МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Пятигорский институт (филиал) СКФУ

МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ВЫПОЛНЕНИЮ ЛАБОРАТОРНЫХ РАБОТ ПО ДИСЦИПЛИНЕ «ВЕРИФИКАЦИЯ И АНАЛИЗ ПРОГРАММ»

Направление подготовки	**09.04.02 Информационные системы и технологии**
Направленность (профиль)	**«Технологии работы с данными и знаниями, анализ информации»**
Квалификация выпускника	Магистр

Пятигорск, 2025

СОДЕРЖАНИЕ

1. ЦЕЛЬ И ЗАДАЧИ ОСВОЕНИЯ ДИСЦИПЛИНЫ (МОДУЛЯ)

Целью освоения дисциплины «Верификация и анализ программ» является получение устойчивых навыков самостоятельного программирования с применением современных программных средств для разработки, верификации, анализа и тестирования программ.

В соответствии с указанной целью при изучении дисциплины ставятся следующие задачи:

- привить навыки работы в среде визуального программирования,
- дать сведения о принципах, стратегиях и этапах верификации программ;
- изучить основные методы верификации, анализа и тестирования программ.

2. НАИМЕНОВАНИЕ ЛАБОРАТОРНЫХ РАБОТ

Лабораторная работа 1. Использование операторов протоколирования при создании и отладке программ
Лабораторная работа 2. Отладка программ путем пошагового выполнения кода
Лабораторная работа 3. Отладка программ с установкой контрольных точек и анализом памяти
Лабораторная работа 4. Анализ качества программы путем тестирования ее функциональных возможностей.

3. СОДЕРЖАНИЕ ЛАБОРАТОРНЫХ РАБОТ

Лабораторная работа 1

Использование операторов протоколирования при создании и отладке программ

Цель и содержание: научиться использовать операторы протоколирования при создании и отладке программ.

Организационная форма занятий: практикум.

Вопросы для обсуждения на лабораторном занятии: использование операторов протоколирования при создании и отладке программ.

Аппаратура и материалы. Для выполнения задания необходим персональный компьютер, а также соответствующее лабораторной работе программное обеспечение.

Указания по технике безопасности. Самостоятельно не производить: установку и удаление программного обеспечения, ремонт персонального компьютера. Соблюдать правила технической безопасности при работе с электрооборудованием.

Теоретическое обоснование

Организация тестирования

Тестирование осуществляется на заданном заранее множестве входных данных X и множестве предполагаемых результатов Y – (X,Y), которые задают график желаемой функции. Кроме того, зафиксирована процедура Оракул (oracle), которая определяет, соответствуют ли выходные данные – Yв (вычисленные по входным данным – X) желаемым результатам – Y, т.е. принадлежит ли каждая вычисленная точка (X,Yв) графику желаемой функции (X,Y).

Оракул дает заключение о факте появления неправильной пары (X,Yв) и ничего не говорит о том, каким образом она была вычислена или каков правильный алгоритм – он только сравнивает вычисленные и желаемые результаты. Оракулом может быть даже Заказчик или программист, производящий соответствующие вычисления в уме, поскольку Оракулу нужен какой-либо альтернативный способ получения функции (X,Y) для вычисления эталонных значений Y.

Пример сравнения словесного описания пункта спецификации с результатом выполнения фрагмента кода

Пункт спецификации: "Метод Power должен принимать входные параметры: x – целое число, возводимое в степень, и n – неотрицательный порядок степени. Метод должен возвращать вычисленное значение xn ".

Выполняем метод со следующими параметрами: Power(2,2)

Проверка результата выполнения возможна, когда результат вычисления заранее известен – 4. Если результат выполнения 22 = 4, то он соответствует спецификации.

В процессе тестирования Оракул последовательно получает элементы множества (X,Y) и соответствующие им результаты вычислений (X,Yв) для идентификации фактов несовпадений (test incident).

При выявлении $(X, Y\text{в}) \notin (X, Y)$ запускается процедура исправления ошибки, которая заключается во внимательном анализе (просмотре) протокола промежуточных вычислений, приведших к (X,Yв), с помощью следующих методов:

"Выполнение программы в уме" (deskchecking).

Вставка операторов протоколирования (печати) промежуточных результатов (logging).

Пример вставки операторов протоколирования промежуточных результатов

Можно выводить промежуточные значения переменных при выполнении программы. Код, осуществляющий вывод, расположен ниже.

Этот метод относится к наиболее популярным средствам автоматизации отладки программистов прошлых десятилетий. В настоящее время он известен как метод внедрения "агентов" в текст отлаживаемой программы.

```
//  Метод вычисляет неотрицательную
//  степень n числа x
static public double Power(double x, int n)
{
  double z=1;

  for (int i=1;n>=i;i++)
  {
    z = z*x;
    Console.WriteLine("i = {0} z = {1}",
                  i, z);
  }
  return z;
}
```

Исходный текст метода Power со вставкой оператора протоколирования

```
double Power(double x, int n)
{
  double z=1;
  int i;
  for (i=1;n>=i;i++)
  {
    z = z*x;
    printf("i = %d z = %f\n",i,z);
  }
  return z;
}
```

Содержание отчета и его форма

– Подготовьте отчет, в котором полностью опишите выполнение заданий.

– Отчет по лабораторной работе должен содержать:
– Название работы;
– Цель лабораторной работы;
– Формулировку задания и технологию его выполнения;
– Ответы на контрольные вопросы.

Контрольные вопросы:

1. Определение понятия «логический язык спецификаций»
2. Суть частичной и тотальной корректности программы
3. Операционная семантика языков программирования
4. Инварианты циклов программы
5. Метод индуктивных утверждений Флойда доказательства частичной корректности элементарных программ
6. Верификация программы вычисления частного и остатка от деления целых чисел методом Флойда.

Индивидуальные задания для письменного отчета

1. Выполните все этапы тестирования для процедуры вычисления факториала. Используйте для программы рассмотренные подходы к верификации и отладке.
2. Выполните все этапы тестирования для задачи решения квадратного уравнения, которая формулируется следующим образом: найти корни квадратного уравнения, заданного своими коэффициентами, с положительным дискриминантом; подстановкой в уравнение, убедиться в погрешности вычислений. Используйте для свой программы рассмотренные в подходы к отладке.

Защита лабораторной работы

По результатам отчета, представленного в письменной форме, проводится собеседование, которое имеет контролирующую и учебную функции.

Лабораторная работа 2

Отладка программ путем пошагового выполнения кода

Цель и содержание: научиться выполнять отладку программ путем пошагового выполнения кода.

Организационная форма занятий: практикум.

Вопросы для обсуждения на лабораторном занятии: отладка программ путем пошагового выполнения кода.

Аппаратура и материалы. Для выполнения задания необходим персональный компьютер, а также соответствующее лабораторной работе программное обеспечение.

Указания по технике безопасности. Самостоятельно не производить: установку и удаление программного обеспечения, ремонт персонального компьютера. Соблюдать правила технической безопасности при работе с электрооборудованием.

Теоретическое обоснование

Интегрированный отладчик предполагает возможность пошагового выполнения программы (трассировки), когда программа выполняется построчно с остановом после выполнения операторов каждой очередной строки текста программы, содержащей исполняемые операторы.

Трассировка возможна тремя способами: с заходом в выполняемые подпрограммы, без захода в них и до ближайшего исполняемого оператора.

В первом случае все подпрограммы рассматриваются как последовательность строк, выполняемых поочередно с остановкой после выполнения каждой строки. Во втором случае подпрограмма рассматривается как единый оператор, выполняемый без каких-либо остановок (если, конечно, в ней не заданы какие-то точки останова). Трассировка с заходом в выполняемые подпрограммы невозможна для стандартных подпрограмм, в том числе и методов стандартных объектов. Трассировка до ближайшего исполняемого оператора выполняет программу до момента выхода на первый исполняемый оператор, написанный программистом. Это может быть удобно, когда, например, в программе есть переопределенные программистом виртуальные методы объектов, вызываемые стандартными частями программы.

При трассировке программы можно сочетать все эти три варианта. В начале отладки, когда подпрограммы еще не отлажены, следует в них заходить, т. е. использовать трассировку с заходом в подпрограммы. Когда же подпрограммы будут полностью отлажены, контроль выполнения ими своих операций можно пропускать. Трассировкой до ближайшего оператора можно пользоваться по мере необходимости. Для трассировки с заходом в подпрограммы можно воспользоваться следующими возможностями.

• выполнить команду главного меню Run|Trace Into (F7);

• нажать на панели быстрого доступа кнопку Trace into.

Трассировку без захода в подпрограммы можно осуществить следующими способами:

• выполнить команду главного меню Run|Step Over (F8);

• нажать на панели быстрого доступа кнопку Step over.

Для трассировки до ближайшего исполняемого оператора следует выполнить команду главного меню Run|Trace to Next Source Line (Shift+F7).

После очередной остановки программы можно повторно выполнить одну из указанных выше операций.

Трассировку можно осуществлять после любого останова программы и с самого ее начала (например, запустить программу можно с помощью кнопки Trace into панели быстрого доступа).

В любой момент трассировку можно прекратить либо запустив программу дальше, как указано в предыдущем параграфе, либо прекратив отладку командой главного меню Run|Program Reset (Ctrl+F2).

При пошаговом выполнении программы код выполняется строчка за строчкой. В среде Microsoft Visual Studio возможны следующие команды пошагового выполнения:

Step Into – если выполняемая строчка кода содержит вызов функции, процедуры или метода, то происходит вызов, и программа останавливается на первой строчке вызываемой функции, процедуры или метода.

Step Over - если выполняемая строчка кода содержит вызов функции, процедуры или метода, то происходит вызов и выполнение всей функции и программа останавливается на первой строчке после вызываемой функции.

Step Out – предназначена для выхода из функции в вызывающую функцию. Эта команда продолжит выполнение функции и остановит выполнение на первой строчке после вызываемой функции.

Пошаговое выполнение до сих пор является мощным методом автономного тестирования и отладки небольших программ.

1. Создать новое консольное приложение, ввести текст программы перевода декартовых координат в полярные. Откомпилировать и выполнить программу.

Решение:

```
using System;//программа использует пространство имен System
class PolarProgram   // объявление нового класса PolarProgram
{
    static void Main()  //определение метода Main. void указывает, что метод
                        //не возвращает значений -результатов, пустые скобки -
                        //что метод не принимает входных параметров
    {
        double x, y, ro, fi;       //Объявление вещественных переменных
        string s;   // объявление строковой переменной для ввода данных
        Console.WriteLine("Введите координату x точки на плоскости: ");// вывод
                  //приглашения пользователю о следующем вводе
        s = Console.ReadLine(); // ввод текстового изображения числа x
        x = Convert.ToDouble(s);// преобразование изображения числа в число x
        Console.WriteLine("Введите координату y точки на плоскости: ");
        s = Console.ReadLine();//ввод текстового изображения числа y
        y = Convert.ToDouble(s);//преобразование изображения числа в число y
        ro = Math.Sqrt(x * x + y * y);// вычисление полярных координат ro, fi
        fi = Math.Atan2(y, x);// с использованием функций класса Math
        Console.WriteLine("x="+x+", y="+y+", ro="+ro+", fi="+fi);// вывод результатов

    }
}
```

Содержание отчета и его форма

– Подготовьте отчет, в котором полностью опишите выполнение заданий.
– Отчет по лабораторной работе должен содержать:
– Название работы;
– Цель лабораторной работы;
– Формулировку задания и технологию его выполнения;
– Ответы на контрольные вопросы.

Контрольные вопросы:

1. Аксиоматическая семантика для операторов условного выбора и перехода.
2. Аксиоматическая семантика для операторов циклов while, repeat и for.
3. Описание верификации программы проверки простоты заданного числа.
4. Аксиоматическая семантика оператора присваивания элементам массива.
5. Аксиоматическая семантика оператора присваивания буферу последовательного файла.
6. Аксиоматическая семантика стандартных процедур обработки последовательных файлов rewrite, reset, get, put.

Индивидуальные задания для письменного отчета

1. Создать новое консольное приложение, ввести текст программы, приведенный ниже. Программа вводит число, если оно положительное – увеличивает его на 5, если нет – умножает на два. Откомпилировать и выполнить программу. Изучить работу программы в пошаговом режиме.
2. Создать новое консольное приложение, ввести текст программы решения квадратного уравнения. Откомпилировать и выполнить программу. Изучить работу программы в пошаговом режиме.

Защита лабораторной работы

По результатам отчета, представленного в письменной форме, проводится собеседование, которое имеет контролирующую и учебную функции.

Лабораторная работа 3

Отладка программ с установкой контрольных точек и анализом памяти

Цель и содержание: научиться выполнять отладку программ с установкой контрольных точек и анализом памяти.

Организационная форма занятий: практикум.

Вопросы для обсуждения на лабораторном занятии: Отладка программ с установкой контрольных точек и анализом памяти.

Аппаратура и материалы. Для выполнения задания необходим персональный компьютер, а также соответствующее лабораторной работе программное обеспечение.

Указания по технике безопасности. Самостоятельно не производить: установку и удаление программного обеспечения, ремонт персонального компьютера. Соблюдать правила технической безопасности при работе с электрооборудованием.

Теоретическое обоснование

Контрольная точка (breakpoint) – точка программы, которая при ее достижении посылает отладчику сигнал. По этому сигналу либо временно приостанавливается выполнение отлаживаемой программы, либо запускается программа "агент", фиксирующая состояние заранее определенных переменных или областей в данный момент.

Когда выполнение в контрольной точке приостанавливается, отлаживаемая программа переходит в режим останова (break mode). Вход в режим останова не прерывает и не заканчивает выполнение программы и позволяет анализировать состояние отдельных переменных или структур данных. Возврат из режима break mode в режим выполнения может произойти в любой момент по желанию пользователя.

Когда в контрольной точке вызывается программа "агент", она тоже приостанавливает выполнение отлаживаемой программы, но только на время, необходимое для фиксации состояния выбранных переменных или структур данных в специальном электронном журнале - Log-файле, после чего происходит автоматический возврат в режим исполнения.

Трасса - это "сохраненный путь " на управляющем графе программы, т.е. зафиксированные в журнале записи о состояниях переменных в заданных точках в ходе выполнения программы.

На Рис. 3.1 условно изображен управляющий граф некоторой программы. Трасса, проходящая через вершины 0-1-3-4-5 зафиксирована в Табл. 3.1. Строки таблицы отображают вершины управляющего графа программы, или breakpoints, в которых фиксировались текущие значения заказанных пользователем переменных.

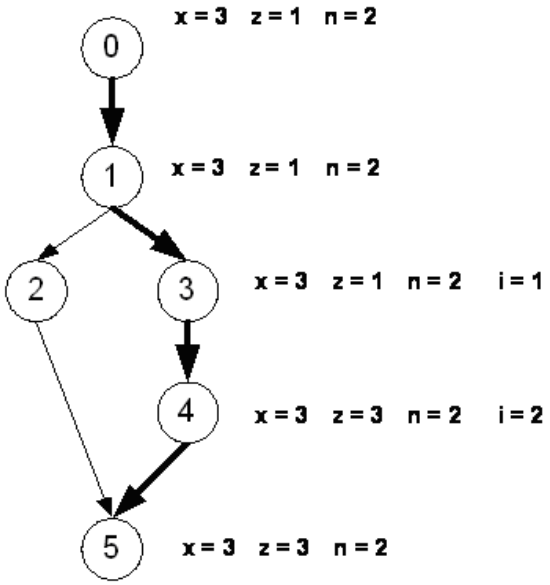

Рис. 3.1. Управляющий граф программы

Таблица 3.1. Трасса, проходящая через вершины 0-1-3-4-5

№ вершины-оператора	Значение переменной x	Значение переменной z	Значение переменной n	Значение переменной i
0	3	1	2	не зафиксировано
1	3	1	2	не зафиксировано
3	3	1	2	1
4	3	3	2	2
5	3	3	2	не зафиксировано

Дамп – область памяти, состояние которой фиксируется в контрольной точке в виде единого массива или нескольких связанных массивов. При анализе, который осуществляется после выполнения трассы в режиме off-line, состояния дампа структурируются, и выделенные области или поля сравниваются с состояниями, предусмотренными спецификацией. Например, при моделировании поведения управляющих программ контроллеров в виде дампа фиксируются области общих и специальных регистров, или целые области оперативной памяти, состояния которой определяет алгоритм управления внешней средой.

Содержание отчета и его форма

– Подготовьте отчет, в котором полностью опишите выполнение заданий.
– Отчет по лабораторной работе должен содержать:
– Название работы;
– Цель лабораторной работы;
– Формулировку задания и технологию его выполнения;
– Ответы на контрольные вопросы.

Контрольные вопросы:

1. Описание верификации программы поиска минимального элемента одномерного массива.
2. Описание верификации программы копирования одного файла в другой файл.
3. Аксиоматическая семантика оператора присваивания переменной с указателем.
4. Спецификация программ над линейными списками
5. Описание верификации программы поиска элемента линейного списка с заданным ключом.
6. Эвристические методы синтеза инвариантов циклов

Индивидуальные задания для письменного отчета

1. Создать новое консольное приложение, ввести текст программы, приведенный ниже. Программа вводит число, если оно положительное – увеличивает его на 5, если нет – умножает на два. Выполнить отладку программ с установкой контрольных точек и анализом памяти. После освоения всех навыков выполнения отладки программы и проверки значений переменных следует продемонстрировать полученные навыки преподавателю для контроля.
2. Создать новое консольное приложение, ввести текст программы решения квадратного уравнения. Выполнить отладку программ с установкой контрольных точек и анализом памяти. После освоения всех навыков выполнения отладки программы и проверки значений переменных следует продемонстрировать полученные навыки преподавателю для контроля.

Защита лабораторной работы

По результатам отчета, представленного в письменной форме, проводится собеседование, которое имеет контролирующую и учебную функции.

Лабораторная работа 4

Анализ качества программы путем тестирования ее функциональных возможностей.

Цель и содержание: научиться выполнять анализ качества программы путем тестирования ее функциональных возможностей и проводить сравнительный анализ с аналогичными программами.

Организационная форма занятий: практикум.

Вопросы для обсуждения на лабораторном занятии:

Анализ качества программы путем тестирования ее функциональных возможностей;

Проведение сравнительного анализа с аналогичными программами.

Аппаратура и материалы. Для выполнения задания необходим персональный компьютер, а также соответствующее лабораторной работе программное обеспечение.

Указания по технике безопасности. Самостоятельно не производить: установку и удаление программного обеспечения, ремонт персонального компьютера. Соблюдать правила технической безопасности при работе с электрооборудованием.

Теоретическое обоснование

Компания Microsoft предоставляет встроенное в интегрированную среду разработки программного обеспечения Visual Studio средство, которое позволяет оценить качества кода проекта.

Результаты работы средства содержат 5 метрик:

- комплексный показатель качества кода. Visual Studio помечает методы/классы зеленым цветом, если значение метрики находится в пределах от 20 до 100, желтым цветом, если значение находится в пределах от 10 до 20, и красным цветом, когда значение меньше 10;
- цикломатическая сложность: чем больше этот показатель, тем больше тестов должно быть написано, для полного покрытия кода;
- глубина наследования;
- степень зависимости классов друг с другом;
- количество строк кода.

В целом, этот продукт удобен для использования, так как не требует установки дополнительного программного обеспечения, так как является интегрированным в среду разработки. Естественно, MS Visual Studio - средство, направленное на работу с операционной системой Windows, языками Си, Си++, .NET. Также большим недостатком этой системы является его стоимость.

Запустите Visual Studio и откройте Team Explorer.

Вы должны быть подключены к командному проекту FabrikamFiber, если этого не произошло, нажмите Connect to Team Projects и инициируйте подключение.

В Team Explorer – Home нажмите два раза на первом решении FabrikamFiber.CallCenter.sln.

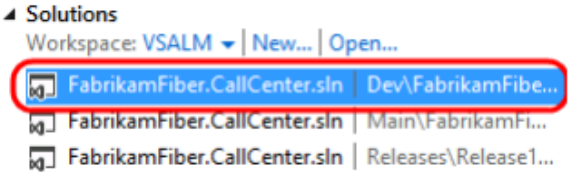

Рис. 1 Team Explorer - Home

Пересоберите решение (Build | Rebuild Solution в меню).

В Solution Explorer нажмите правой кнопкой на FabrikamFiber.Web и вызовите Properties.

Перейдите на вкладку Code Analysis.

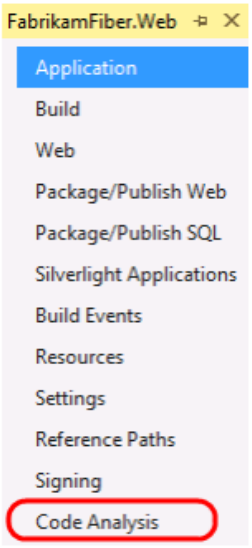

Рис. 2 Настройка Code Analysis

Примечание: на этой вкладке можно выбрать набор правил.

Выберите в Rule Set опцию “Microsoft All Rules”.

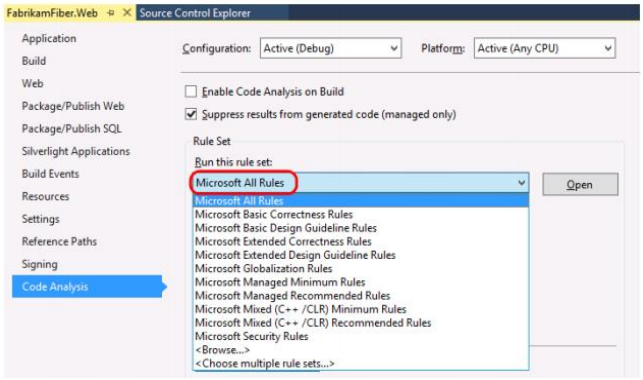

Рис. 3 Настройка набора правил для Code Analysis

Примечание: в Visual Studio редакций Professional, Premium и Ultimate можно создавать собственные правила для C++.

В Solution Explorer нажмите правой кнопкой на проекте FabrikamFiber.Web. Нажмите Analyze | Run Code Analysis.

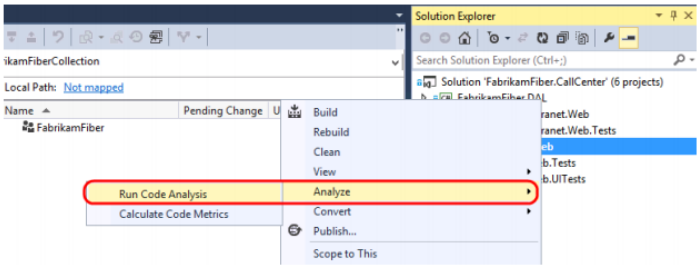

Рис. 4 Запуск Code Analysis

Code Analysis использует правила статического анализа, определенные Microsoft, и показывает результаты в окне Code Analysis. Изучите результаты.

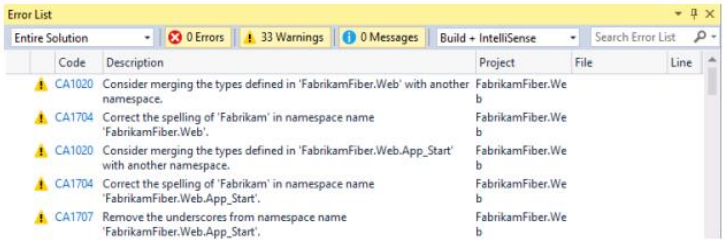

Рис. 5 Результаты Code Analysis

Примечание: в зависимости от версии проекта FabrikamFiber у вас может отображаться разное количество результатов. Правила Code Analysis могут быть настроены таким образом, чтобы показывать еще и ошибки.

В результате анализа Code Analysis мы получаем большое количество информации в виде предупреждений, включая идентификатор категории (например, CA2000), название, описание проблемы, расположение файла и предлагаемое решение.

В окне Code Analysis можно использовать текстовое поле для фильтрации результатов по номеру и тексту в названии, сообщении, пути к файлу и названию функции.

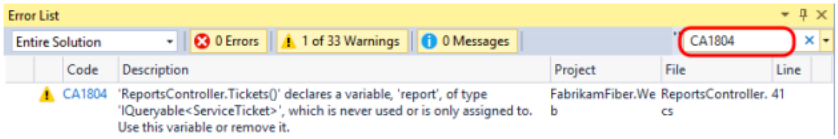

Рис. 6 Фильтрация результатов Code Analysis

Найдите предупреждение, которое будет просто разрешить (например, предупреждение CA1804). Нажмите два раза на нем для перехода в место, где возникла проблема.

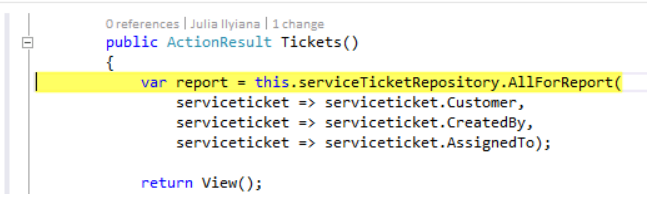

Рис. 7 Ошибки Code Analysis привязываются к коду

Исправьте ошибку. Для CA1804 нужно убрать неиспользуемые переменные – сделайте это, убрав определение report.

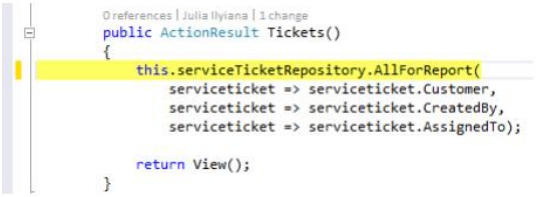

Рис. 8 Удаление неиспользуемых переменных

В окне Code Analysis выберите Analyze | FabrikamFiber.Web и убедитесь, что предупреждение исчезло.

Содержание отчета и его форма

– Подготовьте отчет, в котором полностью опишите выполнение заданий.

– Отчет по лабораторной работе должен содержать:

– Название работы;

– Цель лабораторной работы;

– Формулировку задания и технологию его выполнения;

– Ответы на контрольные вопросы.

Контрольные вопросы:

1. Применение эвристических методов для построения инвариантов циклов в программе бинарного поиска элемента упорядоченного массива.
2. Применение метода ограничивающих функций для доказательства терминации программ
3. Применение метода ограничивающих функций для доказательства терминации программы оптимального вычисления произведения целых чисел.
4. Основные компоненты автоматической системы верификации программ: общая схема системы.
5. Характеристика методов автоматического доказательства условий корректности.
6. Модули блока доказательства условий корректности автоматической системы верификации программ.

Индивидуальные задания для письменного отчета

1. Выполнить тестирование программы, определяющей точку пересечения двух прямых на плоскости. При этом она должна определять параллельность прямой одной из осей координат.
2. Составить тест-план и провести модульное тестирование следующих методов:

```
/// <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 скобок.
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
/// а 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()
```

Защита лабораторной работы

По результатам отчета, представленного в письменной форме, проводится собеседование, которое имеет контролирующую и учебную функции.

4. УЧЕБНО-МЕТОДИЧЕСКОЕ И ИНФОРМАЦИОННОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ (МОДУЛЯ)

4.1. Рекомендуемая литература

4.1.1. Основная литература

1. Луиза Тамре. Введение в тестирование программного обеспечения. - М.: Издательский дом «Вильямс», 2013. - С. 368.

2. Фаронов, В.В. Delphi. Программирование на языке высокого уровня: В. В. Фаронов- СПб.: Лидер, 2010.

4.1.2. Дополнительная литература

1. Гагарина, Л.Г. Современные проблемы информатики и вычислительной техники: учеб. пособие/ Л. Г. Гагарина, А. А. Петров- М.: ИД "ФОРУМ", 2011.

4.1.3. Методическая литература

1. Методические указания по выполнению лабораторных работ по дисциплине «Верификация и анализ программ»;
2. Методические указания по выполнению контрольной работы по дисциплине «Верификация и анализ программ»;
3. Методические рекомендации для студентов по организации самостоятельной работы по дисциплине «Верификация и анализ программ».

4.1.4. Интернет-ресурсы

1. http://www.intuit.ru – сайт дистанционного образования в области информационных технологий
2. http://window.edu.ru – образовательные ресурсы ведущих вузов

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
Пятигорский институт (филиал) СКФУ

МЕТОДИЧЕСКИЕ УКАЗАНИЯ ДЛЯ СТУДЕНТОВ ПО ОРГАНИЗАЦИИ САМОСТОЯТЕЛЬНОЙ РАБОТЫ ПО ДИСЦИПЛИНЕ «ВЕРИФИКАЦИЯ И АНАЛИЗ ПРОГРАММ»

Направление подготовки	**09.04.02 Информационные системы и технологии**
Направленность (профиль)	**«Технологии работы с данными и знаниями, анализ информации»**
Квалификация выпускника	Магистр

Пятигорск, 2025

СОДЕРЖАНИЕ

5. ЦЕЛЬ И ЗАДАЧИ ОСВОЕНИЯ ДИСЦИПЛИНЫ

Целью освоения дисциплины «Верификация и анализ программ» является получение устойчивых навыков самостоятельного программирования с применением современных программных средств для разработки, верификации, анализа и тестирования программ.

В соответствии с указанной целью при изучении дисциплины ставятся следующие задачи:

- привить навыки работы в среде визуального программирования,
- дать сведения о принципах, стратегиях и этапах верификации программ;
- изучить основные методы верификации, анализа и тестирования программ.

6. ТЕХНОЛОГИЧЕСКАЯ КАРТА САМОСТОЯТЕЛЬНОЙ РАБОТЫ СТУДЕНТА

Для студентов заочной формы обучения:

Коды реализуемых компетенций, индикатора(ов)	Вид деятельности студентов	Средства и технологии оценки	Объем часов, в том числе		
			СРС	Контактная работа с преподавателем	Всего
2 семестр					
ПК-3 (ИД 1 ПК-3, ИД 2 ПК-3, ИД 3 ПК-3) ПК-6 (ИД 1 ПК-6, ИД 2 ПК-6, ИД 3 ПК-6)	Подготовка к лекциям	Коллоквиум	0,54	0,06	0,6
ПК-3 (ИД 1 ПК-3, ИД 2 ПК-3, ИД 3 ПК-3) ПК-6 (ИД 1 ПК-6, ИД 2 ПК-6, ИД 3 ПК-6)	Самостоятельное изучение литературы	Коллоквиум	107,64	11,96	119,6
ПК-3 (ИД 1 ПК-3, ИД 2 ПК-3, ИД 3 ПК-3) ПК-6 (ИД 1 ПК-6, ИД 2 ПК-6, ИД 3 ПК-6)	Подготовка и выполнение лабораторных работ	Отчет письменный	1,62	0,18	1,8
ПК-3 (ИД 1 ПК-3, ИД 2 ПК-3, ИД 3 ПК-3) ПК-6 (ИД 1 ПК-6, ИД 2 ПК-6, ИД 3 ПК-6)	Выполнение контрольной работы	Контрольная работа	9	1	10

Итого за 2 семестр	118,8	13,2	132

7. СОДЕРЖАНИЕ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

3.1 Подготовка к лекциям. Самостоятельное изучение литературы

Подготовка к лекциям и самостоятельное изучение литературы подразумевает подготовку ответов по вопросам для собеседования.

Вопросы для собеседования

1. Понятие верификации.
2. Жизненный цикл разработки программного обеспечения.
3. Модели жизненного цикла.
4. Современные технологии разработки программного обеспечения.
5. Ролевой состав коллектива разработчиков, взаимодействие между ролями в различных технологических процессах.
6. Задачи и цели процесса верификации.
7. Тестирование, верификация и валидация - различия в понятиях.
8. Документация, создаваемая на различных этапах жизненного цикла.
9. Типы процессов тестирования и верификации и их место в различных моделях жизненного цикла.
10. Верификация сертифицируемого программного обеспечения.
11. Задачи и цели тестирования программного кода.
12. Методы тестирования.
13. Тестовое окружение.
14. Тестовые примеры.
15. Тест-планы.
16. Оценка качества тестируемого кода - статистика выполнения тестов.
17. Покрытие программного кода.
18. Технологические процессы верификации и роли в проекте, документация, создаваемая в ходе жизненного цикла проекта, ее назначение.
19. Стратегия и планы верификации.
20. Тест-требования.
21. Отчеты о прохождении тестов.
22. Отчеты о покрытии программного кода.
23. Отчеты о проблемах.
24. Трассировочные таблицы.
25. Задачи и цели проведения формальных инспекций.
26. Этапы формальной инспекции и роли ее участников.
27. Документирование процесса формальной инспекции.
28. Формальные инспекции программного кода.
29. Формальные инспекции проектной документации.
30. Уровни процесса верификации.
31. Задачи и цели модульного тестирования.
32. Понятие модуля и его границ.
33. Тестирование классов.
34. Подходы к проектированию тестового окружения.
35. Организация модульного тестирования.
36. Задачи и цели интеграционного тестирования.
37. Организация интеграционного тестирования.

38. Задачи и цели системного тестирования. Виды системного тестирования.
39. Системное тестирование, приемо-сдаточные и сертификационные испытания при разработке сертифицируемого программного обеспечения.
40. Задачи и цели тестирования пользовательского интерфейса.
41. Функциональное тестирование пользовательских интерфейсов.
42. Тестирование удобства использования пользовательских интерфейсов.

3.2 Подготовка и выполнение лабораторных работ

Подготовка и выполнение лабораторных работ подразумевает также выполнение индивидуальных заданий по изученным темам.

Темы индивидуальных заданий для письменного отчета:

Задание 1. Выполните все этапы тестирования для процедуры вычисления факториала. Используйте для программы рассмотренные подходы к верификации и отладке.

Задание 2. Создать новое консольное приложение, ввести текст программы, приведенный ниже. Программа вводит число, если оно положительное – увеличивает его на 5, если нет – умножает на два. Откомпилировать и выполнить программу. Изучить работу программы в пошаговом режиме.

Задание 3. Создать новое консольное приложение, ввести текст программы, приведенный ниже. Программа вводит число, если оно положительное – увеличивает его на 5, если нет – умножает на два. Выполнить отладку программ с установкой контрольных точек и анализом памяти. После освоения всех навыков выполнения отладки программы и проверки значений переменных следует продемонстрировать полученные навыки преподавателю для контроля.

Задание 4. Выполнить тестирование программы, определяющей точку пересечения двух прямых на плоскости. При этом она должна определять параллельность прямой одной из осей координат.

Задание 5. Выполните все этапы тестирования для задачи решения квадратного уравнения, которая формулируется следующим образом: найти корни квадратного уравнения, заданного своими коэффициентами, с положительным дискриминантом; подстановкой в уравнение, убедиться в погрешности вычислений. Используйте для свой программы рассмотренные в подходы к отладке.

Задание 6. Создать новое консольное приложение, ввести текст программы решения квадратного уравнения. Откомпилировать и выполнить программу. Изучить работу программы в пошаговом режиме.

Задание 7. Создать новое консольное приложение, ввести текст программы решения квадратного уравнения. Выполнить отладку программ с установкой контрольных точек и анализом памяти. После освоения всех навыков выполнения отладки программы и проверки значений переменных следует продемонстрировать полученные навыки преподавателю для контроля.

Задание 8. Составить тест-план и провести модульное тестирование следующих методов:

```
/// <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 скобок.
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
/// а 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()
```

3.3 Выполнение контрольной работы

Контрольная работа включат в себя вопросы по темам, которые нужно изучить самостоятельно и по ним подготовить отчет и презентации. Результаты выполнения контрольной работы предоставляются в электронном виде.

Комплект заданий для контрольных работ:

Задание 1. Привести ответы на заданные вопросы.

Вариант 1. Что такое логический язык спецификаций?
Вариант 2. Как определяется частичная и тотальная корректность программы?
Вариант 3. В каких терминах описывается операционная семантика языков программирования?
Вариант 4. Что такое инварианты циклов программы?
Вариант 5. Объяснить метод индуктивных утверждений Флойда доказательства частичной корректности элементарных программ.
Вариант 6. Провести верификацию программы вычисления частного и остатка от деления целых чисел методом Флойда.
Вариант 7. Объяснить аксиоматическую семантику для операторов условного выбора и перехода. Объяснить аксиоматическую семантику для операторов циклов while, repeat и for.
Вариант 8. Объяснить аксиоматическую семантику оператора присваивания элементам массива. Объяснить аксиоматическую семантику оператора присваивания буферу последовательного файла.
Вариант 9. Объяснить аксиоматическую семантику стандартных процедур обработки последовательных файлов rewrite, reset, get, put.
Вариант 10. Провести верификацию программы поиска минимального элемента

одномерного массива.

Задание 2. Привести ответы на заданные вопросы. Привести примеры.

Вариант 1. Из каких основных компонент состоит автоматическая система верификации программ? Объяснить общую схему этой системы.
Вариант 2. Охарактеризовать методы автоматического доказательства условий корректности.
Вариант 3. Какие модули содержит блок доказательства условий корректности автоматической системы верификации программ? Объяснить работу этого блока.
Вариант 4. Какие вы знаете эвристические методы синтеза инвариантов циклов?
Вариант 5. Применить эвристические методы для построения инвариантов циклов в программе бинарного поиска элемента упорядоченного массива.
Вариант 6. Провести верификацию программы проверки простоты заданного числа.
Вариант 7. Как применяется метод ограничивающих функций для доказательства терминации программ? Применить метод ограничивающих функций для доказательства терминации программы оптимального вычисления произведения целых чисел.
Вариант 8. Провести верификацию программы копирования одного файла в другой файл. Объяснить аксиоматическую семантику оператора присваивания переменной с указателем.
Вариант 9. Как проводится спецификация программ над линейными списками?
Вариант 10. Провести верификацию программы поиска элемента линейного списка с заданным ключом

8. Учебно-методическое и информационное обеспечение дисциплины

4.1. Рекомендуемая литература

4.1.1. Основная литература

3. Луиза Тамре. Введение в тестирование программного обеспечения. - М.: Издательский дом «Вильямс», 2013. - С. 368.

4. Фаронов, В.В. Delphi. Программирование на языке высокого уровня: В. В. Фаронов- СПб.: Лидер, 2010.

4.1.2. Дополнительная литература

2. Гагарина, Л.Г. Современные проблемы информатики и вычислительной техники: учеб. пособие/ Л. Г. Гагарина, А. А. Петров- М.: ИД "ФОРУМ", 2011.

4.1.3. Методическая литература

4. Методические указания по выполнению лабораторных работ по дисциплине «Верификация и анализ программ»;

5. Методические указания по выполнению контрольной работы по дисциплине «Верификация и анализ программ»;

6. Методическиерекомендации для студентов по организации самостоятельной работы по дисциплине «Верификация и анализ программ».

4.1.4. Интернет-ресурсы

3. http://www.intuit.ru – сайт дистанционного образования в области информационных технологий

4. http://window.edu.ru – образовательные ресурсы ведущих вузов

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Пятигорский институт (филиал) СКФУ

МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ВЫПОЛНЕНИЮ КОНТРОЛЬНОЙ РАБОТЫ ПО ДИСЦИПЛИНЕ «ВЕРИФИКАЦИЯ И АНАЛИЗ ПРОГРАММ»

Направление подготовки	**09.04.02 Информационные системы и технологии**
Направленность (профиль)	**«Технологии работы с данными и знаниями, анализ информации»**
Квалификация выпускника	Магистр

Пятигорск, 2025

СОДЕРЖАНИЕ

9. Введение

Методические указания содержат перечень вариантов заданий для контрольных работ, требования к оформлению контрольных работ и пример выполнения задания. Теоретической основой подготовки специалиста являются знания в области информатики, вычислительной систем.

1. Цель, задачи и реализуемые компетенции

Методические указания составлены с учетом требований стандарта высшего образования по дисциплине: «Верификация и анализ программ». Целью освоения дисциплины «Верификация и анализ программ» является получение устойчивых навыков самостоятельного программирования с применением современных программных средств для разработки, верификации, анализа и тестирования программ.

Индекс	Формулировка:
ПК-3	способность выполнять администрирование систем управления базами данных, системного программного обеспечения инфокоммуникационной системы организации, управление развитием инфокоммуникационной системы организации
ПК-6	способность проводить организационное сопровождение разработки, отладки, модификации и поддержки информационных технологий и систем

10. 2. Формулировка задания и его объем

Контрольная работа включат в себя вопросы по темам, которые нужно изучить самостоятельно и по ним подготовить отчет и презентации. Результаты выполнения контрольной работы предоставляются в электронном виде. Объем контрольной работы составляет 10-15 печатных листов формата А4.

Варианты заданий выбираются по последней цифре зачетной книжки.

11.

12. 3. Общие требования к написанию и оформлению работы

Контрольная работа выполняется и сдается в электронном виде на CD/CDRW носителе. На конверте необходимо указать название дисциплины, ФИО студента, факультет, номер группы, шифр зачетной книжки, № варианта задания, и список всех созданных в ходе выполнения задания файлов.

Приведенный в конце методических указаний список литературы может использоваться студентами при выполнении контрольной работы.

13. 4. Варианты заданий для студентов заочной формы обучения

Задание 1. Привести ответы на заданные вопросы.

Вариант 1. Что такое логический язык спецификаций?

Вариант 2. Как определяется частичная и тотальная корректность программы?

Вариант 3. В каких терминах описывается операционная семантика языков программирования?

Вариант 4. Что такое инварианты циклов программы?

Вариант 5. Объяснить метод индуктивных утверждений Флойда доказательства частичной корректности элементарных программ.

Вариант 6. Провести верификацию программы вычисления частного и остатка от деления целых чисел методом Флойда.
Вариант 7. Объяснить аксиоматическую семантику для операторов условного выбора и перехода. Объяснить аксиоматическую семантику для операторов циклов while, repeat и for.
Вариант 8. Объяснить аксиоматическую семантику оператора присваивания элементам массива. Объяснить аксиоматическую семантику оператора присваивания буферу последовательного файла.
Вариант 9. Объяснить аксиоматическую семантику стандартных процедур обработки последовательных файлов rewrite, reset, get, put.
Вариант 10. Провести верификацию программы поиска минимального элемента одномерного массива.

Задание 2. Привести ответы на заданные вопросы. Привести примеры.

Вариант 1. Из каких основных компонент состоит автоматическая система верификации программ? Объяснить общую схему этой системы.
Вариант 2. Охарактеризовать методы автоматического доказательства условий корректности.
Вариант 3. Какие модули содержит блок доказательства условий корректности автоматической системы верификации программ? Объяснить работу этого блока.
Вариант 4. Какие вы знаете эвристические методы синтеза инвариантов циклов?
Вариант 5. Применить эвристические методы для построения инвариантов циклов в программе бинарного поиска элемента упорядоченного массива.
Вариант 6. Провести верификацию программы проверки простоты заданного числа.
Вариант 7. Как применяется метод ограничивающих функций для доказательства терминации программ? Применить метод ограничивающих функций для доказательства терминации программы оптимального вычисления произведения целых чисел.
Вариант 8. Провести верификацию программы копирования одного файла в другой файл. Объяснить аксиоматическую семантику оператора присваивания переменной с указателем.
Вариант 9. Как проводится спецификация программ над линейными списками?
Вариант 10. Провести верификацию программы поиска элемента линейного списка с заданным ключом

14. 5. План-график выполнения задания

Дата получения задания	Дата предоставления выполненного задания

Зимняя сессия.	Летняя сессия. За две недели до начала сессии.

15.

16. 6. Критерии оценивания работы

Оценка «отлично» выставляется студенту, если он продемонстрировал глубокие, исчерпывающие знания и творческие способности в понимании, изложении и использовании учебно-программного материала; логически последовательные, содержательные, полные, правильные и конкретные ответы на все поставленные вопросы и дополнительные вопросы преподавателя; свободное владение основной и дополнительной литературой, рекомендованной учебной программой.

Оценка «хорошо» выставляется студенту, если он продемонстрировал твердые и достаточно полные знания всего программного материала, правильное понимание сущности и взаимосвязи рассматриваемых процессов и явлений; последовательные, правильные, конкретные ответы на поставленные вопросы при свободном устранении замечаний по отдельным вопросам; достаточное владение литературой, рекомендованной учебной программой.

Оценка «удовлетворительно» выставляется студенту, если он продемонстрировал твердые знания и понимание основного программного материала; правильные, без грубых ошибок ответы на поставленные вопросы при устранении неточностей и несущественных ошибок в освещении отдельных положений при наводящих вопросах преподавателя; недостаточное владение литературой, рекомендованной учебной программой.

Оценка «неудовлетворительно» выставляется студенту, если он продемонстрировал неправильные ответы на основные вопросы, допущены грубые ошибки в ответах, непонимание сущности излагаемых вопросов; неуверенные и неточные ответы на дополнительные вопросы.

17.

18. 7. Порядок защиты работы

Защита контрольной работы проводится в виде научного дискоса с презентацией выполненных заданий, в соответствии с графиком защиты. После доклада студенту задаются вопросы как преподавателем, так и студентами группы.

В процессе защиты своей работы студент делает доклад продолжительностью 7-10 минут. Доклад должен быть предварительно подготовлен студентом. Лучшее впечатление производит доклад, в форме пересказа, без зачтения текста, которым следует пользоваться только для уточнения цифрового материала. Студент должен свободно ориентироваться в своей работе.

В выступлении необходимо корректно использовать демонстрационные материалы, которые усиливают доказательность выводов и облегчают восприятие доклада студента. Они оформляются в виде презентации в системе Power Point.

19. 8. Учебно-методическое и информационное обеспечение дисциплины

8.1. Рекомендуемая литература

8.1.1. Основная литература

5. Луиза Тамре. Введение в тестирование программного обеспечения. - М.: Издательский дом «Вильямс», 2013. - С. 368.

6. Фаронов, В.В. Delphi. Программирование на языке высокого уровня: В. В. Фаронов- СПб.: Лидер, 2010.

8.1.2. Дополнительная литература

3. Гагарина, Л.Г. Современные проблемы информатики и вычислительной техники: учеб. пособие/ Л. Г. Гагарина, А. А. Петров- М.: ИД "ФОРУМ", 2011.

8.1.3. Методическая литература

7. Методические указания по выполнению лабораторных работ по дисциплине «Верификация и анализ программ»;

8. Методические указания по выполнению контрольной работы по дисциплине «Верификация и анализ программ»;

9. Методические указания для студентов по организации самостоятельной работы по дисциплине «Верификация и анализ программ».

8.1.4. Интернет-ресурсы

5. http://www.intuit.ru – сайт дистанционного образования в области информационных технологий

6. http://window.edu.ru – образовательные ресурсы ведущих вузов