Государственное бюджетное профессиональное образовательное учреждение

Самарской области

«Тольяттинский социально-экономический колледж»

**КУРСОВАЯ РАБОТА**

**ТЕСТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ «УЧЕТ ОБМЕНА ВАЛЮТАМИ»**

**ПМ.05 «ПРОЕКТИРОВАНИЕ И разработкА ИНФОРМАЦИОННЫХ СИСТЕМ»**

**МДК 05.03 «ТЕСТИРОВАНИЕ ИНФОРМАЦИОННЫХ СИСТЕМ»**

**09.02.07 «ИНФОРМАЦИОННЫЕ СИСТЕМЫ И ПРОГРАММИРОВАНИЕ»**

|  |  |  |  |
| --- | --- | --- | --- |
| Студент |  | **/** | *А.А. Васюткин* |
|  | *подпись* |  | *И.О. Фамилия* |
| 22.04.2022 г. |  |  |  |
| Руководитель |  | **/** | *Н.К. Коровина* |
|  | *подпись* |  | *И.О. Фамилия* |
| 22.04.2022 г. |  |  |  |

Тольятти, 2022

Государственное бюджетное профессиональное образовательное учреждение

Самарской области

«Тольяттинский социально-экономический колледж»

|  |  |  |
| --- | --- | --- |
|  |  | Утверждаю:  Заместитель директора по УР  \_\_\_\_\_\_\_\_\_\_\_\_\_\_ М.С. Киронова  *« » 2022 г.* |

**ЗАДАНИЕ**

на курсовую работу

**по ПМ.05 Проектирование и разработка информационных систем модуля, выполняемой в рамках МДК.05.03 Тестирование информационных систем**

студента группы ИСП-31

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Васюткина Арсения Андреевича\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Фамилия Имя отчество студента*

Тема курсовой работы: «Разработка информационной системы **«**Учет обмена валютами»

1. Содержание задания:

1.1 Тестирование информационной системы.

1. Исходные данные:

Исходные данные для практической реализации автоматизированной информационной системы (АИС) берутся из различных информационных источников (Интернет-ресурсы, печатные издания, периодика и др.).

1. Содержание курсовой работы

Введение

1 Анализ методов тестирования

1.1 Критерии и принципы тестирования

1.2 Методы тестирования

2 Тестирование информационной системы (название)

2.1. Разработка тестовой документации (тест-дизайн)

2.2. Разработка тестовых сценариев

Заключение

Список использованных источников

Приложения

Дата выдачи задания: « » 2022 г.

Дата сдачи работы на отделение: « » 2022 г.

Руководитель курсового(ой) проекта(работы) \_\_\_\_\_\_\_\_\_\_\_\_\_ / \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись расшифровка подписи

Государственное бюджетное профессиональное образовательное учреждение

Самарской области

«Тольяттинский социально-экономический колледж»

**КАЛЕНДАРНЫЙ ПЛАН**

выполнения курсовой работы

Студентом 3 курса ИСП-31 группы Васюткиным А.А.

По теме **Учет обмена валютами**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| №  этапа  работы | Содержание этапов работы | Плановый срок выполнения этапа | Планируемый объем выполнения  этапа, % | Отметка  о  выполнении  этапа |
| 1 | Выбор, обоснование темы и объекта исследования | Январь 2022 | 5% |  |
| 2 | Утверждение темы, согласование плана. Введение, библиография | Январь 2022 | 10% |  |
| 3 | Изучение и анализ информационных материалов по теме | Февраль 2022 | 15% |  |
| 4 | Обоснование актуальности выбранной темы применительно к профессиональной деятельности (введение) | Февраль 2022 | 20% |  |
| 5 | Изложение материала основной части по теме курсовой работы | Февраль 2022 | 20% |  |
| 6 | Подведение итогов проведенного анализа, формулировка выводов УИР применительно к профессиональной деятельности (заключение) | Март 2022 | 20% |  |
| 7 | Оформление работы и сдача на проверку | Март 2022 | 10% |  |
| 8 | Защита работы |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Студент |  | **/** | *А.А. Васюткин* |
|  | *подпись* |  | *И.О. Фамилия* |
| 22.04.2022 г. |  |  |  |
| Руководитель |  | **/** | *Н.К. Коровина* |
|  | *подпись* |  | *И.О. Фамилия* |
| 22.04.2022 г. |  |  |  |

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 5](#_Toc103381303)

[1 АНАЛИЗ МЕТОДОВ ТЕСТИРОВАНИЯ 6](#_Toc103381304)

[1.1 Критерии и принципы тестирования 6](#_Toc103381305)

[1.2 Методы тестирования 9](#_Toc103381306)

[2 ТЕСТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ (УЧЕТ ОБМЕНА ВАЛЮТАМИ) 12](#_Toc103381307)

[2.1. Разработка тестовой документации (тест-дизайн) 12](#_Toc103381308)

[2.2. Разработка тестовых сценариев 15](#_Toc103381309)

[ЗАКЛЮЧЕНИЕ 20](#_Toc103381310)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 21](#_Toc103381311)

# ВВЕДЕНИЕ

Информатизация в современном мире превратилась в один из наиболее важных ресурсов. Информационные системы (ИС) стали необходимым инструментом решения задач в информационной сфере практически для всех сфер деятельности малых, средних и больших предприятиях всех видов собственности. ИС обеспечивают:

- хранение, обработку, обобщение данных, поступаемых от объектов управления, пользователей ИС;

- представление информации в удобном для принятия решения виде.

Целью курсовой работы является приобретение навыков решения задачи проектирования ИС «Учет обмена валюты» средствами и инструментами информационных технологий, умения на современном уровне представить информацию при их технической разработке, обеспечивающей возможность ее автономной качественной реализации на практике.

Для достижения поставленной цели были определены следующие задачи:

* описать предметную область;
* спроектировать программный комплекс;
* выполнить программную реализацию;
* проанализировать качество программного обеспечения.

В данной курсовой работе внимание будет уделено анализу и тестированию программного обеспечения, разработанного в рамках курсовой работы.

# 1 АНАЛИЗ МЕТОДОВ ТЕСТИРОВАНИЯ

## 1.1 Критерии и принципы тестирования

Тестирование программного обеспечения (Software Testing) — проверка соответствия реальных и ожидаемых результатов поведения программы, проводимая на конечном наборе тестов, выбранном определённым образом.

Тестирование программного обеспечения – креативная и интеллектуальная работа. Разработка правильных и эффективных тестов – достаточно непростое занятие. Принципы тестирования, представленные ниже, были разработаны в последние 40 лет и являются общим руководством для тестирования в целом.

1. Тестирование показывает наличие дефектов. Тестирование может показать наличие дефектов в программе, но не доказать их отсутствие. Тем не менее, важно составлять тест-кейсы, которые будут находить как можно больше багов. Таким образом, при должном тестовом покрытии, тестирование позволяет снизить вероятность наличия дефектов в программном обеспечении. В то же время, даже если дефекты не были найдены в процессе тестирования, нельзя утверждать, что их нет.

2. Исчерпывающее тестирование невозможно. Невозможно провести исчерпывающее тестирование, которое бы покрывало все комбинации пользовательского ввода и состояний системы, за исключением совсем уж примитивных случаев. Вместо этого необходимо использовать анализ рисков и расстановку приоритетов, что позволит более эффективно распределять усилия по обеспечению качества ПО.

3. Раннее тестирование. Тестирование должно начинаться как можно раньше в жизненном цикле разработки программного обеспечения и его усилия должны быть сконцентрированы на определенных целях.

4. Скопление дефектов. Разные модули системы могут содержать разное количество дефектов, то есть плотность скопления дефектов в разных элементах программы может отличаться. Усилия по тестированию должны распределяться пропорционально фактической плотности дефектов. В основном, большую часть критических дефектов находят в ограниченном количестве модулей. Это проявление принципа Парето: 80% проблем содержатся в 20% модулей.

5. Парадокс пестицида. Прогоняя одни и те же тесты вновь и вновь, можно столкнутся с тем, что они находят все меньше новых ошибок. Поскольку система эволюционирует, многие из ранее найденных дефектов исправляют и старые тест-кейсы больше не срабатывают.

Чтобы преодолеть этот парадокс, необходимо периодически вносить изменения в используемые наборы тестов, рецензировать и корректировать их с тем, чтобы они отвечали новому состоянию системы и позволяли находить как можно большее количество дефектов.

6. Тестирование зависит от контекста. Выбор методологии, техники и типа тестирования будет напрямую зависеть от природы самой программы. Например, программное обеспечение для медицинских нужд требует гораздо более строгой и тщательной проверки, чем, скажем, компьютерная игра. Из тех же соображений сайт с большой посещаемостью должен пройти через серьезное тестирование производительности, чтобы показать возможность работы в условиях высокой нагрузки.

7. Заблуждение об отсутствии ошибок. Тот факт, что тестирование не обнаружило дефектов, еще не значит, что программа готова к релизу. Нахождение и исправление дефектов будет не важным, если система окажется неудобной в использовании и не будет удовлетворять ожиданиям и потребностям пользователя.

И еще несколько важных принципов:

* тестирование должно производиться независимыми специалистами;
* привлекайте лучших профессионалов;
* тестируйте как позитивные, так и негативные сценарии;
* не допускайте изменений в программе в процессе тестирования;
* указывайте ожидаемый результат выполнения тестов.

Можно выделить требования к идеальному критерию тестирования:

* критерий должен быть достаточным, т.е. показывать, когда некоторое конечное множество тестов достаточно для тестирования данной программы;
* критерий должен быть легко проверяемым, например, вычисляемым на тестах;
* критерий должен быть надежным, т.е. любые два множества тестов, удовлетворяющих ему, одновременно должны раскрывать или не раскрывать ошибки программы;
* критерий должен быть полным, т.е. в случае ошибки должен существовать тест из множества тестов, удовлетворяющих критерию, который раскрывает ошибку.

Для нетривиальных классов программ в общем случае не существует полного и надежного критерия, зависящего от программ или спецификаций. Поэтому, как правило, стремятся к идеальному общему критерию через реальные частные.

Классы критериев:

* структурные критерии используют информацию о структуре программы (критерии так называемого «белого ящика»);
* функциональные критерии формулируются в описании требований к программному изделию (критерии так называемого «черного ящика»);
* критерии стохастического тестирования формулируются в терминах проверки наличия заданных свойств у тестируемого приложения, средствами проверки некоторой статистической теории;
* мутационные критерии ориентированы на проверку свойств программного изделия на основе подхода Монте-Карло.

Структурные критерии. Используются на этапах модульного и интеграционного тестирования (Unit testing, Integration testing) и базируются на основных элементах УГП, операторах, ветвях и путях.

* тестирование команд (критерий с0) - набор тестов в совокупности должен обеспечить прохождение каждой команды не менее одного раза.
* тестирование ветвей (критерий с1) - набор тестов в совокупности должен обеспечить прохождение каждой ветви не менее одного раза.
* тестирование путей (критерий с2) - набор тестов в совокупности должен обеспечить прохождение каждого пути не менее 1 раз. если программа содержит цикл (в особенности с неявно заданным числом итераций), то число итераций ограничивается константой.
* тестирование условий - покрытие всех булевских условий в программе. критерии покрытия решений (ветвей - с1) и условий не заменяют друг друга, поэтому на практике используется комбинированный критерий покрытия условий/решений, совмещающий требования по покрытию и решений, и условий.

## 1.2 Методы тестирования

Методы тестирования программного обеспечения — это способы тестирования тестируемого приложения на соответствие функциональным или нефункциональным требованиям, полученным от бизнеса. Каждый метод тестирования помогает найти определенный тип дефекта. Например, методы, которые могут обнаружить структурные дефекты, могут быть не в состоянии найти дефекты в отношении сквозного бизнес-потока. Следовательно, в проекте тестирования применяются несколько методов тестирования, чтобы завершить его с приемлемым качеством.

Есть несколько основных методов тестирования, которые формируют часть режима тестирования программного обеспечения:

* тестирование методом черного ящика;
* тестирование методом белого ящика;
* тестирование методом серого ящика.

Тестирование методом черного ящика осуществляется без каких-либо знаний внутренней работы системы. Тестер будет стимулировать программное обеспечение для пользовательской среды, предоставляя различные входы и тестируя сгенерированные выходы. Этот тест также известен как Black-box, closed-box тестирование или функциональное тестирование.

Из приемуществ тестирования черного ящика можно выделить следующее:

* хорошо подходит и эффективен для больших сегментов кода;
* кодовый доступ не требуется;
* четкое разделение перспективы пользователя с точки зрения разработчика с помощью явно определенных ролей;
* большое количество умеренно квалифицированных тестировщиков может протестировать приложение без каких-либо знаний о реализации, языке программирования или операционных системах.

Проверка белого ящика — это подробное исследование внутренней логики и структуры кода. Тестирование с использованием белого ящика также называется тестированием стекла или открытым тестированием. Чтобы выполнить тестирование белого ящика в приложении, тестер должен знать внутреннюю работу кода.

Тестер должен заглянуть внутрь исходного кода и выяснить, какой блок кода ведет себя некорректно.

Можно отметить следующие преимущества данного тестирования:

* поскольку тестер знает исходный код, становится очень легко узнать, какой тип данных может помочь в эффективном тестировании приложения;
* метод тестирования помогает в оптимизации кода;
* дополнительные строки кода могут быть удалены, что может привести к скрытым дефектам;
* благодаря знаниям тестера о коде, максимальный охват достигается при написании сценария сценария.

Тестирование методом серого ящика или Gray box тестирование, это что-то среднее между White Box и Black Box тестированием, где тестер обладает лишь общими знаниями данного продукта, необходимыми для выполнения теста. Тестирование на серой коробке — это метод тестирования приложения с ограниченным знанием внутренней работы приложения. При тестировании программного обеспечения фраза, чем больше вы знаете, тем лучше переносит массу при тестировании приложения.

Освоение домена системы всегда дает тестеру преимущество над кем-то с ограниченными знаниями домена. В отличие от тестирования черного ящика, где тестер тестирует только пользовательский интерфейс приложения, при тестировании методом серого ящика тестер имеет доступ к проектной документации и базе данных. Имея эти знания, тестер может подготовить лучшие тестовые данные и сценарии тестирования при составлении плана тестирования.

Из основных преимуществ можно отметить следующее:

* предлагает комбинированные преимущества тестирования черного ящика и белого ящика, где это возможно;
* тестировщики серого ящика не полагаются на исходный код; вместо этого они полагаются на определение интерфейса и функциональные спецификации;
* основываясь на имеющейся ограниченной информации, тестер серого ящика может разработать отличные сценарии тестирования, особенно в отношении протоколов связи и обработки данных;
* тест выполняется с точки зрения пользователя, а не дизайнера.

# 2 ТЕСТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ (УЧЕТ ОБМЕНА ВАЛЮТАМИ)

## 2.1. Разработка тестовой документации (тест-дизайн)

Документ описывает методы и подходы к тестированию, которые будут использоваться тестировщиками отдела тестирования, для тестирования приложения. План тестирования может использоваться как тестировщиками, так и менеджерами, разработчиками. Объект тестирования — это деятельность, направленная на проверку работоспособности функций приложения учета обмена валютами.

Целью тестирования приложения учета обмена валютами является проверка корректной работы.

Итогом процесса тестирования будут следующие материалы:

1. Определить существующую информацию о проекте и программных компонентах, подлежащих тестированию.
2. Описать стратегии тестирования, которые будут использоваться.
3. Определить необходимые ресурсы для проведения работ по тестированию.
4. Привести результаты тестирования.

Результаты будут отправлены заказчику в виде отчетов.

Условия для тестирования. Приложение должно удовлетворять потребность пользователя в активностях, связанных с просмотром и добавлением информации о курсах валют, клиентах, счетах и произведенных операциях.

Стратегия процесса тестирования. Приведенный ниже план тестирования является формальным, так как для построения развернутого плана необходимо понимание текущего состояния проекта.

Основными задачами тестирования являются:

* проведение функционального тестирования каждого модуля и компонента системы для обеспечения его соответствия функциональным требованиям;
* тестирование данных и целостности базы данных.

Виды тестирования. Для решения указанных выше задач тестирования будут использоваться следующие виды тестирования.

1. Тестирование данных и целостности базы данных

Базы данных должны тестироваться как отдельные системы внутри учета обмена валютами. Эти системы должны тестироваться отдельно от приложений (таких как интерфейс доступа к данным).

Необходимо провести дополнительное исследование СУБД на тему того, какие инструменты/техники существуют для выполнения нижеописанного тестирования.

1.1 Цель тестирования

Убедится в том, что методы доступа к данным работают правильно и без нарушения целостности БД.

1.2 Способы

* Вызвать каждый метод доступа к БД, предоставляя правильные и не правильные данные (или запросы к данным).
* Исследовать БД на предмет корректного заполнения ее данными, корректной обработки событий

1.3 Критерий завершенности

Все методы и процедуры БД функционируют так, как им положено и без нарушения целостности самой БД.

1.4 Особые замечания

* При тестировании может понадобиться среда разработки СУБД или драйвера для корректного подключения к базам данных.
* Процедуры должны вызываться вручную.
* Для повышения видимости неприемлемых событий БД необходимо использовать небольшие БД или БД с ограниченным количеством записей.

2. Функциональное тестирование

1.1 Цель тестирования

Функциональное тестирование состоит в том, чтобы убедиться, что весь программный продукт работает в соответствии с требованиями, и в приложении не появляется существенных ошибок.

1.2 Способы

**Авторизация**

Авторизация пользователя

**Просмотр информации о клиентах**

Вывод данных с БД

Добавление информации

**Просмотр курсов валют**

Вывод данных с БД

Добавление информации

**Просмотр информации о валютных операциях**

Вывод данных с БД

Добавление информации

**Просмотр информации о счетах**

Вывод данных с БД

Добавление информации

1.3 Критерий завершенности

Программный продукт должен пройти все запланированные тесты.

1.4 Особые замечания

Ожидаемые результаты возникают при использовании достоверных данных.

Соответствующие сообщения об ошибках или предупреждения отображаются, когда используются неверные данные.

Подготовлено тестовое окружение, приложение готово к тестированию на тестовой площадке.

Не будет проведено нагрузочное и тестирование безопасности в виду отсутствия необходимых ресурсов.

Отчеты об ошибках создаются для того, чтобы предоставить команде разработчиков и руководителю проекта исчерпывающую информацию об обнаруженных ошибках. Они должны быть полезны при определении причин ошибок и их исправлении.

Продукт должен работать в соответствии с требованиями и техническим заданием. Продукт не должен содержать критических и блокирующих дефектов в окончательной версии проекта.

## 2.2. Разработка тестовых сценариев

Для C# как правило, проще создать проект модульного теста из кода. Кроме того, можно создать проект модульных тестов и тесты вручную в зависимости от потребностей. Если необходимо создать модульный тест из кода на сторонней платформе, то потребуется установить одно из этих расширений: NUnit или xUnit.

Рассмотрим более детально процесс создания проекта модульного теста.

Прежде всего необходимо создать проект модульного теста для этого нажмем на панели инструментов Файл, выберем Добавить и Создать проект.

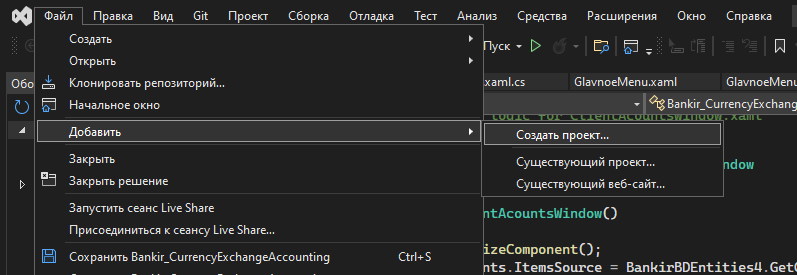


Рисунок 1 – Создание модульного проекта

В новом окне введем test в поле поиска, выберите C# в качестве языка, затем выберем Проект модульного теста MSTest (.NET Core) для C# и в качестве шаблона выберем .NET Core и щелкнем Далее.

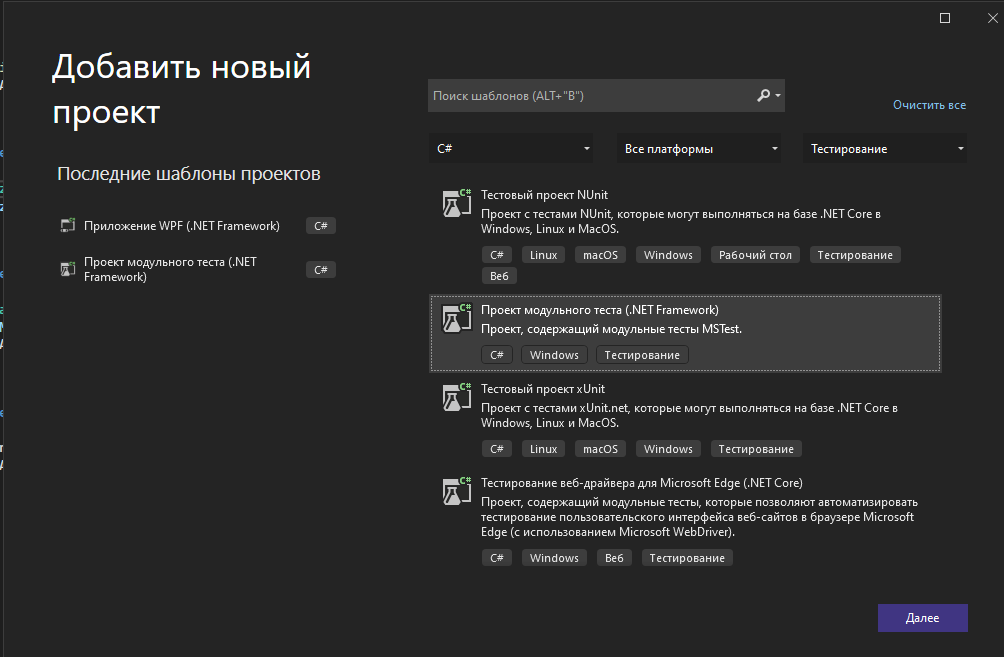


Рисунок 2 – Окно создания модульного теста

Выбор имени проекта сугубо индивидуален, но, чтобы название соответствовало тестируемому решению, назовем его BankTests.

После чего как можно заметить в обозревателе решений появиться ново-созданный тест. Его стоит связать с решением, что будет тестироваться в дальнейших этапах.

В обозревателе решений щелкнем на Зависимости в проекте BankTests, а затем выберите в контекстном меню элемент Добавить ссылку или Добавить ссылку на проект. В диалоговом окне Диспетчер ссылок развернем Проекты, выберем Решение и оставим галочку тому элементу что совпадает с названием решения что будет тестироваться.

Создадим тестовый класс, чтобы проверить класс BankAccount. Можно использовать UnitTest1.cs, созданный в шаблоне проекта, но лучше дать файлу и классу более описательные имена.

Чтобы переименовать файл, в обозревателе решений выберем файл UnitTest1.cs в проекте BankTests. В контекстном меню выберем команду Переименовать (или нажать клавишу F2), а затем переименуем файл в BankAccountTests.cs.

Чтобы переименовать класс, поместим курсор в UnitTest1 в редакторе кода, щелкнем правой кнопкой мыши и выберем команду Переименовать. Введем название BankAccountTests и нажмем клавишу ВВОД.

Файл BankAccountTests.cs теперь содержит следующий код:

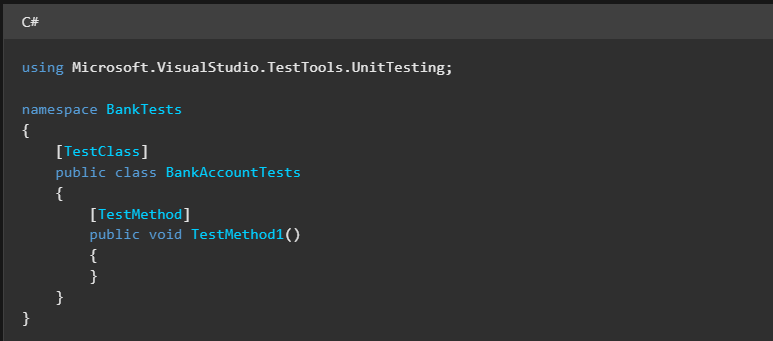


Рисунок 2 – Код файла BankAccountTests.cs

Можно также добавить оператор using в класс, чтобы тестируемый проект можно было вызывать без использования полных имен. Вверху файла класса добавим:

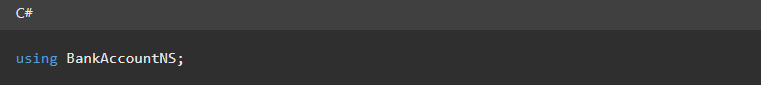


Рисунок 3 – Добавленный using в класс BankAccountTests.cs

Прежде всего перед тем как продолжить стоит усвоить минимальные требования к тестовому классу:

* атрибут [testclass] является обязательным 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].

Можно иметь другие классы в проекте модульного теста, которые не содержат атрибута [TestClass], а также иметь другие методы в тестовых классах, у которых атрибут — [TestMethod]. Можно вызывать эти другие классы и методы в методах теста.

В случае с данным тестированием напишем метод модульного теста для проверки поведения метода Debit класса BankAccount.

Существует по крайней мере 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.

Данный тест, реализуемый в рамках курсовой работы, проверяет, снимается ли со счета нужная сумма при допустимом размере кредита (со значением меньшим, чем баланс счета, и большим, чем ноль). Добавим следующий метод в этот класс BankAccountTests:

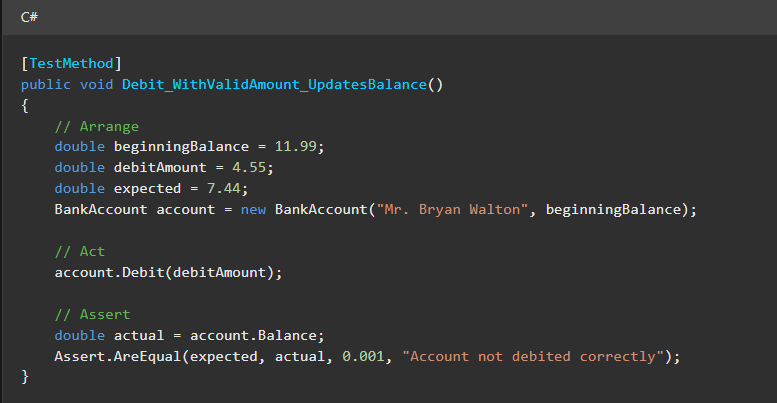


Рисунок 4 – Добавленный метод в классе BankAccountTests

Метод очень прост: он создает новый объект BankAccount с начальным балансом, а затем снимает допустимое значение. Он использует метод Assert.AreEqual, чтобы проверить, что конечный баланс соответствует ожидаемому. Такие методы, как Assert.AreEqual, Assert.IsTrue и другие, зачастую используются в модульном тестировании.

Дальнейшем шагом после написания теста является его сборка и непосредственно сам запуск.

В меню Сборка нажмем Построить решение:

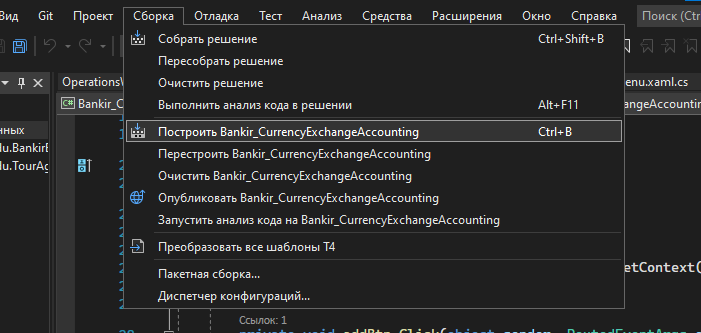


Рисунок 5 – Контекстное меню пункта Сборка

Откроем Обозреватель тестов, выбрав Тест, далее Windows и Обозреватель тестов в верхней строке меню.

Выберем Запустить все, чтобы выполнить тест.

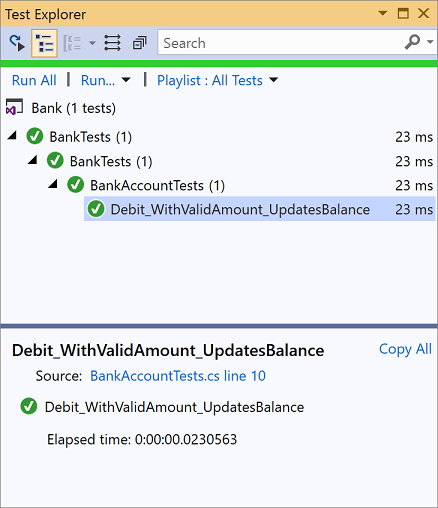


Рисунок 6 – Окно обозревателя тестов

Во время выполнения теста в верхней части окна Обозреватель тестов отображается анимированная строка состояния. По завершении тестового запуска строка состояния становится зеленой, если все методы теста успешно пройдены, или красной, если какие-либо из тестов не пройдены. В случае с данным тестированием тест пройден успешно.

# ЗАКЛЮЧЕНИЕ

Тестирование программного обеспечения — это проверка соответствия реальных и ожидаемых результатов поведения программы, проводимая на конечном наборе тестов, выбранном определённым образом. Различные наборы тест-кейсов и стратегий тестирования направлены на достижение одной общей цели - устранение багов и ошибок в коде, и обеспечения точной и оптимальной производительности программного обеспечения.

В первой части курсовой работы были проанализированы методы тестирования, такие как, метод тестирования черным ящиком, белым и серым.

Были исследованы критерии и принципы тестирования информационной системы.

Во второй части курсовой работы на основе изученных методов тестирования была разработана и подробно описана тестовая документация (тест-дизайн) информационной системы, а также был разработан тестовый сценарий для информационной системы «Учет обмена валютами».

Подробно описаны и проиллюстрированы функционал и структура самого модуля, а также проведено тестирование готового программного продукта в программе MS Visual Studio 2022.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. ГОСТ 34.601 – 90. Информационная технология. Комплекс стандартов на автоматизированные системы. Автоматизированные системы. Стадии создания.
2. ГОСТ 34.602 - 2020. Информационная технология. Комплекс стандартов на автоматизированные системы. Техническое задание на создание автоматизированной системы.
3. ГОСТ 19.201 - 78 ЕСПД. Техническое задание. Требования к содержанию и оформлению.
4. ГОСТ 19.202 - 78 ЕСПД. Спецификация. Требования к содержанию и оформлению.
5. ГОСТ Р ИСО/МЭК 12207. Процессы жизненного цикла программных средств.
6. Золотов С.Ю. Проектирование информационных систем: Учебно-методическое пособие. Методические рекомендации для выполнения курсового проекта, лабораторных работ и практических занятий по дисциплине «Проектирование информационных систем» - Томск: ТУСУР, 2013. - 34 с.
7. Шнайдер, Роберт Microsoft SQL Server 6.5. Проектирование высокопроизводительных баз данных; М.: Лори, 2010. - 361 c
8. Петкович, Душан Microsoft SQL Server 2012. Руководство для начинающих / Душан Петкович. - М.: БХВ-Петербург, 2012. - 460 c.
9. Тейлор, Аллен SQL для чайников / Аллен Тейлор. - М.: Вильямс, 2014. - 416 c.
10. Браст, Э.Дж. Разработка приложений на основе Microsoft SQL Server 2008 / Э.Дж. Браст. - М.: Русская Редакция, 2010. - 751 c.
11. Хетагуров, Я. А. Проектирование автоматизированных систем обработки информации и управления (АСОИУ). Учебник / Я.А. Хетагуров. - М.: Бином. Лаборатория знаний, 2015. - 240 c.
12. Кристофер, Д. Маннинг Введение в информационный поиск / Кристофер Д. Маннинг, ПрабхакарРагхаван ,ХайнрихШютце. - М.: Вильямс, 2014. - 528 c.
13. Бишоп Дж. С# в кратком изложении; Бином. Лаборатория знаний - М., 2015. - 234 c.
14. ГриффитсИэн Программирование на C# 5.0; Эксмо - М., 2014. - 580 c.
15. Гуриков С. Р. Введение в программирование на языке Visual C#; ИЛ - Москва, 2013. - 448 c.
16. Шилдт Герберт C# 4.0. Полное руководство; Вильямс - М., 2015. - 291 c.
17. Эндрю Троелсен Язык программирования C# 5.0 и платформа .NET 4.5; Диалектика / Вильямс - М., 2015. - 126 c.