КЛАСС ӘДІСТЕРІ

7.1 Класс әдістері мен деректер туралы мағлұмат

Оқулықтың бірінші бөлімінде C# тілі бағдарламалаудың объекті-бағытталған тілі екенін және барлық әрекеттерді (бағдарламалардың кодтары) `Program` класының `Main` әдісіне жазу керектігін біз ескертіп өткенбіз.

Класс дегеніміз – бір құрылымда деректер мен деректерді өңдеу әдістерін біріктіретін жаңа тип түрі.

Класс деректері ретінде кластың тұрақтылары мен айнымалылары бола алады. Класта деректерді жариялаған кезде әдетте оған қол жеткізу спецификаторлары көрсетіледі, мысалы, `private int a;`.

Класс деректерін жариялаудың жалпы пішімі:

[спецификаторлар] [const] типі атауы

[= бастапқы_мәні].

Деректерге өзгермейтін мәндерді сақтау үшін арналған класс тұрақтылары мен класс өрістері жатады (класс айнымалыларының типтері мен атаулары).

Егер деректер алдында `public` спецификаторы қолданылса, онда олар «бағдарламаға» ашық болады, біз әрқашан `public` спецификаторын қолданамыз.

Объекті-бағытталған бағдарламалу технологиясында класс деректері әдетте «бағдарлама үшін жабылады» - `private` спецификаторы қолданылады және қалыпты жағдайда деректер мен әдістер үшін `private` спецификаторы қолданылады.

Класс әдістері дегеніміз – осы класпен жұмыс жасауға арналған, атауларға ие код үзіндісі.

Әдістер класта қолдануға болатын әрекеттер жиынтығын анықтайды (олар кластың тәртібін анықтайды).

Әдіс бір рет сипатталады, ал әртүрлі деректер үшін қанша керекті болса, сонша рет шақыртуға болады.

Класс әдістерінің жалпы жазылу пішімі мынандай түрде болады:

```
[ спецификаторлар ] әдіс типі әдіс атауы ( [ параметрлер ] )
      {әдіс денесі}
```

Мысалы:

```
 static void Main(string[] args)
                {}
```

Ең жиі кездесетін спецификаторлар: `private, public` және `static`.

`private` спецификаторымен жарияланған кластың кез келген әдістері осы кластың әдістерінде ғана қолжетімді болады.

`public` спецификаторы арқылы әдіс бағдарламаның кез келген жерінде қолжетімді болады.

`static` спецификаторы әдісті класс объектісін дайындамай-ақ, оны «класс деңгейінде» қолдануға болатындығын сипаттайды. Бұл өте маңызды, өйткені осы пәнде біз статикалық әдістерді жиі қолданамыз.

Әдіс типі кез келген бағдарламада анықталған типте немесе C# тілінің стандарты типінде немесе `void` - типсіз түрде беріле алады. Мысалы:

```
int kol(int a) { … }
public double sym(out float r) { … }
public void poisk(ref float s) { … }
public int funkcij(int a,out int b,params int[] c){ … }
```

Егер әдіс типі берілсе (`void`-тан бөлек), онда әдіс денесінде соңғы оператор ретінде әдіс жұмысының нәтижесін қайтаратын `return` операторы болуы керек. Сонымен қатар әдісті айнымалыға меншіктеу керек немесе C# тілінің операторларында өрнек сияқты қолдану керек. Әдетте осындай әдістерді функция деп атайды.

Егер әдістің алдында `void` типі жазылса, онда әдіс өзінің жұмысын `return` операторы арқылы қайтармауы керек (осы жағдайда `return` операторы әдістің денесінде болмайды). Әдетте осы әдіс процедура деп аталады, оны айнымалыға меншіктеу керек немесе жеке ішкі бағдарлама - процедура түрінде жазуға болады (дөңгелек жақшаларында параметрлері бар әдіс атауы).

Әдіс атауы – бағдарламашы анықтайтын идентификатор. Әдіс атауының мағынасы болғаны дұрыс, мысалы, `sym, max, poisk` және т.б.

Әдіс параметрлері (формалды параметрлер) бағдарлама мен әдіс арасында деректермен алмасу үшін арналған. Әдіс параметрлері әдетте керекті алгоритмді орындайтын «күйге келтіру» құралы деп аталады.

C# тілінде әдістердің келесі параметрлері бар:

– мәндерді анықтайтын параметрлер (мәндік параметрлер, яғни әдіс қабылдайтын кіріс параметрлер);

– шығыстық параметрлер (`out` қызметтік сөзімен белгіленеді);

– сілтемелік параметрлер (`ref` қызметтік сөзімен белгіленеді);

– массивті параметр (`params` қызметтік сөзімен белгіленеді).

Мәндерді анықтайтын параметрлерде қызметтік сөз қолданылмайды.

Класс әдістерінің параметрлері үтірлер арқылы бөлінеді. Әдісте массив параметрі біреу және параметрлер тізімінде соңғы болуы керек.

Егер әдісте мәндерді анықтайтын параметрлер жарияланса, онда бұл әдістің кейбір айнымалылардың көшірмелерін өз құзырына алғандығын көрсетеді. Әдіс осы көшірмелердің мәнін өзгерте алады, бірақ олардың түпнұсқасы (бағдарламада) өзгермеген қалыпта қалады. Әдістің жұмысы аяқталғаннан кейін мәндерді анықтайтын параметрлер компьютер жадысынан жойылады.

Әдістің шығыстық параметрлері бағдарламаға нәтижелерді жеткізу үшін арналған. Әдістің денесіндегі шығыстық параметрлерге кейбір мәндер меншіктелуі тиіс, әйтпесе бағдарлама компиляциясы кезінде қате кеткен туралы хабар шығады.

Егер әдісте сілтемелік параметрлер жарияланған болса, онда әдіс сәйкес айнымалылардың адресін өз құзырына алады және оларды өз алгоритмі бойынша қолдана алады (жаңа мәндерді жаза және оқи алады).

Әдістегі жарияланған массивті параметрі нақты айнымалылардың кез келген санымен жұмыс жасауға арналған. Сонымен қатар `params` қызметтік сөзінен кейін тұрған формалды параметр кез келген өлшемді деректер массивімен сәйкестікке келтіріледі.

Сонымен, әдіске өзінің параметрлері арқылы керекті мәліметтерді (мәндерді анықтайтын параметрлер және сілтемелік параметрлер) алуына немесе өз жұмысының нәтижелерін қайтаруына болады (шығыстық параметрлер және сілтемелік параметрлер).

Әдіс денесінде кейбір алгоритмді орындайтын бағдарлама кодының үзіндісі бар. Бұл ретте әдіс формалды параметрлермен бірге әрекеттер үлгісі ретінде қолданылады. Бағдарламада формалды параметрлердің орнына нақты айнымалылар қолданылуы керек – нақты параметрлер мен әдістің әрекеттер үлгісі нақты айнымалылар үшін қолданылады.

Бағдарламада әдістің формалды параметрлермен бірге жұмыс жасауы мүмкін емес.

7.2 Бағдарламада класс әдістерін қолдану

Әртүрлі есептерді шешу мысалдары арқылы бағдарламаларда әдістерді қолдануды қарастырайық.

7.1-есеп. Диалог режимінде үш бүтін сандар енгізіледі. Ең үлкен санды тауып, оны шығару керек. Бүтін типтегі функция (әдіс) қолданылады.

Есепті шешу алгоритмі келесі амалдардан тұрады:

– Диалог режимінде үш бүтін сандарды енгізу керек, мысалы, a,b,c;.

– бірінші айнымалыда ең үлкен сан жазылғанын жариялаймыз (m айнымалысына a мәні меншіктеледі);

– m айнымалысының мәнін b айнымалысының мәнімен салыстырамыз, ең үлкен мәнді m айнымалысына меншіктейміз;

– m айнымалысының мәнін c айнымалысының мәнімен салыстырамыз, ең үлкен мәнді m айнымалысына меншіктейміз;

– монитор экранына m айнымалысының мәнін шығарамыз.

Салыстыру процесін келесі бүтін типті функция түрінде жазайық:

```
static int maxc(int aa, int bb, int cc)
 {
  int mm;
  mm = aa;
  if (mm < bb) mm = bb;
  if (mm < cc) mm = cc;
  return mm;
 }
```

Бағдарламаның толық коды:

```
using System;
namespace ConsoleApplication1
{
class Program
{
static int maxc(int aa, int bb, int cc)
        {
```

```
        int mm;
        mm = aa;
        if (mm < bb) mm = bb;
        if (mm < cc) mm = cc;
        return mm;
       }
       static void Main()
       {
        int a,b,c,m;
        string buf;
        Console.Write("a bytin canin engiziniz ");
        buf = Console.ReadLine();
        a = Convert.ToInt32(buf);
        Console.Write("b bytin canin engiziniz ");
        buf = Console.ReadLine();
        b = Convert.ToInt32(buf);
        Console.Write("c bytin canin engiziniz ");
        buf = Console.ReadLine();
        c = Convert.ToInt32(buf);
        m = maxc(a,b,c);
     Console.WriteLine("Algashki sandar: {0}, {1}, {2}", a,
b, c);
     Console.WriteLine("Maksimal san = {0}", m);
     Console.WriteLine("Enter pernesin basiniz");
     Console.ReadLine();
     }
     }
     }
```

Бағдарлама жұмысы:

```
a bytin canin engiziniz 21
b bytin canin engiziniz 34
c bytin canin engiziniz 17
Algashki sandar: 21, 34, 17
Maksimal san = 34
Enter pernesin basiniz
```

`maxc()` функциясының денесінде бүтін типті mm айнымалысын жарияладық.

Бағдарламалауда жергілікті және ауқымды айнымалылар ұғымы және олармен жұмыс жасау ережелері бар.

Кез келген функция ішінде жарияланған айнымалылар жергілікті айнымалылар деп аталады.

Класс ішінде, бірақ оның кез келген функциясының сыртында жарияланған айнымалылар осы кластың ауқымды айнымалылары деп аталады.

C# тілінде атаулары бірдей жергілікті және ауқымды айнымалыларды қолдануға рұқсат етілмейді, бағдарламаның барлық айнымалыларында жеке аттары болуы керек.

Әдіс терминологиясында «көз көрерлік аймағы» ұғымы болады. Кластың кез келген әдісі өз класының барлық ауқымды айнымалыларын «көреді» және оларды қолдана алады.

Жергілікті айнымалының «өмір» уақытын әдіс жұмысының уақыты анықтайды.

7.2–есеп. Диалог режимінде үш бүтін сандар енгізіледі. Оларды кему тәртібінде шығару керек. Класс әдісін қолдану керек.

Есеп түсінікті, бірақ алгоритм анық емес. Үш айнымалының орнына екі айнымалы болса, онда есепті шешу оңай болар еді.

A және B айнымалылары бар деп есептейік. Олардың мәнін X және Y айнымалыларына жазу керек, ең үлкен мәнді X айнымалысына, ал ең кіші мәнді Y жазамыз.

Ол үшін A және B айнымалыларының мәнін салыстыру керек, егер A B-дан үлкен болса, онда X айнымалысына A мәнін меншіктейміз, ал B мәнін Y айнымалысына меншіктейміз. Әйтпесе B мәнін X айнымалысына, ал A мәнін Y айнымалысына меншіктейміз.

Екі айнымалы үшін құрылған алгоритмді біз үш айнымалы үшін де былай қолдана аламыз:

Алдымен A және B салыстырамыз, нәтижелерін X және Y жазамыз.

X және C салыстырамыз, нәтижелерін X және Z жазамыз. Ең үлкен мәнді X айнымалысына жазылады.

Z және Y салыстырамыз, нәтижелерін Y және Z жазамыз. Кему тәртібінде табылған сандарды Y және Z айнымалыларына жазылады.

Сонымен, бір алгоритмді – екі санды салыстыруды және ең үлкен және ең кіші санды табуды біз үш рет орындадық. Нәтижесінде X, Y және Z айнымалылар тізбегін аламыз, онда сандар кему тәртібінде орналасады.

Екі айнымалыға арналған алгоритмді орындау үшін static void maxmin(int aa, int bb, out int xx, out int yy) әдісін қолданайық.

Өзіміз үшін maxmin әдісінің атауында мынаны ескереміз: қайтарылатын параметрлер ең үлкен, ең кіші тәртібінде орналасады

Бағдарламаның коды:

```
using System;
namespace ConsoleApplication1
{
 class Program
 {
 static void maxmin(int aa, int bb, out int xx, out int yy)
 {
  if (aa>bb) {xx = aa; yy = bb;}
   else { xx = bb; yy = aa;}
 }
 static void Main(string[] args)
```

```
 {
  int a, b, c, x, y, z;
  string buf;
  Console.Write("a bytin canin engiziniz ");
  buf = Console.ReadLine();
  a = Convert.ToInt32(buf);
  Console.Write("b bytin canin engiziniz ");
  buf = Console.ReadLine();
  b = Convert.ToInt32(buf);
  Console.Write("c bytin canin engiziniz ");
  buf = Console.ReadLine();
  c = Convert.ToInt32(buf);
  Console.WriteLine("Algashki rettilik: {0}, {1}, {2}", a,
b, c);
  maxmin(a,b,out x,out y);
  maxmin(c,x,out x,out z);
  maxmin(y,z,out y,out z);
  Console.WriteLine("Alingan rettilik: {0}, {1}, {2}", x, y,
z);
  Console.WriteLine("Enter pernesin basiniz");
  Console.ReadLine();
 }
 }
}
```

Бағдарлама жұмысы:

```
a bytin canin engiziniz 5
b bytin canin engiziniz -3
c bytin canin engiziniz 8
Algashki rettilik: 5, -3, 8
Alingan rettilik: 8, 5, -3
Enter pernesin basiniz
```

Келтірілген мысалда static void maxmin(int aa, int bb, out int xx, out int yy) әдісі бағдарламадан екі (int aa, int bb) параметрді алады және оған екі (out int xx, out int yy) параметрді қайтарады.

Сонымен бірге, бағдарламада қолданылатын нақты параметрлерде `xx, yy` формалды параметрлердің орнында мәндер болуы тиіс.

Параметрлерді қайтаруды `ref сілтемесімен ұйымдастыруға болады, бірақ бұл үшін` қайтарылатын нақты параметрлер сілтемелік типте болуы керек.

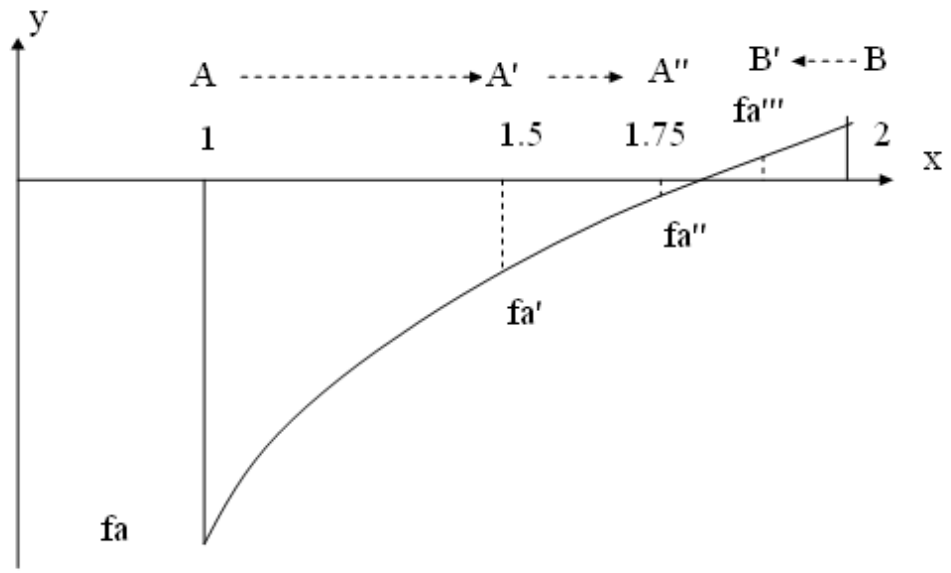
Математикалық есептерде қолданылатын функцияны пайдалану мысалы ретінде теңдеу түбірлерін табатын ортасынан бөлу әдісінің бағдарламалық жүзеге асыруын қарастырайық.

7.3-есебі. Ортасынан бөлу әдісін пайдаланып, 1 мен 2 аралығындағы кесіндіде t = 0.0001 дәлдігімен теңдеу түбірлерін табу:

$$\cos(\frac{2}{x}) - 2 \cdot \sin(\frac{1}{x}) + \frac{1}{x} = 0 \qquad (8)$$

Ортасынан бөлу әдісі берілген кесіндінің ұштарында функцияның мәні әртүрлі таңбада болады деп көздейді, мысалы , «-» и «+».

Демек, осы кесіндіде функция нөлге тең болатын кем дегенде бір х бар.



7.1-суреті – Ортасынан бөлу әдісін қолдану

Х мәнін табу алгоритмі берілген кесіндіні тең бірдей бөліктерге бөлуге және функцияның мәні әртүрлі таңбада болатын бөлікті таңдауға негізделген. Осы процесс функцияның нөлге тең болатын шешім табылғанша немесе кесінді ұзындығы берілген дәлдіктен кем болғанша қайталанады (осы жағдайда теңдеудің түбірі табылды деп есептеледі).

Алгоритмде аргументтің түрлі мәндерінде функция шешімін бірнеше рет есептеу орындалады. Сондықтан осы процессті жеке әдісте бөліп жазу керек, мысалы f атауы бар әдіс.

Бағдарлама коды:

```
using System;
namespace ConsoleApplication1
{
 class Program
 {
 static double f(double a)
  { return (Math.Cos(2/a)-2*Math.Sin(1/a)+1/a); }
 static void Main()
 {
  double a, b, x, ya, yb, t;
  a = 1;
  b = 2;
```

```
  t = 0.0001;
  ya=f(a);
  x=(a+b)/2;
  yb=f(x);
  while (b-a>t && yb!=0)
  {
  if (ya*yb<0)
   {b=x; x=(a+b)/2; yb=f(x);}
      else
   { a=x; ya=yb; x=(a+b)/2; yb=f(x);}
  }
  Console.WriteLine("Tendeydin tybiri = {0}", x);
  Console.WriteLine("Enter pernesin basiniz");
  Console.ReadLine();
 }
 }
}
```

Бағдарлама жұмысы:

```
Tendeydin tybiri = 1,87564086914063
Enter pernesin basiniz
```

f() функциясын қолдану бағдарлама көрнекілігін өсіреді және оның жазылуын жеңілдетеді.

7.3 Әдістерді қайта анықтау

C# тілінде бірнеше әдісті бір атаумен анықтауға болады, сонымен қатар оларда формалды параметрлердің әр түрлі жиыны болуы мүмкін (немесе параметрлердің түрлі типтері). C# тілінде осы тәсілді әдістерді қайта анықтау деп аталайды. Осындай әдісті шақырған кезде бағдарлама сәйкес әдісті аргументтердің санын, типін, тәртібін талдау арқылы анықтайды.

Әдістерді қайта анықтау мысалын қарастырайық:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace ConsoleApplication1
{
 class Program
 {
 public static int funk(int x) { return x*x;}
 public static double funk(double x) { return x+x; }
 public static char funk(char x) { return x; }
 static void Main(string[] args)
```

```
      {
      Console.WriteLine("x = 5 funk(5) = {0} ", funk(5));
      Console.WriteLine("x   =   5.5   funk(5.5)   =   {0}   ",
funk(5.5));
      Console.WriteLine("x   =   'g'   funk('g')   =   {0}   ",
funk('g'));
      Console.WriteLine("Enter pernesin basiniz");
      Console.ReadLine();
      }
      }
     }
```

Бағдарлама жұмысы:

```
x = 5 funk(5) = 25
x = 5.5 funk(5.5) = 11
x = 'g' funk('g') = g
Enter pernesin basiniz
```

Келтірілген мысалда аргументтерінің типтері әртүрлі, қайта жүктелген `funk` функциясын қолданады. Бірақ осындай функциялар аргументтер санымен де өзгеше болуы мүмкін.

Әдетте функцияларды қайта анықтау түрлі типтегі деректерді «түсінетін» әдістерді жазу үшін қолданылады. Мысалы, ағымдағы күнді түрлі жолдармен енгізуге болады. Мысалы:

Бүтін сандармен (ай, күн, жыл – 23 12 07);

мәтінмен (23 желтоқсан 2007 жыл);

қосымша символдарды қолданған жиын (20.10.07 немесе 17/09/2007).

Бағдарлама осы деректерді дұрыс «түсінуі» үшін әрбір тип бойынша әдістерді қайта жүктеуді қолдану қажет.

7.4 Рекурсия

Рекурсия дегеніміз – әдісте операторларды орындау барысында осы әдістің өзіне қайта жүгінуді орындайтын есептеу процесін ұйымдастыру тәсілі.

Рекурсиялық есептеулерді сипаттайтын классикалық мысалы - факториалды есептеу.

N! - есептеу керек болсын

$$N ! = 1 \cdot 2 \cdot \ldots \cdot (N-1) \cdot N = (N - I) ! \cdot N . \quad (9)$$

Сонымен, N! есептеу үшін (N-1)! білу керек.

Өз кезегінде $(N - 1) ! = 1 \cdot 2 \cdot \ldots \cdot (N - 2) ! \cdot (N - 1)$, және т.б. $2 ! = 1 ! \cdot 2$, $1 ! = 1$.

N факториалын қарапайым FOR циклі арқылы есептеуге болады, мысалы:

```
pr = 1;
for (i=1; i<=N;i++) pr = pr*i ;
```

Цикл жұмысының нәтижесі pr айнымалысында жазылады.

Рекурсияны қолдану бір немесе бірнеше шартты циклдік операциялары бар күрделі есептерді оңай шешуге көмектеседі.

Сонымен қатар, шартты цикл денесі рекурсивті функцияға жазылады, ал функцияны қолдану бағдалама кодын қысқартады және оны түсінуді жеңілдетеді.

Рекурсивті функцияны қолданатын бағдарламаның мысалы ретінде диалог режімінде енгізілген бүтін санның факториалын есептеу бағдарламасын дайындаймыз.

Бағдарлама коды:

```
using System;
namespace ConsoleApplication1
{
 class Program
 {
 static int fakt(int n)
 {
  int fu, k;
  fu = n;
  if (n != 1) { k = n - 1; fu = fu * fakt(k); }
  return fu;
 }
 static void Main()
 {
 int a, n;
 string buf;
Console.Write("n bytin canin engiziniz ");
 buf = Console.ReadLine();
 n = Convert.ToInt32(buf);
 a = fakt(n);
 Console.WriteLine("n! = {0}", a);
 Console.WriteLine("Enter pernesin basiniz");
 Console.ReadLine();
 }
 }
}
```

Бағдарлама жұмысы:

```
n bytin canin engiziniz 5
n! = 120
Enter pernesin basiniz
```

Көрсетілген бағдарлама кодында циклдар жоқ, бағдарлама алгоритмі жеңіл қабылданады: `n` айнымалының мәнін енгіземіз, факториалды есептейміз, монитор экранына нәтижені шығарамыз

Бағдарламаның орындалу үдерісінде `fakt()` рекурсивті функцияның жұмысын қарастырайық. Бағдарламаның жұмысы барысында 5 санын енгіздік деп есептейік.

5 саны үшін факториалды есептеу `fakt(5)` функциясы есептеледі. Бірақ `fakt(5)` функциясын есептеу үшін `fakt(4)` функциясын есептеу керек. Сондықтан `fakt(5)` функциясы уақытша тоқтатылады, оның параметрлері бағдарламалық стекке орналастырылады және 4 саны үшін факториалды есептеу `fakt(4)` функциясы іске қосылады.

Бірақ `fakt(4)` функциясын есептеу үшін `fakt(3)` функциясын есептеу керек. Сондықтан `fakt(4)` функциясы уақытша тоқтатылады, оның параметрлері бағдарламалық стекке орналастырылады және 3 саны үшін факториалды есептеу `fakt(3)` функциясы іске қосылады. Одан кейін стекке `fakt(3), fakt(2)` функцияларының параметрлері орналастырылады, тек содан кейін ғана `fakt(1)` функциясы есептеледі.

`fakt(1)` функциясының мәні болса, онда бағдарлама `fakt(2)` функциясының мәнін есептей алады. Ол үшін `fakt(2)` параметрлерін стектен шақыру керек және `fakt(1)` функциясының нәтижесін пайдаланып есептеу жұмысын жалғастыру керек. `fakt(2)` функциясының нәтижесін `fakt(3)` функциясын есептеу үшін қолдануға болады. Ол үшін `fakt(3)` параметрлерін стектен шақыру керек және `fakt(2)` функциясының нәтижесін пайдаланып есептеу жұмысын жалғастыру керек. Осыдан кейін `fakt(4)` және fakt(5) функциялары жоғарыда сипатталған үлгіде есептеледі.

`fakt(5)` есептелгеннен кейін оның нәтижесі а айнымалысына меншіктеледі және монитор экранына шығарылады.

Рекурсивті есептеулерді қолдану бағдарлама көрнектілігін арттырады, бірақ бағдарлама жұмысына әдеттегі циклді оператор көмегімен ұйымдастырылған бағдарламаларға қарағанда уақыт көбірек жұмсалады.

7.4-есеп. Келесі өрнекті есепте.

$$y = \cfrac{1}{1+\cfrac{1}{3+\cfrac{1}{5+\cfrac{1}{\ldots}}}} \qquad (10)$$

$$\ldots$$

$$(N-2)+\frac{1}{N}$$

Мұнда N 5555 тең, бүтін тақ сан.

7.4-есептің шешімін екі жолмен қарастырайық. `while` шартты циклінің көмегімен және рекурсия арқылы.

Шартты цикл көмегімен есепті шешу алгоритмін $\frac{1}{N}$ шамасынан бастап $\frac{1}{(1+\ldots)}$ шамасына дейін есептеу керек, әрбір циклда N 2-ге азайтылып отырады. Есептеу N>1 шарты орындалғанға дейін жүргізіледі.

Бағдарлама коды:

```
using System;
using System.Collections.Generic;
```

```
using System.Linq;
using System.Text;
namespace ConsoleApplication1
{
 class Program
 {
 static void Main()
 {
  double s, k, n;
  n=5555;
  k=1;
  s = 1 / n;
  while (n > 1)
  {
  n = n - 2;
  s = 1 / (n + s);
  }
 Console.WriteLine("Natijesi = {0} ", s);
 Console.WriteLine("Enter pernesin basiniz");
  Console.ReadLine();
 }
 }
}
```

Бағдарлама жұмысының нәтижесі:

```
Natijesi = 0,761594155955765
Enter pernesin basiniz
```

Рекурсия алгоритмі өрнектің формуласымен анықталады, $\frac{1}{(1+\ldots)}$ есептеу үшін $\frac{1}{(3+\ldots)}$ есептеп алу керек, ары қарай $\frac{1}{N}$ дейін қайталанады.

Бағдарлама коды – 2:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace ConsoleApplication1
{
 class Program
 {
 static double rec(double k, double n)
 {
  double rez;
```

```
       if (k< n) { k = k + 2; rez = (k - 2) + 1 / rec(k, n);
}
       else rez = n;
       return rez;
      }
       static void Main()
      {
       double s, k, n;
       n=5555;
       k=1;
       s = 1 / rec(k, n);
      Console.WriteLine("Natijesi = {0} ", s);
      Console.WriteLine("Enter pernesin basiniz ");
       Console.ReadLine();
      }
      }
     }
```

Бағдарлама жұмысының нәтижесі:

```
Natijesi = 0,761594155955765
Enter pernesin basiniz
```

Бағдарлама жұмысының нәтижесі бірдей.

Рекурсия қолданылған бағдарлама кодын түсіну жеңіл болып келеді.

Егер `n` айнымалысының мәнін ұлғайтсақ, мысалы `n = 55555555`, онда рекурсиясы бар бағдарламада стекті өзгертуді қажет етеді (рекурсия саны бағдарлама стегінің көлеміне байланысты), `while` циклді бағдарлама барлық ауқымдағы сандармен жұмыс істейді.

Бағдарламалық стекті пайдаланғандықтан бағдарламаның рекурсиямен жұмыс істеу уақыты циклдармен жұмыс істеу уақытынан көбірек (`n` шамасының мәні үлкен болғанда).

7.5 Өзін-өзі тексеру сұрақтары

1 C# тілінде әдістің типі анықталған болса, әдіс қалай аяқталуы керек?

2 C# тілінде әдістің қандай формалды параметрлері сілтеме-параметрлер деп аталады?

3 C# тілінде әдістің алдында `out` қызметтік сөзі жазылатын формалды параметрлері қалай аталады?

4 Егер функцияның қайтарылатын мәні `double` типінде болса, бағдарламада функцияны қалай қолдану керек?

5 Функция ішінде басқа функцияны жариялауға бола ма?

6 Қандай функцияны рекурсивті функция деп атайды?

7 Атаулары бірдей бірнеше әдісті анықтау процесі қалай аталады?

8 Кейбір әдістердің алдында `static` модификаторы неге қолданылады?

9 Келесі бағдарлама үзіндісі экранға нені шығарады:

```
public static void ttt(int a, int b, out int x, out int
y)
{ if (a>b) {x = a;y = b;}else {x = b; y = a;} }
static void Main(string[] args)
 {
 int a, b, c = 0, d = 0;
 a = 5;
 b = 8;
 ttt(a, b, out c,out d);
 Console.WriteLine("c = {0} d = {1} ", c,d);
 }?
```

10 Келесі бағдарлама үзіндісі экранға нені шығарады:

```
public static void ttt(int a, int b, int x, int y)
 { x = a; a = b; y = a; }
static void Main(string[] args)
 {
  int a, b, c = 0, d = 0;
  a = 5;
  b = 8;
  ttt(a, b, c, d);
  Console.WriteLine(" {0} {1} ", c,d);
 }?
```