Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное

учреждение высшего образования

ИРКУТСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ

ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Институт информационных технологий и анализа данных

Факультет Информационные системы и технологии

Отчёт по лабораторной работе № 2

по дисциплине Технология программирования

Выполнил студент ИСТб-20-3 М. Д. Ярков 29.09.2021

номер группы подпись И. О. Фамилия Дата

Принял З. А. Бахвалова . .2021

должность подпись И. О. Фамилия Дата

18 баллов

Иркутск 2021

### Постановка задачи

**Задача на If** – вариант 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.

Программа должна иметь кнопку «Прочитать задание» и уметь сохранять непустые данные. Данные должны проверяться автоматически. Автотесты должны присутствовать.

### Словесное описание алгоритма

1. При запуске получаем сохранённое с предыдущей сессии значение savedNumData и добавляем в поля numData. Вызываем метод ChangeText()
2. При вводе данных в поля (нажатие клавиши или изменение поля) также вызываем метод ChangeText()
3. С помощью метода ChangeText() получаем необходимую информацию с помощью метода GetMoney() и задаём полученное значение полю outLabel, а также локально сохраняем новые данные
4. С помощью метода GetMoney(n) получаем строку с информацией:
   1. Инициализируем строковую переменную output со значением "", числовые переменные rubs со значением (n / 100) и kops со значением (n % 100)
   2. В случае наличия в переменной rubs числа больше нуля, выводим количество рублей:
      1. Добавляем к output значение rubs
      2. Если rubs в промежутке от 5 до 20 добавляем к output " рублей "
      3. Если rubs имеет на конце 1 добавляем к output " рубль "
      4. Если rubs имеет на конце 2, 3 или 4 добавляем к output " рубля "
      5. В остальных случаях добавляем к output " рублей "
   3. В случае наличия в переменной kops нуля добавляем к output "ровно", иначе:
      1. Добавляем к output значение kops
      2. Если kops в промежутке от 5 до 20 добавляем к output " копеек"
      3. Если kops имеет на конце 1 добавляем к output " копейка"
      4. Если kops имеет на конце 2, 3 или 4 добавляем к output " копейки"
      5. В остальных случаях добавляем к output " копеек"
   4. Выводим output
5. Внутри интерфейса прописаны минимальные и максимальные значения для поля ввода числа, поэтому дополнительная проверка не требуется
6. Кнопка «Прочитать задание» показывает задание, по которому создавалась программа.

### Таблица спецификаций

|  |  |  |
| --- | --- | --- |
| Имя | Назначение | Тип/Диапазон |
| Входные величины | | |
| numData | Стоимость товара в копейках | Числовое поле ввода, целое число от 1 до 9999 |

### Таблица тестов

| Номер теста | Что проверяем | Входные данные | Выходные данные |
| --- | --- | --- | --- |
| 1 | Расчёт 1500 копеек | 1500 | 15 рублей ровно |
| 2 | Расчёт 1 копейки | 1 | 1 копейка |
| 3 | Расчёт 9999 копеек | 9999 | 99 рублей 99 копеек |
| 4 | Расчёт 101 копейки | 101 | 1 рубль 1 копейка |
| 5 | Сохранение поля ввода | \*Данные введены в поля ввода; закрытие и открытие программы\* | \*Данные были выведены вновь\* |
| 6 | Вывод задания | Нажатие кнопки «Показать задание» | \*вывод окна с заданием\* |

### Код программы

Код программы находится по ссылке https://github.com/mxkmn/RepoForMyTeacher\_CSharpEdition/tree/main/Lab2Num1Var12

Файл с автотестами:

|  |
| --- |
| using Microsoft.VisualStudio.TestTools.UnitTesting;  namespace Lab2Num1Var12.Tests {  [TestClass()]  public class Form1Tests {  [TestMethod()]  public void GetMoneyTest1() {  Assert.AreEqual("15 рублей ровно", Logic.GetMoney(1500));  }  [TestMethod()]  public void GetMoneyTest2() {  Assert.AreEqual("1 копейка", Logic.GetMoney(1));  }  [TestMethod()]  public void GetMoneyTest3() {  Assert.AreEqual("99 рублей 99 копеек", Logic.GetMoney(9999));  }  [TestMethod()]  public void GetMoneyTest4() {  Assert.AreEqual("1 рубль 1 копейка", Logic.GetMoney(101));  }  }  } |

### Тесты

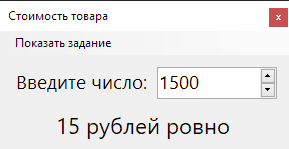


Рисунок 1 – Тест 1

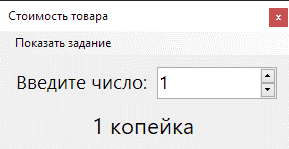


Рисунок 2 – Тест 2

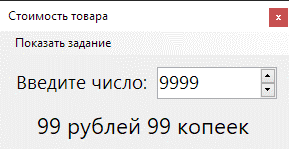


Рисунок 3 – Тест 3

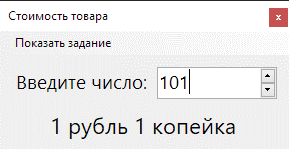


Рисунок 4 – Тест 4

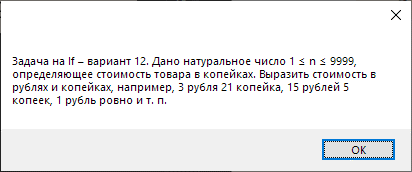


Рисунок 5 – Тест 6

### Постановка задачи

**Задача на For** – вариант 14**.** Дана последовательность натуральных чисел. Определить, есть ли в последовательности хотя бы одна n-ка одинаковых “соседних” чисел (n и элементы последовательности вводятся с клавиатуры). В случае положительного ответа определить порядковые номера чисел первой из таких пар.

Программа должна иметь кнопку «Очистить последовательность», «Прочитать задание» и уметь сохранять непустые данные. Данные должны проверяться автоматически. Автотесты должны присутствовать.

### Словесное описание алгоритма

1. При запуске получаем сохранённые с предыдущей сессии значения numsRowSaved и equalNumsSaved и добавляем в поля numsRow и equalNums. Вызываем метод DataProcessing()
2. При вводе данных в поля (нажатие клавиши или изменение поля) также вызываем метод DataProcessing()
3. С помощью метода DataProcessing() получаем необходимую информацию с помощью метода GetN() и задаём полученное значение полю outText, а также локально сохраняем новые данные (только непустые поля)
4. С помощью метода GetN(numsStr, equals) получаем строку с информацией:
   1. Инициализируем переменные lastNum для последнего числа и sameNums для их количества, задаём значение Int32.MinValue обоим. Инициализируем также isFound как false и outStr как ""
   2. Разбиваем строку numsStr на подстроки, добавляем в новый массив nums
   3. Запускаем цикл, который сработает столько раз, сколько строк содержится в массиве nums:
      1. Если строка, полученная из массива, не является целым числом, выводится "Ошибка ввода!"
      2. Если ошибка не получена, работаем с числом как переменной newNum
      3. Если newNum является ненатуральным числом, выводится "Возможен ввод только натуральных чисел!"
      4. Если проблем не возникло, проверяем, является ли новое число отличающимся от предыдущего. Если не является, добавляем к sameNums единицу, иначе устанавливаем lastNum значение newNum и sameNums значение 1
      5. Если sameNums равно nums и isFound равно false, то искомая последовательность найдена – сохраняем соответствующее сообщение в outStr и устанавливаем isFound значение true для того, чтобы в дальнейшем не добавлять в outStr новые данные
   4. Если isFound равно true, то выводим outStr, в ином случае n-ка не найдена, поэтому выводим соответствующее сообщение
5. Внутри интерфейса прописаны минимальные и максимальные значения для поля ввода количества необходимых чисел, поэтому дополнительная проверка не требуется
6. При повторном запуске программы она вводит данные предыдущей сессии, если они имеются
7. Кнопка «Очистить последовательность» удаляет данные из поля ввода последовательности и вызывает DataProcessing()
8. Кнопка «Прочитать задание» показывает задание, по которому создавалась программа.

### Таблица спецификаций

|  |  |  |
| --- | --- | --- |
| Имя | Назначение | Тип/Диапазон |
| Входные величины | | |
| equalNums | Последовательность чисел | Текстовое поле ввода, строка |
| numsRow | Количество чисел подряд, которые необходимо проверить на схожесть | Числовое поле ввода, целое число от 1 до 100 |

### Таблица тестов

| Номер теста | Что проверяем | Входные данные | Выходные данные |
| --- | --- | --- | --- |
| 1 | Проверка несоседства | 3  123 21 1231231231 1231231231 | 3-ка чисел не найдена. |
| 2 | Проверка соседства | 2  123 21 1231231231 1231231231 | 2-ка чисел 1231231231 найдена. Она была введена с 3 по 4 число. |
| 3 | Проверка ввода слова | 2  123 вфы 1231231231 1231231231 | Ошибка ввода! |
| 4 | Проверка ввода числа, не входящего в нужный промежуток | 2  123 -21 1231231231 1231231231 | Возможен ввод только натуральных чисел! |
| 5 | Проверка ввода пустой строки | 2 | Ничего не введено |
| 6 | Сохранение непустых полей ввода | \*Данные введены в поля ввода; закрытие и открытие программы\* | \*Данные были выведены вновь\* |
| 7 | Вывод задания | Нажатие кнопки «Показать задание» | \*вывод окна с заданием\* |

### Код программы

Код программы находится по ссылке https://github.com/mxkmn/RepoForMyTeacher\_CSharpEdition/tree/main/Lab2Num2Var14

Файл с автотестами:

|  |
| --- |
| using Microsoft.VisualStudio.TestTools.UnitTesting;  namespace Lab2Num2Var14.Tests {  [TestClass()]  public class Form1Tests {  [TestMethod()]  public void GetNTest1() {  Assert.AreEqual("2-ка чисел 1231231231 найдена. Она была введена с 3 по 4 число.", Logic.GetN("123 21 1231231231 1231231231", 2));  }  [TestMethod()]  public void GetNTest2() {  Assert.AreEqual("Ошибка ввода!", Logic.GetN("123 вфы 1231231231 1231231231", 2));  }  [TestMethod()]  public void GetNTest3() {  Assert.AreEqual("Возможен ввод только натуральных чисел!", Logic.GetN("123 -21 1231231231 1231231231", 2));  }  [TestMethod()]  public void GetNTest4() {  Assert.AreEqual("3-ка чисел не найдена.", Logic.GetN("123 21 1231231231 1231231231", 3));  }  }  } |

### Тесты

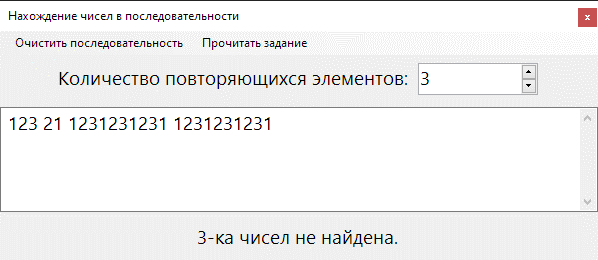


Рисунок 6 – Тест 1

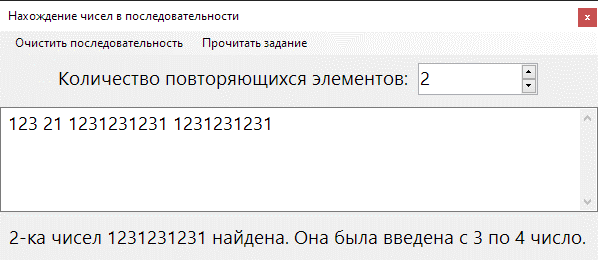


Рисунок 7 – Тест 2

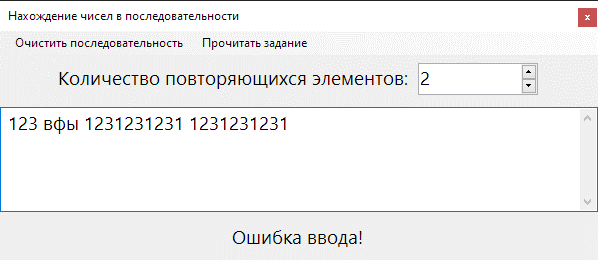


Рисунок 8 – Тест 3

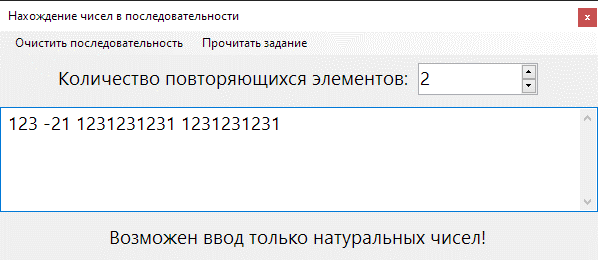


Рисунок 9 – Тест 4

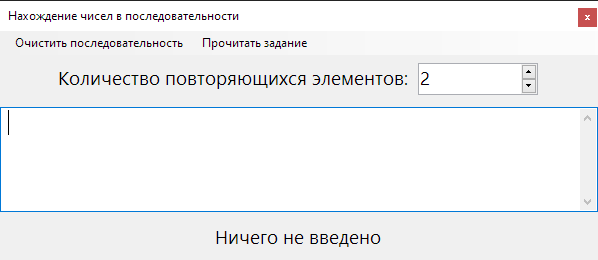


Рисунок 10 – Тест 5

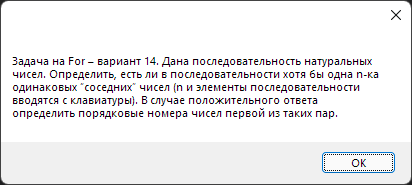


Рисунок 11 – Тест 7