4-Дәріс. C# тілінің негізгі операторлары мен операциялары, Math класы

Дәрістің мақсаты:

- System.Math класының мүмкіндіктерін (тригонометриялық, логарифмдік, дәрежелеу, түбір, дөңгелектеу, Min/Max, т.б.) қолдануды меңгеру.
- Қатынас (==, !=, <, <=, >, >=) және логикалық (&&, ||, !) операторлардың мағынасын, приоритетін және қолданылуын түсіну.
- Консольдік енгізу/шығарумен бірге математикалық өрнектерді дұрыс есептеу, форматтау дағдыларын қалыптастыру.
- Ерекше жағдайларды (exceptions) өңдеу үлгісін: try…catch…finally, стандартты аластамалар және throw қолдануды игеру.

Негізгі тақырыптар

1. **Math класы шолуы**: Sin/Cos/Tan; Asin/Acos/Atan/Atan2; Sinh/Cosh/Tanh; Exp/Log/Log10; Abs/Sqrt/Sign; Ceiling/Floor/Round; Min/Max; Pow; PI, E.
2. **Математикалық өрнектерді есептеу**: тип таңдау (double), аралас формулаларды бірнеше аралық айнымалыларға бөлу, форматтау.
3. **Қатынас және логикалық операторлар**: ақиқат мәндер, күрделі шарттар, оператор приоритеті мен жақша қолдану.
4. **Консольмен жұмыс**: Console.ReadLine, Parse/Convert, Write/WriteLine, формат спецификаторлары {index,width:format}.
5. **Ерекше жағдайлар мен оларды өңдеу**: try, catch, finally, стандартты аластамалар (DivideByZeroException, FormatException, IndexOutOfRangeException, т.б.), қолдан жасалған аластамалар (throw new Exception(...)).
6. **Үздік тәжірибелер**: тип қауіпсіздігі, шекаралық жағдайларды тексеру, қателерді пайдаланушыға түсінікті хабарлау.

Math класы, қатынас операциялары мен салыстыру операторлары

C# тілінде математикалық формулаларды есептеу үшін арнайы Math класы қолданылады. Математикалық функциялар, мысалы, синус, косинус, дәрежелеу System атаулар кеңістігінде анықталған Math класында орындалады. Ондағы әдістер (функциялар) төмендегі атаулар арқылы есептеледі:

- тригонометриялық функциялар: **Sin, Cos, Tan**;
- кері тригонометриялық функциялар: **ASin, ACos, ATan, ATan2**;
- гиперболалық функциялар: **Tanh, Sinh, Cosh**;
- экспонента және логарифмдік функциялар: **Exp, Log, Log10**;
- модуль (абсолюттік шама), квадрат түбір, таңбаны анықтау: **Abs, Sqrt, Sign**;
- дөңгелектеу: **Ceiling, Floor, Round**;
- минимум, максимум: **Min, Max**;
- дәрежелеу, қалдық табу: **Pow, IEEEReminder**;
- екі бүтін шаманың толық көбейтіндісі: **BigMul**;
- бөлу және бөлгендегі қалдықты табу: **DivRem**.

1.5-кесте. Math класының жиі кездесетін статикалық функциялары

Аты	Сипаттамасы	Нәтиже типі	Түсініктемесі
Abs	Модулі, абсолюттік мәні	int, double	$\|x\| \rightarrow$ Abs(x)
Acos	Арккосинус	double	Acos(double x), $-1 \le x \le 1$
Asin	Арксинус	double	Asin(double x) , $-1 \le x \le 1$
Atan	Арктангенс	double	Atan(double x)
Ceiling	Нақты санды үлкен бүтін санға қарай дөңгелектеу	double	Ceiling(double x)
Cos	Косинус	double	Cos(double x)

E	Натурал логарифм негізі	double	2,71828182845905
Exp	Экспонента	double	e^x → Exp(x)
Floor	Нақты санды кіші бүтін санға қарай дөңгелектеу	double	Floor(double x)
Ln	Натурал логарифм	double	lnx → Log(x)
Log10	Ондық логарифм	double	lgx → Log10(x)
Max	Екі санның максимумы	әртүрлі	Max(x, y)
Min	Екі санның минимумы	әртүрлі	Min(x, y)
PI	π санының мәні	double	3,14159265358979
Pow	Дәрежелеу	double	x^y → Pow(x, y)
Sign	Сан таңбасын анықтау	int, double	Аргументтері әртүрлі типте
Sin	Синус	double	Sin(double x)
Sinh	Гиперболалық синус	double	Sinh(double x)
Sqrt	Квадрат түбір	double	Sqrt(x)
Tan	Тангенс	double	Tan(double x)
Tanh	Гиперболалық тангенс	double	Tanh(double x)

C# тілінде кез келген математикалық формулалар Math класының 1.5-кестеге сәйкес көрсетілген әдістері бойынша программаланады. Бұл әдістерді түрлі тақырыптар бойынша мысал келтіру барысында қолданатын боламыз. Math класының әдістері төмендегі берілген формула бойынша есептеу программасын құру арқылы көрсетілген:

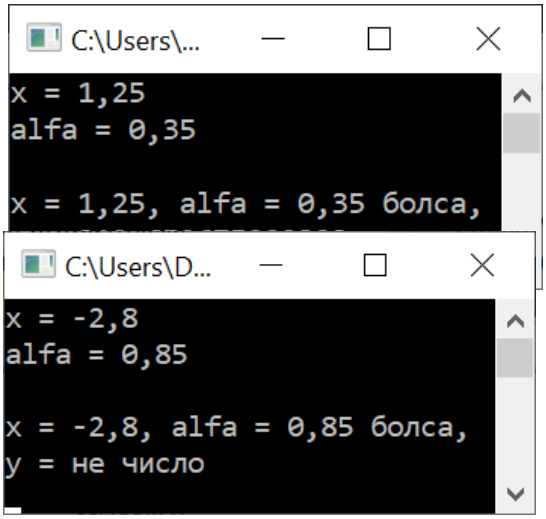
$$y = \sqrt{\pi \cdot x} - e^{0,2\sqrt{a}} + 2tg2\alpha + 1,6 \cdot 10^3 \cdot \log_{10} x^2$$

Мұндағы мәндері енгізілетін айнымалылар – x пен α. Осылар және нәтижелік мәндер типтері үшін double типін таңдаймыз, өйткені стандартты функциялар нәтижелері double типінде шығарылады. Бұл өрнекте квадрат түбір, π, экспонента, дәрежелеу, тангенс, логарифм табу функцияларын анықтау қажет. Сондықтан 1.5-кестеге сәйкес math класының әдістерін қолданамыз. Берілген өрнекті есептеу программасының бір нұсқасы төмендегідей болады, онда α орнына латынның a әрпі қолданылған:

```
using System;
namespace ConsoleApplication2
{
 class Program
 {
  static void Main()
  {
    double x, a, y, y1, y2, y;
    Console.Write("x = ");    // мән осы жолда енгізіледі
    x = Convert.ToDouble(Console.ReadLine());
    Console.Write("alfa = "); // мән осы жолда енгізіледі
    a = double.Parse(Console.ReadLine());
    y1 = Math.Sqrt(Math.PI * x)- Math.Exp(0.2 * Math.Sqrt(a));
    y2 = 2 * Math.Tan(2 * a) + 1.6e3 * Math.Log10(Math.Pow(x, 2));
    y = y1 + y2;
    Console.WriteLine();      // бір бос жол қалдыру
    Console.WriteLine("x = {0}, alfa = {1} болса, ", x, a);
    Console.WriteLine("y = " + y);
    Console.ReadKey();
```

```
    }
  }
}
```

Программаны орындау нәтижелері:



Бұл нұсқада х теріс сан болғандықтан, х*π мәнінен квадрат түбір алу мүмкін емес, сондықтан нәтиже сан болмайды.

Бұл программада Console.Write(...); жолы курсорды бірден келесі жолға төмен түсірмейтіні көрініп тұр, ал Console.WriteLine(...); жолы курсорды төменгі жолға жылжытып отырады.

Аргументсіз берілген Console.Write(); жолы көрнекілік үшін, бос жол қалдыру үшін қолданылады.

Берілген өрнекті y1 және y2 айнымалыларын қолданбай, бір жолға жазуға болар еді, бірақ онда ол жол өте көлемді болып, оқу қиындап кетеді. Сондықтан күрделі формулаларды осы мысалдағыдай екі немесе үш қарапайым өрнектерге бөліп, қосымша айнымалылар арқылы қысқаша ыңғайлы түрде бейнелеу оларды оқу мен түсінуді жеңілдетеді.

Қатынас операциялары (<, <=, >, >=,==,!=) екі санды немесе өрнекті салыстыру үшін жиі қолданылады, мұнда бірінші операндты екінші операндпен салыстырады. Операндтар арифметикалық типтерде болуы тиіс. Операция нәтижесі – true (ақиқат) немесе false (жалған) болады. Мысалы:

x == y болғанда, х пен у тең болса, нәтиже true;

x != y болғанда, х пен у тең болмаса, нәтиже true;

x < y болғанда, х у-тен кіші болса, нәтиже true;

x > y болғанда, х у-тен үлкен болса, нәтиже true;

x <= y болғанда, х у-тен кіші не тең болса, true;

x >= y болғанда, х у-тен үлкен не тең болса, true;

Егер қарастырылатын қатынас нәтижесі ақиқат болса, шартты өрнектің мәні 1-ге тең болып саналады. Егер ол жалған болса, шартты өрнек мәні 0-ге тең деп есептеледі.

Қатынас операцияларының приоритеті арифметикалық операцияларға қарағанда, төмен және меншіктеу операторымен салыстырғанда жоғары болып табылады.

Екі немесе бірнеше шартты өрнектерді біріктіру үшін күрделі логикалық операторлар (ЖӘНЕ, НЕМЕСЕ) қажет.

Логикалық операторлар. C# тілінде келесі логикалық операторлар бар:

1) **және** (&&) операциясы f1 && f2

2) **немесе** | | (or) операциясы f1 | | f2

3) **терістеу** (not) операциясы !f1

Терістеу (емес, қарсы) операциясының приоритеті өте жоғары, одан тек жақша ішіндегі мәндердің приоритеті жоғарылау болады.

&& операциясының приоритеті || (or) операциясымен салыстырғанда жоғары, ал осы екеуінің приоритеттері қатынас операцияларына қарағанда төмен, ал меншіктеу операциясынан жоғары болып саналады.

Аластамалық жағдайларды өңдеу

Ерекше жағдай (исключительный случай) немесе аластама (exception - исключение) – бұл программалау барысында келеңсіз немесе апаттық (авариялық) жағдайдың пайда болуы, ол программа коды дұрыс пайдаланылмағандықтан, орын алады. Мысалы, нөлге бөлу немесе компьютер жадының жоқ адресінен мәлімет шақыру сияқты кездерде ерекше жағдай пайда болады.

Ерекше жағдайлардың туындау процесін логикалық тұрғыдан екіге бөліп қарастыру қалыптасқан – апаттық жағдайды анықтау және оны айналып өтетіндей етіп өңдеу. Мұндай ерекше жағдайларды айналып өту мүмкіндігі оларды *аластау* деп, ал оны жүзеге асыратын код *аластама* деп аталады.

Ерекше жағдай орын алғанда келесі әрекеттер орындалады:

- программаның орындалуын тоқтату;
- «қате» деген мәнді қайтару;
- қате жайлы экранға хабар шығару және негізгі программаға жұмысты жалғастыруға болатындай бір мән қайтару;
- аластаманы айналып өту.

Аластамаларды орындау жүйесі ерекше жағдайды анықтап, оны өңдейтін try...catch...finally конструкторынан тұрады. Жалпы try операторларының жазылу синтаксисі:

```
try
{

}
catch
{

}
finally
{

}
```

Мұндағы *бақылау блогы* – құрама оператор, ол try сөзінен кейін орналасады. Бақылау блогына программаның қауіпті деген операторлары кіреді. Осы блоктан тікелей немесе жанамалы түрде шақырылатын барлық функциялар осыған қатысты болып саналады;

***Catch* блоктары** бір немесе бірнеше *аластамаларды өңдеуіш* блок. Мұнда әртүрлі типтегі қателер қалай өңделетіні сипатталады;

finally аяқтау блогы – қате бақылау блогында туындады ма, әлде басқа себеппен бе, бәрі бір орындалатын, аластамалық жағдайды аяқтайтын блок.

Try ... catch ... finally блоктарын қолданған кезде, try блогындағы барлық операторлар алдымен орындалады. Егер бұл блокта ешқандай ерекше жағдай орын алмаса, онда ол

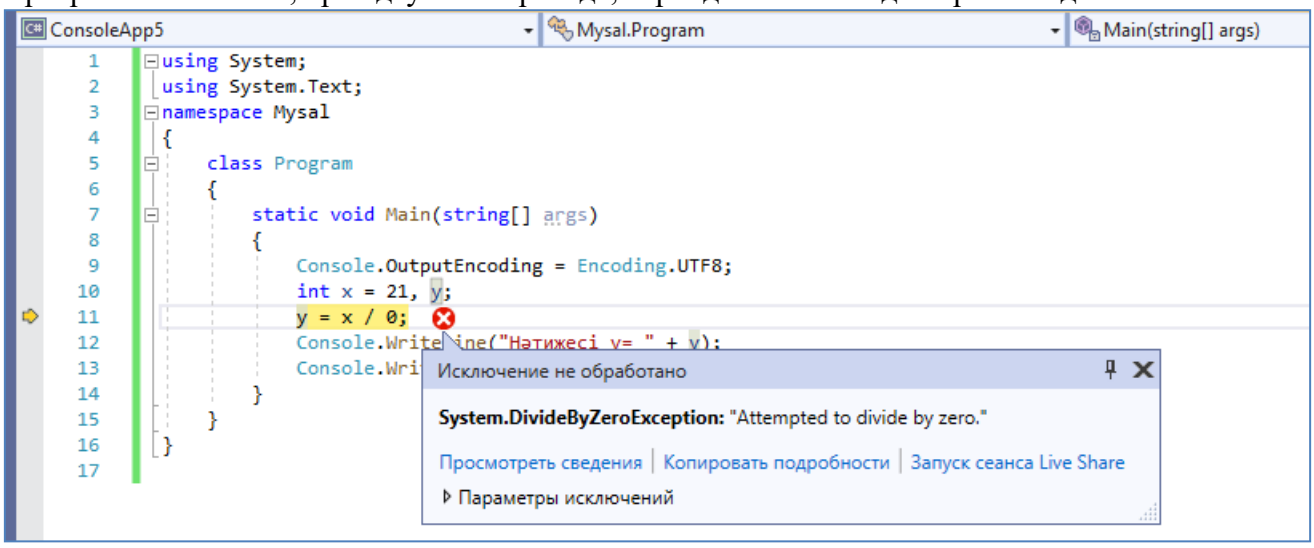
орындалғаннан кейін finally блогы орындала бастайды. Содан кейін try..catch.. finally өз жұмысын аяқтайды.

Егер try блогына ерекше жағдай жіберілсе, онда қалыпты орындау тоқтайды да, CLR ерекше жағдайды басқара алатын catch блогын іздей бастайды. Егер қажетті catch блогы табылса, ол орындалады, ал аяқталғаннан кейін finally блогы орындалады.

Егер қажетті catch блогы табылмаса, онда ерекше жағдай туындаған кезде бағдарлама оның орындалуын қалыптан тыс тоқтатады. Мысалы келесі программаны орындап көрейік:

```
using System;
using System.Text;
namespace Mysal
{
   class Program
   {
      static void Main(string[] args)
      {
         Console.OutputEncoding = Encoding.UTF8;
         int x = 21, y;
         y = x / 0;
         Console.WriteLine("Нәтижесі y= " + y);
         Console.WriteLine("Программа аяқталды");
      }
   }
}
```

Программаны жазып, орындауға жібергенде, экранда келесі жағдай орын алады:



Мұнда x айнымалысын 0-ге бөлуге талпыныс жасалған. Бірақ программа орындалу барысында санды 0-ге бөлмейтіні туралы ерекше жағдай орын алып, программа жұмысы тоқтатылды.

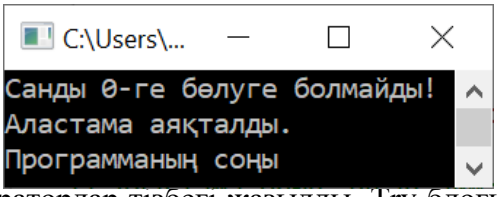
Программаның мұндай әдеттен тыс тоқтатылуын болдырмау үшін, аластамаларды өңдейиін try ... catch ... finally блогын қолдану керек. Сонымен, мысалды ары қарай өңдейтіндей етіп қайта жазайық:

```
using System;
using System.Text;
namespace Mysal
{
```

```
class Program
{
    static void Main(string[] args)
    {
        Console.OutputEncoding = Encoding.UTF8;
        try
        { // аластамалық жағдай орын алуы мүмкін блок
            int x = 21, y;
            y = x / 0;
            Console.WriteLine("Нәтижесі y= " + y);
        }
        catch (DivideByZeroException)
        { // санды 0-ге бөлгенде орын алатын аластама
            Console.WriteLine("Санды 0-ге бөлуге болмайды!");
        }
        catch
        { // жалпы жағдайда орын алатын анықталмаған аластама
            Console.WriteLine("Анықталмаған аластама орын алды");
        }

        finally
        {  // аластаманы аяқтау
            Console.WriteLine("Аластама аяқталды.");
        }
        // Аластама орын алған аймақтан кейінгі оператор
        Console.WriteLine("Программаның соңы");

    }
}
}
```



Программа жұмысының нәтижесі:

Мұнда try бақылау блогында негізгі операторлар тізбегі жазылды. Try блогы орын алған ерекше жағдайды анықтайды. Catch блогінде орын алуы мүмкін аластамалар мен сол аластама орын алғанда орындалатын оператор жазылған. Бұл мысалда санды 0-ге бөлгенде орын алатын аластама мен анықталмаған аластама көрсетілген. Try блогында санды 0-ге бөлген аластамалық жағдай орын алғандықтан, catch блогының осы бөлігі орындалып, санды 0-ге бөлуге болмайтыны туралы хабарлама экранға шығады. Finally блогына аластамалық жағдайдың анықтау аяқталғаны туралы хабарлама шығады. Осыдан кейін программа жалғасады да, аластама туындаған блоктан кейінгі операторды орындап, «программаның соңы» деген мәтінді шығарады. Бұл мысалда finally болгын алып тастасақ та, ол программаға ешқандай әсер етпейді. Ал catch блогысыз аластама туындаған программаны жалғастыру мүмкін емес.

Жоғарыдағы мысалда Exception класының DivideByZeroException аластамасы орын алды. Exception класының бұдан басқа да бірсыпыра стандартты ерекше жағдайлар туындататын аластамаларын атап өтуге болады. Олар 2.4-кестеде көрсетілген.

2.4-кесте. Exception класының стандартты аластамалары

Аты	Түсініктемесі

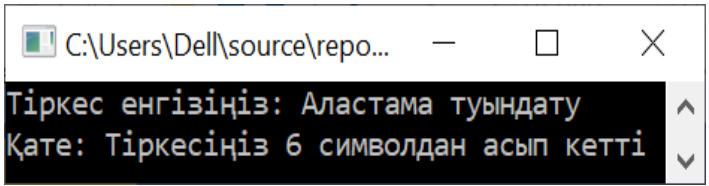
ArithmeticException	Арифметикалық операциялардағы немесе түрлендіру кезіндегі қате (DivideBeZeroException және OverFlowException аластамаларының негізі
DivideByZeroException	Нөлге бөлу орын алған кезде шығатын қате
FormatException	Әдіске (функцияға) форматы қате аргумент берілгенде, пайда болады
IndexOutOfRangeException	Жиым (массив, вектор) индексі берілген шекарадан шығып кеткенде орын алады
InvalidCastException	Типті түрлендіру қатесі
OutOfMemoryException	Жаңа объект құруға қажетті компьютер жады жетіспеген кезде орын алады
OverFlowException	Арифметикалық операцияларды орындауда өте үлкен сан шыққан (шамадан тыс толып кету) кезде орын алады
StackOverFlowException	Стектің (жадының) толып кетуі

Жоғарыдағы стандартты аластамаларды жүйенің өзі белгілі бір жағдайларда, мысалы, сандарды көбейту кезінде өте үлкен сан шығып кетуі сияқты ерекше жағдайлар жасайды. Сонымен қатар C# throw операторының көмегімен ерекше жағдайларды анықтауды жүзеге асыруға болады. Яғни осы оператордың көмегімен біз өзіміз ерекше жағдай жасап, оны жұмысын тексеріп зерттей аламыз. Оның жазылуы:

throw [өрнек];

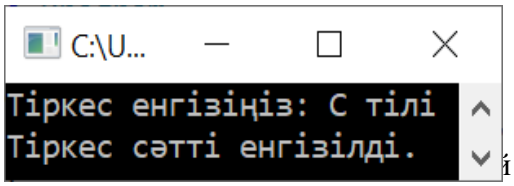
Мысал ретінде, сөз тіркесін енгізіп, оның ұзындығы 6 символдан асатын болса, аластама шығатындай етіп программа жазайық:

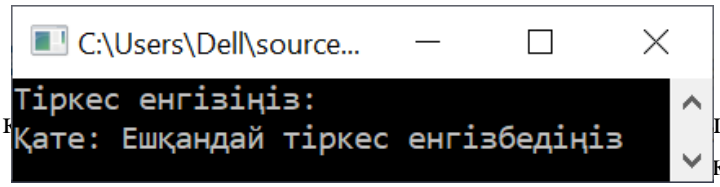
```
using System;
using System.Text;
namespace Mysal
{
 class Program
 {
  static void Main(string[] args)
  {
   Console.OutputEncoding = Encoding.UTF8;
   try
   {
    Console.WriteLine("Тіркес енгізіңіз: ");
    string message = Console.ReadLine();
    // тіркес ұзындығы 6-дан асқанда, пайда болатын аластама жасау
    if (message.Length > 6)
    { throw new Exception("Тіркесіңіз 6 символдан асып кетті"); }
     // тіркес енгізілмеген кезде, пайда болатын аластама жасау
     else if  (message.Length ==0)
    { throw new Exception("Ешқандай тіркес енгізбедіңіз"); }
     // ешқандай аластамалық жағдай орын алмағанда шығатын нәтиже
     else Console.WriteLine("Тіркес сәтті енгізілді.");
   }
   catch (Exception e)
    { Console.WriteLine("Қате: " + e.Message); }
    Console.ReadLine(); // Console.ReadKey(); тәрізді жұмыс істейді
   }
```



```
    }
}
```

Программа нәтижелері:





сәйкесінше осы 2 аластаманың бірі шақырылады. Егер символдың ұзындығы 1 мен 6-ның аралығында болса, программа сәтті аяқталып, «Тіркес сәтті енгізілді» деген нәтиже аламыз.

Бақылау сұрақтары

1. System.Math класына қандай негізгі функциялар кіреді және олардың қайсысы жиі қолданылады?
2. Неліктен күрделі формуланы бірнеше аралық айнымалылар арқылы есептеген дұрыс?
3. PI және E тұрақтылары қайда анықталған және оларды қалай қолданамыз?
4. Ceiling, Floor, Round айырмашылықтарын мысалмен түсіндіріңіз.
5. Pow, Sqrt, Log, Log10 функцияларының мәндік аймақтары мен шектеулері қандай?
6. Қатынас операторларының нәтижесі қандай типте болады және олар логикалық операторлармен қалай бірігеді?
7. Логикалық операторлардың (!, &&, ||) приоритеті қандай? Қай кезде жақша міндетті?
8. Консольден сан енгізгенде неге Parse немесе Convert керек? Қате енгізуді қалай өңдейміз?
9. Формат спецификаторлары {0,10:0.00} нені білдіреді? Мысал келтіріңіз.
10. try…catch…finally блоктарының орындалу реттілігі қандай?
11. DivideByZeroException қашан туындайды және оны қалай ұстауға болады?
12. FormatException жағдайында қолданушыға қандай хабарлама көрсету орынды?
13. throw арқылы қолдан жасалған аластаманы қашан және не үшін жібереміз?
14. catch-тің жалпы және нақты типтерімен ұстаудың айырмашылығы қандай?
15. Аластамаларды өңдеу қолданбаның сенімділігі мен UX-іне қалай әсер етеді?

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.