6 дәріс. Операторларды асыра жүктеу.

Дәрістің мақсаты: студенттерде операторларды асыра жүктеудің қызметі мен синтаксисі туралы түсініктерін көрсету қабілетін қалыптастыру.

Осы дәрісті меңгеру нәтижесінде студенттер келесі қабілеттерге ие болады:

- Арифметикалық операторларды асыра жүктеу ерекшеліктері бойынша түсініктерін көрсету;
- Логикалық операторларды асыра жүктеу ерекшеліктері бойынша түсініктерін көрсету;
- Типтерді келтіру операторларын асыра жүктеу ерекшеліктері бойынша түсініктерін көрсету.

C# тілінде құрылатын класқа байланысты операторлардың атқаратын жұмысын анықтауға болады. Бұл процесс *операторларды асыра жүктеу* деп аталады. Асыра жүктеуге орай кластағы операторлардың қолданылу аймағы кеңейтіледі. Мұнда операторлардың атқаратын жұмысы толық бақыланады да, оның істейтін қосымша жұмысы нақты класқа байланысты өзгеруі мүмкін. Мысалы, **+** операторы бір класта объектіні бір байланысқан тізімдерге енгізуі мүмкін, ал басқа класта ол басқа бір әрекет орындауы мүмкін.

Оператор асыра жүктелген кезде оның бұрынғы атқарып отырған жұмыстары жалғаса береді, ол нақты объектіге байланысты тағы бір жаңа операция орындайтын болады. Сондықтан **+** операторын тізімдерді өңдеуге байланысты асыра жүктеу, мысалы, оның бұрынғы бүтін сандармен орындайтын жұмысын (оларды қосуды) өзгертпейді.

Операторларды асыра жүктеу үшін *операторлық әдісті* анықтайтын operator түйінді сөзін қолданамыз, ал ол, өз кезегінде, қолданылатын класына байланысты оператор жұмысын анықтайды.

Операторлық әдістің (**operator**) екі формасы бар: олардың бірі – унарлық операторлар үшін болса, екіншісі – бинарлық операторлар үшін қолданылады. Төменде осы әдістердің әрқайсысының жазылу формасы көрсетілген.

```
// Унарлық операторды асыра жүктеу түрінің жазылуы
public static қайтарылатын_тип operator
                        оp(параметр_типі операнд)
{
   // операциялар
}
// Бинарлық операторды асыра жүктеу түрінің жазылуы
public static қайтарылатын_тип operator оp (параметр1_типі
                        операнд1, параметр2_типі операнд2)
{
   // операциялар
}
```

Мұнда **op** орнына асыра жүктелетін оператор қойылады, мысалы, **+** немесе **/**; ал *қайтарылатын_тип* көрсетілген операция қайтаратын мәннің типін білдіреді. Бұл мән кез келген типте бола береді, бірақ көбінесе ол асыра жүктелген оператор пайдаланылатын кластың типіндей болады. Унарлық операторлар үшін операнд берілетін операндты білдіреді, ал бинарлық операторлар үшін – олар *операнд1* және *операнд2* болып белгіленген. Операторлық әдістердің екеуі де **public** және **static** типінде болатынына назар салыңыздар.

Унарлық операторлардың операнд типі оператор асыра жүктелетін класс типіндей болуы тиіс, ал бинарлық операторларда кем дегенде оның операндтардың біреуі өз класындағыдай типте болуы керек.

Бинарлық операторларды асыра жүктеу

Мысал 1. Операторларды асыра жүктеуді көрсету үшін, **+** пен **-** сияқты екі операторды асыра жүктейтін қарапайым мысал қарастырайық. Төмендегі программада комплекс санның нақты және жорамал бөліктері арқылы берілген Complex класы құрылады. Асыра жүктелген "**+**" операторы комплекс сандардың нақты және жорамал бөліктерін сәйкесінше қосады. Ал асыра жүктелген "**-**" операторы комплекс сандардың нақты және жорамал бөліктерін сәйкесінше азайтады.

```
// Бинарлық операторларды асыра жүктеу мысалы
using System;
// Комплекс сандармен жұмыс істеуге арналған класс
class Complex {
   int a, b;       // санның нақты және жорамал бөліктері
   public Complex() { a = b = 0; }
   public Complex(int x, int y) { a = x; b = y; }

   // + бинарлық операторын асыра жүктеу
   public static Complex operator +(Complex op1, Complex op2)
   {
      Complex natizhe = new Complex();
      // екі комплекс санның бөліктерін қосып,
      // нәтижесін қайтару
      natizhe.a = op1.a + op2.a;
      natizhe.b = op1.b + op2.b;
      return natizhe;
   }

   // - бинарлық операторын асыра жүктеу.
   public static Complex operator -(Complex op1, Complex op2)
   {
      Complex natizhe = new Complex();
      /* Операндтардың орналасу реттілігіне назар салыңыз:
         op1 – сол жақ операнд, ал op2 – оң жақ операнд. */
      natizhe.a = op1.a - op2.a;
      natizhe.b = op1.b - op2.b;
      return natizhe;
   }
   // комплекс санды шығару
   public void Shygaru()
   {
      Console.WriteLine(a + " + " + b + "i ");
   }
}

class Program {
   static void Main() {
      Complex C1 = new Complex(3, 8);
      Complex C2 = new Complex(10, 2);
      Complex C3;
      Console.Write("C1 complex sany: ");
      C1.Shygaru();
      Console.Write("C2 complex sany:  ");
      C2.Shygaru();
      C3 = C1 + C2;       // C1 және C2 комплекс сандарын қосу
      Console.Write("C3 = C1 + C2 qosu natizhesi: ");
      C3.Shygaru();
      C3 = C1 + C2 + C3;  // үш комплекс санды қосу
      Console.Write("C3 = C1 + C2 + C3 qosu natizhesi: ");
      C3.Shygaru();
      C3 = C3 - C1;       // нәтижелік комплекс саннан бірінші
                          // комплекс санды азайту
      Console.Write("C3 = C3 – C1 azaitu natizhesi: ");
      C3.Shygaru();
      C3 = C1 + C3 - C2;  // операцияларды аралас пайдалану
      Console.Write("C3 = C1 + C3 - C2 amalynyng natizhesi: ");
      C3.Shygaru();
```

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

Бұл программаны орындау нәтижесі:

```
C1 complex sany: 3 + 8i
C2 complex sany: 10 + 2i
C3 = C1 + C2 qosu natizhesi: 13 + 10i
C3 = C1 + C2 + C3 qosu natizhesi: 26 + 20i
C3 = C3 – C1 azaitu natizhesi: 23 + 12i
C3 = C1 + C3 - C2 amalynyng natizhesi: 16 + 18i
```

operator+() әдісінің **Complex** типіндегі объект қайтаратынына назар салыңыздар, осы себеппен **+** операторын мынадай **C1 + C2 + C3** құрамалы өрнектерде қолдана аламыз. Мұнда **C1 + C2** өрнегі **Complex** типіндегі нәтиже береді, оны тағы да осы типтегі объектімен қосуға болады. Егер **C1+C2** өрнегі басқа типтегі нәтиже беретін болса, онда **C1 + C2 + C3** құрамалы өрнегін есептеу мүмкін болмас еді. "**–**" операторы "**+**" операторы сияқты жұмыс істейді, бірақ бұл оператор үшін операндтардың берілу реттілігі маңызды болып табылады. Барлық екілік операторлар үшін операторлық әдістің бірінші параметрі – сол жақ операнд, ал екінші параметр – оң жақ операнд болып табылады. **operator-()** операторы да **Complex** типіндегі объект қайтаратын болғандықтан, **C3 + C1 - C2** өрнегін де есептеуге болады.

Унарлық операторларды асыра жүктеу

Унарлық операторлар да бинарлық операторлар тәрізді асыра жүктеледі. Бұлардың басты айырмашылығы – бір ғана операндының болуы.

Мысал 2. **Complex** класы үшін унарлық минус операторын асыра жүктеу.

```
using System;
class Complex {
   int a, b;       // санның нақты және жорамал бөліктері
   public Complex() { a = b = 0; }
   public Complex(int x, int y) { a = x; b = y; }

   // - унарлық операторын асыра жүктеу
   public static Complex operator -(Complex op1)
   {
      Complex natizhe = new Complex();
      // комплекс санның нақты және жорамал бөліктерін терістеу
      natizhe.a = -op1.a;
      natizhe.b = -op1.b;
      return natizhe;
   }
   public void Shygaru()
   {
      if(b >= 0) Console.WriteLine(a + " + " + b + "i ");
      else Console.WriteLine(a + " " + b + "i ");
   }
}
class Program {
   static void Main() {
      Complex C1 = new Complex(3, 8);
      Complex C2;
      Console.Write("C1 complex sany: ");  C1.Shygaru();
      C2 = -C1;
```

```
      Console.Write("-C1 complex sany:  "); C2.Shygaru();
      Console.ReadKey();
   }
}
```

Бұл программаны орындау нәтижесі:

```
C1 complex sany: 3 + 8i
-C1 complex sany: -3 -8i
```

Бұл мысалда асыра жүктелетін унарлық операторлардың өрістерінде операндтың теріс мәндері сақталатын жаңа объект жасалады, соңынан бұл объект операторлық әдіс арқылы қайтарылады. Мұнда операндтың өзгертілмейтініне назар аударыңыздар. Ол унарлық минус операторының кәдімгі атқаратын қызметінің сақталатынын білдіреді. Мысалы, мынадай өрнектің нәтижесі

```
a = -b
```

b операндының теріс мәні болып табылады, бірақ **b** операндының мәні өзгермейді.

C# тілінде **++** және **--** операторларының асыра жүктелуі қарапайым орындалады. Ол үшін инкременттелген немесе декременттелген мәнді қайтару жеткілікті, бірақ оны шақырған объектіні өзгертпеу керек. Бұл операторлардың префикстік және постфикстік формаларын ажырата отырып, қалғандарының барлығын компилятор орындайды.

Мысал 3. **Complex** класы үшін **operator++()** операторлық әдісі келтірілген.

```
// ++ унарлық операторын асыра жүктеу
using System;
class Complex {
   int a, b;       // санның нақты және жорамал бөліктері
   public Complex() { a = b = 0; }
   public Complex(int x, int y) { a = x; b = y; }

   // ++ унарлық операторын асыра жүктеу
   public static Complex operator ++(Complex op)
   {
      Complex natizhe = new Complex();
      // Инкременттеу нәтижесін қайтару
      natizhe.a = op.a + 1; natizhe.b = op.b + 1;
      return natizhe;
   }

   // комплекс санды шығару
   public void Shygaru()
   {
      if(b >= 0)  Console.WriteLine(a + " + " + b + "i ");
      else Console.WriteLine(a + " " + b + "i ");
   }
}
class Program {
   static void Main() {
      Complex C1 = new Complex(3, 8);
      Complex C2;
      Console.Write("C1 complex sany: ");
      C1.Shygaru();
      C2 = C1++;
      Console.Write("++ postfixtik operaciya natizhesi: ");
      C2.Shygaru();
      C2 = ++C2;
      Console.Write("++ prefixtik operaciya natizhesi:  ");
      C2.Shygaru();
      Console.ReadKey();
   }
}
```

Бұл программаны орындау нәтижесі:

```
C1 complex sany: 3 + 8i
++ postfixtik operaciya natizhesi: 3 + 8i
++ prefixtik operaciya natizhesi:  4 + 9i
```

C# құрамындағы мәліметтер типтерімен операциялар орындау

Кез келген класс және оператор үшін операторлық әдісті асыра жүктеу мүмкіндігі бар. Бұл, мысалы, класс типімен және де басқа мәліметтер типтерімен (құрамдас типтермен де) операциялар орындау кезінде қажет болады.

Мысал 4. 11.1-ші мысалда **+** операторының бір **Complex** объектісінің нақты және жорамал бөліктерін басқа бір объектінің нақты және жорамал бөліктеріне сәйкесінше қосу үшін қалай асыра жүктелетіні көрсетілген болатын. Бірақ бұл **Complex** класы үшін қосу операциясын анықтаудың жалғыз тәсілі емес. **Complex** типіндегі объектінің әрбір бөлігіне бүтін сан қосуды орындаған дұрыс болар еді. Бірақ ол үшін **+** операторын төменде көрсетілгендей тағы да бір рет асыра жүктеу керек болады.

```
using System;
// Комплекс сандармен жұмыс істеуге арналған класс
class Complex {
   int a, b;       // санның нақты және жорамал бөліктері
   public Complex() { a = b = 0; }
   public Complex(int x, int y) { a = x; b = y; }

   // + бинарлық операторын асыра жүктеу
   public static Complex operator +(Complex op1, Complex op2)
   {
      Complex natizhe = new Complex();
      //екі комплекс санның бөліктерін қосып, нәтижесін қайтару
      natizhe.a = op1.a + op2.a;
      natizhe.b = op1.b + op2.b;
      return natizhe;
   }

   public static Complex operator +(Complex op1, int op2)
   {
      Complex natizhe = new Complex();
      // комплекс санға бүтін санды қосып, нәтижесін қайтару
      natizhe.a = op1.a + op2;
      natizhe.b = op1.b + op2;
      return natizhe;
   }
   // комплекс санды шығару
   public void Shygaru()
   {
      Console.WriteLine(a + " + " + b + "i ");
   }
}
class Program {
   static void Main() {
      Complex C1 = new Complex(3, 8);
      Complex C2 = new Complex(10, 2);
      Complex C3; int x = 5;
      Console.Write("C1 complex sany: ");
      C1.Shygaru();
      Console.Write("C2 complex sany:  ");
      C2.Shygaru();
```

```
        C3 = C1 + C2; // C1 және C2 комплекс сандарын қосу
        Console.Write("C3 = C1 + C2 qosu natizhesi: ");
        C3.Shygaru();
        C3 = C1 + x; // комплекс санға бүтін санды қосу
        Console.Write("C3 = C1 + x qosu natizhesi: ");
        C3.Shygaru();
        Console.WriteLine();
        Console.ReadKey();
    }
}
```

Бұл программаны орындау нәтижесі:

```
C1 complex sany: 3 + 8i
C2 complex sany: 10 + 2i
C3 = C1 + C2 qosu natizhesi: 13 + 10i
C3 = C1 + x qosu natizhesi: 8 + 13i
```

Мұндағы операторлық әдістің екінші параметрі int типінде. Сондықтан, бұл әдісте бүтін мәнді **Complex** типіндегі объектінің әрбір өрісімен қосуға рұқсат етіледі. Мұндай операция орындала алады, өйткені бұрын айтылғандай, бинарлық операторды асыра жүктегенде, осы (асыра жүктелу керекті) класта оның бір операнды класс типінде болуы тиіс. Бірақ оның екінші операнды кез келген типте бола береді.

+ операторын асыра жүктеу, **Complex** класының пайдалы функцияларын кеңейтеді, дегенмен, оны ол аяғына дейін орындамайды. Оның себебі **operator + (Complex, int)** әдісі келесідей операцияларды орындауға мүмкіндік береді:

```
C3 = C1 + 10;
```

бірақ ол келесідей операцияларды орындай алмайды,

```
C3 = 10 + C1;
```

Бұл әдістегі екінші бүтін аргумент **+** бинарлық операторының оң жақ операндын белгілейді, бірақ жоғарыда көрсетілген жолда бүтін аргумент сол жақта көрсетіледі. Мұндай қосу операциясын орындауға рұқсат беру үшін, **+** операторын тағы да бір рет асыра жүктеу керек. Мұндағы жағдайда операторлық әдістің бірінші параметрі **int** типінде болуы керек, ал екінші параметрі – тип **Complex** типінде болуы тиіс.

Қатынас операторларын асыра жүктеу

== және **<** тәрізді қатынас операторлары да өте оңай асыра жүктеледі. Бұлар көбінесе асыра жүктелу кезінде логикалық **true** және **false** мәндерін қайтарады. Бұл мүмкіндік осылай операторларды шартты операторларда қолдану мүмкіндігін береді. Егерде нәтиже басқа типте болса, қатынас операторларын қолдану аясы кішірейеді.

5-мысал. **<** және **>** операторларын **Complex** класы арқылы асыра жүктеу керек. Бұл мысалдағы операторлар комплекс сандардың модульдерін салыстырады. Комплекс санның модулі оның нақты және жорамал бөліктерінің квадраттары қосындысын түбір астынан шығару арқылы анықталады. Егерде бір де бір оператор логикалық **true** мәнін қайтармаса, онда екі комплекс санның модульдері тең болып саналады. Әрине, бұдан басқа да реттеу алгоритмдері болуы мүмкін.

```
using System;
// Комплекс сандармен жұмыс істеуге арналған класс
class Complex {
    int a, b;       // санның нақты және жорамал бөліктері
    public Complex() { a = b = 0; }
    public Complex(int x, int y) { a = x; b = y; }

    // < қатынас операторын асыра жүктеу
    public static bool operator <(Complex op1, Complex op2)
    {
```

```
      double mop1 = Math.Sqrt(Math.Pow(op1.a,2) +
                              Math.Pow(op1.b, 2));
      double mop2 = Math.Sqrt(Math.Pow(op2.a, 2) +
                              Math.Pow(op2.b, 2));
      // екі комплекс санның модульдерін салыстыру
      if(mop1 < mop2) return true;
      else return false;
   }
   // > қатынас операторын асыра жүктеу
   public static bool operator >(Complex op1, Complex op2)
   {
      double mop1 = Math.Sqrt(Math.Pow(op1.a, 2) +
                              Math.Pow(op1.b, 2));
      double mop2 = Math.Sqrt(Math.Pow(op2.a, 2) +
                              Math.Pow(op2.b, 2));

      if (mop1 > mop2) return true;
      else return false;
   }
   // комплекс санды шығару
   public void Shygaru()
   {
      Console.WriteLine(a + " + " + b + "i ");
   }
}
class Program {
   static void Main() {
      Complex C1 = new Complex(3, 8);
      Complex C2 = new Complex(10, 2);
      Console.Write("C1 complex sany: ");
      C1.Shygaru();
      Console.Write("C2 complex sany:  ");
      C2.Shygaru();
      if(C1 > C2)       // C1 және C2 комплекс сандарын
                        // > шартына салыстыру
         Console.WriteLine("C1 complex sany C2 complex sanynan
                           ulken");
      else if (C1 < C2)  // C1 және C2 комплекс сандарын
                         // < шартына салыстыру
         Console.WriteLine("C1 complex sany C2 complex sanynan
                           kishi");
      else
         Console.WriteLine("C1 zhane C2 complex sandary ten");
      Console.WriteLine();
      Console.ReadKey();
   }
}
```

Бұл программаның нәтижесі:

```
C1 complex sany: 3 + 8i
C2 complex sany: 10 + 2i
C1 complex sany C2 complex sanynan kishi
```

Қатынас операторларының асыра жүктелуіне мынадай бір шектеу қойылады: олар жұп-жұбымен асыра жүктелуі тиіс. Егер **<** операторы асыра жүктелетін болса, онда **>** операторын да асыра жүктеу керек. Төменде жұптасып асыра жүктелетін қатынас операторлары көрсетілген.

==	!=
<	>
<=	>=

`true` және `false` операторларын асыра жүктеу

`true` және **`false`** түйінді сөздерін асыра жүктеу кезінде унарлық операторлар ретінде қолдануға болады. Бұлардың асыра жүктелу нұсқалары құрылатын кластар үшін **`true`** және **`false`** түйінді сөздерінің атқаратын қызметін анықтай алады. Бұл түйінді сөздерді нақты класс үшін унарлық операторлар ретінде асыра жүктегеннен кейін бұл класс объектіілерін **`if`**, **`while`**, **`for`**, **`do-while`** операторларын басқару мен **`?`** шартты өрнегінде қолдану мүмкіндігі пайда болады.

`true` және **`false`** операторлары жеке-жеке емес, жұбымен (парымен) асыра жүктеледі. Бұл екеуінде де **`bo`**ol типіндегі мән қайтарылады. Төменде осы унарлық операторлардың асыра жүктелуінің жалпы формасы келтірілген.

```
public static bool operator true(параметр _ типі операнд) {
   // true немесе false логикалық мәндерін қайтару
}
public static bool operator false(параметр _ типі операнд) {
   // true немесе false логикалық мәндерін қайтару
}
```

6-мысал. **`Complex`** класында **`true`** және **`false`** операторларын қолдану. Егер **`Complex`** типіндегі объектінің нақты және жорамал бөліктерінің кемінде біреуі нөлге тең болса, онда бұл объект жалған болады деп тұжырымдаймыз.

```
using System;
// Комплекс сандармен жұмыс істеуге арналған класс
class Complex {
   int a, b;       // санның нақты және жорамал бөліктері
   public Complex() { a = b = 0; }
   public Complex(int x, int y) { a = x; b = y; }

   // true операторын асыра жүктеу
   public static bool operator true(Complex op1)
   {
      if(op1.a!=0 && op1.b!=0) return true;
      else return false;
   }
   public static bool operator false(Complex op1)
   {
      if (op1.a == 0 || op1.b == 0) return true;
      else return false;
   }
   public static Complex operator --(Complex op1)
   {
      Complex natizhe = new Complex();
      natizhe.a = op1.a - 1;
      natizhe.b = op1.b - 1;
      return natizhe;
   }
   // комплекс санды шығару
   public void Shygaru()
   {
      Console.WriteLine(a + " + " + b + "i ");
   }
}

class Program {
   static void Main() {
      Complex C1 = new Complex(3, 8);
      Complex C2 = new Complex(0, 2);
      Console.Write("C1 complex sany: ");
```

```
      C1.Shygaru();
      Console.Write("C2 complex sany:  ");
      C2.Shygaru();
      if(C1)  // C1 және C2 комплекс сандарын қосу
      {
         while(C1) {
            Console.WriteLine("C1 complex sany aqiqat");
            --C1;
            C1.Shygaru();
         }
         Console.WriteLine("C1 complex sany zhalgan");
      }
      else Console.WriteLine("C1 complex sany zhalgan");

      if(C2)  // C1 және C2 комплекс сандарын қосу
         Console.WriteLine("C2 complex sany aqiqat");
      else Console.WriteLine("C2 complex sany zhalgan");
      Console.WriteLine();
      Console.ReadKey();
   }
}
```

Программаның нәтижесі:

```
C1 complex sany: 3 + 8i
C2 complex sany: 0 + 2i
C1 complex sany aqiqat
2 + 7i
C1 complex sany aqiqat
1 + 6i
C1 complex sany aqiqat
0 + 5i
C1 complex sany zhalgan
C2 complex sany zhalgan
```

If операторында **Complex** типіндегі объект **true** операторы арқылы тексеріледі. Егер мұндағы нәтиже ақиқат болса, онда **if** операторы орындалады. Ал **while** операторында **C1** объектісі циклдің әрбір қадамында декременттеледі. Сондықтан, цикл **C1** объектісінің тексерілуі ақиқат мән беріп тұрса, орындала береді. Егер **C1** объектісінің кем дегенде бір бөлігі 0-ге тең болса, онда true операторы жалған нәтиже береді де, цикл аяқталады.

Түрлендіру операторлары

Кейде белгілі бір класс объектісін басқа типтегі мәліметтері бар өрнектерде қолдануға тура келеді. Осы мақсатта кей кезде бір немесе бірнеше операторды асыра жүктеу жеткілікті болса, кейде – класс типін керекті типке қарапайым түрлендіру керек болып жатады.

C# тілінде осындай жағдайлар үшін операторлық әдістің арнайы *түрлендіру операторы* деп аталатын бір түрі қарастырылған. Осындай оператор берілген класс объектісін басқа типке түрлендіреді. Түрлендіру операторлары, типтерді түрлендіру тәртібі анықталған жағдайда, кластарды басқа типтермен бірге еркін қолдануға мүмкіндік беріп, кластар типін программалау ортасына толық енгізу ісін атқара алады.

Түрлендіру операторларының екі формасы бар: *тікелей* және *жанамалы*. Олардың жазылу жолдары:

```
public static explicit operator керекті_тип(бастапқы_тип v)
{ return мән; }
public static implicit operator керекті_тип(бастапқы_тип v)
{ return мән ; }
```

мұндағы *керекті_тип* түрлендірілетін жаңа типті білдіреді; *бастапқы_тип* – өзгертілетін бұрынғы тип; *мән* – класс үшін түрлендіруден кейін орнатылатын нақты мән. Түрлендіру операторлары *керекті_типке* көшірілген мәліметтерді қайтарады, мұнда қайтарылатын басқа типтер көрсетуге рұқсат етілмейді.

Егер түрлендіру операторы жанамалы (**implicit**) формада көрсетілсе, онда түрлендіру автоматты түрде шақырылады, бұл, мысалы, объект өрнектегі керекті типтегі мәнмен бірге пайдаланылған кездерде орын алады. Егерде түрлендіру операторы тікелей формада (**explicit**) көрсетілсе, онда түрлендіру, типтерді өзгертуді (келтіруді) орындау кезінде шақырылады. Мәліметтердің берілген бір бастапқы және керекті типтері үшін түрлендіру операторын тікелей және жанамалы түрде қатарластыра көрсетуге болмайды.

7-мысал. Түрлендіру операторын **Complex** класында қолдану. **Complex** типіндегі объектіні нақты мәнге түрлендіру (оны кейін нақты сандық өрнекте қолдану үшін) қажет болсын делік. Мұндай әрекет комплекс санның модулін есептеу арқылы орындалады. Түрлендіруді орындаудың жанамалы формасын құрайық.

```
using System;
// Комплекс сандармен жұмыс істеуге арналған класс
class Complex {
   int a, b;       // санның нақты және жорамал бөліктері
   public Complex() { a = b = 0; }
   public Complex(int x, int y) { a = x; b = y; }

   // түрлендіру операторын асыра жүктеу
   public static implicit operator double(Complex op1)
   {
      return Math.Sqrt(Math.Pow(op1.a,2) + Math.Pow(op1.b,2));
   }

   // комплекс санды шығару
   public void Shygaru()
   {
      Console.WriteLine(a + " + " + b + "i ");
   }
}
class Program {
   static void Main() {
      Complex C1 = new Complex(3, 8);
      Complex C2 = new Complex(0, 2);
      Console.Write("C1 complex sany: ");
      C1.Shygaru();
      Console.Write("C2 complex sany:  ");
      C2.Shygaru();
      double x = C1;
      Console.WriteLine("C1 complex sanynyn moduli = " + x);
      x = x + C2;
      Console.WriteLine("C1 zhane C2 complex sandarynyn
                         modulderinin kosyndysy = " + x);
      Console.WriteLine();
      Console.ReadKey();
   }
}
```

Бұл программаның нәтижесі:

```
C1 complex sany: 3 + 8i
C2 complex sany: 0 + 2i
C1 complex sanynyn moduli = 8.54400374531753
C1 zhane C2 complex sandarynyn modulderinin
kosyndysy = 10.5440037453175
```

Бұл мысалда **Complex** типіндегі объект нақты сандық типте (**x = C1**) пайдаланылған кезде, операторды түрлендіру ісі орындалады. Бірақ жұмыс барысында өрнекті есептеу үшін **double** типіне түрлендіру қажет етілмесе, онда операторды түрлендіру әрекеті шақырылмайды.

Жанамалы түрлендіру операторы келесідей жағдайларда:

А) өрнектерде типтерді түрлендіру қажет болғанда;

Ә) әдіске объект беру кезінде;

Б) меншіктеу әрекеті орындалып, керекті типке тікелей көшу ісін атқару керек болған сәттерде автоматты түрде жүзеге асады.

Басқа жағынан алатын болсақ, тек тікелей типті өзгерту ісі керек болған кездерде шақырылатын тікелей түрлендіру операторын да жасауға болады. Мұндайда тікелей түрлендіру операторы автоматты түрде шақырылмайды.

8-мысал. Тікелей түрлендіру операторын **Complex** класында тікелей қолдану нұсқасын құрайық.

```
using System;
// Комплекс сандармен жұмыс істеуге арналған класс
class Complex {
   int a, b;       // санның нақты және жорамал бөліктері
   public Complex() { a = b = 0; }
   public Complex(int x, int y) { a = x; b = y; }

   // тікелей түрлендіру операторын асыра жүктеу
   public static explicit operator double(Complex op1)
   {
      return Math.Sqrt(Math.Pow(op1.a,2) + Math.Pow(op1.b,2));
   }

   // комплекс санды шығару
   public void Shygaru()
   {
      Console.WriteLine(a + " + " + b + "i ");
   }
}
class Program {
   static void Main() {
      Complex C1 = new Complex(3, 8);
      Complex C2 = new Complex(0, 2);
      Console.Write("C1 complex sany: ");
      C1.Shygaru();
      Console.Write("C2 complex sany:  ");
      C2.Shygaru();
      double x = (double)C1;
      Console.WriteLine("C1 complex sanynyng
                                  moduli = {0:#.##}", x);
      x = x + (double)C2;
      Console.WriteLine("C1 zhane C2 complex sandarynyng
                modulderining qosyndysy = {0:#.##}", x);
      Console.WriteLine();
      Console.ReadKey();
   }
}
```

Бұл программаның нәтижесі:

```
C1 complex sany: 3 + 8i
C2 complex sany: 0 + 2i
C1 complex sanynyng moduli = 8.54
C1 zhane C2 complex sandarynyng modulderining kosyndysy = 10.54
```

Тікелей немесе жанамалы түрлендірулерді операторларын таңдау кезінде бұл шектеулерге қосымша бірсыпыра ұсыныстар беріледі. Жанамалы түрлендірулердің барлық артықшылықтарына қарамастан, оларды тек түрлендіруде қателер болмайтындай жағдайларда қолдану керек.

Қателерді болдырмас үшін жанамалы түрлендірулер келесідей шарттар орындалатын кездерде ғана ұйымдастырылуы тиіс.

1) алдымен, мысалы, қысқарту, толып кету немесе таңбаны жоғалту нәтижелерінде мәлімет жоғалмайтындай болуы керек.
2) түрлендіру ерекше жағдайды туындатпауы тиіс.

Егер де жанамалы түрлендіру осы екі шартты қанағаттандырмаса, онда тікелей түрлендіруді таңдау қажет.