ГОСУДАРСТВЕННОЕ ПРОФЕССИОНАЛЬНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
«ЛЕНИНСК-КУЗНЕЦКИЙ ПОЛИТЕХНИЧЕСКИЙ ТЕХНИКУМ»

УТВЕРЖДАЮ
Заместитель директора по
учебной работе
_____ Е.И. Будасова
«_____»_____

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ
ПО ВЫПОЛНЕНИЮ ПРАКТИЧЕСКИХ РАБОТ

МДК.05.03 Тестирование информационных систем

Специальность: 09.02.07

Информационные системы и программирование

Ленинск-Кузнецкий 2021

Методические рекомендации по выполнению практических работ по МДК.05.03 Тестирование информационных систем для специальности: 09.02.07 Информационные системы и программирование Государственное профессиональное образовательное учреждение «Ленинск-Кузнецкий политехнический техникум». – Ленинск-Кузнецкий, 2021. – 90 с.

Методические рекомендации рассмотрены на заседании цикловой методической комиссии преподавателей профессиональных дисциплин Государственного профессионального образовательного учреждения «Ленинск-Кузнецкий политехнический техникум» (протокол от «___»_____ № «___»).

Методические рекомендации рассмотрены и рекомендованы к использованию методическим советом Государственного профессионального образовательного учреждения «Ленинск-Кузнецкий политехнический техникум» (протокол от «___»_____ № «___»).

Составитель: Щеглова А.А., преподаватель.

Оглавление

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

Методические рекомендации по выполнению практических работ по МДК.05.03 Тестирование информационных систем разработаны на основании положений:

– рабочей программы учебной дисциплины МДК.05.03 Тестирование информационных систем

Согласно рабочей учебной программе МДК.05.03 Тестирование информационных систем предусмотрено 125 часов аудиторных занятий, на практические занятия отводится 54 аудиторных часа.

Практические работы позволят определить прочность и глубину усвоения материала по информационным технологиям, а также повторить пройденные темы и систематизировать знания обучающихся.

В результате освоения учебной дисциплины обучающийся должен:

Уметь:

1. осуществлять постановку задач по обработке информации
2. проводить анализ предметной области
3. осуществлять выбор модели и средства построения информационной системы и программных средств
4. использовать алгоритмы обработки информации для различных приложений
5. решать прикладные вопросы программирования и языка сценариев для создания программ
6. разрабатывать графический интерфейс приложения; создавать и управлять проектом по разработке приложения
7. проектировать и разрабатывать систему по заданным требованиям и спецификациям

Знать:

1. основные виды и процедуры обработки информации, модели и методы решения задач обработки информации
2. основные платформы для создания, исполнения и управления информационной системой
3. основные процессы управления проектом разработки
4. основные модели построения информационных систем, их структуру, особенности и области применения
5. методы и средства проектирования, разработки и тестирования информационных систем
6. систему стандартизации, сертификации и систему обеспечения качества продукции

Практические работы направлены на освоение практических умений и знаний согласно требованиям ФГОС СПО

Методические указания по проведению практических работ содержат краткую инструкцию, практические задания, требования к их оформлению и критерии оценивания практических работ.

Практические работы по МДК.05.03 Тестирование информационных систем проводятся в следующих формах:

- Тестирование требований
- Планирование тестирования
- Составление тест–кейсов
- Составление чек–листов
- Составление модульных тестов
- Разработка тестовых сценариев
- Разработка пакетов
- Анализ качества
- Обработка исключительных ситуаций
- Функциональное тестирование
- Тестирование безопасности
- Нагрузочное тестирование
- Тестирование интеграции
- Конфигурационное тестирование
- Тестирование установки

Обучающимся предлагаются практические работы разного уровня и разного содержания. Это позволяет обеспечить дифференцированный подход к организации выполнения практических работ обучающимися.

ПРАВИЛА ВЫПОЛНЕНИЯ ПРАКТИЧЕСКИХ РАБОТ

Работа должна быть выполнена в той же последовательности, в какой приведены вопросы практического занятия.

Каждый студент после выполнения работы должен представить отчет о проделанной работе. Отчет о проделанной работе следует делать в текстовом редакторе. Содержание отчета указано в описании практической работы.

Если студент не выполнил практическую работу или часть работы, то он может выполнить работу или оставшуюся часть во внеурочное время, согласованное с преподавателем.

Оценку по практической работе студент получает, с учетом срока выполнения работы, если:

- задания выполнены правильно, в полном объеме и в соответствии с требованиями
- сделан анализ задачи работы и вывод по результатам работы
- студент может пояснить выполнение любого этапа работы
- отчет выполняется в соответствии с требованиями к выполнению работы

ИСТРУКЦИЯ ПО ВЫПОЛНЕНИЮ ПРАКТИЧЕСКИХ РАБОТ
ДЛЯ ОБУЧАЮЩИХСЯ

1. Повторить теоретический материал, пройденный на аудиторных занятиях, изучить материал по теме учебника, дополнительной литературы, интернет – ресурсов.
2. Выполнить работу согласно заданию;
3. Ответить на поставленные вопросы;
4. По каждой практической работе представить преподавателю отчет.

Каждая работа оценивается по пятибалльной системе. Критерии оценки приведены в данных методических рекомендациях.

При возникновении затруднений в процессе работы, обратитесь за консультацией к преподавателю.

ПРАКТИЧЕСКАЯ РАБОТА № 1

Тема: Отладка и тестирование информационных систем

Цель: изучить критерии качества требований, описать требования, создать проект по указанным требованиям, выполнить тестирование спецификации требований

Перечень оснащения и оборудования, источников: ПК, раздаточный материал

Время на подготовку и выполнение: 2 часа

Задание: Тестирование требований

Порядок работы

1. Повторить теоретический материал
2. Выполнить задание
3. Оформить отчет в текстовом редакторе. Параметры шрифта: Times New Roman, 12 пт. Параметры абзаца междустрочный интервал одинарный, текст выравнивается по ширине
4. Отчет должен содержать:

– Название проекта
– Предназначение проекта
– Описание требований к пользовательскому интерфейсу
– Описание функциональных требований
– Описание нефункциональных требований
– Анализ и тестирование требований
– Вывод

Задание. Провести анализ и тестирование требований

Порядок выполнения работы

1. Составить три таблицы: требования к интерфейсу, функциональные и нефункциональные требования к приложению (Windows Form), соответствующему Вашему варианту.

№	Название	Требования

Вариант 1

Вычисление площади треугольника, если известна длина основания и высота.

$$S = \frac{1}{2}ab$$

Вариант 2

Вычисление площади треугольника, если известны длины двух его сторон и величина угла между этими сторонами.

$$S = \frac{1}{2}absin\alpha$$

Вариант 3

Вычисление сопротивления электрической цепи, состоящей из двух параллельно соединенных сопротивлений.

$$r = \frac{r_1 r_2}{r_1 + r_2}$$

Вариант 4

Вычисление сопротивления электрической цепи, состоящей из двух последовательно соединенных сопротивлений.

$$r = r_1 + r_2$$

Вариант 5

Вычисление силы тока в электрической цепи.

$$I = \frac{U}{R}$$

U – напряжение, R - сопротивление

Вариант 6

Вычисление пересчета веса фунтов в кг. 1 фунт = 405,9 г

Вариант 7

Вычисление стоимости поездки на автомобиле на дачу (туда и обратно).

$$2\left(\frac{\text{Потребление бензина на 100 км. пути}}{100} \cdot \text{Расстояние до дачи} \cdot \text{Цена одного литра бензина}\right)$$

Вариант 8

Вычисление площади трапеции, если известны длины оснований и высота.

$$S = \frac{a + b}{2} h$$

Вариант 9

Вычисление объема цилиндра, если известны радиус основания и высота.

$$V = \pi R^2 h$$

Вариант 10

Вычисление площади поверхности цилиндра, если известны радиус основания и высота.

$$S_{пп} = S_{бп} + 2S_{осн}, \;\; S_{осн} = \pi R^2, \;\; S_{бп} = 2\pi Rh$$

Вариант 11

Вычисление объема параллелепипеда по трем его измерениям.

$$V = abc$$

Вариант 12

Вычисление пересчета расстояния из верст в километры. 1 верста = 1066,8 м.

2. Обменяться полученными требованиями с другим вариантом. По полученным требованиям составить приложение. В таблице с требованиями добавить и заполнить столбец анализ требований

3. Обменяться полученными требованиями и приложением с другим вариантом. Протестировать соответствие требований приложению. В таблице с требованиями добавить и заполнить столбец тестирование требований

Контрольные вопросы:

1. Как выглядит жизненный цикл проекта?
2. Какие выделяют критерии качества?
3. Какие требования считаются проверяемыми?
4. Какие требования считаются модифицируемыми?
5. Какие требования считаются корректными?
6. Какие требования считаются недвусмысленными?
7. Какие требования считаются полными?
8. Какие требования считаются непротиворечивыми?
9. Какие требования считаются упорядоченными по важности и стабильности?
10. Какие требования считаются трассируемыми?
11. Какие существуют методы тестирования требований?
12. Какие этапы включает в себя метод просмотра при тестировании требований?

Критерии оценки

оценка «5» ставится, если:

– работа выполнена полностью;

– создано приложение, в требованиях и тестировании нет ошибок;

– возможна одна (две) неточность, не являющаяся следствием незнания или не понимания материала.

оценка «4» ставится, если:

– правильно выполнена большая часть работы (свыше 85%), допущено не более трех ошибок;

– работа выполнена полностью, но обоснования шагов решения недостаточны.

оценка «3» ставится, если:

– работа выполнена не полностью, допущено более трех ошибок, но обучающийся владеет основными знаниями, умениями по проверяемой дисциплине.

оценка «2» ставится, если:

– допущены существенные ошибки, показывающие, что обучающийся не владеет обязательными знаниями, умениями по данной дисциплине

ПРАКТИЧЕСКАЯ РАБОТА № 2

Тема: Отладка и тестирование информационных систем

Цель: изучение структуры плана тестирования и получение практических навыков разработки плана тестирования

Перечень оснащения и оборудования, источников: ПК, раздаточный материал

Время на подготовку и выполнение: 2 часа

Текст задания. Планирование тестирования

Порядок выполнения работы

1. Повторить теоретический материал
2. Выполнить задание
3. Оформить отчет в текстовом редакторе. Параметры шрифта: Times New Roman, 12 пт. Параметры абзаца междустрочный интервал одинарный. Текст выравнивается по ширине
4. Отчет должен содержать тестовый план.

Составить план тестирования для проекта, созданного Вами на практической работе № 1.

План тестирования содержит

1. Идентификатор
2. Введение
3. Объекты тестирования
4. Что будет тестироваться?
5. Что не будет тестироваться?
6. Подход к тестированию
7. Критерии прохождения тестов
8. Критерии приостановки и возобновления работ
9. Результаты проведения тестирования
10. Задачи для проведения тестирования
11. Технические требования
12. Необходимые компетенции и тренинги
13. Расписание (срок сдачи)
14. Риски и их устранение. Заполнить таблицу

Риск	Способ устранения	Последствия, связанные с не устранением риска

Контрольные вопросы:

1. Перечислите основные риски при разработке программного обеспечения.
2. Перечислите общие методы оценки рисков.
3. Что такое документ «План тестирования»?
4. Назовите основные разделы планирования тестирования.
5. С какой целью составляется план тестирования?

Критерии оценки

оценка «5» ставится, если:

– работа выполнена полностью;

– возможна одна (две) неточность, не являющаяся следствием незнания или не понимания материала.

оценка «4» ставится, если:

– правильно выполнена большая часть работы (свыше 85%), допущено не более трех ошибок;

– работа выполнена полностью, но обоснования шагов решения недостаточны.

оценка «3» ставится, если:

– работа выполнена не полностью, допущено более трех ошибок, но обучающийся владеет основными знаниями, умениями по проверяемой дисциплине.

оценка «2» ставится, если:

– допущены существенные ошибки, показывающие, что обучающийся не владеет обязательными знаниями, умениями по данной дисциплине

ПРАКТИЧЕСКАЯ РАБОТА № 3

Тема: Отладка и тестирование информационных систем

Цель: приобрести практические навыки создания тест кейсов

Перечень оснащения и оборудования, источников: ПК, раздаточный материал

Время на подготовку и выполнение: 2 часа

Задание. Составить тест кейсы

Порядок выполнения работы

1. Повторить теоретический материал
2. Выполнить задание
3. Оформить отчет в текстовом редакторе. Параметры шрифта: Times New Roman, 12 пт. Параметры абзаца междустрочный интервал одинарный. Текст выравнивается по ширине
4. Отчет должен содержать

– Чек-листы

– Тест-кейсы

– Скрины с тестированием приложения

По созданным в практической работе № 1 требованиям составить тест кейсы и чек-листы

Контрольные вопросы

1. В какой последовательности рекомендуется разрабатывать тесты?

2. Что такое чек-лист?

3. Перечислите элементы тест-кейса.

4. Обязательно ли описывать ожидаемый результат в тест-кейсе и в чек-листе?

5. Чем тест-кейс отличается от чек-листа?

Критерии оценки

оценка «5» ставится, если:

– работа выполнена полностью;

– возможна одна (две) неточность, не являющаяся следствием незнания или не понимания материала.

оценка «4» ставится, если:

– правильно выполнена большая часть работы (свыше 85%), допущено не более трех ошибок;

– работа выполнена полностью, но обоснования шагов решения недостаточны.

оценка «3» ставится, если:

– работа выполнена не полностью, допущено более трех ошибок, но обучающийся владеет основными знаниями, умениями по проверяемой дисциплине.

оценка «2» ставится, если:

– допущены существенные ошибки, показывающие, что обучающийся не владеет обязательными знаниями, умениями по данной дисциплине

ПРАКТИЧЕСКАЯ РАБОТА № 4

Тема: Отладка и тестирование информационных систем

Цель: Изучение правил создания модульных тестов для библиотеки классов

Перечень оснащения и оборудования, источников: ПК, раздаточный материал

Время на подготовку и выполнение: 2 часа

Текст задания. Модульное тестирование

Порядок выполнения работы

1. Повторить теоретический материал

2. Выполнить задание

3. Оформить отчет в текстовом редакторе. Параметры шрифта: Times New Roman, 12 пт. Параметры абзаца междустрочный интервал одинарный. Текст выравнивается по ширине

4. Отчет должен содержать таблицу

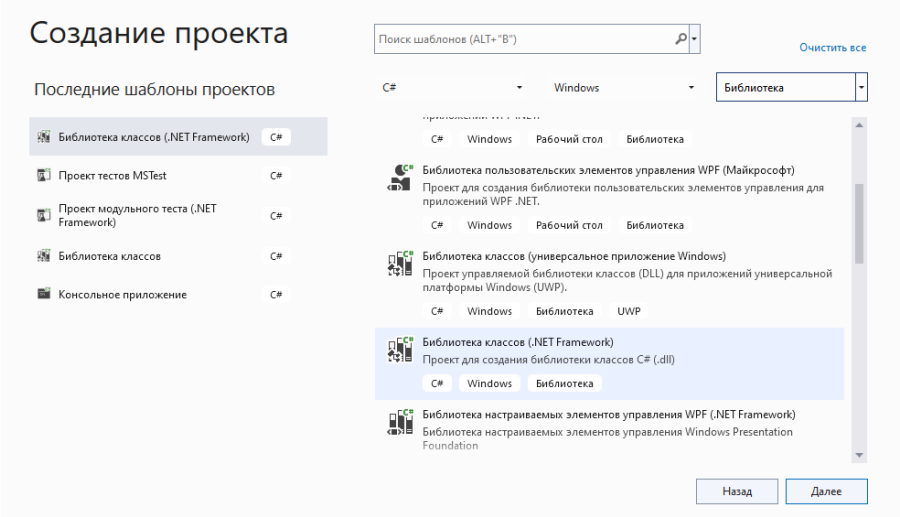
Входящие данные	Ожидаемый результат	Фактический результат	Вывод

5. Сдать отчет вместе с проектом, который включает в себя тестируемое приложение и модульный тест

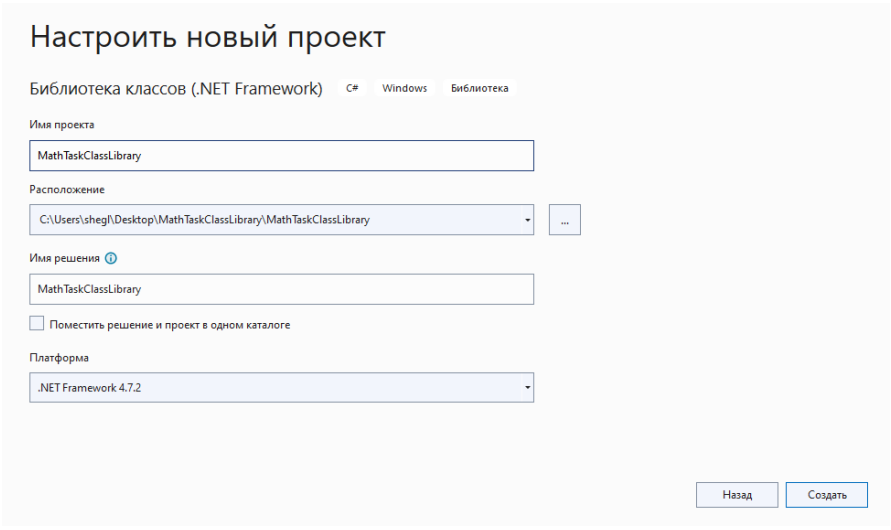
№ 1. Создать модульный тест для библиотеки классов, содержащей метод вычисления площади прямоугольника по длине двух его сторон.

Порядок действий

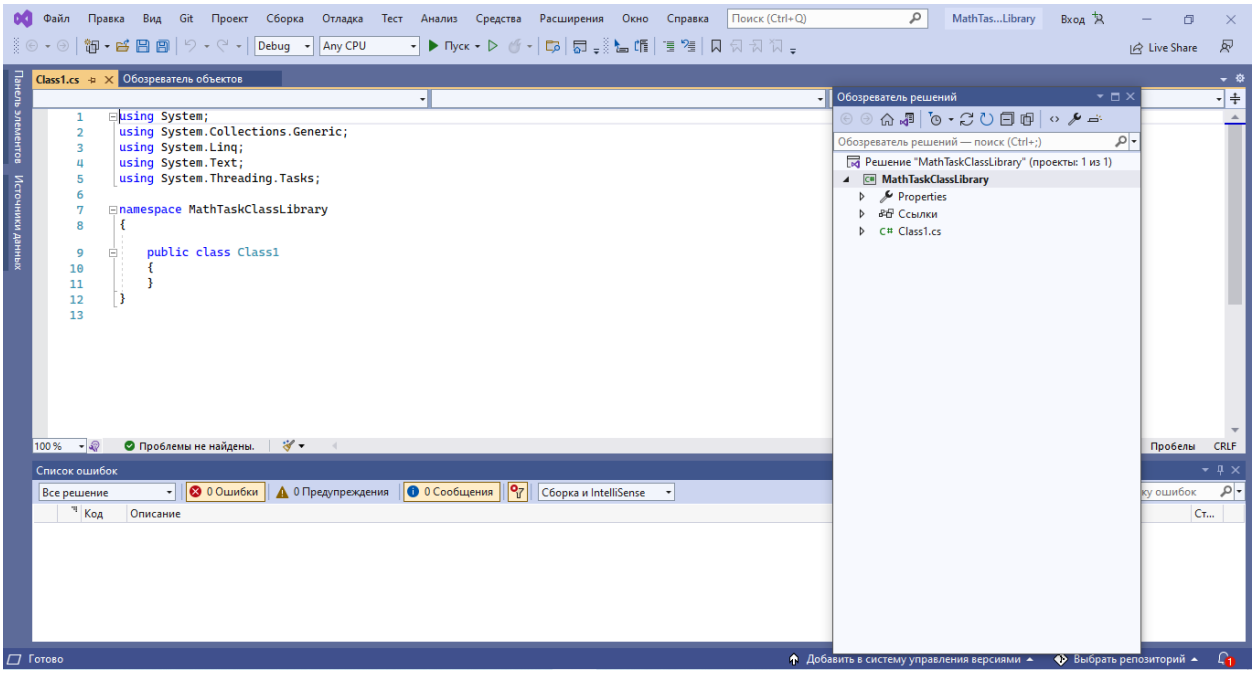
1. Создать в Visual Studio новый проект. Выбрать следующие значения: C#, Windows, Библиотека, Библиотека классов (NetFramework). Нажать кнопку Далее



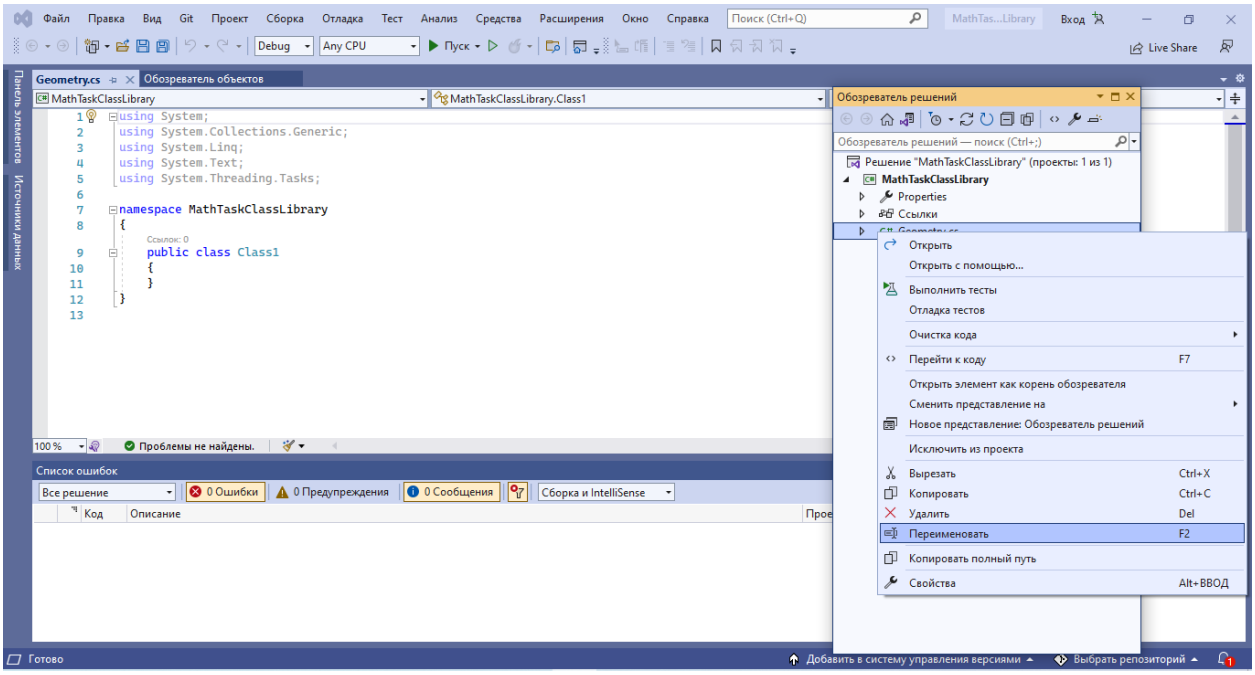
2. Имя проекта, должно отражать суть приложения, MathTaskClassLibrary. Нажать кнопку Создать



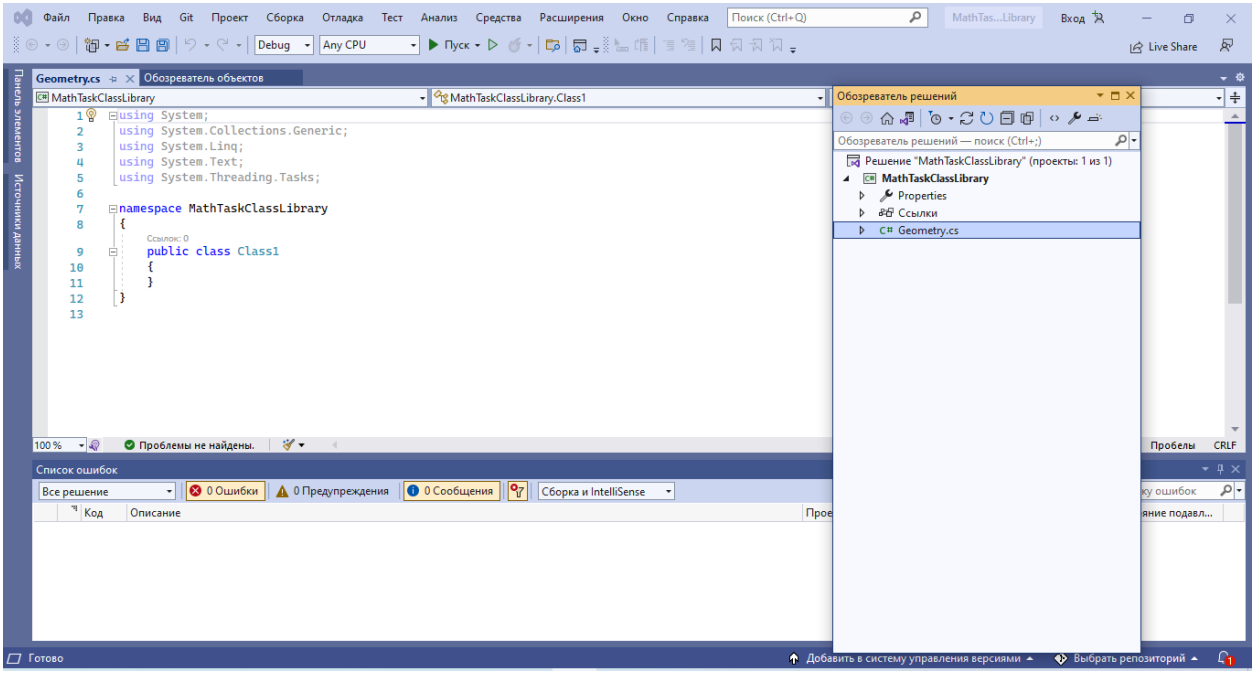
3. В результате откроется окно проекта



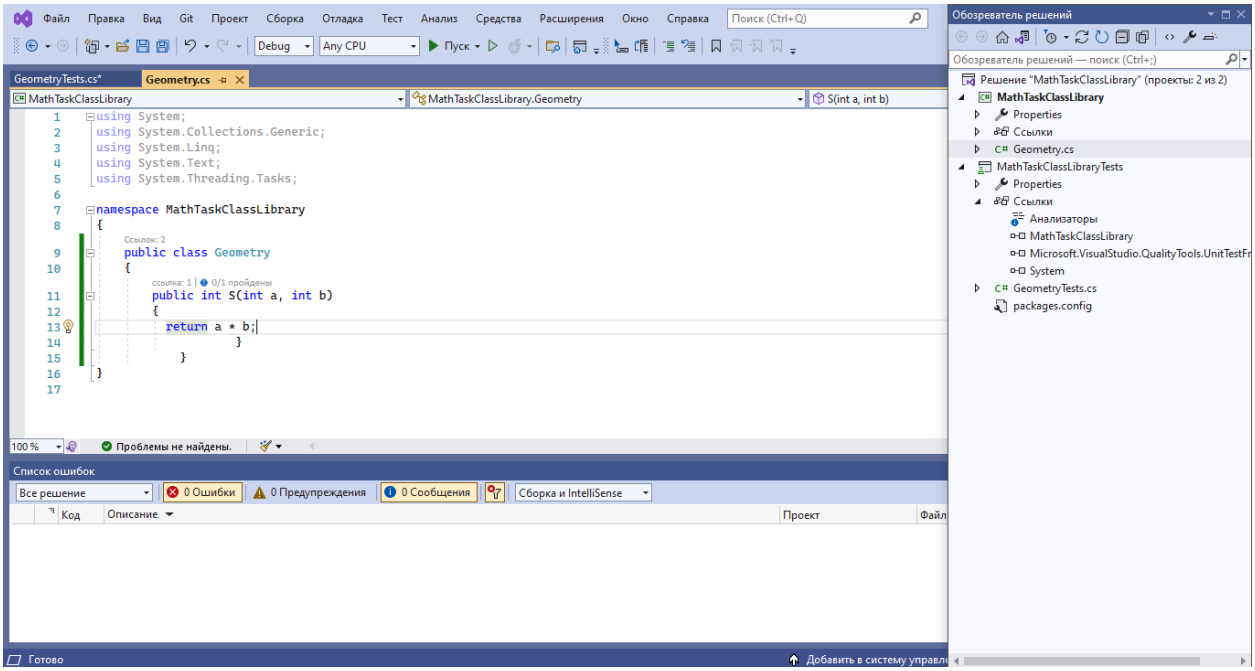
4. В Обозревателе решений Class1.cs переименовать в Geometry.cs. Для этого в Обозревателе решения нажать на имя класса Class1 и в Контекстном меню выбрать Переименовать



5. Заменить выделенный текст Class1 на Geometry

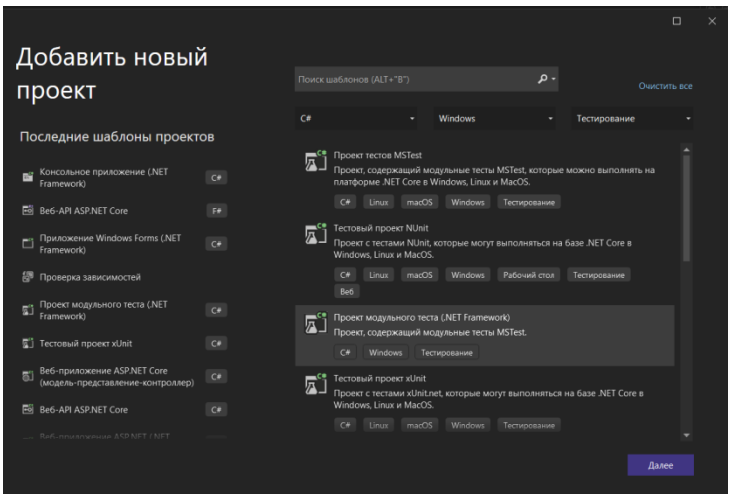


6. В классе Geometry.cs реализовать метод, вычисляющий площадь прямоугольника, записать код программы:

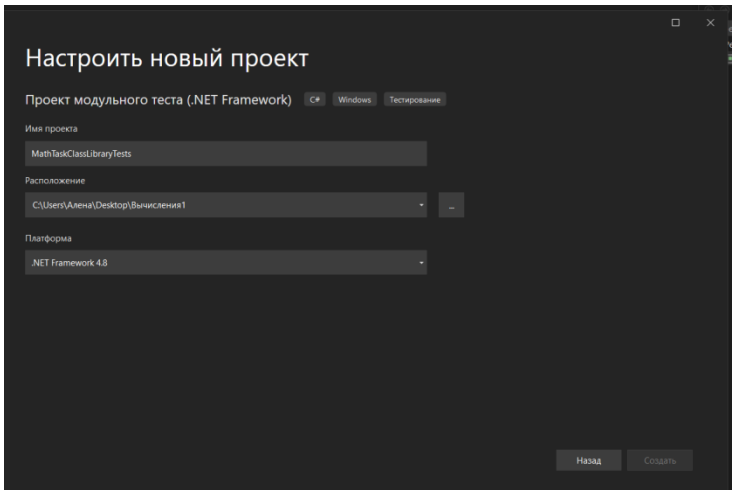


7. Для того, чтобы выполнить unit-тестирование, необходимо в рамках того же самого решения создать ещё один проект соответствующего типа. Для этого выполнить команду Файл/Добавить/Создать проект

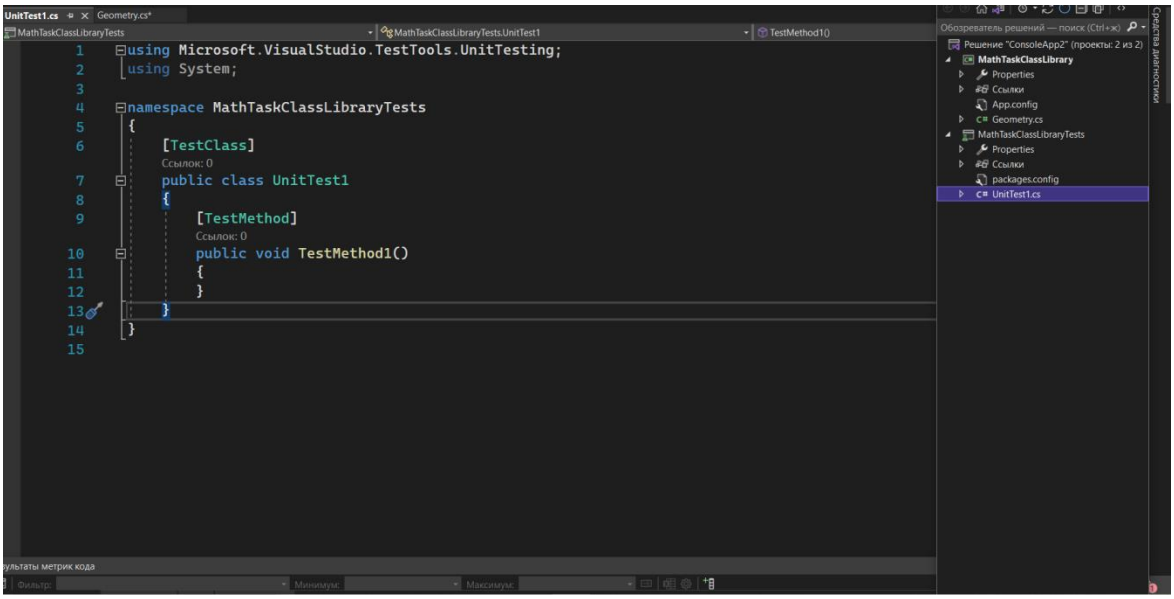
8. Выбрать Тестирование, C#, Windows, Проект модульного теста (Net Framework). Нажать кнопку Далее



9. Ввести имя проекта MathTaskClassLibraryTests



10. В результате откроется окно с кодом модульного теста UnitTest1.cs:

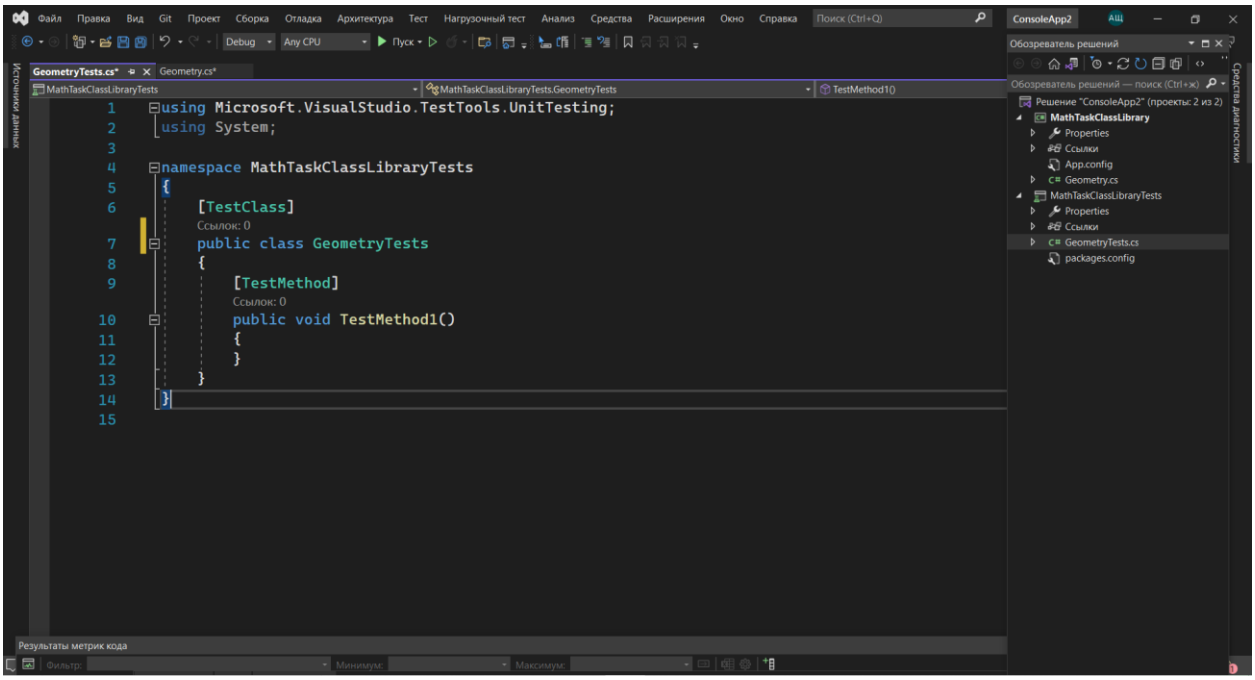


Директива — специальная команда, указывающая компилятору на особенности обработки кода при компиляции.

– Директива [TestMethod] обозначает, что далее идёт метод, содержащий модульный тест.

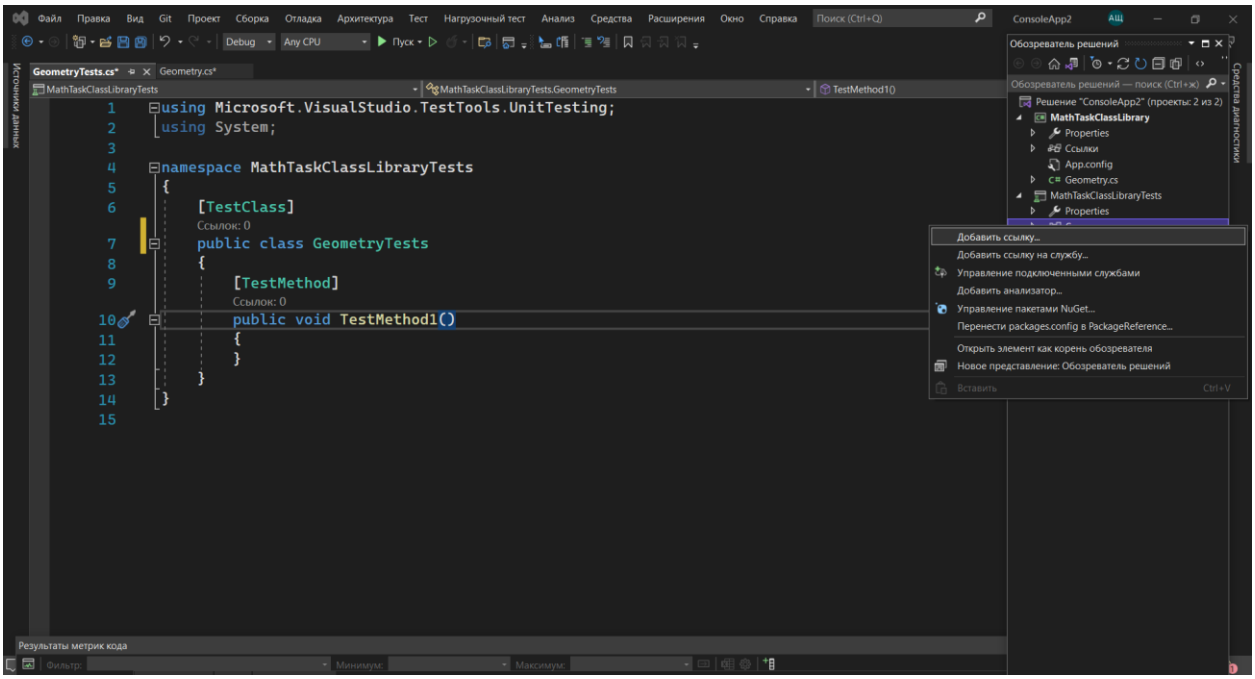
– Директива [TestClass] обозначает, что далее идёт класс, содержащий методы, в которых присутствуют модульные-тесты.

11. В Обозревателе решений переименовать класс UnitTest1 в GeometryTests

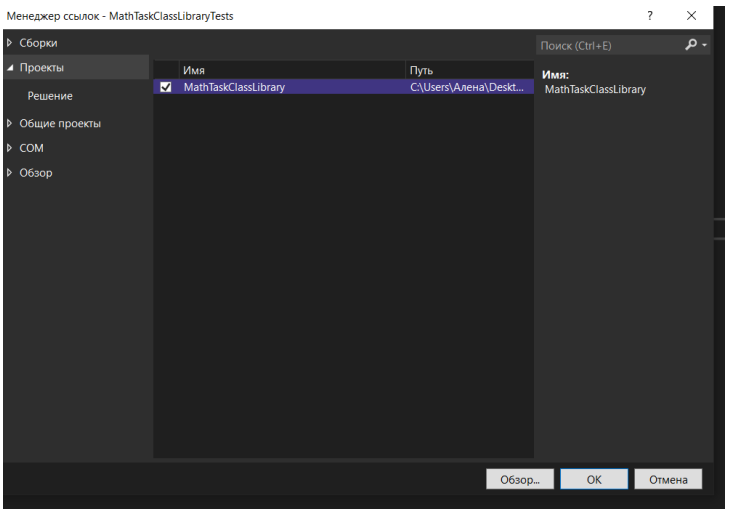


Во всплывающем окне выбрать обновить все ссылки

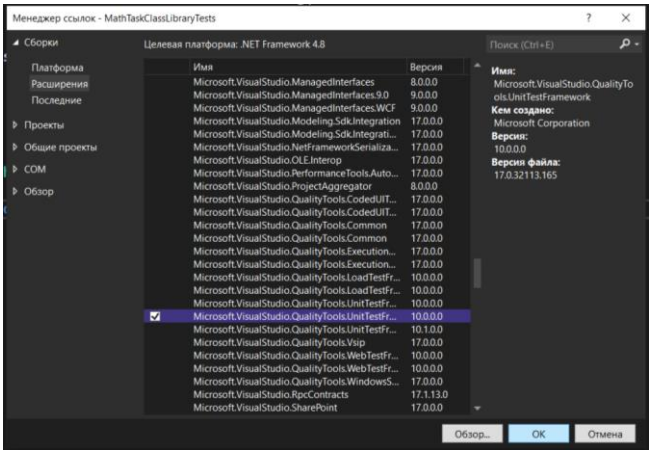
12. В Ссылках проекта MathTaskClassLibraryTests в Обозревателе решений необходимо добавить ссылку на проект, тестируемого кода MathTaskClassLibrary. Правой кнопкой нажать на Ссылки, в контекстном меню выбрать “Добавить ссылку…”



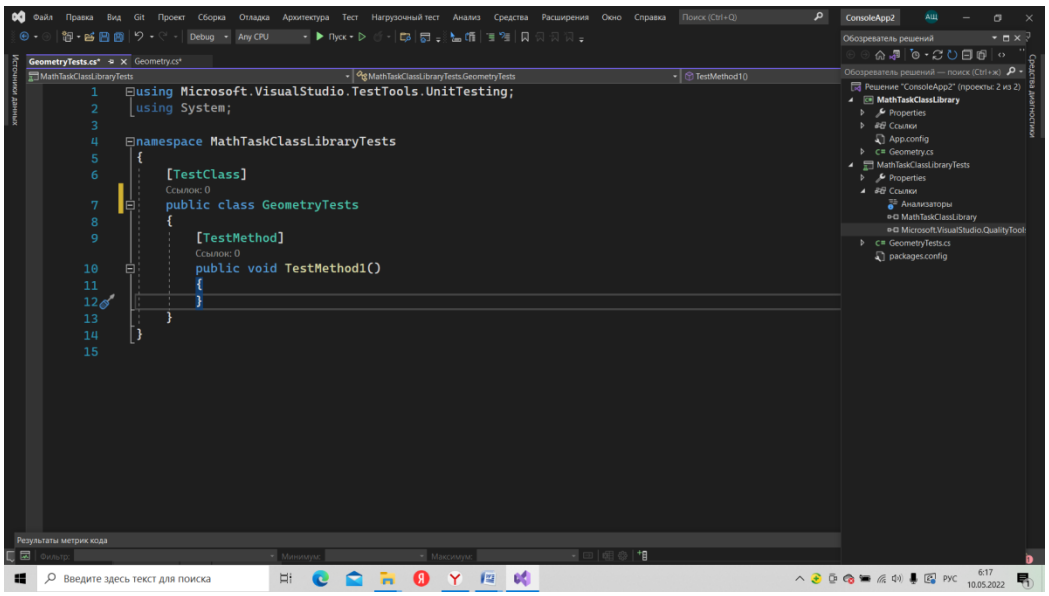
В появившемся окне раскрыть группу Решение, выбрать Проекты и поставить флажок напротив проекта MathTaskClassLibrary



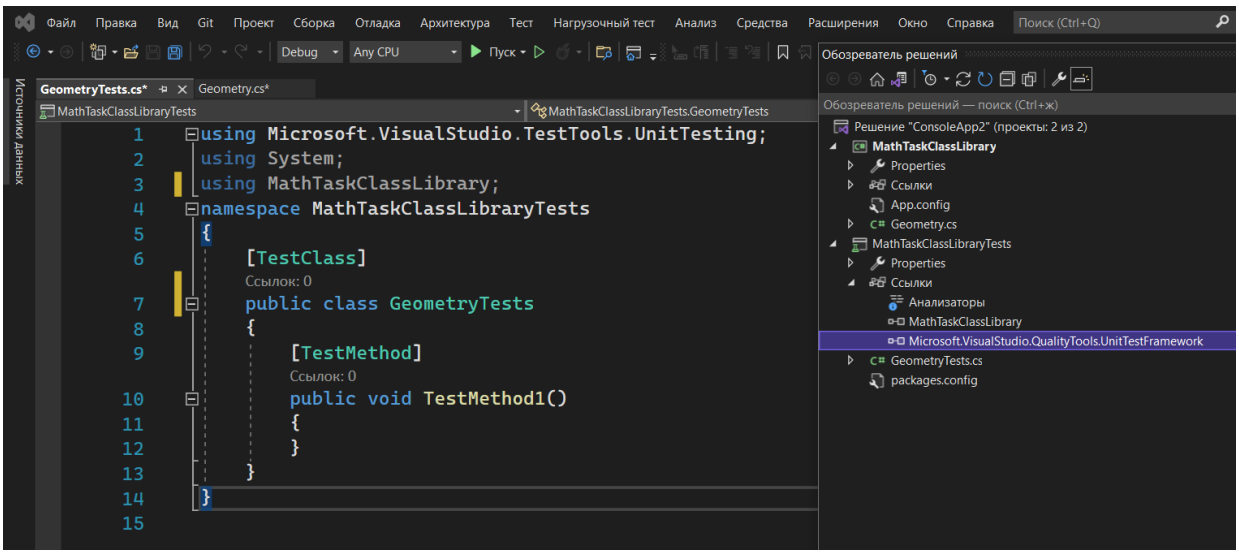
В группе Сборка выбрать Расширения и Microsoft.VisualStudio. Quality Tools.UnitTestFramrwork, версия 10.0.0.0



В результате в ссылках отразится только, то, что выбрали



13. В коде необходимо подключить с помощью директивы using имя тестируемого проекта: using MathTaskClassLibrary;



14. Проверить правильно ли программа вычисляет площадь прямоугольника со сторонами 3 и 5. Ожидаемый результат в данном случае 15, т.к. $3 \cdot 5 = 15$.

Переименуем метод TestMethod1() в RectangleArea_3and5_15returned(). Новое название метода поясняет, что будет проверяться (RectangleArea – площадь прямоугольника) для каких значений (3 и 5) и что ожидается в качестве правильного результата (15 returned).

Тестирующий метод содержит три необходимых компонента:

1. Исходные данные: входные значения и ожидаемый результат:

```
int a = 3; // исходные данные
int b = 5; // исходные данные
int expected = 15; // ожидаемый результат
```

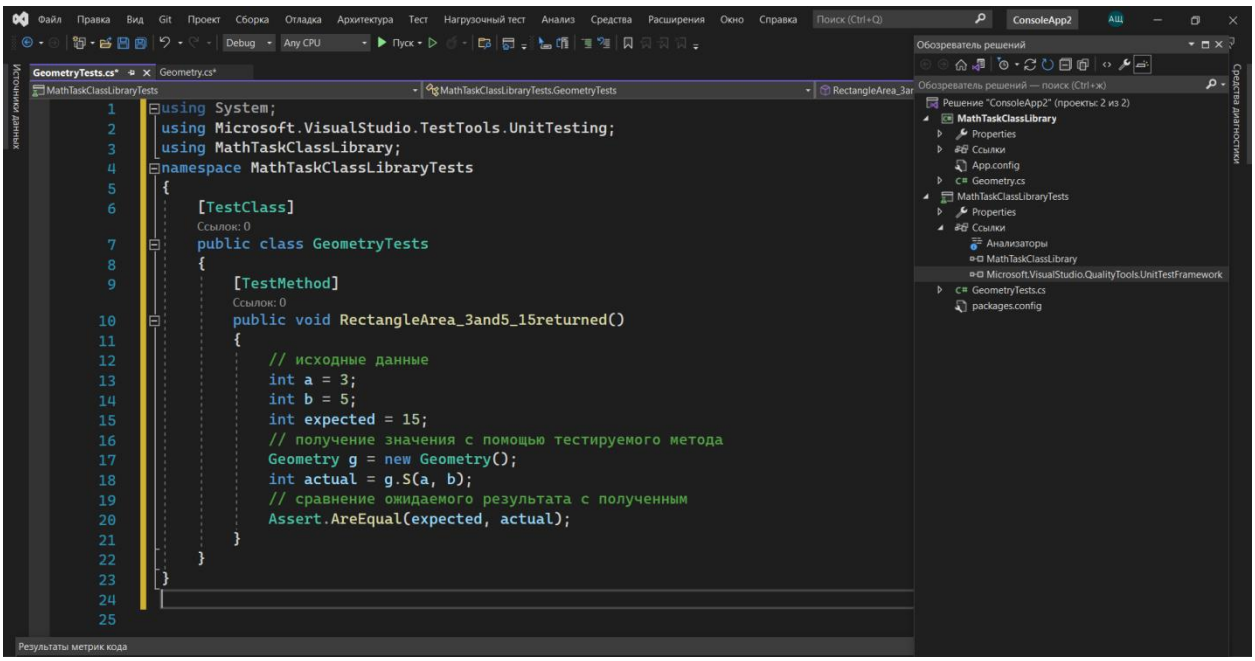
2. Код, вычисляющий значение с помощью тестируемого метода:

```
Geometry g = new Geometry();
int actual = g.S (a, b); // получение значения с помощью тестируемого метода
```

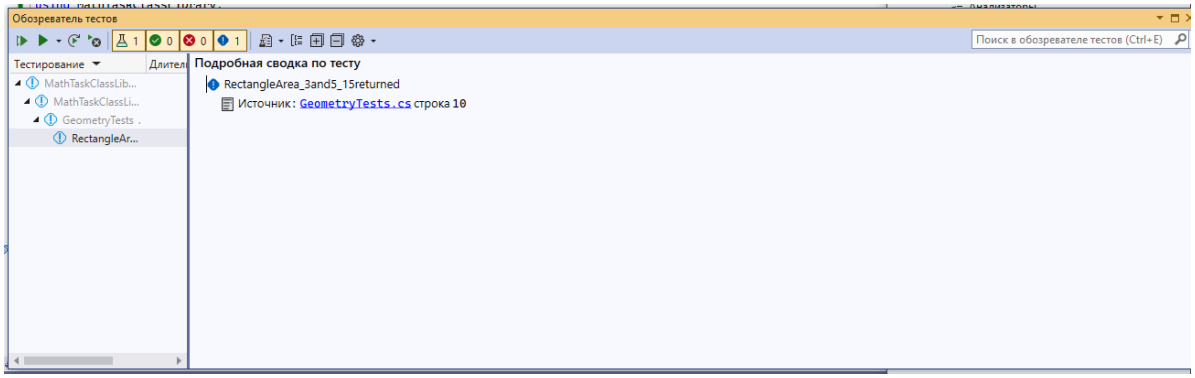
3. Код, сравнивающий ожидаемый результат с полученным.

```
Assert.AreEqual(expected, actual); // сравнение ожидаемого результата с полученным
```

Т.е.



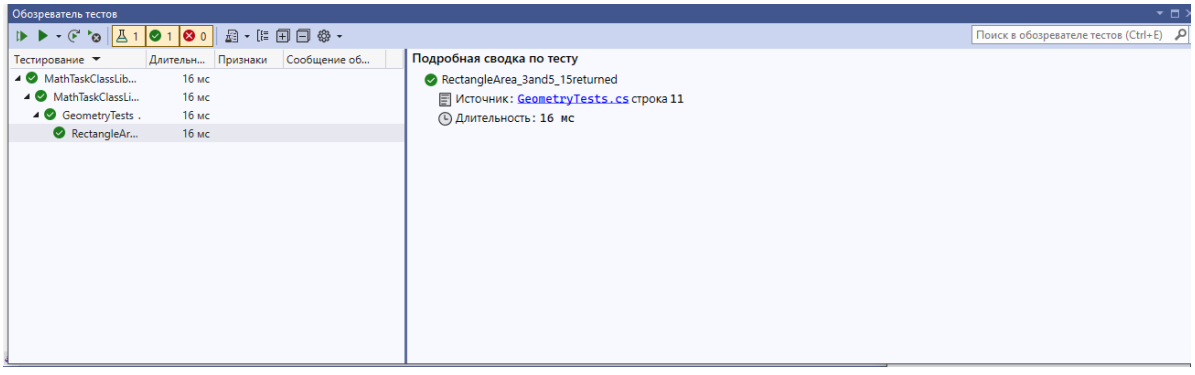
15. Выполнить сборку нажатием клавиш Ctrl + Shift + B или командой Сборка/Собрать решение. После её завершения запустить Обозреватель тестов, для этого выполнить команду Тест/ Обозреватель тестов. В результате откроется окно Обозреватель тестов



16. Нажать на кнопку Выполнено

1. В окне отобразится название теста MathTaskClassLibraryTests

2. Нажать правой кнопкой мыши на названии теста MathTaskClassLibraryTests и в контекстном меню выбрать Запустить.



Тестирование прошло успешно.

17. Создать ошибочную ситуацию. Для этого открыть Geometry.cs. Исправить команду:

return a * b;

на

return a * b+10;

18. В модульном тесте оставить все без изменений

int a = 3; // исходные данные

int b = 5; // исходные данные

int expected = 15; // ожидаемый результат

Т.к. в Geometry.cs была изменена формула $a \cdot b + 10$, то фактический результат, который будет получен при обращении к коду программы при тестировании методом

Geometry g = new Geometry();

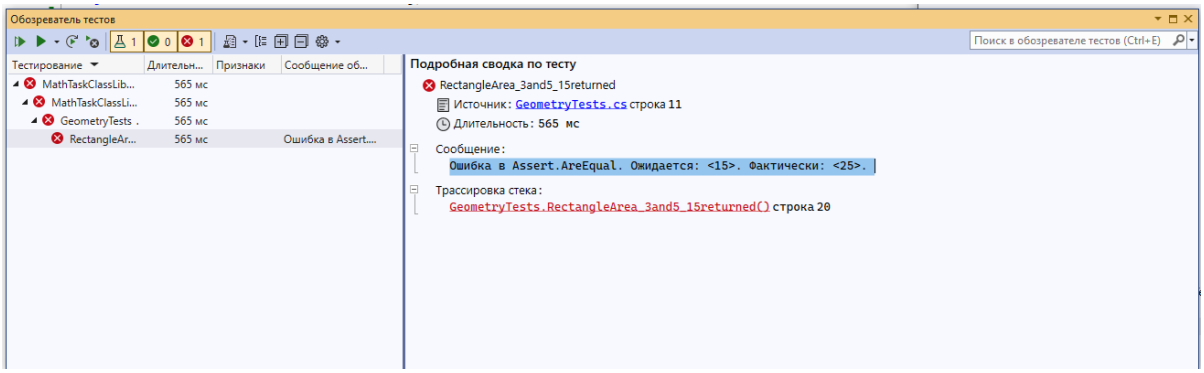
int actual = g.S (a, b); // фактический результат

будет равен 25, т.к. $3 \cdot 5 + 10 = 25$

Поэтому, запуск тестирования должен указать на данную не состыковку.

Выполнить сборку. Сборка/

Повторно выполнить Тест/Обозреватель теста. Запустить тест.



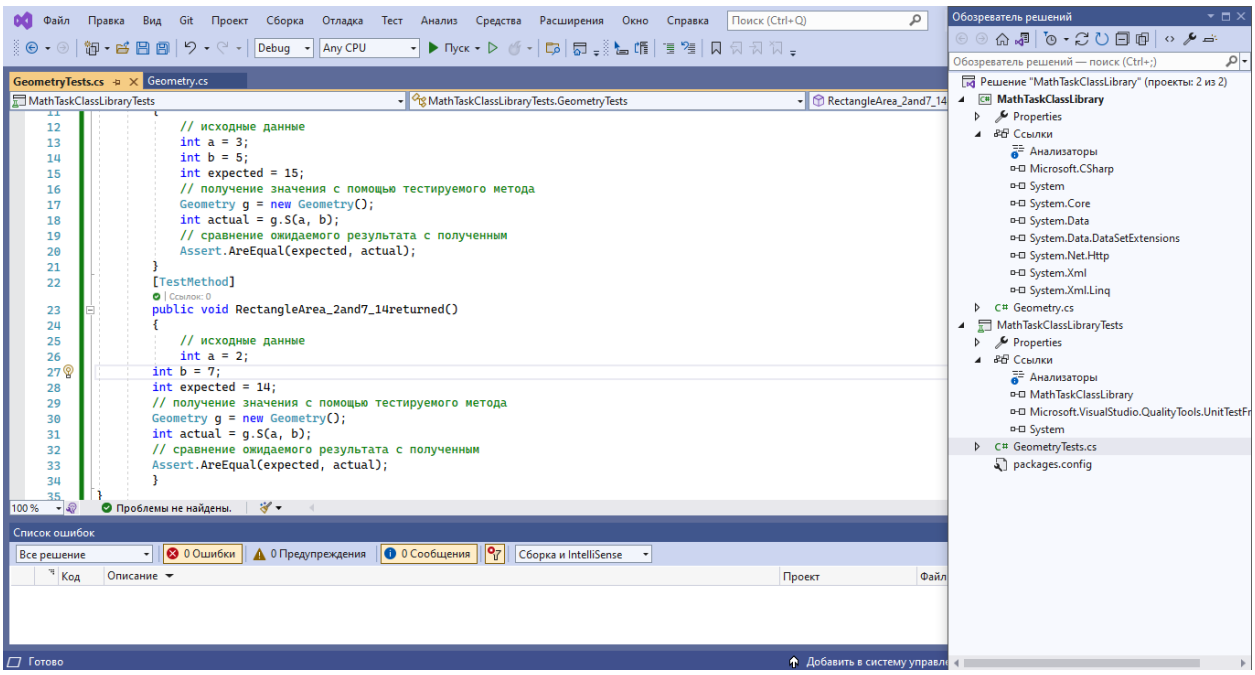
В результате тест выдает ошибку

Ошибка в Assert.AreEqual. Ожидается: <15>. Фактически: <25>.

19. Исправить в программном коде в Geometry.cs формулу a * b+10 на a * b.

20. В модульном тесте проверить другие данные:

Добавить тестовый метод метод в RectangleArea_2and7_14returned(), т.е. проверить правильно ли вычисляет программа площадь прямоугольника со сторонами 2 и 7. Ожидаемый результат в данном случае 14 (2*7=14).



21. Запустить тест и проверить результат.

22. Аналогично добавить пять тестовых методов (два верных и три неверных), проверяющих правильность выполнения программы. По полученным результатам заполнить таблицу

Исходные данные		Ожидаемый результат	Фактический результат	Вывод	Комментарий
a	b				
3	5	15	15	выполнен	
3	5	15	25	не выполнен	изменения в формуле тестируемого метода
2	7	14	14	выполнен	
1	3	4	3	не выполнен	

№ 2. Добавить класс, вычисляющий периметр прямоугольника, к этому классу написать модульный тест, содержащий не менее десяти тестовых методов (пять верных и пять неверных), проверяющих правильность выполнения программы. По полученным результатам заполнить таблицу

Исходные данные		Ожидаемый результат	Фактический результат	Вывод
a	b			

№ 3. Добавить класс, вычисляющий объем цилиндра, к этому классу написать модульный тест, содержащий не менее десяти тестовых методов (пять верных и пять неверных),

проверяющих правильность выполнения программы. По полученным результатам заполнить таблицу

Исходные данные		Ожидаемый результат	Фактический результат	Вывод
R	h			

№ 4. Добавить класс для подсчета объема и площадь параллелепипеда, к этому классу написать модульный тест, содержащий не менее десяти тестовых методов (пять верных и пять неверных), проверяющих правильность выполнения программы. По полученным результатам заполнить таблицу

Объем и площадь параллелепипеда V = abc. S = 2 (ab + bc + ac)

Исходные данные			Ожидаемый результат	Фактический результат	Вывод
a	b	c			

Контрольные вопросы:

1. Какой тест называется модульным?
2. С какой целью используются модульные тесты?
3. Назовите три стадии в модульном тесте

Критерии оценки

оценка «5» ставится, если:

– работа выполнена полностью;

– возможна одна (две) неточность, не являющаяся следствием незнания или не понимания материала.

оценка «4» ставится, если:

– правильно выполнена большая часть работы (свыше 85%), допущено не более трех ошибок;

– работа выполнена полностью, но обоснования шагов решения недостаточны.

оценка «3» ставится, если:

– работа выполнена не полностью, допущено более трех ошибок, но обучающийся владеет основными знаниями, умениями по проверяемой дисциплине.

оценка «2» ставится, если:

– допущены существенные ошибки, показывающие, что обучающийся не владеет обязательными знаниями, умениями по данной дисциплине

ПРАКТИЧЕСКАЯ РАБОТА № 5

Тема: Отладка и тестирование информационных систем

Цель: Изучение правил создания модульных тестов для консольных приложений

Перечень оснащения и оборудования, источников: ПК, раздаточный материал

Время на подготовку и выполнение: 2 часа

Задание: Модульное тестирование

Порядок работы

1. Повторить теоретический материал

2. Выполнить задание

3. Оформить отчет в текстовом редакторе. Параметры шрифта: Times New Roman, 12 пт. Параметры абзаца междустрочный интервал одинарный. Текст выравнивается по ширине

4. Отчет должен содержать таблицу

Входящие данные	Ожидаемый результат	Фактический результат	Вывод

5. Сдать отчет вместе с проектом, который включает в себя тестируемое приложение и модульный тест

№ 1. Протестировать консольное приложение, вычисляющее минимальное значение двух чисел. Модульный тест, содержит не менее десяти тестовых методов (пять верных и пять неверных), проверяющих правильность выполнения программы. По полученным результатам заполнить таблицу

Исходные данные		Ожидаемый результат	Фактический результат	Вывод
a	b			
3	5	7	8	не выполнен
3	5	7	7	выполнен

№ 2. Протестировать консольное приложение, вычисляющее максимальное значение трех чисел. Модульный тест, содержит не менее десяти тестовых методов (пять верных и пять неверных), проверяющих правильность выполнения программы. По полученным результатам заполнить таблицу

Исходные данные			**Ожидаемый результат**		**Фактический результат**		**Вывод**
a	b	c	x_1	x_2	x_1	x_2	

№ 3. Протестировать консольное приложение, вычисляющее корни квадратного уравнения. Модульный тест, содержит не менее десяти тестовых методов (пять верных и пять неверных), проверяющих правильность выполнения программы. По полученным результатам заполнить таблицу

Исходные данные			Ожидаемый результат	Фактический результат	Вывод
a	*b*	c			

№ 4. Протестировать консольное приложение, вычисляющее стоимость покупки с учетом скидки. Скидка в 10% предоставляется, если сумма покупки больше 1000 руб. Модульный тест, содержит не менее десяти тестовых методов (пять верных и пять неверных), проверяющих правильность выполнения программы. По полученным результатам заполнить таблицу

Исходные данные	Ожидаемый результат	Фактический результат	Вывод
S			

№ 5. Даны целые числа m, n. Если числа не равны, то заменить каждое из них тем же числом, равным большему из исходных, а если равны, то заменить числа нулями. Модульный тест, содержит не менее десяти тестовых методов (пять верных и пять неверных), проверяющих правильность выполнения программы. По полученным результатам заполнить таблицу

Исходные данные		Ожидаемый результат	Фактический результат	Вывод
m	*n*			

Контрольные вопросы:

1. Какой тест называется модульным?
2. С какой целью используются модульные тесты?
3. Назовите три стадии в модульном тесте

Критерии оценки

оценка «5» ставится, если:

– работа выполнена полностью;

– возможна одна (две) неточность, не являющаяся следствием незнания или не понимания материала.

оценка «4» ставится, если:

– правильно выполнена большая часть работы (свыше 85%), допущено не более трех ошибок;

– работа выполнена полностью, но обоснования шагов решения недостаточны.

оценка «3» ставится, если:

– работа выполнена не полностью, допущено более трех ошибок, но обучающийся владеет основными знаниями, умениями по проверяемой дисциплине.

оценка «2» ставится, если:

– допущены существенные ошибки, показывающие, что обучающийся не владеет обязательными знаниями, умениями по данной дисциплине

ПРАКТИЧЕСКАЯ РАБОТА № 6

Тема: Отладка и тестирование информационных систем

Цель: Изучение правил создания модульных тестов для консольных приложений

Перечень оснащения и оборудования, источников: ПК, раздаточный материал

Время на подготовку и выполнение: 2 часа

Задание: Модульное тестирование

Порядок работы

1. Повторить теоретический материал

2. Выполнить задание

3. Оформить отчет в текстовом редакторе. Параметры шрифта: Times New Roman, 12 пт. Параметры абзаца междустрочный интервал одинарный. Текст выравнивается по ширине

4. Отчет должен содержать таблицу

Входящие данные	Ожидаемый результат	Фактический результат	Вывод

5. Сдать отчет вместе с проектом, который включает в себя тестируемое приложение и модульный тест

№ 1. Протестировать приложение Windows Form (.Net Framework), вычисляющее стоимость покупки с учетом скидки. Скидка в 3% предоставляется, если сумма покупки больше 500 руб, в 5% — если сумма больше 1000 руб. Модульный тест, содержит не менее десяти тестовых методов (пять верных и пять неверных), проверяющих правильность выполнения программы. При обращении модульного теста к проекту нужно установить

ссылку на сборку System.Windows.Forms 4.0.0. По полученным результатам заполнить таблицу

№ 2. Протестировать приложение Windows Form (.Net Framework), вычисляющее сопротивление электрической цепи, состоящей из двух параллельно соединенных сопротивлений.

Сопротивление электрической цепи:

$$r = \frac{R1 \cdot R2}{R1 + R2}$$

Модульный тест, содержит не менее десяти тестовых методов (пять верных и пять неверных), проверяющих правильность выполнения программы. По полученным результатам заполнить таблицу

№ 3. Протестировать приложение Windows Form (.Net Framework), вычисляющее стоимость поездки на автомобиле на дачу (туда и обратно). Исходными данными являются: расстояние до дачи (км); количество бензина, которое потребляет автомобиль на 100 км пробега; цена одного литра бензина.

summ = 2 * Потребление_бензина/100 * Расстояние * Цена;

Модульный тест, содержит не менее десяти тестовых методов (пять верных и пять неверных), проверяющих правильность выполнения программы. По полученным результатам заполнить таблицу

Контрольные вопросы:

1. Какой тест называется модульным?
2. С какой целью используются модульные тесты?
3. Назовите три стадии в модульном тесте

Критерии оценки

оценка «5» ставится, если:

– работа выполнена полностью;

– возможна одна (две) неточность, не являющаяся следствием незнания или не понимания материала.

оценка «4» ставится, если:

– правильно выполнена большая часть работы (свыше 85%), допущено не более трех ошибок;

– работа выполнена полностью, но обоснования шагов решения недостаточны.

оценка «3» ставится, если:

– работа выполнена не полностью, допущено более трех ошибок, но обучающийся владеет основными знаниями, умениями по проверяемой дисциплине.

оценка «2» ставится, если:

– допущены существенные ошибки, показывающие, что обучающийся не владеет обязательными знаниями, умениями по данной дисциплине

ПРАКТИЧЕСКАЯ РАБОТА № 7

Тема: Отладка и тестирование информационных систем

Цель: закрепление практических навыков создания тестовых сценариев

Перечень оснащения и оборудования, источников: ПК, раздаточный материал

Время на подготовку и выполнение: 2 часа

Задание: Разработка тестовых сценариев

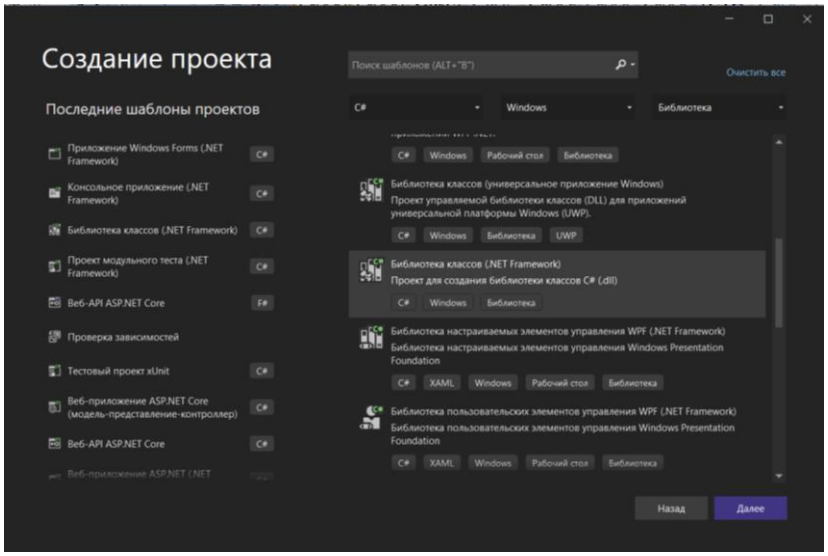
Порядок работы

№ 1. Создание сценария для проекта

Инструкция

I. Создание тестируемого проекта

1. Создать новый проект BankAccountNS для тестирования в Библиотеки классов.



2. Код проекта:

```
using System;
namespace BankAccountNS
{      public class BankAccount
   {  private readonly string m_customerName;
       private double m_balance;
       private BankAccount() { }
public BankAccount(string customerName, double balance)
{        m_customerName = customerName;
          m_balance = balance;}
public string CustomerName
```

{get { return m_customerName; }}

public double Balance

{get { return m_balance; }}

public void Debit(double amount)

{if (amount > m_balance)

{throw new ArgumentOutOfRangeException("amount");}

if (amount < 0)

{throw new ArgumentOutOfRangeException("amount");}

m_balance += amount; // intentionally incorrect code}

public void Credit(double amount)

{if (amount < 0)

{throw new ArgumentOutOfRangeException("amount");}

m_balance += amount;}

public static void Main()

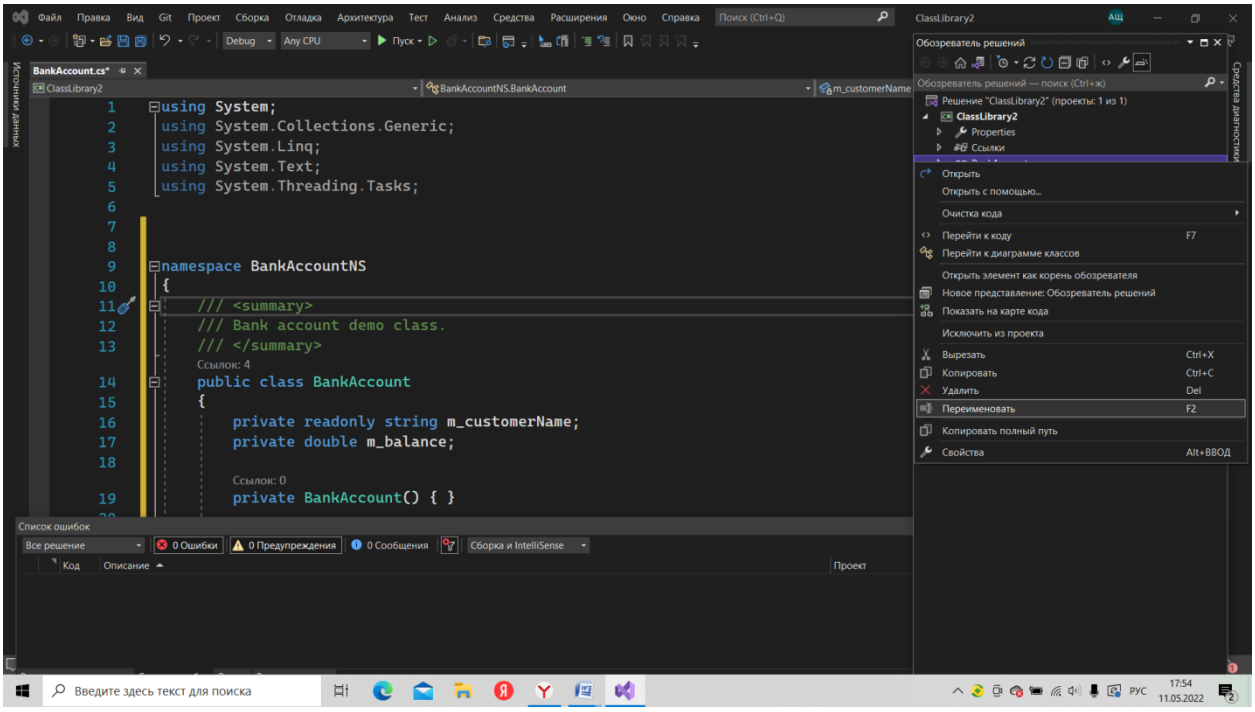
{BankAccount ba = new BankAccount("Mr. Bryan Walton", 11.99);

ba.Credit(5.77);

ba.Debit(11.22);

Console.WriteLine("Current balance is ${0}", ba.Balance);}}}

2. Переименовать файл Class1.cs в BankAccount.cs, щелкнув его правой кнопкой мыши и выбрав команду Переименовать в Обозревателе решений



3. В меню Сборка нажмите Построить решение

4. Таким образом, создан проект с именем «Bank», который содержит исходный код, подлежащий тестированию. Пространство имен BankAccountNS проекта «Bank», содержит открытый класс «BankAccount», методы которого будут тестироваться.

Провести тестирование на примере метода Debit. Метод Debit вызывается, когда денежные средства снимаются со счета. Определение метода Debit имеет вид:

public void Debit(double amount)

{ if(amount > m_balance) { throw new ArgumentOutOfRangeException(«amount»); }

if (amount < 0) { throw new ArgumentOutOfRangeException(«amount»); }

m_balance += amount; }

II. Создание проекта модульного теста

1. Выполнить команду Файл/ Добавить / Создать проект

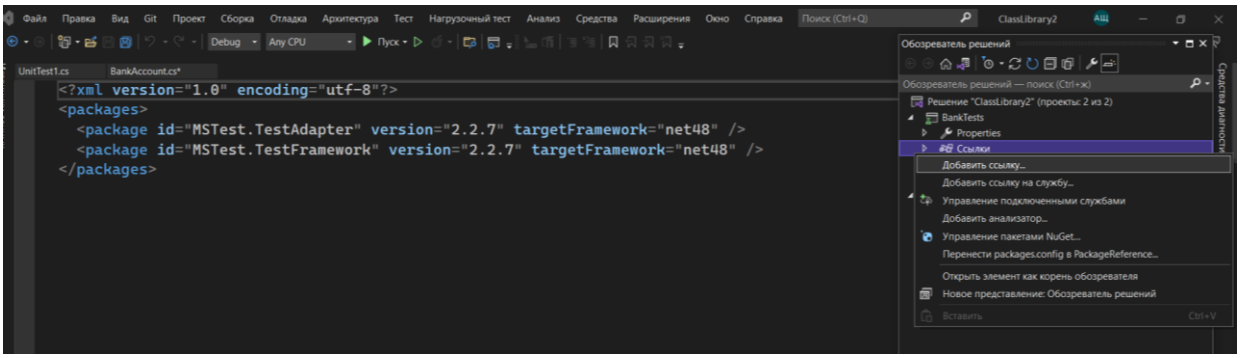
2. В диалоговом окне Новый проект установить параметры: C#, Тестирование

3. В списке шаблонов выберите Проект модульного теста

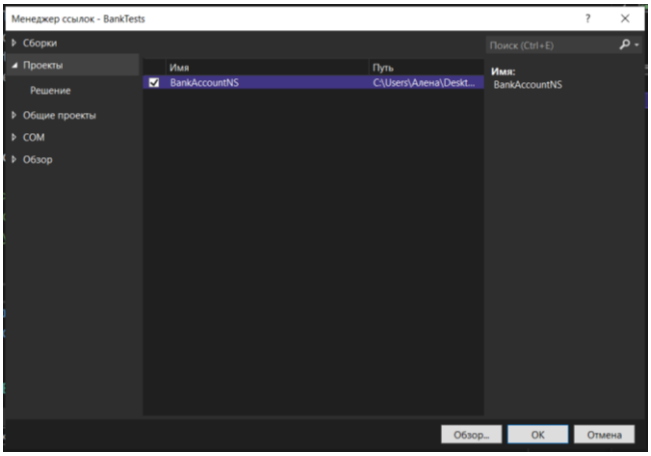
4. В поле Имя введите BankTests

5. В результате проект BankTests добавляется в решение Bank

6. В проекте BankTests добавить ссылку на проект Bank, для этого в Обозревателе решений щелкните Ссылки в проекте BankTests, а затем выберите в контекстном меню Добавить ссылку



7. В диалоговом окне Диспетчер ссылок разверните Решение и выберите название проекта



8. Создать тестовый класс, для того чтобы проверить класс BankAccount.

9. Переименовать имя класса модульного теста UnitTest1.cs в BankAccountTests.cs.

10. Добавить оператор using в класс, чтобы тестируемый проект можно было вызывать без использования полных имен.

using BankAccountNS;

11. Минимальные требования к тестовому классу следующие:

– Атрибут [TestClass] является обязательным для платформы модульных тестов Microsoft для управляемого кода в любом классе, содержащем методы модульных тестов, которые необходимо выполнить в обозревателе тестов.

– Каждый метод теста, предназначенный для запуска в обозревателе тестов, должен иметь атрибут [TestMethod].

12. Написать методы модульного теста для проверки поведения метода Debit класса BankAccount.

Существует три поведения, которые требуется проверить:

–Метод создает исключение ArgumentOutOfRangeException, если сумма по дебету превышает баланс.

–Метод создает исключение ArgumentOutOfRangeException, если сумма по дебету меньше нуля.

–Если значение дебета допустимо, то метод вычитает сумму дебета из баланса счета.

13. Создать тест, который проверяет, снимается ли со счета нужная сумма при допустимом размере кредита (со значением меньшим, чем баланс счета, и большим, чем ноль). Добавьте следующий метод в этот класс BankAccountTests :

```
 [TestMethod]
public void Debit_WithValidAmount_UpdatesBalance()
{
// Arrange
 double beginningBalance = 11.99;
double debitAmount = 4.55;
double expected = 7.44;
BankAccount account = new BankAccount(«Mr. Bryan Walton», beginningBalance);
// Act
account.Debit(debitAmount);
// Assert
double actual = account.Balance; Assert.AreEqual(expected, actual, 0.001, «Account not debited correctly»);
}
```

Метод очень прост: он создает новый объект BankAccount с начальным балансом, а затем снимает допустимое значение. Он использует метод AreEqual, чтобы проверить, что конечный баланс соответствует ожидаемому

14. Метод теста должен удовлетворять следующим требованиям:

– декорируется атрибутом [TestMethod]

– возвращает void

– не должен иметь параметров

15. Сборка и запуск теста

– В меню Построение выбрать Построить решение. Если ошибок нет, появится обозреватель тестов

– Выбрать Запустить все, чтобы выполнить тест

В данном случае тест пройден не будет. Метод теста будет перемещен в группу Неудачные тесты. Выберите этот метод в обозревателе тестов для просмотра сведений в нижней части окна. 16. Анализ результатов теста. Результат теста содержит сообщение, описывающее возникшую ошибку. Для метода AreEqual сообщение отражает ожидаемый результат (параметр Ожидается) и фактически полученный (параметр Фактическое). Ожидалось, что баланс уменьшится, а вместо этого он увеличился на сумму списания. Модульный тест обнаружил ошибку: сумма списания добавляется на баланс счета, вместо того чтобы вычитаться.

17. Исправление ошибки. Для исправления ошибки замените строку:

m_balance += amount; на: m_balance -= amount;

18. Повторный запуск теста. В обозревателе тестов выбрать Запустить все, чтобы запустить тест повторно. Красно-зеленая строка состояния станет зеленой, сигнализируя о том, что тест пройден, а сам тест будет перемещен в группу Пройденные тесты

19. Рассмотрим, как последовательный процесс анализа, разработки модульных тестов и рефакторинга может помочь сделать рабочий код более надежным и эффективным.

20. Анализ проблем. Ранее был создан тестовый метод для подтверждения того, что допустимая сумма правильно вычитается в методе Debit. Проверим, что метод создает исключение ArgumentOutOfRangeException, если сумма по дебету:

– больше баланса

– меньше нуля

21. Создать метод теста для проверки правильного поведения в случае, когда сумма по дебету меньше нуля:

```
[TestMethod]
[ExpectedException(typeof(ArgumentOutOfRangeException))]
```

```
public void Debit_WhenAmountIsLessThanZero_ShouldThrowArgumentOutOfRange()
{
// Arrange
double beginningBalance = 11.99;
double debitAmount = -100.00;
BankAccount account = new BankAccount(«Mr. Bryan Walton», beginningBalance);
// Act
account.Debit(debitAmount);
 // Assert is handled by the ExpectedException attribute on the test method.
}
```

Атрибут ExpectedExceptionAttribute используется для подтверждения правильности созданного исключения. Данный атрибут приводит к тому, что тест не будет пройден, если не возникнет исключения ArgumentOutOfRangeException. Если временно изменить тестируемый метод для вызова более общего исключения ApplicationException при значении суммы по дебету меньше нуля, то тест работает правильно – то есть завершается неудачно. Чтобы проверить случай, когда размер списания превышает баланс, выполните следующие действия:

1. Создать новый метод теста с именем Debit_WhenAmountIsMoreThanBalance_ShouldThrowArgumentOutOfRange.
2. Скопировать тело метода из Debit_WhenAmountIsLessThanZero_ShouldThrowArgumentOutOfRange в новый метод.
3. Присвоить debitAmount значение, превышающее баланс.
4. Запустить тесты. Запуск двух методов теста показывает, что тесты работают правильно.
5. Последние два тестовых метода вызывают беспокойство. Нельзя быть уверенным, какое именно условие тестируемого метода создает исключение при запуске любого из тестов. Если каким-либо способом разделить эти два условия, а именно отрицательную сумму по дебету и сумму, большую, чем баланс, то это увеличит достоверность проведения тестов. Если посмотреть на тестируемый метод, можно заметить, что оба условных оператора используют конструктор ArgumentOutOfRangeException, который просто получает имя аргумента в качестве параметра: throw new ArgumentOutOfRangeException(«amount»);

Так выглядит конструктор, который можно использовать для сообщения более детальной информации: ArgumentOutOfRangeException(String, Object, String) включает имя аргумента, значения аргумента и определяемое пользователем сообщение.

Можно выполнить рефакторинг тестируемого метода для использования данного конструктора.

6. Рефакторинг тестируемого кода.

Определим две константы для сообщений об ошибках в области видимости класса. Добавьте это в тестируемый класс BankAccount:

```
public const string DebitAmountExceedsBalanceMessage = «Debit amount exceeds balance»;
public const string DebitAmountLessThanZeroMessage = «Debit amount is less than zero»;
```

Затем изменим два условных оператора в методе Debit:

```
if (amount > m_balance)
{throw new ArgumentOutOfRangeException(«amount», amount, DebitAmountExceedsBalanceMessage);}
if (amount < 0)
{throw new ArgumentOutOfRangeException(«amount», amount, DebitAmountLessThanZeroMessage);
}
```

Рефакторинг тестовых методов. Удалим атрибут ExpectedException метода теста, и вместо этого будем перехватывать исключение и проверять соответствующее ему сообщение.

Метод StringAssert.Contains обеспечивает возможность сравнения двух строк. В этом случае метод Debit_WhenAmountIsMoreThanBalance_ShouldThrowArgumentOutOf Range может выглядеть следующим образом:

```
[TestMethod]
public void Debit_WhenAmountIsMoreThanBalance_ShouldThrowArgumentOutOfRange()
{
// Arrange
double beginningBalance = 11.99;
double debitAmount = 20.0;
BankAccount account = new BankAccount(«Mr. Bryan Walton», beginningBalance);
// Act
try
{
 account.Debit(debitAmount);
}
 catch (ArgumentOutOfRangeException e)
{
 // Assert
StringAssert.Contains(e.Message, BankAccount.DebitAmountExceedsBalanceMessage); } }
```

7. Повторное тестирование, переписывание и анализ. Предположим, что в тестируемом методе есть ошибка, и метод Debit даже не создает исключение ArgumentOutOfRangeException, не говоря уже о выводе правильного сообщения с исключением. В этом случае метод теста не сможет обработать этот случай. Если значение debitAmount допустимо (то есть меньше баланса, но больше нуля), то исключение не перехватывается, а утверждение никогда не сработает. Однако метод теста проходит успешно. Это не правильно, поскольку метод теста должен был завершиться с ошибкой в том случае, если исключение не создается. Это является ошибкой в методе теста. Для решения этой проблемы добавим утверждение Fail в конце тестового метода для обработки случая, когда исключение не создается. Однако повторный запуск теста показывает, что тест теперь оказывается не пройденным при перехватывании верного исключения. Блок catch перехватывает исключение, но метод продолжает выполняться, и в нем происходит сбой на новом утверждении Fail. Чтобы разрешить эту проблему, добавим оператор return после StringAssert в блоке catch. Повторный запуск теста подтверждает, что проблема устранена. Окончательная версия метода Debit_WhenAmountIsMoreThanBalance_ShouldThrowArgument- OutOfRange выглядит следующим образом:

```
[TestMethod]
public void Debit_WhenAmountIsMoreThanBalance_ShouldThrowArgumentOutOfRange()
{
 // Arrange
double beginningBalance = 11.99;
double debitAmount = 20.0;
BankAccount account = new BankAccount(«Mr. Bryan Walton», beginningBalance);
// Act
try { account.Debit(debitAmount); }
catch (ArgumentOutOfRangeException e)
{
// Assert
StringAssert.Contains(e.Message, BankAccount.DebitAmountExceedsBalanceMessage); return;
}
Assert.Fail(«The expected exception was not thrown.»); }
```

Усовершенствования тестового кода привели к созданию более надежных и информативных методов теста. Но что более важно, в результате был также улучшен тестируемый код.

Контрольные вопросы:

1. С какой целью используются модульные тесты?
2. Что такое рефакторинг кода?
3. Как провести рефакторинг, используя модульные тесты?

Критерии оценки

оценка «5» ставится, если:

– работа выполнена полностью;

– возможна одна (две) неточность, не являющаяся следствием незнания или не понимания материала.

оценка «4» ставится, если:

– правильно выполнена большая часть работы (свыше 85%), допущено не более трех ошибок;

– работа выполнена полностью, но обоснования шагов решения недостаточны.

оценка «3» ставится, если:

– работа выполнена не полностью, допущено более трех ошибок, но обучающийся владеет основными знаниями, умениями по проверяемой дисциплине.

оценка «2» ставится, если:

– допущены существенные ошибки, показывающие, что обучающийся не владеет обязательными знаниями, умениями по данной дисциплине

ПРАКТИЧЕСКАЯ РАБОТА № 8

Тема: Отладка и тестирование информационных систем

Цель: Изучение способов создания требований и матрицы трассировки

Перечень оснащения и оборудования, источников: ПК, раздаточный материал

Время на подготовку и выполнение: 2 часа

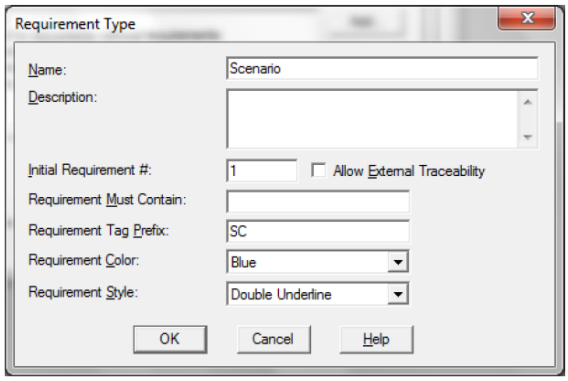
Задание: Разработка тестовых сценариев

Порядок работы

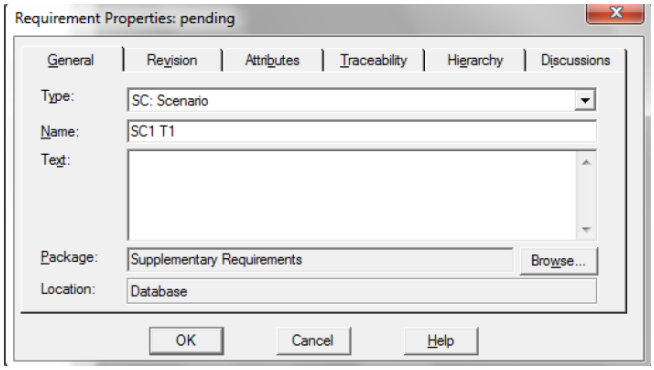
1. Повторить теоретический материал
2. Выполнить задание
3. Оформить отчет в текстовом редакторе. Параметры шрифта: Times New Roman, 12 пт. Параметры абзаца междустрочный интервал одинарный. Текст выравнивается по ширине
4. Сдать отчет вместе с проектом, который включает в себя тестируемое приложение и модульный тест

№ 1. Создать новый тип требования

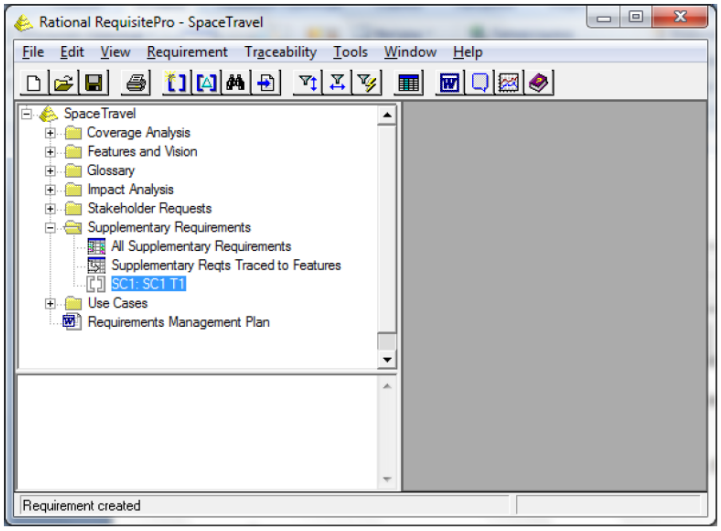
1. Создать новый тип требований: конт. меню SpaceTravel → Properties → ф. Project Properties | вкл. Requirements Type | кн. Add → ф. Requirement Type | Name ← Scenario, Requirement Tag Prefix ← SC, остальные параметры – по умолчанию, кн. Ok → ф. Project Properties | кн. Ok



№ 2. Создать новое требование: SpaceTravel → Supplementary Requirements → конт. меню → New → Requirement… → ф. Requirement Properties: pending | Type ← SC: Scenario, Name ← SC1 T1 (номер сценария использования + номер тестового сценария), кн. Ok



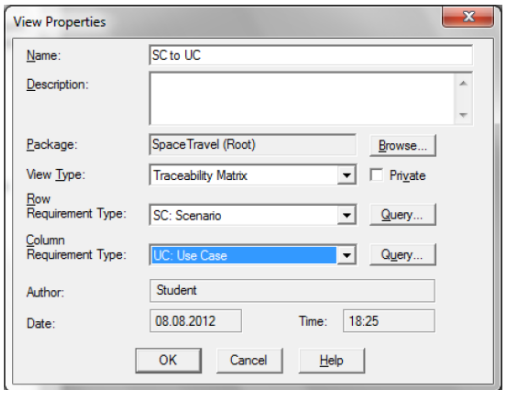
Проверить наличие нового требования в окне проекта. Аналогичным

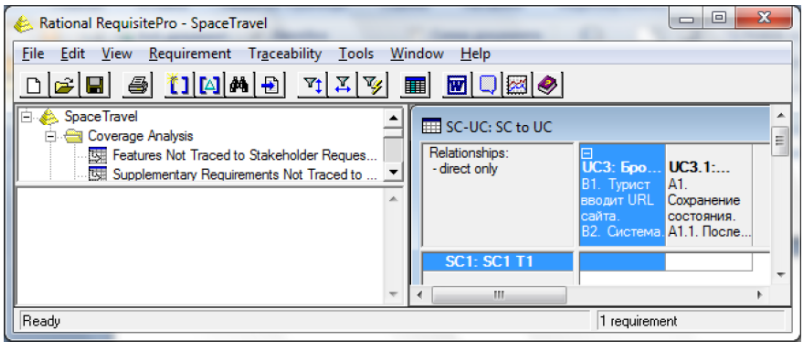


Аналогичным образом определить требования типа Scenario для остальных сценариев использования и тестовых сценариев.

№ 3. Создать матрицу трассировки для связи требований Scenario с требованиями других типов:

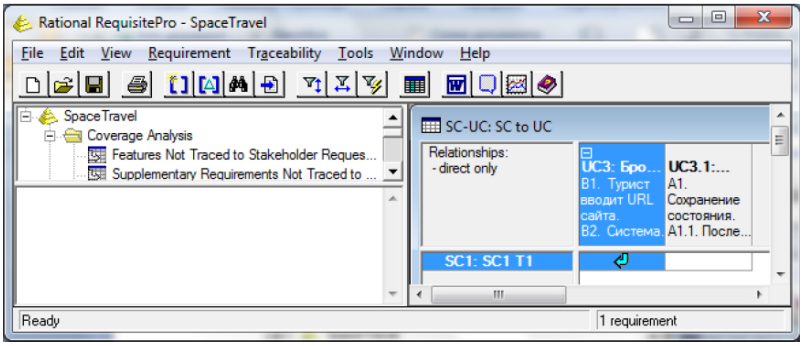
3.1. Создать матрицу трассировки ф. Rational RequisitePro – SpaceTravel → конт. меню → New → View… → ф. View Properties | Name ← SC to UC, View Type ← Tra-ceability Matrix, Row Requirement Type ← SC: Scenario, Column Re-quirement Type ← UC: Use Case, кн. Ok → матрица трассировки на экране





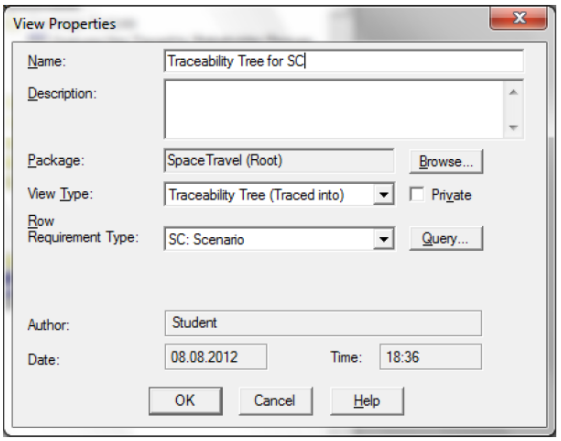
3.2. Установить связи между требованиями типа SC и UC

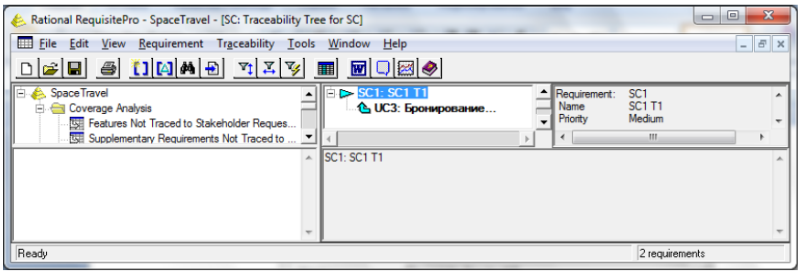
Сценарий использования (Use Case) и основанный на нем тес-товый сценарий (Scenario) → установить курсор на пересечении строки, соответствующей тестовому сценарию, и столбца, соответствующего сценарию использования → контекстное меню → Trace From.



3.3. Аналогичным образом установить связи для всех тестовых сценариев (один сценарий использования может привести к нескольким тестовым сценариям) 3.4. Закрыть матрицу трассировки

4. Создать дерево трассировки ф. Rational RequisitePro – SpaceTravel → конт. меню → New → View… → ф. View Properties | Name ← Traceability Tree for SC, View Type ← Traceability Tree (Traced into), Row Requirement Type ← SC: Scenario, кн. Ok → результат на экране





Контрольные вопросы:

1. Приведите порядок составления тестового сценария.
2. Что такое матрица трассировки?
3. Как можно задать матрицу трассировки?

Критерии оценки

оценка «5» ставится, если:

– работа выполнена полностью;

– возможна одна (две) неточность, не являющаяся следствием незнания или не понимания материала.

оценка «4» ставится, если:

– правильно выполнена большая часть работы (свыше 85%), допущено не более трех ошибок;

– работа выполнена полностью, но обоснования шагов решения недостаточны.

оценка «3» ставится, если:

– работа выполнена не полностью, допущено более трех ошибок, но обучающийся владеет основными знаниями, умениями по проверяемой дисциплине.

оценка «2» ставится, если:

– допущены существенные ошибки, показывающие, что обучающийся не владеет обязательными знаниями, умениями по данной дисциплине

ПРАКТИЧЕСКАЯ РАБОТА № 9

Тема: Отладка и тестирование информационных систем

Цель: закрепить навыки разработки тестовых пакетов

Перечень оснащения и оборудования, источников: ПК, раздаточный материал

Время на подготовку и выполнение: 2 часа

Задание: Разработка тестовых пакетов

Порядок работы

1. Повторить теоретический материал
2. Выполнить задание
3. Оформить отчет в текстовом редакторе. Параметры шрифта: Times New Roman, 12 пт. Параметры абзаца междустрочный интервал одинарный. Текст выравнивается по ширине
4. Отчет должен содержать таблицу

Входящие данные	Ожидаемый результат	Фактический результат	Вывод

5. Сдать отчет вместе с проектом
6. Критерии покрытия кода по методу белого ящика:

– покрытие операторов — каждая ли строка исходного кода была выполнена и протестирована;

– покрытие условий — каждая ли точка решения (вычисления истинно ли или ложно выражение) была выполнена и протестирована;

– покрытие путей — все ли возможные пути через заданную часть кода были выполнены и протестированы;

– покрытие функций — каждая ли функция программы была выполнена;

– покрытие вход/выход — все ли вызовы функций и возвраты из них были выполнены;

– покрытие значений параметров — все ли типовые и граничные значения параметров были проверены.

№ 1. В Древней Греции (II в. до н.э.) был известен шифр, называемый "квадрат Полибия". Шифровальная таблица представляла собой квадрат с пятью столбцами и пятью строками, которые нумеровались цифрами от 1 до 5. В каждую клетку такого квадрата записывалась одна буква. В результате каждой букве соответствовала пара чисел, и шифрование сводилось к замене буквы парой чисел. Для латинского алфавита квадрат Полибия имеет вид:

	1	2	3	4	5
1	A	B	C	D	E
2	F	G	H	I, J	K
3	L	M	N	O	P
4	Q	R	S	T	U
5	V	W	X	Y	Z

Пользуясь вышеизложенным способом написать программу, которая:

– зашифрует введенный текст и выведет на экран;

– считает зашифрованный текст и расшифрует данный текст

Построить блок-схемы ветвления, обозначить буквами или цифрами ветви этих алгоритмов. Выписать пути алгоритма, которые должны быть проверены тестами для выбранного метода тестирования.

Записать тесты, которые позволят пройти по путям алгоритма. Протестировать разработанную программу.

Результаты оформить в виде таблицы:

Входящие данные	Ожидаемый результат	Фактический результат	Вывод

Проверить все виды тестов и сделать выводы об их эффективности.

Оформить отчет.

Контрольные вопросы:

1. Назовите правила покрытия выписки (покрытие операторов)
2. Покрытие решения (покрытие ветвей)
3. Покрытие условий
4. Комбинированное покрытие условий ветвления

Критерии оценки

оценка «5» ставится, если:

– работа выполнена полностью;

– возможна одна (две) неточность, не являющаяся следствием незнания или не понимания материала.

оценка «4» ставится, если:

– правильно выполнена большая часть работы (свыше 85%), допущено не более трех ошибок;

– работа выполнена полностью, но обоснования шагов решения недостаточны.

оценка «3» ставится, если:

– работа выполнена не полностью, допущено более трех ошибок, но обучающийся владеет основными знаниями, умениями по проверяемой дисциплине.

оценка «2» ставится, если:

– допущены существенные ошибки, показывающие, что обучающийся не владеет обязательными знаниями, умениями по данной дисциплине

ПРАКТИЧЕСКАЯ РАБОТА № 10

Тема: Отладка и тестирование информационных систем

Цель: получить навыки разработки тестовых пакетов

Перечень оснащения и оборудования, источников: ПК, раздаточный материал

Время на подготовку и выполнение: 2 часа

Задание: Разработка тестовых пакетов

Порядок работы

1. Повторить теоретический материал
2. Выполнить задание
3. Оформить отчет в текстовом редакторе. Параметры шрифта: Times New Roman, 12 пт. Параметры абзаца междустрочный интервал одинарный. Текст выравнивается по ширине
4. Отчет должен содержать таблицу

Входящие данные	Ожидаемый результат	Фактический результат	Вывод

5. Сдать отчет вместе с проектом.
6. Критерии покрытия кода по методу белого ящика:

– покрытие операторов — каждая ли строка исходного кода была выполнена и протестирована;

– покрытие условий — каждая ли точка решения (вычисления истинно ли или ложно выражение) была выполнена и протестирована;

– покрытие путей — все ли возможные пути через заданную часть кода были выполнены и протестированы;

– покрытие функций — каждая ли функция программы была выполнена;

– покрытие вход/выход — все ли вызовы функций и возвраты из них были выполнены;

– покрытие значений параметров — все ли типовые и граничные значения параметров были проверены.

№ 1. Написать и протестировать программу, которая вычисляет стоимость покупки с учетом скидки. Скидка в 3% предоставляется, если сумма покупки больше 500 руб, в 5% — если сумма больше 1000 руб.

Построить блок-схемы ветвления, обозначить буквами или цифрами ветви этих алгоритмов. Выписать пути алгоритма, которые должны быть проверены тестами для выбранного метода тестирования.

Записать тесты, которые позволят пройти по путям алгоритма. Протестировать разработанную программу.

Результаты оформить в виде таблицы:

Входящие данные	Ожидаемый результат	Фактический результат	Вывод

Проверить все виды тестов и сделать выводы об их эффективности.

Оформить отчет.

№ 2. Написать и протестировать программу, которая сравнивает три введенных с клавиатуры числа. больше, или, если числа равны, вывести соответствующее сообщение.

Построить блок-схемы ветвления, обозначить буквами или цифрами ветви этих алгоритмов. Выписать пути алгоритма, которые должны быть проверены тестами для выбранного метода тестирования.

Записать тесты, которые позволят пройти по путям алгоритма. Протестировать разработанную программу.

Результаты оформить в виде таблицы:

Входящие данные	Ожидаемый результат	Фактический результат	Вывод

Проверить все виды тестов и сделать выводы об их эффективности.

Оформить отчет.

Контрольные вопросы:

1. Назовите правила покрытия выписки (покрытие операторов)
2. Покрытие решения (покрытие ветвей)
3. Покрытие условий
4. Комбинированное покрытие условий ветвления

Критерии оценки

оценка «5» ставится, если:

– работа выполнена полностью;

– возможна одна (две) неточность, не являющаяся следствием незнания или не понимания материала.

оценка «4» ставится, если:

– правильно выполнена большая часть работы (свыше 85%), допущено не более трех ошибок;

– работа выполнена полностью, но обоснования шагов решения недостаточны.

оценка «3» ставится, если:

– работа выполнена не полностью, допущено более трех ошибок, но обучающийся владеет основными знаниями, умениями по проверяемой дисциплине.

оценка «2» ставится, если:

– допущены существенные ошибки, показывающие, что обучающийся не владеет обязательными знаниями, умениями по данной дисциплине

ПРАКТИЧЕСКАЯ РАБОТА № 11

Тема: Отладка и тестирование информационных систем

Цель: приобретение практических навыков по юзабилити-тестированию и оценке качества интерфейса на основе измерения показателей результативности, эффективности и удовлетворенности; оценить качество программы с помощью метрики Чепина.

Перечень оснащения и оборудования, источников: ПК, раздаточный материал

Время на подготовку и выполнение: 2 часа

Задание: Анализ качества интерфейсов юзабилити метрик

Порядок работы

1. Повторить теоретический материал
2. Выполнить задание
3. Оформить отчет в текстовом редакторе. Параметры шрифта: Times New Roman, 12 пт. Параметры абзаца междустрочный интервал одинарный. Текст выравнивается по ширине
4. Сдать отчет

№ 1. В тестировании принимало участие два пользователя, которые при работе с продуктом выполняли два тестовых задания. В результате тестирования первый пользователь успешно завершил только сценарий № 1, а второй - успешно завершил только сценарий №2.

Общая результативность исследуемого продукта рассчитывается по формуле:

$$\overline{E} = \frac{\sum_{j=1}^{R}\sum_{i=1}^{N} n_{ij}}{RN} \cdot 100\%$$

– E – общая результативность продукта;

– N – общее число сценариев (цели);
– R – число респондентов (пользователей);
– n_{ij} – результат выполнения сценария *i* пользователя *j*
– $n_{ij} = 1$, если сценарий i успешно завершен и цель пользователя j была достигнута
– $n_{ij} = 0$, если сценарий i не завершен и пользователя j не смог достичь цели

По полученным данным сделать выводы об общей результативности:

– 0-50% - очень плохой;
– 50-75% - плохой;
– 75-90 – нормальный;
– 90-100% - хороший

№ 2. В тестировании принимало участие четыре пользователя, которые при работе с продуктом выполняли одно тестовое задания. Три пользователя выполнили задание успешно, а один пользователь вышел из строя. Время, затраченное пользователями на выполнение задания:

– первый – 1 сек.,
– второй – 2 сек;
– третий – 3 сек;
– четвертый – 10 сек.

Определить временную относительную эффективность по данному продукту.

$$\overline{\mathrm{E}} = \frac{\sum_{\mathrm{j=1}}^{\mathrm{R}} \sum_{\mathrm{i=1}}^{\mathrm{N}} \mathrm{n_{ij} t_{ij}}}{\sum_{\mathrm{j=1}}^{\mathrm{R}} \sum_{\mathrm{i=1}}^{\mathrm{N}} \mathrm{t_{ij}}} \cdot 100\%$$

P − общая эффективность продукта;

N − общее число сценариев (цели);

R − число респондентов (пользователей);

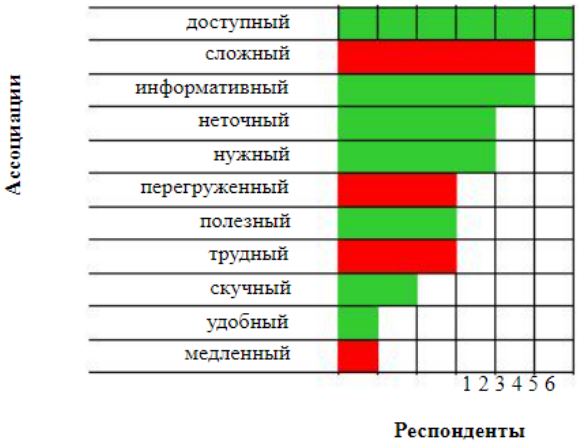
n_{ij} − это результат выполнения сценария i пользователем j;

$n_{ij} = 1$, если сценарий i успешно завершен и цель пользователем j была достигнута;

$n_{ij} = 0$, если сценарий i не завершен и пользователь j не смог достичь цели;

t_{ij} − это время, затраченное на выполнение сценария i пользователем j.

№ 3. Схема «Ассоциации», построенная на основе данных, полученных от 6 пользователей. Из соображений удобства положительные ассоциации выделены зеленым цветом, а отрицательные – красным цветом.



На основе данных, полученных от 6 пользователей, рассчитать удовлетворенность от продукта.

$$\overline{S_A} = \frac{\sum_{j=1}^{R} A_j^+}{\sum_{j=1}^{R}(A_j^+ + A_j^-)} \cdot 100\%$$

S_A – процент удовлетворенности от продукта; R – число респондентов (пользователей);

A_j^+ – количество положительных слов ассоциаций от j-респондента;

A_j^- – количество отрицательных слов ассоциаций от j- респондента

№ 4. Найти на сайте mail.ru раздел про автомобили и выбрать для покупки автомобиль Пежо 408 с пробегом до 10 тыс. км. Фактически необходимо протестировать две пользовательские задачи:

– Задача № 1: поиск автомобильного раздела на портале mail.ru;

– Задача № 2: поиск автомобиля с пробегом на сайте Авто@Mail.Ru

В процессе выполнения заданий наблюдателю необходимо вести заметки, отображающие возникшие проблемы, а также время их выполнения

Обобщить полученные в процессе тестирования данные и рассчитать базовые юзабилити-метрики продукта: такие как:

– результативность;

– эффективность;

– удовлетворённость

Обработать результаты совместно с другими участниками группы. Выделить итоговый список ошибок и недочетов, сформировать рекомендации по дальнейшей модификации интерфейса

Оформить полученный материал в форме отчета

№ 5. Дана целочисленная матрица размером *N*M*. Вычислить и записать в одномерный массив количество простых чисел в каждом столбце матрицы. Размерность матрицы задается с клавиатуры, заполнение матрицы осуществляется посредством датчика

случайных чисел. Разработать программу для решения задачи. На основе лексического анализа исходного текста программы определить значение метрики Чепина.

Порядок выполнения

1.Написать программу

2. Оценить качество программы с помощью метрики Чепина

№ п/п	Наименование переменных	Номера строк
P (для расчетов и для обеспечения вывода)		
M (модифицируемые или создаваемые внутри программы переменные)		
C (управляющие переменные)		
T (не используемые в программе переменные)		

Вывод:

Переменные _____ используются в качестве исходных данных.

Переменные _____ в процессе выполнения программы создаются и модифицируются.

Переменные _____ используются для управления выполнением программы.

Таким образом, исходя из результатов анализа исходного текста программы, получаем следующие значения характеристик:

P = ___ - количество переменных для расчетов;

M =___ - количество модифицируемых переменных;

C =___ - количество переменных, используемых в управлении программой;

T = ___ - количество неиспользуемых переменных (такие переменные в программе отсутствуют).

Расчет метрики Чепина:

$$Q = P + 2M + 3 - C + 0{,}5T$$

Вывод. Например. На основе полученных значений метрики Чепина уровень сложности данного решения можно считать сравнительно низким, как так в исходном тексте программы используется незначительное количество переменных, что не затрудняет понимание программы.

Контрольные вопросы:

1. Каким образом можно оценить качество интерфейса юзабилити-метрик?
2. Как оценить результативность?

3. Как оценить эффективность?

4. Как оценить удовлетворённость?

5. Как оценить качество программы с помощью метрики Чепина?

Критерии оценки

оценка «5» ставится, если:

– работа выполнена полностью;

– возможна одна (две) неточность, не являющаяся следствием незнания или не понимания материала.

оценка «4» ставится, если:

– правильно выполнена большая часть работы (свыше 85%), допущено не более трех ошибок;

– работа выполнена полностью, но обоснования шагов решения недостаточны.

оценка «3» ставится, если:

– работа выполнена не полностью, допущено более трех ошибок, но обучающийся владеет основными знаниями, умениями по проверяемой дисциплине.

оценка «2» ставится, если:

– допущены существенные ошибки, показывающие, что обучающийся не владеет обязательными знаниями, умениями по данной дисциплине

ПРАКТИЧЕСКАЯ РАБОТА № 12

Тема: Отладка и тестирование информационных систем

Цель: изучение способов анализа результатов тестирования ПО

Перечень оснащения и оборудования, источников: ПК, раздаточный материал

Время на подготовку и выполнение: 2 часа

Задание: Использования инструментария анализа качества

Порядок работы

1. Повторить теоретический материал

2. Выполнить задание

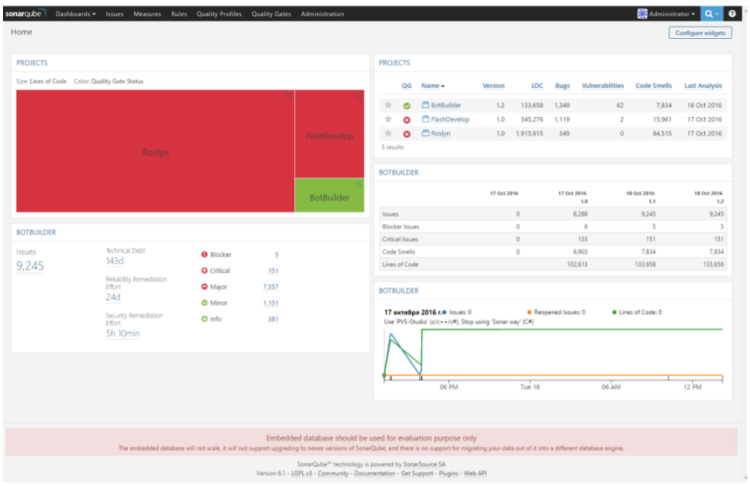
3. Оформить отчет в текстовом редакторе. Параметры шрифта: Times New Roman, 12 пт. Параметры абзаца междустрочный интервал одинарный. Текст выравнивается по ширине

4. Отчет должен содержать скрины с анализом качества

5. Сдать отчет

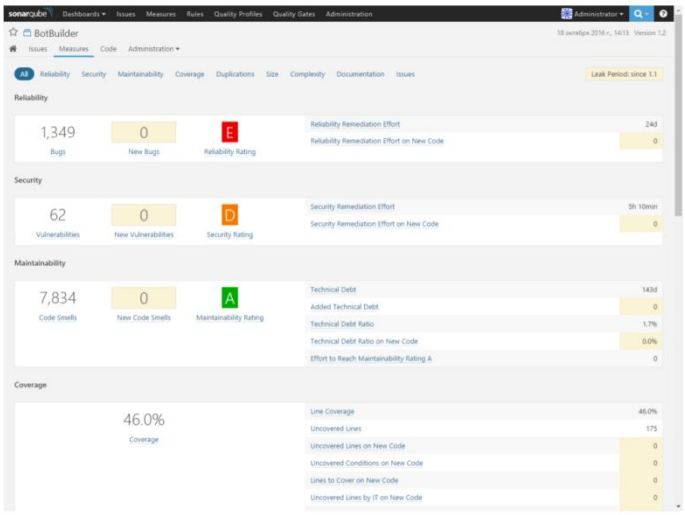
№ 1. Знакомство с интерфейсом программы SonarQube

На главной странице размещен список проектов, добавленных в систему, с краткой статистикой по каждому проекту: версия сборки, количество строк кода, количество багов, уязвимостей, дата последнего анализа:



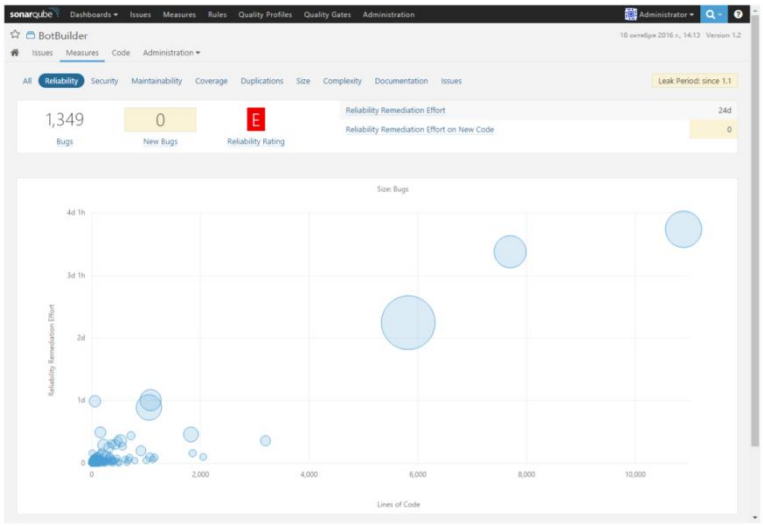
Настроить содержимое главной страницы с помощью большого набора встроенных виджетов, позволяющих визуализировать состояние кода проектов в SonarQube

Метрики проекта. Для получения более детальной информации о состоянии проекта перейти на страницу метрик проекта:

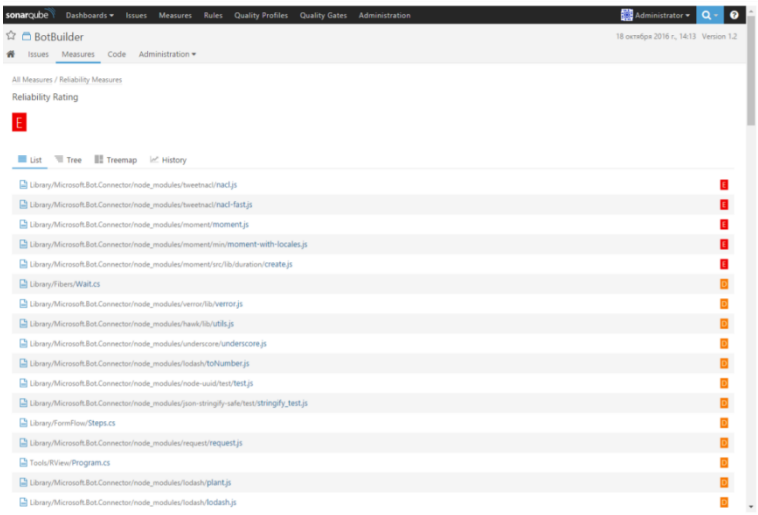


Здесь представлена информация о следующих метриках кода: Reliability (Надежность), Security (Безопасность), Maintainability (Поддерживаемость), Coverage (Покрытие тестами), Duplications (Дублирование), Size (Размер кодовой базы), Complexity (Цикломатическая сложность), Documentation (Документирование кода) и Issues (Ошибки).

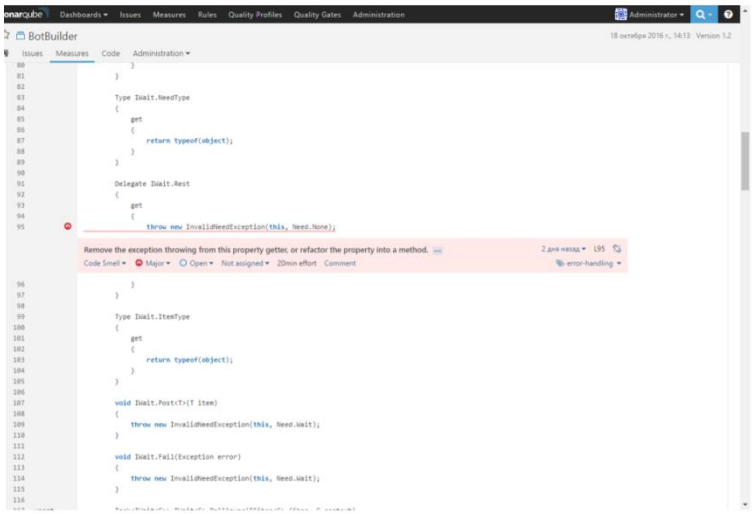
Перейти к метрике Reliability (Надежность). В результате отобразится информация об общем количестве обнаруженных багов и новые баги, обнаруженные во время последнего анализа, рейтинг надежности кода по шкале от A до E (E – наихудший рейтинг, свидетельствующий о том, что был найден, по крайней мере, один blocker баг), время, необходимое на устранение всех найденных ошибок:



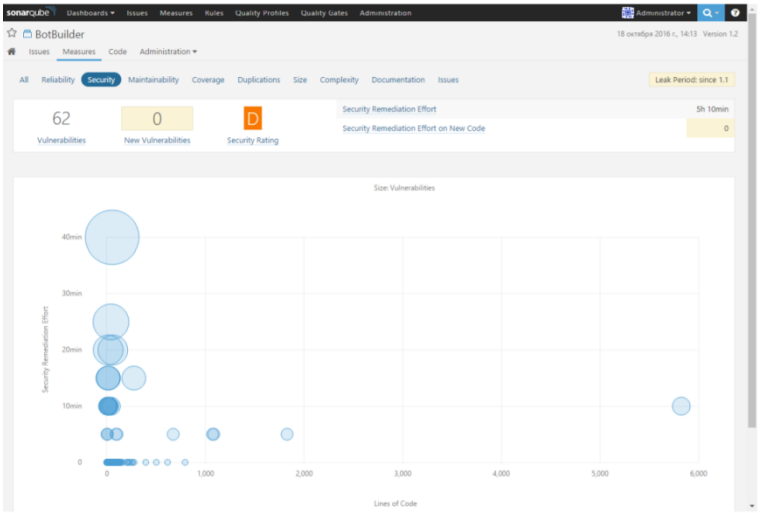
Платформа SonarQube позволяет анализировать метрики кода сверху вниз, от уровня проекта в целом до отдельных модулей и файлов. Так, например, если вы кликните на рейтинг надежности (Reliability Rating), отобразится список файлов проекта, отсортированных по возрастанию рейтинга надежности. Это позволит сфокусироваться на наиболее проблемных участках кода:



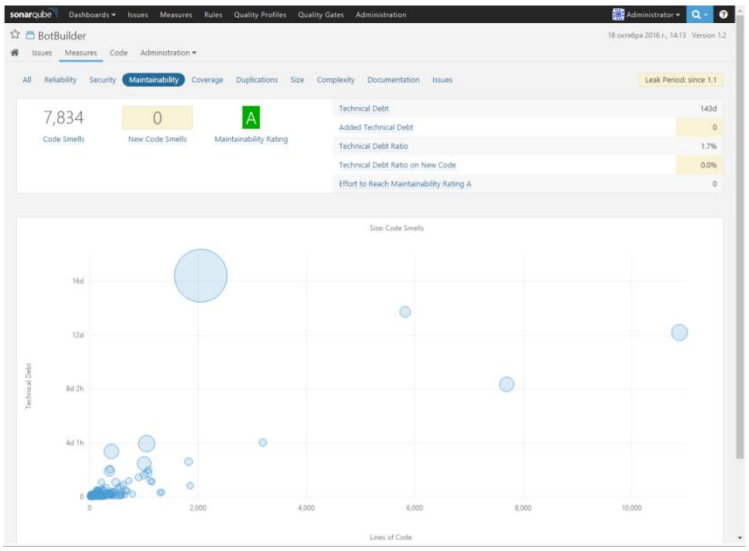
Перейти к файлу с исходным кодом и к конкретным участкам кода, в которых обнаружены ошибки:



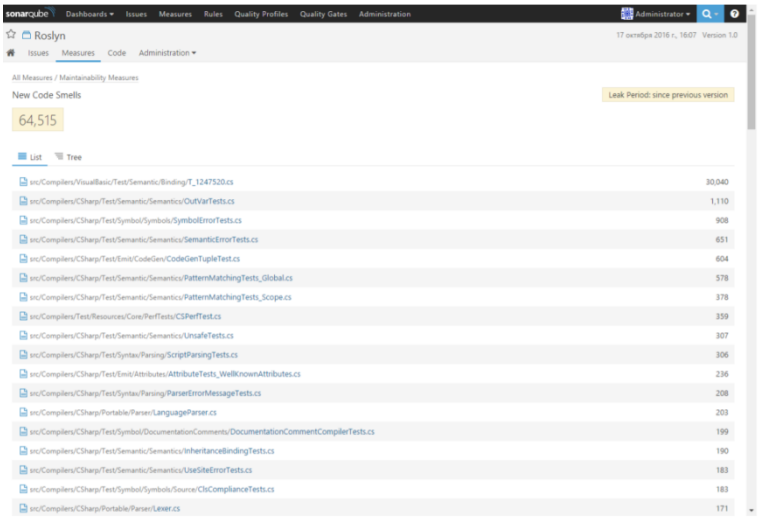
На странице метрики Security доступна информация об общем количестве уязвимостей, новых уязвимостях, рейтинге безопасности (также по шкале от A до E), и времени, которое потребуется на устранение уязвимостей:



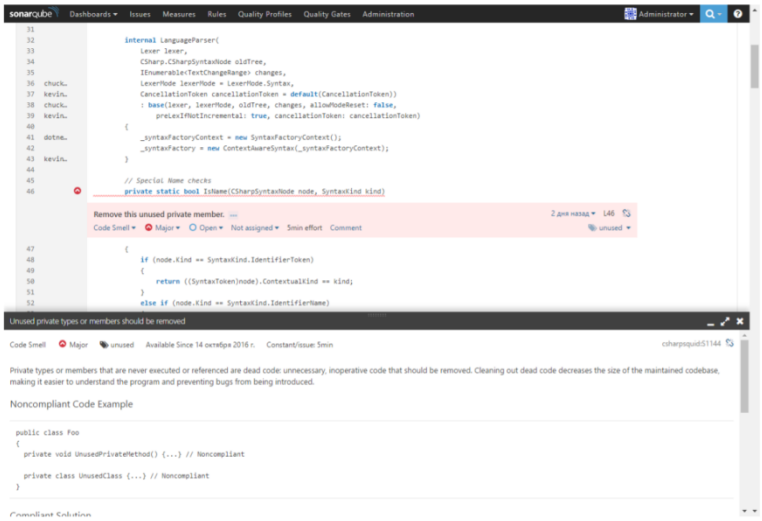
Страница Maintainability (Поддерживаемость) содержит информацию о техническом долге в проекте:



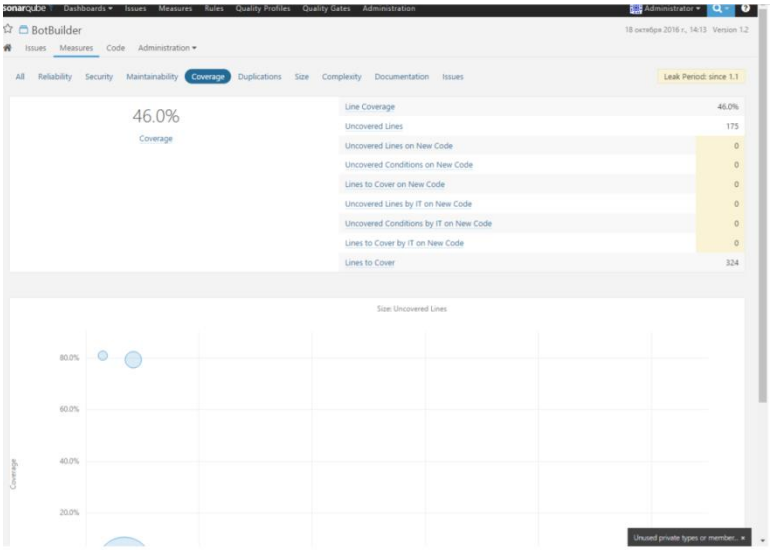
Благодаря навигации «сверху вниз» можно перейти к списку файлов, отсортированных по количеству обнаружений проблемного кода:



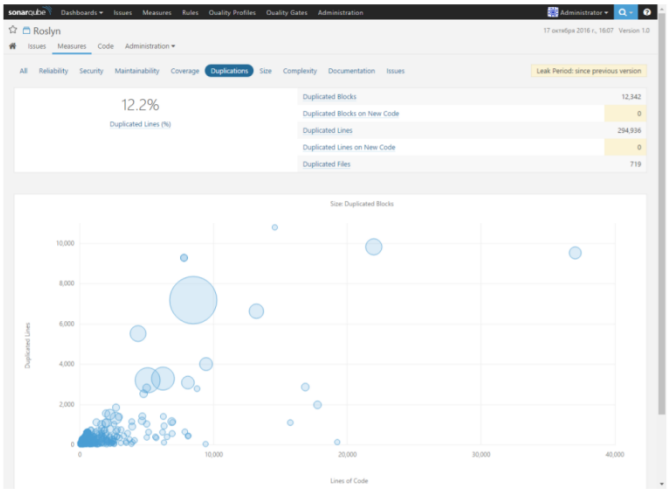
и затем непосредственно к коду, который требует внимания:



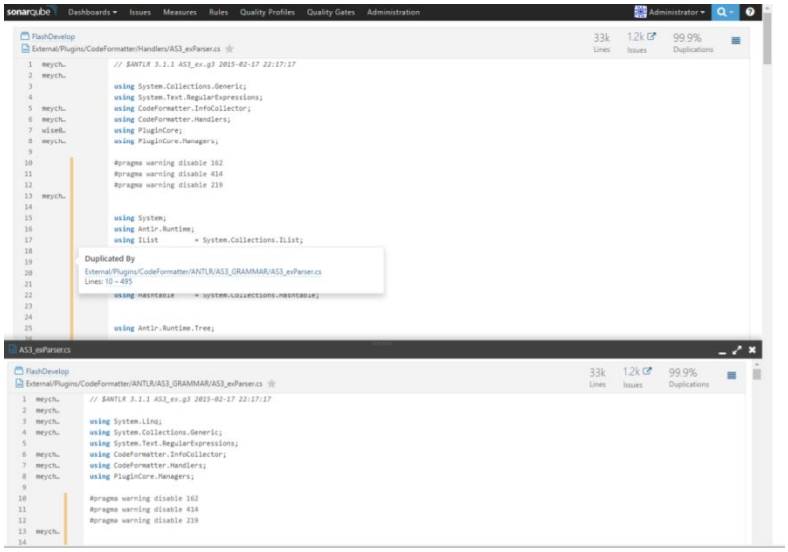
На странице Coverage (Покрытие тестами), представлена информация о покрытии кода тестами:



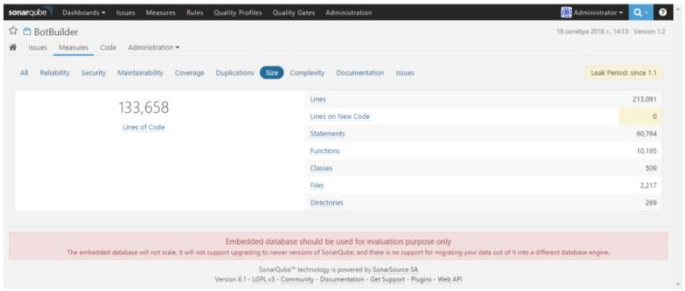
Страница Duplications (Дублирование) содержит информацию о дублировании кода в проекте:



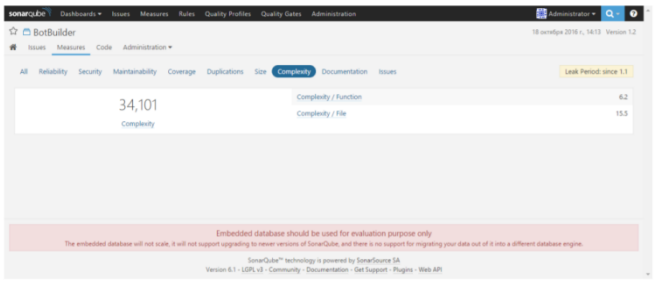
С помощью этой метрики Вы легко можете обнаружить повторяющиеся строки, блоки кода и даже целые файлы:



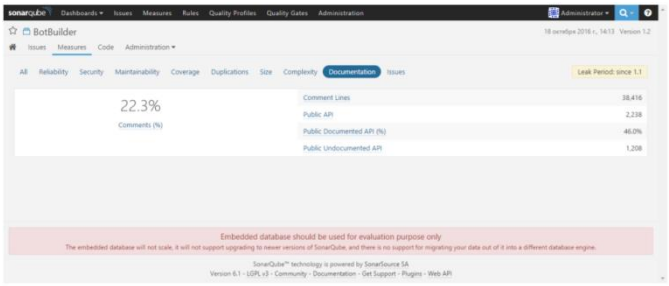
Страница Size (Размер кодовой базы) содержит информацию о размере проекта: количество строк кода, выражений, функций, классов, файлов и директорий:



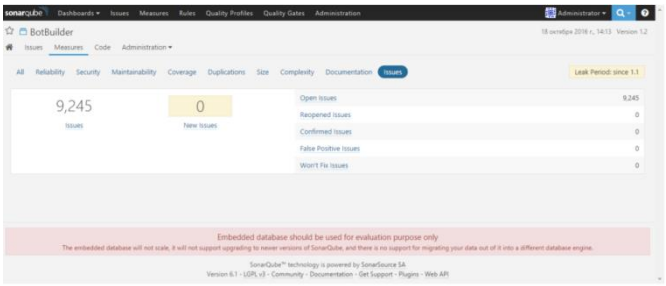
На странице Complexity (Цикломатическая сложность) представлена информация о суммарной цикломатической сложности проекта, а также о средней сложности функций и файлов:



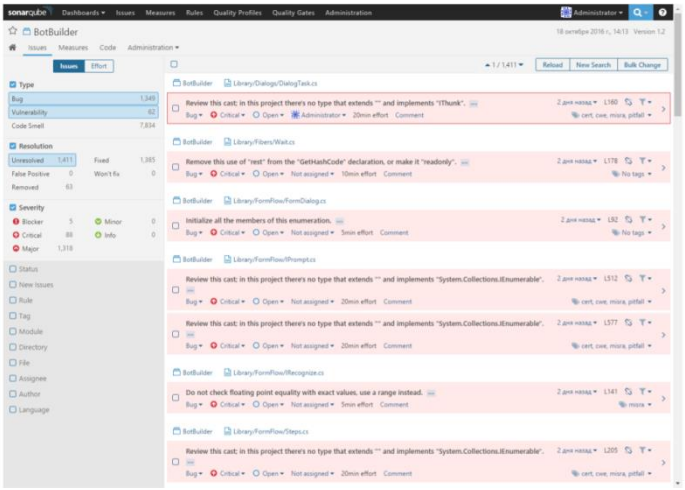
Страница Documentation (Документирование кода) предоставляет информацию о комментариях в коде: отношение строк с комментариями к общему количеству строк в проекте, количество строк с комментариями, количество публичных API и уровень документирования публичных API:



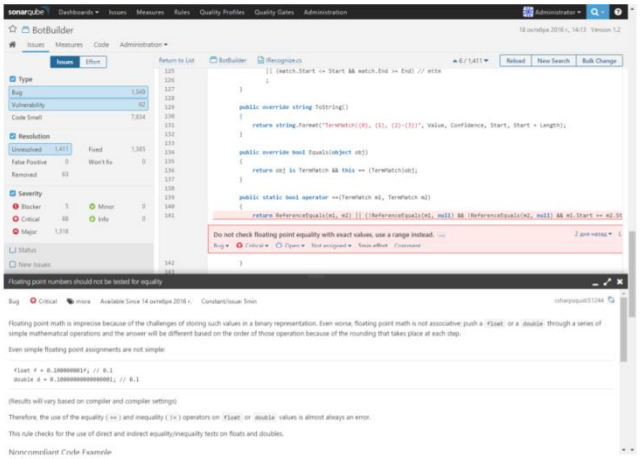
Последняя вкладка в разделе метрик проекта – Issues (Ошибки) – содержит общее количество найденных проблем в коде (сумма количества багов, уязвимостей и code smells), а также распределение проблем по состоянию: открытые, переоткрытые, подтвержденные, ложные срабатывания и won't fix:



Навигация по ошибкам и коду. После анализа метрик кода посмотреть, как SonarQube позволяет работать с найденными проблемами в коде. Для этого перейдем в раздел Issues (Ошибки):

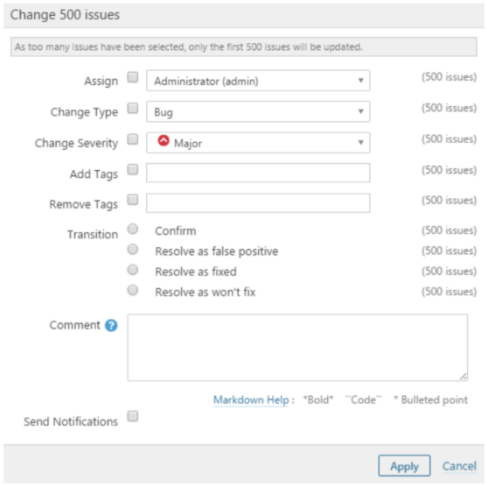


Здесь представлены все найденные проблемы в коде с широкими возможностями фильтрации, что позволяет сфокусироваться на наиболее важных проблемах. Следует отметить, что SonarQube позволяет сохранять настройки фильтров, чтобы повторно их использовать. По двойному клику на сообщении об ошибке вы можете перейти к коду, в котором была найдена проблема. Также доступно детальное описание ошибки и рекомендации, как ее исправить:



Обратите также внимание, что, благодаря интеграции с системами контроля версий, видно, кто и когда внес изменения в код, вызвавшие срабатывание анализатора:

Интеграция с системами контроля версий позволяет также автоматически назначать баги в SonarQube на тех разработчиков, которые их допустили. Также вы можете назначать баги на разработчиков вручную, изменять их тип (bug, vulnerability или code smell), важность, теги, добавлять комментарии. Для большего удобства использования доступна функция массового изменения багов:



Диагностические правила (Rules), профили качества (Quality Profiles) и границы качества (Quality Gates) – ключевые понятия платформы SonarQube. Каждый плагин для SonarQube, осуществляющий статический анализ кода, содержит репозиторий с описанием диагностических правил, которые этот плагин выполняет. Нарушения этих правил используются для определения технического долга в коде и вычисления времени на устранение проблем. Для удобства использования правила объединяются в профили качества (Quality Profiles). По умолчанию, SonarQube создает дефолтный профиль качества для каждого поддерживаемого языка, но Вы можете создавать свои профили качества с тем набором диагностических правил, которые Вам могут быть полезны. Например, для анализа критически важных проектов, требования к качеству кода которых самые строгие, определить профиль качества, содержащий все доступные диагностики, а для менее критичных проектов можно определить менее строгий профиль качества, содержащий только серьезные ошибки, что позволит не отвлекаться на незначительные code smells. Quality Gate – это индикатор соответствия (или несоответствия) кода проекта заданным пороговым значениям метрик. По умолчанию, все проекты, добавленные в SonarQube, используют стандартный quality gate, в котором определены следующие метрики и их пороговые значения:

- Новые баги = 0
- Новые уязвимости = 0
- Коэффициент технического долга на новом коде <= 5%

  Покрытие нового кода >= 80%

1. Проанализировать коды программ, ранее созданных на практических заданиях.
2. Сделать скрин с краткой статистикой: версия сборки, количество строк кода, количество багов, уязвимостей, дата последнего анализа

3. Сделать скрин со следующими метриками кода: надежность, безопасность, поддерживаемость, покрытие тестами, дублирование, размер кодовой базы, цикломатическая сложность, документирование кода и ошибки.

4. Перейти к файлу с исходным кодом и к конкретным участкам кода, в которых обнаружены ошибки

5. Сделать скрин об общем количестве уязвимостей, новых уязвимостях, рейтинге безопасности (также по шкале от A до E), и времени, которое потребуется на устранение уязвимостей

6. Сделать скрин о техническом долге в проекте

7. Сделать скрин о покрытие тестами

8. Сделать скрин о дублировании кода в проекте

Контрольные вопросы:

1. Что такое метрики тестирования?

2. Как измеряется покрытие кода?

3. Какие действия необходимо выполнить, чтобы определить метрики качества кода?

4. Каким образом определяется покрытие кода?

Критерии оценки

оценка «5» ставится, если:

– работа выполнена полностью;

– возможна одна (две) неточность, не являющаяся следствием незнания или не понимания материала.

оценка «4» ставится, если:

– правильно выполнена большая часть работы (свыше 85%), допущено не более трех ошибок;

– работа выполнена полностью, но обоснования шагов решения недостаточны.

оценка «3» ставится, если:

– работа выполнена не полностью, допущено более трех ошибок, но обучающийся владеет основными знаниями, умениями по проверяемой дисциплине.

оценка «2» ставится, если:

– допущены существенные ошибки, показывающие, что обучающийся не владеет обязательными знаниями, умениями по данной дисциплине

ПРАКТИЧЕСКАЯ РАБОТА № 13

Тема: Отладка и тестирование информационных систем

Цель: изучение способов анализа результатов тестирования ПО

Перечень оснащения и оборудования, источников: ПК, раздаточный материал

Время на подготовку и выполнение: 2 часа

Задание: Использования инструментария анализа качества

Порядок работы

1. Повторить теоретический материал
2. Выполнить задание
3. Оформить отчет в текстовом редакторе. Параметры шрифта: Times New Roman, 12 пт. Параметры абзаца междустрочный интервал одинарный. Текст выравнивается по ширине
4. Отчет должен содержать скрины с анализом качества
5. Сдать отчет

№ 1. Проанализировать программу, вычисляющую корни квадратного уравнения

Порядок выполнения

1. Написать программу, вычисления корней квадратного уравнения.
2. Сделать скрин с краткой статистикой: версия сборки, количество строк кода, количество багов, уязвимостей, дата последнего анализа
3. Сделать скрин со следующими метриками кода: надежность, безопасность, поддерживаемость, покрытие тестами, дублирование, размер кодовой базы, цикломатическая сложность, документирование кода и ошибки
4. Перейти к файлу с исходным кодом и к конкретным участкам кода, в которых обнаружены ошибки
5. Сделать скрин об общем количестве уязвимостей, новых уязвимостях, рейтинге безопасности (также по шкале от A до E), и времени, которое потребуется на устранение уязвимостей
6. Сделать скрин о техническом долге в проекте
7. Сделать скрин о покрытие тестами
8. Сделать скрин о дублировании кода в проекте

Контрольные вопросы:

1. Что такое метрики тестирования?
2. Как измеряется покрытие кода?
3. Какие действия необходимо выполнить, чтобы определить метрики качества кода?
4. Каким образом определяется покрытие кода?

Критерии оценки

оценка «5» ставится, если:

– работа выполнена полностью;

– возможна одна (две) неточность, не являющаяся следствием незнания или не понимания материала.

оценка «4» ставится, если:

– правильно выполнена большая часть работы (свыше 85%), допущено не более трех ошибок;

– работа выполнена полностью, но обоснования шагов решения недостаточны.

оценка «3» ставится, если:

– работа выполнена не полностью, допущено более трех ошибок, но обучающийся владеет основными знаниями, умениями по проверяемой дисциплине.

оценка «2» ставится, если:

– допущены существенные ошибки, показывающие, что обучающийся не владеет обязательными знаниями, умениями по данной дисциплине

ПРАКТИЧЕСКАЯ РАБОТА № 14

Тема: Отладка и тестирование информационных систем

Цель: получение навыков анализа и обеспечения обработки исключительных ситуаций.

Перечень оснащения и оборудования, источников: ПК, раздаточный материал

Время на подготовку и выполнение: 2 часа

Задание: Обработка исключительных ситуаций

Порядок работы

1. Повторить теоретический материал
2. Выполнить задание
3. Оформить отчет в текстовом редакторе. Параметры шрифта: Times New Roman, 12 пт. Параметры абзаца междустрочный интервал одинарный. Текст выравнивается по ширине
4. Отчет должен содержать таблицу

Входящие данные	Ожидаемый результат	Фактический результат	Вывод

5. Сдать отчет вместе с проектом, который включает в себя тестируемое приложение и модульный тест.

№ 1. Обработать исключительную ситуацию для программы, вычисляющей деление одного числа на другое. Обработать исключения, возникающие при делении на ноль и при неккоректном вводе числа в консоль. Составить модульный тест.

№ 2. Обработать исключительную ситуацию для программы, вычисляющей длину строки, введенную пользователем, длина строки не должна превышать десять символов. Составить модульный тест.

№ 3. Обработать исключительную ситуацию для программы, которая позволяет пользователям вводить только отрицательные числа. Обработать исключения, возникающие при вводе положительных чисел и при неккоректном вводе числа в консоль. Составить модульный тест.

№ 4. Написать программу, выполняющую основные арифметически операции (сложение, умножение и вычитание), используя обработку исключительных ситуаций (результат вычитания должен быть положительным числом, проверка ввода чисел). Составить модульный тест.

Контрольные вопросы:

1. Что такое исключительная ситуация?
2. Как можно обработать исключительную ситуацию?

Критерии оценки

оценка «5» ставится, если:

– работа выполнена полностью;

– возможна одна (две) неточность, не являющаяся следствием незнания или не понимания материала.

оценка «4» ставится, если:

– правильно выполнена большая часть работы (свыше 85%), допущено не более трех ошибок;

– работа выполнена полностью, но обоснования шагов решения недостаточны.

оценка «3» ставится, если:

– работа выполнена не полностью, допущено более трех ошибок, но обучающийся владеет основными знаниями, умениями по проверяемой дисциплине.

оценка «2» ставится, если:

– допущены существенные ошибки, показывающие, что обучающийся не владеет обязательными знаниями, умениями по данной дисциплине

ПРАКТИЧЕСКАЯ РАБОТА № 15

Тема: Отладка и тестирование информационных систем

Цель: получение навыков анализа и обеспечения обработки исключительных ситуаций.

Перечень оснащения и оборудования, источников: ПК, раздаточный материал

Время на подготовку и выполнение: 2 часа

Задание: Обработка исключительных ситуаций

Порядок работы

1. Повторить теоретический материал

2. Выполнить задание

3. Оформить отчет в текстовом редакторе. Параметры шрифта: Times New Roman, 12 пт. Параметры абзаца междустрочный интервал одинарный. Текст выравнивается по ширине

4. Отчет должен содержать таблицу

Входящие данные	Ожидаемый результат	Фактический результат	Вывод

5. Сдать отчет вместе с проектом, который включает в себя тестируемое приложение и модульный тест

№ 1. Написать программу, в которой обрабатываются следующие исключительные ситуации: "отрицательное значение возраста" и "год рождения больше текущего".

№ 2. Написать программу, вычисляющую площадь прямоугольника, используя обработку исключительных ситуаций (вводиться должны числа, а не символы, стороны прямоугольника не должны принимать отрицательные значения).

№ 3. Даны действительные числа x и y, не равные друг другу. Меньшее из этих двух чисел заменить половиной их суммы, а большее - их удвоенным произведением.

Контрольные вопросы:

1. Что такое исключительная ситуация?

2. Как можно обработать исключительную ситуацию?

Критерии оценки

оценка «5» ставится, если:

– работа выполнена полностью;

– возможна одна (две) неточность, не являющаяся следствием незнания или не понимания материала.

оценка «4» ставится, если:

– правильно выполнена большая часть работы (свыше 85%), допущено не более трех ошибок;

– работа выполнена полностью, но обоснования шагов решения недостаточны.

оценка «3» ставится, если:

– работа выполнена не полностью, допущено более трех ошибок, но обучающийся владеет основными знаниями, умениями по проверяемой дисциплине.

оценка «2» ставится, если:

– допущены существенные ошибки, показывающие, что обучающийся не владеет обязательными знаниями, умениями по данной дисциплине

ПРАКТИЧЕСКАЯ РАБОТА № 16

Тема: Отладка и тестирование информационных систем

Цель: получение навыков проведения функционального тестирования

Перечень оснащения и оборудования, источников: ПК, раздаточный материал

Время на подготовку и выполнение: 2 часа

Задание: Функциональное тестирование

Порядок работы

1. Повторить теоретический материал
2. Написать программу
3. Составить набор тестов
4. Заполнить таблицу

№	Назначение теста	Исходные данные	Ожидаемый результат	Реакция программы	Вывод

5. Оформить отчет в текстовом редакторе. Параметры шрифта: Times New Roman, 12 пт. Параметры абзаца междустрочный интервал одинарный. Текст выравнивается по ширине

№ 1. Написать программу решения квадратного уравнения. Провести тестирование методом черного ящика

№ 2. Разработать программу определения вида треугольника, заданного длинами его сторон: равносторонний, равнобедренный, прямоугольный, разносторонний. Провести тестирование методом черного ящика

Дополнительное задание

Протестировать методом классов эквивалентности с построением дерева разбиения области данных программу, формализующую алгоритм варианта задачи.

Вариант 1

Решить алгебраическое уравнение 2-й степени (квадратное уравнение)

$$a*x^2+b*x+c=0.$$

Вариант 2

Решить алгебраическое уравнение 3-й степени (кубическое уравнение)

$$a*x^3+b*x^2+c*x+d=0.$$

Корни приведенного уравнения рассчитать по формулам Кардано.

Вариант 3

Решить алгебраическое уравнение 3-й степени (кубическое уравнение)

$$a*x^3+b*x^2+c*x+d=0.$$

Корни рассчитать по тригонометрической формуле Виета.

Вариант 4

Решить биквадратное уравнение

$$a*x^4+b*x^2+c=0$$

Вариант 5

В одномерном динамическом массиве, состоящем из n элементов, выполнить поиск элемента по заданному ключу последовательным методом поиска.

Вариант 6

В одномерном динамическом массиве, состоящем из n элементов, выполнить поиск элемента по заданному ключу бинарным методом поиска.

Вариант 7

В одномерном динамическом массиве, состоящем из n элементов, выполнить поиск элемента по заданному ключу интерполяционным методом поиска.

Вариант 8

В одномерном динамическом массиве, состоящем из n элементов, определить количество и индексы элементов массива, больших C.

Вариант 9

В одномерном динамическом массиве, состоящем из n элементов, определить количество и индексы элементов массива, меньших C.

Вариант 10

В одномерном динамическом массиве, состоящем из n элементов, определить количество и индексы элементов массива, больших A и меньших B (A<B).

Вариант 11

В одномерном динамическом массиве, состоящем из n элементов, определить количество и индексы элементов массива, меньших A и больших B (A<B).

Контрольные вопросы:

1. Опишите методы формирования тестовых наборов при использовании стратегии "черного ящика"

3. Эквивалентное разбиение

4. Анализ граничных значений

5. Анализ причинно-следственных связей

Критерии оценки

оценка «5» ставится, если:

– работа выполнена полностью;

– возможна одна (две) неточность, не являющаяся следствием незнания или не понимания материала.

оценка «4» ставится, если:

– правильно выполнена большая часть работы (свыше 85%), допущено не более трех ошибок;

– работа выполнена полностью, но обоснования шагов решения недостаточны.

оценка «3» ставится, если:

– работа выполнена не полностью, допущено более трех ошибок, но обучающийся владеет основными знаниями, умениями по проверяемой дисциплине.

оценка «2» ставится, если:

– допущены существенные ошибки, показывающие, что обучающийся не владеет обязательными знаниями, умениями по данной дисциплине

ПРАКТИЧЕСКАЯ РАБОТА № 17

Тема: Отладка и тестирование информационных систем

Цель: получить практические навыки в разработке программы тестирования методами по стратегии «белого ящика» и «черного ящика»

Перечень оснащения и оборудования, источников: ПК, раздаточный материал

Время на подготовку и выполнение: 2 часа

Задание: Функциональное тестирование

Порядок работы

1. Повторить теоретический материал
2. Написать программу
3. Разработать визуальное приложение, осуществляющее тестирование программы по вариантам задания методами структурного тестирования:

– покрытие операторов;
– покрытие решений (покрытие переходов);
– покрытие условий;
– покрытие условий / решений;
– комбинаторное покрытие условий.

4. Результаты тестирования представить в виде таблиц

5. Оформить отчет в текстовом редакторе. Параметры шрифта: Times New Roman, 12 пт. Параметры абзаца междустрочный интервал одинарный. Текст выравнивается по ширине

6. Оформить отчет по лабораторной работе, включающий разделы:

- Постановка задачи.
- Блок-схема программы.
- Тестовые варианты и результаты тестирования.
- Скриншоты программы.
- Код программы (функции обработчиков событий).
- Выводы.

Вариант 1

Текст задания. Протестировать программу решения квадратного уравнения

Вариант 2

Текст задания. Протестировать программу, вычисляющую функцию, где a, b, c – действительные числа

$$y=\begin{cases} a*x^2+b, при\ x+2<0, \mathrm{b}\neq 0 \\ \dfrac{\mathrm{x-a}}{\mathrm{x-c}}, при\ x+2>0\ и\ c\neq 0 \\ \dfrac{10*x}{c-4}, в\ остальных\ случаях \end{cases}$$

Вариант 3

Текст задания. Протестировать программу, вычисляющую функцию, где a, b, c – действительные числа

$$y=\begin{cases} \dfrac{1}{a*x}-b, при\ x+5<0, \mathrm{c}=0 \\ \dfrac{\mathrm{x-a}}{\mathrm{x}}, при\ x+5>0\ и\ c\neq 0 \\ \dfrac{10*x}{c-4}, в\ остальных\ случаях \end{cases}$$

Вариант 4

Текст задания. Протестировать программу, вычисляющую функцию, где a, b, c – действительные числа

$$y=\begin{cases} a*x^2+b*x+c, при\ a<0, \mathrm{c}\neq 0 \\ \dfrac{\mathrm{-a}}{\mathrm{x-c}}, при\ a>0\ и\ c=0 \\ a*(x+c), в\ остальных\ случаях \end{cases}$$

Вариант 5

Текст задания. Протестировать программу, вычисляющую функцию, где a, b, c – действительные числа

$$y=\begin{cases}-a*x-c, при\ c<0,\ \mathrm{x}\neq 0\\ \dfrac{\mathrm{x}-\mathrm{a}}{-\mathrm{c}}, при\ c>0\ и\ x=0\\ \dfrac{b*x}{c-a}, в\ остальных\ случаях\end{cases}$$

Вариант 6

Текст задания. Протестировать программу, вычисляющую функцию, где a, b, c – действительные числа

$$y=\begin{cases}a-\dfrac{x}{10+b}, при\ x<0,\ \mathrm{b}\neq 0\\ \dfrac{\mathrm{x}-\mathrm{a}}{\mathrm{x}-\mathrm{c}}, при\ x>0\ и\ b=0\\ 3*x+\dfrac{2}{c}, в\ остальных\ случаях\end{cases}$$

Вариант 7

Текст задания. Протестировать программу, вычисляющую функцию, где a, b, c – действительные числа

$$y=\begin{cases}a*x^2+b^2*x, при\ c<0,\ \mathrm{b}\neq 0\\ \dfrac{\mathrm{x}+\mathrm{a}}{\mathrm{x}+\mathrm{c}}, при\ c>0\ и\ b=0\\ \dfrac{x}{c}, в\ остальных\ случаях\end{cases}$$

Вариант 8

Текст задания. Протестировать программу, вычисляющую функцию, где a, b, c – действительные числа

$$y=\begin{cases}-a*x^2-b, при\ x<5,\ \mathrm{c}\neq 0\\ \dfrac{\mathrm{x}-\mathrm{a}}{\mathrm{x}}, при\ x>5\ и\ c=0\\ \dfrac{-x}{c}, в\ остальных\ случаях\end{cases}$$

Вариант 9

Текст задания. Протестировать программу, вычисляющую функцию, где a, b, c – действительные числа

$$y=\begin{cases}-a*x^2, при\ c<0,\ \mathrm{a}\neq 0\\ \dfrac{\mathrm{a}-\mathrm{x}}{\mathrm{c}*\mathrm{x}}, при\ c>0\ и\ a=0\\ \dfrac{x}{c}, в\ остальных\ случаях\end{cases}$$

Вариант 10

Текст задания. Протестировать программу, вычисляющую функцию, где a, b, c – действительные числа

$$y = \begin{cases} a * x^2 - b * x + c, при\ x < 3, \text{b} \neq 0 \\ \dfrac{\text{x - a}}{\text{x - c}}, при\ x > 3\ и\ b = 0 \\ \dfrac{x}{c}, в\ остальных\ случаях \end{cases}$$

Вариант 11

Текст задания. Протестировать программу, вычисляющую функцию, где a, b, c – действительные числа

$$y = \begin{cases} a * x^2 + \dfrac{b}{c}, при\ x < 15, \text{c} \neq 0 \\ \dfrac{\text{x - a}}{(\text{x - c})^2}, при\ x > 15\ и\ c = 0 \\ \dfrac{x^2}{c^2}, в\ остальных\ случаях \end{cases}$$

Вариант 12

Текст задания. Протестировать программу, вычисляющую функцию, где a, b, c – действительные числа

$$y = \begin{cases} a * x^2 + b, при\ x - 1 < 0, \text{b - x} \neq 0 \\ \dfrac{\text{x - a}}{\text{x}}, при\ x - 1 > 0\ и\ b + x = 0 \\ \dfrac{x}{c}, в\ остальных\ случаях \end{cases}$$

Дополнительное задание. Протестировать программу, определяющую вид четырехугольника, заданного координатами вершин на плоскости: квадрат, прямоугольник, параллелограмм, ромб, равнобедренная трапеция, прямоугольная трапеция, трапеция общего вида, четырехугольник общего вида

Контрольные вопросы:

1. Чем отличается метод тестирования белым ящиком от черного ящика?
2. Покрытие операторов
3. Покрытие решений
4. Покрытие условий
5. Покрытие решений/условий
6. Комбинаторное покрытие условий
7. Опишите методы формирования тестовых наборов при использовании стратегии "черного ящика"
8. Эквивалентное разбиение
9. Анализ граничных значений
10. Анализ причинно-следственных связей

Критерии оценки

оценка «5» ставится, если:

– работа выполнена полностью;

– возможна одна (две) неточность, не являющаяся следствием незнания или не понимания материала.

оценка «4» ставится, если:

– правильно выполнена большая часть работы (свыше 85%), допущено не более трех ошибок;

– работа выполнена полностью, но обоснования шагов решения недостаточны.

оценка «3» ставится, если:

– работа выполнена не полностью, допущено более трех ошибок, но обучающийся владеет основными знаниями, умениями по проверяемой дисциплине.

оценка «2» ставится, если:

– допущены существенные ошибки, показывающие, что обучающийся не владеет обязательными знаниями, умениями по данной дисциплине

ПРАКТИЧЕСКАЯ РАБОТА № 18

Тема: Отладка и тестирование информационных систем

Цель: получение навыков тестирования безопасности информационной системы

Перечень оснащения и оборудования, источников: ПК, раздаточный материал

Время на подготовку и выполнение: 2 часа

Задание: Тестирование безопасности

Порядок работы

1. Повторить теоретический материал
2. Выполнить задание
3. Оформить отчет в текстовом редакторе. Параметры шрифта: Times New Roman, 12 пт. Параметры абзаца междустрочный интервал одинарный. Текст выравнивается по ширине
4. Сдать отчет

№ 1. Изучите и опишите одно из средств выявления уязвимостей:

Таблица 1. Обзор средств выявления уязвимостей, работающих на уровне кода

Наименование средства	Назначение	Поддерживаемые языки программирования	Примечание
Иностранные средства выявления уязвимостей			
Its4	Статически просматривает исходный код для обнаружения потенциальных уязвимостей защиты	C/c++	Отмечает вызовы потенциально опасных функций, таких, как strcpy/memcpy, и выполняет поверхностный семантический анализ, пытаясь оценить, насколько опасен такой код, а также дает советы по его улучшению
Rats(rough auditing tool for security)	Просматривает исходный текст, находя потенциально опасные	C/c++, php, perl, python	Использует сочетание проверок надежности защиты от семантических проверок в its4 до

Наименование средства	Назначение	Поддерживаемые языки программирования	Примечание
	обращения к функциям		глубокого семантического анализа в поисках дефектов, способных привести к переполнению буфера, полученных из mops
Flawfinder	Просматривает исходный текст, находя потенциально опасные обращения к функциям	C/c++	Выполняет поиск функций, которые чаще всего используются некорректно, присваивает им коэффициенты риска (опираясь на такую информацию, как передаваемые параметры) и составляет список потенциально уязвимых мест, упорядочивая их по степени риска
Flexelint (pc-lint)	Производит семантический анализ исходного кода, анализ потоков данных и управления	C/c++	В конце работы выдаются сообщения нескольких основных типов: – возможен нулевой указатель – проблемы с выделением памяти (например, нет free() после malloc()) – проблемный поток управления (например, недостижимый код); – возможно переполнение буфера, арифметическое переполнение; – предупреждения о плохом и потенциально опасном стиле кода
Parasoft c++ test	Формирование тестов анализа уязвимостей на уровне метода, класса, файла и проекта	C++	Генерирует тестовый код, вызывая для его подготовки компилятор visual c++
Coverity	Используется для выявления и исправления дефектов безопасности и качества в приложениях критического назначения	C/c++, java	Способен с минимальной положительной погрешностью обрабатывать десятки миллионов строк кода, обеспечивая 100-процентное покрытие трассы
Codesurfer	Может применяться для поиска ошибок в исходном коде, для улучшения понимания исходного кода	C/c++	Позволяет проводить анализ указателей, использовать и определять переменные, зависимости данных, строить графы вызовов
Fxcop	Способен обнаружить более 200 недочетов (или ошибок) в следующих областях: – архитектура библиотеки; – правила именования;	C/c++	Откомпилированный код проверяется с помощью механизмов рефлексии, парсинга msil и анализа графа вызовов
Российские средства выявления уязвимостей			
Ак-вс	Автоматизированный анализ исходных текстов, с целью выявления потенциально опасных сигнатур	C/c++, java, pascal, c#, php, assembler	Позволяет проводить статический анализ исходных текстов, динамический анализ, имеет базы сигнатур для каждого из поддерживаемых языков программирования
Аист-с	Автоматизированный анализ исходных текстов	C/c++	Позволяет проводить статический анализ исходных текстов
Ксаит	Автоматизированный анализ исходных текстов	C/c++	Позволяет проводить статический анализ исходных текстов
Uca	Предназначено для выявления потенциально опасных сигнатур	C/c++, pascal, perl, plm	Имеет базы сигнатур для каждого из поддерживаемых языков программирования
Viva64	Помогает отслеживать в исходном коде потенциально опасные фрагменты, связанные с переходом от 32-битных систем к 64-битным	C/c++	Помогает писать корректный и оптимизированный код для 64-битных систем

№ 2. Протестировать функцию, которая будем проверять сложность пароля по следующим правилам:

- количество символов от 8 до 20
- наличие цифр
- наличие спецсимволов
- наличие прописных и строчных букв

Создать таблицу тестовые данные

требование	пароль	результат
Не должно быть пустое поле	Пустое поле	Не выполнено

Контрольные вопросы:

1. Перечислите основные атаки на код.
2. Что такое внедрение кода?
3. Какие основные причины появления уязвимостей в коде?

Критерии оценки

оценка «5» ставится, если:

– работа выполнена полностью;

– возможна одна (две) неточность, не являющаяся следствием незнания или не понимания материала.

оценка «4» ставится, если:

– правильно выполнена большая часть работы (свыше 85%), допущено не более трех ошибок;

– работа выполнена полностью, но обоснования шагов решения недостаточны.

оценка «3» ставится, если:

– работа выполнена не полностью, допущено более трех ошибок, но обучающийся владеет основными знаниями, умениями по проверяемой дисциплине.

оценка «2» ставится, если:

– допущены существенные ошибки, показывающие, что обучающийся не владеет обязательными знаниями, умениями по данной дисциплине

ПРАКТИЧЕСКАЯ РАБОТА № 19

Тема: Отладка и тестирование информационных систем

Цель: получение навыков тестирования безопасности информационной системы

Перечень оснащения и оборудования, источников: ПК, раздаточный материал

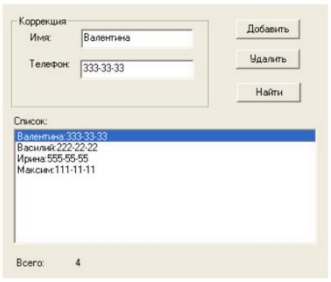
Время на подготовку и выполнение: 2 часа

Задание: Тестирование безопасности

Порядок работы

1. Повторить теоретический материал
2. Выполнить задание
3. Оформить отчет в текстовом редакторе. Параметры шрифта: Times New Roman, 12 пт. Параметры абзаца междустрочный интервал одинарный. Текст выравнивается по ширине
4. Сдать отчет вместе с проектом

№ 1. Разработать приложение, интерфейс которого представлен на рисунке



№ 2. Добавить в программу форму авторизазации по имени и паролю

Контрольные вопросы:

1. Перечислите основные атаки на код.
2. Что такое внедрение кода?
3. Какие основные причины появления уязвимостей в коде?

Критерии оценки

оценка «5» ставится, если:

– работа выполнена полностью;

– возможна одна (две) неточность, не являющаяся следствием незнания или не понимания материала.

оценка «4» ставится, если:

– правильно выполнена большая часть работы (свыше 85%), допущено не более трех ошибок;

– работа выполнена полностью, но обоснования шагов решения недостаточны.

оценка «3» ставится, если:

– работа выполнена не полностью, допущено более трех ошибок, но обучающийся владеет основными знаниями, умениями по проверяемой дисциплине.

оценка «2» ставится, если:

– допущены существенные ошибки, показывающие, что обучающийся не владеет обязательными знаниями, умениями по данной дисциплине

ПРАКТИЧЕСКАЯ РАБОТА № 20

Тема: Отладка и тестирование информационных систем

Цель: получение навыков проведения нагрузочного и стрессового тестирования

Перечень оснащения и оборудования, источников: ПК, раздаточный материал

Время на подготовку и выполнение: 2 часа

Задание: Нагрузочное тестирование, стрессовое тестирование

Порядок работы

1. Повторить теоретический материал

2. Выполнить задание

3. Оформить отчет в текстовом редакторе. Параметры шрифта: Times New Roman, 12 пт. Параметры абзаца междустрочный интервал одинарный. Текст выравнивается по ширине

4. Сдать отчет

№ 1. Открыть приложение ASP.NET Core «Прогноз погоды»

№ 2. Разработать тестовый сценарий нагрузочного тестирования.

Ответить на вопрос – сколько запросов в секунду может обработать приложение при условии, что они идут последовательно.

Построить график зависимости времени ответа от количества параллельных запросов

Ответить на вопрос – какое максимальное количество параллельных запросов может обработать приложение без сбоев.

Провести тест при использовании максимального числа запросов.

Контрольные вопросы:

1. Как создать тест для нагрузочного тестирования?

2. Как анализировать данные нагрузочного тестирования?

3. Как запустить тест в облаке Azure?

Критерии оценки

оценка «5» ставится, если:

– работа выполнена полностью;

– возможна одна (две) неточность, не являющаяся следствием незнания или не понимания материала.

оценка «4» ставится, если:

– правильно выполнена большая часть работы (свыше 85%), допущено не более трех ошибок;

– работа выполнена полностью, но обоснования шагов решения недостаточны.

оценка «3» ставится, если:

– работа выполнена не полностью, допущено более трех ошибок, но обучающийся владеет основными знаниями, умениями по проверяемой дисциплине.

оценка «2» ставится, если:

– допущены существенные ошибки, показывающие, что обучающийся не владеет обязательными знаниями, умениями по данной дисциплине

ПРАКТИЧЕСКАЯ РАБОТА № 21

Тема: Отладка и тестирование информационных систем

Цель: получение навыков проведения нагрузочного и стрессового тестирования

Перечень оснащения и оборудования, источников: ПК, раздаточный материал

Время на подготовку и выполнение: 2 часа

Задание: Нагрузочное тестирование, стрессовое тестирование

Порядок работы

1. Повторить теоретический материал
2. Выполнить задание
3. Оформить отчет в текстовом редакторе. Параметры шрифта: Times New Roman, 12 пт. Параметры абзаца междустрочный интервал одинарный. Текст выравнивается по ширине
4. Сдать отчет

№ 1. Разработать Компилятор простых арифметических выражений, например

2+ (-5)*(7-8)

Вход и выход осуществляются в виде строк.

№ 2. Разработать тестовый сценарий нагрузочного тестирования.

Ответить на вопрос – сколько запросов в секунду может обработать приложение при условии, что они идут последовательно. Построить график зависимости времени ответа от количества параллельных запросов (рассматривать логарифмическую шкалу по основанию два, т.е. 1, 2,4,8,16,32 и т.д. запроса)

Ответить на вопрос – какое максимальное количество параллельных запросов может обработать приложение без сбоев.

Контрольные вопросы:

1. Как создать тест для нагрузочного тестирования?
2. Как анализировать данные нагрузочного тестирования?
3. Как запустить тест в облаке Azure?

Критерии оценки

оценка «5» ставится, если:

– работа выполнена полностью;

– возможна одна (две) неточность, не являющаяся следствием незнания или не понимания материала.

оценка «4» ставится, если:

– правильно выполнена большая часть работы (свыше 85%), допущено не более трех ошибок;

– работа выполнена полностью, но обоснования шагов решения недостаточны.

оценка «3» ставится, если:

– работа выполнена не полностью, допущено более трех ошибок, но обучающийся владеет основными знаниями, умениями по проверяемой дисциплине.

оценка «2» ставится, если:

– допущены существенные ошибки, показывающие, что обучающийся не владеет обязательными знаниями, умениями по данной дисциплине

ПРАКТИЧЕСКАЯ РАБОТА № 22

Тема: Отладка и тестирование информационных систем

Цель: получение навыков тестирования интеграции

Перечень оснащения и оборудования, источников: ПК, раздаточный материал

Время на подготовку и выполнение: 2 часа

Задание: Тестирование интеграции

Порядок работы

1. Повторить теоретический материал

2. Разработать приложение, осуществляющее тестирование программы методом нисходящего тестирования:

– построение модулей-заглушек;
– тестирование верхнего головного модуля программы;
– тестирование модулей программы;
– представление результатов тестирования.

3. Оформить отчет по лабораторной работе, включающий разделы:

– Постановка задачи.
– Блок-схема программы.
– Структурная схема комплекса программы
– Таблицы результатов тестирования.
– Скриншоты программы.
– Код программы.
– Выводы.

4. Оформить отчет в текстовом редакторе. Параметры шрифта: Times New Roman, 12 пт. Параметры абзаца междустрочный интервал одинарный. Текст выравнивается по ширине

Вариант 1

В одномерном массиве, состоящем из п вещественных элементов, вычислить:

1) сумму отрицательных элементов массива;

2) произведение элементов массива, расположенных между максимальным и минимальным элементами.

Упорядочить элементы массива по возрастанию.

Вариант 2

В одномерном массиве, состоящем из п вещественных элементов, вычислить:

1) сумму положительных элементов массива;

2) произведение элементов массива, расположенных между максимальным по модулю и минимальным по модулю элементами.

Упорядочить элементы массива по убыванию.

Вариант 3

В одномерном массиве, состоящем из п целых элементов, вычислить:

1) произведение элементов массива с четными номерами;

2) сумму элементов массива, расположенных между первым и последним нулевыми элементами.

Преобразовать массив таким образом, чтобы сначала располагались все положительные элементы, а потом – все отрицательные (элементы, равные нулю, считать положительными).

Вариант 4

В одномерном массиве, состоящем из п вещественных элементов, вычислить:

1) сумму элементов массива с нечетными номерами;

2) сумму элементов массива, расположенных между первым и последним отрицательными элементами.

Сжать массив, удалив из него все элементы, модуль которых не превышает 1. Освободившиеся в конце массива элементы заполнить нулями.

Вариант 5

В одномерном массиве, состоящем из п вещественных элементов, вычислить:

1) максимальный элемент массива;

2) сумму элементов массива, расположенных до последнего положительного элемента.

Сжать массив, удалив из него все элементы, модуль которых находится в интервале [a,b]. Освободившиеся в конце массива элементы заполнить нулями.

Вариант 6

В одномерном массиве, состоящем из п вещественных элементов, вычислить:

1) минимальный элемент массива;

2) сумму элементов массива, расположенных между первым и последним положительными элементами.

Преобразовать массив таким образом, чтобы сначала располагались все элементы, равные нулю, а потом – все остальные.

Вариант 7

В одномерном массиве, состоящем из п целых элементов, вычислить:

1) номер максимального элемента массива;

2) произведение элементов массива, расположенных между первым и вторым нулевыми элементами.

Преобразовать массив таким образом, чтобы в первой его половине располагались элементы, стоявшие в нечетных позициях, а во второй – элементы, стоявшие в четных позициях.

Вариант 8

В одномерном массиве, состоящем из п вещественных элементов, вычислить:

1) номер минимального элемента массива;

2) сумму элементов массива, расположенных между первым и вторым отрицательными элементами.

Преобразовать массив таким образом, чтобы сначала располагались все элементы, модуль которых не превышает 1, а потом – все остальные.

Вариант 9

В одномерном массиве, состоящем из п вещественных элементов, вычислить:

1) максимальный по модулю элемент массива;

2) сумму элементов массива, расположенных между первым и вторым положительными элементами.

Преобразовать массив таким образом, чтобы элементы, равные нулю, располагались после всех остальных.

Вариант 10

В одномерном массиве, состоящем из п целых элементов, вычислить:

1) минимальный по модулю элемент массива;

2) сумму модулей элементов массива, расположенных после первого элемента, равного нулю.

Преобразовать массив таким образом, чтобы в первой его половине располагались элементы, стоявшие в четных позициях, а во второй половине – элементы, стоявшие в нечетных позициях.

Вариант 11

В одномерном массиве, состоящем из п вещественных элементов, вычислить:

1) номер минимального по модулю элемента массива;

2) сумму модулей элементов массива, расположенных после первого отрицательного элемента.

Сжать массив, удалив из него все элементы, величина которых находится в интервале [a,b]. Освободившиеся в конце массива элементы заполнить нулями.

Вариант 12

В одномерном массиве, состоящем из п вещественных элементов, вычислить:

1) номер максимального по модулю элемента массива;

2) сумму элементов массива, расположенных после первого положительного элемента.

Преобразовать массив таким образом, чтобы сначала располагались все элементы, целая часть которых лежит в интервале [a,b], а потом – все остальные.

Контрольные вопросы:

1. Что называется интеграционным тестированием?
2. Что такое модуль-заглушка?
3. Какие виды модулей-заглушек?
4. Какие этапы алгоритма метода нисходящего тестирования?
5. Какие преимущества и недостатки метода нисходящего тестирования?

Критерии оценки

оценка «5» ставится, если:

– работа выполнена полностью;

– возможна одна (две) неточность, не являющаяся следствием незнания или не понимания материала.

оценка «4» ставится, если:

– правильно выполнена большая часть работы (свыше 85%), допущено не более трех ошибок;

– работа выполнена полностью, но обоснования шагов решения недостаточны.

оценка «3» ставится, если:

– работа выполнена не полностью, допущено более трех ошибок, но обучающийся владеет основными знаниями, умениями по проверяемой дисциплине.

оценка «2» ставится, если:

– допущены существенные ошибки, показывающие, что обучающийся не владеет обязательными знаниями, умениями по данной дисциплине

ПРАКТИЧЕСКАЯ РАБОТА № 23

Тема: Отладка и тестирование информационных систем

Цель: получить практические навыки в разработке программы тестирования методом восходящего тестирования

Перечень оснащения и оборудования, источников: ПК, раздаточный материал

Время на подготовку и выполнение: 2 часа

Задание: Тестирование интеграции

Порядок работы

1. Разработать приложение, осуществляющее тестирование программы методом восходящего тестирования:

– построение модулей-драйверов;
– тестирование терминальных модулей;
– тестирование кластеров программы;
– представление результатов тестирования.

2. Оформить отчет по лабораторной работе, включающий разделы:

– Постановка задачи.
– Блок-схема программы.
– Структурная схема комплекса программы
– Таблицы результатов тестирования.
– Скриншоты программы.
– Код программы.
– Выводы.

3. Оформить отчет в текстовом редакторе. Параметры шрифта: Times New Roman, 12 пт. Параметры абзаца междустрочный интервал одинарный. Текст выравнивается по ширине

Вариант 1

В одномерном динамическом массиве, состоящем из n вещественных элементов, вычислить:

1) количество элементов массива, меньших C;

2) сумму целых частей элементов массива, расположенных после последнего отрицательного элемента.

Преобразовать массив таким образом, чтобы сначала располагались все элементы, отличающиеся от максимального не более чем на 20%, а потом – все остальные.

Вариант 2

В одномерном динамическом массиве, состоящем из n вещественных элементов, вычислить:

1) произведение отрицательных элементов массива;

2) сумму положительных элементов массива, расположенных до максимального элемента.

Изменить порядок следования элементов в массиве на обратный.

Вариант 3

В одномерном динамическом массиве, состоящем из n вещественных элементов, вычислить:

1) произведение положительных элементов массива;

2) сумму элементов массива, расположенных до минимального элемента.

Упорядочить по возрастанию отдельно элементы, стоящие на четных местах, и элементы, стоящие на нечетных местах.

Вариант 4

В одномерном динамическом массиве, состоящем из n вещественных элементов, вычислить:

1) среднее арифметическое элементов, расположенных до первого и после последнего нулевых значений;

2) поменять местами первый и максимальный элементы, последний и минимальный элементы.

Упорядочить по убыванию отдельно элементы, стоящие на четных местах, и элементы, стоящие на нечетных местах.

Вариант 5

В одномерном динамическом массиве, состоящем из n целых элементов, вычислить:

1) произведение элементов, находящихся между минимальным и максимальным элементами массива;

2) удалить из массива простое число. Упорядочить по возрастанию элементы массива.

Вариант 6

В одномерном динамическом массиве, состоящем из n вещественных элементов, вычислить:

1) произведение отрицательных элементов массива;

2) сумму элементов массива, расположенных до максимального элемента.

Упорядочить по возрастанию отдельно элементы, стоящие на четных местах, и элементы, стоящие на нечетных местах.

Вариант 7

В одномерном динамическом массиве, состоящем из n вещественных элементов, вычислить:

1) среднее значение элементов, расположенных в массиве между первым последним нулевыми элементами;

2) поменять местами максимальный и минимальный элементы.

Упорядочить по возрастанию отдельно элементы, стоящие на четных местах, и элементы, стоящие на нечетных местах.

Вариант 8

В одномерном динамическом массиве, состоящем из n целых элементов, определить:

1) количество совершенных чисел;

2) удалить из массива все нулевые элементы.

Упорядочить по убыванию отдельно элементы, стоящие на четных местах, и элементы, стоящие на нечетных местах.

Вариант 9

В одномерном динамическом массиве, состоящем из n вещественных элементов, вычислить:

1) сумму отрицательных элементов массива;

2) произведение элементов массива, расположенных между максимальным и минимальным элементами. Упорядочить элементы массива по возрастанию.

Вариант 10

В одномерном динамическом массиве, состоящем из n вещественных элементов, вычислить:

1) сумму положительных элементов массива;

2) произведение элементов массива, расположенных между максимальным по модулю и минимальным по модулю элементами.

Упорядочить элементы массива по убыванию.

Вариант 11

В одномерном динамическом массиве, состоящем из n целых элементов, вычислить:

1) произведение элементов массива с четными номерами;

2) сумму элементов массива, расположенных между первым и последним нулевыми элементами.

Преобразовать массив таким образом, чтобы сначала располагались все положительные элементы, а потом – все отрицательные (элементы, равные нулю, считать положительными).

Вариант 12

В одномерном динамическом массиве, состоящем из п вещественных элементов, вычислить:

1) сумму элементов массива с нечетными номерами;

2) сумму элементов массива, расположенных между первым и последним отрицательными элементами.

Сжать массив, удалив из него все элементы, модуль которых не превышает 1. Освободившиеся в конце массива элементы заполнить нулями.

Контрольные вопросы:

1. Что такое интеграционное тестирование?
2. Какие виды интеграционного тестирования существуют?
3. Что такое «заглушка»?

Критерии оценки

оценка «5» ставится, если:

– работа выполнена полностью;

– возможна одна (две) неточность, не являющаяся следствием незнания или не понимания материала.

оценка «4» ставится, если:

– правильно выполнена большая часть работы (свыше 85%), допущено не более трех ошибок;

– работа выполнена полностью, но обоснования шагов решения недостаточны.

оценка «3» ставится, если:

– работа выполнена не полностью, допущено более трех ошибок, но обучающийся владеет основными знаниями, умениями по проверяемой дисциплине.

оценка «2» ставится, если:

– допущены существенные ошибки, показывающие, что обучающийся не владеет обязательными знаниями, умениями по данной дисциплине

ПРАКТИЧЕСКАЯ РАБОТА № 24

Тема: Отладка и тестирование информационных систем

Цель: получение навыков проведения конфигурационного тестирования

Перечень оснащения и оборудования, источников: ПК, раздаточный материал

Время на подготовку и выполнение: 2 часа

Задание: Конфигурационное тестирование

Порядок работы

1. Повторить теоретический материал

2. Выполнить задание

3. Оформить отчет в текстовом редакторе. Параметры шрифта: Times New Roman, 12 пт. Параметры абзаца междустрочный интервал одинарный. Текст выравнивается по ширине

Текст задания. Напишите универсальную программу тестирования. Тест, последовательность вопросов и варианты ответов должны находиться в текстовом файле. Имя файла теста программа должна получать из командной строки запуска программы. Количество вопросов теста неограниченно. Вместе с тем, предлагается ввести следующее ограничение: текст вопроса и альтернативных ответов не должен занимать более одной строки экрана. Программа должна выставлять оценку по следующему правилу:

– ОТЛИЧНО — за правильные ответы на все вопросы,
– ХОРОШО — если испытуемый правильно ответил не менее чем на 80% вопросов,
– УДОВЛЕТВОРИТЕЛЬНО — если правильных ответов более 60%,
– ПЛОХО — если правильных ответов меньше 60%.

Ниже приведена рекомендуемая структура файла вопросов теста (N_i — количество альтернативных ответов к i-ому вопросу, K_i — номер правильного ответа), пример файла теста и вид экрана во время работы программы (номера ответов, введенные пользователем, выделены полужирным шрифтом).

Вопрос$_1$

N_1 M_1

Ответ

…

Ответ

Вопрос$_2$

N_2 M_2

Ответ

…

Ответ

Вопрос$_к$

N_k $M_к$

Ответ

…

Ответ

Архитектор Исаакиевского собора

3 2

Доменико Трезини

Огюст Монферран

Карл Росси

Архитектор Зимнего дворца

2 2

Франческо Бартоломее

Огюст Монферран

Невский проспект получил свое название

3 2

По имени реки, на которой стоит Санкт-Петербург.

По имени близко расположенного монастыря,

Александро-Невской лавры.

в память о знаменитом полководце — Александре Невском.

Сейчас Вам будет предложен тест. К каждому вопросу дается несколько вариантов ответов.

Вы должны ввести номер правильного ответа и нажать клавишу

Архитектор Исаакиевского собора:

1. Доменико Трезини

2. Огюст Монферран

3. Карл Росси

-> **2**

Архитектор Зимнего дворца:

1. Франческо Бартоломее

2. Карл Росси

-> **2**

Невский проспект получил свое название:

1. По имени реки, на которой стоит Санкт-Петербург.

2. По имени близко расположенного монастыря, Александро-Невской лавры.

3.в память о знаменитом полководце — Александре Невском.

->**2**

Ваша оценка ОТЛИЧНО!

Описать и обосновать итоги тестирования работы разработанного приложения на различных платформах: различных вариантах аппаратной конфигурации, версиях операционной системы и окружения.

Контрольные вопросы:

1. Что такое конфигурационное тестирование?

2. Какие основные цели преследует конфигурационное тестирование?

3. Что такое матрица покрытия?

Критерии оценки

оценка «5» ставится, если:

– работа выполнена полностью;

– возможна одна (две) неточность, не являющаяся следствием незнания или не понимания материала.

оценка «4» ставится, если:

– правильно выполнена большая часть работы (свыше 85%), допущено не более трех ошибок;

– работа выполнена полностью, но обоснования шагов решения недостаточны.

оценка «3» ставится, если:

– работа выполнена не полностью, допущено более трех ошибок, но обучающийся владеет основными знаниями, умениями по проверяемой дисциплине.

оценка «2» ставится, если:

– допущены существенные ошибки, показывающие, что обучающийся не владеет обязательными знаниями, умениями по данной дисциплине

ПРАКТИЧЕСКАЯ РАБОТА № 25

Тема: Отладка и тестирование информационных систем

Цель: получение навыков проведения конфигурационного тестирования

Перечень оснащения и оборудования, источников: ПК, раздаточный материал

Время на подготовку и выполнение: 2 часа

Задание: Конфигурационное тестирование

Порядок работы

1. Повторить теоретический материал
2. Выполнить задание
3. Оформить отчет в текстовом редакторе. Параметры шрифта: Times New Roman, 12 пт. Параметры абзаца междустрочный интервал одинарный. Текст выравнивается по ширине

Текст задания. Дана структура с именем ZNAK, состоящая из полей:

– фамилия, имя;

– знак Зодиака;

– дата рождения (массив из трех чисел).

Написать программу, которая выполняет следующие действия:

– ввод с клавиатуры данных в массив, состоящий из 8 элементов типа ZNAK, и занесение их в файл данных;

– чтение данных из файла и вывод их на экран;

– вывод на экран информации о людях, родившихся в месяц, значение которого введено с клавиатуры (если таких нет – вывести об этом сообщение);

– список должен быть упорядочен по знакам Зодиака.

Описать и обосновать итоги тестирования работы разработанного приложения на различных платформах: различных вариантах аппаратной конфигурации, версиях операционной системы и окружения.

Контрольные вопросы:

1. Конфигурационное тестирование
2. Особенности конфигурационного тестирования

Критерии оценки

оценка «5» ставится, если:

– работа выполнена полностью;

– возможна одна (две) неточность, не являющаяся следствием незнания или не понимания материала.

оценка «4» ставится, если:

– правильно выполнена большая часть работы (свыше 85%), допущено не более трех ошибок;

– работа выполнена полностью, но обоснования шагов решения недостаточны.

оценка «3» ставится, если:

– работа выполнена не полностью, допущено более трех ошибок, но обучающийся владеет основными знаниями, умениями по проверяемой дисциплине.

оценка «2» ставится, если:

– допущены существенные ошибки, показывающие, что обучающийся не владеет обязательными знаниями, умениями по данной дисциплине

ПРАКТИЧЕСКАЯ РАБОТА № 26

Тема: Отладка и тестирование информационных систем

Цель: получение навыков тестирования установки

Перечень оснащения и оборудования, источников: ПК, раздаточный материал

Время на подготовку и выполнение: 2 часа

Задание: Тестирование установки

Порядок работы

1. Повторить теоретический материал
2. Выполнить задание

3. Оформить отчет в текстовом редакторе. Параметры шрифта: Times New Roman, 12 пт. Параметры абзаца междустрочный интервал одинарный. Текст выравнивается по ширине

Текст задания. Разработать приложение, интерфейс которого представлен на рисунке

Провести комплексное тестирование разработанного приложения.

Оформить отчет.

Контрольные вопросы:

1. Что такое тестирование инсталляции?
2. Какие этапы содержит тестирование инсталляции?
3. Как тестировать установщик?

Критерии оценки

оценка «5» ставится, если:

– работа выполнена полностью;

– возможна одна (две) неточность, не являющаяся следствием незнания или не понимания материала.

оценка «4» ставится, если:

– правильно выполнена большая часть работы (свыше 85%), допущено не более трех ошибок;

– работа выполнена полностью, но обоснования шагов решения недостаточны.

оценка «3» ставится, если:

– работа выполнена не полностью, допущено более трех ошибок, но обучающийся владеет основными знаниями, умениями по проверяемой дисциплине.

оценка «2» ставится, если:

– допущены существенные ошибки, показывающие, что обучающийся не владеет обязательными знаниями, умениями по данной дисциплине

ПРАКТИЧЕСКАЯ РАБОТА № 27

Тема: Отладка и тестирование информационных систем

Цель: получение навыков тестирования установки

Перечень оснащения и оборудования, источников: ПК, раздаточный материал

Время на подготовку и выполнение: 2 часа

Задание: Тестирование установки

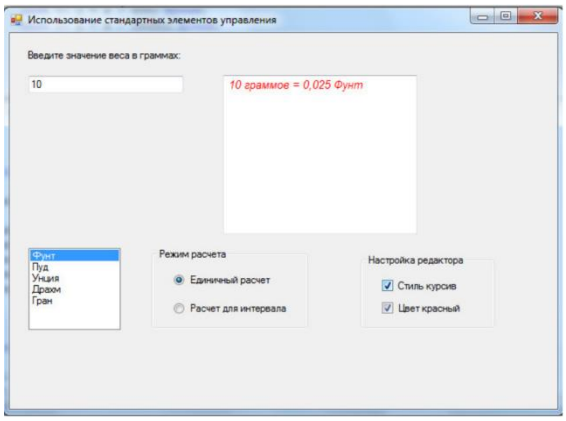
Порядок работы

1. Повторить теоретический материал

2. Выполнить задание

3. Оформить отчет в текстовом редакторе. Параметры шрифта: Times New Roman, 12 пт. Параметры абзаца междустрочный интервал одинарный. Текст выравнивается по ширине

Текст задания. Разработать приложение, интерфейс которого представлен на рисунке



Провести комплексное тестирование разработанного приложения.

Оформить отчет.

Контрольные вопросы:

1. Что такое тестирование инсталляции?

2. Какие этапы содержит тестирование инсталляции?

3. Как тестировать установщик?

Критерии оценки

оценка «5» ставится, если:

– работа выполнена полностью;

– возможна одна (две) неточность, не являющаяся следствием незнания или не понимания материала.

оценка «4» ставится, если:

– правильно выполнена большая часть работы (свыше 85%), допущено не более трех ошибок;

– работа выполнена полностью, но обоснования шагов решения недостаточны.

оценка «3» ставится, если:

– работа выполнена не полностью, допущено более трех ошибок, но обучающийся владеет основными знаниями, умениями по проверяемой дисциплине.

оценка «2» ставится, если:

– допущены существенные ошибки, показывающие, что обучающийся не владеет обязательными знаниями, умениями по данной дисциплине

СПИСОК РЕКОМЕНДОВАННЫХ ИСТОЧНИКОВ:

Основные источники:

1. Федорова, Г.И. Разработка, внедрение и адаптация программного обеспечения отраслевой направленности [Текст]: учебное пособие / Г.И. Федорова. – М.: КУРС, Инфра-М, 2016. - 336 с.

Дополнительные источники:

1. Васильев, Р. Стратегическое управление информационными системами [Текст]: учебник / Р. Васильев, Г. Калянов, Г. Левочкина, О. Лукинова. – М.: Бином. Лаборатория знаний, Интернет-университет информационных технологи, 2014. – 512 с.