МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

федеральное государственное бюджетное образовательное учреждение высшего образования

**«Вологодский государственный университет»**

**Институт математики, естественных и компьютерных наук**

**Информатика и вычислительная техника**

**ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №1**

«Управление разработкой и разработка программного продукта в группах»

Дисциплина: «Технология разработки программного обеспечения»

Направление подготовки: 09.03.01. Информатика и вычислительная техника

|  |  |
| --- | --- |
| Руководитель | Сергушичева А.П. |
| Выполнили студенты | Пчелкина О.С. |
| Группа, курс | 4Б09 ВМ-41 |
| Дата сдачи | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| Дата защиты | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  *(подпись преподавателя)* |

Вологда

2023 г.

**Цель работы***:* освоение методов организации группы разработчиков программного обеспечения, приобретение навыков работы в группе.

**Обычная бригада:**

1. Ольга – старший программист;
2. Ольга – младший программист;
3. Ольга – тестировщик;

**Программная задача:**

Программа проверки а)устного счета, б)умножения, в)деления. Программа должна генерировать вопрос, проверять ответ, а также по завершении работы выдавать статистику в виде графика.

История разработки

**Приказы и соглашения:**

1. Разработка программы жизненного цикла ПО;
2. Техническое задание;
3. Документация по интеграции программного обеспечения;
4. Результаты тестирования готового продукта.

**Обоснование выбора языка разработки:**

Язык программирования WPF (Windows Presentation Foundation) был выбран для создания интерфейса программы проверки устного счета, умножения и деления с выводом статистики в виде графика в связи с рядом преимуществ:

* Кроссплатформенность: WPF поддерживается на различных операционных системах, включая Windows и macOS. Это означает, что разработанное приложение может работать на различных устройствах и операционных системах, что повышает его удобство использования.
* Большой набор готовых элементов управления: WPF содержит множество встроенных элементов управления, которые позволяют быстро и просто создавать интерфейсы. Например, в данном проекте использовались элементы Label, TextBox, Button и PieChart.
* Гибкость и расширяемость: WPF позволяет создавать интерфейсы с использованием XAML-разметки, что позволяет отделить интерфейс от логики программы. Это упрощает разработку и поддержку кода. Кроме того, WPF имеет широкий набор расширяемых библиотек, которые позволяют создавать настраиваемые элементы управления.
* Высокая производительность: WPF использует аппаратное ускорение для отображения элементов управления, что обеспечивает высокую производительность и быструю отрисовку интерфейса.

В итоге, использование WPF в данном проекте обеспечило быструю и удобную разработку графического интерфейса для программы проверки устного счета, умножения и деления с возможностью отображения статистики в виде графика.

**Внешняя спецификация:**

Программа проверки устного счета представляет собой приложение, которое генерирует вопросы по математическим операциям (сложение, вычитание, умножение, деление) и проверяет ответы пользователя на правильность. Программа должна иметь графический интерфейс пользователя, включающий поле ввода ответа, кнопку "Проверить", а также окно статистики с графиком количества правильных и неправильных ответов.

**Алгоритм работы программы:**

1. Сгенерировать два случайных числа.
2. Сгенерировать случайную операцию из списка доступных операций: сложение, вычитание, умножение, деление.
3. Отобразить вопрос пользователю, используя сгенерированные числа и операцию.
4. Получить ответ пользователя.
5. Проверить ответ пользователя на правильность.
6. Отобразить сообщение о правильности ответа и обновить график статистики.

**Модульная структура программы:**

1. Модуль генерации вопросов - генерирует случайные числа и операции для задания вопросов.
2. Модуль проверки ответов - проверяет ответы пользователя на правильность.
3. Модуль интерфейса пользователя - отображает вопросы, получает ответы от пользователя и отображает статистику.

**Тестирование программы:**

*Описание сценариев в BDD стиле*

Сценарий: Проверка корректности выполнения операции сложения

Дано:

* Значение первого числа равно 5;
* Значение второго числа равно 3;

Выбрана операция сложения.

Когда: пользователь вводит значения чисел и нажимает кнопку "Check Answer".

Тогда:

* Счетчик «Correct» увеличивается на 1, если ответ равен 8, счетчик «Count» увеличивается на 1;
* Счетчик «Correct» остается неизменным, если ответ равен 8, счетчик «Count» увеличивается на 1;

Сценарий: Проверка корректности выполнения операции вычитания

Дано:

* Значение первого числа равно 10;
* Значение второго числа равно 4;

Выбрана операция вычитания.

Когда: пользователь вводит значения чисел и нажимает кнопку "Check Answer".

Тогда:

* Счетчик «Correct» увеличивается на 1, если ответ равен 6, счетчик «Count» увеличивается на 1;
* Счетчик «Correct» остается неизменным, если ответ не равен 6, счетчик «Count» увеличивается на 1.

Сценарий: Проверка корректности выполнения операции умножения

Дано:

* Значение первого числа равно 7;
* Значение второго числа равно 8;

Выбрана операция умножения.

Когда: пользователь вводит значения чисел и нажимает кнопку "Check Answer".

Тогда:

* Счетчик «Correct» увеличивается на 1, если ответ равен 56, счетчик «Count» увеличивается на 1;
* Счетчик «Correct» остается неизменным, если ответ не равен 56, счетчик «Count» увеличивается на 1.

Сценарий: Проверка корректности выполнения операции деления

Дано:

* Значение первого числа равно 20;
* Значение второго числа равно 5;

Выбрана операция деления.

Когда: пользователь вводит значения чисел и нажимает кнопку "Check Answer".

Тогда:

* Счетчик «Correct» увеличивается на 1, если ответ равен 4, счетчик «Count» увеличивается на 1;
* Счетчик «Correct» остается неизменным, если ответ не равен 4, счетчик «Count» увеличивается на 1.

**Перечень работ, выполненные участниками бригады:**

1. Ольга – старший программист.

Распределение обязанностей между участниками. Разработка модуля генерации вопросов. Разработка модуля проверки ответов. Интеграция модулей в единую систему.

1. Ольга – младший программист.

Проектирование пользовательского интерфейса. Разработка модуля интерфейса пользователя

1. Ольга –тестировщик.

Подготовка тестов к ПО и их реализация.

Пояснительная записка

**Введение.** В ходе лабораторной работы была разработана программа проверки а)устного счета, б)умножения, в)деления. Программа должна генерировать вопрос, проверять ответ, а также по завершении работы выдавать статистику в виде графика.

**Назначение и область применения**. Назначение программы проверки устного счета - помочь ученикам и студентам улучшить свои навыки математических вычислений, особенно в области быстрого устного счета.

Программа может быть использована в образовательных учреждениях, в дополнительном образовании, а также в домашних условиях для самостоятельного обучения. Она может помочь повысить уверенность и скорость в решении математических задач, что может быть полезно не только в школе, но и в жизни в целом.

Также программа может быть использована в качестве тренажера для подготовки к математическим олимпиадам или экзаменам, где быстрый устный счет может быть необходимым навыком для успешной сдачи теста.

**Технические характеристики.**

Постановка задачи: программа проверки а)устного счета, б)умножения, в)деления. Программа должна генерировать вопрос, проверять ответ, а также по завершении работы выдавать статистику в виде графика.

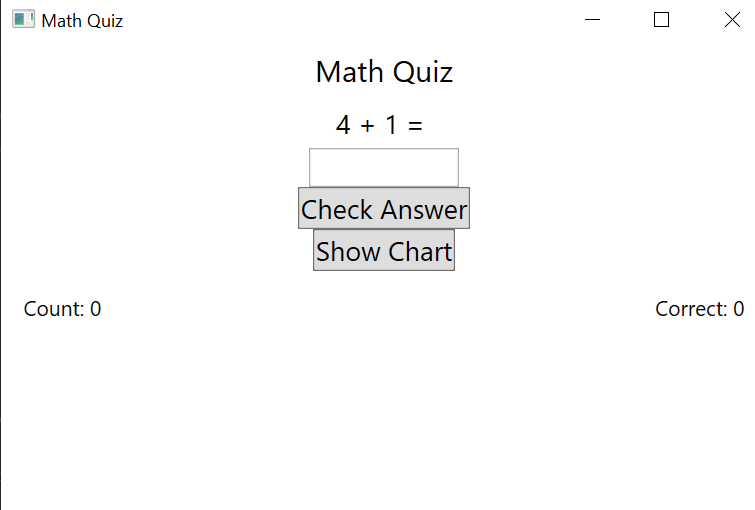
* Язык программирования: C#, WPF.
* Входные данные: математические операции.
* Выходные данные: количество правильных ответов, статистика правильных ответов.

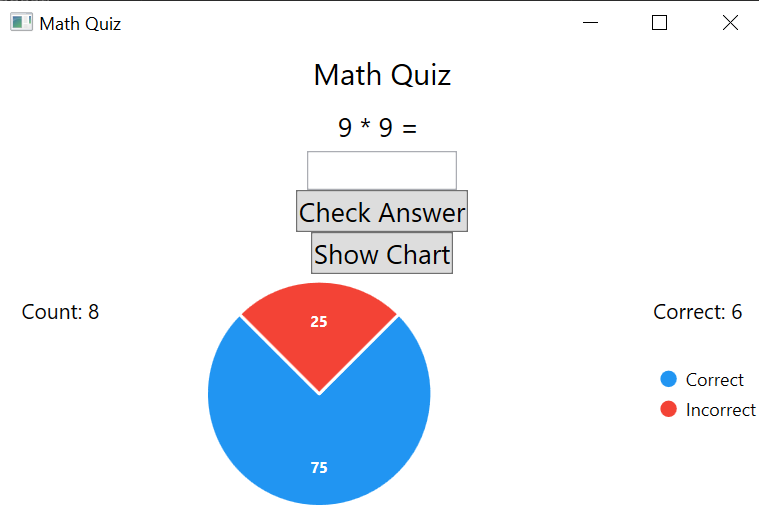
**Минимальные требования к оборудованию:**

* Процессор Pentium с частотой 233 МГц или более быстрый (рекомендуется не менее 300 МГц);
* Не менее 64 МБ оперативной памяти (рекомендуется не менее 128 МБ);
* Не менее 1,5 ГБ свободного места на жестком диске.

**Эксплуатационные документы:**

* Руководство пользователя
* Руководство по развертыванию приложения
* Соглашение о процессе и этапах поддержки приложения
* Требования к безопасности

****

****

**Текст программы:**

<Window x:Class="MathQuiz.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:liveCharts="clr-namespace:LiveCharts.Wpf;assembly=LiveCharts.Wpf"

Title="Math Quiz" Height="350" Width="525">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="Auto"/>

<RowDefinition Height="Auto"/>

<RowDefinition Height="Auto"/>

<RowDefinition Height="Auto"/>

<RowDefinition Height="Auto"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<Label Grid.Row="0" Content="Math Quiz" FontSize="20" HorizontalAlignment="Center"/>

<Label Grid.Row="1" x:Name="questionLabel" Content="" FontSize="18" HorizontalAlignment="Center" VerticalAlignment="Center"/>

<TextBox Grid.Row="2" x:Name="AnswerTextBox" FontSize="18" HorizontalAlignment="Center" VerticalAlignment="Center" Width="100"/>

<Button Grid.Row="3" x:Name="CheckAnswerButton" Content="Check Answer" FontSize="18" HorizontalAlignment="Center" VerticalAlignment="Center" Click="CheckAnswerButton\_Click"/>

<Button Grid.Row="4" x:Name="ShowChartButton" Content="Show Chart" FontSize="18" HorizontalAlignment="Center" VerticalAlignment="Center" Click="ShowChartButton\_Click"/>

<Grid Grid.Row="5">

<liveCharts:PieChart x:Name="StatisticsChart" LegendLocation="Right" />

</Grid>

<Label Grid.Row="5" x:Name="CountLabel" Content="Count: 0" FontSize="14" HorizontalAlignment="Left" VerticalAlignment="Top" Margin="10,10,0,0"/>

<Label Grid.Row="5" x:Name="CorrectLabel" Content="Correct: 0" FontSize="14" HorizontalAlignment="Right" VerticalAlignment="Top" Margin="0,10,10,0"/>

</Grid>

</Window>

using System;

using System.Windows;

using LiveCharts;

using LiveCharts.Wpf;

namespace MathQuiz

{

public partial class MainWindow : Window

{

private int count;

private int correctCount;

private Random random = new Random();

private int questionType;

private int firstNumber;

private int secondNumber;

private int answer;

public MainWindow()

{

InitializeComponent();

count = 0;

correctCount = 0;

questionType = 0;

firstNumber = 0;

secondNumber = 0;

answer = 0;

UpdateQuestion();

}

private void CheckAnswerButton\_Click(object sender, RoutedEventArgs e)

{

int userAnswer;

if(int.TryParse(AnswerTextBox.Text, out userAnswer))

{

if(userAnswer == answer)

{

correctCount++;

CorrectLabel.Content = $"Correct: {correctCount}";

}

}

count++;

CountLabel.Content = $"Count: {count}";

UpdateQuestion();

}

private void UpdateQuestion()

{

questionType = random.Next(0, 3);

firstNumber = random.Next(0, 11);

secondNumber = random.Next(1, 11);

switch(questionType)

{

case 0:

questionLabel.Content = $"{firstNumber} + {secondNumber} = ";

answer = firstNumber + secondNumber;

break;

case 1:

questionLabel.Content = $"{firstNumber} - {secondNumber} = ";

answer = firstNumber - secondNumber;

break;

case 2:

questionLabel.Content = $"{firstNumber} \* {secondNumber} = ";

answer = firstNumber \* secondNumber;

break;

case 3:

questionLabel.Content = $"{firstNumber} / {secondNumber} = ";

answer = firstNumber / secondNumber;

break;

}

AnswerTextBox.Text = "";

}

private void ShowChartButton\_Click(object sender, RoutedEventArgs e)

{

double correctPercentage = (double)correctCount / count \* 100;

double incorrectPercentage = 100 - correctPercentage;

SeriesCollection series = new SeriesCollection

{

new PieSeries

{

Title = "Correct",

Values = new ChartValues<double> { correctPercentage },

DataLabels = true

},

new PieSeries

{

Title = "Incorrect",

Values = new ChartValues<double> { incorrectPercentage },

DataLabels = true

}

};

StatisticsChart.Series = series;

}

}

}