**Выполнил:** Штырев Глеб Сергеевич, группа ИСТб-23-2

# Лабораторное задание №1

## Задача на If

### Задание

### **Задача №12.** Дано натуральное число 1≤n≤9999, определяющее стоимость товара в копейках. Выразить стоимость в рублях и копейках, например, 3 рубля 21 копейка, 15 рублей 5 копеек, 1 рубль ровно и т. п.

### Постановка задачи

Необходимо полученное на вход число копеек преобразовать в количество рублей путем деления числа на 100 (целочисленного деления и остатка от деления). Для количества рублей и копеек необходимо прописать все случаи склонения слов.

### Таблица спецификаций

**Функция Main**

|  |  |  |
| --- | --- | --- |
| Имя | Назначение | Тип/Диапазон |
| n | Стоимость товара в копейках | Текст |

**Logic.Convert(string n) –** Функция получает на вход число и выдает ошибку или количество рублей и копеек, учитывая склонение существительных.

|  |  |  |
| --- | --- | --- |
| Имя | Назначение | Тип/Диапазон |
| outMessage | Выводящееся сообщение с ответом или ошибкой | Текст |
| x | Стоимость товара в копейках | Целое положительное число |

### Таблица тестов

| Номер теста | Что проверяем | Входные данные | Выходные данные |
| --- | --- | --- | --- |
| 1 | Вводим текст | n = "AAAAAAAAA" | outMessage = "некорректный ввод" |
| 2 | Вводим число за пределами ОДЗ | n = "0" | outMessage = "число за пределом ОДЗ" |
| 3 | Вводим количество копеек, делящееся на 100 | n = "5500" | outMessage = "55 рублей ровно" |
| 4 | Вводим количество копеек, не делящееся на 100 | n = "5545" | outMessage = "55 рублей 45 копеек" |

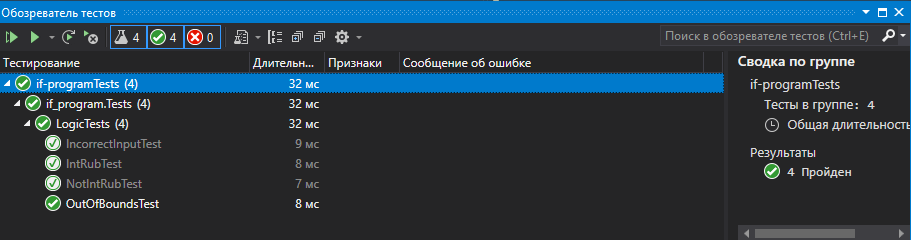
### Код программы

|  |
| --- |
| using System;  namespace if\_program  {  public class Logic  {  public static string Convert(string n)  {  string outMessage;  if (!int.TryParse(n, out int x))  {  outMessage = "некорректный ввод";  }  else  {  if (x < 1 || x > 9999)  {  outMessage = "число за пределом ОДЗ";  }  else  {  if (x % 100 == 0)  {  if ((x / 100)%10 == 1)  {  if (x / 1000 == 1)  {  outMessage = (x / 100) + " рублей ровно";  }  else  {  outMessage = (x / 100) + " рубль ровно";  }  }  else  {  if ((x / 100) % 10 > 4)  {  outMessage = (x / 100) + " рублей ровно";  }  else  {  if (x / 1000 == 1)  {  outMessage = (x / 100) + " рублей ровно";  }  else  {  outMessage = (x / 100) + " рубля ровно";  }  }  }  }  else  {  if ((x / 100) % 10 == 1)  {  if (x / 1000 == 1)  {  outMessage = (x / 100) + " рублей ";  }  else  {  outMessage = (x / 100) + " рубль ";  }  }  else  {  if ((x / 100) % 10 > 4)  {  outMessage = (x / 100) + " рублей ";  }  else  {  if (x / 1000 == 1)  {  outMessage = (x / 100) + " рублей ";  }  else  {  outMessage = (x / 100) + " рубля ";  }  }  }  if ((x % 100) % 10 == 1)  {  if ((x % 100)/10 == 1)  {  outMessage += (x % 100) + " копеек";  }  else  {  outMessage += (x % 100) + " копейка";  }  }  else  {  if ((x % 100) % 10 > 4)  {  outMessage += (x % 100) + " копеек";  }  else  {  if ((x % 100)/10 == 1)  {  outMessage += (x % 100) + " копеек";  }  else  {  outMessage += (x % 100) + " копейки";  }  }  }  }  }  }  return outMessage;  }  }  class Program  {  static void Main(string[] args)  {  Console.WriteLine("Введите натуральное число 1<=n<=9999");  string n = Console.ReadLine();  while (Logic.Convert(n) == "некорректный ввод" || Logic.Convert(n) == "число за пределом ОДЗ")  {  Console.WriteLine(Logic.Convert(n));  Console.WriteLine("Введите натуральное число 1<=n<=9999");  n = Console.ReadLine();  }  Console.WriteLine(Logic.Convert(n));  Console.ReadKey();  }  }  } |

### Код тестов

|  |
| --- |
| using Microsoft.VisualStudio.TestTools.UnitTesting;  namespace if\_program.Tests  {  [TestClass()]  public class LogicTests  {  [TestMethod()]  public void IncorrectInputTest()  {  string n = "AAAAAAAAA";  Assert.AreEqual(Logic.Convert(n), "некорректный ввод");  }  [TestMethod()]  public void OutOfBoundsTest()  {  string n = "0";  Assert.AreEqual(Logic.Convert(n), "число за пределом ОДЗ");  }  [TestMethod()]  public void IntRubTest()  {  string n = "5500";  Assert.AreEqual(Logic.Convert(n), "55 рублей ровно");  }  [TestMethod()]  public void NotIntRubTest()  {  string n = "5545";  Assert.AreEqual(Logic.Convert(n), "55 рублей 45 копеек");  }  }  } |

### Результаты тестирования



## Задача на For

### Задача

**Задача №15.** Гражданин 1 марта открыл счет в банке, вложив A руб. Через каждый месяц размер вклада увеличивается на 2% от имеющейся суммы. Определить: а) за какой месяц величина ежемесячного увеличения вклада превысит B руб.; б) через сколько месяцев размер вклада превысит C руб.

### Постановка задачи

Программа получает на вход 3 числа (количество денег на счету при открытии счета, ожидаемый размер увеличения вклада, Ожидаемое количество денег на счету). Путем прибавления 2-х процентов к первому числу, программа вычисляет и выдает срок в месяцах, после которого увеличения вклада и размер вклада превысят ожидаемую сумму)

### Таблица спецификаций

**Main**

|  |  |  |
| --- | --- | --- |
| Имя | Назначение | Тип/Диапазон |
| a | Сумма, вложенная при открытии вклада | Текст |
| b | Интересующий размер увеличения вклада | Текст |
| c | Ожидаемая сумма на счету | Текст |

**Logic.IncreaseMonthCount(string a, string b, string c) –** Функция получает на вход три числа и выдает ошибку либо срок через который размер увеличения вклада увеличит второе введенное число

|  |  |  |
| --- | --- | --- |
| Имя | Назначение | Тип/Диапазон |
| aAnswer | Выводящееся сообщение с ответом или ошибкой | Текст |
| floatA | Сумма, вложенная при открытии вклада | Вещественное положительное число |
| floatB | Интересующий размер увеличения вклада | Вещественное положительное число |
| floatC | Ожидаемая сумма на счету | Вещественное положительное число |
| incr | Размер увеличение вклада | Вещественное положительное число |
| aMonth | Месяц, когда размер увеличения вклада, больше введенного значение | Целое число |
| i | Параметр цикла | Целое число |

**Logic.DepositMonthCount(string a, string b, string c) -** Функция получает на вход три числа и выдает ошибку либо срок через который размер вклада увеличит третье введенное число

|  |  |  |
| --- | --- | --- |
| Имя | Назначение | Тип/Диапазон |
| bAnswer | Выводящееся сообщение с ответом или ошибкой | Текст |
| floatA | Сумма, вложенная при открытии вклада | Вещественное положительное число |
| floatB | Интересующий размер увеличения вклада | Вещественное положительное число |
| floatC | Ожидаемая сумма на счету | Вещественное положительное число |
| incr | Размер увеличение вклада | Вещественное положительное число |
| bMonth | Месяц, когда размер вклада, больше введенного значение | Целое число |
| i | Параметр цикла | Целое число |

### Таблица тестов

| Номер теста | Что проверяем | Входные данные | Выходные данные |
| --- | --- | --- | --- |
| 1 | Вводим текст | a = "AAAAA"  b = "BBBBB"  c = "CCCCC" | aAnswer = "некорректный ввод"  bAnswer = "некорректный ввод" |
| 2 | Вводим числа за пределами ОДЗ | a = "-100"  b = "5"  c = "200" | aAnswer = "число за пределом ОДЗ"  bAnswer = "число за пределом ОДЗ" |
| 3 | Вводим корректный формат чисел | a = "100"  b = "5"  c = "200" | aAnswer = 48  bAnswer = 36 |

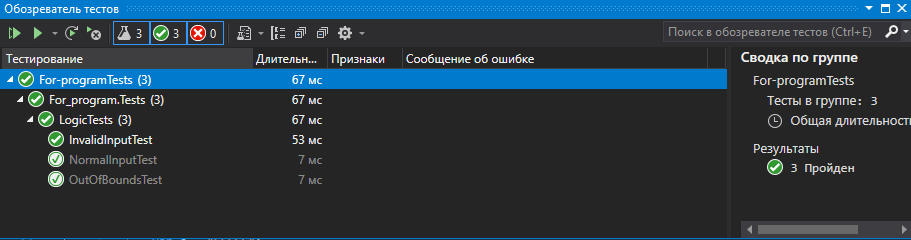
### Код программы

|  |
| --- |
| using System;  namespace For\_program  {  public class Logic  {  public static string IncreaseMonthCount(string a, string b, string c)  {  string aAnswer;  if (!float.TryParse(a, out float floatA) || !float.TryParse(b, out float floatB) || !float.TryParse(c, out float floatC))  {  aAnswer = "некорректный ввод";  }  else  {  floatA = float.Parse(a);  floatB = float.Parse(b);  floatC = float.Parse(c);  if (!(floatA >= 0 && floatB >= 0 && floatC >= 0))  {  aAnswer = "числа за пределами ОДЗ";  }  else  {  float incr = 0;  int aMonth = -1;  for (int i = 1; incr <= floatB; i++)  {  incr = floatA / 50;  floatA += incr;  if (incr > floatB && aMonth == -1)  {  aMonth = i;  }  }  aAnswer = aMonth.ToString();  }  }  return aAnswer;  }  public static string DepositMonthCount(string a, string b, string c)  {  string bAnswer;  if (!float.TryParse(a, out float floatA) || !float.TryParse(b, out float floatB) || !float.TryParse(c, out float floatC))  {  bAnswer = "некорректный ввод";  }  else  {  floatA = float.Parse(a);  floatB = float.Parse(b);  floatC = float.Parse(c);  if (!(floatA >= 0 && floatB >= 0 && floatC >= 0))  {  bAnswer = "числа за пределами ОДЗ";  }  else  {  float incr;  int bMonth = -1;  for (int i = 1; floatA <= floatC; i++)  {  incr = floatA / 50;  floatA += incr;  if (floatA > floatC && bMonth == -1)  {  bMonth = i;  }  }  bAnswer = bMonth.ToString();  }  }  return bAnswer;  }  }  class Program  {  static void Main(string[] args)  {  Console.WriteLine("Введите сумму, вложенную при открытии вклада");  var a = Console.ReadLine();  Console.WriteLine("Введите интересующий размер увеличения вклада");  var b = Console.ReadLine();  Console.WriteLine("Введите ожидаемую сумму на счету");  var c = Console.ReadLine();  while (Logic.IncreaseMonthCount(a, b, c) == "некорректный ввод" || Logic.IncreaseMonthCount(a, b, c) == "числа за пределами ОДЗ")  {  Console.WriteLine(Logic.IncreaseMonthCount(a, b, c));  Console.WriteLine("Введите сумму, вложенную при открытии вклада");  a = Console.ReadLine();  Console.WriteLine("Введите интересующий размер увеличения вклада");  b = Console.ReadLine();  Console.WriteLine("Введите ожидаемую сумму на счету");  c = Console.ReadLine();  }  Console.WriteLine("a) Небходимое число месяцев: " + Logic.IncreaseMonthCount(a,b,c));  Console.WriteLine("b) Небходимое число месяцев: " + Logic.DepositMonthCount(a,b,c));  Console.ReadKey();  }  }  } |

### Код тестов

|  |
| --- |
| using Microsoft.VisualStudio.TestTools.UnitTesting;  namespace For\_program.Tests  {  [TestClass()]  public class LogicTests  {  [TestMethod()]  public void InvalidInputTest()  {  var a = "AAAAA";  var b = "BBBBB";  var c = "CCCCC";  Assert.AreEqual(Logic.IncreaseMonthCount(a, b, c), "некорректный ввод");  Assert.AreEqual(Logic.DepositMonthCount(a, b, c), "некорректный ввод");  }  [TestMethod()]  public void OutOfBoundsTest()  {  var a = "-100";  var b = "5";  var c = "200";  Assert.AreEqual(Logic.IncreaseMonthCount(a, b, c), "числа за пределами ОДЗ");  Assert.AreEqual(Logic.DepositMonthCount(a, b, c), "числа за пределами ОДЗ");  }  [TestMethod()]  public void NormalInputTest()  {  var a = "100";  var b = "5";  var c = "200";  Assert.AreEqual(Logic.IncreaseMonthCount(a, b, c), "48");  Assert.AreEqual(Logic.DepositMonthCount(a, b, c), "36");  }  }  } |

### Результаты тестирования



## Задача на String

### Задание

**Задача №13.** Дан текст, имеющий вид: d1±d2±...±dnd1±d2±...±dn, где didi — цифры (n > 1). Вычислить записанную в тексте сумму

### Постановка Задачи

Программа получает на вход строку цифр, разделенную знаками + или -. Путем сложения и вычитания программа должна выдать сумму, записанную в тексте.

### Таблица спецификаций

**Main**

|  |  |  |
| --- | --- | --- |
| Имя | Назначение | Тип/Диапазон |
| line | Строку цифр, разделенных зхнаками + или - | Текст |

**Logic.CheckLine(string line) –** Функция получает на вход строку и на выходе выдает true если строка совпадает шаблону и false в противном случае.

|  |  |  |
| --- | --- | --- |
| Имя | Назначение | Тип/Диапазон |
| i | Параметр цикла | Целое число |

**Logic.CountLine(string line) -** Функция получает на вход строку и на выходе выдает ошибку или сумму, записанную в тексте.

|  |  |  |
| --- | --- | --- |
| Имя | Назначение | Тип/Диапазон |
| outMessage | Выводящееся сообщение с ответом или ошибкой | Текст |
| result | Сумма, введенной | Целое число |
| i | Параметр цикла | Целое число |

### Таблица тестов

| Номер теста | Что проверяем | Входные данные | Выходные данные |
| --- | --- | --- | --- |
| 1 | Вводим текст | line = "incorrect" | outMessage = "некорректный ввод" |
| 2 | Вводим почти корректную строку | line = "-5+5-5" | outMessage = "некорректный ввод" |
| 3 | Вводим корректную строку | line = "5+5-5+6" | outMessage = 11 |

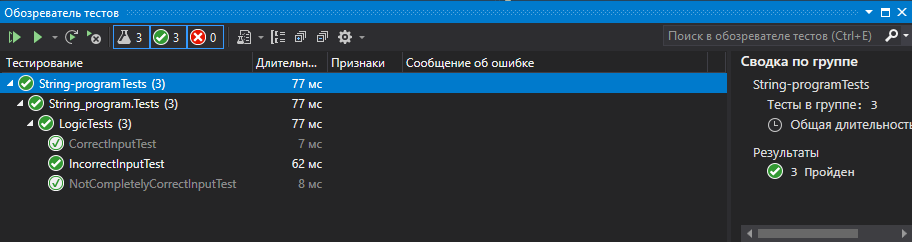
### Код программы

|  |
| --- |
| using System;  namespace String\_program  {  public class Logic  {  public static bool CheckLine(string line)  {  for (int i=0; i<line.Length; i++)  {  if (i%2==0)  {  if (!int.TryParse(line[i].ToString(), out int res))  {  return false;  }  }  else  {  if(line[i]!='+' && line[i]!='-')  {  return false;  }  }  }  return true;  }  public static string CountLine(string line)  {  string outMessage = "";  if (CheckLine(line))  {  int result = int.Parse(line[0].ToString());  for(int i = 1; i < line.Length; i++)  {  if (i % 2 == 0)  {  if (line[i-1] == '+')  {  result += int.Parse(line[i].ToString());  }  if(line[i - 1] == '-')  {  result -= int.Parse(line[i].ToString());  }  }  }  outMessage = result.ToString();  }  else  {  outMessage = "некорректный ввод";  }  return outMessage;  }  }  class Program  {  static void Main(string[] args)  {  Console.WriteLine("введите строку цифр, разделенных зхнаками + или -");  string line = Console.ReadLine();  while (Logic.CountLine(line) == "некорректный ввод")  {  Console.WriteLine(Logic.CountLine(line));  Console.WriteLine("введите строку цифр, разделенных зхнаками + или -");  line = Console.ReadLine();  }  Console.WriteLine(Logic.CountLine(line));  Console.ReadKey();  }  }  } |

### Код тестов

|  |
| --- |
| using Microsoft.VisualStudio.TestTools.UnitTesting;  namespace String\_program.Tests  {  [TestClass()]  public class LogicTests  {  [TestMethod()]  public void IncorrectInputTest()  {  string line = "incorrect";  Assert.AreEqual(Logic.CountLine(line), "некорректный ввод");  }  [TestMethod()]  public void NotCompletelyCorrectInputTest()  {  string line = "-5+5-5";  Assert.AreEqual(Logic.CountLine(line), "некорректный ввод");  }  [TestMethod()]  public void CorrectInputTest()  {  string line = "5+5-5+6";  Assert.AreEqual(Logic.CountLine(line), "11");  }  }  } |

### Результаты тестирования



### Ссылка на ГИТ

<https://github.com/Glebwen/TP-Lab1>