **a** жиымының теріс элемент-терінің қосындысы мен санын және ең үлкен элемен-тін анықтайтын программа құрамыз.

```
using System;
namespace Listing6_1
{ class Class1
   { static void Main( )
      {  const int n = 6;
 int[ ] a = new int[n] { 3, 12, 5, -9, 8, -4 };
 Console.WriteLine("Берілген жиым:" );
 for ( int i = 0; i < n; ++i )
     Console.Write( "\t" + a[i] );
 Console.WriteLine( );
 long sum = 0; // теріс элементтері қосындысы
 int num = 0; // теріс элементтері  саны
```

```
for ( int i =0; i < n; ++i )
        if ( a[i] < 0 )
         {  sum += a[i];
            ++num;
         }
Console.WriteLine(" Терiс сандар қосындысы = " + sum );
Console.WriteLine(" Терiс сандар саны = " + num );
int max = a[0];       // максимал элемент
for ( int i = 1; i < n; ++i )
    if ( a[i] > max ) max = a[i];
Console.WriteLine( "Максимал элементi = " + max );
    }
  }
}
```

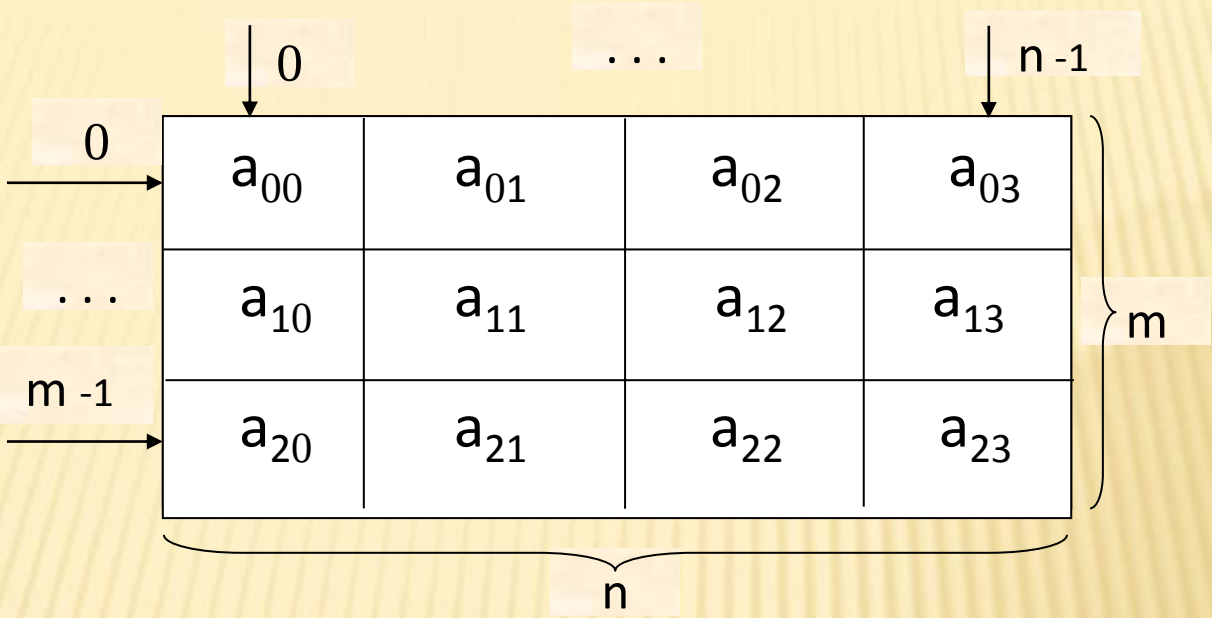


Төртбұрышты жиымдар ретінде программаларда екі өлшемді массивтер жиі қолданылады. Оларды сипаттау нұсқалары:

```
типі[,] аты;
типі[,] аты = new тип [1_өлшем, 2_өлшем];
типі[,] аты = { инициалдау_тізімі };
типі[,] аты = new типі [,] { инициалдау_тізімі };
типі[,] аты = new типі [1_өлшем, 2_өлшем]
                                { инициалдау_тізімі };
```

Бұларға бір-бір мысал:

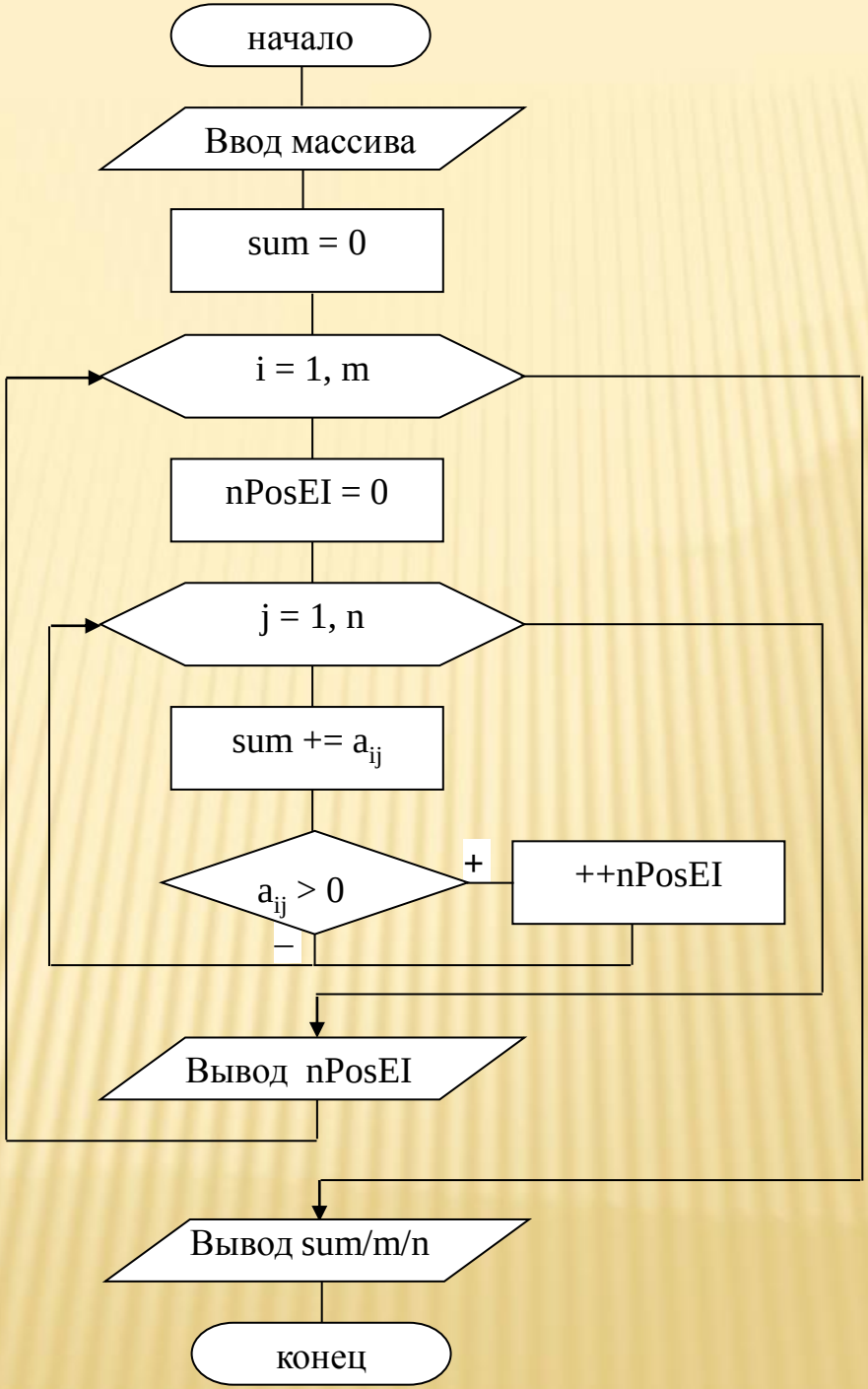
```
int[,]  a;       //  әзірше элементтері жоқ
int[,] b = new int[2,3];  //  элементтері  0-ге тең
int[,] c = {{1, 2, 3},{4, 5, 6 };  // new жазбаса да болады
int[,] d = new int[,] {{1, 2, 3},{4, 5, 6 };   //  өлшемі
                                    //  есептеледі
int[,] e = new int[2,3] {{1, 2, 3},{4, 5, 6 };  // өлшемін
                                // артығымен сипаттау
```



m жолдан және n бағанадан тұратын төрт бұрышты жиым (матрица),

мұнда $a_{3,4}$

Осы жиым элементтерінің арифметикалық ортасы мен әр жолдағы оң элементтері санын анықтайық.


начало
Ввод массива
sum = 0
i = 1, m
nPosEI = 0
j = 1, n
sum += a_ij
a_ij > 0
+
++nPosEI
–
Вывод nPosEI
Вывод sum/m/n
конец

```
using System;
namespace Listing6_2
{  class Classl
   { static void Main( )
     { const int m = 3, n = 4;
       int[,] a = new int[m, n] {
                                  { 2,-2, 8, 9 },
                                  {-4,-5, 6,-2 },
                                  { 7, 0, 1, 1 }
                                };
       Console.WriteLine("Исходный массив:");
       for (int i = 0; i < m; ++i)
         {
           for (int j = 0; j < n; ++j)
               Console.Write("\t" + a[i,j]);
           Console.WriteLine( );
         }
```

```
        double sum = 0;
        int nPosE1;
        for (int i = 0; i < m; ++i)
          {
           nPosE1 = 0;
           for (int j = 0; j < n; ++j)
           {
             sum += a[i, j];
             if (a[i, j] > 0) ++nPosE1;
           }
           Console.WriteLine("В строке {0} {1} положит-х
                                        элементов", i, nPosE1);
         }
         Console.WriteLine("Среднее арифметическое всех
                                       эл-тов: " + sum/m/n);
       }
    }
}
```

Сатылы жиымдар

Сатылы жиымдарда оның жолдарындағы элементтер саны бірдей болмайды.

Компьютер жадында сатылы жиым төрт-бұрышты жиымнан басқаша сақталады: *оның әр жолы элементтер саны әртүрлі болып келетін ішкі жиымдар түрінде болады.*

Оның сипатталуы:

```
типі [][] аты;
```

Жиымның әр сатысына жады тікелей элементтер санына қарай жеке-жеке бөлінеді.

Мысалы, 2-өлшемді 3 жолдан тұратын сатылы жиым ашу керек болсын:

```
int[ ][ ] a = new int [3][ ];
// Оның 0-жолын анықтайық (5 элемент болсын)
a[0]=new int[5];
// Оның 1-жолын анықтайық (3 элемент болсын)
a[1] = new int[3];
// Оның 2-жолын анықтайық (4 элемент болсын)
a[2] = new int[4];
```

Мұндағы **a[0]** , **a[1]** және **a[2]** аттары арқылы пайдаланылатын жеке-жеке жиымдар.

Былай да сипаттай отырып, жады бөлуге болады:

```
int[ ][ ] a = { new int [5], new int [3], new int [4]};
```

Сатылы жиым элементтерін жазу кезінде әр өлшем бір тік жақшалар ішінде орналасады:

a[1][2] **a[i][j]** **a[j][i]**

System.Array класы

C# жиымдарының барлығы да Array атты базалық класс негізінде құрылған, оның қолданушыға пайдалы бірнеше тәсілдері мен қасиеттері бар, солардың бірсыпырасын қарап шығайық.

Элементі	Түрі	Сипатталуы
Length	Қасиет	Жиым элементтерінің саны (барлық өлшемдері бойынша)
Rank	Қасиет	Жиым өлшемдерінің саны
Binary-Search	Статикалық тәсіл	Сұрыпталған жиымдағы екілік түрде іздеу
Clear	Статикалық тәсіл	Жиым элементтеріне келісім бойынша мән беру
Copy	Статикалық тәсіл	Бір жиым элементтерінің берілген диапазонын екінші жиымға көшіру
CopyTo	Тәсіл	Ағымдағы бірөлшемді жиым элементтерінің бәрін басқа жиымға көшіру

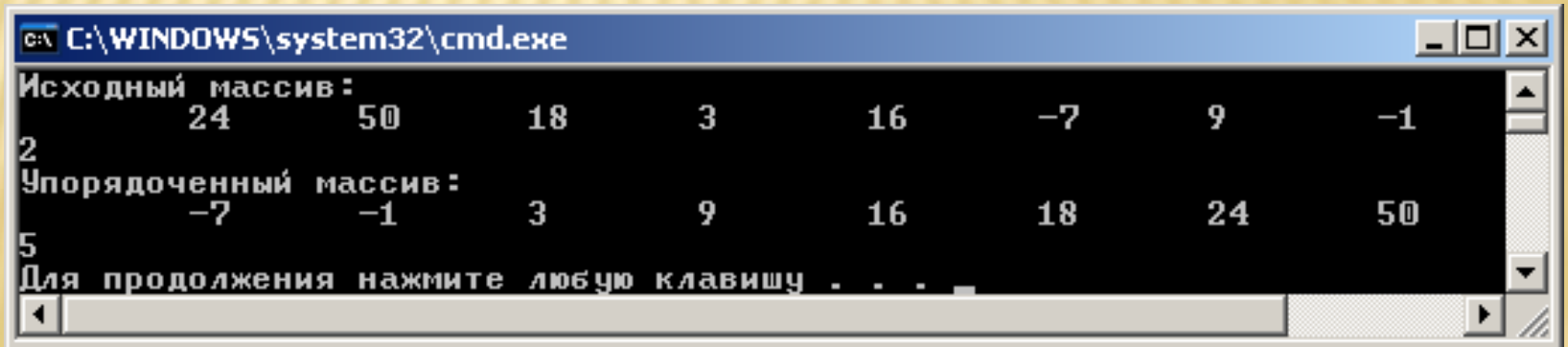
Элементі	Түрі	Сипатталуы
`GetValue`	Тәсіл	Жиым элементінің мәнін алу
`IndexOf`	Статикалық тәсіл	Бір өлшемді жиымдағы элементтің алғашқы кездесу орнын анықтау
`LastIndex-Of`	Статикалық тәсіл	Бір өлшемді жиымдағы элементтің соңғы кездесу орнын анықтау
`Reverse`	Статикалық тәсіл	Жиым элементтерін кері бағытта орналастыру
`SetValue`	Тәсіл	Жиым элементінің мәнін орнату
`Sort`	Статикалық тәсіл	Бір өлшемді жиымдағы элементтерді реттеу

Length қасиеті әртүрлі ұзындықтағы жиым-дарды өңдеу алгоритмін жүзеге асырады, мысалы, сатылы жиымдарды өңдеу. Жиым өлшемін беру орнына осыны қолдану жиым шегінен шығып кетуді болдырмайды.

Келесі программада бір өлшемді жиыммен жұмыс істеу кезіндегі Array класы элемент-терін қолдану мысалы көрсетілген.

```
using System;
namespace Listing6_3
{ class Classl
  {
   static void Main()
    {
     int[ ] a = {24, 50, 18, 3, 16, -7, 9, -1 };
     PrintArray( "Исходный массив:", a );
    Console.WriteLine( Array.IndexOf( a, 18 ) );
```

```
    Array.Sort(a);
    PrintArray( "Упорядоченный массив:“,a);
    Console.WriteLine(Array.BinarySearch(a, 18) );
  }
  public static void PrintArray( string header,
                                 int[] a )
  {
     Console.WriteLine( header );
     for ( int i = 0; i < a.Length; ++i  )
           Console.Write( "\t" + a[i] );
     Console.WriteLine();
  }
 }
}
```



Sort, **IndexOf** және **BinarySearch** тәсілдері статикалық тәсіл болып табылады, сол себепті оларды класс атын (экземпляр атын емес) көрсету арқылы пайдаланамыз да, жиым атын да береміз. Екілік іздеу тек реттелген жиымға ғана қолданылады, ол IndexOf тәсіліндегі сызықтық іздеуден жылдам орындалады.

Программада 18-ге тең мәні бар жиым элементін осы екі тәсілмен де іздеу жүргізіледі.

Class1 класында қосымша PrintArray статикалық тәсілі сипатталған, ол жиымды экранға шығару үшін қолданылады. Оған екі параметр беріледі: header тақырып жолы және жиым. Жиым элементтерінің саны тәсіл ішінде Length қасиеті арқылы анықталады.

Сонымен, бұл тәсілді кез келген бір өлшемді жиымды экранға шығару үшін пайдалануға болады екен.

PrintArray тәсілін бір өлшемдіден басқа жиымдарға қолдану үшін оның екінші Array параметрін сипаттау керек, бірақ мұнда жиым элементінің мәнін **GetValue** тәсілі көмегімен алу қажет, өйткені **Array** класы үшін индекс арқылы элементті анықтау ісі қарастырыл-маған.

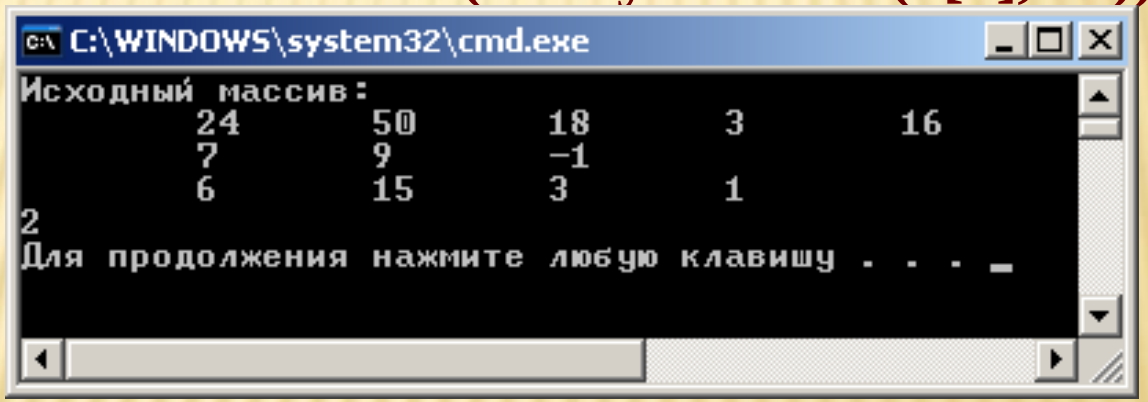
Жалпы түрде жиымды экранға шығару әрекеті былай атқарылады.

```
public static void PrintArray( string header.Array a )
{  Console.WriteLine( header );
   for ( int i = 0; i < a.Length; ++i )
         Console.Write( "\t" + a.GetValue(i) );
   Console.WriteLine();
}
```

Келесі программада Array класы элементтері сатылы жиым үшін қолданылған.

```
using System;
namespace ConsoleApplication1
{
    class Classl
    {
        static void Main( )
        {
            int[ ][ ] a = new int[3][ ];
            a[0] = new int[5] { 24, 50, 18, 3, 16 };
            a[1] = new int[3] { 7, 9, -1 };
            a[2] = new int[4] { 6, 15, 3, 1 };
```

```
        Console.WriteLine("Исходный массив:");
        for (int i = 0; i < a.Length; ++i)
        {
            for (int j = 0; j < a[i].Length; ++j)
                Console.Write("\t" + a[i][j]);
            Console.WriteLine();
        }
        Console.WriteLine(Array.IndexOf(a[0], 18));
    }
}
```



Цикл ішінде жолдар бойынша әрбір жиым ұзындығының қалай анықталғанына назар аудару керек.

Foreach циклі (әрбірі үшін)

Бұл оператор топталған мәліметтер элементтерін бір-бірден қарастыру үшін керек. Мұндай топқа жиым (массив) жатады. Foreach операторының қолайлылығы – жиым элементінің санын анықтап, оның индексін бір-бірден арттырып отырмаймыз, тек жиымның барлық элементтерін қарастыру қажеттілігін көрсетеміз. Операторлардың жазылуы:

foreach (типі аты in өрнек)
{ операторлар }

Мұндағы **аты** циклдің жергілікті айнымалысын көрсетеді, ол **өрнек** түрінде аты берілген жиымның барлық элементтерінің мәнін біртіндеп қабылдайды. Цикл ішінде оның айнымалысымен амалдар орындалады.

Мысалы, мынадай жиым берілген болсын:

```
int [ ]  a = {24, 50, 18, 3, 16, -7, 9, -1};
```

Жиым элементтерін экранға шығару былай орындалады:

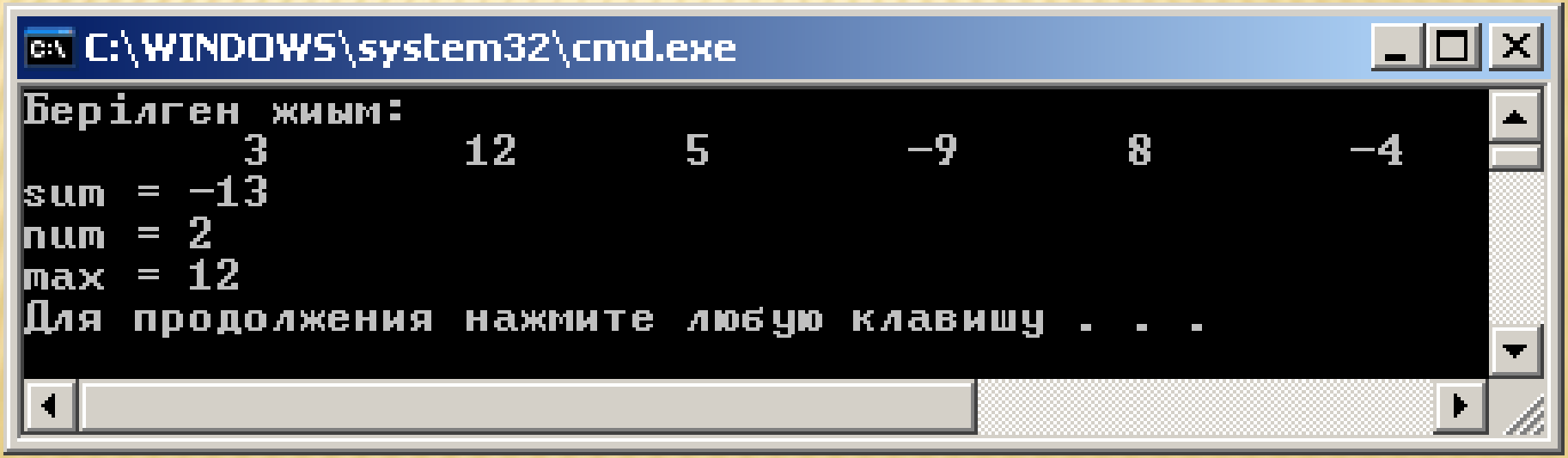
```
foreach ( int  x  in  a )
                { Console.WriteLine( x ); }
```

Мұнда цикл ішінде орындалатын жалғыз операторды жүйелі жақшаға алмаса да болады.

Енді бір өлшемді жиымды өңдейтін программаны қарастырайық, онда жиымның теріс элементтерінің қосындысы, саны және ең үлкен элементі анықталады.

```
using System;
namespace Listing6_5
{ class Class1
  { static  void  Main()
    { int[ ] a = { 3, 12, 5, -9, 8, -4 };
      Console.WriteLine( "Берілген жиым:" );
      foreach ( int elem in a )       // жиымды экранға
          Console.Write( "\t" + elem );    // шығару
       Console.WriteLine();
       long sum = 0;   // теріс элементтер қосындысы
       int num = 0;     // теріс элементтер саны
       foreach ( int elem in a )
              if ( elem < 0 )
               {                          // теріс элементтер
                  sum += elem;    // қосындысы
                  ++num;              // саны
               }
```

```
Console.WriteLine( "sum = " + sum );
    Console.WriteLine( "num = " + num );
    int max = a[0];  // максимал элемент
    foreach ( int elem in a )
      if ( elem > max ) max = elem;
    Console.WriteLine( "max = " + max );
   }
  }
 }
```



Тындағандарыңызға

рахмет!