

Министерство цифрового развития, связи и массовых коммуникаций  
Российской Федерации  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования «Сибирский государственный университет  
телекоммуникаций и информатики» (СибГУТИ)

Кафедра прикладной математики и кибернетики

Лабораторная работа №14  
«Полином»

Выполнил:  
Студент группы ИП-013  
Копытина Т.А.  
Работу проверил:  
ассистент кафедры ПМиК  
Агалаков А.А.

Новосибирск 2023 г.

## Содержание

1. Задание .....	3
2. Исходный код программы.....	4
2.1. Код программы .....	4
2.2. Код тестов.....	7
3. Результаты модульных тестов .....	10
4. Вывод .....	11

## 1. Задание

1. Реализовать тип «полином», в соответствии с приведенной ниже спецификацией.
  2. Протестировать каждую операцию, определенную на типе данных, используя средства модульного тестирования.
  3. Если необходимо, предусмотрите возбуждение исключительных ситуаций.
- Спецификация абстрактного типа данных «Полином».

ADT TPolу

### Данные

Полиномы Tpolу - это неизменяемые полиномы с целыми коэффициентами.

### Операции

Операции могут вызываться только объектом «полином» (тип TPolу), указатель на который передаётся в них по умолчанию. При описании операций этот объект в разделе «Вход» не указывается.

Спецификация абстрактного типа данных Одночлен.

ADT TMember

### Данные

Одночлен TMember - это изменяемые одночленные полиномы с целыми коэффициентами. Коэффициент и степень хранятся в полях целого типа FCoeff и FDegree соответственно.

### Операции

Операции могут вызываться только объектом «одночлен» (тип TMember), указатель на который передаётся в них по умолчанию. При описании операций этот объект в разделе «Вход» не указывается.

## 2. Исходный код программы

### 2.1. Код программы

#### TPolynom.cs

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Linq;

namespace lab14
{
    public class TPolynom
    {
        public SortedSet<TMember>
Members;
        public TPolynom()
        {
            Members = new
SortedSet<TMember>();
            Members.Add(new TMember(0,
0));
        }
        public TPolynom Add(TPolynom tp)
        {
            TPolynom newtpoly = new
TPolynom();
            foreach (TMember mem in
tp.Members)
            {
                newtpoly.Members.Add(new
TMember(mem.FCoeff, mem.FDegree));
            }
            foreach (TMember mem in
this.Members)
            {
                newtpoly.Members.Add(new
TMember(mem.FCoeff, mem.FDegree));
            }
            return newtpoly;
        }
        public TPolynom Mul(TPolynom tp)
        {
            TPolynom newtpoly = new
TPolynom();
            foreach (TMember mem in
tp.Members)
            {
                foreach (TMember newmem
in this.Members)
                {
                    newtpoly.Members.Add(new
TMember(newmem.FCoeff * mem.FCoeff,
newmem.FDegree + mem.FDegree));
                }
            }
            return newtpoly;
        }
        public TPolynom Res(TPolynom tp)
        {
            TPolynom newtpoly = new
TPolynom();
            foreach (TMember mem in
tp.Members)
            {
                newtpoly.Members.Add(new
TMember(-mem.FCoeff, mem.FDegree));
            }
            foreach (TMember mem in
this.Members)
            {
                newtpoly.Members.Add(new
TMember(mem.FCoeff, mem.FDegree));
            }
            return newtpoly;
        }
        public TPolynom Rev()
        {
            TPolynom newtpoly = new
TPolynom();
            foreach (TMember mem in
this.Members)
            {
                newtpoly.Members.Add(new
TMember(-mem.FCoeff, mem.FDegree));
            }
            return newtpoly;
        }
        public int RetDegree()
        {
            return
Members.Last().FDegree;
        }
        public int RetCoeff(int n)
        {
            try { return Members.Single(x
=> x.FDegree == n).FCoeff; }
            catch
(InvalidOperationException) { return 0; }
        }
        public override bool
Equals(object tp)
        {
            if
(((TPolynom)tp).Members.SequenceEqual(thi
s.Members)) return true;
            else return false;
        }
        public TPolynom Diff()
        {
            TPolynom newtpoly = new
TPolynom();
            foreach (TMember mem in
this.Members)
            {
                newtpoly.Members.Add(new
TMember(mem.FCoeff, mem.FDegree).Diff());
            }
            return newtpoly;
        }
        public double Calculate(double a)
```

```

    {
        double an = 0.0;
        foreach (TMember mem in
this.Members)
        {
            an += mem.Calculate(a);
        }
        return an;
    }
    public void Clear()
    {
        Members = new
SortedSet<TMember>
        {
            new TMember(0, 0)
        };
    }
    public Tuple<int, int>
ElementAt(int i)
    {
        try { return
Tuple.Create(this.Members.Reverse().Eleme

```

```

ntAt(i).FCoeff,
this.Members.Reverse().ElementAt(i).FDegr
ee); }
        catch
        (ArgumentOutOfRangeException) { return
Tuple.Create(0, 0); }
    }
    public string Show()
    {
        string str = "";
        foreach (TMember x in
Members.Reverse())
        {
            str += (x.FCoeff > 0) ?
"+" : "";
            str += x.TMember2Str();
        }
        return str.TrimStart('+');
    }
}

```

## TMember.cs

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

namespace lab14
{
    public class TMember : IComparable<TMember>
    {
        private int fcoeff;
        private int fdegree;
        public int FCoeff
        {
            get { return fcoeff; }
            set {
                if (value == 0)
                    fdegree = 0;
                fcoeff = value;
            }
        }
        public int FDegree
        {
            get { return fdegree; }
            set {
                if (fcoeff == 0)
                    fdegree = 0;
                else fdegree = value;
            }
        }
        public TMember(int coeff = 0, int degree = 0)
        {
            FCoeff = coeff;
            FDegree = degree;
        }
        public override bool Equals(object tmem)
        {
            if (((TMember)tmem).FCoeff == this.FCoeff) && (((TMember)tmem).FDegree == this.FDegree))
                return true;
            else
                return false;
        }
        public TMember Diff()
        {
            return new TMember() { FCoeff = (this.FCoeff * this.FDegree), FDegree = this.FDegree - 1 };
        }
        public double Calculate(double a)
        {
            return this.FCoeff * Math.Pow(a, this.FDegree);
        }
        public string TMember2Str()
        {
            return this.FCoeff == 0 ? "" : $"{this.FCoeff}x^{this.FDegree}";
        }

        public int CompareTo(TMember other)
        {
            if (this.FDegree.CompareTo(other.FDegree) != 0)
                return this.FDegree.CompareTo(other.FDegree);
            else
            {
                other.FCoeff += this.FCoeff;
                return 0;
            }
        }
    }
}
```

## 2.2. Код тестов

### UnitTest1.cs

```
using
Microsoft.VisualStudio.TestTools.UnitTesting;
using lab14;

namespace TestProject1
{
    [TestClass]
    public class UnitTest1
    {
        [TestMethod]
        public void TestAdd1()
        {
            TPolynom tpoly = new
            TPolynom();
            tpoly.Members.Add(new
            TMember(1, 0));
            Assert.AreEqual(tpoly.Show(),
            "1x^0");
        }

        [TestMethod]
        public void TestAdd2()
        {
            TPolynom tpoly = new
            TPolynom();
            tpoly.Members.Add(new
            TMember(1, 0));
            tpoly.Members.Add(new
            TMember(1, 1));
            Assert.AreEqual(tpoly.Show(),
            "1x^1+1x^0");
        }

        [TestMethod]
        public void TestMul1()
        {
            TPolynom tpoly = new
            TPolynom();
            TPolynom newtpoly = new
            TPolynom();
            tpoly.Members.Add(new
            TMember(1, 0));
            tpoly.Members.Add(new
            TMember(1, 1));
            newtpoly.Members.Add(new
            TMember(1, 0));
            newtpoly.Members.Add(new
            TMember(1, 1));

            TPolynom addpoly =
            tpoly.Add(newtpoly);
            addpoly =
            tpoly.Mul(newtpoly);

            Assert.AreEqual(addpoly.Show(),
            "1x^2+2x^1+1x^0");
        }

        [TestMethod]
        public void TestMul2()
        {
            TPolynom tpoly = new
            TPolynom();
            TPolynom newtpoly = new
            TPolynom();
            tpoly.Members.Add(new
            TMember(1, 0));
            tpoly.Members.Add(new
            TMember(1, 2));
            newtpoly.Members.Add(new
            TMember(2, 0));
            newtpoly.Members.Add(new
            TMember(1, 1));

            TPolynom addpoly =
            tpoly.Add(newtpoly);
            addpoly =
            tpoly.Mul(newtpoly);

            Assert.AreEqual(addpoly.Show(),
            "1x^3+2x^2+1x^1+2x^0");
        }

        [TestMethod]
        public void TestClear()
        {
            TPolynom tpoly = new
            TPolynom();
            tpoly.Members.Add(new
            TMember(1, 0));
            tpoly.Clear();
            Assert.AreEqual(tpoly.Show(),
            "");
        }

        [TestMethod]
        public void TestCalc1()
        {
            TPolynom tpoly = new
            TPolynom();
            tpoly.Members.Add(new
            TMember(1, 2));
            tpoly.Members.Add(new
            TMember(3, 3));
            tpoly.Members.Add(new
            TMember(4, 2));

            Assert.AreEqual(tpoly.Calculate(2), 44);
        }

        [TestMethod]
        public void TestCalc2()
        {
            TPolynom tpoly = new
            TPolynom();
```

```

        tpoly.Members.Add(new
TMember(1, 2));
        tpoly.Members.Add(new
TMember(3, 0));
        tpoly.Members.Add(new
TMember(4, 0));

Assert.AreEqual(tpoly.Calculate(2), 11);
    }

    [TestMethod]
    public void TestEquals1()
    {
        TPolynom tpoly1 = new
TPolynom();
        tpoly1.Members.Add(new
TMember(1, 2));

        TPolynom tpoly2 = new
TPolynom();
        tpoly2.Members.Add(new
TMember(1, 2));

Assert.AreEqual(tpoly1.Equals(tpoly2),
true);
    }

    [TestMethod]
    public void TestEquals2()
    {
        TPolynom tpoly1 = new
TPolynom();
        tpoly1.Members.Add(new
TMember(1, 2));

        TPolynom tpoly2 = new
TPolynom();
        tpoly2.Members.Add(new
TMember(0, 2));

Assert.AreEqual(tpoly1.Equals(tpoly2),
false);
    }

    [TestMethod]
    public void TestDiff1()
    {
        TPolynom tpoly1 = new
TPolynom();
        tpoly1.Members.Add(new
TMember(1, 3));

Assert.AreEqual(tpoly1.Diff().Show(),
"3x^2");
    }

    [TestMethod]
    public void TestElementAt()
    {

```

```

        TPolynom tpoly1 = new
TPolynom();
        tpoly1.Members.Add(new
TMember(11, 11));
        tpoly1.Members.Add(new
TMember(22, 22));

        Assert.AreEqual(new
System.Tuple<int, int>(11, 11),
tpoly1.ElementAt(1));
    }

    [TestMethod]
    public void TestRetDegree()
    {
        TPolynom tpoly1 = new
TPolynom();
        tpoly1.Members.Add(new
TMember(11, 11));
        tpoly1.Members.Add(new
TMember(22, 22));

Assert.AreEqual(tpoly1.RetDegree(), 22);
    }

    [TestMethod]
    public void TestRetCoeff()
    {
        TPolynom tpoly1 = new
TPolynom();
        tpoly1.Members.Add(new
TMember(11, 11));
        tpoly1.Members.Add(new
TMember(22, 33));

Assert.AreEqual(tpoly1.RetCoeff(33), 22);
    }

    [TestMethod]
    public void TestRetCoeff2()
    {
        TPolynom tpoly1 = new
TPolynom();
        tpoly1.Members.Add(new
TMember(11, 11));
        tpoly1.Members.Add(new
TMember(22, 33));
        tpoly1.Members.Add(new
TMember(20, 33));

Assert.AreEqual(tpoly1.RetCoeff(33), 42);
    }

    [TestMethod]
    public void TestRev()
    {
        TPolynom tpoly1 = new
TPolynom();
        tpoly1.Members.Add(new
TMember(11, 1));
        tpoly1.Members.Add(new
TMember(22, 3));

```



```
        tpoly1.Members.Add(new
TMember(20, 2));
    }
}
```

```
Assert.AreEqual(tpoly1.Rev().Show(), "-
22x^3-20x^2-11x^1");
```

### 3. Результаты модульных тестов

Обозреватель тестов			
<div>▶ ▶ ▶ 🔍 🧪 15 ✅ 15 ❌ 0 📄 [≡] [⊞] [⊞] ⚙️</div>			
Запуск тестов завершен: тестов запущено в 294 мс: 15 (пройдено: 15, не пройдено: 0, пропущено: 0).			
Тестирование ▾	Длительн...	Признаки	Сообщение об ошибке
▲ ✅ TestProject1 (15)	67 мс		
▲ ▲ ✅ TestProject1 (15)	67 мс		
▲ ▲ ▲ ✅ UnitTest1 (15)	67 мс		
✅ TestRev	18 мс		
✅ TestRetDegree	9 мс		
✅ TestRetCoeff2	< 1 мс		
✅ TestRetCoeff	< 1 мс		
✅ TestMul2	< 1 мс		
✅ TestMul1	< 1 мс		
✅ TestEquals2	< 1 мс		
✅ TestEquals1	< 1 мс		
✅ TestElementAt	< 1 мс		
✅ TestDiff1	< 1 мс		
✅ TestClear	< 1 мс		
✅ TestCalc2	< 1 мс		
✅ TestCalc1	1 мс		
✅ TestAdd2	< 1 мс		
✅ TestAdd1	39 мс		

#### **4. Вывод**

По итогам данной лабораторной работе были сформированы практические навыки реализации типов данных в соответствии с заданной спецификацией с помощью классов C# и их модульного тестирования.