6-Дәріс. C# тілінде бір өлшемді жиымдарды өңдеу және foreach циклі

Дәрістің мақсаты:

- C# тілінде бір өлшемді жиымдарды жариялау, жад бөлу (new), инициалдау (төрт тәсіл) және индекстеуді меңгеру.
- Жиыммен базалық амалдарды орындау: шығару/енгізу, қосынды/көбейтінді, шартқа сай санау, min/max және индекстерін табу.
- foreach циклінің синтаксисі мен мінез-құлқын түсіну (индекссіз айналу, итерация айнымалысын өзгерте алмау, қай кезде for тиімді).
- Жиымдарды сұрыптау (сызықтық/«көпіршік») ұғымын және System.Array әдістерін (Length, Sort, Reverse, IndexOf, BinarySearch, Copy, CopyTo) қолдану.
- Кездейсоқ сандар генераторын (Random) пайдаланып жиымды жылдам толтыру, шекарадан шығу қатесін (IndexOutOfRangeException) болдырмау.
- Есептердің есептік күрделілігіне (жалпы түсінік: O(n), O(n²)) назар аудару және өнімділікке әсерін сезіну.

Негізгі тақырыптар

1. **Жиымды сипаттау және жад бөлу**: int[] a; a = new int[n]; — жиым сілтемелік тип; нөмірлеу 0-ден; Length қасиеті.
2. **Инициалдау тәсілдері**:new int[10]{…}, new int[]{…}, new[]{…}, {…} — айырмашылықтары және қашан қолдану.
3. **Енгізу/шығару үлгілері**: for арқылы индекспен өту; форматталған шығару; қазақша мәтін үшін Console.OutputEncoding.
4. **Типтік үлгілер (patterns)**:қосынды/көбейтінді жинақтау; шартқа сай санау; min/max және индексі; элементтерді орын алмастыру (буфер айнымалы).
5. **foreach циклі**: синтаксисі, артықшылықтары (қарапайымдылық, қауіпсіз айналу), шектеулері (итерация айнымалысы read-only, индекске тікелей қол жоқ), қашан for жақсы (индекс керек болғанда, элементті өзгерткенде).
6. **Сұрыптау негіздері**:
   Сызықтық (таңдау) сұрыптау: идеясы, қадамдары, O(n²).
   «Көпіршік» сұрыптау: көршілерді салыстыру/алмастыру, «қалқу» эффекті, O(n²).
   Array.Sort vs қолмен жазу; тұрақтылық ұғымы (қысқаша).
7. **System.Array әдістері**: Sort, Reverse, IndexOf, LastIndexOf, BinarySearch (тек сұрыпталған жиымда), Copy, CopyTo, Clear, GetValue/SetValue, Rank.
8. **Random арқылы толтыру**: Next(), Next(max), Next(min,max), NextDouble(); аралық құру, типке құю (cast).
9. **Қателер мен қорғаныс**: шекарадан шығу, дұрыс цикл шарттары, off-by-one, кириллица шығару, енгізу қателері (FormatException) және қысқаша өңдеу.

Бір өлшемді жиымдарды сипаттау және енгізу-шығару

Жиым бір типтегі айнымалылардың ортақ ат қойылған тізбегі, яғни мәліметтер жиыны. Жиымдар біріккен айнымалыларды, мысалы, ай бойы тіркелген тәуліктік температураларды, программалаудан кітаптар тізімін, қызметкерлердің жалақыларын, т.с.с. өңдеуге қолайлы.

Жиым элементтері бір атпен белгіленеді де, тек индекстерімен (реттік нөмірлері) ажыратылады. Жиым сілтемелі типтерге жатады, сондықтан оларды құру жиым элементтеріне орын бөліп беруден басталады. Жиым элементтерін өңдеу кезінде мәндік және сілтемелік типтер қолданылады.

C# тіліндегі жиымдардың басты ерекшелігі – олар объектілер түрінде жасалған. Жиымдарды құру үшін алдымен жиымды өңдейтін айнымалыны жариялап алып, сонан соң new операторы арқылы жиым данасын (экземплярын) құру керек.

Бір өлшемді жиым біріктірілген айнымалылар тізімі болып саналады. Оны жариялау формасында тип атынан кейін тік жақшалар қойылады:

```
айнымалы_типі[] жиым_аты;
```

мұндағы типі жиым элементтерінің барлығына ортақ типті жариялайды. Жиымның әрбір элементі осы типте болады. Мысалы, num атты бүтін сандардан тұратын san жиымын анықтайық: int[] san;

Жиым айнымалысы анықталған соң, тағы да қайталанатын тип атынан кейін тұратын тік жақшаларда бір өлшемді жиым элементтері санын анықтаймыз:

int[] san = new int[4];

Алдымен int типіндегі san жиымы анықталды, сонан соң new операторы арқылы компьютер жадынан жиымның 4 элементін сақтайтын орын бөлінді.

Келесі мысалда int типіндегі 10 элементтен тұратын sandar атымен аталған жиым құрылады:

int[] sandar = new int[10];

sandar айнымалысында new операторы арқылы жиымға бөлінген жады аймағына сілтеме сақталып тұрады.

Тік жақшадағы 10 санын жиымның ұзындығы (ені) деп атайды. Осылай анықталған бүтін типтегі жиым элементтеріне, келісім бойынша, 0 мәні меншіктеледі.

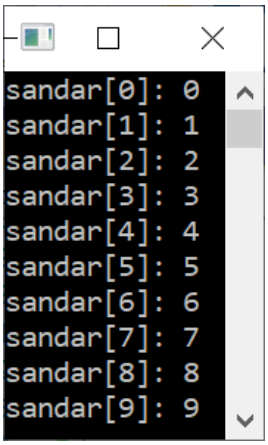
Жиымға арналған компьютер жадының аймағы оның int типіндегі 10 элементін толық орналастыра алатындай көлемде болады.

Жиымның жеке элементтерін пайдалану үшін индекстер қолданылады. **Индекс** жиымдағы элементтің реттік нөмірін көрсетіп тұрады, нөмірлеу нөлден басталады, сондықтан бірінші элемент нөмірі 0 болады.

Мысалы, а жиымы 5 элементтен тұрады делік, оның индекстері 0 мен 4 арасындағы сандармен белгіленеді. Сонымен, жиымның 1-элементінің нөмірі 0 болады, яғни a[0], ал 5-элементі a[4] болады.

Төменде 10 элементтен тұратын sandar жиымын солардың индекстеріне тең сандармен толтыру мысалы көрсетілген.

```
// Бір өлшемді жиымды сандармен толтыру
using System;
namespace Mysal
{
 class Program
 {
  static void Main(string[] args)
  {
   int[] sandar = new int[10];
   int i;
   for (i = 0; i < 10; i = i + 1)
   {  //  sandar[0] = 0 ; sandar[1] = 1; …
      sandar[i] = i;
      Console.WriteLine("sandar[" + i + "]: " +
                        sandar[i]);
   }
  }
 }
}
```



Программаның нәтижесі:

Бұл мысалда жиым индексі і айнымалысының мәні 0-ден жиымның ұзындығына дейін цикл бойынша, жиымның әр мүшесіне соған сәйкес индексі меншіктеліп, консольға нәтижесі шығып отырады. Жиым ұзындығы 10 болғандықтан, индексі 0-ден 9-ға дейін өзгереді. Соған сәйкес жиымның элементтері де 0-ден 9-ға дейінгі мәндерге ие болады.

Жиымды инициалдау. Жиым элементтеріне бастапқы мән берудің бірнеше жолы бар. Алдыңғы мысалдағы сияқты меншіктеу арқылы арфиметикалық операторлар көмегімен немесе ReadLine() әдісін қолдана отырып, мәндер енгізуге болады. Бұдан бөлек бастапқы мәндерді беруді, яғни инициалдауды программа басында бірден келесідей түрде жазу формасы да кең қолданылады:

типі[] жиым_аты = { мән1, мән2, мән3, ..., мәнN };

мұндағы мән1 ... мәнN жиымның сандық мән түрінде берілген бастапқы шамалары, олар солдан оңға қарай индекстердің өсуі бойынша реттеліп орналасады. Программада бұларға автоматты түрде жеткілікті орын бөлініп беріледі.

Программа басында жиымды сипаттаумен бірге жиым элементтеріне мән берудің төмендегі 4 түрлі жолының кез келгенін қолдануға болады:

```
int[] a = new int[10] { 1,12,3,5,9,14,3,12,0,8 };
int[] b = new int[] { 1,12,3,5,9,14,3,12,0,8 };
int[] c = new[] { 1,12,3,5,9,14,3,12,0,8 };
int[] d = { 1,12,3,5,9,14,3,12,0,8 };
```

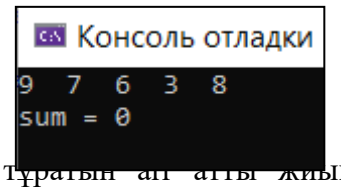
Мұнда a жиымының мәнін беру жиымның сипатталуы мен мән беруі толық қамтылған түрі. Онда жиым элементінің саны, типі, new операторы бәрі толық жазылса, b жиымында жиым элементтерінің саны көрсетілмеген. Онда енгізілген жиым элементтерінің саны бойынша жиымның ұзындығы анықталады. C жиымында тек new операторы мен жиым элементтері ғана жазылса, d жиымында тек жиым элементтері ғана көрсетілген. Осы тәсілдердің барлығы да дұрыс инициалдау тәсілдері болып табылады. Жиымды сипаттап, оған мәндер меншіктегенде, компьютер жадынан олардың элементтері саны мен типіне сәйкес, орын бөлініп беріледі.

Жоғарыдағы a жиымын сипаттаған тәсілдерде, көбінесе оның ұзындығы (жиым элементтерінің саны) көрсетілмеген, оны компьютер өзі есептеп алады. Жалпы осындай жиымдарды өңдеу кезінде, олардың ұзындығын анықтау үшін Length деген қасиет қолданылады. Ол қолданыстағы (мәндері анықталған немесе анықталуы тиіс) жиымның элементтерінің жалпы санын (ұзындығын) анықтайды. Мысал ретінде, жиымның элементтері қосындысын табатын келесі программаны қарастырайық:

```
using System;
namespace Mysal
{
   class Program
   {
      static void Main(string[] args)
      {
         int[] arr = new int[] { 9, 7, 6, 3, 8 };
         int sum = 0;   // алғашқыда қосынды  sum = 0
         for (int k = 0; k < arr.Length; k++)
            Console.Write(arr[k] + "  "); // мәндер арасында 2 босорын
         Console.WriteLine();    // келесі жолға көшу
         while (k < arr.Length)
         {
            sum += arr[k];      // sum = sum + arr[k];
            k++;
         }
         Console.WriteLine("sum = {0}", sum);

      }
   }
}
```

Программаның нәтижесі:



Бұл мысалда бүтін сандардан тұратын arr атты жиым беріліп, программа басында инициалданған. Ары қарай осы жиым элементтерін және олардың қосындысын шығару үшін цикл құрылған. Жиым элементтерін консольға шығару үшін жиымның 0-индексінен бастап, жиым ұзындығына дейін цикл бойынша Console.Write әдісі қолданылады. Бұл мысалда arr жиымына 5 элемент енгізілген. Сондықтан arr.Length мәні 5-ке тең. Мысалда жиым элементтерін консольға шығару үшін for циклі қолданылып, цилдің басы 0-ден циклдің соңы arr.Length мәніне дейін бір қадам арта отырып, цикл орындалған сайын жиымның соған сәйкес элементі консольға шығады.

Қосындыны табуда while циклі қолданылып, циклдік айнымалы k-нің мәні 0-ден бастап, sum айнымалысына жиымның arr[k] элементі қосылып отырады. 5 элементі бар жиымның

соңғы элементінің нөмірі 4 болатындықтан, k айнымалысы arr.Length мәнінен кіші мәндерді қабылдап, arr.Length мәніне тең болғанда, цикл тоқтап, жиым қосындысы консольға шығарылады.

Жоғарыдағы мысалда жиым элементтерінің қосындысын табу көрсетілді. Бұл жерде жиым элементінің санына байланысты цикл құрылып, цикл ішінде қосынды табу sum+=arr[k] өрнегі қолданылды. Жиымдарды өңдеуде осы сияқты әрекеттерді орындау жиі кездеседі. Мұнда n элементтен тұратын a жиымы қарастырылған. Программалау барысында оның орнына кез келген басқа аты бар айнымалыны алуға болады.

Жиым элементтері қосындысын табу:

```
 …
s=0;
for(i=0; i<n; i++)
   s+=a[i];   // s айнымалысында қосындыны жинақтау
 …
```

Жиым элементтері көбейтіндісін табу:

```
 …
f=1;
for(i=0; i<n; i++)
   f*=a[i];   // f айнымалысында көбейтіндіні жинақтау
…
```

Белгілі бір шартты қанағаттандыратын элементтер санын анықтау. Мысалы, бүтін сандар жиымындағы жұп сандардың қанша екенін анықтау:

```
…
k = 0;                  // жұп сандардың саны алғашқыда 0-ге тең
for(i=0; i<n; i++)
  if (a[i]%2 == 0) k++; // егер сан 2-ге қалдықсыз бөлінсе,
 …                                  // санауышты 1-ге арттыру
```

Жиымның ең үлкен (максимал) элементі мен оның нөмірін анықтау. Max айнымалысы максимал элементтің мәнін, ал k оның индексі нөмірін сақтайды. Мысалы:

```
...
max=a[0]; k=0;    // іздеуді алғашқы элементтен бастау
for(i=1;i<n;i++)  // келесіден соңына дейін қарап шығу
 if(a[i]>max)    // алдыңғысынан үлкен элементтің
{ max=a[i];       // мәнін және нөмірін (индексін)
   k=i;         // есте сақтайды
} ...
```

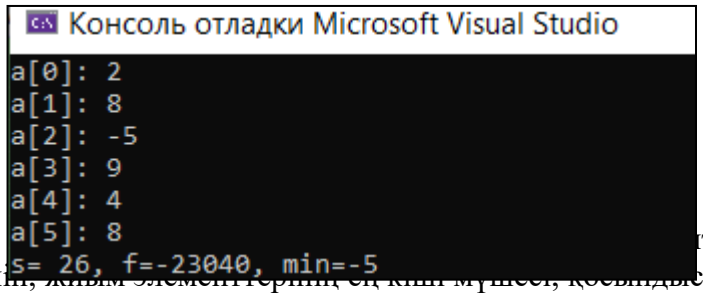
Осы тәрізді a[i]<min болғанда, ең кіші (минимал) элемент анықталады. Енді мысалдар қарастырайық. Төмендегі программада 6 элементтен тұратын жиымға пернетақтадан мәндер енгізіліп, солардың ең кішісі, қосындысы және көбейтіндісі анықталады:

```
using System;

namespace Mysal
{
   class Program
   {
      static void Main(string[] args)
      {
         const int n = 6;        // тұрақты мүшені сипаттау
         int[] a = new int[n];   // a жиымын сипаттау
         int i, s = 0, f = 1, min = a[0];
         for (i = 0; i < n; i++) // жиым элементтерін енгізу
         {
            Console.Write("a[{0}]: ", i);
            a[i] = int.Parse(Console.ReadLine());
         }
```

```
        for (i = 0; i < n; i++) // жиымның ең кіші элементін табу
        {
           if (a[i] < min)
              min = a[i];
        }
        for (i = 0; i < n; i++)
        { //  Жиым элементтерінің қосындысы мен көбейтіндісін табу
           s += a[i]; // қосындыны жинақтау
           f *= a[i]; // көбейтіндіні жинақтау
        }
        Console.WriteLine("s= {0}, f={1}, min={2}", s, f, min);
     }
  }
}
```

Программаның нәтижесі:



Бұл мысалда 6 элементтен ...ттерін Console.ReadLine() әдісі арқыл енгізілін, жиым элементтерінің ең кіші мүшесі, қосындысы мен көбейтіндісі табылған. Жиымды сипаттаудан бұрын жиым ұзындығы ретінде алынған n айнымалысы тұрақты мүшелерді сипаттау бөлімінде (const) анықталған. Мысалда үш рет цикл қолданылып, бірінші цикл Console.ReadLine() әдісі арқылы жиым элементтерін енгізуге арналса, екінші, үшінші циклдерде жиымның ең кіші мүшесі мен жиым элементтерінің қосындысы және көбейтіндісі табылған.

Жиымдарды өңдеу барысында жиым шекараларын қатаң сақтау керек. Егер жиым шекараларынан асып кетіп жатсақ, онда программа орындалуы кезінде қате шығады, яғни ерекше жағдай орын алады.

Жиымдарды сұрыптау және foreach циклін қолдану

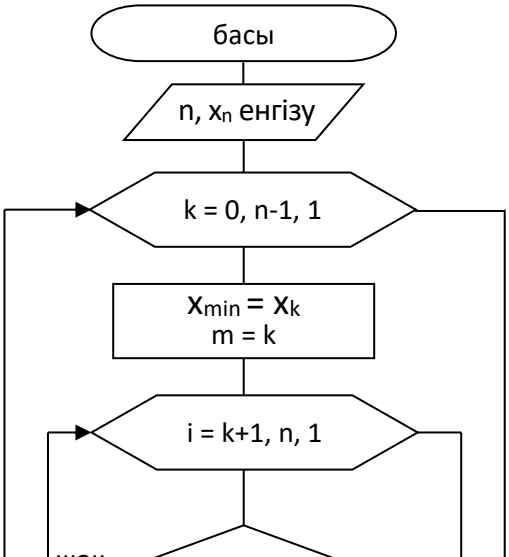
Сұрыптау (реттеу) – бір типтегі мәліметтер жиынтығын, яғни жиымды сондағы сандардың өсуі немесе кемуі бойынша реттеп орналастыру. Сұрыптау кезінде жиым элементтері олардың мәндерінің өсуі не кемуі бойынша реттеліп орналасатындай түрде алмастырылып, индекстері өзгертіледі. Көбінесе бір өлшемді жиымдарға сұрыптау қолданылады. Мысалы, студенттерді бойлары бойынша сапқа тұрғызу кезіндегі тәрізді сұрыптаулар қарастырылады.

Жиымдарды сұрыптауда сызықтық сұрыптау және «көпіршік» (“bubble sort”) тәсілімен сұрыптау әдістері жиі қолданылады.

Сызықтық сұрыптау. Жиым элементтерінің кемуі бойынша сызықтық сұрыптау тәсілінде жиым элементтері толық қарастырылып, ең үлкен мәні анықталады да, оны алғашқы элементпен алмастырамыз. Содан кейін екінші элементтен бастап, қалған элементтер толық қарастырылып, олардың да үлкені екінші элементпен орын алмастырады. Осылайша жиым элементтері соңына жеткенге дейін, әр цикл сайын жиымның ең үлкен элементін алдыға шығарып отырады. Өсу реті бойынша сұрыптағанда, керісінше, жиымның кіші элементін алға, яғни 0-индекске ығыстырып орналастырады. Ары қарай ретімен жиымның кіші элеметтерін өсу реті бойынша алға қарай орындарын жылжытып отырады.

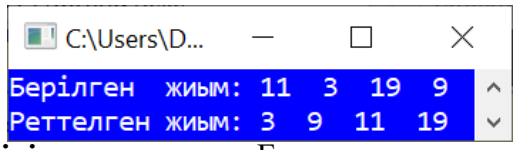
Келесі мысалда 4 элементтен тұратын жиымды өсу реті бойынша сызықтық сұрыптау әдісі қолданылған:

```
// сызықтық сұрыптау
using System;
class Program
{   static void Main()
    {   Console.ForegroundColor =
            ConsoleColor.White; // ақ символ
        Console.BackgroundColor =
            ConsoleColor.Blue;  // көк фон
        const int n=4;    // жиым ұзындығы
        int m, k, i, xmin; // жұмыс айнымалылары
        int[] x = new int[4] { 11, 3, 19, 9 };
```



```
        Console.Write("\nБерілген  жиым: ");
        for (k = 0; k < n; k++)
           Console.Write(x[k] + "  ");
         // Сұрыптау басы
        for(k=0; k<n-1; k++)
        {  xmin=x[k]; m=k;
          for(i=k+1; i<n; i++)
            if (x[i] < xmin)
                    { xmin = x[i];
                m = i;
              }
          x[m]=x[k]; x[k]=xmin;
        }
        Console.Write("\nРеттелген жиым: ");
        for (k = 0; k < n; k++)
           Console.Write(x[k] + "  ");
        Console.ReadKey();
    }
}
```

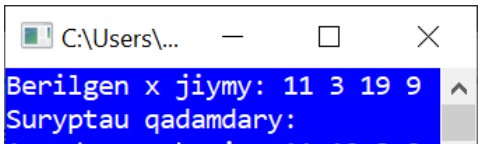
Жұмыс нәтижесі:

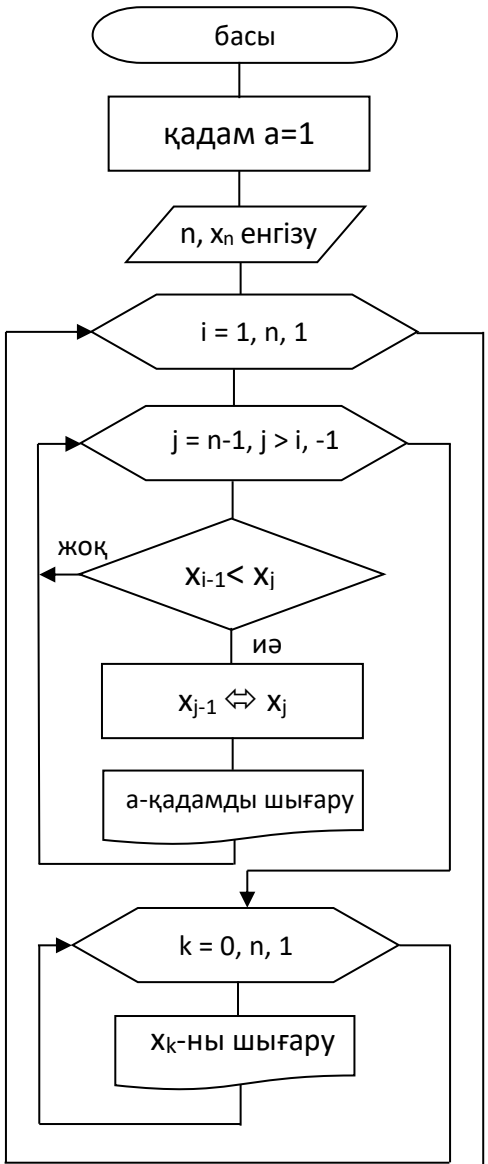


«Көпіршік» тәсілімен сұрыптау. Ең кең таралған сұрыптау тәсіліне жататын «көпіршік» тәсілінде – алгоритмнің орындалуы барысында жиымның «жеңілірек», яғни кіші элементтері біртіндеп «қалқып» жоғары (алға) шығады. Бұл тәсілдің ерекшелігі қатар тұрған екі элемент салыстырылып, қажет болса, олар орын алмастырады. Енді жоғарыдағы мысалды «көпіршік» тәсілмен орындап көрейік:

```
// "көпіршік" тәсілімен сұрыптау:
using System;
class Program
{ static void Main()
  { Console.ForegroundColor =
        ConsoleColor.White; // ақ символ
    Console.BackgroundColor =
        ConsoleColor.Blue;  // көк фон
    const int n = 4;  // элементтер саны
    int[] x = new int[n] { 11, 3, 19, 9 };
    int i, j, k, b, a = 1;  // жұмыс айнымалылары
    Console.Write("Berilgen x jiymy: ");
    for (i = 0; i < n; i++)
       Console.Write(x[i] + " ");

    Console.WriteLine("\nSuryptau qadamdary:");
    for (i = 1; i < n; i++)
    { for (j = n - 1; j >= i; j--)
      { if (x[j-1]<x[j]) // оң жақтағы элемент
        // сол жақтағыдан үлкен болса, солға жылжыту
        { b = x[j-1]; x[j-1] = x[j]; x[j] = b;
        Console.Write(a++ + "-qadamnan keyin: ");
          for (k = 0; k < n; k++)
             Console.Write(x[k] + " ");
          Console.ReadLine();
        }
      }
    }
```





```
  }
}
```

Нәтижесі:

Бұл мысалда жиымды сұрыптау үшін і айнымалысы бойынша сыртқы, j айнымалысы бойынша ішкі цикл құрылып, әр сыртқы цикл орындалған сайын жиымның үлкен элементі жиымның x[j] элементімен орын ауыстырып отырады. Әр сыртқы цикл орындалғанда x[j-1]<x[j] шарты тексеріліп, шарт ақиқат болса, x[j-1] элементке x[j] элементтің мәні меншіктеліп отырады. Бұл программада қарастырылатын аралықтағы қатар тұрған екі элементтің орнын ауыстыру үшін { b = x[j-1]; x[j-1] = x[j]; x[j] = b; } операторлар тізбегі жазылады. Мұнда b айнымалысына x[j-1] мәні уақытша меншіктеледі де, босаған x[j-1] – элементіне x[j] элементінің мәні меншіктеледі, одан кейін x[j] элементіне b айнымалысының мәні меншіктеледі. Бұл программада сыртқы цикл 1 рет орындалғанда ішкі цикл бойынша 2 орын ауыстыру орындалады. Бірінші рет 19 бен 3-тің, одан кейін 11 мен 19-дың орындары ауысады. Сыртқы цикл 3-рет орындалғанда 9 бен 3-тің орны ауысады. Ішкі циклдегі j >= і шарты әр сыртқы цикл орындалғанда, көпіршік тәрізді жоғарыға (алдыға) шыққан элементтер келесі цикл орындалғанда, салыстырылмайтынын білдіреді. Бұдан көпіршік тәсілмен сұрыптаудың біршама тез орындалатынын көруге болады.

Осы сияқты 2 айнымалының орнын ауыстыруға пайдаланылатын, уақытша мәндер сақталып тұратын b айнымалысы буферлік айнымалылар деп аталады.

Жиымдарды сұрыптауда төмендегідей ұғымдарды есте сақтаған жөн:

- *жиымдарды сұрыптау* – ақпаратты сақтау және іздеу жүйелеріндегі жұмыстардың бірі, сұрыпталған мәліметтер ішінен керектісін тауып алу жылдам орындалады;
- бір-бірінен *жылдамдығы* бойынша әртүрлі болып келетін сұрыптау алгоритмдері бар;
- жиымдарды *«жылдам» сұрыптау* тәсілдері мыңдаған элементтерден тұратын өте көлемді мәліметтер үшін аса ыңғайлы, олар мәліметтер іздеу кезінде уақыт үнемдеу мүмкіндігін береді.

Foreach циклі. Бұл оператор жиымның элементтерін бір-бірден қарастырып шығу үшін керек. Foreach операторының қолайлылығы – жиым элементтерінің санын анықтап, оның индексін бір-бірден арттырып отырмаймыз, тек жиымның барлық элементтерін қарастыру қажеттілігін көрсетеміз. Foreach циклінің жазылуы:

```
foreach ( типі аты  in  өрнек )
{ операторлар }
```

Мұндағы **аты** циклдің өзіміз таңдап алатын бір жергілікті айнымалысын көрсетеді, ол **өрнек** түрінде аты берілген жиымның барлық элементтерінің мәнін біртіндеп қабылдап отырады. Цикл ішінде соның 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 тағы бір программа қарастырайық, онда жиымның теріс элементтерінің саны мен қосындысы, және ең үлкен элементі анықталады:

```
using System;
namespace Mysal
{
  class Program
  {
    static void Main()
    {
      int[] a = { 3, 12, 5, -9, 8, -4 };
```

```
        Console.Write("Berilgen jiym: ");
        foreach (int elem in a)      // жиымды экранға
           Console.Write(elem + "   ");    // шығару
        Console.WriteLine();
        int sum = 0;       // теріс элементтер қосындысы
        int num = 0;       // теріс элементтер саны
        foreach (int elem in a)
           if (elem < 0)
           {                // теріс элементтер
              sum += elem;   // қосындысын есептеу,
              ++num;         // олардың санын табу
           }
        Console.WriteLine("Teris elementterining qosyndysy = " + sum);
        Console.WriteLine("Teris elementterining sany = " + num);
        int max = a[0];  // максимал элемент
        foreach (int elem in a)
           if (elem > max) max = elem;
        Console.WriteLine("En ulken elementining mani = " + max);
     }
  }
}
```

Программаның нәтижесі:

```
Консоль отладки Microsoft Visual Studio
Berilgen jiym: 3   12   5   -9   8   -4
Teris elementterining qosyndysy = -13
Teris elementterining sany = 2
En ulken elementining mani = 12
```

Бұл мысалда foreach цикл 3 рет қолданылған. Жиым элементтерін консольға шығаруда, теріс элементтердің қосындысын және санын табуда, жиымның ең үлкен элементін табуда жиым элеметтерін индексі бойынша емес элементтердің өзін бірден тікелей қарастырады. Мұнда жиым элементтерін білдіретін elem деген айнымалы алынып, оның а жиымына жататыны анықталды. Ары қарай цикл орындалу барысында elem айнымалысы арқылы жиымның барлық элементін қарастырады.

Foreach циклі орындалуы кезінде жиымның 1-элементі алынып, цикл айнымалысына меншіктеледі. Әрбір келесі қадамда жиымның кезекті элементі таңдалып алынады да, ол цикл айнымалысында сақталады.

Жиымның барлық элементтері қарастырылып біткен соң, цикл аяқталады. Сонымен, foreach операторы циклдік түрде жиым элементтерін оның басынан соңына дейін біртіндеп өңдеп шығады екен.

Жиымдарға кездейсоқ сандарды енгізу және System.Array класы

Сандық мәліметтерді жылдам енгізу үшін көбінесе жиымдар кездейсоқ сандармен толтырылады. C# тілінде кездейсоқ сандар генераторы бар. Оны пайдалану үшін алдымен Random атты құрамдас кластың жаңа бір данасын (экземплярын) төмендегідей түрде жасап алу керек:

```
Random kez = new Random();
```

Мұндағы «kez» деген айнымалы – кездейсоқ санды пайдалануға арналған Random классының бір данасы екендігін білдіретін кез келген айнымалы. Осындай айнымалымен Random класына сілтеме жасалған соң, кластың әдістерін шақыру арқылы кездейсоқ сандарды пайдаланамыз. 4.1-кестеде түрлі аралықтағы (диапазондағы) сандарды пайдалануға арналған Random класының негізгі әдістері мен олардың сипаттамалары берілген.

4.1- кесте. Random класының негізгі әдістері

Әдіс аты	Сипаттамасы
Next ()	int типіндегі сандар диапазонындағы кез келген оң бүтін санды қайтарады
Next (max)	0 мен max аралығынан алынған кез келген бір оң бүтін санды қайтарады

Next (min,max)	min мен max аралығынан алынған кез келген бір оң бүтін санды қайтарады
NextBytes (жиым)	byte диапазонынан алынған сандар жиымын қайтарады
NextDouble ()	0 мен 1 аралығынан алынған кез келген бір оң бөлшек санды қайтарады

Кестеде көрсетілгендей ешқандай шектеусіз кездейсоқ бүтін сандарды алу үшін Next() әдісі пайдаланылады. Ол үшін Random класына сілтеме жасалған айнымалыға Next() әдісі келесідей түрде қолданылады:

```
a[i] = kz.Next();
```

Сондай-ақ программалау барысында белгілі аралықтағы сандарды шығарып алуға болады, ол үшін сандардың бастапқы және соңғы аралық мәндерін (шекараларын) төмендегідей етіп көрсету керек:

```
a[i] = kz.Next(1,100);
```

Мұнда 1 мен 100 аралығындағы бүтін сандар шақырылады. Next() әдісі int типіндегі бүтін сандарды береді, ал егер бізге басқа типтегі сандар керек болса, онда компиляторға ол туралы тікелей түрлендіру мәліметін береміз:

```
uint b = (uint)kz.Next(0,20);
```

Келесі мысалда жиымдарда әртүрлі кездейсоқ сандарды пайдалану мүмкіндігін қарастырылып, оларда Console класында қазақ әріптерін тікелей шығару мүмкіндігі келтірілген.

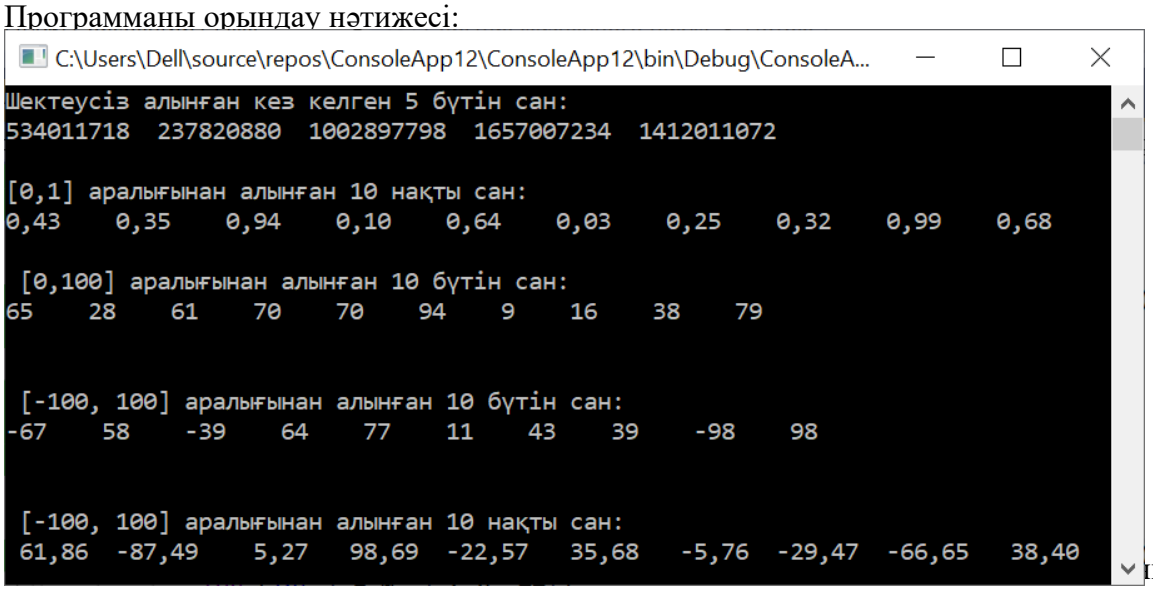
```
// Кездейсоқ сандар тізбегін алу мысалдары

// Қазақша мәтін шығару үшін...
// using System.Text; программа басында және
// Console.OutputEncoding = Encoding.UTF8;
// Main функциясы ішінде тұруы тиіс
using System;
using System.Text; // Қазақ әріптерін шығару үшін керек жол
namespace Mysal
{ class Program
  { // Кездейсоқ сандар тізбегін алу мысалдары
    static void Main(string[] args)
    { // Қазақ әріптерін шығару үшін келесі жол керек
      Console.OutputEncoding = Encoding.UTF8;
      Random a = new Random(1);
      const int n = 10;
      int[] жиым1 = new int[5];
      double[] жиым2 = new double[n];
      int[] жиым3 = new int[n];
      int[] жиым4 = new int[n];
      double[] жиым5 = new double[n];

      Console.WriteLine("\nШектеусіз алынған кез келген 5 бүтін сан: ");
      for (int i = 0; i < 5; ++i)
      {   // шектеусіз алынған кез келген 5 бүтін сан
         жиым1[i] = a.Next();
         Console.Write(жиым1[i] + "  ");
      }
      Console.WriteLine("\n");
      Console.WriteLine("[0,1] аралығынан алынған 10 нақты сан: ");
      for (int i = 0; i < n; ++i)
      {   // 0 мен 1  арасындағы  нақты сандар
         жиым2[i] = a.NextDouble();
         Console.Write("{0,4:f2}    ", жиым2[i]);
```

```
        }
        Console.WriteLine("\n");
        Console.WriteLine(" [0,100] аралығынан алынған 10 бүтін сан: ");
        for (int i = 0; i < n; ++i)
        { // 0 мен 100 арасындағы  бүтін сандар
          жиым3[i] = a.Next(100);
          Console.Write(жиым3[i] + "    ");
        }
        Console.WriteLine("\n");
        Console.WriteLine("\n[-100,100] аралығынан алынған 10 бүтін сан: ");
        for (int i = 0; i < n; ++i)
        { // -10 мен 10 арасындағы  бүтін сандар
          жиым4[i] = a.Next(-100,100);
          Console.Write(жиым4[i] + "    ");
        }
        Console.WriteLine("\n");
        Console.WriteLine("\n[-10,10] аралығынан алынған 10 нақты сан: ");
        for (int i = 0; i < n; ++i)
        { // -10 мен 10 арасындағы  нақты сандар
          жиым5[i] = a.Next(-100,100) + a.NextDouble();
          Console.Write("{0,6:f2}  ", жиым5[i]);
        }
       Console.WriteLine();
       Console.ReadKey();
      }
    }
}
```

Программаны орындау нәтижесі:



ның а данасы құрылған. Мұнда жиым1-ге a.Next() тіркесі меншіктеліп, жиымға кез келген кездейсоқ бүтін сандар енгізілген. Келесіде нақты сан типіндегі жиым2-ге a.NextDouble() меншіктеліп, 0 мен 1 арасындағы бөлшек сандар енгізілген. Одан кейін жиым3-ке a.Next(100) тіркесі меншіктеліп, оған 0 мен 100 арасындағы нақты сандар енгізілген. Содан соң жиым4-ке -100 бен 100 аралығындағы бүтін сандар беріліп, соңғы жиым5-ке бүтіні бар нақты сандарды меншіктеу үшін -10 мен 10 арасындағы бүтін сандарға 0 мен 1 аралығындағы бөлшек сандар қосындысы меншіктелген. Сондықтан жиымды аралас нақты сандармен толтыру кезінде жиым5[i] = a.Next(-10, 10) + a.NextDouble() – өрнегі қолданылған Random класының әдістері пайдаланылды.

System.Array класы. C# жиымдарының барлығы да Array атты базалық класс негізінде құрылған, оның қолданушыға пайдалы бірнеше әдістері мен қасиеттері бар, солардың бірсыпырасын қарастырып шығайық.

4.2-кесте. System.Array класының әдістері мен қасиеттері

Элемент аты	Түрі	Сипатталуы
Length	Қасиет	Жиым элементтерінің саны (барлық өлшемдері бойынша)
Rank	Қасиет	Жиым өлшемдерінің саны
Binary-Search	Статикалық әдіс	Сұрыпталған жиымдағы элементті екілік түрде іздеу
Clear	Статикалық әдіс	Жиым элементтеріне келісім бойынша мән беру
Copy	Статикалық әдіс	Бір жиым элементтерінің берілген диапазонын екінші жиымға көшіру
CopyTo	Әдіс	Ағымдағы бірөлшемді жиым элементтерінің барлығын басқа бір жиымға көшіру
GetValue	Әдіс	Жиым элементінің мәнін алу
IndexOf	Статикалық әдіс	Бір өлшемді жиымдағы элементтің алғашқы кездесу орнын анықтау
Lastlndex-Of	Статикалық әдіс	Бір өлшемді жиымдағы элементтің соңғы кездесу орнын анықтау
Reverse	Статикалық әдіс	Жиым элементтерін кері бағытта жазып шығу
SetValue	Әдіс	Жиым элементінің мәнін орнату
IndexOf	Статикалық әдіс	Бір өлшемді жиымдағы элементтің алғашқы тұрған орнын анықтау
Sort	Статикалық әдіс	Бір өлшемді жиымдағы элементтерді реттеу

Енді бір өлшемді жиыммен жұмыс істеу кезіндегі **Array** класы қасиеттері мен әдістерін қолдану мысалын қарастырайық.

```
// A[8] жиымымен жұмыс істеу кезіндегі Array класын қолдану мысалы.
using System;
using System.Text;

namespace Mysal
{
 class Program
 {
  static void Main(string[] args)
  { Console.OutputEncoding = Encoding.UTF8; // қазақша мәтін шығару
   int[] a = { 24, 50, 18, 3, 16, -7, 9, -1 };

   Console.WriteLine("Берілген a жиымы:");
   for (int i = 0; i < a.Length; i++)
      Console.Write(a[i] + "\t");
   Console.WriteLine();

   int[] b = new int[8];  // b жиымын құру
   // a-ның элементтері b-ға 0-индекстен бастап көшіріледі
   a.CopyTo(b, 0);
   Console.WriteLine("\na жиымынан көшіріліп алынған b жиымы: ");
   for (int i = 0; i < b.Length; i++)
      Console.Write(b[i] + "\t");
   Console.WriteLine();

   int[] c = new int[5];  // c жиымын құру
```

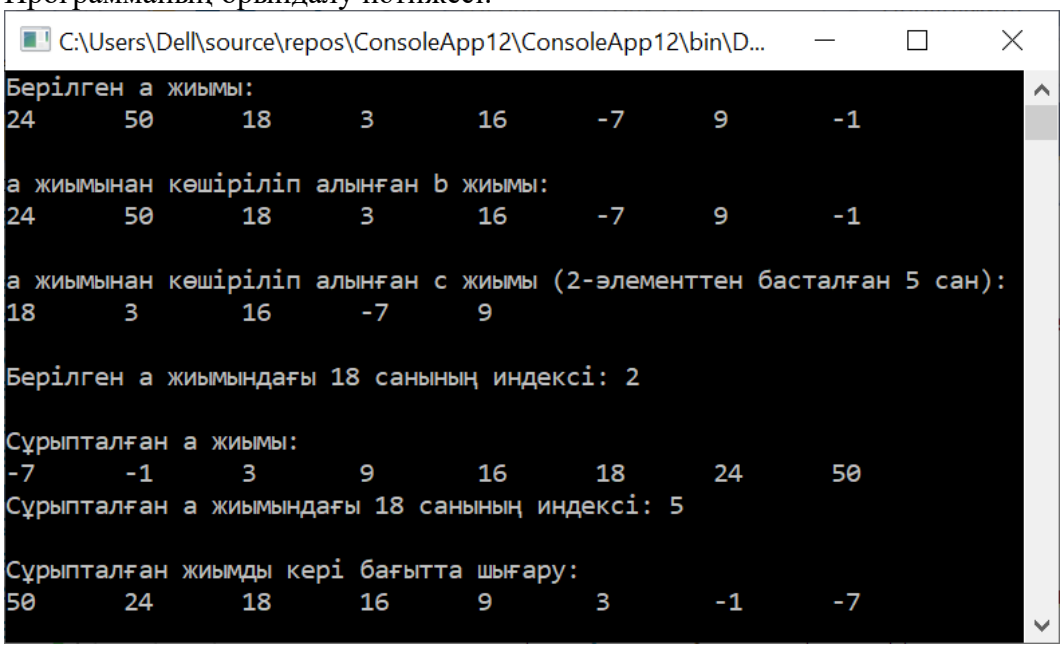
```
        // a-ның 2-элементінен басталған 5 элементі c-ға көшіріледі
        Array.Copy(a, 2, c, 0, 5);
        Console.WriteLine("\na жиымынан көшіріліп алынған c жиымы " +
                          "(2-элементтен басталған 5 сан): ");
        for (int i = 0; i < c.Length; i++)
            Console.Write(c[i] + "\t");
        Console.WriteLine();

        Console.Write("\nБерілген a жиымындағы 18 санының индексі: ");
        Console.WriteLine(Array.IndexOf(a, 18));

        Array.Sort(a);
        Console.WriteLine("\nСұрыпталған a жиымы: ");
        for (int i = 0; i < a.Length; i++)
            Console.Write(a[i] + "\t");
        Console.WriteLine();
        Console.Write("Сұрыпталған a жиымындағы 18 санының индексі: ");
        Console.WriteLine(Array.BinarySearch(a, 18));
        Array.Reverse(a);
        Console.WriteLine("\nСұрыпталған жиымды кері бағытта шығару: ");
        for (int i = 0; i < a.Length; i++)
            Console.Write(a[i] + "\t");
        Console.WriteLine();

        Console.ReadKey();
    }
  }
}
```

Программаның орындалу нәтижесі:



Бұл мысалда Sort, IndexOf, BinarySearch, Reverse, CopyTo және Copy әдістері қолданылған.

Sort, IndexOf және BinarySearch әдістері статикалық әдіс 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 әдісіндегі сызықтық іздеуден жылдам орындалады.

Программада 18-ге тең мәні бар жиым элементін осы екі әдіспен де іздеу жүргізіледі. Sort әдісі жиымды өсу реті бойынша сұрыптайды. Reverse әдісі болса жиым элементтерін кері бағытта орналастырады. Оны Sort әдісін қолданып сұрыптаған жиымға қолдансақ, жиымды кему реті бойынша шығарады. Сонан соң берілген a жиымы CopyTo және Copy әдістері арқылы c және b жиымдарына көшіріледі. Мұндағы Array.Copy(a,2,c,0,5) тіркесі Copy әдісі арқылы a жиымының 2-элементінен бастап, қатар орналасқан 5 элементінің c жиымына көшірілетінін білдіреді. Ал CopyTo әдісі (a.CopyTo(b,0)) тіркесі көмегімен a жиымының элементтерін b жиымына 0-индекстен бастап көшіреді.

Бақылау сұрақтары

1. Жиымның сілтемелік тип болуы нені білдіреді? int[] a = new int[5]; жолында не орындалады?
2. Жиымның ұзындығын қалай білеміз және неге «10 элемент үшін соңғы индекс 9»?
3. Жиымды инициалдаудың төрт тәсілін жазыңыз, айырмашылығын түсіндіріңіз.
4. for мен foreach айырмашылығы неде? Қай жағдайда for тиімдірек? Мысал келтіріңіз.
5. Теріс элементтердің санын және қосындысын табудың үлгілік кодын жазыңыз (қысқаша).
6. Max мәні мен оның индексін табу алгоритмін сипаттаңыз. Неге max=a[0] деп бастаймыз?
7. Сызықтық (таңдау) және «көпіршік» сұрыптаудың идеясы мен күрделілігін салыстырыңыз.
8. Array.Sort пен қолмен жазылған сұрыптаудың артық/кем тұстары қандай?
9. BinarySearch қашан дұрыс нәтиже береді? Неге алдын ала Sort керек?
10. Array.Copy мен CopyTo айырмашылығы неде? Мысал келтіріңіз.
11. Random.Next(min,max) пен NextDouble() айырмашылығы; [-10,10] аралығында нақты сан алу қалай жүзеге асады?
12. IndexOutOfRangeException қай кезде туады және оны болдырмау үшін қандай ережелер сақталады?
13. Off-by-one қатесі деген не? Индексті циклдерде оны қалай алдын аламыз?
14. foreach ішінде элементті өзгерту керек болса, қандай тәсіл қолданасыз?
15. Үлкен өлшемді жиымдарда өнімділікті қалай жақсартуға болады (қысқаша ұсыныстар)?

Ұсынылатын әдебиеттер

1. Колдаев, В.Д. Основы алгоритмизации и программирования: Уч. пособие/ под ред. проф. Л.Г.Гагариной. -М.: ИД «ФОРУМ»: ИНФРА-М,2015.-416 с.
2. Бөрібаев Б. Компьютердің арифметикалық негіздері: Жоғары оқу орындары студенттеріне арналған оқу құралы. -Алматы: Қазақ университеті, 2009. -80 б.
3. Бөрібаев Б. Алгоритмдеу, мәліметтер құрылымы және программалау тілдері: оқулық. – Алматы: Қазақ университеті, 2012. -235 б.
4. Вирт Н. Алгоритмы и структуры данных: Пер. с англ. -М.: Мир, 2011.-360с.
5. Бөрібаев Б., Абдрахманова М. C# тілінде программалау (практикалық курс): оқу құралы –Алматы: Қазақ университеті, 2018. -258 б.
6. Troy Dimes. C# Programming for Beginners. Paperback – January 25, 2015.
7. Шилдт Г. C# 4.0: полное руководство. М.: ООО ”И.Д.Вильямс”, 2017. -1056 с.
8. Джуст В. Разработка обслуживаемых программ на языке C#. СПб.: Изд. дом «Питер», 2017.
9. Шарп Д. Microsoft Visual C#. Подроб. руководство. 8-е изд.С-Петербург: Изд. дом «Питер», 2016.
10. A. Troelsen, P. Japikse. Pro C# 7: With .NET and .NET Core. – Apress, 2017. -1372 p.