5-Дәріс. C# тілінде циклдерді қолдану

Дәрістің мақсаты:

- C# тіліндегі цикл түрлерін (while, do…while, for, foreach) түсіну және дұрыс қолдану.
- Қайталау логикасын жобалау: инициалдау, шарт, қадам (итерация) арасындағы байланыс.
- Қабаттасқан циклдер, шексіз цикл, break/continue/goto операторларының мінез-құлқын меңгеру.
- Есептерді циклдермен тиімді шешу: шекаралық жағдайлар, off-by-one қатесі, өнімділік және деректердің түрі (int/long/double).
- Консольмен жұмыс істей отырып, циклдермен қатар шартты тексерулер мен форматтауды қолдану.

Негізгі тақырыптар

1. While және do…while
   Алдын ала/кейін тексерілетін циклдер; кемінде бір рет орындалу қасиеті (do…while).
   Санақ, цифрлар санын табу, кері жазу сияқты үлгілер.
2. For циклі
   Инициалдау → шарт → модификация; қадамның оң/теріс болуы, үтірмен бірнеше айнымалыны басқару.
   Бос өрнектер, шексіз for(;;); функцияны кестелеу, факториал/көбейтінді.
3. Foreach
   Жиым/тізім элементтерін қауіпсіз айналып өту; мутацияға шектеулер.
4. Қабаттасқан циклдер
   Екі өлшемді есептер, көбейту кестесі, қосарланған параметрлер.
5. Басқарушы операторлар
   break (ішкі циклден шығу), continue (келесі итерацияға өту), goto (қолдану шектеулері, switch-тегі тармақтарға көшу).
6. Тәжірибелік нюанстар
   Off-by-one, бүтін сандардың толып кетуі (overflow), дәлдік мәселелері (float/double), күрделі формуланы аралық айнымалыларға бөлу.
   Оқу ыңғайы және өнімділік: шартты жеңілдету, инварианттар, циклді ерте тоқтату.

While және do..while циклі

Көп жағдайларда программада аргументтердің әр түрлі мәндері бойынша операторлардың белгілі бөліктерін немесе бірнеше операторлар тобын қайталап орындауға тура келеді. Осындай процестерді ұйымдастыру үшін циклдік операторлар пайдаланылады. Қайталанатын бөліктер әртүрлі заңдылықтар мен ережелер бойынша жұмыс істейді.

Циклдер қайталану санының алдын ала белгілі және белгісіз болуына байланысты екі топқа бөлінеді. Қайталану сандары алдын ала белгілі болып келген циклдер тобы арифметикалық цикл болып есептеледі, ал орындалу саны белгісіз циклдер қадамдық (итерациялық) цикл болып саналады.

Практикада белгілі бір айнымалының сандық мәніне байланысты орындалатын арифметикалық циклдер жиі кездеседі. Мұнда арифметикалық прогрессияға ұқсас болып келетін циклдер ең қарапайым арифметикалық цикл болып табылады. Оны басқару қайталану кезеңінде прогрессияның заңына сәйкес тұрақты шамаға өзгеріп отыратын цикл параметрінің сандық мәнімен байланысты болуы тиіс.

Цикл орындалуы алдында оның айнымалы аргументі – параметрі алғашқы мәнге ие болуы керек, содан кейін қайталану кезеңінде цикл параметрі белгілі бір шамаға (қадамға) өзгере отырып, ол алдын ала берілген ең соңғы мәнге дейін жетуі қажет.

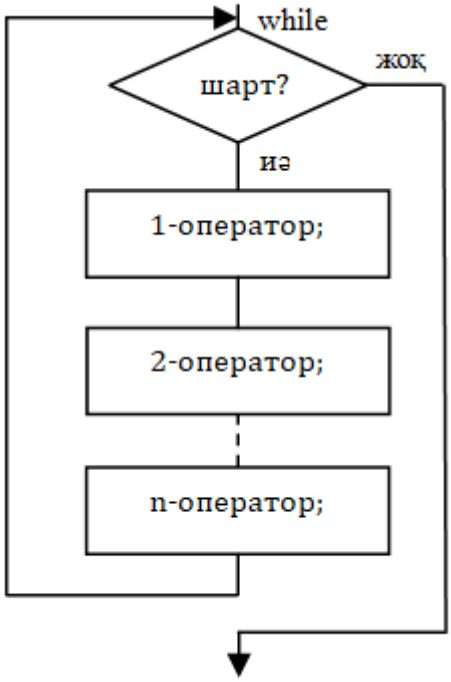
C# тілінде төрт циклдік оператор бар, олар:

- while (шарт-өрнек) {операторлар};
- do {операторлар} while (шарт-өрнек);
- for (инициалдау; шарт-өрнек; модификация)
  { операторлар };
- foreach (тип айнымалы-аты in шарт-өрнек)
  { операторлар };

While циклі шартты алдын ала тексеретін қадамдық цикл операторы. While циклінің жазылу формасы келесідей:

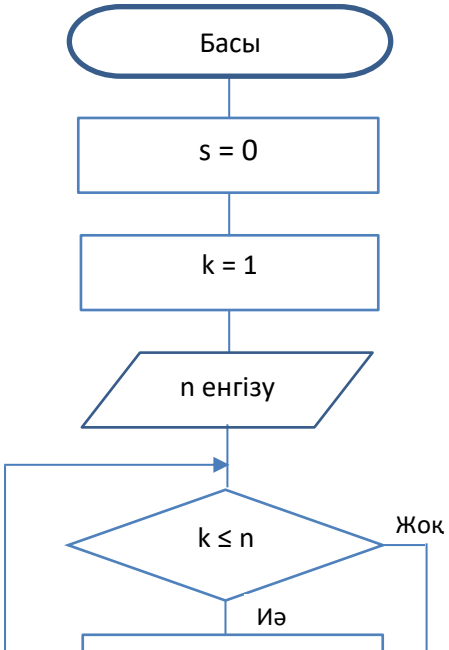
```
while (шарт-өрнек)
{   1-оператор;
    2-оператор;
     . . .
    n-оператор;
}
```

Мұнда шарт ретінде шартты өрнек немесе логикалық оператор сияқты кез келген типтегі өрнек пайдаланылуы мүмкін. Цикл денесі ретінде берілген оператор қарапайым немесе құрама түрде болады. Ол құрама оператор болса, онда операторлар жиыны жүйелі жақшаға алынып жазылады. **While** операторы орындалғанда, алдымен, жақша ішіндегі өрнек есептеліп тексеріледі. Егер өрнек мәні ақиқат болса, онда *оператор* атқарылады. Содан соң жақшадағы өрнек тағы да есептеледі. Егер өрнек мәні қайтадан ақиқат болса, цикл тағы орындалады. Ал егер жалған болса, онда цикл операторы өз жұмысын аяқтайды. While циклінің жұмыс істеу алгоритмі 3.1-суретте көрсетілген.

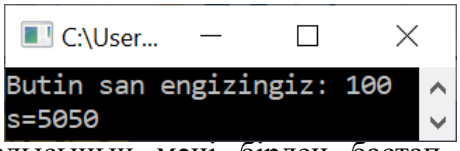


3.1-сурет. While циклінің жұмыс істеу алгоритмі

While операторының орындалу саны шексіз болып кетпес үшін мұнда шартты-өрнек құрамына кіретін айнымалы цикл ішінде өзгеріп отырады. Мысалы 1-ден 100-ге дейінгі бүтін сандар қосындысын табатын программа құрайық:



```
using System;
namespace Example
{
   class Program
   {  // Енгізілген n санына дейінгі бүтін сандар қосындысы
      static void Main(string[] args)
      {
         int s, k, n;
         s = 0; k = 1;
         Console.Write("Butin san engizingiz: ");
         n = int.Parse(Console.ReadLine());
         while (k <= n)
         {
            s += k;
            k++;
         }
         Console.WriteLine("s=" + s);
         Console.ReadKey();
      }
   }
}
```



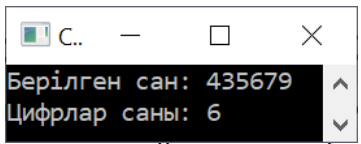
Программа нәтижесі:

Бұл жерде цикл k айнымалысының мәні бірден бастап енгізілген санға дейін s айнымалысына қосылады. K айнымалысының мәні n-нен кіші және оған тең болғанда шарт ақиқат болып, s=s+k және k=k+1 өрнектері орындалып, k айнымалысының мәні n мәнінен артқанша шарт жалған болады да, қайталану сол сәтте тоқталып, циклдің сыртында есептелген қосындыны консольге шығару операторы орындалады.

Келесі мысалда бүтін санның разрядтары санын есептеу программасы құрылған:

```
using System;
class While
{  // Бүтін сан разрядтарының қанша екенін табу

   static void Main()
   {
      int san = -435679, oryn = 0;
      Console.WriteLine("Берілген сан: " + san);
      while (Math.Abs(san) > 0)
      {
         oryn++;
         san = san / 10;
      };
      Console.WriteLine("Цифрлар саны: " + oryn);
      Console.ReadKey();
   }
}
```



Программа орындалуының нәтижесі:

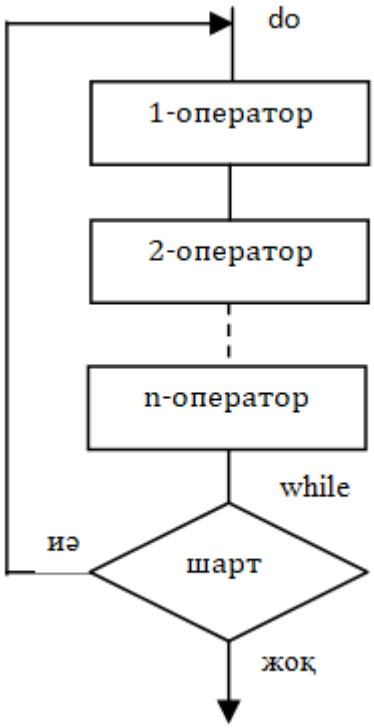
Бұл мысалда шарт san айнымалысының абсолюттік мәні 0-ден үлкен болса ақиқат болып, oryn айнымалысының мәні 1-ге артып, san айнымалысы 10-ға бөлінеді, сонда санның соңғы разряды қысқартылады. Цикл қайталанып, san мәні нөлге тең болғанда, шарт жалған мәнге ие болады да, цикл тоқталып, бүтін санның қанша орыннан тұратыны анықталады. Егер бұл мысалда san айнымалысына теріс сан енгізетін болсақ, оның абсолюттік мәні бәрі бір нөлден артық болып, цикл дұрыс нәтиже береді. Ал егер программаға бөлшек сан енгізілетін болса, онда ерекше жағдай орын алып, программа қате нәтиже береді (ондай жағдай кейін қарастырылады).

Do-while циклі. While циклдерінде шарт алдын ала тексерілсе, do-while циклінде шарт қайталанатын операторлардың соңында тексеріледі. Бұл оның, кем дегенде, бір рет орындалатынын білдіреді. Do-while циклінің жалпы жазылу формасы:

```
do {
```

операторлар ;
}
while (*шартты-өрнек*);

Цикл тұлғасы (қайталанатын ішкі операторлар, яғни цикл денесі) ретінде қарапайым немесе құрама оператор қолданылуы мүмкін. Жақшадағы өрнек цикл тұлғасынан кейін тексеріледі. Цикл ішінде бір ғана оператор тұрса, жүйелі жақшаны қоймай кетуге болады, бірақ оқылуы жеңіл болуы үшін және while циклімен шатастырмас үшін жүйелі жақшаларды қойып отыру қалыптасқан. Цикл тұлғасынан кейін жазылған шартты өрнек ақиқат болса, цикл қайтадан орындалады. Ал өрнек жалған болса, цикл аяқталады. Do…while циклінің соңында міндетті түрде нүктелі үтір қойылатынына назар аударыңыздар. Do-while циклінің жұмыс істеу алгоритмі 3.2-суретте көрсетілген.

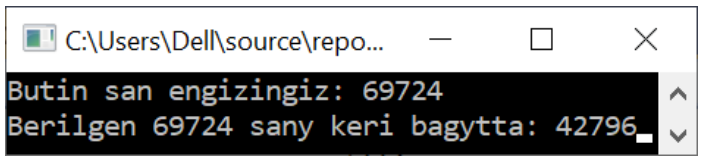


3.2-сурет. Do-wnıle циклінің жұмыс істеу алгоритмі

Келесі мысалда do-while циклі арқылы оң бүтін сан цифрларын кері бағытта жазу программасы келтірілген.

```
using System;
class DoWhile
{   // Оң бүтін сан разрядтарын кері бағытта жазу
   static void Main()
   {
      int num, nextdigit;
      Console.Write("Butin san engizingiz: ");
      num = int.Parse(Console.ReadLine());
      Console.Write("Berilgen " + num);
      Console.Write(" sany keri bagytta: ");
      do
      {   nextdigit = num % 10;
         Console.Write(nextdigit);
         num = num / 10;
      }
      while (num > 0);
      Console.ReadKey();
   }
}
```

Программаның нәтижесі:



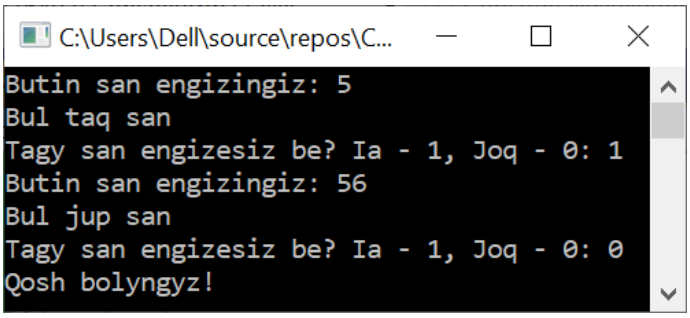
Бұл циклдің әрбір қадамында енгізілген санның оң жақ шеткі цифры сол санды 10-ға бөліп қалдық табу (num айнымалысы мәнін) арқылы айқындалады да, нәтижесі экранға шығарылады.

Ары қарай num айнымалысының мәні 10-ға қайта бөлінеді, ол да бүтін сан болғандықтан, соңғы цифр жойылады, осылай цикл num айнымалысы 0 болғанша, жалғаса береді. Бұл программа теріс және аралас сандар үшін дұрыс орындалмайды, ол үшін программа басқаша түрде құрылады.

Келесі *мысалда* енгізілген бүтін санның жұп (тақ) екендігі анықталады.

```
using System;
class DoWhile2
{   // Бүтін санның жұп немесе тақ екендігін табу
   static void Main()
   {   int k;      // тексеру үшін енгізілетін бүтін сан
      byte m;     // m=1 болса, қайталау жалғасады
      do
      {
         Console.Write("Butin san engizingiz: ");
         k = int.Parse(Console.ReadLine());
         Console.Write("Bul ");
         if (k % 2 == 0)
            Console.WriteLine("jup san");
         else
            Console.WriteLine("taq san");
         Console.Write("Tagy san engizesiz be? Ia - 1, Joq - 0: ");
         m = byte.Parse(Console.ReadLine());
      }
      while (m == 1);
      Console.WriteLine("Qosh bolyngyz!");
      Console.ReadKey();
   }
}
```

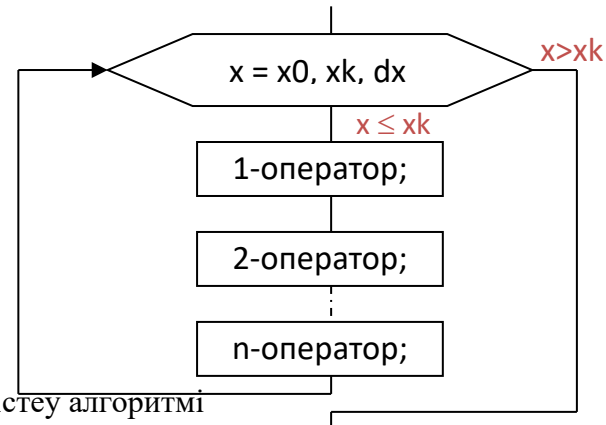
Программа нәтижесі:



C# тілінде циклдерді пайдалану. For циклі

For циклі айнымалы ретінде берілген цикл параметрінің алғашқы, соңғы мәні мен өзгеру қадамы белгілі болғанда, соған сәйкес бір немесе бірнеше операторларды қайталап орындау кезінде қолданылады. Бұл цикл параметрлі цикл немесе арифметикалық цикл деп аталады, for циклінің алгоритмі мен жалпы жазылу түрі төменде көрсетілген:

```
for(x=x0; x<=xk; x=x+dx)
{
  1-оператор;
  2-оператор;
  . . .
  n-оператор;
}
```



3.3-сурет. For циклінің жазылуы мен жұмыс істеу алгоритмі

Мұндағы *өрнек*1: **x=x0;** – цикл параметрінің (айнымалысының бастапқы) мәні,

*өрнек*2: **x<=xk;** – циклдің қайталану шарты,

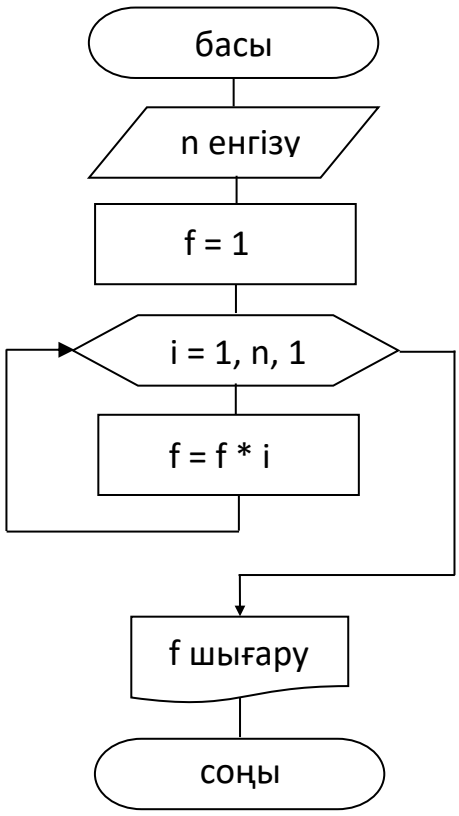
*өрнек*3: **x+=dx** – цикл параметрінің қадамы, оның артуы (кейде кемуі).

Жақша ішіндегі **x=x0** өрнегі цикл операторы орындаларда, бір рет есептеледі де, **x<=xk** өрнегі ақиқат болса, цикл тұлғасы ретіндегі *ішкі операторлар* атқарылады. Содан соң **x=x+dx** өрнегі есептеліп, **x<=xk** мәні қайта анықталады. Егер **x<=xk** мәні жалған болса, **for** циклінің жұмысы аяқталады.

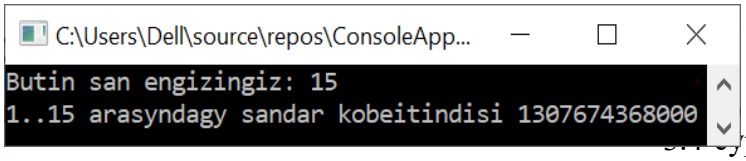
Сонымен қатар циклден белгілі бір шарт арқылы оның аяғына жетпей, шығып кету мүмкіндігі де (**break, goto** операторлары арқылы) қарастырылған.

Келесі мысалда 1-ден n-ге дейінгі сандар көбейтіндісін табу алгоритмі мен оның **for** циклі арқылы жазылуы көрсетілген.

```
using System;
namespace Example2
{ class Program
  { // 1..n арасындағы сандар көбейтіндісі
    static void Main(string[] args)
    { int i, n; long f = 1;
      Console.Write("Butin san engizingiz: ");
      n = int.Parse(Console.ReadLine());
      for (i = 1; i <= n; i++)
         f *= i;   // f = f * i;
      Console.WriteLine("1.." + n +
            " arasyndagy sandar" +
            " kobeitindisi {0}", f);
      Console.ReadKey();
    }
  }
}
```



Программаның нәтижесі:



3.4-сурет. Көбейтінді табу алгоритмі

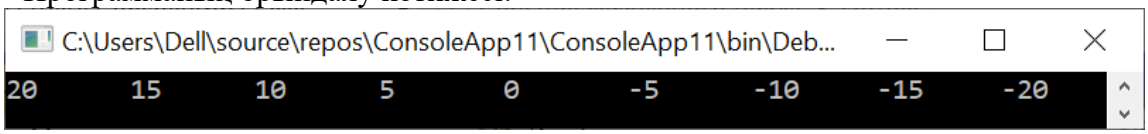
Мұнда цикл параметрінің бастапқы мәні 1-ге, соңғы мәні n-ге (i<=23), цикл қадамы 1-ге (i++) тең. N саны 23-тен асқанда, өте үлкен сан шығады да, ол long типінде көрсетілмейді (нақты сан типін қолдану керек). Сандардың көбейтіндісін f айнымалысына меншіктейміз, цикл басталмай тұрып, көбейтінділерді жинақтайтын f айнымалысының мәні 1-ге тең (қосынды табуда 0-ге тең) болады. Мұнда і айнымалысы 1-ден бастап n-ге дейін қадамы 1-ге арта отырып,

f айнымалысына көбейтінділер мәнін, яғни n!=1*2*3*…n (факториал) мәнін меншіктеп отырады.

For циклінің параметрінің өсуі де, кемуі де мүмкін. Оның өзгеру қадамы да оң, теріс сан, бүтін, нақты сан болып келе береді. Келесі мысалда цикл параметрі 20-дан -20-ға дейін қадамы -5 болып өзгерген кезде, сол сандар экранға шығарылады:

```
using System;
class DecrFor
{   // цикл параметрінің біртіндеп кемуі
   static void Main()
   {
      int x;
      for (x = 20; x >= -20; x -= 5)
         Console.Write(x + "\t");
      Console.WriteLine();
      Console.ReadKey();
   }
}
```

Программаның орындалу нәтижесі:

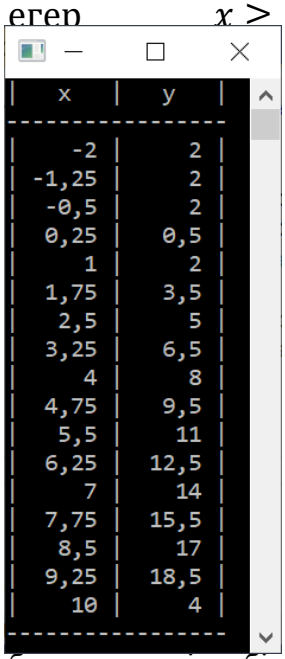


Мұндағы "\t" тіркесі – келесі бағанаға көшу белгісі, ол тек қостырнақшалар (кейде апостроф) ішінде орналасып, шығарылатын мәндер арасын ашып тұрады.

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

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

```
using System;
namespace FunkciaKestesi1
{
 class Program
 {
  static void Main()
  {
   double Xn = -2, Xk = 10, dX = 0.75, t = 2, y;
   Console.WriteLine("|  x  |  y  |");
   Console.WriteLine("----------------");
   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("| {0,5} | {1,5} |", x, y);
   }
   Console.WriteLine("----------------");
   Console.ReadLine();
  }
 }
}
```

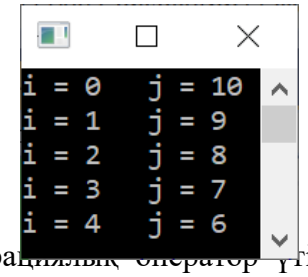


Программаны орындау нәтижесі:

Циклде бірнеше айнымалыларды пайдалану. For циклін басқару үшін бірнеше айнымалыларды қатар қолдануға болады. Мұндайда әрбір айнымалыны инициалдау мен оның қадамы үтірлермен бөлініп жазылады да, тізбектелген өрнектер солдан оңға қарай есептеледі. Келесі мысалды қарастырайық:

```
//for циклінде үтірлерді пайдалану
using System;
class Comma
{
   static void Main()
   {
      int i, j;
```

```
        for (i = 0, j = 10; i < j; i++, j--)
            Console.WriteLine("i = " + i +
                              "\tj = " + j);
        Console.ReadKey();
    }
}
```

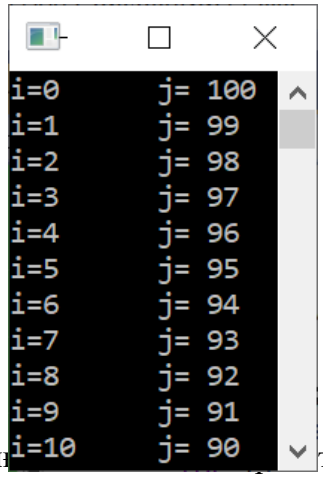


Бұл программаның нәтижесі:

Мұнда екі инициалдау операторлары және екі итерациялық оператор үтірлер арқылы бөлініп жазылған. Цикл басында екі параметр де инициалданады. Цикл қайталанған сайын i инкременттеледі, ал j декременттеледі. I айнымалысының мәні 0-ден бастап бірге артып, j айнымалысы 10-нан бастап бірге кеми отырып, экранға шығарылады. Шарт бойынша i < j жалған мәнге ие болғанда, цикл тоқтайды.

For циклiндегi шартты өрнек. For циклін басқаратын шарт рөлін bool типіндегі нәтиже беретін кез келген өрнек атқара алады. Оған міндетті түрде циклді басқаратын айнымалы кіруі тиіс. Келесі мысалда for циклін басқару done айнымалысының мәні арқылы орындалады.

```
using System;
class forDemo
{  static void Main()
   {  // циклде шартты өрнекті пайдалану
      int i, j;
      bool done = false;
      for (i = 0, j = 100; !done; i++, j--)
      {  if (i * i >= j)
            done = true;
         Console.WriteLine("i=" + i + "\t j= " + j);
      }
      Console.ReadLine();
   }
}
```



Программаның орындалу нәтижесі:

For цикл операторында **өрнек3** ретінде жалпы дұрыс жазылған өрнектерді пайдалануға болады. Мысалы:

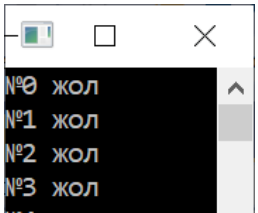
```
        for (d=0.1; d<50; d*=5)
            Console.WriteLine("d = " + d + "\t");
```

For цикл операторы форматындағы бір немесе бірнеше өрнектерді жазбауға да болады, бірақ мұндайда **;** символын міндетті түрде өз орындарына жазып отыру керек, мысалы:

```
    x=2;
    for(n=4; x<=100;)
        x=x*n;
```

Сондай-ақ for циклінің анықталуындағы жақша ішіндегі инициалдау, шартын беру және итерациялық бөліктерін түгел алып тастауға да болады. Мысал қарастырайық.

```
// for циклінің кейбір бөліктерін алып тастау
using System;
class Empty
{
   static void Main()
   {
      for (int i = 0; i < 5;)
      {
         Console.WriteLine("№" + i + " жол ");
         i++; // цикл айнымалысын инкременттеу
      }
      Console.ReadKey();
   }
}
```



Программаның нәтижесі:

Бұл мысалда циклдің басы мен соңы (шарты) ғана анықталып, қадамы цикл ішінде орындалатын операторлардан кейін берілген.

Келесі мысалда қосынды табуда цикл параметрінің бастапқы мәні циклге дейін беріліп, цикл ішіне тек нүктелі үтір қойылған.

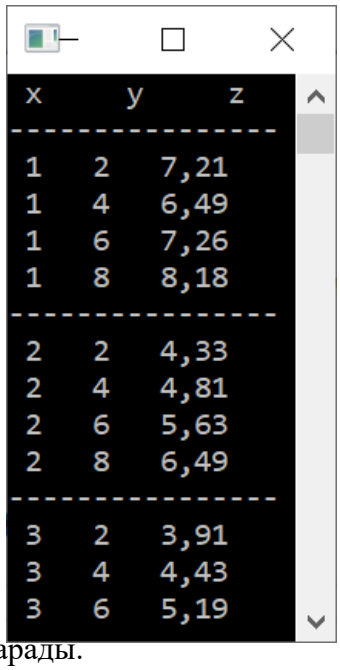
```
// for циклінің кейбір бөліктерін алып тастау
using System;
class Example5
{
        static void Main()
        {
           int i=10, s=0;
           for (; i>0; i--) //10+9+8+7+6+5+4+3+2+1
              s += i;
           Console.WriteLine("s = " + s);
           Console.ReadKey();
        }
}
```

Программа нәтижесі:

Қабаттасқан циклдер. Кейбір есептерді шығару үшін бірінің ішінде бірі орналасқан күрделі қабаттасқан циклдерді пайдалануға тура келеді. Бұндай программаларды құрғанда ішкі цикл толығымен сыртқы циклдің ішінде орналасуы қажет. Ішкі циклдің өзі де басқа ішкі циклдерді қамтуы мүмкін. Күрделі циклдің құрылымын төмендегі мысалдан көруге болады.

x = 1, 2, 3 және y = 2, 4, 6, 8 болған кездерде $z = (2y+x)/\ln x$ функциясы мәндерін есептейтін программа құру керек.

```
// x=1,2,3 және y=2,4,6,8 болғанда, z=(2y+x)/ln(xy) функциясы мәндері
using System;
class Program
{ static void Main()
  { int x, y;
    double z;
    Console.WriteLine(" x     y     z");
    Console.WriteLine("----------------");
    for (x = 1; x <= 3; x++) // сыртқы цикл
    { y = 2;
      while (y <= 8)                  // ішкі цикл
      { z = (2 * y + x) / Math.Log(x * y);
        Console.WriteLine(" {0}   {1,2} {2,6:f2}",
                          x, y, z);
        y += 2;
      }
      Console.WriteLine("----------------");
    }
    Console.ReadKey();
  }
}
```



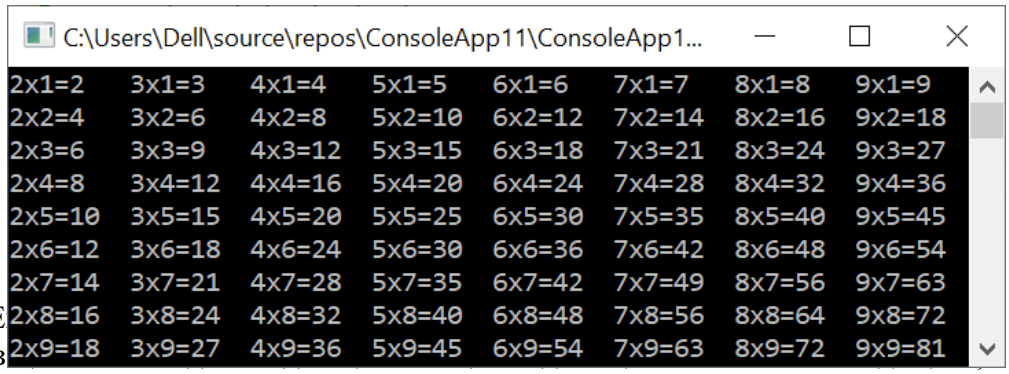
Программаның орындалу нәтижесі:

Келесі программа 2 мен 9 арасындағы көбейту кестесін шығарады.

```
using System;
namespace Kobeitu_keste1
{   class Program
    {   //  2 men 9 arasyndagy kobeitu kestesi
        static void Main()
        {   int m;      // kobeitu kestesi shygarylatyn san
            int n;      // kobeytkish
            for (n = 1; n <= 9; n++)      // сыртқы цикл басы
            {
                for (m = 2; m <= 9; m++)   // ішкі цикл басы
                {
```

```
            Console.Write($"{m}x{n}={m * n}\t");
        }           // ішкі цикл соңы
        Console.WriteLine();
    }                               // сыртқы цикл соңы
    Console.ReadKey();
  }
 }
}
```

Прогамма нәтижесі:



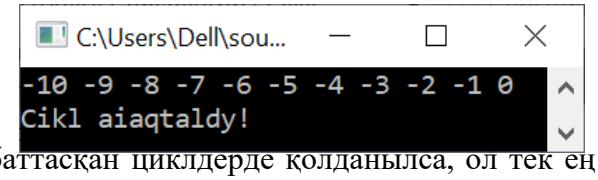
For шексіз циклі. Е
қалдырсақ, онда шексіз
одан шығу цикл ішінде орналасатын goto немесе break операторлары арқылы орындалады.

```
for(; ;)      // әдейі шексіз етілген цикл
 {
    . . .
 }
```

Мұндай цикл операциялық жүйелерде немесе арнайы программаларда кездесуі мүмкін, оларды аяқтау белгілі бір талапқа байланысты іске асады.

Циклден шығаратын break операторы. Циклдің шартын тексеруді айналып өтіп, одан бірден шығып кету үшін break операторы қолданылады. Break операторы кездескенде, цикл аяқталады да, цикл сыртындағы программа бөлігі орындала бастайды. Келесі қарастырылатын мысалда for циклінің i параметрі -10-нан 10-ға дейін берілгенмен, break операторы оны ерте i=0 болғанда, аяқтайды.

```
// Циклден шығу үшін break операторын қолдану
using System;
class BreakDemo
{   // циклді тек теріс сандар үшін ғана пайдалану
   static void Main()
   {
      for (int i = -10; i <= 10; i++)
      {
         if (i > 0) break;  // i > 0 болса, break операторын қолдану
         Console.Write(i + " ");
      }
      Console.WriteLine("\nCikl aiaqtaldy!");
      Console.ReadKey();
   }
}
```

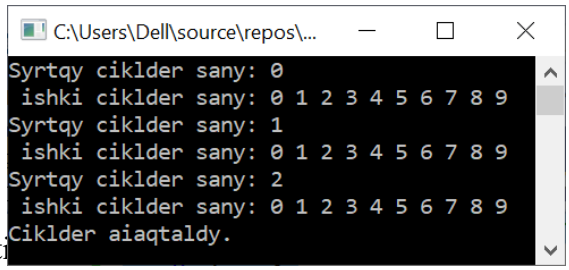


Программа нәтижесі:

Егер break операторы бірнеше қабаттасқан циклдерде қолданылса, ол тек ең ішкі циклді аяқтайды. Мысал ретінде келесі программаны қарастырайық.

```
// break операторын қабаттасқан циклдерде қолдану
using System;
class BreakNested
{ static void Main()
  { for (int i = 0; i < 3; i++)
    { Console.WriteLine("Syrtqy ciklder sany: " + i);
      Console.Write(" ishki ciklder sany: ");
      int t = 0;
      while (t < 100)
      {   if (t == 10) break;    // егер t=10 болса, аяқтау
          Console.Write(t + " "); t++;
```

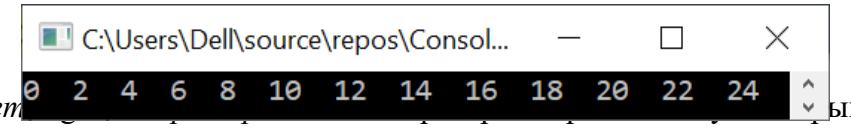
```
    }
    Console.WriteLine();
  }
  Console.WriteLine("Ciklder aiaqtaldy.");
  Console.ReadKey();
 }
}
```



Программа нәтижесі:

Келесі қадамға көшіретін continue операторы. Conti бөлігін орындамай, циклдің келесі қадамына мәжбүрлі түрде өтуді орындайды. Келесі мысал 0...100 арасындағы жұп сандарды экранға шығарады.

```
// continue операторын қолдану
using System;
class ContDemo
{  static void Main()
   {  // 0-ден 25-ке дейінгі жұп сандарды экранға шығару
      for (int i = 0; i <= 25; i++)
      {  if ((i % 2) != 0) continue; // келесі қадамға өту
         Console.Write(i + "  ");
      }
      Console.ReadKey();
   }
}
```



Программа нәтижесі:

Басқа орынға көшірет ындайды. Программада goto операторы кедессе, келесі болып осы оператор көрсететін жол орындалады. Жалпы оны программалауда пайдаланбауға тырысады, өйткені ол "араласып кеткен" код жасап, түсініксіздік туғызады.

Дегенмен, кей кездерді ол if операторымен бірге тиімді код құрастыруға мүмкіндік береді. Бұл операторды орындалуы үшін белгі – қос нүктемен аяқталатын идентификатор (жол нөмірі) қажет. Белгі goto операторынан кейін көрсетіледі және келесі орындалатын жол алдында тұруы керек.

Келесі программа мысалында 1-ден 100-ге дейінгі сандарды қосу goto операторы мен белгі (loop1 – кез келген сөз тіркесі) арқылы жүзеге асырылған.

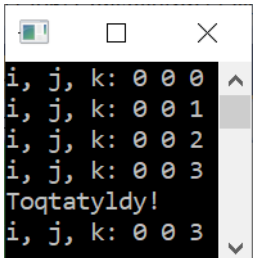
```
x = 1;
loop1: x++;
if (x < 100)  goto loop1;
```

бұлар да цикл сияқты жұмыс істейді

Бұған қоса, goto операторы switch операторының case немесе default тармағына көшу үшін де қолданылуы мүмкін. Switch операторындағы case және default сөздері оның ішкі белгілер ретінде қолданылады. Бірақ ондағы goto операторы сол switch операторының ішінде тұруы тиіс. Бұл сырттан switch операторының ішіне кіруге мүмкіндік бермейді.

Келесі программа goto операторын switch операторында қолдануды жүзеге асырады. Мұнда goto операторы қабаттасып жатқан программа бөлігінен сыртқа шығу үшін қолданылып отыр.

```
// goto операторын практикада қолдану.
using System;
class Use_goto
{ static void Main()
  { int i = 0, j = 0, k = 0;
    for (i = 0; i < 10; i++)
    { for (j = 0; j < 10; j++)
      { for (k = 0; k < 10; k++)
        {  Console.WriteLine("i, j, k: " + i + " " + j + " " + k);
           if (k == 3) goto stop;
        }
      }
    }
    stop: Console.WriteLine("Toqtatyldy!\ni, j, k: " +
                        i + " " + j + " " + k);
    Console.ReadKey();
  }
```



}

Программа нәтижесі:

Бақылау сұрақтары

1. while мен do…while айырмашылығы неде? Әрқайсысына мысал келтіріңіз.
2. for циклінің үш бөлігінің (инициалдау, шарт, қадам) логикалық рөлі қандай?
3. Бірнеше айнымалыны for ішінде қалай басқарамыз (инициалдау/қадамда үтір қолдану)?
4. Қай кезде foreach қолданған дұрыс, ал for не үшін қажет болуы мүмкін?
5. Қабаттасқан циклдерде ішкі цикл қай кезде және қалай аяқталады? Екі циклде break не істейді?
6. continue мен break айырмашылықтарын түсіндіріңіз, қолданылу мысалын жазыңыз.
7. goto операторын қолдануды қауіптері мен рұқсат етілетін жағдайларын атаңыз.
8. Шексіз циклды (for(;;) немесе while(true)) қалай қауіпсіз тоқтатамыз? Қандай жағдайларда ол орынды?
9. Off-by-one қатесі деген не? Индекстермен жұмыс істегенде қалай алдын аламыз?
10. Үлкен n үшін факториалды есептегенде int/long/BigInteger таңдауы неліктен маңызды?
11. Цикл ішіндегі қымбат есептеулерді (мысалы, Math.Log) өнімділік үшін қалай оңтайландырамыз?
12. Күрделі формуланы бір жолға емес, бірнеше аралық айнымалыға бөлудің пайдасы?
13. Енгізу/талдау (ReadLine + int.Parse/Convert) кезінде қандай ерекше жағдайлар туындауы мүмкін және оларды қалай өңдейміз?

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.