Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное

учреждение высшего образования

ИРКУТСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ

ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Институт/Факультет – Информационные системы и технологии   
 наименование

**Название работы** – Свой Тип

Отчет по лабораторной работе № 3

по дисциплине Технология программирования   
 наименование учебной дисциплины

Выполнилcтудент ИСТб-20-3 Д.А. Пинегин

номер группы подпись И. О. Фамилия

дата

Принял Доцент З.А. Бахвалова

должность подпись И.О. Фамилия

дата

Иркутск – 2021

# Лабораторная работа №3

### Постановка задачи

**Задача №2.** Мера объема, задаваемая в виде пары (значение, тип), допустимые типы: м3, миллилитры, баррель.

* сложение
* вычитание
* умножение на число
* сравнение двух объемов
* вывод значения в любом типе

**Словесное описание алгоритма**

1. Создаем программу GUI
2. Вторым шагом создадим новый класс Length.cs
3. В нем пропишем условия выполненной задачи
4. Далее проведем тест кода
5. После у нас откроется новый класс Length.Tests
6. После успешного выполнения всех Автотестов приступаем к графическому созданию программы
7. В итоге присваиваем textBox значение textChanchged для автоматического вычисления без добавления кнопок
8. В строковой массив заносим типы мер величин
9. Прописываем код, который имеет ссылки на класс Length.cs

### Таблица спецификаций

|  |  |  |  |
| --- | --- | --- | --- |
| Имя | Назначение | Тип | Модификтор доступа |
| **Класс Lenght** | | | |
| m | Литры | double | public |
| km | Баррели | double | public |
| au | М3 | double | public |
| ps | Миллилитры | double | public |
| numbers | Числа | double | public |
| Lenght | Тип | float | public |
| value | Для проведения операций | double | public |
| **Класс Form1– Поля** | | | |
| firstValue | Хранит число из первой строчки | Double | public |
| secondValue | Хранит число из второй строчки | Double | public |
| l | Для сравнения | Double | public |
| k | Для сравнения | Double | public |
| sumLength | Для выдачи ответа | Lenght | public |
| firstLength | Для преобразования типа | Lenght | public |
| secondLength | Для преобразования типа | Lenght | public |
| m | Литры | double | public |
| km | Баррели | double | public |
| au | М3 | double | public |
| ps | Миллилитры | double | public |

### Таблица тестов

| Тест | Что проверяем | Входные данные | Результат теста |
| --- | --- | --- | --- |
| 1 | Проверка сложение 25 баррелей с 124 литрами | 4,074 м3 | Успешно |
| 2 | Проверка умножение 1291 миллилитров и 3 баррелей | 475,291 л. | Успешно |
| 3 | Проверка вычитания 500 баррелей  и набора букв asas | Ничего не произошло | Успешно |

Рисунок 1 – Результат теста 1.

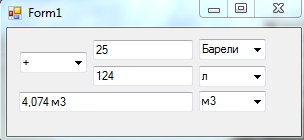


Рисунок 2 – Результат теста 2.

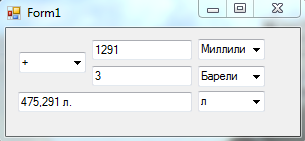
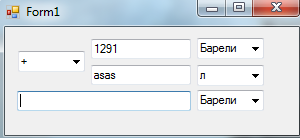


Рисунок 3 – Результат теста 3.



### Код программы

**Файл Lenght.cs**

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace WindowsFormsApp8  {  public enum MeasureType { m, km, au, ps };  public class Length  {  private double value;  private MeasureType type;  public Length(double value, MeasureType type)  {  this.value = value;  this.type = type;  }  public string Verbose()  {  string typeVerbose = "";  switch (this.type)  {  case MeasureType.m:  typeVerbose = "л.";  break;  case MeasureType.km:  typeVerbose = "барлей.";  break;  case MeasureType.au:  typeVerbose = "м3";  break;  case MeasureType.ps:  typeVerbose = "миллилитры";  break;  }  return String.Format("{0} {1}", this.value, typeVerbose);  }  public static Length operator +(Length instance, double number)  {  var newValue = instance.value + number;  var length = new Length(newValue, instance.type);  return length;  }  public static Length operator +(double number, Length instance)  {  return instance + number;  }  public static Length operator \*(Length instance, double number)  {  return new Length(instance.value \* number, instance.type); ;  }  public static Length operator \*(double number, Length instance)  {  return instance \* number;  }  public static Length operator -(Length instance, double number)  {  return new Length(instance.value - number, instance.type); ;  }  public static Length operator -(double number, Length instance)  {  return instance - number;  }  public static Length operator /(Length instance, double number)  {  return new Length(instance.value / number, instance.type); ;  }  public static Length operator /(double number, Length instance)  {  return instance / number;  }  public Length To(MeasureType newType)  {  var newValue = this.value;  if (this.type == MeasureType.m)  {  switch (newType)  {  case MeasureType.m:  newValue = this.value;  break;  case MeasureType.km:  newValue = this.value / 158;  break;  case MeasureType.au:  newValue = this.value / 1000;  break;  case MeasureType.ps:  newValue = this.value \* 1000;  break;  }  }  else if (newType == MeasureType.m)  {  switch (this.type)  {  case MeasureType.m:  newValue = this.value;  break;  case MeasureType.km:  newValue = this.value \* 158;  break;  case MeasureType.au:  newValue = this.value \* 1000;  break;  case MeasureType.ps:  newValue = this.value / 1000;  break;  }  }  else  {  newValue = this.To(MeasureType.m).To(newType).value;  }  return new Length(newValue, newType);  }  public static Length operator +(Length instance1, Length instance2)  {  return instance1 + instance2.To(instance1.type).value;  }  public static Length operator -(Length instance1, Length instance2)  {  return instance1 - instance2.To(instance1.type).value;  }  }  } |

**Файл Form1.cs**

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.ComponentModel;  using System.Data;  using System.Drawing;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using System.Windows.Forms;  namespace WindowsFormsApp8  {  public partial class Form1 : Form  {  public Form1()  {  InitializeComponent();  var measureItems = new string[]  {  "л",  "Барели",  "м3",  "Миллилитры",  };  cmbFirstType.DataSource = new List<string>(measureItems);  cmbSecondType.DataSource = new List<string>(measureItems);  cmbResultType.DataSource = new List<string>(measureItems);  }  private void txtFirst\_TextChanged(object sender, EventArgs e)  {  Calculate();  }  private void txtSecond\_TextChanged(object sender, EventArgs e)  {  Calculate();  }  private void cmbOperation\_SelectedIndexChanged(object sender, EventArgs e)  {  Calculate();  }  private MeasureType GetMeasureType(ComboBox comboBox)  {  MeasureType measureType;  switch (comboBox.Text)  {  case "л":  measureType = MeasureType.m;  break;  case "Барели":  measureType = MeasureType.km;  break;  case "м3":  measureType = MeasureType.au;  break;  case "Миллилитры":  measureType = MeasureType.ps;  break;  default:  measureType = MeasureType.m;  break;  }  return measureType;  }  private void Calculate()  {  try  {  var firstValue = double.Parse(txtFirst.Text);  var secondValue = double.Parse(txtSecond.Text);  double l = double.Parse(txtFirst.Text);  double k = double.Parse(txtSecond.Text);  // вместо трех страшных свитчей, три вызова нашей новой функции  MeasureType firstType = GetMeasureType(cmbFirstType);  MeasureType secondType = GetMeasureType(cmbSecondType);  MeasureType resultType = GetMeasureType(cmbResultType);  // тут сразу тип полученный передаем в момент создания экземпляра класса  var firstLength = new Length(firstValue, firstType);  var secondLength = new Length(secondValue, secondType);  Length sumLength;  switch (cmbOperation.Text)  {  case "+":  sumLength = firstLength + secondLength;  break;  case "-":  sumLength = firstLength - secondLength;  break;  case "\*":  sumLength = firstValue \* secondLength;  break;  case "==":  if (l - k < 0)  {  sumLength = secondLength;  }  else  {  sumLength = firstLength;  }  break;  default:  sumLength = new Length(0, MeasureType.m);  break;  }    txtResult.Text = sumLength.To(resultType).Verbose();  }  catch (FormatException)  {  // если тип преобразовать не смогли  }  }  private void cmbFirstType\_SelectedIndexChanged(object sender, EventArgs e)  {  Calculate();  }  private void cmbSecondType\_SelectedIndexChanged(object sender, EventArgs e)  {  Calculate();  }  private void cmbResultType\_SelectedIndexChanged(object sender, EventArgs e)  {  Calculate();  }  }  } |

### Код Автотестов

|  |
| --- |
| using Microsoft.VisualStudio.TestTools.UnitTesting;  using WindowsFormsApp8;  using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace WindowsFormsApp8.Tests  {  [TestClass()]  public class LengthTests  {  [TestMethod()]  public void VerboseTest()  {  var length = new Length(38, MeasureType.m);  Assert.AreEqual("38 л.", length.Verbose());  length = new Length(38, MeasureType.km);  Assert.AreEqual("38 барлей.", length.Verbose());  length = new Length(38, MeasureType.au);  Assert.AreEqual("38 м3", length.Verbose());  length = new Length(38, MeasureType.ps);  Assert.AreEqual("38 миллилитры", length.Verbose());  }  [TestMethod()]  public void AddNumberTest()  {  var length = new Length(0, MeasureType.m);  length = length + 4.25;  Assert.AreEqual("4,25 л.", length.Verbose());  }  [TestMethod()]  public void SubNumberTest()  {  var length = new Length(3, MeasureType.m);  length = length - 1.75;  Assert.AreEqual("1,25 л.", length.Verbose());  }  [TestMethod()]  public void MulByNumberTest()  {  var length = new Length(3, MeasureType.m);  length = length \* 3;  Assert.AreEqual("9 л.", length.Verbose());  }  [TestMethod()]  public void DivByNumberTest()  {  var length = new Length(3, MeasureType.m);  length = length / 3;  Assert.AreEqual("1 л.", length.Verbose());  }  [TestMethod()]  public void MeterToAnyTest()  {  Length length;  length = new Length(1000, MeasureType.m);  Assert.AreEqual("6,32911392405063 барлей.", length.To(MeasureType.km).Verbose());  length = new Length(149597870700 \* 2, MeasureType.m);  Assert.AreEqual("299195741,4 м3", length.To(MeasureType.au).Verbose());  length = new Length(3 \* 3.0856776 \* Math.Pow(10, 16), MeasureType.m);  Assert.AreEqual("9,2570328E+19 миллилитры", length.To(MeasureType.ps).Verbose());  }  [TestMethod()]  public void AnyToMeterTest()  {  Length length;  length = new Length(1, MeasureType.km);  Assert.AreEqual("158 л.", length.To(MeasureType.m).Verbose());  length = new Length(1, MeasureType.au);  Assert.AreEqual("1000 л.", length.To(MeasureType.m).Verbose());  length = new Length(1, MeasureType.ps);  Assert.AreEqual("0,001 л.", length.To(MeasureType.m).Verbose());  }  [TestMethod()]  public void AddSubKmMetersTest()  {  var m = new Length(100, MeasureType.m);  var km = new Length(1, MeasureType.km);  Assert.AreEqual("258 л.", (m + km).Verbose());  Assert.AreEqual("1,63291139240506 барлей.", (km + m).Verbose());  Assert.AreEqual("0,367088607594937 барлей.", (km - m).Verbose());  Assert.AreEqual("-58 л.", (m - km).Verbose());  }  }  } |

### Результат работы Автотестов

