Федеральное государственное бюджетное образовательное учреждение высшего образования
«Сибирский государственный университет телекоммуникаций и информатики»
(СибГУТИ)

Кафедра прикладной математики и кибернетики

Лабораторная работа № 12
по дисциплине «Современные технологии программирования»

Выполнил:
студент группы ИП-712
Алексеев Степан
Владимирович
ФИО студента

Работу проверил:
ассистент кафедры Агалаков А.А.
ФИО преподавателя

Новосибирск 2020 г.

Оглавление

ЗАДАНИЕ

1. Реализовать тип «полином», в соответствии с приведенной ниже спецификацией.

2. Протестировать каждую операцию, определенную на типе данных, используя средства модульного тестирования.

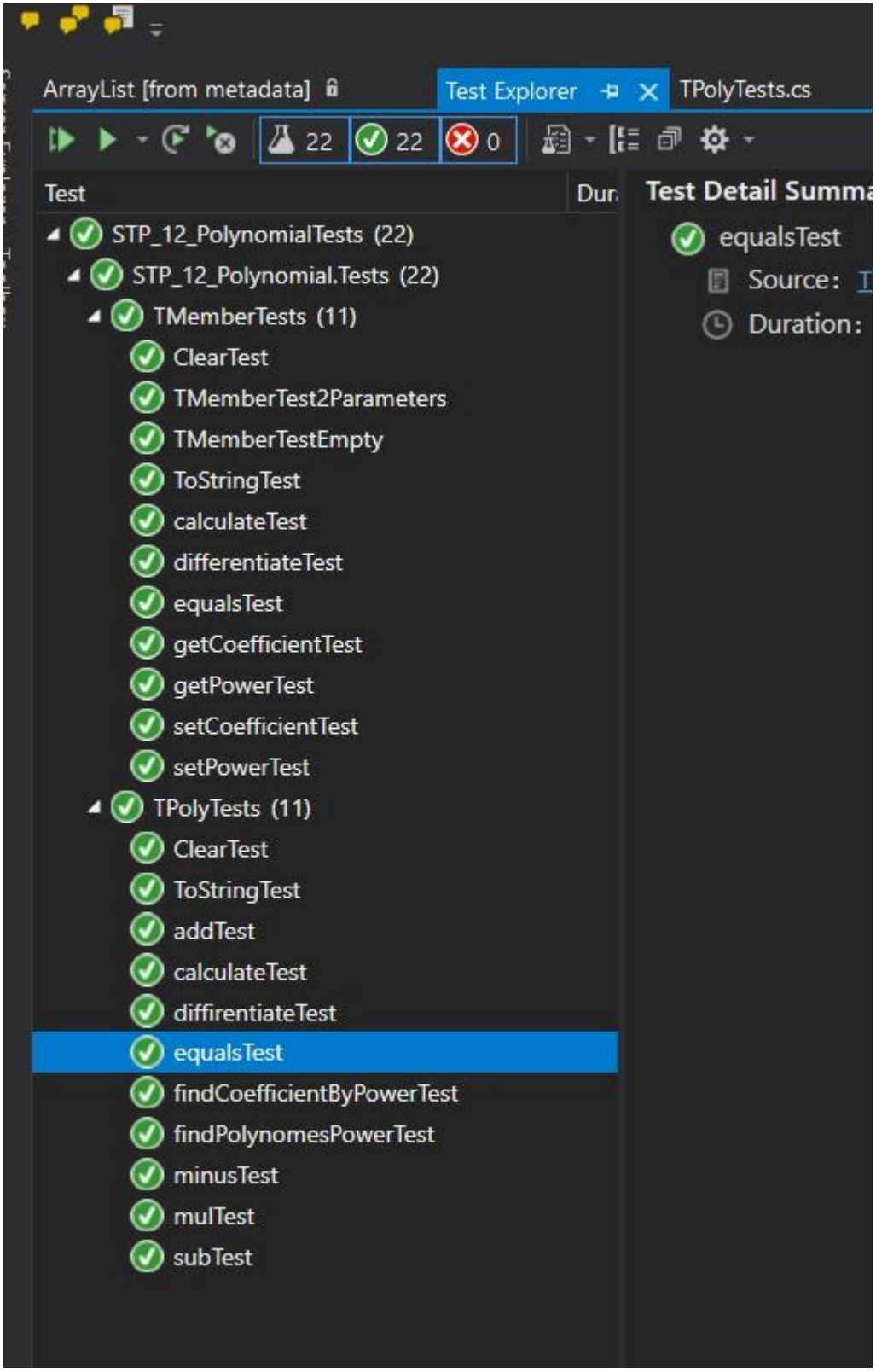
3. Если необходимо, предусмотрите возбуждение исключительных ситуаций.

ТЕСТОВЫЕ НАБОРЫ ДАННЫХ

```
ArrayList arr = new ArrayList();
        arr.Add(new TMember(2, 5));
        arr.Add(new TMember(3, 2));
        arr.Add(new TMember(4, 8));
        TPoly tp = new TPoly(arr);

        ArrayList arr2 = new ArrayList();
        arr2.Add(new TMember(3, 2));
        arr2.Add(new TMember(2, 5));
        arr2.Add(new TMember(4, 8));
        TPoly tp2 = new TPoly(arr2);
        Assert.IsTrue(tp.equals(tp2));
```

ДЕМОНСТРАЦИЯ РАБОТЫ ПРОГРАММЫ



ВЫВОД

Освоил работу с библиотечным классом ArrayList на новом уровне. По новому начал смотреть на ООП благодаря использованию отдельного класса для одночленного полинома. Вынес все операции с одночленным полиномом в отдельный класс, а работу с полиномом организовал в своём классе.

ПРИЛОЖЕНИЕ

Листинг 1. TMember.cs

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

namespace STP_12_Polynomial
{
    public class TMember : IComparable
    {
        private int FCoeff;//FCoeff - целый коэффициент
        private int FDegree;// FDegree - степень одночленного
полинома
        public TMember(int c, int n)
        {
            this.FCoeff = c;
            this.FDegree = n;
        }
        public TMember()
        {
            this.FCoeff = 0;
            this.FDegree = 0;
        }
        public int getPower()
        //вообще это для поиска наибольшей степени полинома, в
котором может быть несколько
        //членов(и несколько степеней, соответственно)
        {
            return FDegree;
        }
        public int getCoefficient()
        {
            return FCoeff;
        }
        public void Clear(ref TMember t)
        {
            if (t == null) throw new NullPointer();
            t.FCoeff = 0;
            t.FDegree = 0;
        }
        public void setPower(int n)
        {
            this.FDegree = n;
        }
        public void setCoefficient(int c)
        {
            this.FCoeff = c;
```

```
        }
        public bool equals(TMember t)
        {
            if (FCoeff == t.getCoefficient() && FDegree ==
t.getPower())
                return true;
            else return false;
        }
        public TMember differentiate()
        {
            if (FDegree == 0)
            {
                FCoeff = 0;
                FDegree = 0;
            }
            else
            {
                FCoeff *= FDegree;
                FDegree -= 1;
            }
            return this;
            /* int newCoeff = FCoeff *= FDegree;
             int newDegree = FDegree -= 1;
             if (FDegree == 0)
             {
                 newCoeff = 0;
                 newDegree = 0;
             }
             return new TMember(newCoeff, newDegree);*/
        }
        public double calculate(double x)
        {
            return FCoeff * Math.Pow(x, FDegree);
        }
        override
        public string ToString()
        {
            return FCoeff.ToString() + "*x^" +
FDegree.ToString();
        }

        public int CompareTo(object obj)
        {
            return FDegree.CompareTo(obj);
        }
        public class WrongInput : Exception
        {
            public WrongInput()
            {
                Console.WriteLine("wrong input");
            }
```

```
        }
        public class NullPointer : Exception
        {
            public NullPointer()
            {
                Console.WriteLine("wrong link");
            }
        }

    }
}
```

Листинг 2. TPoly.cs

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

namespace STP_12_Polynomial
{
    public class TPoly
    {
        ArrayList arr;
        public TPoly()
        {
            arr = new ArrayList();
            arr.Add(new TMember());
        }
        public TPoly(int c, int n)
        {
            arr = new ArrayList();
            arr.Add(new TMember(c, n));
        }
        public TPoly(ArrayList a)
        {
            arr = new ArrayList();
            arr.AddRange(a);
        }
        static void Main(string[] args)
        {
            TPoly tp = new TPoly();
            tp.arrInit();
            tp.ToString();
            tp.printPoly();
        }
        public int findPolynomesPower()
        {
            int max = 0;
```

```
            for (int i = 0; i < arr.Count; i++)
            {
                int current = (((TMember)arr[i]).getPower());
                if (max < current) max = current;
            }
            return max;
        }
        public int findCoefficientByPower(int n)
        {
            for (int i = 0; i < arr.Count; i++)
            {
                int currentPower =
(((TMember)arr[i]).getPower());
                if (n == currentPower) return
(((TMember)arr[i]).getCoefficient());
            }
            return 0;//=> n > polynome's degree
        }
        public TPoly Clear()
        {
            return new TPoly();
        }
        public void shortenAndSortPolinomial(ref ArrayList arr)
        {//ЭТОТ КОД СОКРАЩАЕТ ПОЛИНОМ(складывает элементы с
одинаковыми степенями)
            for (int i = 0; i < arr.Count; i++)
            {
                for (int j = i + 1; j < arr.Count; j++)
                {
                    int pow1 = ((TMember)arr[i]).getPower();
                    int pow2 = ((TMember)arr[j]).getPower();
                    if (pow1 == pow2)
                    {
                        int coef2 =
((TMember)arr[j]).getCoefficient();
                        int coef1 =
((TMember)arr[i]).getCoefficient();
                        coef1 += coef2;
                        ((TMember)arr[i]).setCoefficient(coef1);
                        ((TMember)arr[j]).setCoefficient(0);
                    }
                }
            }
            for (int i = 0; i < arr.Count; i++)
            {
                int coef1 = ((TMember)arr[i]).getCoefficient();
                if (coef1 == 0)
                {
                    arr.RemoveAt(i);
                    i--;//После удаления i будет указывать на 1
число дальше, чем надо, поэтому уменьшаю его
```

```
                }
            }
            int tempPow;
            int tempCoef;
            for (int i = 0; i < arr.Count - 1; i++)
            {
                int pow1 = ((TMember)arr[i]).getPower();
                int coef1 = ((TMember)arr[i]).getCoefficient();
                for (int j = i + 1; j < arr.Count; j++)
                {
                    int coef2 =
((TMember)arr[j]).getCoefficient();
                    int pow2 = ((TMember)arr[j]).getPower();
                    if (pow1 > pow2)
                    {
                        tempPow = pow1;
                        ((TMember)arr[i]).setPower(pow2);
                        ((TMember)arr[j]).setPower(pow1);

                        tempCoef = coef1;
                        ((TMember)arr[i]).setCoefficient(coef2);
                        ((TMember)arr[j]).setCoefficient(coef1);
                    }
                }
            }
        }
        public TPoly add(TPoly t)
        {
            arr.AddRange(t.arr);
            shortenAndSortPolinomial(ref arr);
            return this; // new TPoly(arr);//можно было и не
возвращать ничего, но в задании написано, что надо возвратить
        }
        public TPoly mul(TPoly t)//на самом деле здесь нужны
алгебраические операции уже...
        {
            ArrayList arrNew = new ArrayList();
            for (int i = 0; i < arr.Count; i++)
            {
                int coef1 = ((TMember)arr[i]).getCoefficient();
                int pow1 = ((TMember)arr[i]).getPower();
                for (int j = 0; j < t.arr.Count; j++)
                {
                    int coef2 =
((TMember)t.arr[j]).getCoefficient();
                    int pow2 = ((TMember)t.arr[j]).getPower();
                    arrNew.Add(new TMember(coef1 * coef2, pow1 +
pow2));
                }
            }
            shortenAndSortPolinomial(ref arrNew);
```

```
            return new TPoly(arrNew);
        }
        public TPoly sub(TPoly q)
        {
            for (int i = 0; i < q.arr.Count; i++)
            {

((TMember)q.arr[i]).setCoefficient(((TMember)q.arr[i]).getCoeffi
cient() * (-1));
            }//домножил все коэффициенты полинома q на -1
            arr.AddRange(q.arr);
            shortenAndSortPolinomial(ref arr);
            return this;
        }
        public TPoly minus()
        {
            for (int i = 0; i < arr.Count; i++)
            {

((TMember)arr[i]).setCoefficient(((TMember)arr[i]).getCoefficien
t() * (-1));
            }
            return this;
        }
        public bool equals(TPoly t)
        {
            shortenAndSortPolinomial(ref arr);
            shortenAndSortPolinomial(ref t.arr);
            if (arr.Count != t.arr.Count) return false;
            else
            {
                for (int i = 0; i < arr.Count; i++)
                {
                    int pow1 = ((TMember)arr[i]).getPower();
                    int coef1 =
((TMember)arr[i]).getCoefficient();
                    int coef2 =
((TMember)t.arr[i]).getCoefficient();
                    int pow2 = ((TMember)t.arr[i]).getPower();
                    if (pow1 != pow2 || coef1 != coef2) return
false;
                }
            }
            return true;
        }
        public TPoly diffirentiate()
        {
            for (int i = 0; i < arr.Count; i++)
            {
                ((TMember)arr[i]).differentiate();
```

```
            }
            return this;
        }
        public double calculate(double x)
        {
            double result = 0;
            for (int i = 0; i < arr.Count; i++)
            {
                result += ((TMember)arr[i]).calculate(x);
            }
            return result;
        }

        public void arrInit()
        {
            arr = new ArrayList();
            arr.Add(new TMember(2, 5));
            arr.Add(new TMember(3, 10));
            arr.Add(new TMember(4, 8));

        }
        public string ToString()
        {
            string poly = "";
            for (int i = 0; i < arr.Count; i++)
            {
                poly += "(" + arr[i].ToString() + ") + ";

            }
            return poly.Substring(0, poly.Length - 3);
        }
        public void printPoly()
        {
            for (int i = 0; i < arr.Count; i++)
            {
                Console.Write(arr[i].ToString() + " + ");
            }
            Console.ReadLine();
        }


    }
}
```

Листинг 3. TMemberTests.cs

```
using Microsoft.VisualStudio.TestTools.UnitTesting;
using STP_12_Polynomial;
using System;
using System.Collections;
using System.Collections.Generic;
```

```
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace STP_12_Polynomial.Tests
{
    [TestClass()]
    public class TMemberTests
    {
        [TestMethod()]
        public void TMemberTestEmpty()
        {
            ArrayList arr = new ArrayList();
            arr.Add(new TMember());
            TMember tm1 = (TMember)arr[0];
            Assert.AreEqual(tm1.ToString(), "0*x^0");
        }

        [TestMethod()]
        public void TMemberTest2Parameters()
        {
            ArrayList arr = new ArrayList();
            arr.Add(new TMember(12, 6));
            TMember tm1 = (TMember)arr[0];
            Assert.AreEqual(tm1.ToString(), "12*x^6");
        }

        [TestMethod()]
        public void getPowerTest()
        {
            ArrayList arr = new ArrayList();
            arr.Add(new TMember(3, 2));
            TMember tm1 = (TMember)arr[0];
            int t = tm1.getPower();
            Assert.AreEqual(t, 2);
        }

        [TestMethod()]
        public void getCoefficientTest()
        {
            ArrayList arr = new ArrayList();
            arr.Add(new TMember(3, 2));
            TMember tm1 = (TMember)arr[0];
            int t = tm1.getCoefficient();
            Assert.AreEqual(t, 3);
        }

        [TestMethod()]
        public void ClearTest()
        {
            ArrayList arr = new ArrayList();
```

```
        arr.Add(new TMember(3, 2));
        TMember tm1 = (TMember)arr[0];
        tm1.Clear(ref tm1);
        Assert.AreEqual(tm1.ToString(), "0*x^0");
    }

    [TestMethod()]
    public void setPowerTest()
    {
        ArrayList arr = new ArrayList();
        arr.Add(new TMember(3, 2));
        TMember tm1 = (TMember)arr[0];
        tm1.setPower(5);
        Assert.AreEqual(tm1.getPower(), 5);
    }

    [TestMethod()]
    public void setCoefficientTest()
    {
        ArrayList arr = new ArrayList();
        arr.Add(new TMember(3, 2));
        TMember tm1 = (TMember)arr[0];
        tm1.setCoefficient(5);
        Assert.AreEqual(tm1.getCoefficient(), 5);
    }

    [TestMethod()]
    public void equalsTest()
    {
        ArrayList arr = new ArrayList();
        arr.Add(new TMember(3, 2));
        arr.Add(new TMember(3, 2));
        TMember tm1 = (TMember)arr[0];
        TMember tm2 = (TMember)arr[1];
        bool x = tm1.equals(tm2);
        Assert.IsTrue(x);
    }

    [TestMethod()]
    public void differentiateTest()
    {
        ArrayList arr = new ArrayList();
        arr.Add(new TMember(2, 5));
        arr.Add(new TMember(3, 2));
        arr.Add(new TMember(4, 8));
        TMember tm = (TMember)arr[2];
        tm.differentiate();
        Assert.AreEqual(tm.ToString(), "32*x^7");
    }

    [TestMethod()]
```

```
        public void calculateTest()
        {
            ArrayList arr = new ArrayList();
            arr.Add(new TMember(2, 5));
            arr.Add(new TMember(3, 2));
            TMember tm = (TMember)arr[1];
            double res = tm.calculate(3.0);
            Assert.AreEqual(res, 27);
        }

        [TestMethod()]
        public void ToStringTest()
        {
            ArrayList arr = new ArrayList();
            arr.Add(new TMember(2, 5));
            string str = arr[0].ToString();
            Assert.AreEqual(str, "2*x^5");
        }

    }
}
```

Листинг 4. TPolyTests.cs

```
using Microsoft.VisualStudio.TestTools.UnitTesting;
using STP_12_Polynomial;
using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace STP_12_Polynomial.Tests
{
    [TestClass()]
    public class TPolyTests
    {
        [TestMethod()]
        public void findPolynomesPowerTest()
        {
            ArrayList arr = new ArrayList();
            arr.Add(new TMember(2, 5));
            arr.Add(new TMember(3, 12));
            arr.Add(new TMember(4, 8));
            TPoly tp = new TPoly(arr);
           int i = tp.findPolynomesPower();
            Assert.AreEqual(i, 12);
        }
```

```
        [TestMethod()]
        public void findCoefficientByPowerTest()
        {
            ArrayList arr = new ArrayList();
            arr.Add(new TMember(2, 5));
            arr.Add(new TMember(3, 12));
            arr.Add(new TMember(4, 8));
            TPoly tp = new TPoly(arr);
            int i = tp.findCoefficientByPower(8);
            Assert.AreEqual(i, 4);
        }

        [TestMethod()]
        public void ClearTest()
        {
            ArrayList arr = new ArrayList();
            arr.Add(new TMember(2, 5));
            arr.Add(new TMember(3, 12));
            arr.Add(new TMember(4, 8));
            TPoly tp = new TPoly(arr);
           tp = tp.Clear();
            Assert.AreEqual(tp.ToString(), "(0*x^0)");
        }

        [TestMethod()]
        public void addTest()
        {
            ArrayList arr = new ArrayList();
            arr.Add(new TMember(2, 5));
            arr.Add(new TMember(3, 2));
            arr.Add(new TMember(4, 8));
            TPoly tp = new TPoly(arr);
            ArrayList arr2 = new ArrayList();
            arr2.Add(new TMember(2, 5));
            arr2.Add(new TMember(3, 2));
            arr2.Add(new TMember(4, 8));
            arr2.Add(new TMember(71, 15));
            TPoly tp2 = new TPoly(arr2);
            tp.add(tp2);
            string str = tp.ToString();
            Assert.AreEqual("(6*x^2) + (4*x^5) + (8*x^8) +
(71*x^15)", str);

        }

        [TestMethod()]
        public void mulTest()
        {
            ArrayList arr1 = new ArrayList();
            arr1.Add(new TMember(2, 3));
            arr1.Add(new TMember(-3, 4));
```

```
            TPoly tp1 = new TPoly(arr1);

            ArrayList arr2 = new ArrayList();
            arr2.Add(new TMember(3, 5));
            arr2.Add(new TMember(5, 10));
            TPoly tp2 = new TPoly(arr2);
            TPoly tp3 = tp1.mul(tp2);

            Assert.AreEqual("(6*x^8) + (-9*x^9) + (10*x^13) + (-
15*x^14)", tp3.ToString());
        }

        [TestMethod()]
        public void subTest()
        {
            ArrayList arr = new ArrayList();
            arr.Add(new TMember(2, 5));
            arr.Add(new TMember(3, 2));
            arr.Add(new TMember(4, 8));
            //TMember tm = (TMember)arr[2];
            TPoly tp = new TPoly(arr);
            ArrayList arr2 = new ArrayList();
            arr2.Add(new TMember(2, 5));
            arr2.Add(new TMember(3, 2));
            arr2.Add(new TMember(3, 8));
            arr2.Add(new TMember(71, 15));
            TPoly tp2 = new TPoly(arr2);
            tp.sub(tp2);
            string str = tp.ToString();
            Assert.AreEqual("(1*x^8) + (-71*x^15)", str);
        }

        [TestMethod()]
        public void minusTest()
        {
            ArrayList arr = new ArrayList();
            arr.Add(new TMember(2, 5));
            arr.Add(new TMember(3, 2));
            arr.Add(new TMember(4, 8));
            TPoly tp = new TPoly(arr);
        }

        [TestMethod()]
        public void equalsTest()
        {
            ArrayList arr = new ArrayList();
            arr.Add(new TMember(2, 5));
            arr.Add(new TMember(3, 2));
            arr.Add(new TMember(4, 8));
            TPoly tp = new TPoly(arr);
```

```
            ArrayList arr2 = new ArrayList();
            arr2.Add(new TMember(3, 2));
            arr2.Add(new TMember(2, 5));
            arr2.Add(new TMember(4, 8));
            TPoly tp2 = new TPoly(arr2);
            Assert.IsTrue(tp.equals(tp2));
        }

        [TestMethod()]
        public void diffirentiateTest()
        {
            ArrayList arr = new ArrayList();
            arr.Add(new TMember(2, 5));//10x^4
            arr.Add(new TMember(3, 2));//6x^1
            arr.Add(new TMember(4, 8));//32x^7
            TPoly tp = new TPoly(arr);
            tp = tp.diffirentiate();
            Assert.AreEqual(tp.ToString(), "(10*x^4) + (6*x^1) +
(32*x^7)");
        }

        [TestMethod()]
        public void calculateTest()
        {
            ArrayList arr = new ArrayList();
            arr.Add(new TMember(2, 3));//16
            arr.Add(new TMember(3, 2));//12
            arr.Add(new TMember(4, 2));//16
            TPoly tp = new TPoly(arr);
            double res = tp.calculate(2);
            Assert.AreEqual(res, 44);
        }

        [TestMethod()]
        public void ToStringTest()
        {
            ArrayList arr = new ArrayList();
            arr.Add(new TMember(2, 3));
            arr.Add(new TMember(3, 2));
            TPoly tp = new TPoly(arr);
            Assert.AreEqual(tp.ToString(), "(2*x^3) + (3*x^2)");
        }
    }
}
```