Федеральное государственное бюджетное образовательное учреждение высшего образования

«Сибирский государственный университет телекоммуникаций и информатики»

(СибГУТИ)

Кафедра прикладной математики и кибернетики

Лабораторная работа № 6

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 2](#_Toc53830857)

[ТЕСТОВЫЕ НАБОРЫ ДАННЫХ 3](#_Toc53830858)

[ДЕМОНСТРАЦИЯ РАБОТЫ ПРОГРАММЫ 4](#_Toc53830859)

[ВЫВОД 4](#_Toc53830860)

[ПРИЛОЖЕНИЕ 5](#_Toc53830861)

[Листинг 1. TPNumber.cs 5](#_Toc53830862)

[Листинг 2. UnitTest1.cs 12](#_Toc53830863)

# ЗАДАНИЕ

1. Реализовать абстрактный тип данных «р-ичное число», используя класс, 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.

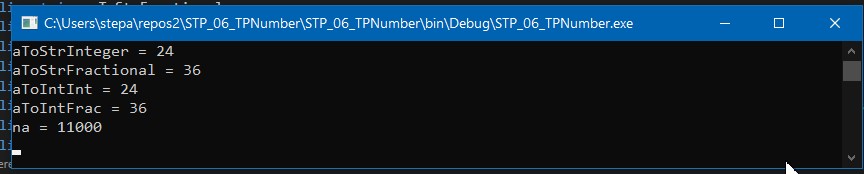
# ТЕСТОВЫЕ НАБОРЫ ДАННЫХ

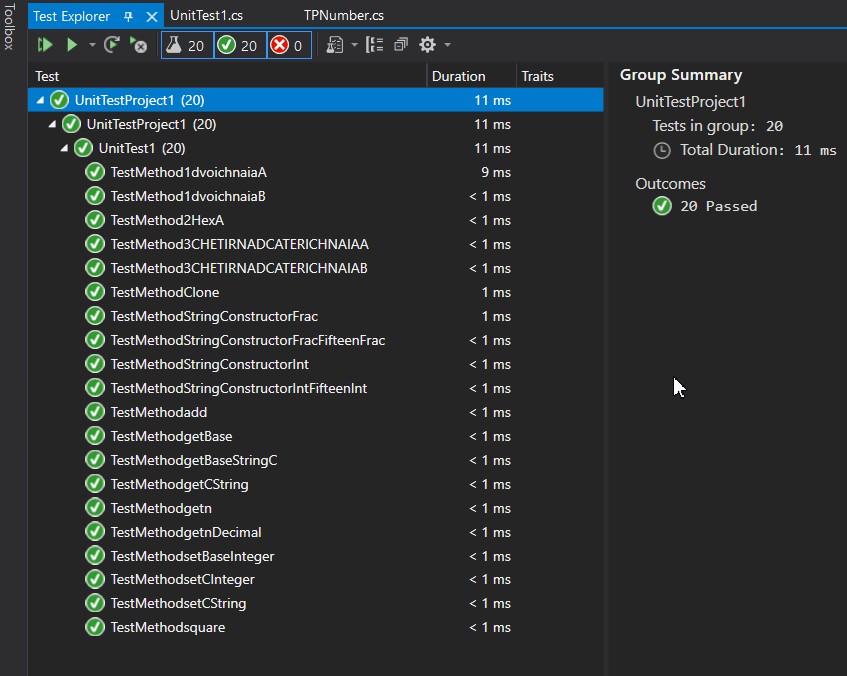
TPNumber tp = new TPNumber("212.22", 3, 7);//"212.22(троичная) = 23.888888888888(десятичная)

TPNumber tp = new TPNumber("AB4C.B9A", 15, 7);//AB4C.B9A(пятнадцатеричная) = 36297.776296296(десятичная)

TPNumber tp = new TPNumber("AB4C.B9A", 15, 7);//AB4C.B9A(пятнадцатеричная) = 36297.776296296(десятичная)

# ДЕМОНСТРАЦИЯ РАБОТЫ ПРОГРАММЫ





# ВЫВОД

Работа по переводу чисел из одной системы счисления в другую может быть реализована одним алгоритмом, это очень удобно и приятно. Обучился новым методам разбора строк и чисел на части.

# ПРИЛОЖЕНИЕ

## Листинг 1. TPNumber.cs

using System;

using System.Collections;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace STP\_06\_TPNumber

/\*Р-ичное число TPNumber - это действительное число (n) со знаком в системе

счисления с основанием (base) (в диапазоне 2..16), содержащее целую и дробную части.

Точность представления числа – (c >= 0). Р-ичные числа изменяемые.

\*/

{//реализую p-ичное число. base - основание(base) 2 <= base <= 16 ВОПРОС по КонструкторЧисло - каким образом передать в этот конструктор, например

// шестнадцатиричное число в виде вещественного? Понял. Передаётся вещественно число, например double, а потом переделывается в число с тем же

//значением, но с основанием b

/\*Например:

NCreate(s2,3,3) = число s2 в системе

счисления 3 с тремя разрядами после

троичной точки.

NCreate(s2,3,2) = число s2 в системе

счисления 3 с двумя разрядами после

троичной точки.\*/

public class TPNumber : ICloneable

{

public double nDecimal;

public string aToStrInteger;

public string aToStrFractional;

public int integerPartOfaInIntegerDecimal;

public int fractionalPartOfaInIntegerDecimal;

public int b;//base

public int c;//precision

public string na = "";//В случае числа в системе счисления больше 10 храню числа в виде строк. na - целая часть, nb - дробная, n - всё вместе

public string nb = "";

public string n = "";//число в его оригинальной системе счисления (b-ичное)

static void Main(string[] args)

{

TPNumber tp = new TPNumber(24.36, 2, 7);

TPNumber tp1 = new TPNumber("212.22", 3, 7);

TPNumber tp2 = new TPNumber("AC.B9A", 15, 7);//AC.B9A(пятнадцатеричная) = 162.776296296(десятичная)

Console.ReadLine();

}

public void setCString(string newc)

{

c = Int32.Parse(newc);

}

public void setCInteger(int newc)

{

c = newc;

}

public void setBaseString(string bs)

{

int bas = Int32.Parse(bs);

if (bas >= 2 && bas <= 16)

{

b = bas;

}

}

public void setBaseInteger(int newb)

{

if (newb >= 2 && newb <= 16)

{

b = newb;

}

}

public string getCString()

{

return c.ToString();

}

public int getC()

{

return c;

}

public string getBaseString()

{

return b.ToString();

}

public int getBase()

{

return b;

}

public string getn()

{

return n;

}

public double getnDecimal()

{

return nDecimal;

}

public TPNumber square()

{

return new TPNumber(nDecimal \* nDecimal, b, c);

}

public TPNumber add(TPNumber tpn)

{

if (b == tpn.b)

return new TPNumber(nDecimal + tpn.nDecimal, b, c);

else

{

Console.WriteLine("Bases don't match");

return null;

}

}

public object Clone()

{

return this.MemberwiseClone();

}

public TPNumber(double a, int b, int c)

{

if (b < 2 || b > 16 || c < 0)

{

throw new WrongInputInConstructor();

}

string aToStrInteger = a.ToString().Split(',')[0];

string aToStrFractional = a.ToString().Split(',')[1];

Console.WriteLine("aToStrInteger = " + aToStrInteger);

Console.WriteLine("aToStrFractional = " + aToStrFractional);

integerPartOfaInIntegerDecimal = Int32.Parse(aToStrInteger);

fractionalPartOfaInIntegerDecimal = Int32.Parse(aToStrFractional);

Console.WriteLine("aToIntInt = " + integerPartOfaInIntegerDecimal);

Console.WriteLine("aToIntFrac = " + fractionalPartOfaInIntegerDecimal);

// Console.ReadLine();

this.b = b;

this.c = c;

nDecimal = a;

translateFromDecimalAandB(integerPartOfaInIntegerDecimal, fractionalPartOfaInIntegerDecimal, b, c);

Console.WriteLine("na = " + na);

//Console.ReadLine();

}

//Вещественное число (s2). Система счисления(base), точность представления числа(c) – целые числа.

public void translateFromDecimalAandB(int a, int b, int bas, int c)

{

na = "";

nb = "";

n = "";

ArrayList ostatki = new ArrayList();

int chastnoe;

int ostatok;

do

{

chastnoe = a / bas;

ostatok = a % bas;

ostatki.Add(ostatok);

a = chastnoe;

} while (chastnoe > 0);

if (bas < 10)

{

for (int i = ostatki.Count - 1; i >= 0; i--)

{

na += ostatki[i];

}//это всё сработает для основания меньше 10. Если основание больше 10, надо будет числа большие 10 в ostatki превратить в буквы A, B, C и т.д.

}

else

{

for (int i = ostatki.Count - 1; i >= 0; i--)

{

if (ostatki[i].ToString() == "10")

{

na += "A";

}

else if (ostatki[i].ToString() == "11")

{

na += "B";

}

else if (ostatki[i].ToString() == "12")

{

na += "C";

}

else if (ostatki[i].ToString() == "13")

{

na += "D";

}

else if (ostatki[i].ToString() == "14")

{

na += "E";

}

else if (ostatki[i].ToString() == "15")

{

na += "F";

}

else

{

na += ostatki[i].ToString();

}

}

}//end of integer translation. Now Fractional:

int celoe;

double drobnoe;

string bstr = "0," + b.ToString();

ArrayList integerParts = new ArrayList();

//Console.WriteLine("bstr = " + bstr);

double bdouble = drobnoe = Double.Parse(bstr);

//Console.WriteLine("bdouble = " + bdouble);

//Console.ReadLine();

for (int i = 0; i < c; i++)

{

double multiplication = drobnoe \* (double)bas;

string strInt = multiplication.ToString().Split(',')[0];//0101110

celoe = Int32.Parse(strInt);

string strFrac = multiplication.ToString().Split(',')[1];

drobnoe = multiplication - (double)celoe;

integerParts.Add(celoe);

}

if (bas < 10)

{

for (int i = 0; i < integerParts.Count; i++)

{

nb += integerParts[i];

}//это всё сработает для основания меньше 10. Если основание больше 10, надо будет числа большие 10 в ostatki превратить в буквы A, B, C и т.д.

}

else

{

for (int i = 0; i < integerParts.Count; i++)

{

if (integerParts[i].ToString() == "10")

{

nb += "A";

}

else if (integerParts[i].ToString() == "11")

{

nb += "B";

}

else if (integerParts[i].ToString() == "12")

{

nb += "C";

}

else if (integerParts[i].ToString() == "13")

{

nb += "D";

}

else if (integerParts[i].ToString() == "14")

{

nb += "E";

}

else if (integerParts[i].ToString() == "15")

{

nb += "F";

}

else

{

nb += integerParts[i].ToString();

}

}

}

n += (na + "," + nb);

}

public TPNumber(string a, int b, int c)

{

string naa = a.Split('.')[0];

string nbb = a.Split('.')[1];

char[] sToCharArr = nbb.ToCharArray();//работаю с дробной частью

int numOfchars = sToCharArr.Length;

int[] sToCharArrToInt = new int[numOfchars];//decimal representations of chars of s

for (int i = 0; i < numOfchars; i++)

{

if (sToCharArr[i] == 'A')

sToCharArrToInt[i] = 10;

else if (sToCharArr[i] == 'B')

sToCharArrToInt[i] = 11;

else if (sToCharArr[i] == 'C')

sToCharArrToInt[i] = 12;

else if (sToCharArr[i] == 'D')

sToCharArrToInt[i] = 13;

else if (sToCharArr[i] == 'E')

sToCharArrToInt[i] = 14;

else if (sToCharArr[i] == 'F')

sToCharArrToInt[i] = 15;

else sToCharArrToInt[i] = sToCharArr[i] - '0';//it's kinda ugly way of converting char to int

}

double sum = 0;

for (int i = 1; i < numOfchars + 1; i++)

{

sum += Math.Pow(b, -i) \* sToCharArrToInt[i - 1];

}

fractionalPartOfaInIntegerDecimal = Int32.Parse(sum.ToString().Substring(2, c));

//---------------------------------

sToCharArr = naa.ToCharArray();//работаю с целой частью

numOfchars = sToCharArr.Length;

sToCharArrToInt = new int[numOfchars];//decimal representations of chars of s

for (int i = 0; i < numOfchars; i++)

{

if (sToCharArr[i] == 'A')

sToCharArrToInt[i] = 10;

else if (sToCharArr[i] == 'B')

sToCharArrToInt[i] = 11;

else if (sToCharArr[i] == 'C')

sToCharArrToInt[i] = 12;

else if (sToCharArr[i] == 'D')

sToCharArrToInt[i] = 13;

else if (sToCharArr[i] == 'E')

sToCharArrToInt[i] = 14;

else if (sToCharArr[i] == 'F')

sToCharArrToInt[i] = 15;

else sToCharArrToInt[i] = sToCharArr[i] - '0';//it's kinda ugly way of converting char to int

}

sum = 0;

for (int i = numOfchars - 1, j = 0; i >= 0; i--, j++)

{

sum += Math.Pow(b, i) \* sToCharArrToInt[j];

}

string temp = sum.ToString().Split('.')[0];

integerPartOfaInIntegerDecimal = Int32.Parse(temp);

}

}

public class WrongInputInConstructor : Exception

{

public WrongInputInConstructor()

{

Console.WriteLine("wrong input in constructor exception");

}

}

}

## Листинг 2. UnitTest1.cs

using System;

using Microsoft.VisualStudio.TestTools.UnitTesting;

using STP\_06\_TPNumber;

namespace UnitTestProject1

{

[TestClass]

public class UnitTest1

{

[TestMethod]

public void TestMethod1dvoichnaiaA()

{

TPNumber tp = new TPNumber(24.36, 2, 7);// 24.36=11000.0101110

tp.translateFromDecimalAandB(tp.integerPartOfaInIntegerDecimal, tp.fractionalPartOfaInIntegerDecimal, tp.b, tp.c);

Assert.AreEqual(tp.na, "11000");

}

[TestMethod]

public void TestMethod1dvoichnaiaB()

{

TPNumber tp = new TPNumber(24.36, 2, 7);// 24.36=11000.0101110

tp.translateFromDecimalAandB(tp.integerPartOfaInIntegerDecimal, tp.fractionalPartOfaInIntegerDecimal, tp.b, tp.c);

Assert.AreEqual(tp.nb, "0101110");

}

[TestMethod]

public void TestMethod2HexA()

{

TPNumber tp = new TPNumber(193.36, 16, 7);// 193.36=C1.5C28F5C28F6

tp.translateFromDecimalAandB(tp.integerPartOfaInIntegerDecimal, tp.fractionalPartOfaInIntegerDecimal, tp.b, tp.c);

Assert.AreEqual(tp.na, "C1");

}

[TestMethod]

public void TestMethod3CHETIRNADCATERICHNAIAA()

{

TPNumber tp = new TPNumber(193.36, 14, 7);// 193.36=DB.507BA8D

tp.translateFromDecimalAandB(tp.integerPartOfaInIntegerDecimal, tp.fractionalPartOfaInIntegerDecimal, tp.b, tp.c);

Assert.AreEqual(tp.na, "DB");

}

[TestMethod]

public void TestMethod3CHETIRNADCATERICHNAIAB()

{

TPNumber tp = new TPNumber(193.36, 14, 7);// 193.36=DB.507BA8D

tp.translateFromDecimalAandB(tp.integerPartOfaInIntegerDecimal, tp.fractionalPartOfaInIntegerDecimal, tp.b, tp.c);

Assert.AreEqual(tp.nb, "507BA8D");

}

[TestMethod]

public void TestMethodsetCString()

{

TPNumber tp = new TPNumber(24.36, 2, 7);

tp.setCString("20");

Assert.AreEqual(tp.c, 20);

}

[TestMethod]

public void TestMethodsetCInteger()

{

TPNumber tp = new TPNumber(24.36, 2, 7);

tp.setCInteger(20);

Assert.AreEqual(tp.c, 20);

}

[TestMethod]

public void TestMethodsetBaseInteger()

{

TPNumber tp = new TPNumber(24.36, 2, 7);

tp.setBaseInteger(12);

Assert.AreEqual(tp.b, 12);

}

[TestMethod]

public void TestMethodgetCString()

{

TPNumber tp = new TPNumber(24.36, 2, 7);

string str = tp.getCString();

Assert.AreEqual(str, "7");

}

[TestMethod]

public void TestMethodgetBaseStringC()

{

TPNumber tp = new TPNumber(24.36, 2, 7);

string str = tp.getBaseString();

Assert.AreEqual(str, "2");

}

[TestMethod]

public void TestMethodgetBase()

{

TPNumber tp = new TPNumber(24.36, 2, 7);

int str = tp.getBase();

Assert.AreEqual(str, 2);

}

[TestMethod]

public void TestMethodgetn()

{

TPNumber tp = new TPNumber(24.36, 2, 7);

string str = tp.getn();

Assert.AreEqual(str, "11000,0101110");

}

[TestMethod]

public void TestMethodgetnDecimal()

{

TPNumber tp = new TPNumber(24.36, 2, 7);

double str = tp.getnDecimal();

Assert.AreEqual(str, 24.36);

}

[TestMethod]

public void TestMethodsquare()

{

TPNumber tp = new TPNumber(24.36, 2, 7);

TPNumber str = tp.square();

Assert.AreEqual(str.nDecimal, 593.4096);

}

[TestMethod]

public void TestMethodadd()

{

TPNumber tp = new TPNumber(24.36, 2, 7);

TPNumber tp2 = new TPNumber(20.36, 2, 7);

TPNumber tp3 = tp2.add(tp);

Assert.AreEqual(tp3.nDecimal, 44.72);

}

[TestMethod]

public void TestMethodClone()

{

TPNumber tp = new TPNumber(24.36, 2, 7);

TPNumber tp2 = (TPNumber) tp.Clone();

Assert.AreEqual(tp2.nDecimal, 24.36);

}

[TestMethod]

public void TestMethodStringConstructorFrac()

{

TPNumber tp = new TPNumber("212.22", 3, 7);//"212.22(троичная) = 23.888888888888(десятичная)

Assert.AreEqual(tp.fractionalPartOfaInIntegerDecimal, 8888888);

}

[TestMethod]

public void TestMethodStringConstructorInt()

{

TPNumber tp = new TPNumber("212.22", 3, 7);//"212.22(троичная) = 23.888888888888(десятичная)

Assert.AreEqual(tp.integerPartOfaInIntegerDecimal, 23);

}

[TestMethod]

public void TestMethodStringConstructorFracFifteenFrac()

{

TPNumber tp = new TPNumber("AB4C.B9A", 15, 7);//AB4C.B9A(пятнадцатеричная) = 36297.776296296(десятичная)

Assert.AreEqual(tp.fractionalPartOfaInIntegerDecimal, 7762962);

}

[TestMethod]

public void TestMethodStringConstructorIntFifteenInt()

{

TPNumber tp = new TPNumber("AB4C.B9A", 15, 7);//AB4C.B9A(пятнадцатеричная) = 36297.776296296(десятичная)

Assert.AreEqual(tp.integerPartOfaInIntegerDecimal, 36297);

}

}

}