Лабораторная работа №4

«Общие принципы разработки тестовой программы с использованием массивов и записью результатов во внешние программы»

**Задание**

Разработать тестовую программу по теме «Тестирование на тему "Основные этапы развития информационного общества"» на основе использования следующих элементов управления: TextBox, CheckBox, ComboBox, ListBox, TextBox+Label. Программа должна обладать следующими функциональными характеристиками:

1. Вывод результатов тестирования в элемент интерфейса DataGridView;
2. Построение столбчатой диаграммы, отражающей результаты ответа пользователя;
3. Вывод результатов теста в Word, Excel и Блокнот;
4. Информирование пользователя о результатах тестирования в пятибалльной шкале;
5. Производить поиск правильных и неправильных ответов в массиве при помощи LINQ-запросов;
6. Сохранение имени пользователя, номера задания, правильности ответа на задание, даты и времени начала выполнения задания и затраченное на него время в СУБД “SQLite”;
7. Возможность просмотра и фильтрации результатов тестирования из базы данных по имени пользователя, номеру задания, правильности ответа и затраченному на выполнение времени;
8. Возможность установки приложения в указанную директорию на персональный компьютер при помощи установщика приложений “Microsoft Visual Studio Installer Projects 2022”.
   1. **Руководство пользователя**

Для того, чтобы начать работу с программой, следует воспользоваться установщиком для инсталляции приложения на персональный компьютер. Сначала необходимо открыть файл “Installer.msi”, после чего пользователь увидит окно установщика (см. рисунок 1).

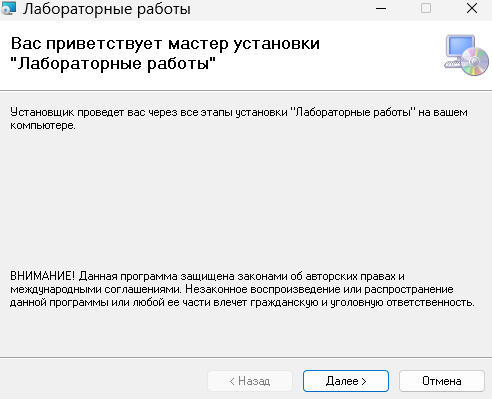


Рисунок 1 – Окно установщика

После ознакомления с приведённой информацией следует нажать кнопку “Далее >”. В результате этого действия инсталлятор предложит выбрать путь установки, посмотреть свободное место на накопителях, а также предоставит возможность настроить установку программы для всех пользователей ОС Windows (см. рисунок 2).

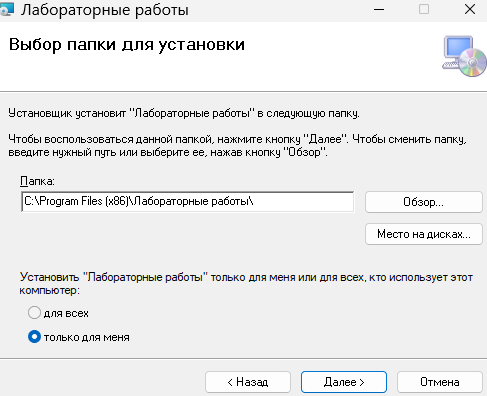


Рисунок 2 – Параметры установки

При повторном нажатии кнопки “Далее >” инсталлятор сообщит, что всё готово для установки программы на ПК, после чего при очередном нажатии на ту же кнопку начнётся процесс установки (см. рисунок 3).

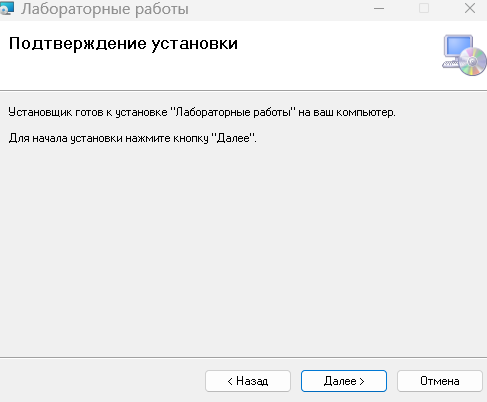


Рисунок 3 – Готовность к установке

После незначительного ожидания инсталлятор сообщит, что установка выполнена успешно (см. рисунок 4).

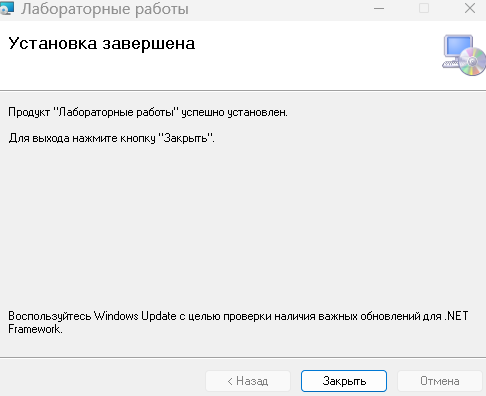


Рисунок 4 – Успешное завершение установки

В результате вышеописанных действий на рабочем столе пользователя появится ярлык-ссылка, нажав на который пользователь сможет запустить данное приложение (см. рисунок 5).



Рисунок 5 – Ярлык на рабочем столе

* 1. **Описание элементов тестирования**

Выполним описание использованных в программе элементов управления:

1. TextBox.

Интерфейс вопроса, выполненного на основе данного компонента, показан на рисунке 6. Данный вид задания относится к типу текстовых заданий в виде ответа на вопрос.

Преимущества: минимальная возможность угадать правильный ответ, универсальность, объективная оценка знаний;

Недостатки: сложность в построении вопроса, непригодность для оценки глубоких знаний тестируемого.

Программный код, реализующий данный вопрос, представлен ниже:

using MainDLL;

using System;

using System.Windows.Forms;

namespace MainApplication

{

public partial class TestFormTextBox : TestFormBase

{

private string rightAnswer;

public TestFormTextBox(string question1\_, string rightAnswer)

{

InitializeComponent();

this.rightAnswer = rightAnswer;

question.Text = question1\_;

ok.Click += new System.EventHandler(ok\_Click);

return;

}

public static void HandleOkPress(TextBox answer, string rightAnswer, Label status, Button ok, Button next)

{

if (answer.Text.Length == 0)

{

Utilities.SetAnswerStatus(status, 2);

ok.Enabled = true;

next.Enabled = false;

answer.Focus();

return;

}

double similarity = Convert.ToDouble(answer.Text == rightAnswer) \* 100;

// является ли введённое значение числом

try

{

Convert.ToDouble(answer.Text);

}

catch (FormatException)

{

similarity = Utilities.CompareStrings(answer.Text.ToLower(), rightAnswer.ToLower());

}

if (similarity > 85)

{

Utilities.SetAnswerStatus(status, 1);

TestLabMainForm.testResults[TestLabMainForm.currentTask] = true;

}

else

{

TestLabMainForm.testResults[TestLabMainForm.currentTask] = false;

Utilities.SetAnswerStatus(status, 0);

}

TestLabMainForm.AddRowSQL(startTime);

ok.Enabled = false;

next.Enabled = true;

next.Focus();

}

private void ok\_Click(object sender, EventArgs e)

{

HandleOkPress(answer, rightAnswer, status, ok, next);

}

private void TestFormTextBox\_Paint(object sender, PaintEventArgs e)

{

answer.Focus();

}

private void answer\_KeyDown(object sender, KeyEventArgs e)

{

if (e.KeyCode == Keys.Enter)

{

ok\_Click(sender, e);

}

}

}

}

1. CheckBox.

Интерфейс вопроса, выполненного на основе данного компонента, показан на рисунке 7. Данный вид задания относится к типу заданий с оценкой суждений.

Преимущества: малая временная затратность, высокая надёжность результатов, объективность, простота составления;

Недостатки: возможность угадывания, необходимость усложнения задачи для минимизации вероятности случайного ответа, отсутствие универсальности.

Программный код, реализующий данный вопрос, представлен ниже:

using MainDLL;

using System;

using System.Windows.Forms;

namespace MainApplication

{

public partial class TestFormCheckBoxes : TestFormBase

{

private string rightAnswers;

private CheckBox[] cBoxes;

public TestFormCheckBoxes(string question\_, string rightAnswer, string variants)

{

InitializeComponent();

this.rightAnswers = rightAnswer;

cBoxes = new CheckBox[4] { check1, check2, check3, check4 };

ok.Click += new System.EventHandler(ok\_Click);

question.Text = question\_;

string[] vars = variants.Split('|');

for (int i = 0; i < cBoxes.Length; i++)

{

cBoxes[i].Text = vars[i];

}

}

private void ok\_Click(object sender, EventArgs e)

{

bool isRightAnswerFlag = true;

foreach (CheckBox cBox in cBoxes)

{

if (rightAnswers.Contains(cBox.Text) && cBox.Checked)

{

Utilities.SetControlStatus(cBox, true);

}

else if (cBox.Checked || (rightAnswers.Contains(cBox.Text) && !cBox.Checked))

{

isRightAnswerFlag = false;

Utilities.SetControlStatus(cBox, false);

}

}

if (isRightAnswerFlag)

{

Utilities.SetAnswerStatus(status, 1);

TestLabMainForm.testResults[TestLabMainForm.currentTask] = true;

}

else

{

Utilities.SetAnswerStatus(status, 0);

TestLabMainForm.testResults[TestLabMainForm.currentTask] = false;

}

TestLabMainForm.AddRowSQL(startTime);

}

private void CheckedChanged(object sender, EventArgs e)

{

ok.Focus();

}

}

}

1. ComboBox.

Интерфейс вопроса, выполненного на основе данного компонента, показан на рисунке 8. Данный вид задания относится к типу заданий с выбором правильного ответа из нескольких предложенных.

Преимущества: объективность, высокая надёжность результата, объективность, универсальность;

Недостатки: охват только фактов (не сущности) вопроса, сложность конструирования вопроса, зависимость от лексических способностей тестируемого.

Программный код, реализующий данный вопрос, представлен ниже:

using MainDLL;

using System;

namespace MainApplication

{

public partial class TestFormComboBox : TestFormBase

{

private string rightAnswer;

public TestFormComboBox(string question\_, string rightAnswer, string variants)

{

InitializeComponent();

this.rightAnswer = rightAnswer;

ok.Click += new System.EventHandler(ok\_Click);

question.Text = question\_;

foreach (string variant in variants.Split('|'))

{

answers.Items.Add(variant);

}

answers.Text = answers.Items[0].ToString();

}

private void ok\_Click(object sender, EventArgs e)

{

if (answers.GetItemText(answers.SelectedItem) == rightAnswer)

{

Utilities.SetAnswerStatus(status, 1);

TestLabMainForm.testResults[TestLabMainForm.currentTask] = true;

}

else

{

Utilities.SetAnswerStatus(status, 0);

TestLabMainForm.testResults[TestLabMainForm.currentTask] = false;

}

TestLabMainForm.AddRowSQL(startTime);

}

private void answers\_SelectedIndexChanged(object sender, EventArgs e)

{

ok.Focus();

}

}

}

1. ListBox. Интерфейс вопроса, выполненного на основе данного компонента, показан на рисунке 9. Данный вид задания относится к типу заданий на соответствие.

Преимущества: точность результата, высокая надёжность результатов;

Недостатки: непригодность к глубокой проверке знаний, сложность составления вопросов.

Программный код, реализующий данный вопрос, представлен ниже:

using MainDLL;

using System;

using System.Linq;

using System.Windows.Forms;

namespace MainApplication

{

public partial class TestFormListBox : TestFormBase

{

private string[] firstGroup, secondGroup;

private ListBox movedFrom;

private bool isRecieved;

public TestFormListBox(string question\_, string firstGroup\_, string secondGroup\_, string groupNames)

{

InitializeComponent();

startTime = DateTime.Now;

ok.Click += new System.EventHandler(ok\_Click);

this.firstGroup = firstGroup\_.Split('|');

this.secondGroup = secondGroup\_.Split('|');

question.Text = question\_;

containerName1.Text = groupNames.Split('|')[0];

containerName2.Text = groupNames.Split('|')[1];

Random rnd = new Random();

double random;

string[] minArray, maxArray;

if (firstGroup\_.Length < secondGroup\_.Length)

{

minArray = firstGroup;

maxArray = secondGroup;

}

else

{

minArray = secondGroup;

maxArray = firstGroup;

}

for (int i = 0; i < minArray.Length; i++)

{

random = rnd.NextDouble();

if (random < 0.5)

{

answers.Items.Add(minArray[i]);

answers.Items.Add(maxArray[i]);

}

else

{

answers.Items.Add(maxArray[i]);

answers.Items.Add(minArray[i]);

}

}

for (int i = minArray.Length; i < maxArray.Length; i++)

{

answers.Items.Add(maxArray[i]);

}

answers.SelectedItems.Add(answers.Items[0]);

}

private void ok\_Click(object sender, EventArgs e)

{

if (answers.Items.Count != 0)

{

Utilities.SetAnswerStatus(status, 2);

ok.Enabled = true;

next.Enabled = false;

answers.Focus();

return;

}

bool isCorrectFlag = container1.Items.Count == firstGroup.Length && container2.Items.Count == secondGroup.Length;

for (int i = 0; i < container1.Items.Count; i++)

{

if (!this.firstGroup.Contains(container1.Items[i]))

isCorrectFlag = false;

}

for (int i = 0; i < container2.Items.Count; i++)

{

if (!this.secondGroup.Contains(container2.Items[i]))

isCorrectFlag = false;

}

if (isCorrectFlag)

{

Utilities.SetAnswerStatus(status, 1);

TestLabMainForm.testResults[TestLabMainForm.currentTask] = true;

}

else

{

Utilities.SetAnswerStatus(status, 0);

TestLabMainForm.testResults[TestLabMainForm.currentTask] = false;

}

TestLabMainForm.AddRowSQL(startTime);

}

private void container\_MouseDown(object sender, MouseEventArgs e)

{

ListBox list = (ListBox)sender;

movedFrom = list;

isRecieved = false;

list.DoDragDrop(list.Text, DragDropEffects.Copy | DragDropEffects.Move);

}

private void container\_DragOver(object sender, DragEventArgs e)

{

ListBox list = (ListBox)sender;

e.Effect = DragDropEffects.Copy;

string text = e.Data.GetData(DataFormats.Text).ToString();

if (!list.Items.Contains(text) && !isRecieved)

{

list.Items.Add(text);

movedFrom.Items.Remove(text);

isRecieved = true;

ok.Focus();

}

}

}

}

1. TextBox+Label.

Интерфейс вопроса, выполненного на основе данного компонента, показан на рисунке 10. Данный вид задания относится к типу заданий на дополнение суждения.

Преимущества: минимальная возможность угадать правильный ответ, универсальность, объективная оценка знаний;

Недостатки: сложность в построении вопроса, непригодность для оценки глубоких знаний тестируемого.

Программный код, реализующий данный вопрос, представлен ниже:

using System;

using System.Windows.Forms;

namespace MainApplication

{

public partial class TestFormUnitedTextBox : TestFormBase

{

private string rightAnswer;

public TestFormUnitedTextBox(string question1\_, string rightAnswer, string question2\_)

{

InitializeComponent();

this.rightAnswer = rightAnswer;

question.Text = question1\_;

question2.Text = question2\_;

ok.Click += new System.EventHandler(ok\_Click);

return;

}

private void ok\_Click(object sender, EventArgs e)

{

TestFormTextBox.HandleOkPress(answer, rightAnswer, status, ok, next);

}

private void TestFormUnitedTextBox\_Paint(object sender, System.Windows.Forms.PaintEventArgs e)

{

answer.Focus();

}

private void answer\_KeyDown(object sender, System.Windows.Forms.KeyEventArgs e)

{

if (e.KeyCode == Keys.Enter)

{

ok\_Click(sender, e);

}

}

}

}

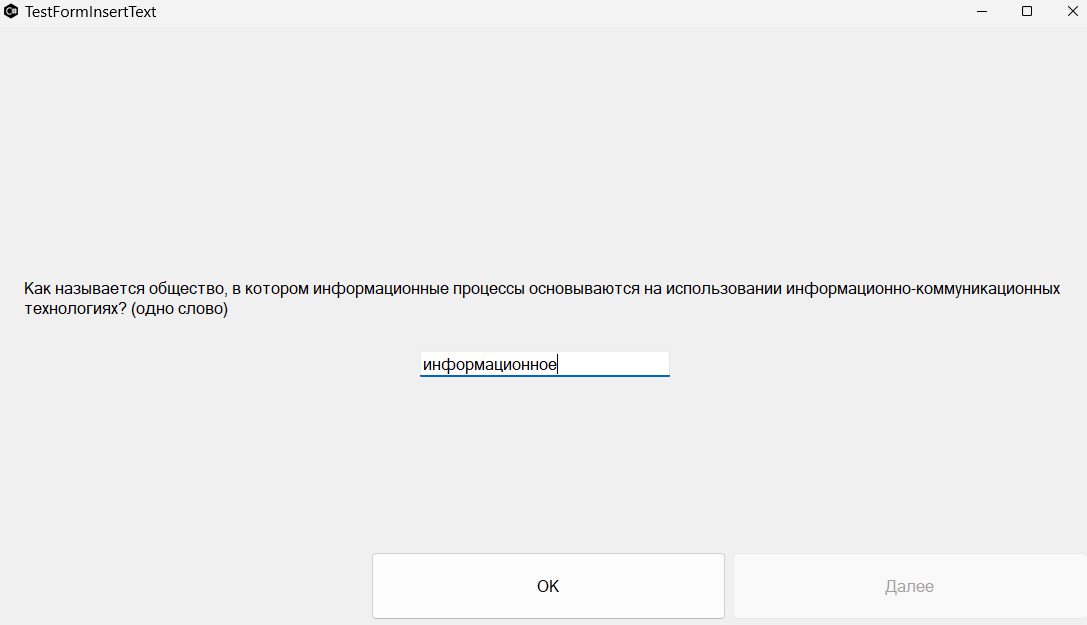


Рисунок 6 – Задание с ответом на вопрос

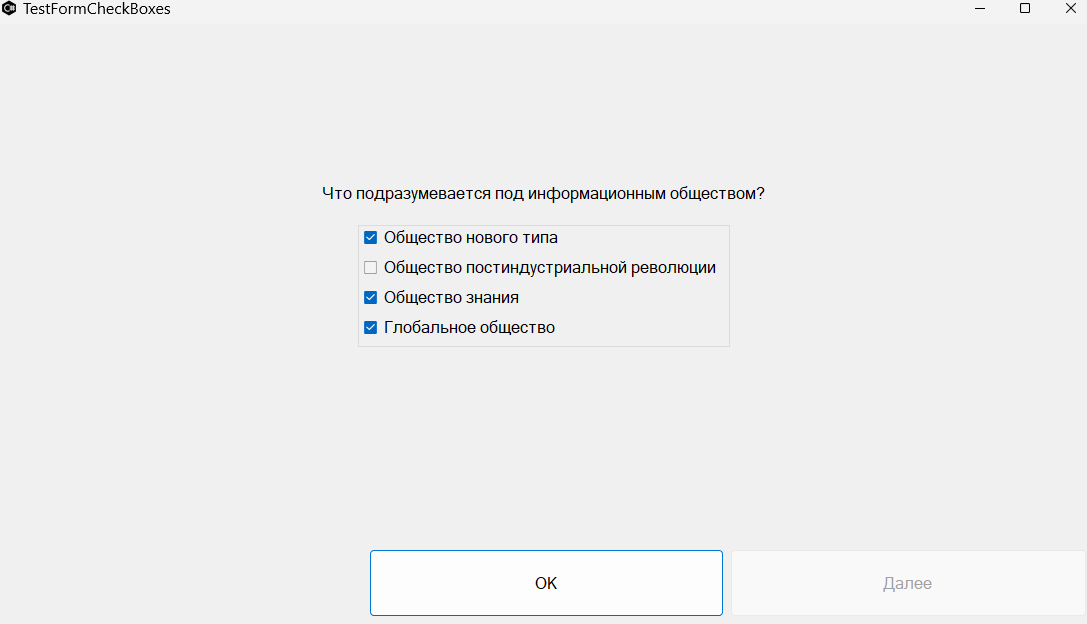


Рисунок 7 – Задание с выбором верных суждений

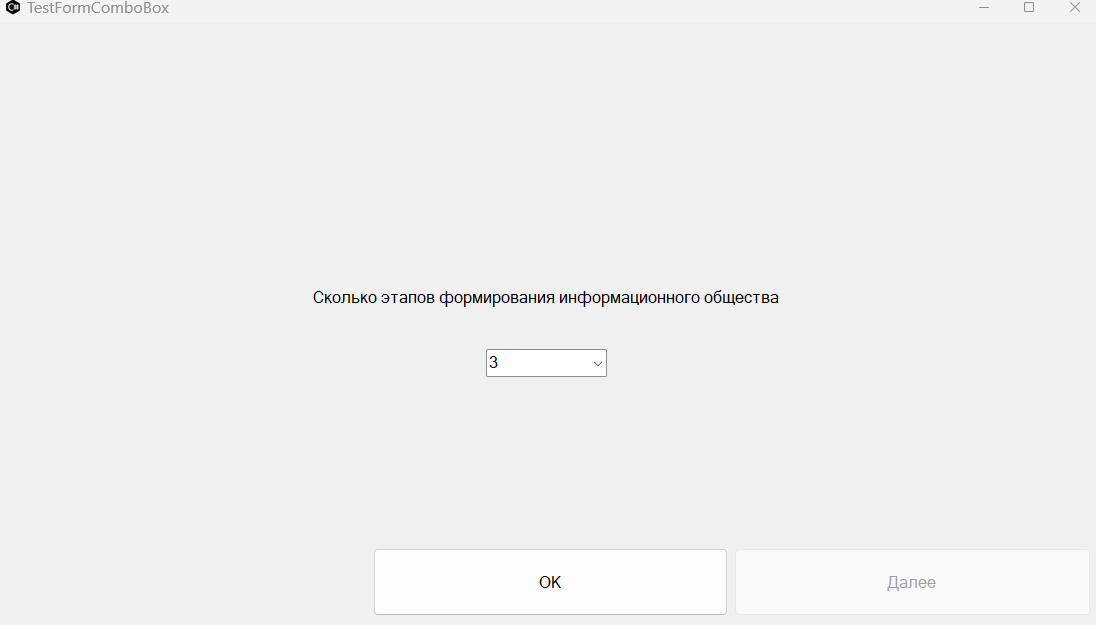


Рисунок 8 – Задание с выбором ответа из предложенных

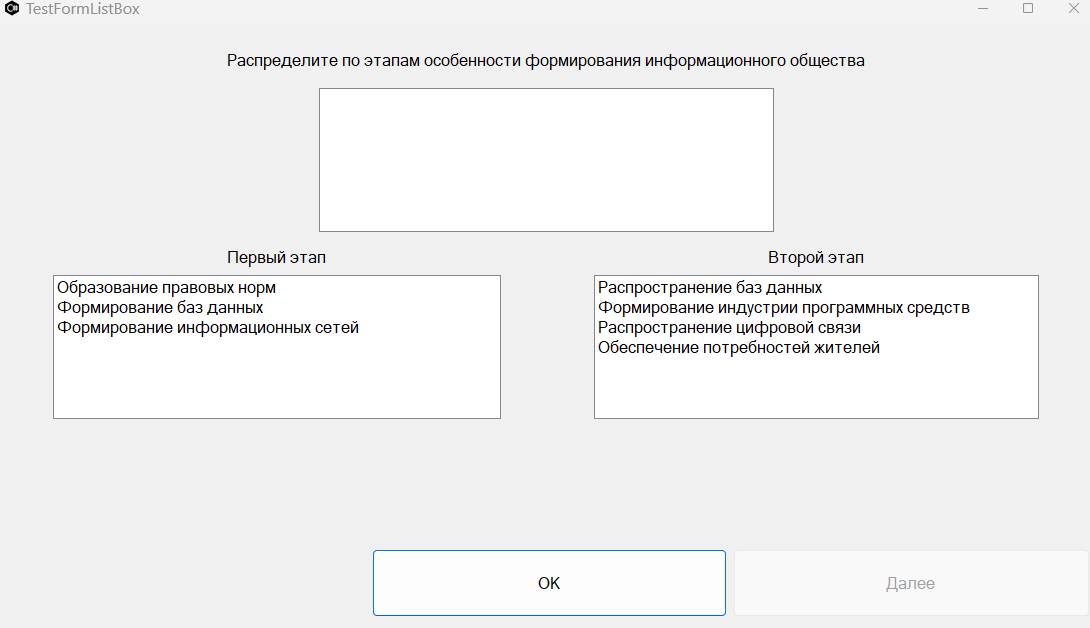


Рисунок 9 – Задание с соответствием

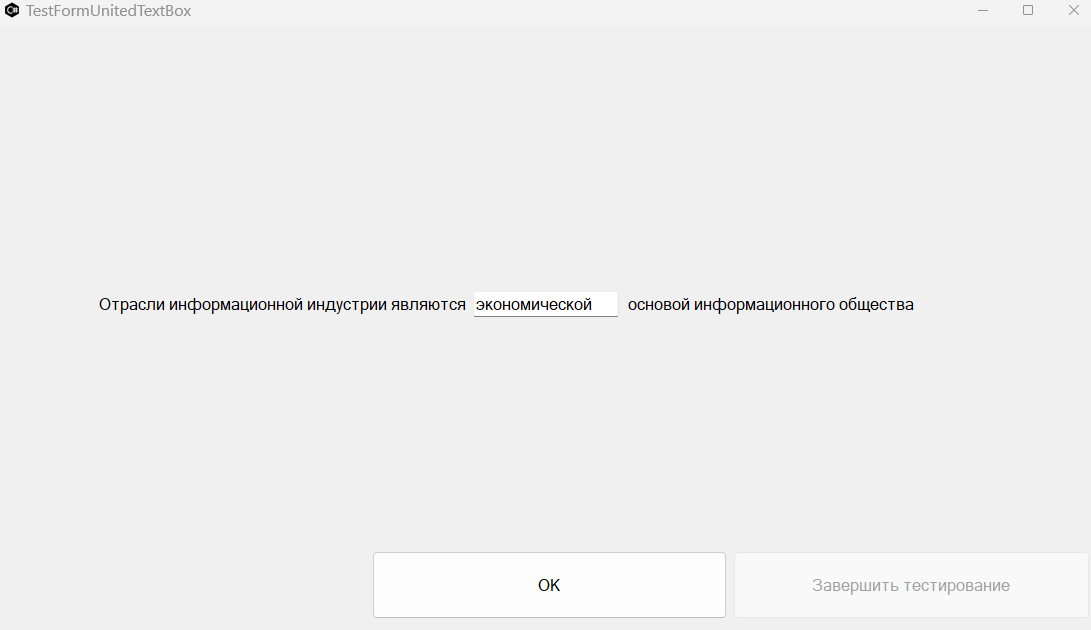


Рисунок 10 – Задание с дополнением суждения

**1.3. Описание итогов тестирования**

На рисунке 11 представлен результат работы программы, на котором отображены итоги тестирования, включающие в себя элемент DataGridView, отображающий правильность ответа на каждый из вопросов, график отношения правильных и неправильных ответов и Label, отформатированный с использованием LINQ-запроса к массиву с результатами тестирования.

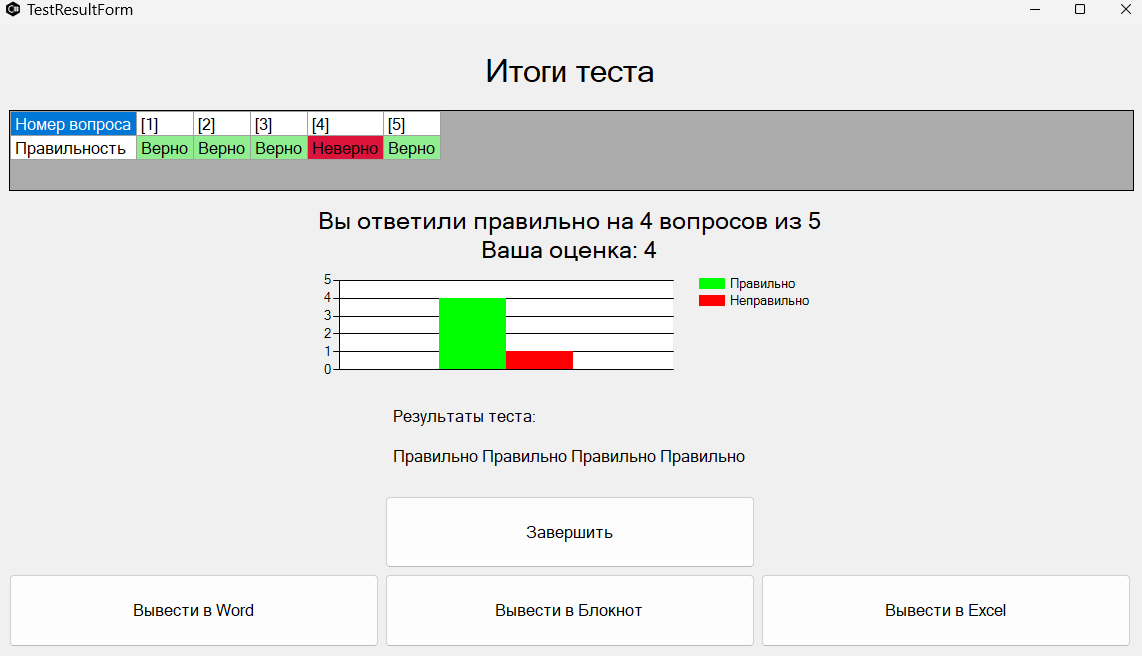


Рисунок 12 – Итоговая форма

Также на итоговой форме находятся две дополнительные кнопки, позволяющие вывести результаты тестирования в Word, Excel и Блокнот соответственно (см. рисунки 13-15).

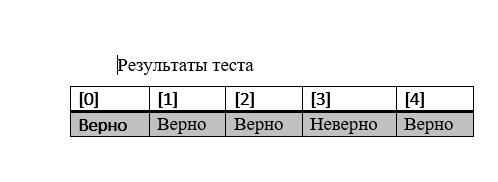
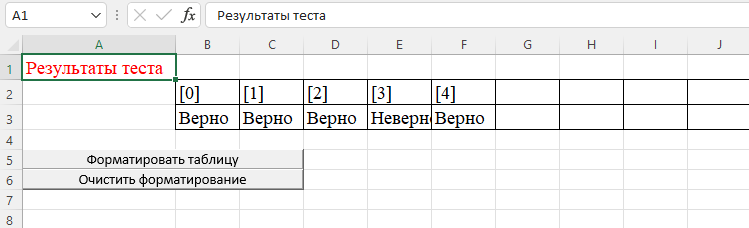


Рисунок 13 – Результаты тестирования в MS Word

Рисунок 14 – Результаты тестирования в MS Excel

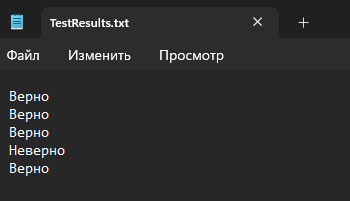


Рисунок 15 – Результаты тестирования в Блокноте

Программный код, реализующий данную итоговую форму, представлен ниже:

using System;

using System.Linq;

using System.Drawing;

using System.Windows.Forms;

using System.Collections.Generic;

namespace MainApplication

{

public partial class TestResultForm : BaseForm

{

public TestResultForm()

{

InitializeComponent();

bool[] testResults = TestLabMainForm.testResults;

results.ColumnCount = testResults.Length + 1;

results.RowCount = 2;

float rightAnswers = 0;

results.Rows[0].Cells[0].Value = "Номер вопроса";

results.Rows[1].Cells[0].Value = "Правильность";

results.Font = new Font("Microsoft Sans Serif", 12F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));

DataGridViewCellStyle wrongCellStyle = new DataGridViewCellStyle

{

BackColor = Color.Crimson

};

DataGridViewCellStyle rightCellStyle = new DataGridViewCellStyle

{

BackColor = Color.LightGreen

};

for (int i = 1; i < testResults.Length + 1; i++)

{

results.Rows[0].Cells[i].Value = "[" + i + "]";

if (testResults[i - 1])

{

results.Rows[1].Cells[i].Style = rightCellStyle;

results.Rows[1].Cells[i].Value = "Верно";

rightAnswers++;

}

else

{

results.Rows[1].Cells[i].Style = wrongCellStyle;

results.Rows[1].Cells[i].Value = "Неверно";

}

}

resultsChart.Series[0].Points.AddY(rightAnswers);

resultsChart.Series[1].Points.AddY(testResults.Length - rightAnswers);

note.Text = $"Вы ответили правильно на {rightAnswers} вопросов из {testResults.Length}\n" +

$"Ваша оценка: {Math.Max(Math.Round((double)(rightAnswers / testResults.Length) \* 5, 2), 2)}";

// вывод результатов в строку

resultLabel.Text = "Результаты теста: \r\n\r\n";

List<bool> list = new List<bool>(testResults);

var request = from s in list

where s == true

orderby s

select s;

foreach (bool result in request)

{

resultLabel.Text += "Правильно ";

}

}

private void ok\_Click(object sender, EventArgs e)

{

Close();

}

private string[] PrepareResultsArray()

{

string[] resultArray = new string[TestLabMainForm.testResults.Length];

for (int i = 0; i < TestLabMainForm.testResults.Length; i++)

{

resultArray[i] = TestLabMainForm.testResults[i] ? "Верно" : "Неверно";

}

return resultArray;

}

private void button1\_Click(object sender, EventArgs e)

{

DLLib2.IO.ArrayToWord(PrepareResultsArray(), "Результаты теста");

}

private void button2\_Click(object sender, EventArgs e)

{

string[][] resultArray = new string[1][] { new string[TestLabMainForm.testResults.Length] };

for (int i = 0; i < TestLabMainForm.testResults.Length; i++)

{

resultArray[0][i] = TestLabMainForm.testResults[i] ? "Верно" : "Неверно";

}

DLLib2.IO.ArraysToExcel(resultArray, new string[1] { "Результаты теста" });

}

private void button3\_Click(object sender, EventArgs e)

{

DLLib2.IO.ArrayToTXT(PrepareResultsArray(), "TestResults.txt", open:true);

}

}

}

**1.4. Описание менеджера результатов**

На рисунке 16 представлен менеджер результатов тестирования, на котором отображены выбранные результаты, их количество (включая количество правильных и неправильных), число отображаемых пользователей, среднее время выполнения отображаемых заданий, процентное отношение правильных и неправильных ответов, диаграмма данного соотношения по каждому из заданий и диаграмма среднего времени выполнения каждого из них.

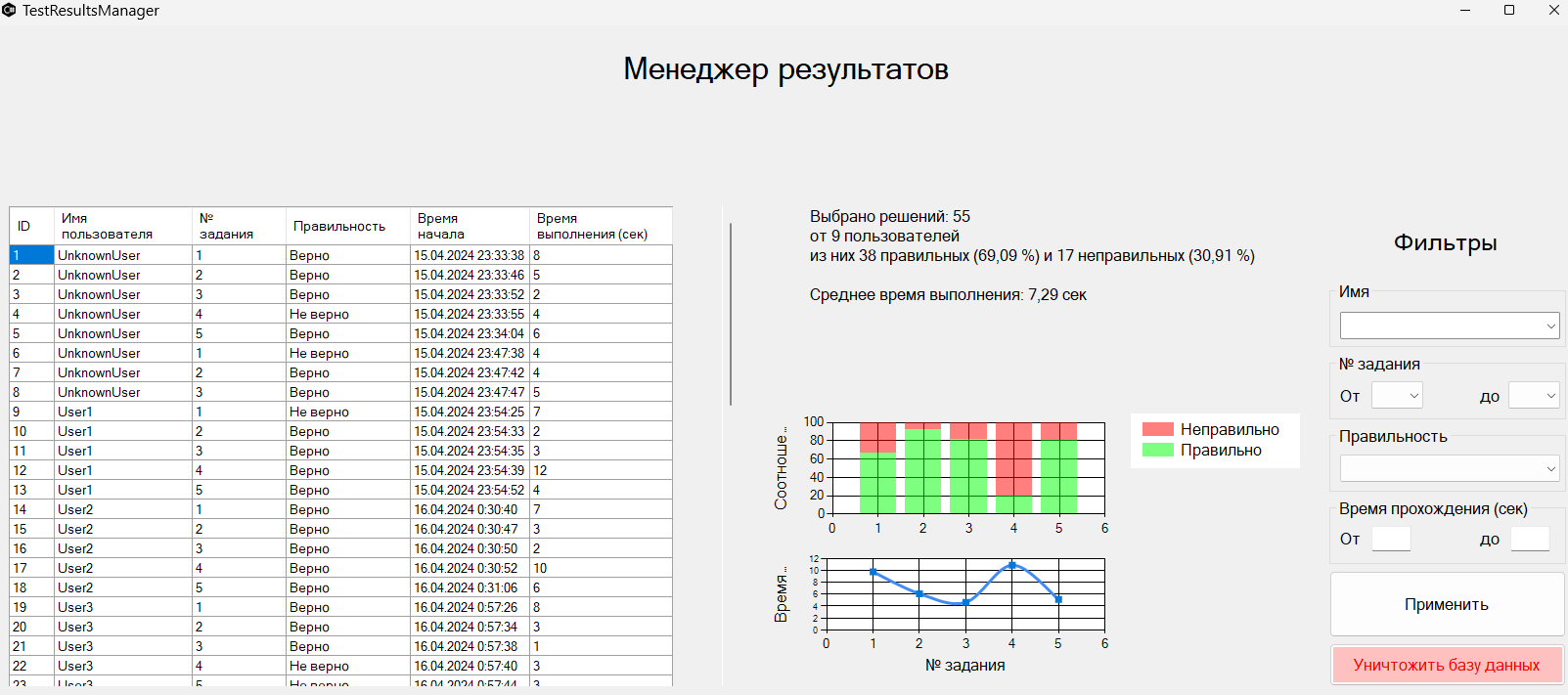


Рисунок 16 – Менеджер результатов

Также существует возможность применить фильтры к итогам тестирований. В качестве примера я получу статистику правильности выполнения третьего задания (задание с выбором правильных ответов из предложенных). Для этого в фильтр “№ задания” введём “От 3 до 3”, после чего нажмём кнопку “Применить”. В результате мы получаем данные о выполнении только задания №3, из которых было 81,82 % правильных ответов и 18,18 % неправильных (см. рисунок 17).

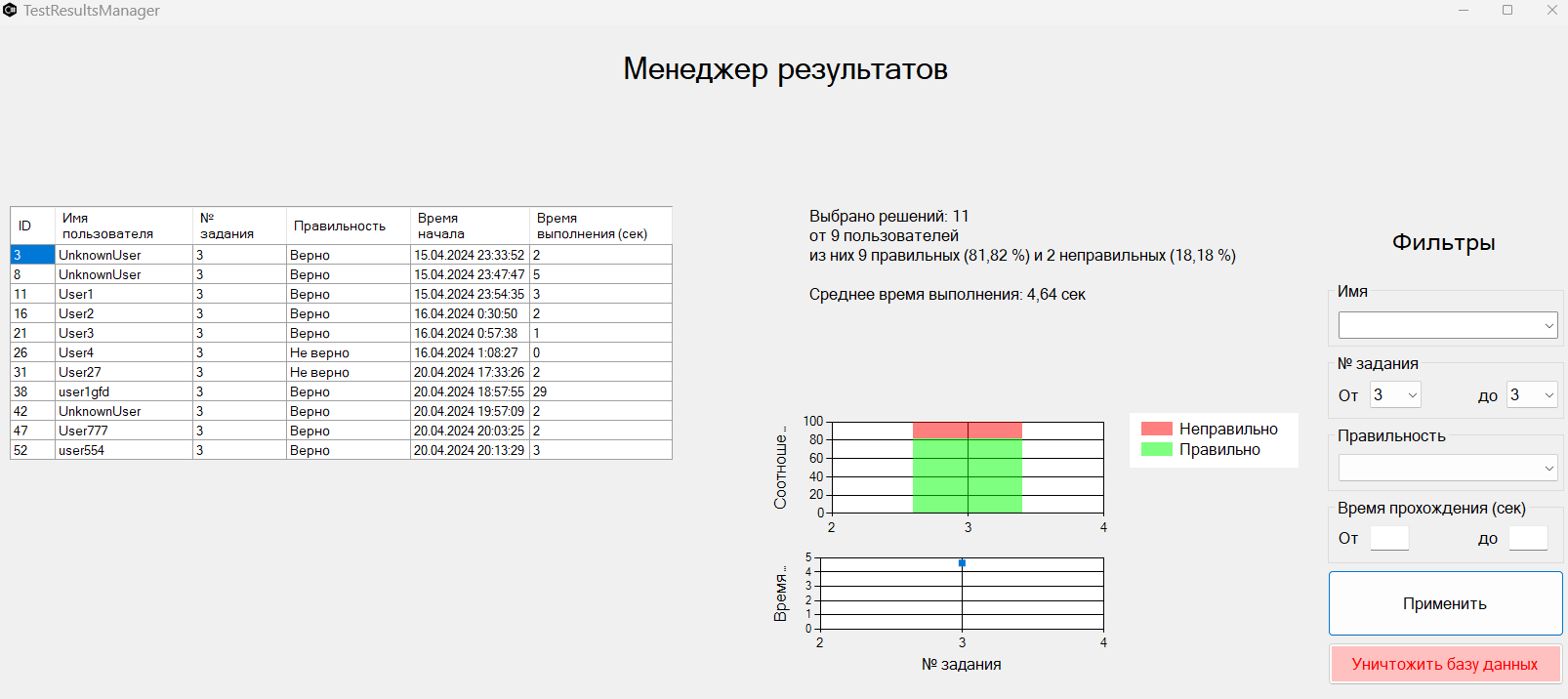


Рисунок 17 – Выбор задания №3

Программный код, реализующий данную форму, представлен ниже:

using MainDLL;

using Microsoft.Office.Interop.Excel;

using System;

using System.Data.SQLite;

using System.Windows.Forms;

namespace MainApplication

{

public partial class TestResultsManager : BaseForm

{

public TestResultsManager()

{

InitializeComponent();

SQLiteDataReader users = TestLabMainForm.database.Select("Results", "UserName", null, null, "GROUP BY UserName");

while (users.HasRows && users.Read())

{

nameCombo.Items.Add(users.GetString(0));

}

long totalTaskIndices = (long)TestLabMainForm.database.SelectScalar("MAX(TaskIndex)", "Results", null, null);

for (int i = 1; i < totalTaskIndices+2; i++)

{

taskFromCombo.Items.Add(Convert.ToString(i));

taskToCombo.Items.Add(Convert.ToString(i));

}

apply\_Click(null, null);

}

private void apply\_Click(object sender, EventArgs e)

{

string name = "%";

int taskFrom = 1, timeElapsedFrom = 1, isTrue = 1;

long taskTo = 9223372036854775807, timeElapsedTo = 9223372036854775807;

System.Drawing.Color[] colors = new System.Drawing.Color[] {

System.Drawing.Color.Coral,

System.Drawing.Color.Cyan,

System.Drawing.Color.Blue,

System.Drawing.Color.Crimson,

System.Drawing.Color.DarkRed};

// нормализация пользовательского ввода

if (nameCombo.Text != "")

name = nameCombo.Text;

if (taskFromCombo.Text != "")

taskFrom = Convert.ToInt32(taskFromCombo.Text);

if (taskToCombo.Text != "")

taskTo = Convert.ToInt32(taskToCombo.Text);

try

{

if (timeElapsedFromInput.Text != "")

timeElapsedFrom = Convert.ToInt32(timeElapsedFromInput.Text);

if (timeElapsedToInput.Text != "")

timeElapsedTo = Convert.ToInt32(timeElapsedToInput.Text);

}

catch (FormatException)

{

MessageBox.Show(caption: "Ошибка 0x00005", text: "Неверный ввод максимального/минимального времени", buttons:MessageBoxButtons.OK, icon: MessageBoxIcon.Error);

timeElapsedFromInput.Focus();

return;

}

switch (isTrueCombo.Text)

{

case "":

isTrue = -1;

break;

case "Любая":

isTrue = -1;

break;

case "Правильно":

isTrue = 1;

break;

case "Неправильно":

isTrue = 0;

break;

}

// очистка графиков и DataGridView

results.Rows.Clear();

foreach (System.Windows.Forms.DataVisualization.Charting.Series series in resultsChart.Series)

{

series.Points.Clear();

}

// создание SQL-запроса

string isTrueFilter = isTrue != -1 ? $"AND IsTrue ={isTrue}" : "";

string filter = $"WHERE UserName LIKE @Name AND (TaskIndex BETWEEN @TaskFrom AND @TaskTo) AND (TimeSpent BETWEEN @TimeElapsedFrom AND @TimeElapsedTo)" + isTrueFilter;

string[] keys = new string[] { "@Name", "@TaskFrom", "@TaskTo", "@TimeElapsedFrom", "@TimeElapsedTo" };

object[] values = new object[] { name, taskFrom - 1, taskTo-1, timeElapsedFrom - 1, timeElapsedTo-1 };

SQLiteDataReader reader = TestLabMainForm.database.Select(

"Results",

"id, UserName, TaskIndex, IsTrue, Date, TimeSpent",

keys, values, filter);

// получение значений для строки статистика

long totalUsers = (long)TestLabMainForm.database.SelectScalar("COUNT(\*)", $"(SELECT COUNT(\*) FROM Results {filter} GROUP BY UserName)", keys, values);

double avgTimeSpent = Math.Round((double)TestLabMainForm.database.SelectScalar("AVG(TimeSpent)", $"Results {filter}", keys, values), 2);

double trueAnswers = (long)TestLabMainForm.database.SelectScalar("COUNT(\*)", $"Results {filter} AND IsTrue=1", keys, values);

double totalAnswers = (long)TestLabMainForm.database.SelectScalar("COUNT(\*)", $"Results {filter}", keys, values);

statusLine.Text = $"Выбрано решений: {totalAnswers}\nот {totalUsers} пользователей\nиз них {trueAnswers} правильных ({Math.Round(trueAnswers / totalAnswers \* 100, 2)} %) и {totalAnswers - trueAnswers} неправильных ({Math.Round((totalAnswers-trueAnswers) / totalAnswers \* 100, 2)} %)" +

$"\n\nСреднее время выполнения: {avgTimeSpent} сек";

// заполнение DataGridView

while (reader.HasRows && reader.Read())

{

results.Rows.Add(

reader.GetInt64(0),

reader.GetString(1),

reader.GetInt32(2) + 1,

reader.GetBoolean(3) ? "Верно" : "Не верно",

reader.GetString(4),

reader.GetInt32(5));

}

// заполнение графика

long sumTrue, taskNumber;

SQLiteDataReader statisticReader = TestLabMainForm.database.Select("Results", "TaskIndex, AVG(TimeSpent), SUM(IsTrue), COUNT(\*)", keys, values, $" {filter} GROUP BY TaskIndex");

while (statisticReader.HasRows && statisticReader.Read())

{

taskNumber = statisticReader.GetInt64(0) + 1;

sumTrue = statisticReader.GetInt64(2);

resultsChart.Series["AvgTime"].Points.AddXY(taskNumber, statisticReader.GetDouble(1));

resultsChart.Series["Правильно"].Points.AddXY(taskNumber, sumTrue);

resultsChart.Series["Неправильно"].Points.AddXY(taskNumber, statisticReader.GetDouble(3)-sumTrue);

}

}

private void button1\_Click(object sender, EventArgs e)

{

DialogResult result = MessageBox.Show(caption: "Предупреждение", text: "Данное действие удалит все данные из таблицы результатов\nПродолжить",

buttons: MessageBoxButtons.YesNo, icon: MessageBoxIcon.Warning);

if (result == DialogResult.No)

return;

TestLabMainForm.database.Drop("Results");

TestLabMainForm.CreateResultsTable();

Close();

}

}

}

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Гуриков, С. Р. Введение в программирование на языке Visual C#: учебное пособие / С. Р. Гуриков. — Москва: ФОРУМ: ИНФРА-М, 2019. — 447 с. — (Среднее профессиональное образование). - ISBN 978-5-00091-540-0. - Текст: электронный. - URL: <https://znanium.com/catalog/product/1012397> (дата обращения: 16.04.2024).

2. ГОСТ 7.32-2017. “Межгосударственный стандарт. Система стандартов по информации, библиотечному и издательскому делу. Отчет о научно-исследовательской работе. Структура и правила оформления” (введен в действие Приказом Росстандарта от 24.10.2017 N 1494-ст) (дата обращений 16.04.2024).

3. ГОСТ 15.016-2016. Межгосударственный стандарт. Система разработки и постановки продукции на производство. Техническое задание. Требования к содержанию и оформлению (введен в действие Приказом Росстандарта от 01.09.2017 - МКС 01.100.01) (дата обращения 16.04.2024).

4. С. П. Мамай. Методика составления тестовых заданий: учебное пособие / С. П. Мамай. – Екатеринбург: Министерство образования Российской Федерации. Уральский государственный профессионально-педагогический университет “Уральское отделение Российской академии образования”, 2001 - Текст: электронный. - URL: [https://elar.rsvpu.ru/bitstream/123456789/20133/1/Mamay\_2001.pdf](https://elar.rsvpu.ru/bitstream/123456789/20133/1/Mamay_2001.pdf%20) (дата обращения: 16.04.2024).