МИНОБРНАУКИ РОССИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«САРАТОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ ГАГАРИНА Ю.А.»

«СОГЛАСОВАНО»	«УТВЕРЖДАЮ»
Решением П(Ц)МК	Заместитель директора по УР
Протокол № ____	_____ С.В. Клюквина
от «__» _____20__г.	«___» _____ 20__ г.

Методические рекомендации для студентов по проведению лабораторных занятий по дисциплине

«МДК 01.02. Поддержка и тестирование программных модулей» для специальности 09.02.07 «Информационные системы и программирование»

Саратов 2019

Разработал преподаватель СКМ и Э ФГБОУ ВО «СГТУ имени Гагарина Ю.А» Пояркова Т.А.

Данные методические рекомендации предназначены для студентов 3 курса, специальности 09.02.07 «Информационные системы и программирование»

СОДЕРЖАНИЕ

Введение

Лабораторные занятия, как виды учебных занятий, направлены на экспериментальное подтверждение теоретических положений и формирование учебных и профессиональных лабораторных умений и составляют важную часть теоретической и профессиональной практической подготовки. Семинар является видом лабораторных занятий.

В процессе лабораторного занятия обучающиеся выполняют одно или несколько лабораторных заданий под руководством преподавателя в соответствии с изучаемым содержанием учебного материала.

Выполнение обучающимися лабораторных занятий проводится с целью:

- формирования лабораторных умений в соответствии с требованиями к уровню подготовки обучающихся, установленными рабочей программой дисциплины по конкретным разделам/ темам дисциплины;
- обобщения, систематизации, углубления, закрепления полученных теоретических знаний;
- совершенствования умений применять полученные знания на практике, реализации единства интеллектуальной и практической деятельности;
- развития интеллектуальных умений у будущих специалистов: аналитических, проектировочных, конструктивных и др.;
- выработки таких профессионально значимых качеств, как самостоятельность, ответственность, точность, творческая инициатива при решении поставленных задач при освоении общих компетенций.

Цель лабораторных занятий

В результате освоения учебной дисциплины «МДК 01.02. Поддержка и тестирование программных модулей» обучающийся должен **уметь:**

В результате освоения дисциплины обучающийся должен уметь:

– У1 Осуществлять разработку кода программного модуля на языках высокого уровней.

– У2 Создавать программу по разработанному алгоритму как отдельный модуль.

– У3 Выполнять отладку и тестирование программы на уровне модуля.

– У4 Осуществлять разработку кода программного модуля на современных языках программирования

– У5 Уметь выполнять оптимизацию и рефакторинг программного кода;

– У6 Оформлять документацию на программные средства

В результате освоения дисциплины обучающийся должен знать:

– З1 Основные этапы разработки программного обеспечения.

– З2 Основные принципы технологии структурного и объектноориентированного программирования.

– З3 Способы оптимизации и приемы рефакторинга

– З4 Основные принципы отладки и тестирования программных продуктов

Изучение дисциплины направлено на формирование следующих компетенций:

ОК 01. Выбирать способы решения задач профессиональной деятельности, применительно к различным контекстам.

ОП 02.Осуществлять поиск, анализ и интерпретацию информации, необходимой для выполнения задач профессиональной деятельности.

ОК 03. Планировать и реализовывать собственное профессиональное и личностное развитие.

ОК 04. Работать в коллективе и команде, эффективно взаимодействовать с коллегами, руководством, клиентами.

ОК 05. Осуществлять устную и письменную коммуникацию на государственном языке с учетом особенностей социального и культурного контекста.

ОК 06. Проявлять гражданско-патриотическую позицию, демонстрировать осознанное поведение на основе традиционных общечеловеческих ценностей.

ОК 07. Содействовать сохранению окружающей среды, ресурсосбережению, эффективно действовать в чрезвычайных ситуациях.

ОК 08. Использовать средства физической культуры для сохранения и укрепления здоровья в процессе профессиональной деятельности и поддержания необходимого уровня физической подготовленности.

ОК 09. Использовать информационные технологии в профессиональной деятельности.

ОК 10. Пользоваться профессиональной документацией на государственном и иностранном языках.

ОК 11. Планировать предпринимательскую деятельность в профессиональной сфере.

ПК 1.1 Формировать алгоритмы разработки программных модулей в соответствии с техническим заданием

ПК 1.2 Разрабатывать программные модули в соответствии с техническим заданием

ПК 1.3 Выполнять отладку программных модулей с использованием специализированных программных средств

ПК 1.4 Выполнять тестирование программных модулей

ПК 1.5 Осуществлять рефакторинг и оптимизацию программного кода

ПК 1.6 Разрабатывать модули программного обеспечения для мобильных платформ

Организация лабораторного занятия

Лабораторное занятие должно проводиться в учебной лаборатории «Программирования и баз данных» оснащенная необходимым для реализации программы учебной дисциплины оборудованием.

В соответствии с требованиям ФГОС 09.02.07 «Информационные системы и программирование» реализация ППССЗ должна обеспечивать выполнение обучающимися и лабораторных занятий, включая как обязательный компонент *практические занятия (с использованием персональных компьютеров).*

Выполнению лабораторных занятий предшествует проверка знаний обучающихся - их теоретической готовности к выполнению задания.

Практические занятия могут носить репродуктивный, частично-поисковый и поисковый характер.

Работы, носящие *репродуктивный характер,* отличаются тем, что при их проведении обучающиеся пользуются подробными инструкциями, в которых указаны: цель работы, пояснения (теория, основные характеристики), оборудование, аппаратура, материалы и их характеристики, порядок выполнения работы, таблицы, выводы (без формулировки), контрольные вопросы, учебная и специальная литература.

Работы, носящие *частично-поисковый характер*, отличаются тем, что при их проведении обучающиеся не пользуются подробными инструкциями, им не дан порядок выполнения необходимых действий, и они требуют от обучающихся самостоятельного подбора оборудования, выбора способов выполнения работы в инструктивной и справочной литературе и др.

Работы, носящие *поисковый характер*, характеризуются тем, что обучающиеся, опираясь на имеющиеся у них теоретические знания, должны решить новую для них проблему.

При планировании лабораторных занятий необходимо находить оптимальное соотношение репродуктивных, частично-поисковых и поисковых работ, чтобы обеспечить высокий уровень интеллектуальной деятельности.

Формы организации обучающихся при проведении лабораторных занятий - фронтальная, групповая и индивидуальная.

При *фронтальной форме* организации занятий все обучающиеся выполняют одновременно одну и ту же работу.

При *групповой форме* организации занятий одна и та же работа выполняется бригадами по 2 - 5 человек.

При *индивидуальной форме* организации занятий каждый обучающийся выполняет индивидуальное задание.

Для повышения эффективности проведения лабораторных занятий рекомендуется:

- разработка сборников задач, заданий и упражнений;
- разработка контрольно-диагностических материалов для контроля за подготовленностью обучающихся к практическим занятиям, в том числе в форме педагогических тестовых материалов для автоматизированного контроля;
- подчинение методики проведения лабораторных занятий ведущим дидактическим целям с соответствующими установками обучающимся;
- применение коллективных и групповых форм работы, максимальное использование индивидуальных форм с целью повышения ответственности каждого обучающегося за самостоятельное выполнение полного объема работ;

- проведение лабораторных занятий на повышенном уровне трудности с включением в них заданий, связанных с выбором обучающимися условий выполнения работы, конкретизацией целей, самостоятельным отбором необходимого оборудования;
- подбор дополнительных задач и заданий для обучающихся, работающих в более быстром темпе, для эффективного использования времени, отводимого на практические занятия.

Структура лабораторного занятия

Лабораторное занятие № 1

Тема: Выявление несоответствие результата выполнения модуля его спецификации

Цели и задачи урока: Изучить особенностей спецификаций.

Тип урока – практическое занятие.

Методы обучения – *репродуктивный*

Обеспечение занятия – персональный компьютер.

Ход урока

1. Объяснение нового материала
2. Выполнение практической работы
3. Закрепление пройденного материала (ответы на контрольные вопросы)
4. Подведение итогов

Литература

1. Федорова Н .Г., Разработка программных модулей программного обеспечения для компьютерных систем : учебник для сред. проф. образ-я / Г. Н. Федорова. - М. : Академия, 2020. - 336 с. - (Профессиональное образование).
2. Тестирование программного обеспечения: Учебное пособие. / Куликов С. С.. - МИНСК ИЗДАТЕЛЬСТВО «ЧЕТЫРЕ ЧЕТВЕРТИ», 2021. - 314 с.: доступ http:// svyatoslav.biz/software_testing_book/
3. Голицына О.Л., Попов И.И., Программирование на языках высокого уровня Голицына О.Л., Попов И.И., – М.: Форум 2020.
4. Михеева Е.В. Информационные технологии в профессиональной деятельности : учеб. пособие для студ. сред. проф. образования. – 8-е изд., стер. - М.: Издательский центр «Академия», 2020. - 384 с.

Теоретические сведения

Согласно ГОСТ 34.602-89, основными разделами ТЗ являются:

1. Общие сведения
2. Назначение и цели создания (развития) системы
3. Характеристика объектов автоматизации
4. Требования к системе
5. Состав и содержание работ по созданию системы
6. Порядок контроля и приемки системы
7. Требования к составу и содержанию работ по подготовке объекта автоматизации к вводу системы в действие
8. Требования к документированию
9. Источники разработки

Задание на лабораторную работу

Данная спецификация требований далеко не полна, в частности, не полна спецификация пользовательского интерфейса, функциональных требований. Вам предполагается дополнить спецификацию самостоятельно.

1. **Общее описание**

Калькулятор состоит из трех модулей – "Графический интерфейс", "Модуль, анализирующий и вычисляющий введенное выражение" (AnalaizerClass.dll) и "Модуль, реализующий математические функции" (CalcClass.dll). После того, как пользователь введет вычисляемое выражение одним из двух вышеописанных способов, управление передается анализирующему модулю, который форматирует выражение, выделяя числа и операторы, проверяет корректность скобочной структуры, а также выявляет неверные с точки зрения

математики конструкции (например, 3+*+3), переводит выражение в *обратную польскую запись*, после чего вычисляет выражения, используя математические функции из модуля CalcClass.

2. **Описание интерфейса**.

1. **Входные данные**

- **Параметры вызова (формат командной строки)**

calc.exe [expression]

expression – математическое выражение, удовлетворяющее требованию 3.2

- **Состояние информационного окружения**.

В папке с программой также находятся файлы CalcClass.dll, AnalaizerClass.dll

2. **Выходные данные**.

- **Коды возврата программы**.

Число и 0 на новой строке – результат вычислений выражения.

Error: <сообщение об ошибке> и код ошибки на новой строке — сообщение об ошибке в случае несоответствия входного выражения требованиям 3.2

- **Состояние информационного окружения после завершения программы**.

В папке с программой также находятся файлы CalcClass.dll, AnalaizerClass.dll

- **Сообщения об ошибках, выдаваемые программой (коды ошибок)**.

Error 01 at <i> — Неправильная скобочная структура, ошибка на <i> символе

Error 02 at <i> — Неизвестный оператор на <i> символе

Error 03 — Неверная синтаксическая конструкция входного выражения

Error 04 at <i> — Два подряд оператора на <i> символе

Error 05 — Незаконченное выражение

Error 06 — Слишком малое или слишком большое значение числа для int. Числа должны быть в пределах от -2147483648 до 2147483647

Error 07 — Слишком длинное выражение. Максмальная длина — 65536 символов.

Error 08 — Суммарное количество чисел и операторов превышает 30

Error 09 – Ошибка деления на 0

3. **Описание файлов, входящих в пакет калькулятора**.

CalcClass.dll – библиотека, в которой реализованы все необходимые математические функции.

AnalaizerClass.dll – модуль, в котором реализован синтаксический разбор выражения, а также его вычисление.

calc.exe – графическая оболочка, главный модуль.

4. **Интерфейс пользователя**.

Рис. 1. Интерфейс пользователя системы "Калькулятор"

Клавиши "1" "2" "3" "4" "5" "6" "7" "8" "9" "0" "/" "*" "-" "+" "mod" "(" ")" – вводят соответствующий символ в поле выражение. Клавиша "Сброс" очищает поле "Выражение", клавиша "Стереть" удаляет последний введенный символ. Клавиша "=" начинает выполнение вычислений. "MR", "M+" и "MC" управляют памятью калькулятора, "+/-" — триггер унарного плюса унарного минуса.

3. Описание архитектуры

Как уже отмечалось выше, в архитектуре системы выделено 3 модуля. Каждый из модулей занимается определенной задачей. Соответственно, Система – это взаимодействие этих 3-х модулей. Рассмотрим их подробнее.

1. Модуль математических операций (CalcClass.dll)

Модуль содержит все математические функции, используемые в программе.

```
/// <summary>
        /// Функция сложения числа a и b
        /// </summary>
        /// <param name="a">слагаемое</param>
        /// <param name="b">слагаемое</param>
        /// <returns>сумма</returns>
        public static int Add(long a, long b)

        /// <summary>
        /// функция вычитания чисел a и b
        /// </summary>
        /// <param name="a">уменьшаемое</param>
        /// <param name="b">вычитаемое</param>
        /// <returns>разность</returns>
        public static int Sub(long a, long b)


        /// <summary>
        /// функция умножения чисел a и b
        /// </summary>
        /// <param name="a">множитель</param>
        /// <param name="b">множитель</param>
        /// <returns>произведение</returns>
        public static int Mult(long a, long b)

        /// <summary>
        /// функция нахождения частного
        /// </summary>
        /// <param name="a">делимое</param>
        /// <param name="b">делитель</param>
        /// <returns>частное</returns>
        public static int Div(long a, long b)

        /// <summary>
        /// функция деления по модулю
        /// </summary>
        /// <param name="a">делимое</param>
        /// <param name="b">делитель</param>
        /// <returns>остаток</returns>
```

```
public static int Mod(long a, long b)

/// <summary>
/// унарный плюс
/// </summary>
/// <param name="a"></param>
/// <returns></returns>
public static int ABS(long a)

/// <summary>
/// унарный минус
/// </summary>
/// <param name="a"></param>
/// <returns></returns>
public static int IABS(long a)

Используется также глобальная переменная:
/// <summary>
/// Последнее сообщение об ошибке
/// Поле и свойство для него
/// </summary>
private static string _lastError = "";

public static string lastError
```

Листинг 2.1. Модуль математических операций

2. Модуль анализа и вычисления выражений

Состоит из следующих методов и свойств:

```
/// <summary>
/// позиция выражения, на которой отловлена синтаксическая ошибка (в случае ловли на уровне выполнения - не определяется)
/// </summary>
private static int erposition = 0;
/// <summary>
/// Входное выражение
/// </summary>
public static string expression = "";
/// <summary>
/// Показывает, есть ли необходимость в выводе сообщений об ошибках. В случае консольного запуска программы это значение - false.
/// </summary>
public static bool ShowMessage = true;
/// <summary>
/// Проверка корректности скобочной структуры входного выражения
/// </summary>
/// <returns>true - если все нормально, false - если есть ошибка</returns>
/// метод бежит по входному выражению, символ за символом анализируя его и считая количество скобочек. В случае возникновения
/// ошибки возвращает false, а в erposition записывает позицию, на которой возникла ошибка.
public static bool CheckCurrency()
```

```
        /// <summary>
        /// Форматирует входное выражение, выставляя между операторами
        пробелы и удаляя лишние, а также отлавливает неопознанные операторы, следит за концом строки
        /// также отлавливает ошибки на конце строки
        /// </summary>
        /// <returns>конечную строку или сообщение об ошибке, начинающиеся со
        спец. символа &</returns>
        public static string Format()

        /// <summary>
        /// Создает  массив, в котором располагаются операторы и символы,
        представленные в обратной польской записи (безскобочной)
        /// На этом же этапе отлавливаются почти все остальные ошибки (см  код). По сути - это компиляция.
        /// </summary>
        /// <returns>массив обратной польской записи</returns>
        public static System.Collections.ArrayList CreateStack()

        /// <summary>
        /// Вычисление обратной польской записи
        /// </summary>
        /// <returns>результат вычислений или сообщение об ошибке</returns>
        public static string RunEstimate()

        /// <summary>
        /// Метод, организующий вычисления. По очереди запускает
        CheckCorrncy, Format, CreateStack и RunEstimate
        /// </summary>
        /// <returns></returns>
        public static string Estimate()
```

Листинг 2.2. Модуль анализа и вычисления выражений

3. Модуль графического интерфейса – обеспечивает управление системы в графической форме. Основные функции этого модуля – ввод и вывод данных.

Взаимодействие модулей показано на рисунке:

Рис. 1. Взаимодействие модулей системы "Калькулятор"

4. **Функциональные требования** [1]

1. **Требования к программе**

- Калькулятор должен выполнять следующие арифметические операции: сложение, вычитание, умножение, нахождение частного, нахождение остатка. Спецификацию на них см. 3.2.
- Калькулятор должен поддерживать работу с целыми числами в пределах от -2147483648 до 2147483647 (в дальнейшем MININT и MAXINT). В случае выхода за эти пределы должно выдаваться сообщение об ошибке Error 06.
- Калькулятор должен иметь память на одно целое число, а также возможность выводить это число на экран, сбрасывать его значение на 0 и прибавлять к нему любое другое число, введенное в поле ввода.
- При нажатии на клавишу M+ к числу, записанному в память, прибавляется число, записанное в поле "Результат". При этом на сложение накладываются ограничения из 3.2.1.
- Если в поле "Результат" записан код ошибки, то при нажатии на клавишу M+ должно выдаваться сообщение "Невозможно преобразовать к числу".
- При нажатии на кнопку MC число в памяти обнуляется.
- При нажатии на кнопку MR число из памяти приписывается в конец выражения в строке "Выражение".
- Калькулятор должен предоставлять возможность пользователю работать с операциями унарного плюса и унарного минуса.
- Если между нажатиями на кнопку <+/-> проходит менее 3 секунд, то введенный оператор меняется на противоположный.
- Если между нажатиями на кнопку <+/-> проходит более 3 секунд, то к выражению дописывается знак "-".
- Калькулятор должен иметь графический интерфейс, содержащий кнопки с цифрами и арифметическими операциями, кнопкой равенства, кнопками работы с памятью, кнопками редактирования скобочек и кнопками сброса, переключателем унарного минуса/унарного плюса, текстовыми полями для ввода выражения и вывода результата.
- При нажатии на клавишу <Enter> калькулятор должен проводить вычисления выражения.
- При нажатии на клавишу <ESC> программа должна прекращать свою работу.
- В случае неверно построенного вычисляемого выражения или несоответствия его требованиям 3.2 в текстовое окно результата должно выводиться соответствующие сообщение (см 2.2.3)

2. **Арифметические операции**

- Сложение.
- Для чисел, каждое из которых меньше либо равно MAXINT и больше либо равно MININT, функция суммирования должна возвращать правильную сумму с точки зрения математики[2].
- Для чисел, сумма которых больше чем MAXINT и меньше чем MININT, а также в случае, если любое из слагаемых больше чем MAXINT или меньше чем MININT, программа должна выдавать ошибку Error 06 (см 2.2.3[3])
- Вычитание.
- Для чисел, каждое из которых меньше либо равно MAXINT и больше либо равно MININT, функция вычитания должна возвращать правильную разность с точки зрения математики [4]
- Для чисел, разность которых больше чем MAXINT и меньше чем MININT, а также в случае, если любое из чисел больше чем MAXINT или меньше чем MININT, программа должна выдавать ошибку Error 06 (см 2.2.3[5])
- Умножение [6]

- Для чисел, произведение которых меньше либо равно MAXINT и больше либо равно MININT, функция умножения должна возвращать правильное произведение с точки зрения математики.
- Для чисел, произведение которых больше чем MAXINT и меньше чем MININT, а также, в случае если любой из множителей больше чем MAXINT или меньше чем MININT, программа должна выдавать ошибку Error 06 (см 2.2.3)
- Нахождение частного [7]
- Для чисел, меньших либо равных MAXINT и больших либо равных MININT, частное которых меньше либо равно MAXINT и больше либо равно MININT и делитель не равен 0, функция деления должна возвращать правильное частное с точки зрения математики.
- Для чисел, частное которых больше чем MAXINT и меньше чем MININT, а также в случае, если любое из чисел больше чем MAXINT или меньше чем MININT, и для делителя, не равного 0, программа должна выдавать ошибку Error 06 (см 2.2.3)
- Если делитель равен 0, программа должна выдавать ошибку Error 09
- Деление с остатком [8]
- Для чисел, меньших либо равных MAXINT и больших либо равных MININT, остаток которых меньше либо равен MAXINT и больше либо равен MININT и делитель не равен 0, функция деления должна возвращать правильный остаток с точки зрения математики.
- Для чисел, остаток которых больше чем MAXINT и меньше чем MININT, а также в случае, если любое из чисел больше чем MAXINT или меньше чем MININT, и для делителя, не равного 0, программа должна выдавать ошибку Error 06 (см 2.2.3)
- Если делитель равен 0, программа должна выдавать ошибку Error 09
- Унарный плюс \ минус.
- Для чисел, меньших либо равных MAXINT и больших либо равных[9] MININT, операция унарного плюса / минуса должна возвращать число соответствующего знака.
- Для чисел больших MAXINT или меньших MININT функция должна выдавать ошибку Error 06 (см 2.2.3)
- **Дополнительные требования к входному выражению**
- Максимальное суммарное число операторов и чисел – 30.
- Максимальная глубина вложенности скобочной структуры – 3.
- В качестве унарного минуса используется символ " m " , в качестве унарного плюса — " p ".
- Для операции нахождения частного – " / ", для нахождения остатка — " mod ".
- Между операторами скобками и числами может быть любое количество пробелов.
- Разрешается использовать лишь скобки вида " (" и ") "
- Максимальная длина выражения – 65535 символов.

Лабораторное занятие № 2
Тема: Рефакторинг программного кода

Цели и задачи урока: Изучить технологию рефокторинга программного кода

Тип урока – практическое занятие.

Методы обучения – *репродуктивный*

Обеспечение занятия – персональный компьютер.

Ход урока

1. Объяснение нового материала
2. Выполнение практической работы
3. Закрепление пройденного материала (ответы на контрольные вопросы)
4. Подведение итогов

Литература

1. Федорова Н .Г., Разработка программных модулей программного обеспечения для компьютерных систем : учебник для сред. проф. образ-я / Г. Н. Федорова. - М. : Академия, 2020. - 336 с. - (Профессиональное образование).
2. Тестирование программного обеспечения: Учебное пособие. / Куликов С. С.. - МИНСК ИЗДАТЕЛЬСТВО «ЧЕТЫРЕ ЧЕТВЕРТИ», 2021. - 314 с.: доступ http:// svyatoslav.biz/software_testing_book/
3. Голицына О.Л., Попов И.И., Программирование на языках высокого уровня Голицына О.Л., Попов И.И., – М.: Форум 2020.
4. Михеева Е.В. Информационные технологии в профессиональной деятельности : учеб. пособие для студ. сред. проф. образования. – 8-е изд., стер. - М.: Издательский центр «Академия», 2020. - 384 с.

Теоретические сведения

Рефакторинг — это переработка исходного кода программы, чтобы он стал более простым и понятным. Рефакторинг не меняет поведение программы, не исправляет ошибки и не добавляет новую функциональность. Он делает код более понятным и удобочитаемым.

Задания

Задание 1:

В рамках лабораторной работы приведите примеры реализации рефакторинга в следующих системах:

−Visual Studio.

−Visual Assist X.

−Refactor.

−JustCode

−ReSharper

−CodeIt

Для поиска информации используйте сеть Интернет.

Задание 2:

Построить схему с признаками плохого кода, также, графически отразите и опишите причины применения рефакторинга.

Результаты работы представить в MS Word. Минимальное количество анализируемых систем – 5.

Лабораторная работа 3
Тема: Модульное тестирование

Цели и задачи урока: Изучить особенностей спецификаций.

Тип урока – практическое занятие.

Методы обучения – *репродуктивный*

Обеспечение занятия – персональный компьютер.

Ход урока

1. Объяснение нового материала
2. Выполнение практической работы
3. Закрепление пройденного материала (ответы на контрольные вопросы)
4. Подведение итогов

Литература

1. Федорова Н .Г., Разработка программных модулей программного обеспечения для компьютерных систем : учебник для сред. проф. образ-я / Г. Н. Федорова. - М. : Академия, 2020. - 336 с. - (Профессиональное образование).
2. Тестирование программного обеспечения: Учебное пособие. / Куликов С. С.. - МИНСК ИЗДАТЕЛЬСТВО «ЧЕТЫРЕ ЧЕТВЕРТИ», 2021. - 314 с.: доступ http:// svyatoslav.biz/software_testing_book/
3. Голицына О.Л., Попов И.И., Программирование на языках высокого уровня Голицына О.Л., Попов И.И., – М.: Форум 2020.
4. Михеева Е.В. Информационные технологии в профессиональной деятельности : учеб. пособие для студ. сред. проф. образования. – 8-е изд., стер. - М.: Издательский центр «Академия», 2020. - 384 с.

Теоретические сведения

Модульное тестирование - это тестирование программы на уровне отдельно взятых модулей, функций или классов. Цель модульного тестирования состоит в выявлении локализованных в модуле ошибок в реализации алгоритмов, а также в определении степени готовности системы к переходу на следующий уровень разработки и тестирования. Модульное тестирование проводится по принципу "белого ящика", то есть основывается на знании внутренней структуры программы, и часто включает те или иные методы анализа покрытия кода.

Задания

Выполнить модульное тестирование в соответствии с приведенным примером

Создание проекта программы, модули которой будут тестироваться

Разработаем проект содержащий класс, который вычисляет площадь прямоугольника по длине двух его сторон.

Создадим в Visual Studio новый проект Visual C# -> Библиотека классов. Назовём его **MathTaskClassLibrary**.

Class1 переименуем в **Geometry**.

В классе реализуем метод, вычисляющий площадь прямоугольника. Для демонстрации остановимся на работе с целыми числами. Код программы приведён ниже.

```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
```

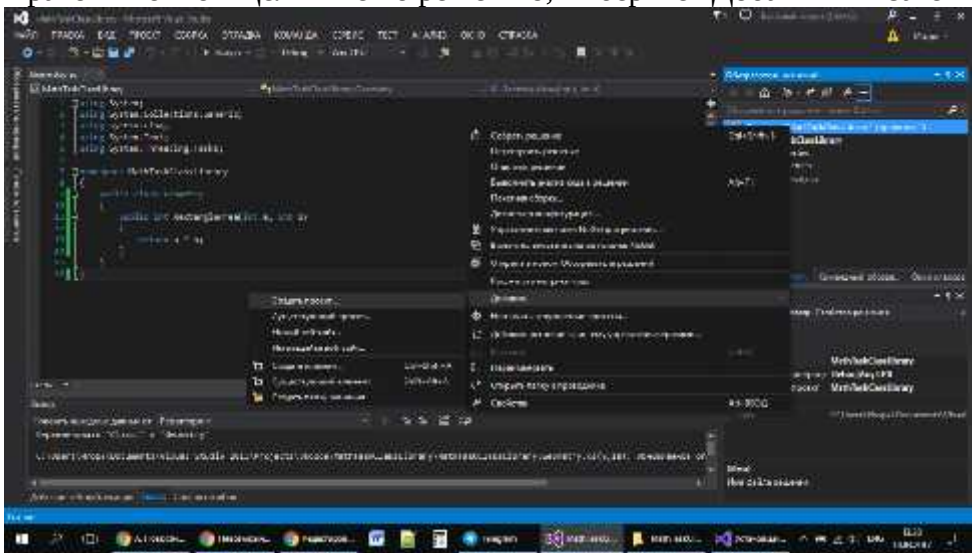
```
 4  using System.Text;
 5  using System.Threading.Tasks;
 6
 7  namespace MathTaskClassLibrary
 8  {
 9      public class Geometry
10      {
11          public int RectangleArea(int a, int b)
12          {
13              return a * b;
14          }
15      }
16  }
```

Площадь прямоугольника, как известно, это произведение двух его сторон.

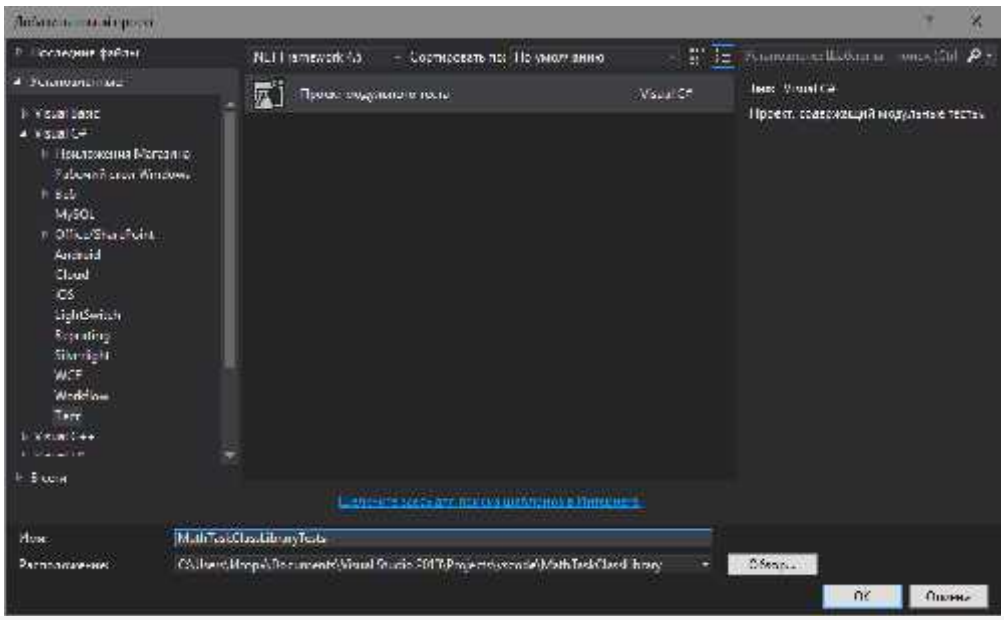
Создание проекта для модульного тестирования в Visual Studio

Чтобы выполнить unit-тестирование, необходимо в рамках того же самого решения создать ещё один проект соответствующего типа.

Правой кнопкой щёлкните по решению, выберите “Добавить” и затем “Создать проект…”.



В открывшемся окне в группе Visual C# щёлкните “Тест”, а затем выберите “Проект модульного теста”. Введите имя проекта **MathTaskClassLibraryTests** и нажмите “ОК”. Таким образом проект будет создан.

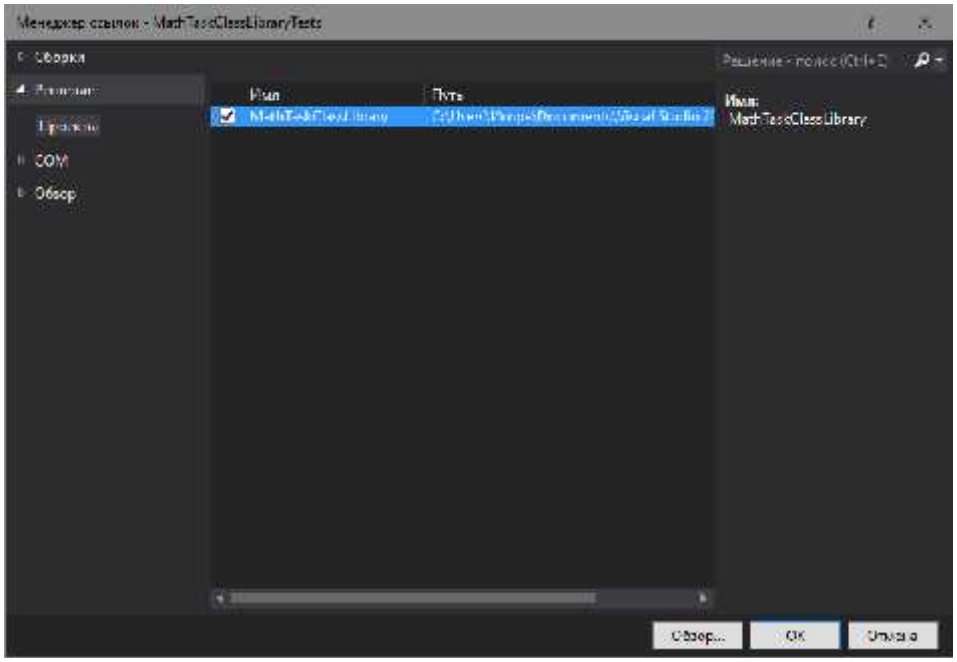


Перед Вами появится следующий код:

```
1  using System;
2  using Microsoft.VisualStudio.TestTools.UnitTesting;
3
4  namespace MathTaskClassLibraryTests
5  {
6      [TestClass]
7      public class UnitTest1
8      {
9          [TestMethod]
10         public void TestMethod1()
11         {
12         }
13     }
14 }
```

Директива [TestMethod] обозначает, что далее идёт метод, содержащий модульный (unit) тест. А [TestClass] в свою очередь говорит о том, что далее идёт класс, содержащий методы, в которых присутствуют unit-тесты.

В соответствии с принятыми соглашениями переименуем класс UnitTest1 в **GeometryTests**. Затем в References проекта необходимо добавить ссылку на проект, код которого будем тестировать. Правой кнопкой щёлкаем на References, а затем выбираем “Добавить ссылку…”. В появившемся окне раскрываем группу “Решение”, выбираем “Проекты” и ставим галочку напротив проекта **MathTaskClassLibrary**. Затем жмём “OK”.



Также в коде необходимо подключить с помощью директивы using следующее пространство имён:

```
1 using MathTaskClassLibrary;
```

Займёмся написание теста. Проверим правильно ли вычисляет программа площадь прямоугольника со сторонами 3 и 5. Ожидаемый результат (правильное решение) в данном случае это число 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 обычно содержит три необходимых компонента:

1. исходные данные: входные значения и ожидаемый результат;
2. код, вычисляющий значение с помощью тестируемого метода;
3. код, сравнивающий ожидаемый результат с полученным.

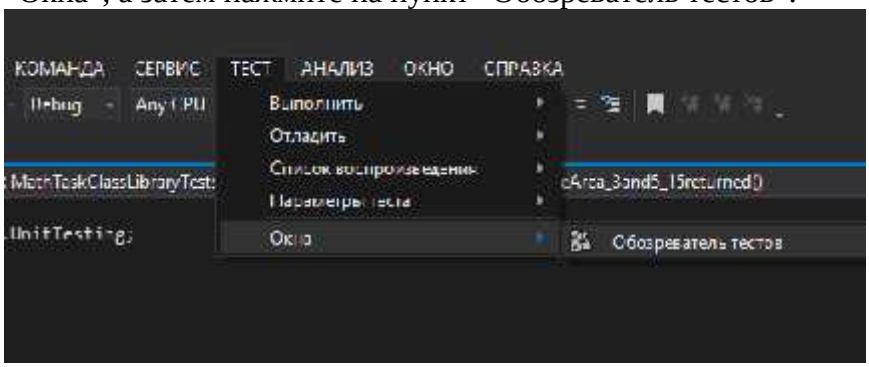
Соответственно тестирующий код будет таким:

```
 1  using System;
 2  using Microsoft.VisualStudio.TestTools.UnitTesting;
 3  using MathTaskClassLibrary;
 4
 5  namespace MathTaskClassLibraryTests
 6  {
 7      [TestClass]
 8      public class GeometryTests
 9      {
10          [TestMethod]
11          public void RectangleArea_3and5_15returned()
12          {
13              // исходные данные
14              int a = 3;
15              int b = 5;
16              int expected = 15;
17
18              // получение значения с помощью тестируемого метода
```

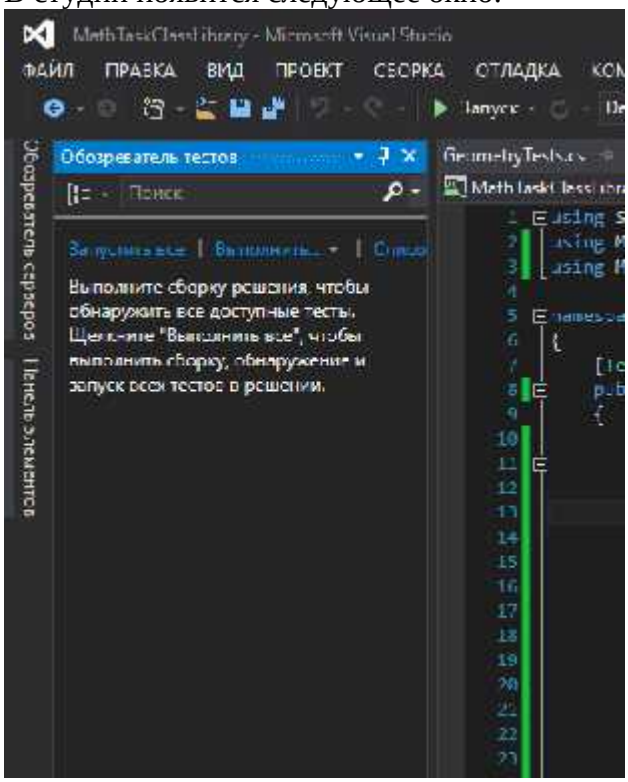
```
19            Geometry g = new Geometry();
20            int actual = g.RectangleArea(a, b);
21
22            // сравнение ожидаемого результата с полученным
23            Assert.AreEqual(expected, actual);
24        }
25    }
26 }
```

Для сравнения ожидаемого результата с полученным используется метод **AreEqual** класса **Assert**. Данный класс всегда используется при написании unit тестов в Visual Studio.

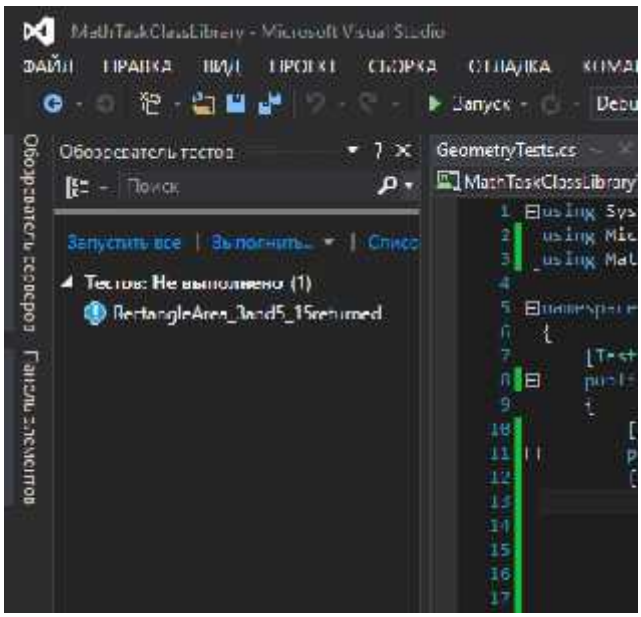
Теперь, чтобы просмотреть все тесты, доступные для выполнения, необходимо открыть окно “Обозреватель тестов”. Для этого в меню Visual Studio щёлкните на кнопку “ТЕСТ”, выберите “Окна”, а затем нажмите на пункт “Обозреватель тестов”.



В студии появится следующее окно:

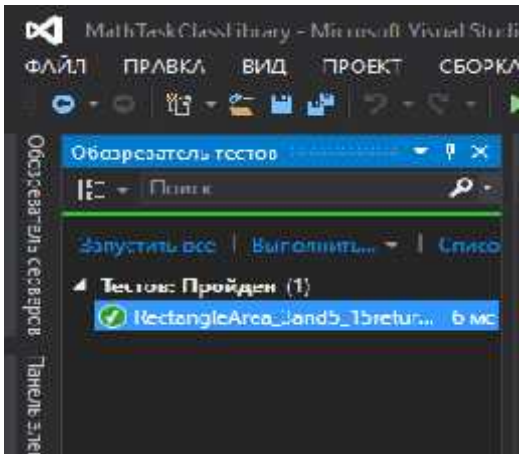


В данный момент список тестов пуст, поскольку решение ещё ни разу не было собрано. Выполним сборку нажатием клавиш Ctrl + Shift + B. После её завершения в “Обозревателе тестов” появится наш тест.



Синяя табличка с восклицательным знаком означает, что указанный тест никогда не выполнялся. Выполним его.

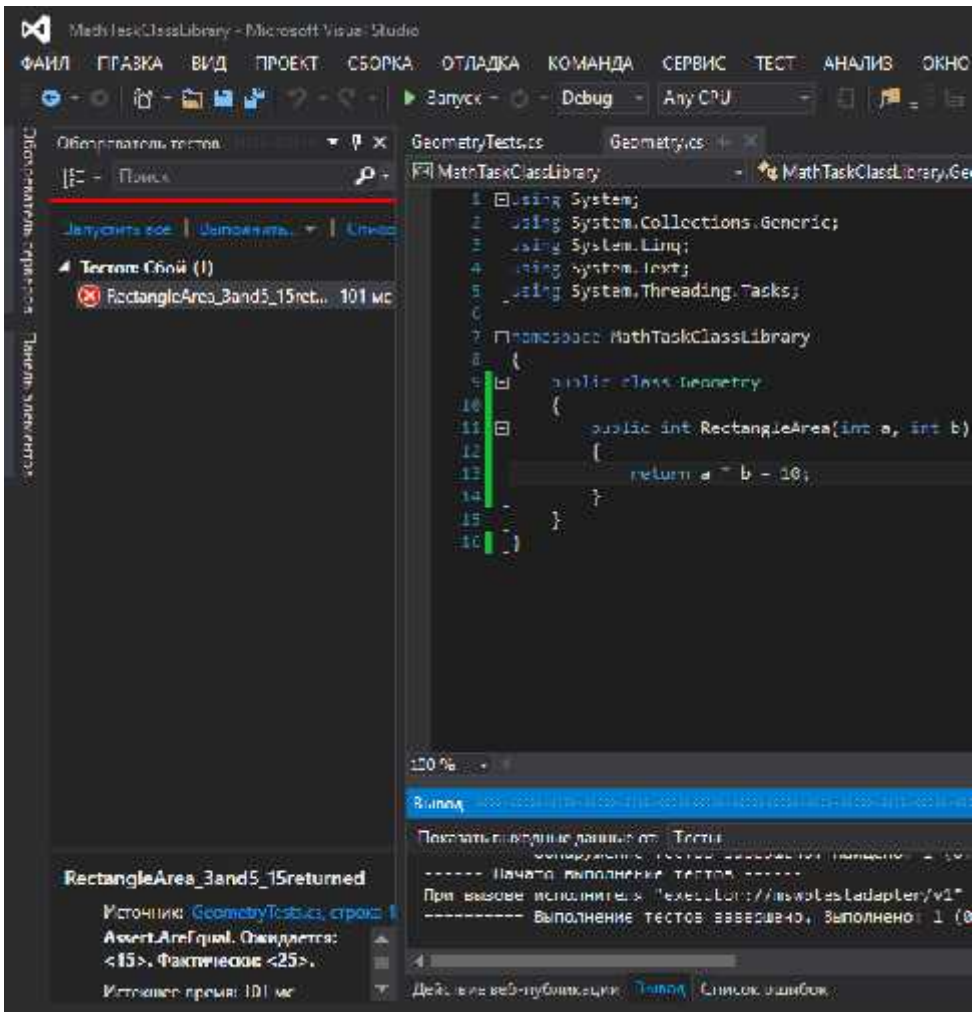
Для этого нажмём правой кнопкой мыши на его имени и выберем “Выполнить выбранные тесты”.



Зелёный кружок с галочкой означает, что модульный тест успешно пройден: ожидаемый и полученный результаты равны.

Изменим код метода **RectangleArea**, вычисляющего площадь прямоугольника, чтобы сымитировать провал теста и посмотреть, как поведёт себя Visual Studio. Прибавим к возвращаемому значению 10.

Запустим unit-тест.



Как Вы видите, красный круг с крестиком показывает провал модульного теста, а ниже указано, что при проверке ожидалось значение 15, а по факту оно равно 25.

Задание 2: Выполнить реализацию следующих методов и написание модульных тестов

Вариант	Метод
1	Метод находит площадь поверхности цилиндра
2	Метод находит стоимость поездки на дачу и обратно на машине
3	Метод находит стоимость покупки из нескольких тетрадок и карандашей
4	Метод находит сопротивление в цепи при параллельном соединении
5	Метод находит объем конуса
6	Метод находит длину отрезка по координатам
7	Метод находит площадь трапеции

Задание 3 .оформить отчет по лабораторной работе, содержащий: титульный лист , задание на выполнение и описание хода работы

Лабораторная работа №4

Тема: Тестирование программного модуля методом стеклянного ящика

Цели и задачи урока: закрепить теоретические знания и получить практические навыки в разработке программы тестирования методами по стратегии «белого ящика»..

Тип урока – практическое занятие.

Методы обучения – *репродуктивный*

Обеспечение занятия – персональный компьютер.

Ход урока

1. Объяснение нового материала
2. Выполнение практической работы
3. Закрепление пройденного материала (ответы на контрольные вопросы)
4. Подведение итогов

Литература

1. Федорова Н .Г., Разработка программных модулей программного обеспечения для компьютерных систем : учебник для сред. проф. образ-я / Г. Н. Федорова. - М. : Академия, 2020. - 336 с. - (Профессиональное образование).
2. Тестирование программного обеспечения: Учебное пособие. / Куликов С. С.. - МИНСК ИЗДАТЕЛЬСТВО «ЧЕТЫРЕ ЧЕТВЕРТИ», 2021. - 314 с.: доступ http:// svyatoslav.biz/software_testing_book/
3. Голицына О.Л., Попов И.И., Программирование на языках высокого уровня Голицына О.Л., Попов И.И., – М.: Форум 2020.
4. Михеева Е.В. Информационные технологии в профессиональной деятельности : учеб. пособие для студ. сред. проф. образования. – 8-е изд., стер. - М.: Издательский центр «Академия», 2020. - 384 с.

Задания

Протестировать методами «белого ящика» программу, вычисляющую функцию, где a, b, c – действительные числа.

Вариант 1

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

Вариант 2

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

Вариант 3

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

Вариант 4

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

Вариант 5

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

Вариант 6

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

Вариант 7

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

Вариант 8

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

Ход выполнеия:

1 Разработать алгоритм решения задачи.

2 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.

3 Результаты тестирования представить в виде таблиц.

4 Оформить отчет по лабораторной работе, включающий разделы:

- Постановка задачи.
- Блок-схема программы.
- Тестовые варианты и результаты тестирования. •

Скриншоты программы.

- Код программы (функции обработчиков событий).
- Вывод.

Лабораторная работа 5

Тема: Тестирование методом черного ящика

Цели и задачи урока: закрепить теоретические знания и получить практические навыки в разработке программы тестирования методом эквивалентного разбиения классов..

Тип урока – практическое занятие.

Методы обучения – *репродуктивный*

Обеспечение занятия – персональный компьютер.

Ход урока

1. Объяснение нового материала
2. Выполнение практической работы
3. Закрепление пройденного материала (ответы на контрольные вопросы)
4. Подведение итогов

Литература

1. Федорова Н .Г., Разработка программных модулей программного обеспечения для компьютерных систем : учебник для сред. проф. образ-я / Г. Н. Федорова. - М. : Академия, 2020. - 336 с. - (Профессиональное образование).
2. Тестирование программного обеспечения: Учебное пособие. / Куликов С. С.. - МИНСК ИЗДАТЕЛЬСТВО «ЧЕТЫРЕ ЧЕТВЕРТИ», 2021. - 314 с.: доступ http:// svyatoslav.biz/software_testing_book/
3. Голицына О.Л., Попов И.И., Программирование на языках высокого уровня Голицына О.Л., Попов И.И., – М.: Форум 2020.
4. Михеева Е.В. Информационные технологии в профессиональной деятельности : учеб. пособие для студ. сред. проф. образования. – 8-е изд., стер. - М.: Издательский центр «Академия», 2020. - 384 с.

Теоретические сведения

закрепить теоретические знания и получить практические навыки в разработке программы тестирования методом эквивалентного разбиения классов.

Задания

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.

Ход выполнения

TortoiseSVN — это бесплатный Windows-клиент с открытыми исходным кодом для системы управления версиями Apache™ Subversion®. То есть TortoiseSVN управляет файлами и директориями во времени. Файлы хранятся в центральномхранилище. Хранилище больше похоже на обычный файловый сервер

Методические указания

1 Разработать алгоритм программы.

2 Разработать визуальное приложение, осуществляющее тестирование программы методом классов эквивалентности: •построение дерева области разбиения данных; •определение количества тестовых вариантов (количество листьев дерева); •генерация тестовых вариантов; •представление таблиц результатов тестирования эталонной и проверяемой программ. 3 Оформить отчет по лабораторной работе, включающий разделы: •Постановка задачи. •Блок-схема программы. •Дерево разбиения области данных с пронумерованными листьями. •Таблицы результатов тестирования. •Скриншоты программы. •Код программы (функции обработчиков событий). •Выводы.

Контрольные вопросы

1 Что называется функциональным тестированием?

2 Что называется классом эквивалентности? 3 Какие правила формирования классов эквивалентности?

4 Какие этапы включает алгоритм метода эквивалентного разбиения классов? 5 В чём сущность метода анализ граничных значений? 6 Какие правила метода анализ граничных значений?

7 Что называется деревом разбиения области данных?

8 Что такое предусловия и постусловия?

Лабораторная работа 6
Тема: Тестирование программного модуля с использованием уровня покрытия операторов

Цели и задачи урока: Закрепить навыки тестирования приложения методом белого ящика.

Тип урока – практическое занятие.

Методы обучения – *репродуктивный*

Обеспечение занятия – персональный компьютер.

Ход урока

1. Объяснение нового материала
2. Выполнение практической работы
3. Закрепление пройденного материала (ответы на контрольные вопросы)
4. Подведение итогов

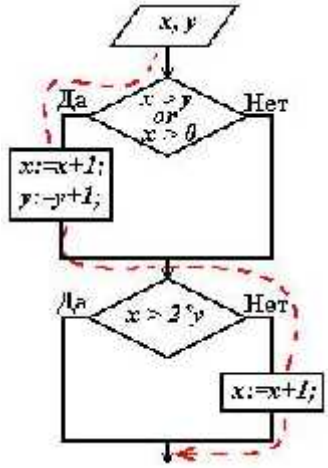
Литература

1. Федорова Н .Г., Разработка программных модулей программного обеспечения для компьютерных систем : учебник для сред. проф. образ-я / Г. Н. Федорова. - М. : Академия, 2020. - 336 с. - (Профессиональное образование).
2. Тестирование программного обеспечения: Учебное пособие. / Куликов С. С.. - МИНСК ИЗДАТЕЛЬСТВО «ЧЕТЫРЕ ЧЕТВЕРТИ», 2021. - 314 с.: доступ http:// svyatoslav.biz/software_testing_book/
3. Голицына О.Л., Попов И.И., Программирование на языках высокого уровня Голицына О.Л., Попов И.И., – М.: Форум 2020.
4. Михеева Е.В. Информационные технологии в профессиональной деятельности : учеб. пособие для студ. сред. проф. образования. – 8-е изд., стер. - М.: Издательский центр «Академия», 2020. - 384 с.

Теоретические сведения

1. Метод покрытия операторов. Этот метод требует написания такого количества тестов, чтобы при выполнении их всех каждый оператор был выполнен хотя бы один раз.

Рассмотрим для примера постенькую блок-схему



Очевидно, что тест ***x=1, y=1*** покроет все операторы, т. к. вычисление поедет по траектории, помеченной красным пунктиром.

Задания

Составить задачу, и протестировать ее методом покрытия операторов

Варианты

1. Составьте программу, которая определяет, принадлежит ли точка A(x_0, y_0) графику функции y=2x-3.

2. 8. Составить программу, которая определяет возможность существования треугольника по трем введенным сторонам. *Треугольник существует только тогда, когда сумма любых двух его сторон больше третьей*.

10. Составить программу вычисления квадратного уравнения ax^2+bx+c. Коэффициенты a,b,c вводятся с клавиатуры.

11. Даны два числа, не равных нулю. Определить имеют ли эти числа одинаковые знаки.

12. Написать программу вычисления стоимости покупки с учетом скидки. Скидка в 3% предоставляется в том случае, если сумма покупки больше 1000 руб., в 5% - если сумма больше 1500 руб.

13. Составьте программу определения большего из двух чисел, введенных с клавиатуры.

Лабораторная работа 7

Тема: Тестирование программного модуля с использованием уровня покрытия ветвей

Цели и задачи урока: Изучить особенностей спецификаций.

Тип урока – практическое занятие.

Методы обучения – *репродуктивный*

Обеспечение занятия – персональный компьютер.

Ход урока

1. Объяснение нового материала
2. Выполнение практической работы
3. Закрепление пройденного материала (ответы на контрольные вопросы)
4. Подведение итогов

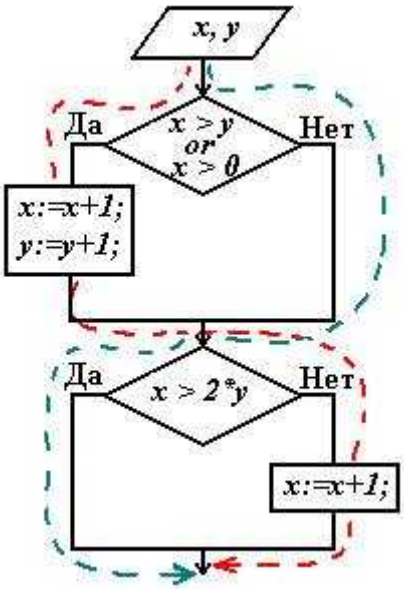
Литература

1. Федорова Н .Г., Разработка программных модулей программного обеспечения для компьютерных систем : учебник для сред. проф. образ-я / Г. Н. Федорова. - М. : Академия, 2020. - 336 с. - (Профессиональное образование).
2. Тестирование программного обеспечения: Учебное пособие. / Куликов С. С.. - МИНСК ИЗДАТЕЛЬСТВО «ЧЕТЫРЕ ЧЕТВЕРТИ», 2021. - 314 с.: доступ http:// svyatoslav.biz/software_testing_book/
3. Голицына О.Л., Попов И.И., Программирование на языках высокого уровня Голицына О.Л., Попов И.И., – М.: Форум 2020.
4. Михеева Е.В. Информационные технологии в профессиональной деятельности : учеб. пособие для студ. сред. проф. образования. – 8-е изд., стер. - М.: Издательский центр «Академия», 2020. - 384 с.

Теоретические сведения

Метод покрытия решений (путей). Под ***решениями*** здесь подразумеваются высказывания в операторых условного перехода, выбора, цикла, взятые целиком. Эти всказывания часто составлены из более простых – элементарных высказываний, которые мы будем до концы параграфа называть ***условиями***. Так, в примере “***x>y or x>0”*** – решение, а его части “***x>y***” и “***x>0”*** – условия. Метод покрытия решений требует такого количества тестов, чтобы при выполнении их всех по каждой траектории, соединяющей соседние элементы блок-схемы вычисление прошло хотя бы один раз. Это означает, что каждое решение должно принимать

как истинные, так и ложные значения. Именно это обеспечивает использование всех путей, выходящих из точек ветвления. Что касается операторов выбора, к ним это также относится.
Для того, чтобы сохранит общность рассуждений, надо только переделать их (мысленно) в эквивалентную цепочку условных операторов.
Для нашего примера к тесту ***x=1, y=1*** надо добавить еще один. Например, ***x = -3, y = -2***.



Для него вычисление пойдет по зеленой траектории и все пути будут покрыты.

Задания

Составить задачу, и протестировать ее методом покрытия операторов

Варианты

1. Составьте программу, которая определяет принадлежность точки к интервалу (-1, 6).

2. Составить программу, которая определяет сумму только положительных из введенных трех чисел.

3. Составьте программу, удваивающую значение переменной x, если x>7.

4. Используя оператор if…then…else составьте программу, которая бы в ответ на введённую оценку по информатике выводила на экран следующий текст:
5. если оценка «5», то «молодец, я тобой горжусь!»(1)
6. если оценка «4», то «я рад, надеюсь, будет «5»» (2)
7. если оценка «3», то «не ленись и всё получится» (3)
8. иначе «ты, наверное, не ходишь на уроки» (4)

9. Составьте программу, удваивающую значение целой переменной a, если a<10 и утраивающую значение переменной, если a>=10.

10. Дано два числа. Вычесть от большего меньшее и результат вывести на экран.

Лабораторная работа 8
Тема: Тестирование программного модуля с использованием уровня покрытия условий

Цели и задачи урока: Изучить особенностей спецификаций.

Тип урока – практическое занятие.

Методы обучения – *частично поисковый*

Обеспечение занятия – персональный компьютер.

Ход урока

1. Объяснение нового материала
2. Выполнение практической работы
3. Закрепление пройденного материала (ответы на контрольные вопросы)
4. Подведение итогов

Литература

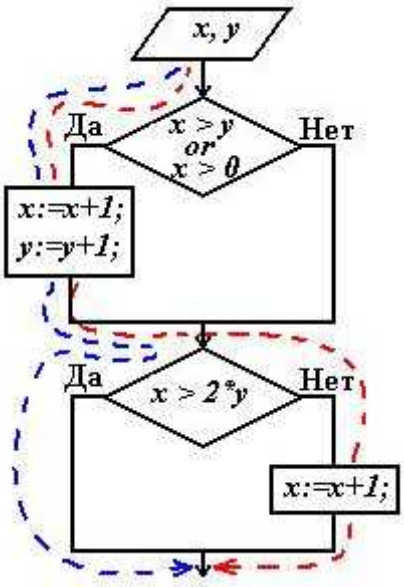
1. Федорова Н .Г., Разработка программных модулей программного обеспечения для компьютерных систем : учебник для сред. проф. образ-я / Г. Н. Федорова. - М. : Академия, 2020. - 336 с. - (Профессиональное образование).
2. Тестирование программного обеспечения: Учебное пособие. / Куликов С. С.. - МИНСК ИЗДАТЕЛЬСТВО «ЧЕТЫРЕ ЧЕТВЕРТИ», 2021. - 314 с.: доступ http:// svyatoslav.biz/software_testing_book/
3. Голицына О.Л., Попов И.И., Программирование на языках высокого уровня Голицына О.Л., Попов И.И., – М.: Форум 2020.
4. Михеева Е.В. Информационные технологии в профессиональной деятельности : учеб. пособие для студ. сред. проф. образования. – 8-е изд., стер. - М.: Издательский центр «Академия», 2020. - 384 с.

Теоретические сведения

Метод покрытия условий. В нашем примере имеется решение ***(x > y) or (x > 0)***, состоящее из двух условий "***x > y***" и "***x > 0"***. Для первого теста в п.2 это решение истинно, для второго теста – ложно. Для первого теста оно истинно потому, что истинна его вторая его часть. Для второго теста оно ложно, потому, что оба суждения ложны. Получается, что для обоих тестов условие ***x > y*** ложно. Такое тестирование явно ущербно.

Метод покрытия условий состоит в таком подборе тестов, когда каждое условие (элементарное суждения в условных операторах) принимает как истинное так и ложное значение.

В нашем примере имется три условия: "***x > y***", "***x > 0" и "x > 2*y***". Тесты: ***x = 1, y = 1*** и ***x = 0, y = -1*** покрывают все эти условия.



Но, как видим, этот метод не гарантирует покрытия решений.

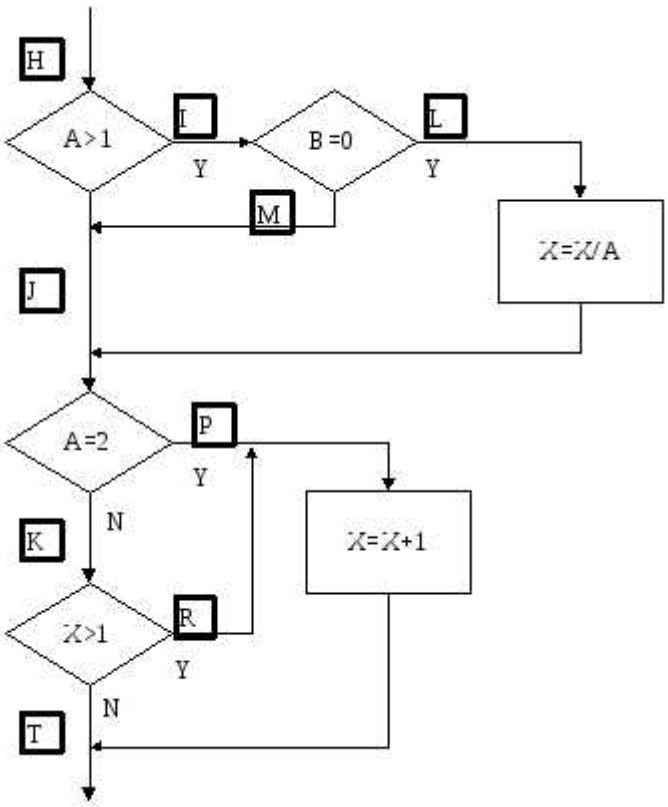
Задания

Заданы следующие схемы. Преобразовать их в программные коды и провести тестирование методом покрытия условий. Заполнить соответствующею таблицу

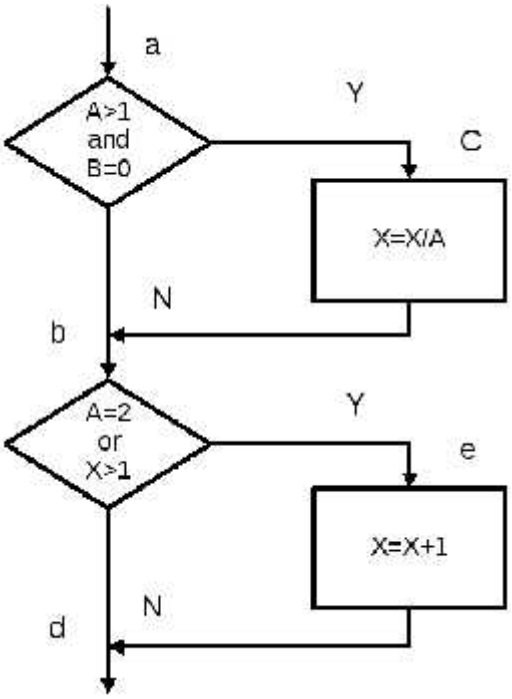
Результаты тестирования методом комбинаторного покрытия условий

Тест	Ожидаемый результат	Фактический результат	Результат тестирования

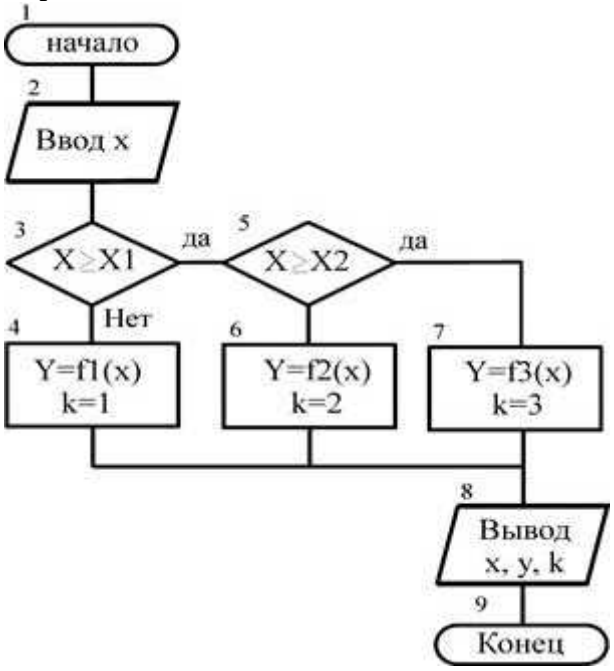
Вариант 1



Вариант 2


a
A>1
and
B=0
Y
C
X=X/A
N
b
A=2
or
X>1
Y
e
X=X+1
N
d

Вариант 3


1
начало
2
Ввод x
3
X≥X1
да
5
X≥X2
да
4
Нет
Y=f1(x)
k=1
6
Y=f2(x)
k=2
7
Y=f3(x)
k=3
8
Вывод
x, y, k
9
Конец

Лабораторная работа №9

Тема: Разработка плана тестирования по Rational Unified Process

Цели и задачи урока: Научиться составлять план тестирования.

Тип урока – практическое занятие.

Методы обучения – *частично поисковый*

Обеспечение занятия – персональный компьютер.

Ход урока

1. Объяснение нового материала
2. Выполнение практической работы
3. Закрепление пройденного материала (ответы на контрольные вопросы)
4. Подведение итогов

Литература

1. Федорова Н .Г., Разработка программных модулей программного обеспечения для компьютерных систем : учебник для сред. проф. образ-я / Г. Н. Федорова. - М. : Академия, 2020. - 336 с. - (Профессиональное образование).
2. Тестирование программного обеспечения: Учебное пособие. / Куликов С. С.. - МИНСК ИЗДАТЕЛЬСТВО «ЧЕТЫРЕ ЧЕТВЕРТИ», 2021. - 314 с.: доступ http:// svyatoslav.biz/software_testing_book/
3. Голицына О.Л., Попов И.И., Программирование на языках высокого уровня Голицына О.Л., Попов И.И., – М.: Форум 2020.
4. Михеева Е.В. Информационные технологии в профессиональной деятельности : учеб. пособие для студ. сред. проф. образования. – 8-е изд., стер. - М.: Издательский центр «Академия», 2020. - 384 с.

Теоретические сведения

Rational Unified Process - это методология создания программного обеспечения, RUP не ставит своей целью добиться абсолютного качества разрабатываемого продукта. RUP вообще не призывает к достижению недостижимых целей. В частности, этим объясняются и принятая в RUP концепция фаз. В идеале, хорошо бы сначала все проанализировать, потом спроектировать и только потом взяться за программирование. И быть уверенным, что ничего не придется переделывать, потому, что все качественно проанализировано и спроектировано. К сожалению, это недостижимый идеал, как и абсолютное качество программного обеспечения. Как гласит программистская мудрость, каждая обнаруженная в программе ошибка, в лучшем случае, предпоследняя...

Значит ли это, что можно сдавать Заказчику систему "как есть", не задумываясь о том, может ли он ею воспользоваться или ошибки не позволяют этого? Нет, ни в коем случае! RUP предлагает использовать при оценке качества произведенного продукта понятие "достаточно хорошего качества". Использование этого понятия означает, что Разработчик с открытыми глазами оценивает качество продукта, который он представляет Заказчику. И, как минимум, уверен, что поиск и устранение следующей ошибки сейчас обойдутся дороже, чем возможные потери Заказчика при проявлении ошибки и затраты на ее устранение в будущем. Впрочем, для критических систем критерий качественного продукта может быть и более жестким. Важно, что критерий есть, и что он должен быть выполнен. И что при этом качественный продукт — это не обязательно продукт БЕЗ ЕДИНОЙ ошибки.

Достижение достаточно хорошего качества невозможно без тщательного анализа статистических характеристик результатов тестирования. И это тоже одна из принципиальных особенностей подхода RUP.

Теперь, зная, "что такое хорошо и что такое плохо", прейдем к тому, как же "делать хорошо и не делать плохо".

Как следует из особенностей **RUP**, тестирование должно проводиться практически во всех итерациях. Однако цели и задачи группы тестирования на различных итерациях и, тем более, в различных фазах разработки могут существенно различаться.

В фазе Начало задача группы тестирования — подготовиться к последующей работе по проекту. Понять назначение системы и ориентировочный объем задач тестирования. Подготовить необходимый инструментарий. Продумать критерии качества и критерии завершения тестирования, которые необходимо применить в данном случае.

Есть еще одна работа, о которой часто забывают. В этой фазе тестировщики должны оценить в наиболее общих чертах подход к созданию системы и сформулировать требования, которые необходимо выполнять всем участникам разработки для обеспечения простоты тестирования системы. Например, подготовка специальных имитирующих программ, генераторов тестовых данных или использование инструментальных средств и сред, поддерживаемых имеющимися инструментами тестирования.

В фазе Разработка, как правило, не создаются полностью завершенные фрагменты системы. Поэтому нет смысла тратить время на проверку чистоты графического интерфейса, отработку всех исключительных ситуаций и другие проблемы, которые еще не решались. Тем не менее, их имеет смысл фиксировать и сообщать о них проектировщикам и программистам. Возможно, это поможет.

Основной задачей тестирования в этой фазе является тестирование архитектуры системы. Проводятся наиболее принципиальные проверки производительности системы, ее устойчивости к нагрузкам, надежности выбранного метода взаимодействия между компонентами системы и т.п. Если, как часто бывает в жизни, отложить эти вопросы "на потом", на тот момент, когда система будет уже практически готова то в результате за неделю до сдачи системы Заказчику может оказаться, что она явно не удовлетворяет требованиям к производительности. Нужно переделывать всю схему обработки информации, а времени и ресурсов на это нет.

Фаза Построение — это фаза, в которой тестировщики должны в каждой итерации проверять все более полное выполнение системой требований Заказчика. Основной особенностью работы тестировщиков на этой фазе является необходимость многократно (обычно, в каждой итерации) проверять практически все модули разрабатываемой системы. Ведь в них самих были внесены изменения и дополнения, либо они должны взаимодействовать с измененными или доработанными модулями. Таким образом, в условиях итерационной разработки существенно возрастает необходимый объем тестирования. При этом наиболее массовым становится регрессионное тестирование, на которое приходится основной объем выполняемых тестов. Однако регрессионное тестирование во многом сводится к повторению ранее разработанных тестов. Это позволяет эффективно использовать для регрессионного тестирования средства автоматизации, позволяющие проводить тестирование с минимальным участием тестировщиков. В результате из недостатка большой объем регрессионного тестирования превращается в достоинство. Наиболее принципиальные фрагменты системы, разрабатываемые первыми, проходят несколько циклов тестирования и передаются Заказчику более качественными.

Программный продукт создается в последовательных итерациях. В каждой итерации коллектив разработчиков выполняет несколько сборок программы. Каждая сборка является потенциальным кандидатом для тестирования. Итерация завершается выпуском внутренней версии программы. Эта версия проходит обязательное тестирование. Для каждой версии могут разрабатываться или уточняться тесты. Поскольку процесс разработки является инкрементным (каждая новая итерация добавляет новые возможности разрабатываемой системы), тесты, используемые на ранних итерациях, как правило, могут быть использованы и на последующих для регрессионного тестирования, то есть для проверки того, что ранее реализованная функциональность системы сохранилась в новой итерации. Таким образом, каждая новая итерация подразумевает повторное тестирование всех компонентов, разработанных в предыдущих итерациях, плюс тестирование новых компонентов.

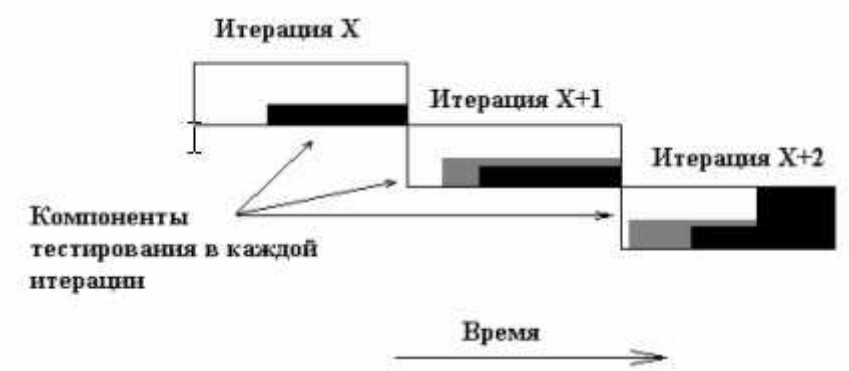


Рисунок 4. Пример тестируемых компонент на различных итерациях

Итерационный подход позволяет повысить качество системы за счет многократного регрессионного тестирования ключевых компонентов системы. Так как на каждой итерации нужно повторять ранее проводившиеся тесты, то желательно иметь определенный набор инструментальных средств, которые обеспечивают автоматизированное тестирование ранее разработанных компонентов.

Еще одна существенная особенность работы в этой и последующей фазах, как уже упоминалось — необходимость собирать статистику обнаруженных и исправленных дефектов. Именно эта статистика, с одной стороны, позволяет оптимально перераспределить силы разработчиков и тестировщиков между модулями и компонентами системы. А с другой стороны, она служит основой для принятия обоснованных решений о достижении продуктом требуемого качества. Сбор достаточного объема статистики, как правило, требует использования средств автоматизации тестирования.

Также уже в фазе Построение желательно наладить активное взаимодействие с Заказчиком и будущими пользователями системы. Без обратной связи с будущими пользователями вряд ли вам удастся разработать не то, что "великую", но хотя бы "приличную" систему. Какой родитель скажет, что у его ребенка столько недостатков, сколько их видит сосед!?

В начале фазы такое взаимодействие может происходить в форме демонстрации системы "из своих рук". По мере совершенствования системы следует предоставлять пользователям все большую свободу в обращении с системой. Конечно, это потребует от всех разработчиков, и от тестировщиков в частности, дополнительных усилий по, хотя бы, минимальному обучению пользователей и сбору и анализу их предложений и замечаний, но результат обычно того стоит.

Передача. В этой фазе тестирование может приобретать специфические формы. Это могут быть и формальные приемочные испытания, и бета тестирование в той или иной форме. Проводить эти виды тестирования в той или иной мере (полностью или частично) может сам Заказчик или третья фирма по поручения Заказчика.

К сожалению, даже при самом высоком качестве разработки редко удается избежать в фазе Передачи внесения изменений в разрабатываемую систему. Хотя в этой фазе желательно ограничиваться минимальными доработками, связанными скорее с "отсечением" недостаточно качественных фрагментов кода, иногда приходится возвращаться и к анализу и проектированию, не говоря уж про разработку. В этом случае приходится возвращаться к автономным и комплексным проверкам доработанных модулей и системы в целом. Более того, придется вернуться к отладке модуля. Отладка модуля, которую наиболее эффективно может провести разработчик, не является тестированием по RUP.

Задания

Используя шаблон для составления плана тестирования rup и макета составить план тестирования для курсового проекта

Лабораторная работа 10
Тема: Разработка плана тестирования по IEEE 829

Цели и задачи урока: Научиться составлять план тестирования

Тип урока – практическое занятие.

Методы обучения – *частично поисковый*

Обеспечение занятия – персональный компьютер.

Ход урока

1. Объяснение нового материала
2. Выполнение практической работы
3. Закрепление пройденного материала (ответы на контрольные вопросы)
4. Подведение итогов

Литература

1. Федорова Н .Г., Разработка программных модулей программного обеспечения для компьютерных систем : учебник для сред. проф. образ-я / Г. Н. Федорова. - М. : Академия, 2020. - 336 с. - (Профессиональное образование).
2. Тестирование программного обеспечения: Учебное пособие. / Куликов С. С.. - МИНСК ИЗДАТЕЛЬСТВО «ЧЕТЫРЕ ЧЕТВЕРТИ», 2021. - 314 с.: доступ http:// svyatoslav.biz/software_testing_book/
3. Голицына О.Л., Попов И.И., Программирование на языках высокого уровня Голицына О.Л., Попов И.И., – М.: Форум 2020.
4. Михеева Е.В. Информационные технологии в профессиональной деятельности : учеб. пособие для студ. сред. проф. образования. – 8-е изд., стер. - М.: Издательский центр «Академия», 2020. - 384 с.

Теоретические сведения

TortoiseSvn

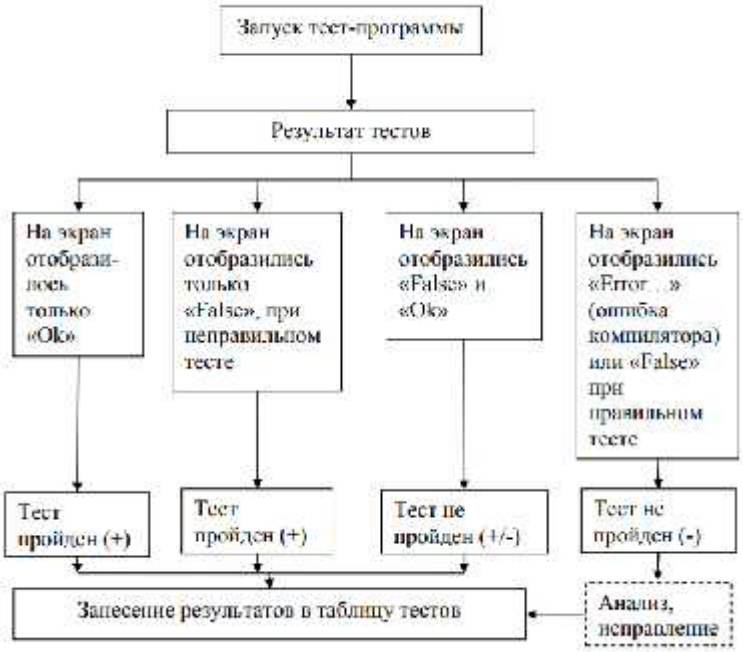
TortoiseSVN — это бесплатный Windows-клиент с открытыми исходным кодом для системы управления версиями Apache™ Subversion®. То есть TortoiseSVN управляет файлами и директориями во времени. Файлы хранятся в центральномхранилище. Хранилище больше похоже на обычный файловый сервер

Задания

Выполнить тестирование компиляторов языка программирования Си на соответствие стандарту ANSI ISO/IEC 9899:1999 в части декларативных операторов. На рисунке 1 показана постановка задачи. Для выполнения тестирования используется набор тестовых вариантов. Результаты тестирования приведены на рисунке 2 и сводятся в таблицу 1. Принимается решение о соответствии компилятора стандарту.

Таблица 1 – Результаты тестирования

Тест	Результаты тестирования	
	Компилятор языка	Результат



Компилятор языка программирования C# выбирается студентом или выдается преподавателем. Набор тестовых вариантов выдается преподавателем.

Методические указания

1 Выбрать компилятор языка C#.

2 Получить набор тестовых вариантов у преподавателя: •запустить тестовые варианты; •занести результаты выполнения тестовых вариантов в таблицу; •представить результаты тестирования в графическом виде (диаграммы); •провести анализ результатов тестирования;

3 Оформить отчет по лабораторной работе, включающий разделы: Запуск тест-программы Результат тестов На экран отобразилось только «Ok» На экран отобразились только «False», при неправильном тесте На экран отобразились «False» и «Ok» На экран отобразились «Error…» (ошибка компилятора) или «False» при правильном тесте Тест пройден (+) Тест пройден (+) Тест не пройден (+/-) Тест не пройден (-) Занесение результатов в таблицу тестов Анализ, исправление

7 •Постановка задачи.

•Таблица выполнения тестовых вариантов.

•Графическое представление результатов тестирования.

•Выводы.

Лабораторная работа №11

Тема: Оформление документа баг-дефект репорта

Цели и задачи урока: Закрепить навыки составления баг-дефект репорта.

Тип урока – практическое занятие.

Методы обучения – *частично поисковый*

Обеспечение занятия – персональный компьютер.

Ход урока

1. Объяснение нового материала
2. Выполнение практической работы
3. Закрепление пройденного материала (ответы на контрольные вопросы)
4. Подведение итогов

Литература

1. Федорова Н .Г., Разработка программных модулей программного обеспечения для компьютерных систем : учебник для сред. проф. образ-я / Г. Н. Федорова. - М. : Академия, 2020. - 336 с. - (Профессиональное образование).
2. Тестирование программного обеспечения: Учебное пособие. / Куликов С. С.. - МИНСК ИЗДАТЕЛЬСТВО «ЧЕТЫРЕ ЧЕТВЕРТИ», 2021. - 314 с.: доступ http:// svyatoslav.biz/software_testing_book/
3. Голицына О.Л., Попов И.И., Программирование на языках высокого уровня Голицына О.Л., Попов И.И., – М.: Форум 2020.
4. Михеева Е.В. Информационные технологии в профессиональной деятельности : учеб. пособие для студ. сред. проф. образования. – 8-е изд., стер. - М.: Издательский центр «Академия», 2020. - 384 с.

Теоретические сведения

Баг репорт - **это технический документ** и в связи с этим хотим отметить, что язык описания проблемы должен быть техническим. Должна использоваться правильная терминология при использовании названий элементов пользовательского интерфейса (editbox, listbox, combobox, link, text area, button, menu, popup menu, title bar, system tray и т.д.), действий пользователя (click link, press the button, select menu item и т.д.) и полученных результатах (window is opened, error message is displayed, system crashed и т.д.).

Требования к обязательным полям баг репорта

Отметим, что обязательными полями баг репорта являются: **короткое описание** (*Bug Summary*), **серьезность** (*Severity*), **шаги к воспроизведению** (*Steps to reproduce*), **результат** (*Actual Result*), **ожидаемый результат** (*Expected Result*). Ниже приведены требования и примеры по заполнению этих полей.

Короткое описание

Название говорит само за себя. В одном предложении вам надо уместить смысл всего баг репорта, а именно: коротко и ясно, используя правильную терминологию сказать что и где не работает. Например:

1. Приложение зависает, при попытке сохранения текстового файла размером больше 50Мб.
2. Данные на форме "Профайл" не сохраняются после нажатия кнопки "Сохранить".

В дополнение предлагаем вам изучить **Принцип "Где? Что? Когда?"**, описанный на страницах блога **"QA Nest"**:

"В чем этот принцип состоит?

Составьте предложение, в котором факты дефекта изложены в следующей последовательности:

- ***Где?**: В каком месте интерфейса пользователя или архитектуры программного продукта находится проблема. Причем, начинайте предложение с существительного, а не предлога.*

- ***Что?****: Что происходит или не происходит согласно спецификации или вашему представлению о нормальной работе программного продукта. При этом указывайте на наличие или отсутствие объекта проблемы, а не на его содержание (его указывают в описании). Если содержание проблемы варьируется, все известные варианты указываются в описании.*
- ***Когда?****: В какой момент работы программного продукта, по наступлению какого события или при каких условиях проблема проявляется.*

Почему последовательность должна быть именно такой?
В таком виде незнакомые дефекты удобнее сортировать по summary как показывает практика (ведь, скорее всего, именно среди дефектов других инженеров будет производиться поиск дубликатов). Если вы другого мнения - придумайте свою последовательность, но она должна стать единой для всех без исключения членов проекта, иначе вы не добьетесь необходимого результата."

Серьезность

В двух словах можно отметить, что если проблема найдена в ключевой функциональности приложения и после ее возникновения приложение становится полностью недоступно, и дальнейшая работа с ним невозможна, то она **блокирующая**. Обычно все блокирующие проблемы находятся во время первичной проверки новой версии продукта (**Build Verification Test**, **Smoke Test**), т.к. их наличие не позволяет полноценно проводить тестирование. Если же тестирование может быть продолжено, то серьезность данного дефекта будет **критическая**. На счет **значительных**, **незначительных** и **тривиальных** ошибок вопрос достаточно прозрачный и на наш взгляд не требует лишних объяснений.

Шаги к воспроизведению / Результат / Ожидаемый результат

Очень важно четко описать все шаги, с упоминаем всех вводимых данных (имени пользователя, данных для заполнения формы) и промежуточных результатов.
Например:

Шаги к воспроизведению

1. Войдите в системы: Пользователь Тестер1, пароль xxxXXX

--> Вход в систему осуществлен

2. Кликните линк Профайл

--> Страница Профайл открылась

3. Введите Новое имя пользователя: Тестер2

4. Нажмите кнопку Сохранить

Результат

На экране появилась ошибка. Новое имя пользователя не было сохранено

Ожидаемый результат

Страница профайл перегрузилась. Новое значение имени пользователя сохранено.

Основные ошибки при написании багов репортов

Недостаточность предоставленных данных

Не всегда одна и таже проблема проявляется при всех вводимых значениях и под любым вошедшим в систему пользователем, поэтому настоятельно рекомендуется вносить все необходимые данные в баг репорт

Определение серьезности

Очень часто происходит либо завышение, либо занижение серьезности дефекта, что может привести к неправильной очередности при решении проблемы.

Язык описания

Часто при описании проблемы используются неправильная терминология или сложные речевые обороты, которые могут ввести в заблуждение человека, ответственного за решение проблемы.

Отсутствие ожидаемого результата

В случаях, если вы не указали, что же должно быть требуемым поведением системы, вы тратите время разработчика, на поиск данной информации, тем самым замедляете исправления дефекта. Вы должны указать пункт в требованиях, написанный тест кейс или же ваше личное мнение, если эта ситуация не была документирована.

Заполнение полей баг репорта

В описанной ниже таблице представлены основные поля баг репорта и роль работника, ответственного за заполнение данного поля. Красным цветом выделены обязательные для заполнения поля:

Поле	Ответственный за заполнение поля
Короткое описание (Summary)	Автор баг репорта (обычно это Тестировщик)
Проект (Project)	Автор баг репорта (обычно это Тестировщик)
Компонент приложения (Component)	Автор баг репорта (обычно это Тестировщик)
Номер версии (Version)	Автор баг репорта (обычно это Тестировщик)
Серьезность (Severity)	Автор баг репорта (обычно это Тестировщик), однако данный атрибут может быть изменен вышестоящим менеджером
Приоритет (Priority)	Менеджер проекта или менеджер ответственный за разработку компонента, на который написан баг репорт
Статус (Status)	Автор баг репорта (обычно это Тестировщик), но многие системы баг трекинга выставляют статус по умолчанию
Автор (Author)	Устанавливается по умолчанию, если нет, то указывается имя автора баг репорта
Назначен на (Assigned To)	Менеджер проекта или менеджер ответственный за разработку компонента, на который написан баг репорт
ОС / Сервис Пак и т.д. / Браузера + версия / ...	Автор баг репорта (обычно это Тестировщик)
Шаги воспроизведения (Steps to Reproduce)	Автор баг репорта (обычно это Тестировщик)
Фактический Результат	Автор баг репорта (обычно это Тестировщик)

(Result)	
Ожидаемый результат (Expected Result)	Автор баг репорта (обычно это Тестировщик)
Прикрепленный файл (Attachment)	Автор баг репорта (обычно это Тестировщик), а также любой член командной группы, считающий, что прикрепленные данные помогут в исправлении бага

Задания

1. Получить задание у преподавателя.
2. Протестировать web-приложение в соответствии с составленной ранее тестовой документацией.
3. Описать все найденные дефекты.
4. Оформить отчет и защитить лабораторную работу.

Содержание отчета

1. Цель работы.
2. Краткие теоретические сведения.
3. Отчет о найденных дефектах.
4. Выводы по работе. Контрольные вопросы

1. Что такое дефект?
2. Какие характеристики необходимо указать при описании дефекта?
3. Что такое Headline/Summary в описании дефекта?
4. На какие три вопроса должен отвечать Headline/Summary?
5. Что такое Severity в описании дефекта? 6. Какие существуют степени Severity? Приведите примеры.
7. Что такое Description в описании дефекта?
8. Что такое Expected result в описании дефекта?
9. Зачем нужен Attachment при описании дефекта?
10. Какие существуют рекомендации по описанию дефектов? Контрольные вопросы

Лабораторная работа 12

Тема: Оформление документации по тестированию с использованием инструментальных средств

Цели и задачи урока: Изучить особенностей спецификаций.

Тип урока – практическое занятие.

Методы обучения – *частично поисковый*

Обеспечение занятия – персональный компьютер.

Ход урока

1. Объяснение нового материала
2. Выполнение практической работы
3. Закрепление пройденного материала (ответы на контрольные вопросы)
4. Подведение итогов

Литература

1. Федорова Н .Г., Разработка программных модулей программного обеспечения для компьютерных систем : учебник для сред. проф. образ-я / Г. Н. Федорова. - М. : Академия, 2020. - 336 с. - (Профессиональное образование).
2. Тестирование программного обеспечения: Учебное пособие. / Куликов С. С.. - МИНСК ИЗДАТЕЛЬСТВО «ЧЕТЫРЕ ЧЕТВЕРТИ», 2021. - 314 с.: доступ http:// svyatoslav.biz/software_testing_book/
3. Голицына О.Л., Попов И.И., Программирование на языках высокого уровня Голицына О.Л., Попов И.И., – М.: Форум 2020.

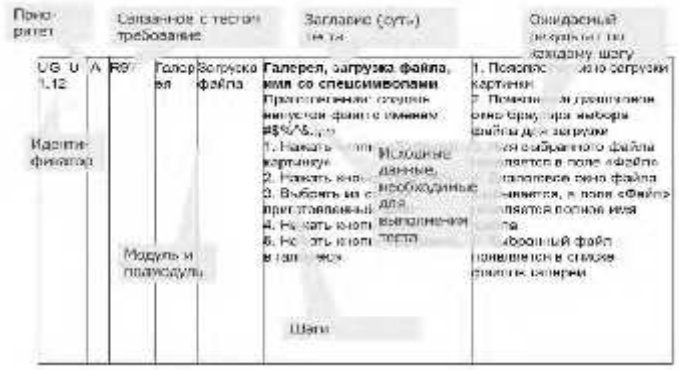
4. Михеева Е.В. Информационные технологии в профессиональной деятельности : учеб. пособие для студ. сред. проф. образования. – 8-е изд., стер. - М.: Издательский центр «Академия», 2020. - 384 с.

Теоретические сведения

Рекомендации по разработке тестов.

Начинайте с простых очевидных тестов.
Затем переходите к более сложным тестам.
Помните о граничных условиях.
Если остаётся время, занимайтесь исследовательским тестированием.
Последовательность разработки и выполнения тестов:
– простые позитивные;
– простые негативные;
– сложные позитивные; – сложные негативные.

Оформление тест-кейсов. Общие идеи по разработке тест-кейсов приведены на рисунке



Задания

Контрольные вопросы

На основе представленного ниже набора требований сформируйте для разрабатываемых приложений: – смоук-тест; – чек-лист для теста критического пути; – тест критического пути (насколько хватит времени – расписывайте идеи из чек-листа в полноценные тесты).

Требования к разрабатываемому приложению 1

Приложение должно выполнять математические вычисления.

1 Приложение должно работать под всеми версиями ОС Windows.

2 Приложение должно быть максимально похоже на стандартный калькулятор Windows (рисунок 2.2) за исключением некоторых особенностей.

3 Несколько приложений должны иметь возможность работать одновременно.

4 При запуске приложения должно отображаться окно со стандартными для калькулятора кнопками и полем ввода и отображения данных.

5 Для начала вычислений пользователь должен нажать кнопку «Начать».

6 Приложение должно позволять легко сохранять вычисления в выбранном пользователем формате.

7 Опционально предусматривается поддержка нескольких языков.

8 Приложение должно позволять выполнять вычисления сразу же после запуска.

9 Скорость вычислений должна быть максимально высокой.

10 Приложение должно позволять выполнять следующие операции: сложение, умножение, вычитание и деление чисел.

11 Приложение должно позволять строит графики простых функций.

12 Приложение должно запрашивать подтверждение («Результат не сохранён. Выйти?») в случае, если пользователь не сохранил результаты работы.

1 Приложение должно работать под версиями ОС Windows7 и Windows 8.

2 Несколько приложений должны иметь возможность работать одновременно, т. е. можно открыть несколько калькуляторов и вести в них независимые вычисления.

3 При запуске приложения должно отображаться окно с кнопками калькулятора (рисунок 2.2) и полем отображения данных.

4 Данные в приложение могут вводиться как с помощью кнопок приложения, так и с помощью клавиатуры.

5 Приложение должно позволять сохранять вычисления во внешний файл с расширением, задаваемым пользователем.

6 Должна быть предусмотрена поддержка английского и русского языков. Отображается тот язык, который выбран в ОС по умолчанию.

7 Вычисления должны производиться со скоростью не более 1 с.

8 Приложение должно позволять выполнять следующие операции: сложение, умножение, вычитание и деление чисел, взятие квадратного корня, возведение в степень, вычисление процентов, ввод отрицательного числа.

Вопросы:

1 В какой последовательности рекомендуется разрабатывать тесты?

2 Что такое смоук-тест? Приведите пример.

3 Что такое чек-лист?

4 Перечислите элементы тест-кейса.

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