

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования «Саратовский государственный технический
университет имени Гагарина Ю.А.»

Энгельсский технологический институт (филиал)



УТВЕРЖДАЮ
Зам. директора по СПДО
О.Г. Коваленко

**Методические указания
к выполнению практических занятий
ПМ.02 Осуществление интеграции программных модулей**

по специальности:
09.02.07 Информационные системы и программирование

Методические указания
рассмотрены на заседании
предметной (цикловой) методической комиссии
специальности 09.02.07
«25» июня 2024 года, протокол № 11

Председатель ПЦМК  А.А. Слободова

Энгельс 2024

ОРГАНИЗАЦИЯ - РАЗРАБОТЧИК:

Энгельсский технологический институт (филиал) федерального государственного бюджетного образовательного учреждения высшего образования «Саратовский государственный технический университет имени Гагарина Ю.А.»

РАЗРАБОТЧИК: Зотова А.А., преподаватель спецдисциплин ОСПДО

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

По учебному плану в соответствии с рабочей программой на изучение **ПМ.02 Осуществление интеграции программных модулей** обучающимися предусмотрено аудиторных занятий - 181 часа, из них практических занятий – 107 часов. В методические указания включены практические работы по темам курса. Каждая Практическое занятие содержит сведения о цели ее проведения и практическом использовании результатов исследования, необходимых для проведения работы, включает краткие теоретические сведения, этапы выполнения работы.

Целью практических занятий по **ПМ.02 Осуществление интеграции программных модулей** является: закрепление студентами теоретического материала по специальности и приобретение практического навыка.

После проведения практических занятий студент должен:

иметь практический опыт:

- модели процесса разработки программного обеспечения;
- основные принципы процесса разработки программного обеспечения;
- основные подходы к интегрированию программных модулей;
- основы верификации и аттестации программного обеспечения.

уметь:

- использовать выбранную систему контроля версий;
- использовать методы для получения кода с заданной функциональностью и степенью качества.

знать:

- основные положения теории баз данных, хранилищ данных, баз знаний; - основные принципы структуризации и нормализации базы данных;
- основные принципы построения концептуальной, логической и физической модели данных; методы описания схем баз данных в современных системах управления базами данных;
- структуры данных систем управления базами данных, общий подход к организации представлений, таблиц, индексов и кластеров;
- методы организации целостности данных;
- способы контроля доступа к данным и управления привилегиями;
- основные методы и средства защиты данных в базах данных.

Формируются общие и профессиональные компетенции:

Перечень общих компетенций

Код	Наименование общих компетенций
ОК 01.	Выбирать способы решения задач профессиональной деятельности, применительно к различным контекстам
ОК 02.	Использовать современные средства поиска, анализа и интерпретации информации и информационные технологии для выполнения задач профессиональной деятельности;
ОК 03.	Планировать и реализовывать собственное профессиональное и личностное развитие, предпринимательскую деятельность в профессиональной сфере, использовать знания по финансовой грамотности в различных жизненных ситуациях
ОК 04.	Эффективно взаимодействовать и работать в коллективе и команде
ОК 05.	Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста
ОК 06.	Проявлять гражданско-патриотическую позицию, демонстрировать осознанное поведение на основе традиционных общечеловеческих ценностей, в том числе с учетом гармонизации межнациональных и межрелигиозных отношений, применять стандарты антикоррупционного поведения
ОК 07.	Содействовать сохранению окружающей среды, ресурсосбережению, применять знания об изменении климата, принципы бережливого производства, 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.	Пользоваться профессиональной документацией на государственном и иностранном языках

Перечень профессиональных компетенций

ВД 2 Осуществление интеграции программных модулей

ПК 2.1. Разрабатывать требования к программным модулям на основе анализа проектной и технической документации на предмет взаимодействия компонент

ПК 2.2. Выполнять интеграцию модулей в программное обеспечение

ПК 2.3 Выполнять отладку программного модуля с использованием специализированных программных средств

ПК 2.4 Осуществлять разработку тестовых наборов и тестовых сценариев для программного обеспечения.

ПК 2.5. Производить инспектирование компонент программного обеспечения на предмет соответствия стандартам кодирования

ПРАКТИЧЕСКИЕ ЗАНЯТИЯ

МДК.02.01 Технология разработки программного обеспечения

Наименование раздела, темы	Номер практического занятия	Номер, название практического занятия /лабораторной работы	Количество часов	Форма представления результата
Раздел 1. Разработка программного обеспечения Тема 1.1 Основные понятия и стандартизация требований к программному обеспечению	1	Практическое занятие №1. Построение архитектуры программного средства	2	отчет
	2	Практическое занятие № 2. Изучение работы в системе контроля версий	2	
Тема 1.2. Описание и анализ требований. Диаграммы IDEF	3	Практическое занятие №3. Построение диаграммы Вариантов использования и диаграммы. Последовательности	2	Документы, сформированные и оформленные согласно требованиям ГОСТ
	4	Практическое занятие №4. Построение диаграммы Кооперации и диаграммы Развертывания.	2	
	5	Практическое занятие №5 Построение диаграмм деятельности, состояний и классов	2	
	6	Практическое занятие № 6. Построение диаграммы компонентов. Поток данных	2	
Тема 1.3. Оценка качества	7	Практическое занятие № 6. Оценка необходимого количества тестов»	4	Документы, сформиров

программных средств	8	Практическое занятие № 7. Оценка программных средств с помощью метрик	4	анн ые согласно требовани ям ГОСТ
	9	Практическое занятие № 8 «Разработка тестовых пакетов»	4	
Всего			24	

МДК.02.02 Инструментальные средства разработки программного обеспечения

Наименование раздела, темы	Номер практического занятия	Номер, название практической /лабораторной работы	Количество часов	Форма представления результата
Тема 2.1 Современные технологии и инструменты интеграции	1	Практическое занятие №1 «Разработка структуры проект, модульной структуры проекта (диаграммы модулей)»	2	Отчет о выполнении работы (на усмотрения преподавателя устный или в виде текстового документа)
	2	Практическое занятие №2. «Разработка перечня артефактов и протоколов проекта»	2	
	3	Практическое занятие №3. «Настройка работы системы контроля версий (типов импортируемых файлов, путей, фильтров и др. параметров импорта в репозиторий)»	2	
	4	Практическое занятие №4. «Разработка и интеграция модулей проекта (командная работа)»	2	
	5	Практическое занятие №5. «Отладка отдельных модулей программного проекта»	2	
	6	Практическое занятие №6. «Организация обработки исключений»	2	
Тема 2.2 Инструментарий тестирования и анализа качества программных средств	7	Практическое занятие №7. Применение отладочных классов в проекте	3	Отчет о выполнении работы (на усмотрения преподавателя устный или в виде текстового документа)
	8	Практическое занятие № 8. Отладка однопоточного приложения	2	
	9	Практическое занятие № 9. Отладка многопоточного приложения	3	

				документ а)
	10	Практическое занятие № 10. Инспекция кода модулей проекта	4	Отчет о результат ах тестирова ния
	11	Практическое занятие № 11 Тестирование интеграции	2	Набор тестовых модулей
Всего			27	

МДК.02.03 Математическое моделирование

Наименование раздела, темы	Номер практического занятия	Номер, название практической /лабораторной работы	Количество часов	Форма представления результата
Тема 3.1. Основы моделирования. Детерминированные задачи	1	Практическое занятие № 1 «Построение простейших математических моделей. Построение простейших статистических моделей»	2	Математическая модель, Блок/схема решения задачи
	2	Практическое занятие № 2. «Решение простейших однокритериальных задач»,»	2	
	3	Практическое занятие № 3 Графический метод решения задач линейного программирования	2	
	4	Практическое занятие № 4 Сведение произвольной задачи линейного программирования к основной задаче линейного программирования	2	
	5	Практическое занятие № 5 Решение задач линейного программирования симплекс–методом	2	
	6	Практическое занятие № 6 Нахождение начального решения транспортной задачи. Решение транспортной задачи методом потенциалов		
	7	Практическое занятие № 7 Применение метода стрельбы для решения линейной краевой задачи	2	
	8	Практическое занятие № 8. Задача о распределении средств между предприятиями», «Задача о замене оборудования		
Тема 3.2 Графовые модели отказоустойчивости	9	Практическое занятие № 9. Характеристики графов	2	Математическая модель, Блок/схема решения
	10	Практическое занятие № 10. Изоморфизм графов	2	
	11	Практическое занятие № 11.	2	

		Свойства деревьев		задачи
	12	Практическое занятие № 12. Восстановление графа по коду Прюфера	2	
Всего			26	

2. ОБЩИЕ РЕКОМЕНДАЦИИ ПО ВЫПОЛНЕНИЮ ПРАКТИЧЕСКИХ РАБОТ

Студент должен:

- строго выполнять весь объем домашней подготовки, указанный в описаниях соответствующих практических работ;
- знать, что выполнению каждой работы предшествует проверка готовности студента, которая производится преподавателем;
- знать, что после выполнения работы бригада, которая назначается преподавателем на весь период работы, должна представить отчет о проделанной работе с обсуждением полученных результатов и выводов.

Процедура выставления окончательной оценки студенту по работе и порядок выполнения пропущенных работ по уважительным и неуважительным причинам следующая:

Объем работы исчисляется количеством полно и правильно выполненных заданий вне зависимости от порядка следования. Если задание не предполагает сохранения результатов в файле, то оценка выполнения производится по принципу вопрос-ответ или устное задание - выполненное действие. Процент выполнения рассчитывается для каждой работы исходя из общего числа заданий.

Отметка	Объем выполнения работы в %
«5» (отлично)	90 – 100
«4» (хорошо)	70 – 89
«3» (удовлетворительно)	50 – 69
«2» (неудовлетворительно)	менее 50

Для допуска студента к итоговой аттестации по МДК необходимо выполнение не менее 95% заданий.

Общие рекомендации обучающемуся по выполнению практических работ

1. Внимательно прочитайте задание, при необходимости повторите лекционный материал по конспектам и другим источникам, относящийся к теме практической/ лабораторной работы.

2. Ответьте на контрольные вопросы, если они предложены.

1. Подготовьте все необходимое для выполнения задания, рационально подготовьте рабочее место.

2. Продумайте ход выполнения работы.

3. Если ваша работа связана с использованием ИКТ, проверьте наличие и работоспособность программного обеспечения, необходимого для выполнения задания.

4. Если при выполнении практической работы применяется групповое или коллективное выполнение задания, старайтесь поддерживать в коллективе нормальный психологический климат, грамотно распределить роли и обязанности. Вместе проводите анализ организации и промежуточные результаты практической работы микрогруппы.

5. При выполнении практического задания соблюдайте правила техники безопасности и охраны труда.

6. В процессе выполнения практической работы обращайтесь за консультациями к преподавателю, чтобы вовремя скорректировать свою деятельность, проверить правильность выполнения задания.

7. По окончании выполнения практической работы составьте письменный или устный отчет в соответствии с теми методическими указаниями по оформлению отчета, которые вы получили от преподавателя или в методических указаниях.

8. Сдайте готовую работу преподавателю для проверки.

9. Участвуйте в обсуждении и оценке полученных результатов практической работы (общегрупповом или в микрогруппах).

Методические указания для выполнения практических работ

Раздел 1. Разработка программного обеспечения

Практическое занятие № 1

Тема раздела: Основные понятия и стандартизация требований к программному обеспечению

Тема практического занятия: Построение архитектуры программного средства

Цель: приобретение навыков создания формальных моделей и на их основе определение спецификаций разрабатываемого программного обеспечения, приобретение навыков проектирования программного обеспечения

Материально-техническое и комплексно-методическое обеспечение: Для проведения практической работы используется следующее обеспечение: персональный компьютер, подключённый к Интернету, MS Visual Studio

Время выполнения: 90 минут

Форма отчетности по занятию: вывод о проделанной работе

Последовательность выполнения работы

Задание 1. Изучить теоретические материалы.

Теоретические материалы:

Эскизный проект

Эскизный проект предусматривает разработку предварительных проектных решений по системе и ее частям.

Выполнение стадии эскизного проектирования не является строго обязательной. Если основные проектные решения определены ранее или достаточно очевидны для конкретной ИС и объекта автоматизации, то эта стадия может быть исключена из общей последовательности работ.

Содержание эскизного проекта задается в ТЗ на систему. Как правило, на этапе эскизного проектирования определяются:

- функции ИС;
- функции подсистем, их цели и ожидаемый эффект от внедрения;
- состав комплексов задач и отдельных задач;
- концепция информационной базы и ее укрупненная структура;
- функции системы управления базой данных;
- состав вычислительной системы и других технических средств;
- функции и параметры основных программных средств.

По результатам проделанной работы оформляется, согласовывается и утверждается документация в объеме, необходимом для описания полной совокупности принятых проектных решений и достаточном для дальнейшего выполнения работ по созданию системы.

На основе технического задания (и эскизного проекта) разрабатывается технический

проект.

Разработка эскизного проекта программы. Этапы выполнения эскизного проекта.

Эскизный проект	Разработка эскизного проекта	Предварительная разработка структуры входных и выходных данных.
		Уточнение методов решения задачи.
		Разработка общего описания алгоритма решения задачи
		Разработка технико-экономического обоснования.
	Утверждение эскизного проекта	Разработка пояснительной записки.
		Согласование и утверждение эскизного проекта.

Основная задача эскизного проекта — создать прообраз будущей автоматизированной системы. При разработке эскизного проекта разработчик определяет основные контуры будущей системы, а заказчик в свою очередь получает представление об основных чертах будущего объекта автоматизации и анализирует их возможную применимость в последующей работе.

При разработке эскизного проекта составляются:

- Ведомость эскизного проекта. Общая информация по проекту.
- Пояснительная записка к эскизному проекту. Вводная информация, позволяющая ее потребителю быстро освоить данные по конкретному проекту.
- Схема организационной структуры. Описание организационной структуры организации, которая будет использовать создаваемую автоматизированную систему в практической работе.
- Структурная схема комплекса технических средств. Техническая составляющая автоматизированной системы, включающая в себя набор серверов, рабочих станций, схему локальной вычислительной сети и структурированной кабельной системы.
- Схема функциональной структуры. Описание задач, которые будут использоваться в работе подсистем. Видение участков информационной системы и порядок и их взаимодействия.
- Схема автоматизации. Логический процесс создания автоматизированной системы от начала до конца.
- Согласно ГОСТ 34.201-89, дополнительно в эскизный проект по необходимости может быть включено техническое задание на разработку новых технических средств.

Эскизный проект чаще всего не разделяют, он выполняется в рамках общего (первоначального) этапа всего проекта.

Перечень работ, составляющих эскизный проект, может варьироваться в зависимости от конкретного технического задания заказчика (его пожеланий) и сложности проектируемого проекта. Соответственно варьируется и цена этого этапа.

Эскизный проект не всегда создается под конкретного заказчика. Нередко разработчики с помощью эскизного проекта стремятся показать свой творческий потенциал и найти потенциальных заказчиков. Не случайно на различные конкурсы представляются именно эскизные проекты.

Разработка спецификаций

Разработка программного обеспечения начинается с анализа требований к нему.

В результате анализа получают спецификации разрабатываемого программного обеспечения, строят общую модель его взаимодействия с пользователем или другими программами и конкретизируют его основные функции.

При структурном подходе к программированию на этапе анализа и определения спецификаций разрабатывают три типа моделей: модели функций, модели данных и модели потоков данных. Поскольку разные модели описывают проектируемое программное обеспечение с разных сторон, рекомендуется использовать сразу несколько моделей, разрабатываемых в виде диаграмм, и пояснять их текстовыми описаниями, словарями и т. п.

Структурный анализ предполагает использование следующих видов моделей:

- диаграмм потоков данных (DFD - Data Flow Diagrams), описывающих взаимодействие источников и потребителей информации через процессы, которые должны быть реализованы в системе;
- диаграмм «сущность-связь» (ERD Entity-Relationship Diagrams), описывающих базы данных разрабатываемой системы;
- диаграмм переходов состояний (STD - State Transition Diagrams), характеризующих поведение системы во времени;
- функциональных диаграмм (методика SADT);
- спецификаций процессов;
- словаря терминов. Спецификации процессов

Спецификации процессов обычно представляют в виде краткого текстового описания, схем алгоритмов, псевдокодов, Flow-форм или диаграмм Насси - Шнейдермана.

Словарь терминов

Словарь терминов представляет собой краткое описание основных понятий, используемых при составлении спецификации. Он должен включать определение основных понятий предметной области, описание структур элементов данных, их типом и форматов, а также всех сокращений и условных обозначений.

Диаграммы переходов состояний

С помощью диаграмм переходов состояний можно моделировать последующее функционирование системы на основе ее предыдущего и текущего функционирования.

Моделируемая система в любой заданный момент времени находится точно в одном из конечного множества состояний. С течением времени она может изменить свое состояние, при этом переходы между состояниями должны быть точно определены.

Функциональные диаграммы

Функциональные диаграммы отражают взаимосвязи функций разрабатываемого программного обеспечения.

Они создаются на ранних этапах проектирования систем, для того чтобы помочь проектировщику выявить основные функции и составные части проектируемой системы и, по возможности, обнаружить и устранить существенные ошибки. Для создания функциональных диаграмм предлагается использовать методологию SADT.

Диаграммы потоков данных

Для описания потоков информации в системе применяются диаграммы потоков данных (DFD – Data Flow Diagrams). DFD позволяет описать требуемое поведение системы в виде совокупности процессов, взаимодействующих посредством связывающих их потоков данных. DFD показывает, как каждый из процессов преобразует свои входные потоки данных в выходные потоки данных и как процессы взаимодействуют между собой.

Диаграммы «сущность - связь»

Диаграмма сущность-связь - инструмент разработки моделей данных, обеспечивающий стандартный способ определения данных и отношений между ними. Она включает сущности и взаимосвязи, отражающие основные бизнес-правила предметной области. Такая диаграмма не слишком детализирована, в нее включаются основные сущности и связи между ними, которые удовлетворяют требованиям, предъявляемым к ИС.

Разработка документации. Стадия «Технический проект».

Проект технический - образ намеченного к созданию объекта, представленный в виде его описания, схем, чертежей, расчетов, обоснований, числовых показателей.

Цель технического проекта - определение основных методов, используемых при создании информационной системы, и окончательное определение ее сметной стоимости.

Техническое проектирование подсистем осуществляется в соответствии с утвержденным техническим заданием.

Технический проект программной системы подробно описывает:

- выполняемые функции и варианты их использования;
- соответствующие им документы;
- структуры обрабатываемых баз данных;
-
- взаимосвязи данных;
- алгоритмы их обработки.

Технический проект должен включать данные об объемах и интенсивности потоков обрабатываемой информации, количестве пользователей программной системы, характеристиках оборудования и программного обеспечения, взаимодействующего с проектируемым программным продуктом.

При разработке технического проекта оформляются:

- ведомость технического проекта. Общая информация по проекту;
- пояснительная записка к техническому проекту. Вводная информация, позволяющая ее потребителю быстро освоить данные по конкретному проекту;
- описание систем классификации и кодирования;
- перечень входных данных (документов). Перечень информации, которая используется как входящий поток и служит источником накопления;
- перечень выходных данных (документов). Перечень информации, которая используется для анализа накопленных данных;
- описание используемого программного обеспечения. Перечень программного обеспечения и СУБД, которые планируется использовать для создания информационной системы;
- описание используемых технических средств. Перечень аппаратных средств, на которых планируется работа проектируемого программного продукта;
- проектная оценка надежности системы. Экспертная оценка надежности с выявлением наиболее благополучных участков программной системы и ее узких мест;
- ведомость оборудования и материалов. Перечень оборудования и материалов, которые потребуются в ходе реализации проекта.

Структурная схема

Структурной называют схему, отражающую состав и взаимодействие по управлению частями разрабатываемого программного обеспечения. Структурная схема определяется архитектурой разрабатываемого ПО.

Функциональная схема

Функциональная схема - это схема взаимодействия компонентов программного обеспечения с описанием информационных потоков, состава данных в потоках и указанием используемых файлов и устройств.

Разработка алгоритмов

Метод пошаговой детализации реализует нисходящий подход к программированию и предполагает пошаговую разработку алгоритма.

Структурные карты

Методика структурных карт используется на этапе проектирования ПО для того, чтобы продемонстрировать, каким образом программный продукт выполняет системные требования. Структурные карты Константайна предназначены для описания отношений между модулями.

Техника структурных карт Джексона основана на методе структурного программирования Джексона, который выявляет соответствие между структурой потоков данных и структурой программы. Основное внимание в методе сконцентрировано на соответствии входных и выходных потоков данных.

Задание 2.

1. На основе технического задания из своей курсовой работы выполнить анализ функциональных и эксплуатационных требований к программному продукту.
2. Определить основные технические решения (выбор языка программирования, структура программного продукта, состав функций ПП, режимы функционирования) и занести результаты в документ, называемый «Эскизным проектом».
3. Определить диаграммы потоков данных для решаемой задачи.
4. Определить диаграммы «сущность-связь», если программный продукт содержит базуданных.
5. Добавить словарь терминов.
6. Оформить результаты, используя MS Office или MS Visio в виде эскизного проекта.
7. Сдать и защитить работу.

Задание 3.

1. Разработать функциональную схему программного продукта.
2. Представить структурную схему в виде структурных карт Константайна.
3. Представить структурную схему в виде структурных карт Джексона.
4. Оформить результаты, используя MS Office или MS Visio в виде технического проекта.
5. Сдать и защитить работу.

Критерии оценивания:

Оценка 5 «отлично» работа выполнена полностью и правильно, сделаны правильные выводы

Оценка 4 «хорошо» работа выполнена правильно с учетом 1-2 несущественных ошибок, исправленных самостоятельно по требованию преподавателя

Оценка 3 «удовлетворительно» работа выполнена правильно не менее чем на половину или допущены 3-4 существенные ошибки

Оценка 2 «неудовлетворительно» допущены 5 и более существенные ошибки в ходе работы, которые обучающийся не может исправить даже по требованию преподавателя

Практическое занятие № 2

Тема раздела: Основные понятия и стандартизация требований к программному обеспечению

Тема практического занятия: Изучение работы в системе контроля версий

Цель: ознакомиться с возможностями Git и научиться создавать репозиторий

Материально-техническое и комплексно-методическое обеспечение: Для проведения практической работы используется следующее обеспечение: персональный компьютер, подключённый к Интернету, MS Visual Studio

Время выполнения: 90 минут

Форма отчетности по занятию: вывод о проделанной работе

Последовательность выполнения работы

Задание 1. Изучить теоретические сведения

Git. Git — это набор консольных утилит, которые отслеживают и фиксируют изменения в файлах (чаще всего речь идет об исходном коде программ, но можно использовать его для любых файлов).

С его помощью можно откатиться на более старую версию вашего проекта, сравнивать, анализировать, сливать изменения и многое другое, т.е. проводить контроль версий (или управление версиями). Существуют различные системы для контроля версий, например SVN, Mercurial, Perforce, CVS, Bitkeeper и др.

Git является распределенным, т.е. не зависит от одного центрального сервера, на котором хранятся файлы. Вместо этого он работает полностью локально, сохраняя данные в папках на жестком диске, которые называются репозиторием. Тем не менее вы можете хранить копию репозитория онлайн, это облегчает работу над одним проектом для нескольких людей. Для этого используются различные сайты, например github и bitbucket.

Задание 2. Первоначально необходимо установить Git на компьютер.

Скачайте exe-файл инсталлятора со страницы проекта на GitHub и запустите его: <http://msysgit.github.com/>. После установки у вас появится как консольная версия (включающая SSH-клиент, который пригодится позднее), так и стандартная графическая.

Git необходимо использовать только из командной оболочки, входящей в состав msysGit, потому что так вы сможете запускать сложные команды, приведенные в примерах. Командная оболочка Windows использует иной синтаксис, из-за чего примеры в ней могут работать некорректно. Для начинающих разработчиков клиент с графическим интерфейсом (например, GitHub Desktop и Sourcetree) будет полезен, но тем не менее знать команды очень важно.

Задание 3. После установки нужно добавить настройки. Настроим самые важные опции — имя пользователя и адрес электронной почты.

Откройте терминал и запустите команды:

```
git config --global user.name "My Name"
git config --global user.email myEmail@example.com
git config --global user.name "My Name"
git config --global user.email myEmail@example.com
```

Теперь каждое действие будет отмечено именем и почтой. Таким образом, пользователи всегда будут в курсе, кто за какие изменения отвечает, это обеспечивает порядок

Задание 4. Создать новый репозиторий. Как мы уже отмечали, Git хранит свои файлы и историю прямо в папке проекта. Чтобы создать новый репозиторий, нужно открыть терминал, зайти в папку проекта и выполнить команду `init`. Это включит приложение в этой конкретной папке и создаст скрытую директорию `.git`, где будут храниться история репозитория и настройки.

Создайте на рабочем столе папку под названием `git_exercise`. Для этого в окне терминала введите:

```
$ mkdir Desktop/git_exercise/
$ cd Desktop/git_exercise/
$ git init
$ mkdir Desktop/git_exercise/
$ cd Desktop/git_exercise/
$ git init
```

Командная строка должна содержать, например, следующее:

Initialized	empty	Git	repository	in
/home/user/Desktop/git_exercise/.git/1				
Initialized	empty	Git	repository	in
/home/user/Desktop/git_exercise/.git/				

Это значит, что репозиторий был успешно создан, но пока он пуст. Теперь создайте текстовый файл под названием `hello.txt` и сохраните его в директории `git_exercise`.

Задание 5. Определить состояние репозитория.

Status — это еще одна важная команда, которая показывает информацию о текущем состоянии репозитория: актуальна ли информация на нем, нет ли чего-то нового, что поменялось и т. д. Запуск `git status` на нашем недавно созданном репозитории выдаст:

```
$ git status
```

```
On branch masterInitial commit
```

```
Untracked files:
```

```
(use "git add ..." to include in what will be committed)
```

```
hello.txt
```

```
$ git status
```

```
On branch masterInitial commit
```

```
Untracked files:
```

```
(use "git add ..." to include in what will be committed) hello.txt
```

Сообщение говорит о том, что файл `hello.txt` неотслеживаемый. Это значит, что файл новый и система еще не знает, нужно ли следить за изменениями в файле или можно просто игнорировать его. Чтобы начать отслеживать новый файл, нужно его специальным образом объявить.

Задание 6. Подготовить файлы. В Git есть концепция области подготовленных файлов. Можно представить ее как холст, на который наносят изменения, нужные в коммите.

Коммит — состояние репозитория в определенный момент времени. Первоначально он пустой, но затем мы добавляем на него файлы (или части файлов, или одиночные строки) командой `add` и наконец коммитим все нужное в репозиторий (создаем слепок нужного нам состояния) командой `commit`.

В нашем случае у нас только один файл, так что добавим его:

```
$ git add hello.txt
```

```
$ git add hello.txt
```

Если нам нужно добавить все, что находится в директории, мы можем использовать:

```
$ git add -A
```

```
$ git add -A
```

Проверим статус снова, на этот раз мы должны получить другой ответ:

```
$ git status
```

```
On branch masterInitial commit
```

```
Changes to be committed:
```

```
(use "git rm --cached ..." to unstage)new file: hello.txt
```

```
$ git status
```

```
On branch masterInitial commit
```

```
Changes to be committed:
```

```
(use "git rm --cached ..." to unstage)new file: hello.txt
```

Файл готов к коммиту. Сообщение о состоянии также говорит о том, какие изменения относительно файла были проведены в области подготовки, в данном случае это новый файл, но файлы могут быть модифицированы или удалены.

Задание 7. Зафиксировать изменения (коммит). Чтобы зафиксировать изменения, нужно хотя бы одно изменение в области подготовки (мы только что создали его при помощи `git add`), после которого мы можем коммитить:

```
$ git commit -m "Initial commit."1
```

```
$ git commit -m "Initial commit."
```

Эта команда создаст новый коммит со всеми изменениями из области подготовки (добавление файла `hello.txt`). Ключ **-m** и сообщение `"Initial commit."` — это созданное пользователем описание всех изменений, включенных в коммит. Считается хорошей практикой делать коммиты часто и всегда писать содержательные комментарии.

Критерии оценивания:

Оценка 5 «отлично» работа выполнена полностью и правильно, сделаны правильные выводы

Оценка 4 «хорошо» работа выполнена правильно с учетом 1-2 несущественных ошибок, исправленных самостоятельно по требованию преподавателя

Оценка 3 «удовлетворительно» работа выполнена правильно не менее чем на половину или допущены 3-4 существенные ошибки

Оценка 2 «неудовлетворительно» допущены 5 и более существенные ошибки в ходе работы, которые обучающийся не может исправить даже по требованию преподавателя

Практическое занятие № 3

Тема раздела: описание и анализ требований

Тема практического занятия: Построение диаграмм вариантов использования и диаграммы. Последовательности

Цель: изучение построения диаграммы вариантов и диаграммы последовательностей

Материально-техническое и комплексно-методическое обеспечение: Для проведения практической работы используется следующее обеспечение: персональный компьютер, подключённый к Интернету, MS Visual Studio

Время выполнения: 90 минут

Форма отчетности по занятию: вывод о проделанной работе

Задание 1. Исходя из темы курсового проекта построить диаграмму вариантов использования и диаграмму последовательностей.

Критерии оценивания:

Оценка 5 «отлично» работа выполнена полностью и правильно, сделаны правильные выводы

Оценка 4 «хорошо» работа выполнена правильно с учетом 1-2 несущественных ошибок, исправленных самостоятельно по требованию преподавателя

Оценка 3 «удовлетворительно» работа выполнена правильно не менее чем на половину или допущены 3-4 существенные ошибки

Оценка 2 «неудовлетворительно» допущены 5 и более существенные ошибки в ходе работы, которые обучающийся не может исправить даже по требованию преподавателя

Практическое занятие № 4

Тема раздела: описание и анализ требований

Тема практического занятия: Построение диаграмм кооперации и развертывания

Цель: изучение построения диаграмм кооперации и развертывания

Материально-техническое и комплексно-методическое обеспечение: Для проведения практической работы используется следующее обеспечение: персональный компьютер, подключённый к Интернету, MS Visual Studio

Время выполнения: 90 минут

Форма отчетности по занятию: вывод о проделанной работе

Задание 1. Исходя из темы курсового проекта построить диаграммы коопераций и диаграмму развертывания

Критерии оценивания:

Оценка 5 «отлично» работа выполнена полностью и правильно, сделаны правильные выводы

Оценка 4 «хорошо» работа выполнена правильно с учетом 1-2 несущественных ошибок, исправленных самостоятельно по требованию преподавателя

Оценка 3 «удовлетворительно» работа выполнена правильно не менее чем на половину или допущены 3-4 существенные ошибки

Оценка 2 «неудовлетворительно» допущены 5 и более существенные ошибки в ходе работы, которые обучающийся не может исправить даже по требованию преподавателя

Практическое занятие № 5

Тема раздела: описание и анализ требований

Тема практического занятия: Построение диаграмм деятельности, состояний и классов

Цель: изучение построения диаграмм деятельности, состояний и классов

Материально-техническое и комплексно-методическое обеспечение: Для проведения практической работы используется следующее обеспечение: персональный компьютер, подключённый к Интернету, MS Visual Studio

Время выполнения: 90 минут

Форма отчетности по занятию: вывод о проделанной работе

Задание 1. Исходя из темы курсового проекта построить диаграммы деятельности, состояний и классов

Критерии оценивания:

Оценка 5 «отлично» работа выполнена полностью и правильно, сделаны правильные выводы

Оценка 4 «хорошо» работа выполнена правильно с учетом 1-2 несущественных ошибок, исправленных самостоятельно по требованию преподавателя

Оценка 3 «удовлетворительно» работа выполнена правильно не менее чем на половину или допущены 3-4 существенные ошибки

Оценка 2 «неудовлетворительно» допущены 5 и более существенные ошибки в ходе работы, которые обучающийся не может исправить даже по требованию преподавателя

Практическое занятие № 6

Тема раздела: описание и анализ требований

Тема практического занятия: Построение диаграммы компонентов и потоков данных

Цель: изучение построения диаграммы компонентов

Материально-техническое и комплексно-методическое обеспечение: Для проведения практической работы используется следующее обеспечение: персональный компьютер, подключённый к Интернету, MS Visual Studio

Время выполнения: 90 минут

Форма отчетности по занятию: вывод о проделанной работе

Задание 1. Исходя из темы курсового проекта построить диаграмму компонентов и потоков

Критерии оценивания:

Оценка 5 «отлично» работа выполнена полностью и правильно, сделаны правильные выводы

Оценка 4 «хорошо» работа выполнена правильно с учетом 1-2 несущественных ошибок, исправленных самостоятельно по требованию преподавателя

Оценка 3 «удовлетворительно» работа выполнена правильно не менее чем на половину или допущены 3-4 существенные ошибки

Оценка 2 «неудовлетворительно» допущены 5 и более существенные ошибки в ходе работы, которые обучающийся не может исправить даже по требованию преподавателя

Практическое занятие № 7

Тема раздела: описание и анализ требований

Тема практического занятия: Построение диаграммы потоков

Цель: изучение построения диаграммы потоков

Материально-техническое и комплексно-методическое обеспечение: Для проведения практической работы используется следующее обеспечение: персональный компьютер, подключённый к Интернету, MS Visual Studio

Время выполнения: 90 минут

Форма отчетности по занятию: вывод о проделанной работе

Задание 1. Исходя из темы курсового проекта построить диаграмму потоков

Критерии оценивания:

Оценка 5 «отлично» работа выполнена полностью и правильно, сделаны правильные выводы

Оценка 4 «хорошо» работа выполнена правильно с учетом 1-2 несущественных ошибок, исправленных самостоятельно по требованию преподавателя

Оценка 3 «удовлетворительно» работа выполнена правильно не менее чем на половину или допущены 3-4 существенные ошибки

Оценка 2 «неудовлетворительно» допущены 5 и более существенные ошибки в ходе работы, которые обучающийся не может исправить даже по требованию преподавателя

Практическое занятие № 7

Тема раздела: оценка качества программных средств

Тема практического занятия: оценка необходимого количества тестов

Материально-техническое и комплексно-методическое обеспечение: Для проведения практической работы используется следующее обеспечение: персональный компьютер, подключённый к Интернету, MS Visual Studio

Время выполнения: 180 минут

Форма отчетности по занятию: вывод о проделанной работе

Задание 1. Разработайте набор тестовых сценариев (как позитивных, так и негативных) для следующей программы:

Имеется консольное приложение (разработайте самостоятельно). Ему на вход подается 2 строки. На выходе приложение выдает число вхождений второй строки в первую.

Например:

Строка 1	Строка 2	Вывод
абвгабвг	аб	2
стстсап	стс	2

Критерии оценивания:

Оценка 5 «отлично» работа выполнена полностью и правильно, сделаны правильные выводы

Оценка 4 «хорошо» работа выполнена правильно с учетом 1-2 несущественных ошибок, исправленных самостоятельно по требованию преподавателя

Оценка 3 «удовлетворительно» работа выполнена правильно не менее чем на половину или допущены 3-4 существенные ошибки

Оценка 2 «неудовлетворительно» допущены 5 и более существенные ошибки в ходе работы, которые обучающийся не может исправить даже по требованию преподавателя

Практическое занятие № 8

Тема раздела: оценка качества программных средств

Тема практического занятия: оценка программных средств с помощью метрик

Материально-техническое и комплексно-методическое обеспечение: Для проведения практической работы используется следующее обеспечение: персональный компьютер, подключённый к Интернету, MS Visual Studio

Время выполнения: 180 минут

Форма отчетности по занятию: вывод о проделанной работе

Задание №1. Провести сравнение понятий «качество» государственным и международным стандартами. Выписать документы, в которых даны данные определения.

Задание №2. Опишите методы получения информации о ПС по ГОСТу. Для каждого метода выделите источник информации.

Задание №3. Выберите стандарты для оценки качества ПС. Перечислите критерии надежности ПС по ГОСТу.

Задание №4. Методика оценки качественных показателей ПП основана на составлении метрики ПП. В лабораторной работе необходимо выполнить следующее:

- 1) Выбрать показатели качества (не менее 5) и сформулировать их сущность. Каждый показатель должен быть существенным, т. е. должны быть ясны потенциальные выгоды его использования. Показатели представить в виде таблицы (таблица 1).

Показатели качества	Сущность показателя	Экспертная оценка (вес) w_i	Оценка, установленная экспериментом r_i
---------------------	---------------------	-------------------------------	---

- 2) Установить веса показателей w_i ($\sum w_i = 1$).

- 3) Для каждого показателя установить конкретную численную оценку r_i от 0 до 1, исходя из следующего:

0 – свойство в ПП присутствует, но качество его неприемлемо;

0.5 — 1 – свойство в ПП присутствует и обладает приемлемым качеством;

1 – свойство в ПП присутствует и обладает очень высоким качеством.

Возможно, присвоение промежуточных значений в соответствии с

$$K = \frac{\sum w_i \cdot r_i}{\text{общее количество показателей}}$$

мнением оценивающего лица относительно полезности того или иного свойства программного продукта

Критерии оценивания:

Оценка 5 «отлично» работа выполнена полностью и правильно, сделаны правильные выводы

Оценка 4 «хорошо» работа выполнена правильно с учетом 1-2 несущественных ошибок, исправленных самостоятельно по требованию преподавателя

Оценка 3 «удовлетворительно» работа выполнена правильно не менее чем на половину или допущены 3-4 существенные ошибки

Оценка 2 «неудовлетворительно» допущены 5 и более существенные ошибки в ходе работы, которые обучающийся не может исправить даже по требованию преподавателя

Практическое занятие № 9

Тема раздела: оценка качества программных средств

Тема практического занятия: инспекция программного кода на пример соответствия стандартам кодирования

Материально-техническое и комплексно-методическое обеспечение: Для проведения практической работы используется следующее обеспечение: персональный компьютер, подключённый к Интернету, MS Visual Studio

Время выполнения: 180 минут

Форма отчетности по занятию: вывод о проделанной работе

Задание №1. Выполнить анализ программного кода для разрабатываемого ПО и модульных тестов с целью плохо организованного кода.

Задание №2. Используя шаблоны рефакторинга, выполнить реорганизацию программного кода разрабатываемого ПО.

Задание №3. Выполнить описание произведенных операций рефакторинга (было-стало-шаблон рефакторинга).

Задание №4 В случае необходимости скорректировать проектную документацию.

Задание №5 Сделать выводы по результатам выполнения работ.

Критерии оценивания:

Оценка 5 «отлично» работа выполнена полностью и правильно, сделаны правильные выводы

Оценка 4 «хорошо» работа выполнена правильно с учетом 1-2 несущественных ошибок, исправленных самостоятельно по требованию преподавателя

Оценка 3 «удовлетворительно» работа выполнена правильно не менее чем на половину или допущены 3-4 существенные ошибки

Оценка 2 «неудовлетворительно» допущены 5 и более существенные ошибки в ходе работы, которые обучающийся не может исправить даже по требованию преподавателя

Методические указания для выполнения практических работ

Практическое занятие № 1

Тема раздела: современные технологии и методы интеграции

Тема практического занятия: разработка структуры проекта

Цель: Формирование навыков постановки задачи и разработки технического задания на программный продукт

Материально-техническое и комплексно-методическое обеспечение: Для проведения практической работы используется следующее обеспечение: персональный компьютер, подключённый к Интернету, MS Visual Studio

Время выполнения: 90 минут

Форма отчетности по занятию: вывод о проделанной работе

Задание 1. Выбрать вариант задания на проектирование и разработку учебной программы. (№ варианта = № по списку в журнале)

Задание 2. В соответствии с вариантом выполнить разработку технического задания, которое должно включать:

- 1) введение;
- 2) основание для разработки;
- 3) назначение;
- 4) требования к программе и программному продукту;
- 5) требования к программной документации (технического задания)

Индивидуальные задания:

№ вар.	Задача
1	Ввести два целочисленных массива – по 10 элементов в каждом. Сформировать новый массив, на четных местах которого будут элементы с нечетными индексами из первого массива, а на нечетных – с четными индексами из второго.
2	Ввести массив, состоящий из 8 элементов (восемь двузначных чисел) целого типа. Получить новый массив, состоящий из цифр, находящихся в младших разрядах элементов исходного массива.
3	Ввести целочисленный массив, состоящий из 17-ти элементов (двузначные целые числа). Вычислить сумму цифр этого массива.
4	Ввести два массива действительных чисел, состоящих из 9 и 7 элементов. Сформировать третий массив из упорядоченных по возрастанию значений обоих массивов.
5	Ввести два массива X и Y , состоящих из 10-ти элементов целого типа. Сформировать массив S , состоящий из одинаковых элементов исходных массивов.
6	Рассчитать значения 12-ти элементов массива Y по формуле $y_i = i^2 - 2i + 19,3 \cos i$. Вывести на экран этот массив и новый, разместив в нем первоначально элементы, значения которых меньше среднего арифметического, а потом остальные, не меняя их последовательности.
7	Дан массив вещественных чисел $Z(16)$. Определить разность между суммой элементов с четными индексами и суммой элементов, индексы которых кратны трем.
8	В заданном целочисленном массиве $R(9)$ определить индекс наибольшего из нечетных по значению положительных элементов.
9	Ввести с клавиатуры массив X , состоящий из 15 элементов целого типа. Рассчитать элементы массива Y по формуле $y_i = \cos x_i^2 + 2,97 \lg^2 i^2$. Сформировать третий массив из упорядоченных по убыванию значений обоих массивов.

№ вар.	Задача
10	<p>Ввести с клавиатуры массив X, состоящий из 17 элементов целого типа. Рассчитать элементы массива Y по формуле</p> $y_i = \begin{cases} (x_i)^3 - 7,5, & \text{если } \cos(x_i) > 0 \\ x_i^2 - 5e^{\sin(x_i)}, & \text{если } \cos(x_i) \leq 0 \end{cases}$ <p>Упорядочить массив Y по возрастанию, массив X по убыванию и сформировать новый массив R, элементами которого являются четные по индексу элементы массива X и Y.</p>
11	Ввести массив, состоящий из 9 элементов (девять двузначных чисел) целого типа. Получить новый массив, состоящий из сумм цифр элементов исходного массива.
12	Ввести массив, состоящий из 12 элементов действительного типа. Расположить элементы в порядке убывания. Определить количество происшедших при этом перестановок.
13	Ввести с клавиатуры целочисленный массив, состоящий из 11 элементов. Вычислить сумму нечетных по значению отрицательных элементов и заменить элементы кратные трем на эту сумму.
14	Ввести массив, состоящий из 14 элементов действительного типа. Поменять местами первую половину со второй. Определить количество произведенных при этом перестановок.
15	Дан массив вещественных чисел. Определить элемент массива (значение и индекс), который наиболее удален от заданного вещественного числа S .
16	Ввести целочисленный массив, состоящий из 10 элементов. Определить сумму и количество элементов, расположенных до первого отрицательного числа.
17	Определить количество локальных минимумов в заданном числовом массиве. (Локальный минимум в числовом массиве – это последовательность трех рядом стоящих чисел, в которой среднее число меньше стоящих слева и справа от него).
18	Определить количество локальных максимумов в заданном числовом массиве. (Локальный максимум в числовом массиве – это последовательность трех рядом стоящих чисел, в которой среднее число больше стоящих слева и справа от него).
19	В заданном целочисленном массиве $Z(15)$ положительных, отрицательных и нулевых чисел определить сумму и вывести последовательность значений элементов, которые расположены между первым отрицательным и нулевым элементами.
20	В заданном числовом массиве определить и вывести индексы последовательностей чисел, которые монотонно убывают (каждое следующее число меньше предыдущего).

№ вар.	Задача
21	В заданном целочисленном массиве удалить элементы, которые встречаются более двух раз.
22	Ввести массив, состоящий из 10-ти элементов целого типа. Сформировать новый, расположив сначала все отрицательные элементы и нули, после чего - положительные, сохраняя порядок их следования.
23	Ввести массив X , состоящий из 10-ти элементов целого типа. Вычислить элементы массива Y по формуле $y_i = x_i^2 + 0,3$ $P = \frac{x_1 y_1 \cdot x_3 y_3 \cdot \dots \cdot x_9 y_9}{x_0 y_0 \cdot x_2 y_2 \cdot \dots \cdot x_8 y_8}$ и найти P . Определить остаток от деления.
24	Ввести массив, состоящий из 10 элементов (десять двузначных чисел) целого типа. Получить новый массив, состоящий из разностей цифр элементов исходного массива.
25	Ввести массив, состоящий из 15 элементов целого типа. Упорядочить массив так, чтобы все отрицательные числа были расположены вначале по возрастанию, а все положительные – в конце по убыванию.
26	Даны два массива действительных чисел по 12 элементов в каждом. Заменить нулями те элементы первого массива, которые есть во втором.
27	Задан целочисленный массив. Определить количество участков массива, на котором элементы монотонно возрастают (каждое следующее число больше предыдущего).
28	Задан целочисленный массив. Определить остаток от деления суммы элементов с четными индексами на сумму элементов с нечетными индексами.
29	Задан целочисленный массив. Определить процентное содержание элементов, превышающих среднеарифметическое всех элементов массива.
30	Ввести два массива действительных чисел. Определить максимальные элементы в каждом массиве и поменять их местами.

Критерии оценивания:

Оценка 5 «отлично» работа выполнена полностью и правильно, сделаны правильные выводы

Оценка 4 «хорошо» работа выполнена правильно с учетом 1-2 несущественных ошибок, исправленных самостоятельно по требованию преподавателя

Оценка 3 «удовлетворительно» работа выполнена правильно не менее чем на половину или допущены 3-4 существенные ошибки

Оценка 2 «неудовлетворительно» допущены 5 и более существенные ошибки в ходе работы, которые обучающийся не может исправить даже по требованию преподавателя

Практическое занятие № 2

Тема раздела: современные технологии и методы интеграции

Тема практического занятия: разработка перечня артефактов и протоколов проекта

Цель: Формирование навыков работы с техническим заданием на программный продукт

Материально-техническое и комплексно-методическое обеспечение: Для проведения практической работы используется следующее обеспечение: персональный компьютер, подключённый к Интернету, MS Visual Studio

Время выполнения: 90 минут

Задание В соответствии с подготовленным техническим заданием выполнить разработку спецификаций на программный продукт, которые должны включать:

- 1) спецификации процессов;
- 2) словарь терминов;
- 3) диаграммы переходов состояний;
- 4) диаграммы потоков с детализацией.

Критерии оценивания:

Оценка 5 «отлично» работа выполнена полностью и правильно, сделаны правильные выводы

Оценка 4 «хорошо» работа выполнена правильно с учетом 1-2 несущественных ошибок, исправленных самостоятельно по требованию преподавателя

Оценка 3 «удовлетворительно» работа выполнена правильно не менее чем на половину или допущены 3-4 существенные ошибки

Оценка 2 «неудовлетворительно» допущены 5 и более существенные ошибки в ходе работы, которые обучающийся не может исправить даже по требованию преподавателя

Практическое занятие № 3

Тема раздела: современные технологии и методы интеграции

Тема практического занятия: Настройка работы системы контроля версий (типов импортируемых файлов, путей, фильтров и др. параметров импорта в репозиторий)

Цель: Формирование навыков работы с репозиториями

Материально-техническое и комплексно-методическое обеспечение: Для проведения практической работы используется следующее обеспечение: персональный компьютер, подключённый к Интернету, MS Visual Studio

Время выполнения: 90 минут

Задание

1. Настроить подключение к репозиторию
2. Скачать проект
3. Добавить свой класс к проекту
4. Внести изменения к класс
5. Обновить класс в репозитории
6. Удалить все локальные файлы и скачать проект из репозитория
7. Добавить "лишний" файл в репозиторий и затем удалить его из репозитория.
8. Изучить журнал изменений файлов, посмотреть какие изменения внесены другими разработчиками.

Критерии оценивания:

Оценка 5 «отлично» работа выполнена полностью и правильно, сделаны правильные выводы

Оценка 4 «хорошо» работа выполнена правильно с учетом 1-2 незначительных ошибок, исправленных самостоятельно по требованию преподавателя

Оценка 3 «удовлетворительно» работа выполнена правильно не менее чем на половину или допущены 3-4 существенные ошибки

Оценка 2 «неудовлетворительно» допущены 5 и более существенные ошибки в ходе работы, которые обучающийся не может исправить даже по требованию преподавателя

Практическое занятие № 4

Тема раздела: современные технологии и методы интеграции

Тема практического занятия: Разработка интеграции модуля.

Цель: Формирование навыков работы с репозиториями

Материально-техническое и комплексно-методическое обеспечение: Для проведения практической работы используется следующее обеспечение: персональный компьютер, подключённый к Интернету, MS Visual Studio

Время выполнения: 90 минут

Задание

- 1) Описать этапы проектирования модулей программы в соответствии с индивидуальным заданием.
- 2) Составить в виде блок-схемы алгоритм решения задачи.
- 3) Составить отчет по практической работе.

Варианты индивидуальных заданий.

1. Даны два двумерных массива вещественных элементов. Размер исходных массивов не превосходит 10×10 элементов. Для каждого из массивов указать номера столбцов, содержащих только положительные элементы. Если таковых столбцов в массиве нет, то вывести соответствующее сообщение. Проверку столбца на положительность элементов оформить в виде процедуры с передачей в нее всех элементов текущего столбца.
2. Даны два двумерных массива натуральных элементов. Размер исходных массивов не превосходит 10×10 элементов. Для каждого из массивов указать номера столбцов, содержащих только кратные 5 или 7 элементы. Если таких столбцов в массиве нет, то вывести соответствующее сообщение. Проверку столбца на наличие указанных элементов оформить в виде процедуры с передачей в нее всех элементов текущего столбца.
3. Даны пять одномерных массива вещественных элементов. Размер каждого массива не превосходит 100 элементов. Для каждого из массивов определить, составляют ли его элементы знакопеременяющуюся последовательность. Если да, то указать порядковый номер такого массива, в противном случае вывести отрицательный ответ. Проверку массива на выполнение условия оформить в виде процедуры с передачей в нее всех элементов рассматриваемого массива.
4. Даны два двумерных массива символьных (буквы русского алфавита) элементов. Размер исходных массивов не превосходит 10×10 элементов. Для каждого из массивов указать номера строк, содержащих элементы только строчных букв, если таких строк нет ни для какого массива, то вывести соответствующее сообщение. Проверку строки на наличие указанных элементов оформить в виде процедуры с передачей в нее всех элементов текущей строки.

5. Даны два двумерных массива вещественных элементов. Размер исходных массивов не превосходит 10×10 элементов. Для каждого из массивов указать количество столбцов, содержащих только не положительные элементы. Если таких столбцов нет ни для одного из массивов, то вывести соответствующее сообщение. Проверку столбца на наличие указанных элементов оформить в виде процедуры с передачей в нее всех элементов текущего столбца.
6. Даны пять одномерных массива вещественных элементов. Размер каждого массива не превосходит 100 элементов. Для каждого из массивов определить, составляют ли его элементы одного знака. Если да, то указать порядковый номер такого массива, в противном случае вывести отрицательный ответ. Проверку массива на выполнение условия оформить в виде процедуры с передачей в нее всех элементов рассматриваемого массива.
7. Даны два двумерных массива целочисленных элементов. Размер исходных массивов не превосходит 10×10 элементов. Для каждого из массивов указать количество строк, содержащих элементы, четность которых чередуется, а вторым в четных строках является нечетный элемент. Если таких строк нет ни для одного из массивов, то вывести соответствующее сообщение. Проверку строки на наличие указанных элементов оформить в виде процедуры с передачей в нее всех элементов текущего столбца.
8. Даны пять одномерных массива символьных (только латинские буквы) элементов. Размер каждого массива не превосходит 100 элементов. Для каждого из массивов определить, чередуются ли в нем буквы строчные и прописные. Если да, то указать порядковый номер такого массива, в противном случае вывести отрицательный ответ. Проверку массива на выполнение условия оформить в виде процедуры с передачей в нее всех элементов рассматриваемого массива.
9. Даны два двумерных массива целочисленных элементов. Размер исходных массивов не превосходит 10×10 элементов. Для каждого из массивов указать количество строк, для которых сумма элементов, стоящих на нечетных местах в строке, является положительным числом. Если таких строк нет ни для одного из массивов, то вывести соответствующее сообщение. Проверку строки на выполнение условия и расчет оформить в виде процедуры с передачей в нее всех элементов текущей строки.
10. Даны два двумерных массива вещественных элементов. Размер исходных массивов не превосходит 10×10 элементов. Для каждого из массивов указать номера столбцов, произведение отрицательных элементов которых является положительным числом. Если таких столбцов нет ни для одного из массивов, то вывести соответствующее сообщение. Проверку столбца на выполнение условия и расчет оформить в виде процедуры с передачей в нее всех элементов текущего столбца.

11. Даны пять одномерных массива символьных (только латинские буквы) элементов. Размер каждого массива не превосходит 100 элементов. Для каждого из массивов определить, расположены ли в нем строчные буквы в алфавитном порядке. Если да, то указать порядковый номер такого массива, в противном случае вывести отрицательный ответ.
12. Проверку массива на выполнение условия оформить в виде процедуры с передачей в нее всех элементов рассматриваемого массива.
13. Даны два двумерных массива целочисленных элементов. Размер исходных массивов не превосходит 10×10 элементов. Для каждого из массивов проверить выполнение условия: все четные строки массива таковы, что суммы их элементов образуют возрастающую последовательность. Вывести соответствующее сообщение. Вычисление суммы элементов массива и проверку последовательности чисел на выполнение условия оформить в виде процедуры с передачей в нее всех необходимых элементов.
14. Даны два двумерных массива вещественных элементов. Размер исходных массивов не превосходит 10×10 элементов. Преобразовать все нечетные строки каждого массива так, чтобы элементы составляли возрастающую по абсолютной величине последовательность. Вывести преобразованные массивы. Упорядочивание элементов оформить в виде процедуры с передачей в нее всех необходимых элементов.
15. Даны два двумерных массива целочисленных элементов. Размер исходных массивов не превосходит 10×10 элементов. Для каждого столбца массивов вычислить суммы и количества элементов, значения которых находятся в заданном диапазоне. Если чисел, удовлетворяющих этому условию нет, то вывести соответствующее сообщение. Вычисление для элементов столбца массива оформить в виде процедуры с передачей в нее всех необходимых элементов.
16. Даны пять одномерных массива символьных (только латинские буквы) элементов. Размер каждого массива не превосходит 100 элементов. Преобразовать все массивы так, чтобы все строчные буквы были расположены по алфавиту. При этом переставлять только строчные буквы, оставив прописные буквы на своих местах. Преобразование каждого массива оформить в виде процедуры с передачей в нее всех необходимых элементов. Если перестановка элементов не потребовалась, то есть исходные массивы удовлетворяют требуемому условию, то вывести соответствующее сообщение.

Критерии оценивания:

Оценка 5 «отлично» работа выполнена полностью и правильно, сделаны правильные выводы

Оценка 4 «хорошо» работа выполнена правильно с учетом 1-2 несущественных ошибок, исправленных самостоятельно по требованию преподавателя

Оценка 3 «удовлетворительно» работа выполнена правильно не менее чем на половину или допущены 3-4 существенные ошибки

Оценка 2 «неудовлетворительно» допущены 5 и более существенные ошибки в ходе работы, которые обучающийся не может исправить даже по требованию преподавателя

Практическое занятие № 5

Тема раздела: современные технологии и методы интеграции

Тема практического занятия: Отладка отдельных модулей программного проекта

Цель: Формирование навыков работы с отладкой

Материально-техническое и комплексно-методическое обеспечение: Для проведения практической работы используется следующее обеспечение: персональный компьютер, подключённый к Интернету, MS Visual Studio

Время выполнения: 90 минут

Задание 1. Оформить внешнюю спецификацию.

Задание 2. Составить в виде блок-схемы алгоритм решения задачи, исходя из индивидуального задания .

Задание 3. Создать программу решения задачи

Задание 4. Составить набор тестов и провести тестирование созданной программы с помощью методов «белого ящика» (покрытия операторов, покрытия решений, покрытия условий, комбинаторного покрытия условий).

Индивидуальные задания:

№ вар.	Задание
1	В произвольной матрице - отсортировать по убыванию элементы последовательности, расположенные после второго отрицательного числа.
2	Необходимо заполнить двухмерный массив из 0 и 1. А после его вывода - массив должен иметь следующий вид: 0 1 0 1 1 0 1 0 0 1 0 1 1 0 1 0

№ вар.	Задание
3	Необходимо заполнить двухмерный массив . А после его вывода - массив должен иметь следующий вид: 01 02 03 04 12 13 14 05 11 16 15 06 10 09 08 07
4	Дан массив $A(n, m)$. Удалить строки массива, не имеющие ни одного повторяющегося элемента.
5	Заполнить массив 3x3 числами по возрастанию, по спирали начиная с центра. 7 8 9 6 1 2 5 4 3
6	Элементы матрицы A сделать с помощью генератора случайных чисел. Сделать новую матрицу B , в которой удалить с матрицы A ряд, в котором минимальный элемент среди элементов главной диагонали.
7	Составить программу, которая заполняет квадратную матрицу порядка n натуральными числами 1, 2, 3, ..., n^2 , записывая их в нее "по спирали" против часовой стрелки.
8	Составить программу, которая заполняет квадратную матрицу порядка n натуральными числами 1, 2, 3, ..., n^2 , записывая их в нее "по спирали" по часовой стрелке.
9	Дан двухмерный целочисленный массив $A(M, N)$. Составить одномерный массив B из номеров строк этого массива.
10	Написать программу, которая в матрице чисел $A(N, M)$ находит все элементы, превышающие по абсолютной величине заданное число B . Подсчитать число таких элементов и записать их в массив C .
11	Написать программу, которая в матрице чисел $A(N, M)$ находит все элементы, равные числу, введенному с клавиатуры. Подсчитать число таких элементов.
12	Задан двумерный массив $A[5, 10]$. Получить новую матрицу путем деления всех элементов исходной матрицы на ее наибольший по модулю элемент.
13.	Дан двумерный массив. Вставьте первую строку после строки, в которой находится первый встреченный минимальный элемент.
14.	Дан целочисленный массив $B[1..5, 1..5]$. Вычислить произведение элементов этого массива, расположенных ниже левой диагонали.
15	Дан целочисленный массив $B[1..5, 1..5]$. Вычислить сумму элементов этого массива, расположенных выше левой диагонали.
16	Дана целочисленная матрица размера 5x5. Заменить в данной матрице все отрицательные элементы первой строки числом 0.

№ вар.	Задание
17	Дана целочисленная матрица размера 5×5 . Получить новую матрицу путем деления всех элементов данной матрицы на ее наибольший по модулю элемент.
18	Дана целочисленная прямоугольная матрица размера $M \cdot N$. Отсортировать каждый столбец с четным номером по неубыванию, а каждый столбец с нечетным номером - по невозрастанию.
19	Дана целочисленная матрица размера 8×5 . Определить: а) сумму всех элементов второго столбца массива; б) сумму всех элементов 3-й строки массива.
20	Дана целочисленная прямоугольная матрица размера $M \cdot N$. Сформировать одномерный массив, состоящий из элементов, лежащих в интервале $[1, 20]$. Найти среднеарифметическое полученного одномерного массива.
21	Дана целочисленная прямоугольная матрица размера $M \cdot N$. Сформировать одномерный массив, состоящий из элементов, лежащих в интервале $[1, 10]$. Найти произведение элементов полученного одномерного массива.
22	Дана целочисленная квадратная матрица. Найти в каждой строке наибольший элемент и поменять его местами с элементом главной диагонали.
23	Дана целочисленная квадратная матрица. Указать столбец (назвать его номер), где минимальное количество элементов, кратных сумме индексов.
24	Дана целочисленная квадратная матрица. Найти сумму элементов матрицы, лежащих выше главной диагонали.
25	Определить, является ли данный квадратный массив симметричным относительно своей главной диагонали.
26	Определить, является ли данный квадратный массив не симметричным относительно своей главной диагонали.
27	Даны два числа n и m . Создайте двумерный массив <code>int A[n][m]</code> , заполните его таблицей умножения $A[i][j] = i * j$ и выведите на экран. При этом нельзя использовать вложенные циклы, все заполнение массива должно производиться одним циклом, например, <code>for(i=0; i<n*m; ++i)</code> .
28	Дана матрица целых чисел размера $N \times M$. Вывести номер строки, содержащей минимальное число одинаковых элементов.
29	Дана целочисленная квадратная матрица. Найти произведение элементов матрицы, лежащих ниже главной диагонали.
30	Дана матрица целых чисел размера $N \times M$. Вывести номер строки, содержащей максимальное число одинаковых элементов.

Критерии оценивания:

Оценка 5 «отлично» работа выполнена полностью и правильно, сделаны правильные выводы

Оценка 4 «хорошо» работа выполнена правильно с учетом 1-2 несущественных ошибок, исправленных самостоятельно по требованию преподавателя

Оценка 3 «удовлетворительно» работа выполнена правильно не менее чем на половину или допущены 3-4 существенные ошибки

Оценка 2 «неудовлетворительно» допущены 5 и более существенные ошибки в ходе работы, которые обучающийся не может исправить даже по требованию преподавателя

Практическое занятие № 6

Тема раздела : современные технологии и методы интеграции

Цель работы: Обработка исключительных ситуаций

Материально-техническое и комплексно-методическое обеспечение: Для проведения практической работы используется следующее обеспечение: персональный компьютер, подключённый к Интернету,

Время выполнения: 90 минут

Форма отчетности по занятию: файл с выполненной работой

Последовательность выполнения работы

1. Выполнить задание
2. Сохранить файл
3. Оформить отчет о выполненной работе

Задание. Разработать программу, в которой обрабатываются следующие исключительные ситуации:

- 1) Отрицательное значение возраста
- 2) Год рождения больше текущей даты

Критерии оценивания:

Оценка 5 «отлично» работа выполнена полностью и правильно, сделаны правильные выводы

Оценка 4 «хорошо» работа выполнена правильно с учетом 1-2 несущественных ошибок, исправленных самостоятельно по требованию преподавателя

Оценка 3 «удовлетворительно» работа выполнена правильно не менее чем на половину или допущены 3-4 существенные ошибки

Оценка 2 «неудовлетворительно» допущены 5 и более существенные ошибки в ходе работы, которые обучающийся не может исправить даже по требованию преподавателя

Практическое занятие № 7

Тема раздела : Инструментарий тестирования и анализа качества программных средств

Цель работы: применение отладочных классов в проекте

Материально-техническое и комплексно-методическое обеспечение: Для проведения практической работы используется следующее обеспечение: персональный компьютер, подключённый к Интернету,

Время выполнения: 180 минут

Форма отчетности по занятию: файл с выполненной работой

Задание 1. Изучить теоретические сведения

Теоретические сведения

Сущность объектно-ориентированного подхода к программированию заключается в том, что основные идеи объектно-ориентированного подхода опираются на следующие положения:

- программа представляет собой модель некоторого реального процесса, части реального мира;

- модель реального мира или его части может быть описана как совокупность взаимодействующих между собой объектов;

- объект описывается набором параметров, значения которых определяют состояние объекта, и набором операций (действий), которые может выполнять объект;

- взаимодействие между объектами осуществляется посылкой специальных сообщений от одного объекта к другому. Сообщение, полученное объектом, может потребовать выполнения определенных действий, например, изменения состояния объекта;

- объекты, описанные одним и тем же набором параметров и способные выполнять один и тот же набор действий представляют собой класс однотипных объектов.

С точки зрения языка программирования класс объектов можно рассматривать как тип данного, а отдельный объект - как данное этого типа. Определение программистом собственных классов объектов для конкретного набора задач должно позволить описывать отдельные задачи в терминах самого класса задач (при соответствующем выборе имен типов и имен объектов, их параметров и выполняемых действий).

Таким образом, объектно-ориентированный подход предполагает, что при разработке программы должны быть определены классы используемых в программе объектов и построены их описания, затем созданы экземпляры необходимых объектов и определено взаимодействие между ними.

Классы объектов часто удобно строить так, чтобы они образовывали иерархическую структуру. Например, класс «Студент», описывающий абстрактного студента, может служить основой для построения классов «Студент 1 курса», «Студент 2 курса» и т.д., которые обладают всеми свойствами студента вообще и некоторыми дополнительными свойствами, характеризующими студента конкретного курса. При

разработке интерфейса с пользователем программы могут использовать объекты общего класса «Окно» и объекты классов специальных окон, например, окон информационных сообщений, окон ввода данных и т.п. В таких иерархических структурах один класс может рассматриваться как базовый для других, производных от него классов. Объект производного класса обладает всеми свойствами базового класса и некоторыми собственными свойствами, он может реагировать на те же типы сообщений от других объектов, что и объект базового класса и на сообщения, имеющие смысл только для производного класса. Обычно говорят, что объект производного класса наследует все свойства своего базового класса.

Некоторые параметры объекта могут быть локализованы внутри объекта и недоступны для прямого воздействия извне объекта. Например, во время движения объекта-автомобиля объект-водитель может воздействовать только на ограниченный набор органов управления (рулевое колесо, педали газа, сцепления и тормоза, рычаг переключения передач) и ему недоступен целый ряд параметров, характеризующих состояние двигателя и автомобиля в целом.

Существует ряд техник, предназначенных для обеспечения качества кода, выполняемых по мере его конструирования.

Основные техники обеспечения качества, используемые в процессе конструирования, включают:

1. рефакторинг;
2. модульное (unit) и интеграционное (integration) тестирование;
3. разработка с первичностью тестов (test-first development - тесты пишутся до конструирования кода);
4. пошаговое кодирование (деятельность по конструированию кода разбивается на мелкие шаги, только после тестирования результатов которых производится переход к следующему шагу кодирования; известен также как итеративное кодирование с тестированием);
5. использование процедур утверждений (assertion);
6. отладка (в привычном понимании — debugging);
7. технические обзоры и оценки (review);
8. статический анализ.

Выбор и использование конкретных техник часто диктуется стандартами (внутренними и внешними), используемыми проектной командой, а также зависят от опыта и подготовленности специалистов, занимающихся конструированием кода.

Ключевым аспектом защитного программирования является использование отладчиков.

Отладчик (debugger) — программа, предназначенная для поиска ошибок в других программах и позволяет выполнять трассировку, отслеживать, устанавливать или изменять значения переменных в процессе выполнения кода, устанавливать и удалять контрольные точки или условия остановки и т. д.

При программировании важно:

1. Не использовать ограничения промышленной версии в отладочной версии.
2. Внедрять поддержку отладки как можно раньше.

3. Использовать наступательное программирование:

- реализовать так, чтобы все утверждения завершали работу программы;
- заполнять всю выделенную память, для обнаружения ошибок выделения памяти;
- заполнять все файлы и потоки;
- при попадании в операторе case в ветви default или else программа прекращает работу;
- заполнять объекты мусором перед их удалением;
- настроить отправку журналов ошибок по электронной почте.

4. Запланировать удаление отладочных средств. Для реализации этого использовать средства автоматизации контроля версий и сборки программ (ant, maven, make), встроенный или собственный препроцессор, отладочные заглушки.

В промышленной версии желательно придерживаться следующих рекомендаций:

1. Оставить код, который проверяет только существенные ошибки.
2. Удалить код, проверяющий незначительные ошибки.
3. Удалить код, приводящий к прекращению работы программы.
4. Оставить код, который позволяет аккуратно завершить программу.
5. Регистрировать ошибки для технической поддержки.
6. Оставленные сообщения об ошибках должны быть дружелюбны.

Задание 2. Создать основной класс исходя из варианта индивидуального задания. Создать 3 класса наследника. Создать в основном классе 5 методов и 3 свойства, связанных с его назначением, отличных от методов и свойств других подгрупп.

Индивидуальные задания

1,11,21. Основной класс «Автомобили», 3 наследника «Грузовики», «Легковые» и «Автобусы».

2,12,22. Основной класс «Одежда», 3 наследника «Куртки», «Шубы» и «Пуховики».

3,13,23. Основной класс «Еда», 3 наследника «Супы», «Закуски» и «Напитки».

4,14,24. Основной класс «Гаджеты», 3 наследника «Планшеты», «Смартфоны» и «Нетбуки».

5,15,25. Основной класс «Овощи», 3 наследника «Морковь», «Свёкла» и «Лук».

6,16,26. Основной класс «Песни», 3 наследника «Частушки», «Баллады» и «Романсы».

7,17,27.

Основной класс «Оружие», 3 наследника «Пистолеты», «Автоматы» и «Пулеметы».

8,18,28.

Основной класс «Документы», 3 наследника «Справки», «Приказы» и «Заявления».

9,19,29. Основной класс «Погода», 3 наследника «Пасмурно», «Ясно» и «Ураган».

10,20,30. Основной класс «Звери», 3 наследника «Зайцы», «Волки» и «Лисы».

Задание 3. Придумать и создать, как минимум, по 2 метода в каждом классе-наследнике (всего не меньше 6) и хотя бы по 2 свойства (всего не меньше 6). Методы и свойства должны быть связаны с особенностями класса-наследника.

Задание 4. Сделать, как минимум, 3 метода основного класса переопределёнными в классах наследниках. Сделать, как минимум, 1 метод, который нельзя переопределить.

Задание 5. Скомпилировать программу, демонстрирующую работу каждого из методов каждого из классов.

Критерии оценивания:

Оценка 5 «отлично» работа выполнена полностью и правильно, сделаны правильные выводы

Оценка 4 «хорошо» работа выполнена правильно с учетом 1-2 несущественных ошибок, исправленных самостоятельно по требованию преподавателя

Оценка 3 «удовлетворительно» работа выполнена правильно не менее чем на половину или допущены 3-4 существенные ошибки

Оценка 2 «неудовлетворительно» допущены 5 и более существенные ошибки в ходе работы, которые обучающийся не может исправить даже по требованию преподавателя

Практическое занятие № 8

Тема раздела : Инструментарий тестирования и анализа качества программных средств

Цель работы: отладка однопоточного приложения

Материально-техническое и комплексно-методическое обеспечение: Для проведения практической работы используется следующее обеспечение: персональный компьютер, подключённый к Интернету,

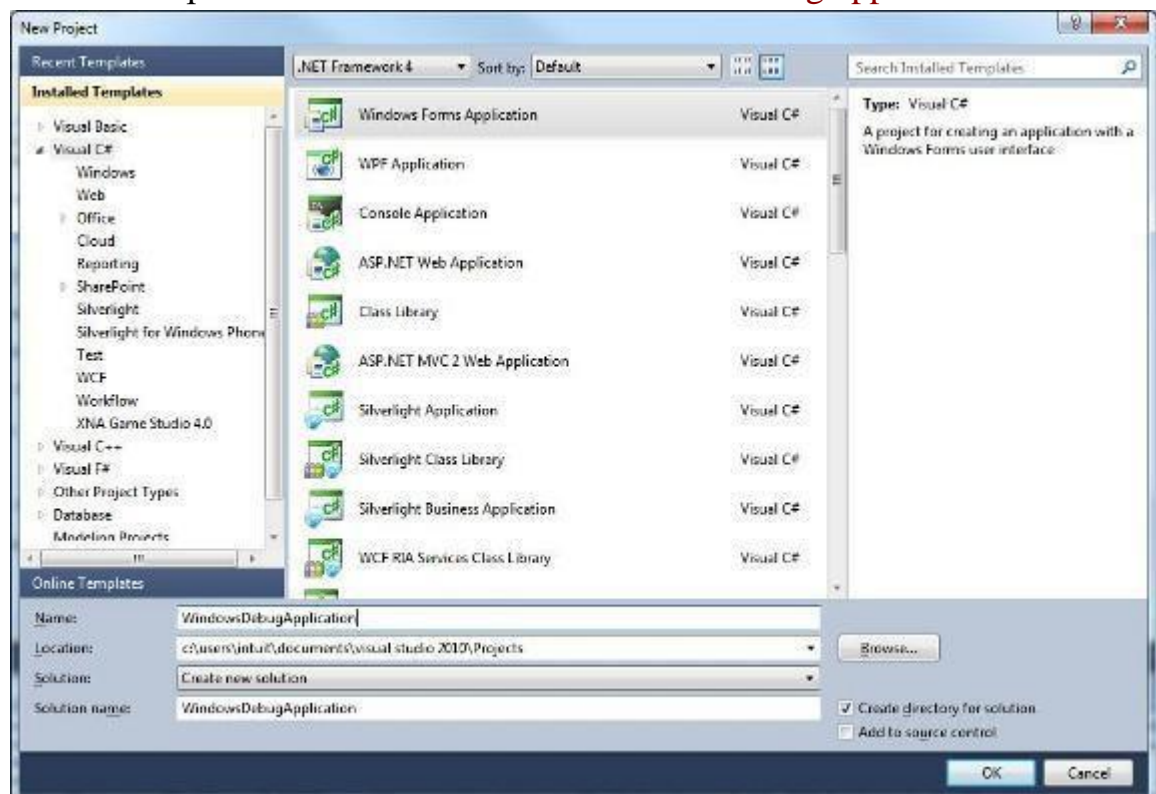
Время выполнения: 180 минут

Форма отчетности по занятию: файл с выполненной работой

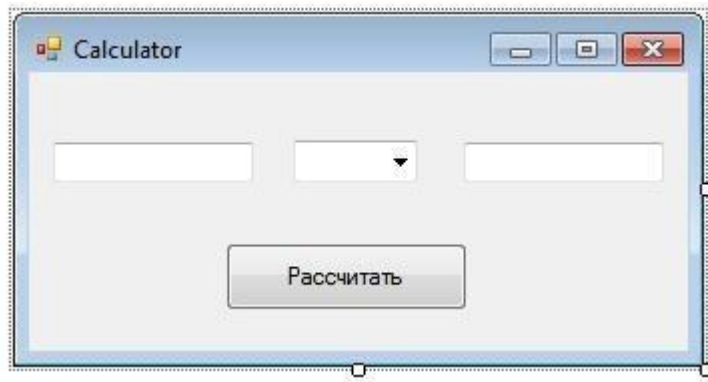
Задание 1. Изучить теоретические сведения

Теоретические сведения

Создадим Windows приложение с названием **"WindowsDebugApplication"**:



Создадим простейший калькулятор. Для этого разместим на форме 4 элемента (2 **TextBox**, 1 **ComboBox**, 1 **Button**):



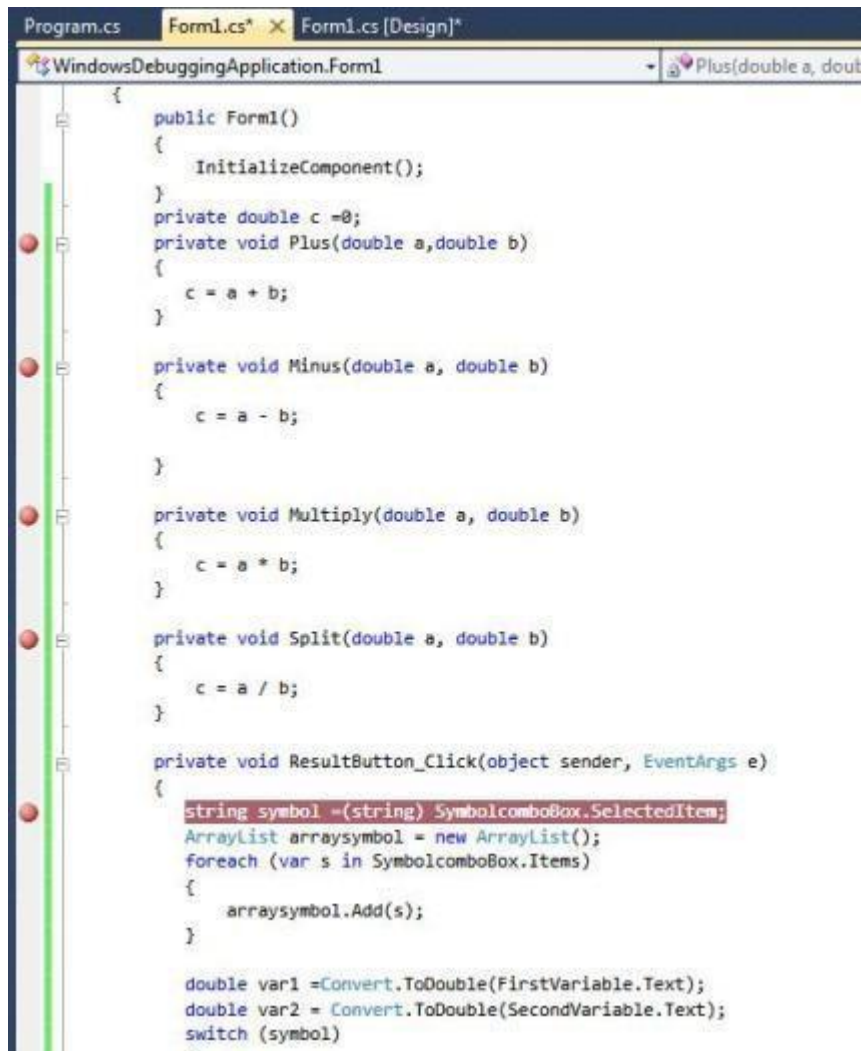
1. Добавим в программу следующий код (Коллекция **arraysymbol** и цикл **foreach** добавлены для наглядности отладки приложения):
2. `using System;`
3. `using System.Collections.Generic;`
4. `using System.ComponentModel;`
5. `using System.Data;`
6. `using System.Drawing;`
7. `using System.Linq;`
8. `using System.Text;`
9. `using System.Windows.Forms;`
10. `using System.Collections;`
11. `namespace WindowsDebugApplication`
12. `{`
13. `public partial class Form1 : Form`
14. `{`
15. `public Form1()`
16. `{`
17. `InitializeComponent();`
18. `}`
19. `private double c = 0;`
20. `private void Plus(double a, double b)`
21. `{`
22. `c = a + b;`
23. `}`
24. `private void Minus(double a, double b)`
25. `{`
26. `c = a - b;`
27. `}`
28. `private void Multiply(double a, double b)`
29. `{`
30. `c = a * b;`
31. `}`
32. `private void Split(double a, double b)`

```

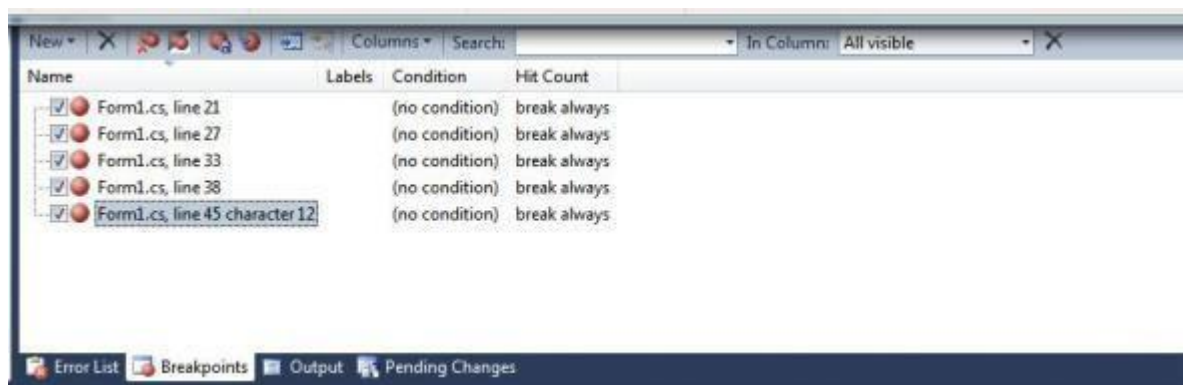
33.     {
34.         c = a / b;
35.     }
36. private void ResultButton_Click(object sender, EventArgs e)
37.     {
38.         string symbol =(string) SymbolcomboBox.SelectedItem;
39.         ArrayList arraysymbol = new ArrayList();
40.         foreach (var s in SymbolcomboBox.Items)
41.         {
42.             arraysymbol.Add(s);
43.         }
44.         double var1 =Convert.ToDouble(FirstVariable.Text);
45.         double var2 = Convert.ToDouble(SecondVariable.Text);
46.         switch (symbol)
47.         {
48.             case "+": Plus(var1, var1); break;
49.             case "-": Minus(var1, var1); break;
50.             case "*": Multiply(var1, var1); break;
51.             case "/": Split(var1, var1); break;
52.         }
53.         MessageBox.Show("Результат: " + c);
54.         MessageBox.Show("Количество символов в коллекции: "+
arraysymbol.Count);
55.     }
56. }
    }

```

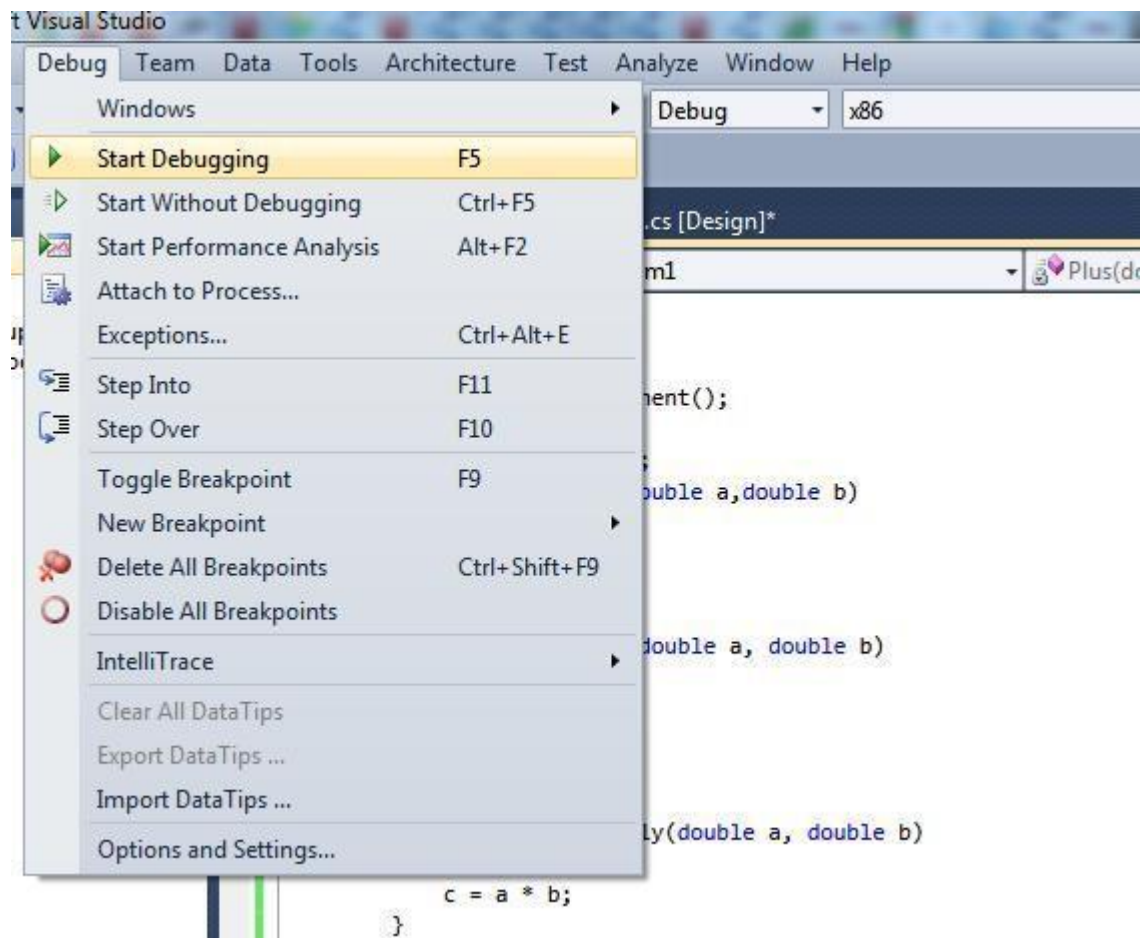
Теперь, расставим точки останова (**breakpoints**) в программе. В примере точки останова расставлены напротив методов математических операций и в событии кнопки:



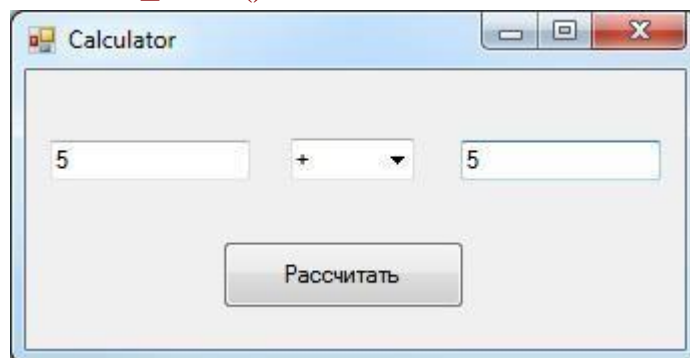
Список доступных всех точек останова (**Breakpoints**), можно посмотреть в специальном окне, которое вызывается из меню **Debug** пункт меню "**Breakpoints**" или сочетанием клавиш **CTRL+ALT+B**:



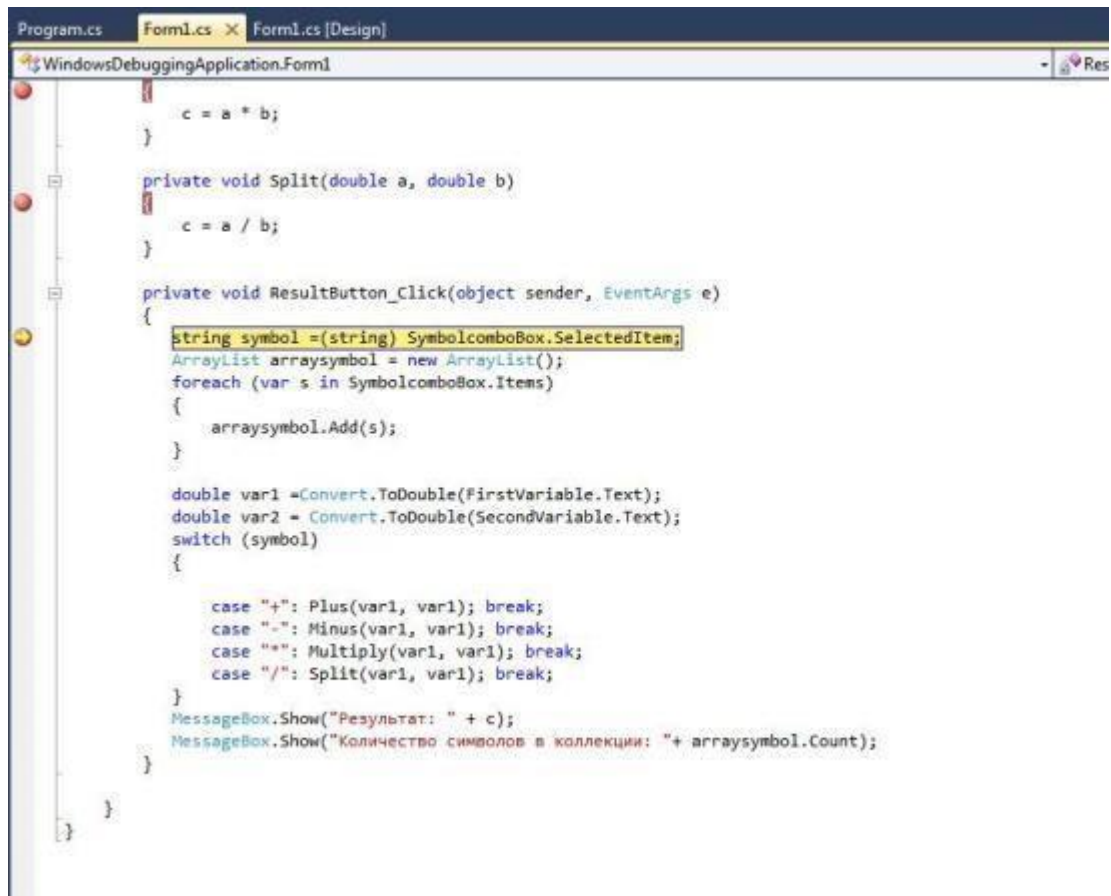
Запустите отладку приложения с помощью пункта "**Start Debugging**" - меню "**Debbug**" или с помощью клавиши "**F5**":



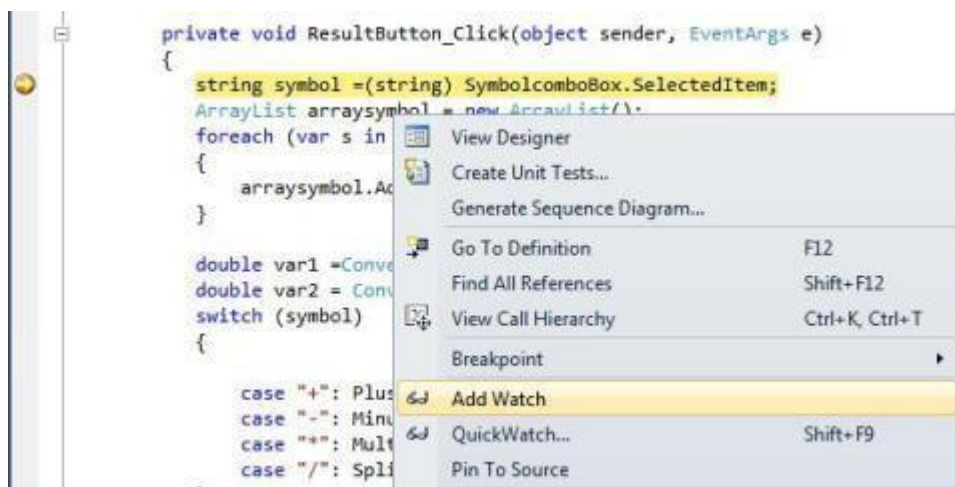
После ввода значений в поля программы и выбора соответствующей операции (сложение, вычитание и т.д), жмем кнопку "Рассчитать", тем самым вызовется метод **ResulButton_Click()**:



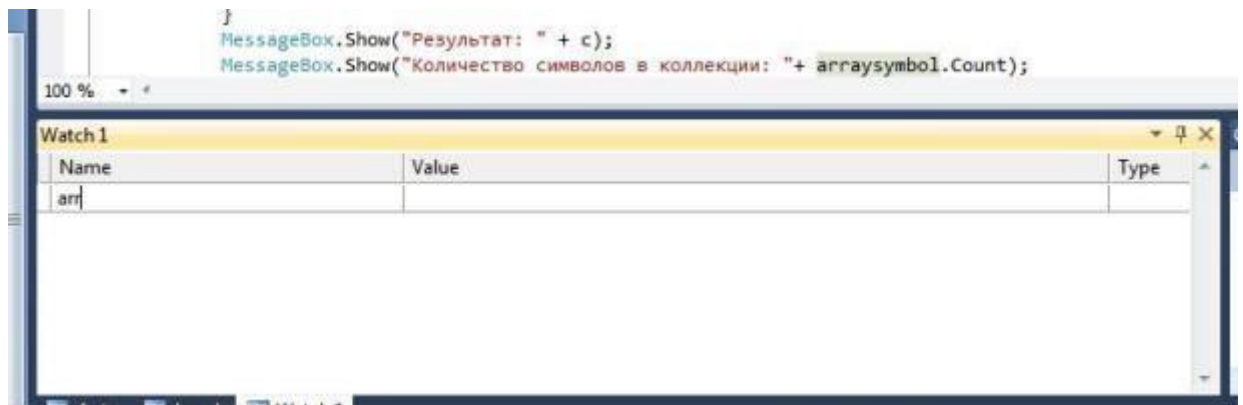
Запустится пошаговый процесс отладки приложения с точки останова (**Breakpoint**) в методе **ResulButton_Click()**:



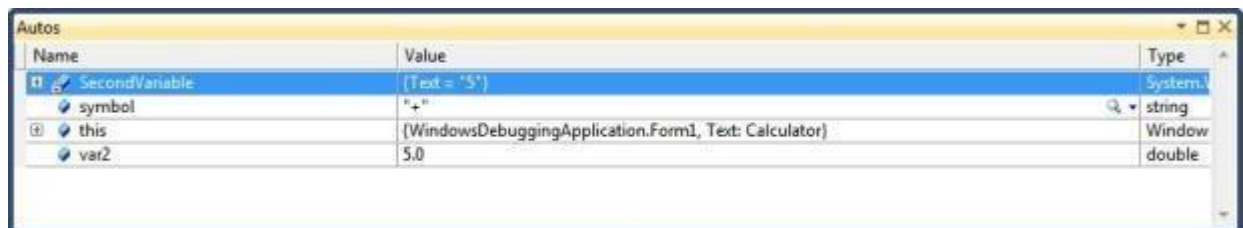
Добавим для просмотра значений переменных - переменные **arraysymbol** (коллекция), и переменную **"c"**. Для этого щелкните на нужной переменной правой кнопкой мыши и выберите из списка **"Add Watch"**:



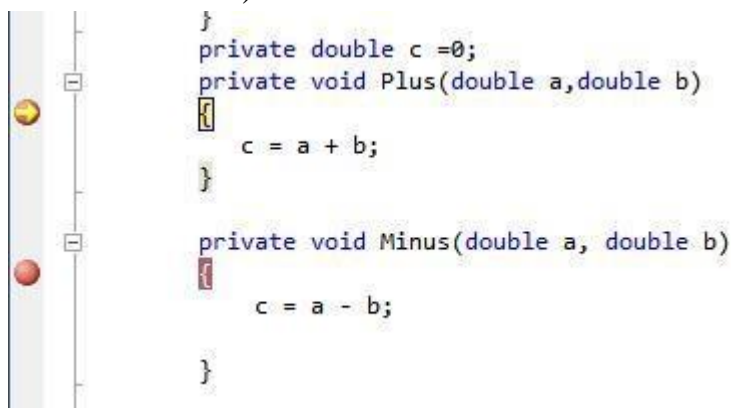
Переменные также, можно вручную добавлять в список **Watch**, для этого, достаточно написать имя нужной переменной в колонке **Name**:



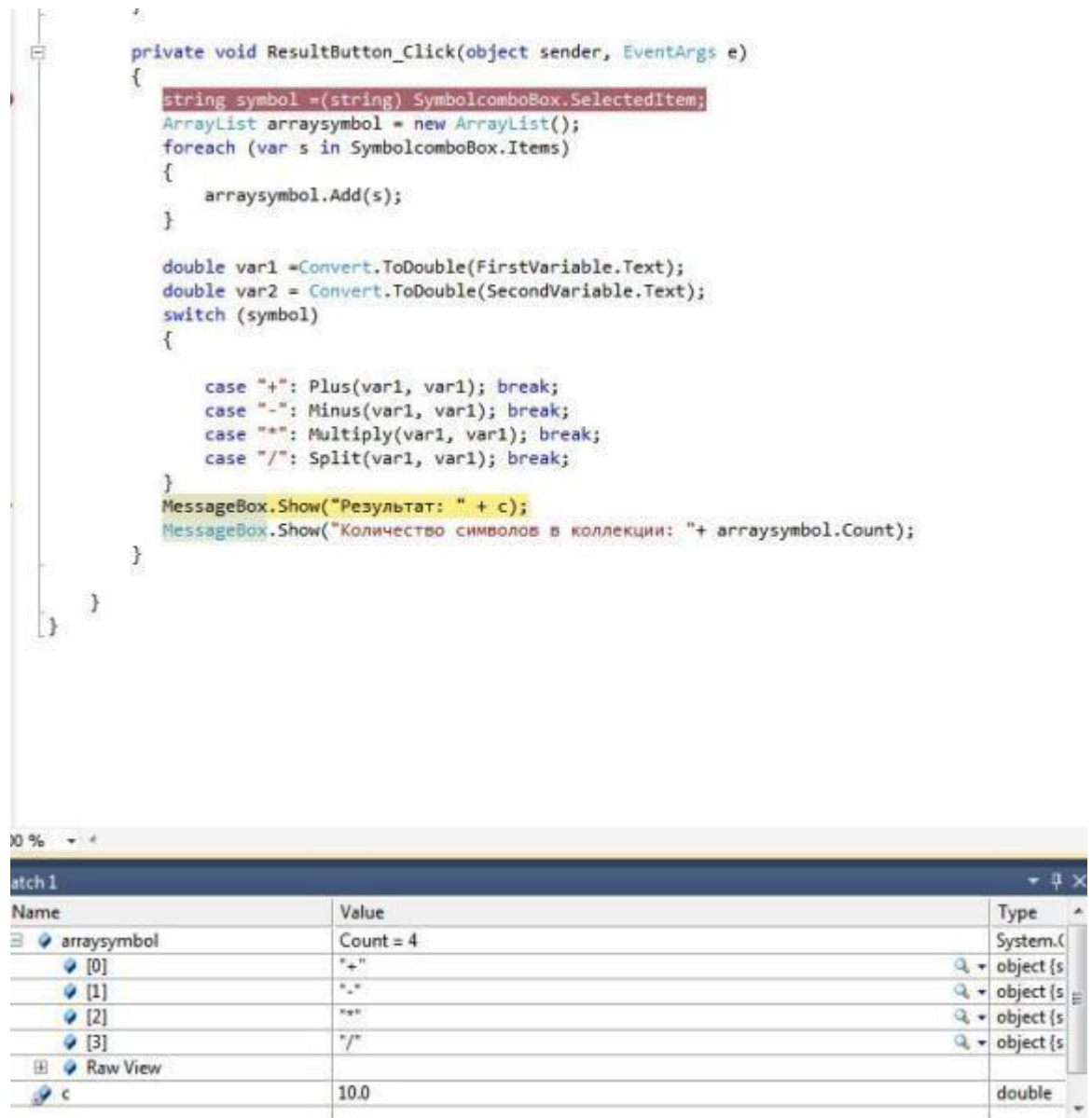
Если нужно просматривать состояние всех переменных во время отладки, используется окно **Autos** (переменные будут появляться в окне автоматически, в зависимости от шага отладчика):



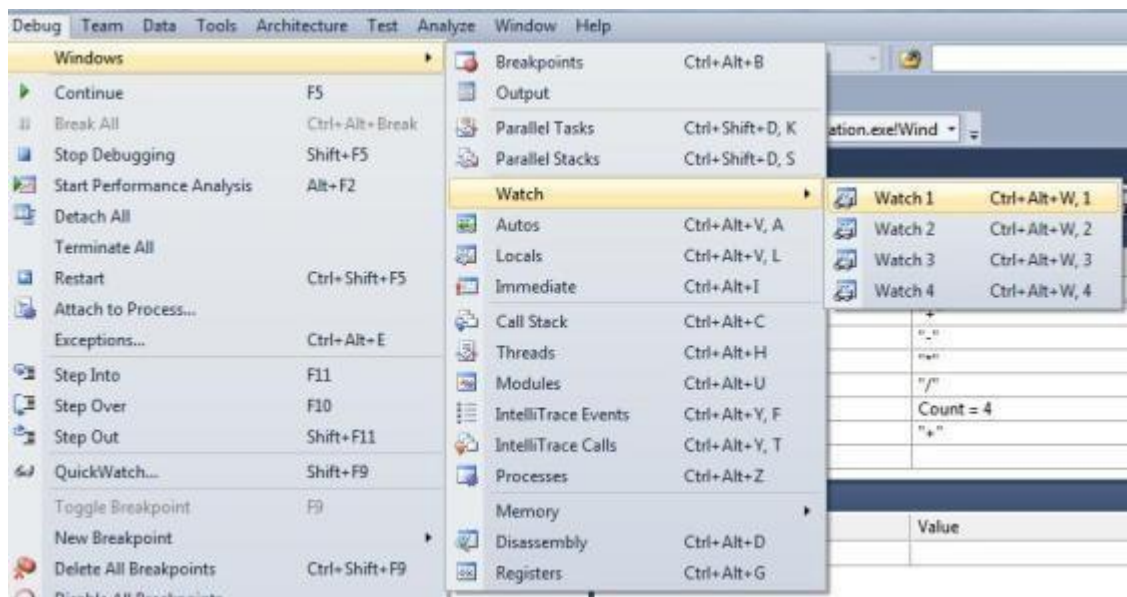
Используйте кнопку **"F10"** для пошаговой отладки приложения. В процессе пошаговой отладки, курсор отладчика будет заходить в те методы, которые вызываются в методе **ResulButton_Click()**, в нашем случае это метод **Plus()** (т.к была выбрана операция сложения - "+"):



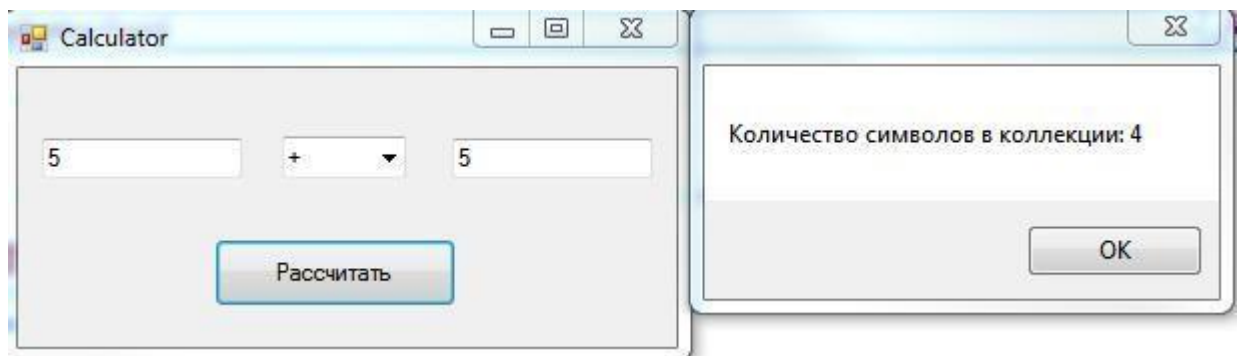
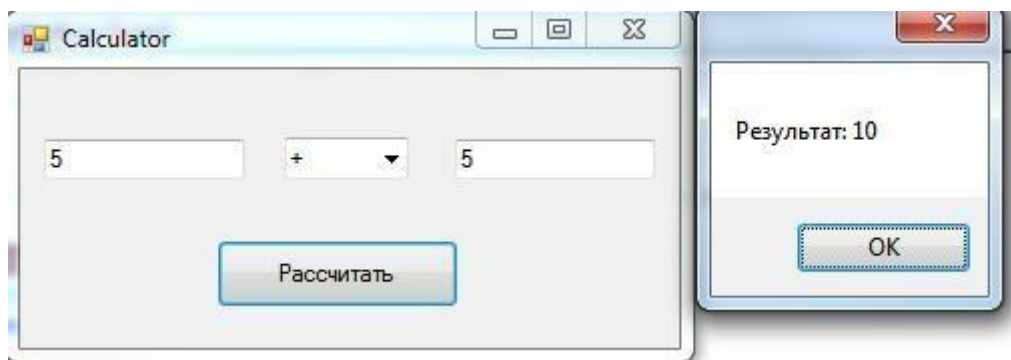
В процессе отладки приложения, значения переменных, в списке **Watch**, будут изменяться в зависимости от шага:



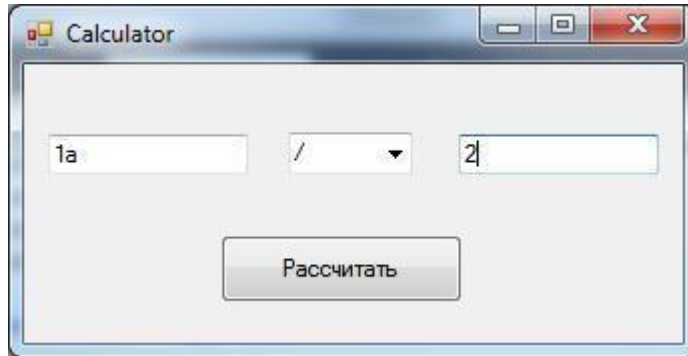
Всего разработчику доступны четыре списка **Watch**, которые вызываются из меню **Debug** или с помощью горячих клавиш **"Ctrl+Alt+W,1"**:



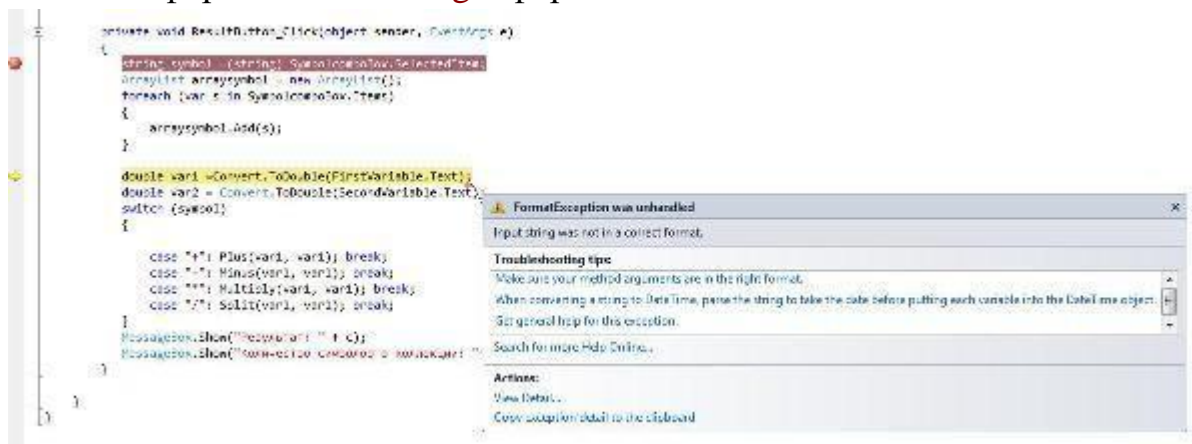
После завершения отладки (и если не возникло не каких ошибок) программа выдаст результаты:



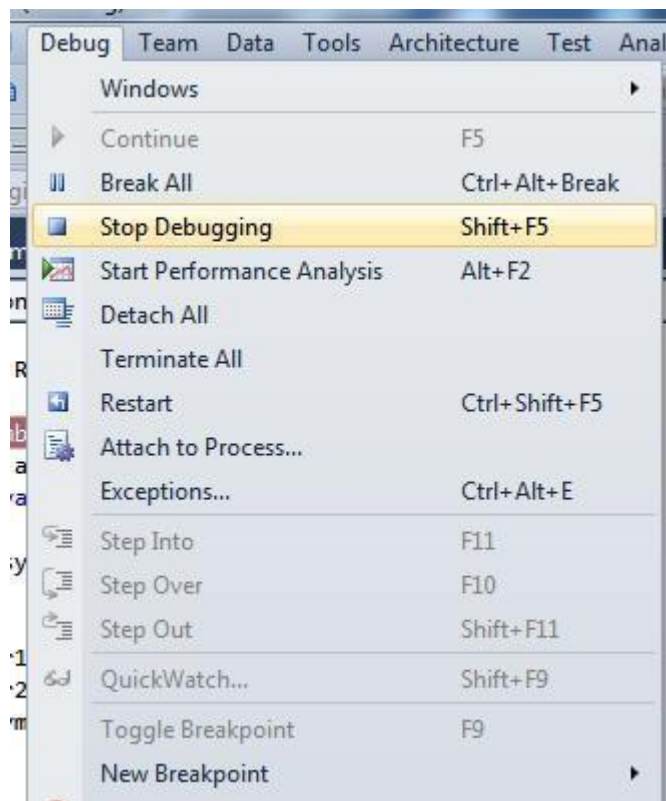
Повторно запустим отладку приложения и намеренно введем значения, вызывающие исключение. В нашем случае это "1a" и "2":



Если в программе не обрабатываются исключения (блок try, catch), отладчик выдаст ошибку, на строке, где возникает исключение. В нашем случае исключение, связанное с преобразованием формата типа **string** в формат **double**:



Для того что бы остановить отладку используйте пункт меню **"Stop Debugging"** меню **Debug** или используя сочетание клавиш **"Shift+F5"**:



Задание 2. Реализовать программу на формах, согласно индивидуальному варианту, со следующими дополнениями:

- 1) заполнение букв -не выполнено
- 2) заполнение отрицательных элементов-выполнено
- 3) – ручной ввод значений – выполнено
- 4) Заполнение автоматическое - выполнено

Индивидуальные задания:

Вариант 1. В одномерном динамическом массиве, состоящем из p элементов, вычислить сумму элементов массива, расположенных после минимального элемента.

Вариант 2. В одномерном динамическом массиве, состоящем из p элементов, вычислить сумму элементов массива, расположенных между минимальным и максимальным элементами.

Вариант 3. В одномерном динамическом массиве, состоящем из p элементов, вычислить среднеарифметическое значение элементов массива.

Вариант 4. В одномерном динамическом массиве, состоящем из p элементов, вычислить среднеквадратическое значение элементов массива.

Вариант 5. В одномерном динамическом массиве, состоящем из n элементов, вычислить среднегеометрическое значение ненулевых элементов массива.

Вариант 6. В одномерном динамическом массиве, состоящем из n элементов, вычислить среднегармоническое значение положительных элементов массива.

Вариант 7. В одномерном динамическом массиве, состоящем из n элементов, поменять местами максимальный и минимальный элементы.

Вариант 8. В одномерном динамическом массиве, состоящем из n элементов, вычислить сумму элементов массива, расположенных между первым и последним отрицательными элементами.

Вариант 9. В одномерном динамическом массиве, состоящем из n элементов, вычислить сумму элементов массива, расположенных до последнего положительного элемента.

Вариант 10. В одномерном динамическом массиве, состоящем из n элементов, вычислить сумму элементов массива, расположенных между первым и последним нулевыми элементами.

Вариант 11. В одномерном динамическом массиве, состоящем из n элементов, вычислить сумму модулей элементов массива, расположенных после первого элемента, равного нулю.

Вариант 12. В одномерном динамическом массиве, состоящем из n элементов, вычислить сумму положительных элементов массива, расположенных до максимального элемента.

Вариант 13. В одномерном динамическом массиве, состоящем из n элементов, вычислить сумму элементов массива, расположенных до минимального элемента.

Вариант 14. В одномерном динамическом массиве, состоящем из n элементов, вычислить сумму элементов массива, расположенных между первым и последним положительными элементами.

Вариант 15. В одномерном динамическом массиве, состоящем из n элементов, вычислить среднее значение элементов, расположенных в массиве между первым последним нулевыми элементами.

Вариант 16. В одномерном динамическом массиве, состоящем из n элементов, вычислить произведение элементов массива, расположенных между первым и вторым нулевыми элементами.

Вариант 17. В одномерном динамическом массиве, состоящем из n элементов, определить номер минимального и максимального элементов массива.

Вариант 18. В одномерном динамическом массиве, состоящем из n элементов, определить сумму модулей элементов массива, расположенных после первого отрицательного элемента.

Вариант 19. В одномерном динамическом массиве, состоящем из n элементов, определить количество элементов массива, больших C .

Вариант 20. В одномерном динамическом массиве, состоящем из n элементов, определить количество элементов массива, меньших C .

Вариант 21. В одномерном динамическом массиве, состоящем из n элементов, вычислить сумму отрицательных элементов массива.

Вариант 22. В одномерном динамическом массиве, состоящем из n элементов, вычислить сумму положительных элементов массива.

Вариант 23. В одномерном динамическом массиве, состоящем из n элементов, вычислить произведение элементов массива с четными номерами.

Вариант 24. В одномерном динамическом массиве, состоящем из n элементов, вычислить сумму элементов массива, расположенных после минимального элемента.

Вариант 25. В одномерном динамическом массиве, состоящем из n элементов, вычислить сумму элементов массива, расположенных до максимального элемента.

Вариант 26. В одномерном динамическом массиве, состоящем из n элементов, выполнить поиск элемента по заданному ключу последовательным методом поиска.

Вариант 27. В одномерном динамическом массиве, состоящем из n элементов, выполнить поиск элемента по заданному ключу бинарным методом поиска.

Вариант 28. В одномерном динамическом массиве, состоящем из n элементов, выполнить поиск элемента по заданному ключу интерполяционным методом поиска.

Вариант 29. В одномерном динамическом массиве, состоящем из n элементов, определить количество и индексы элементов массива, больших C .

Вариант 30. В одномерном динамическом массиве, состоящем из n элементов, определить количество и индексы элементов массива, меньших C .

Задание 3. Выполнить отладку.

Критерии оценивания:

Оценка 5 «отлично» работа выполнена полностью и правильно, сделаны правильные выводы

Оценка 4 «хорошо» работа выполнена правильно с учетом 1-2 несущественных ошибок, исправленных самостоятельно по требованию преподавателя

Оценка 3 «удовлетворительно» работа выполнена правильно не менее чем на половину или допущены 3-4 существенные ошибки

Оценка 2 «неудовлетворительно» допущены 5 и более существенные ошибки в ходе работы, которые обучающийся не может исправить даже по требованию преподавателя

Практическое занятие № 9

Тема раздела : Инструментарий тестирования и анализа качества программных средств

Цель работы: отладка многопоточного приложения

Материально-техническое и комплексно-методическое обеспечение: Для проведения практической работы используется следующее обеспечение: персональный компьютер, подключённый к Интернету,

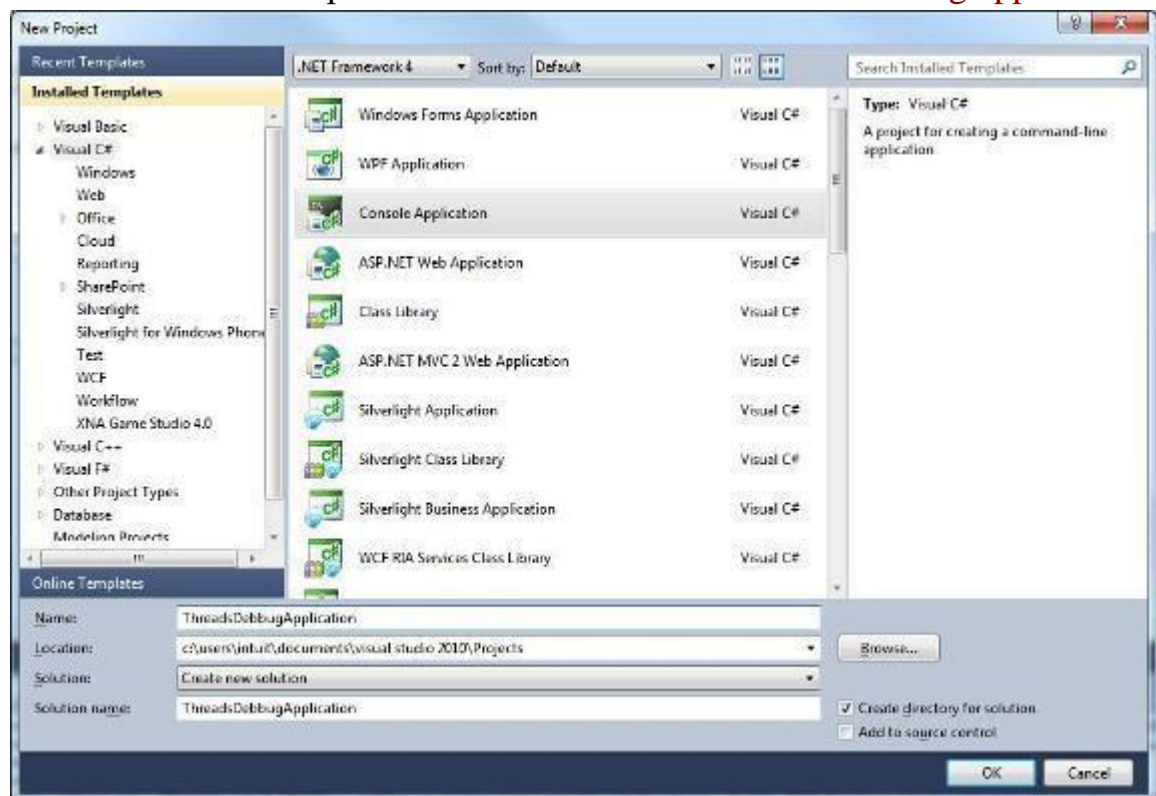
Время выполнения: 180 минут

Форма отчетности по занятию: файл с выполненной работой

Задание 1. Изучить теоретические сведения

Теоретические сведения

Создадим новое консольное приложение и назовем его **"ThreadsDebugApplication"**:



Добавим в созданное приложение следующий код:

1. using System;
2. using System.Collections.Generic;
3. using System.Linq;
4. using System.Text;
5. using System.Threading;
6. namespace ThreadsDebugApplication
7. {
8. class Program

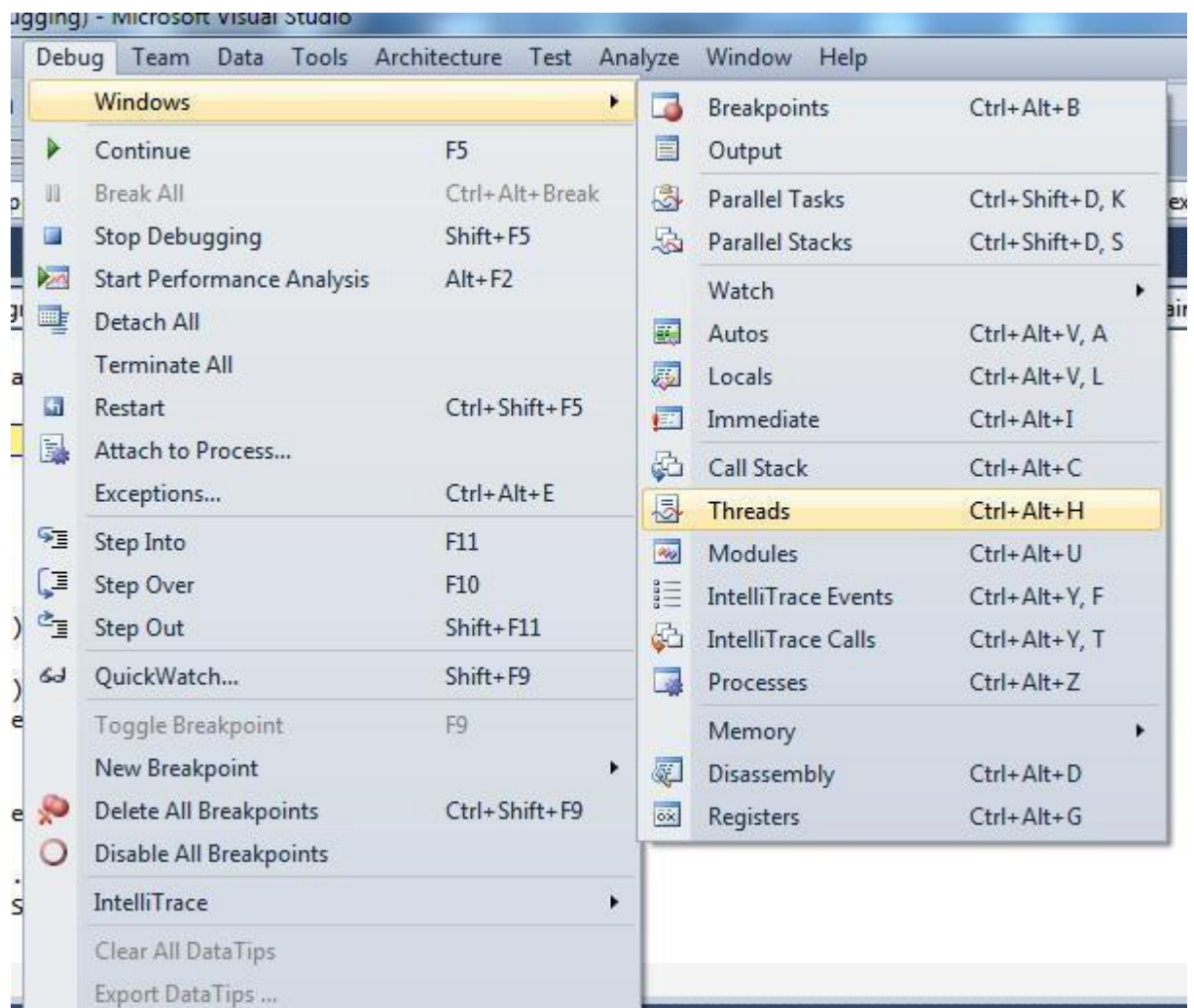
```
9.  {
10.
11.  static void Main(string[] args)
12.  {
13.      Thread t1 = new Thread(new ThreadStart>Hello));
14.      t1.Name = "Thread 1";
15.      Thread t2 = new Thread(new ThreadStart(World));
16.      t2.Name = "Thread 2";
17.      t1.Start();
18.      t2.Start();
19.      Console.ReadLine();
20.  }
21.  static void Hello()
22.  {
23.      Console.Write("Hello ");
24.      Thread.Sleep(10000);
25.  }
26.  static void World()
27.  {
28.      Console.WriteLine("world!");
29.      Thread.Sleep(10000);
30.  }
31.
32. }
```

Установим точку останова (**Breakpoint**) в самое начало метода **Main()** и запустим отладку с помощью клавиши **F5**:

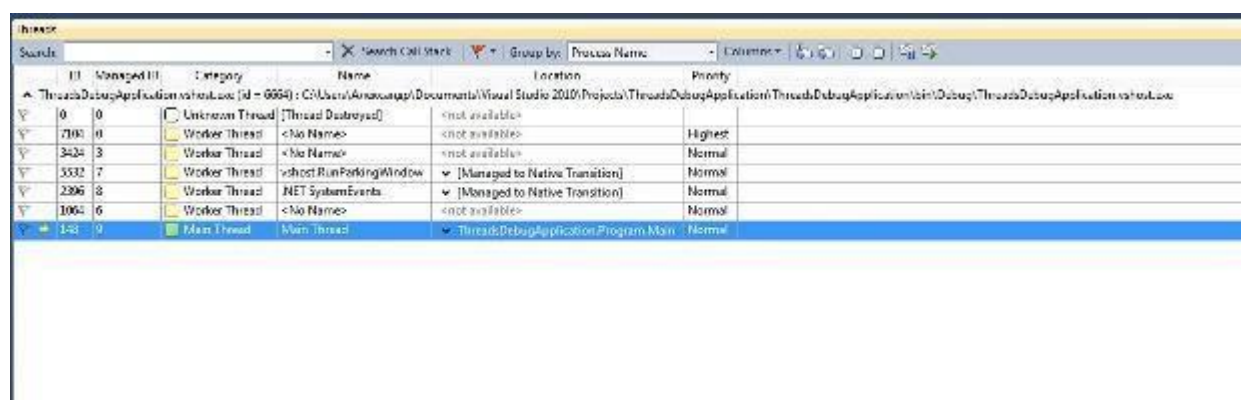


Рис. 21.2.

Откройте окно отладки потоков из меню **"Debug"** пункт меню **"Threads"** или с помощью сочетания клавиш **"Ctrl+Alt+H"**:



В появившемся окошке будут отображаться все текущие рабочие потоки:



С помощью клавиши "F10" доберитесь до строчки с кодом `Console.ReadLine()`, как показано на рисунке:

```

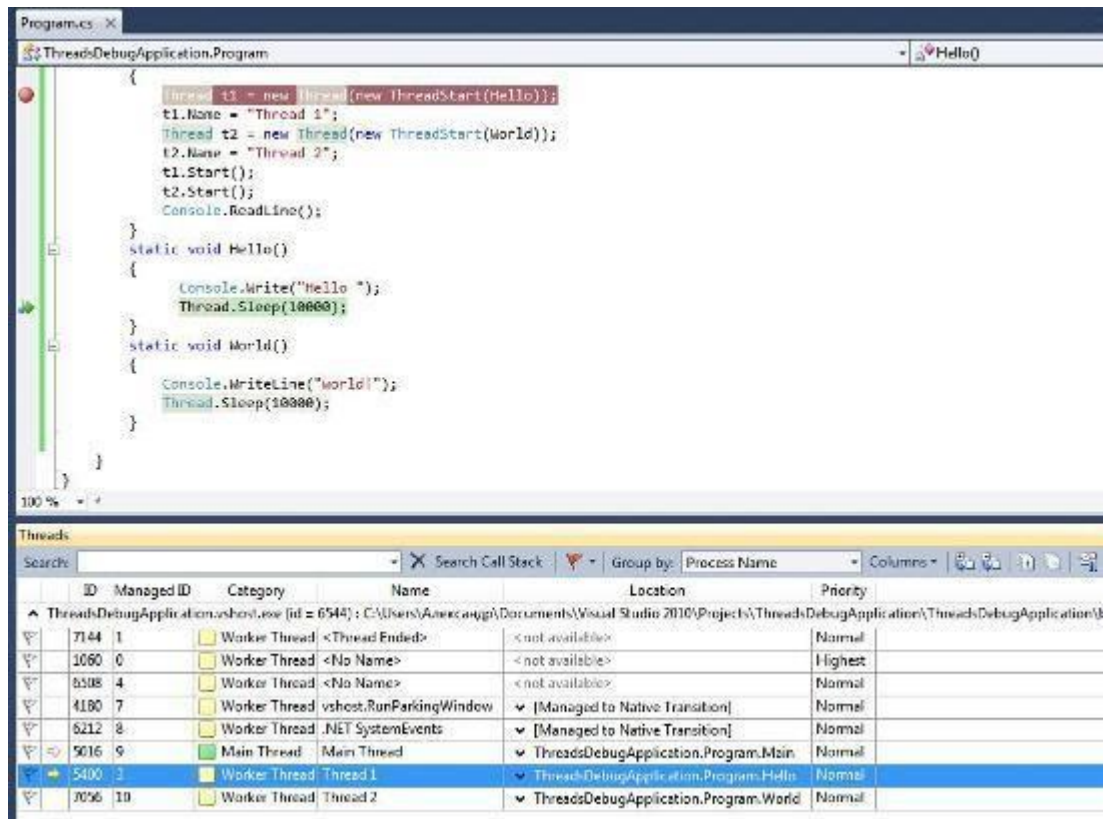
Program.cs
ThreadsDebugApplication.Program
namespace ThreadsDebugApplication
{
    class Program
    {
        static void Main(string[] args)
        {
            Thread t1 = new Thread(new ThreadStart>Hello));
            t1.Name = "Thread 1";
            Thread t2 = new Thread(new ThreadStart(World));
            t2.Name = "Thread 2";
            t1.Start();
            t2.Start();
            Console.ReadLine();
        }
        static void Hello()
        {
            Console.WriteLine("Hello ");
            Thread.Sleep(10000);
        }
    }
}

```

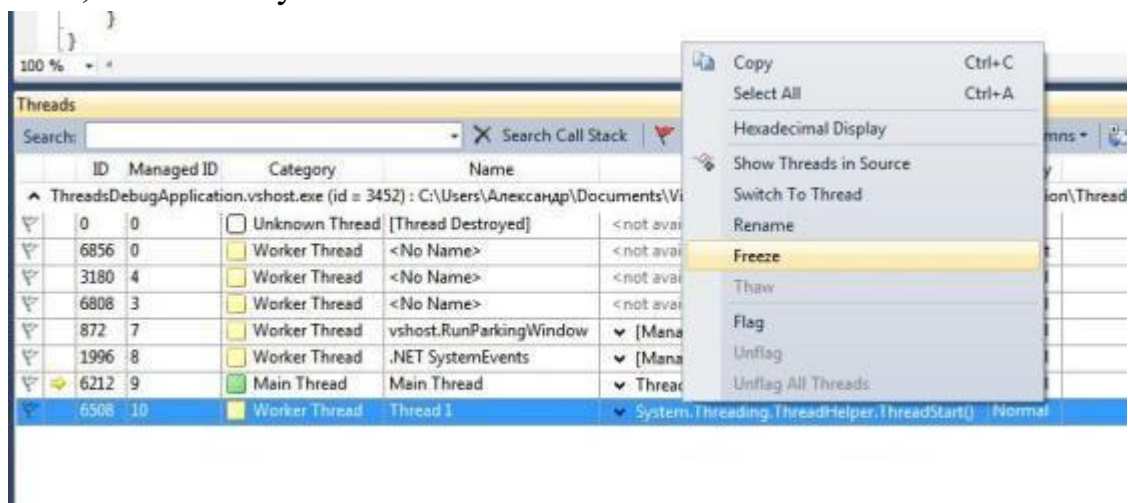
В окне отладки потоков можно увидеть два созданных потока:

Threads						
Search: [X] Search Call Stack [v] Group by: Process Name Columns [v]						
	ID	Managed ID	Category	Name	Location	Priority
ThreadsDebugApplication.vshost.exe (id = 6544) : C:\Users\Aneecanp\Documents\Visual Studio 2010\Projects\ThreadsDebugApplication\ThreadsDebugApplication\bin\Debug\1						
7144	1		Worker Thread	<Thread Ended>	<not available>	Normal
1060	0		Worker Thread	<No Name>	<not available>	Highest
6306	4		Worker Thread	<No Name>	<not available>	Normal
4190	7		Worker Thread	vshost.RunParkingWindow	[Managed to Native Transition]	Normal
6712	8		Worker Thread	.NET SystemEvents	[Managed to Native Transition]	Normal
5016	9		Main Thread	Main Thread	ThreadsDebugApplication.Program.Main	Normal
5400	3		Worker Thread	Thread 1	ThreadsDebugApplication.Program.Hello	Normal
7056	10		Worker Thread	Thread 2	ThreadsDebugApplication.Program.World	Normal

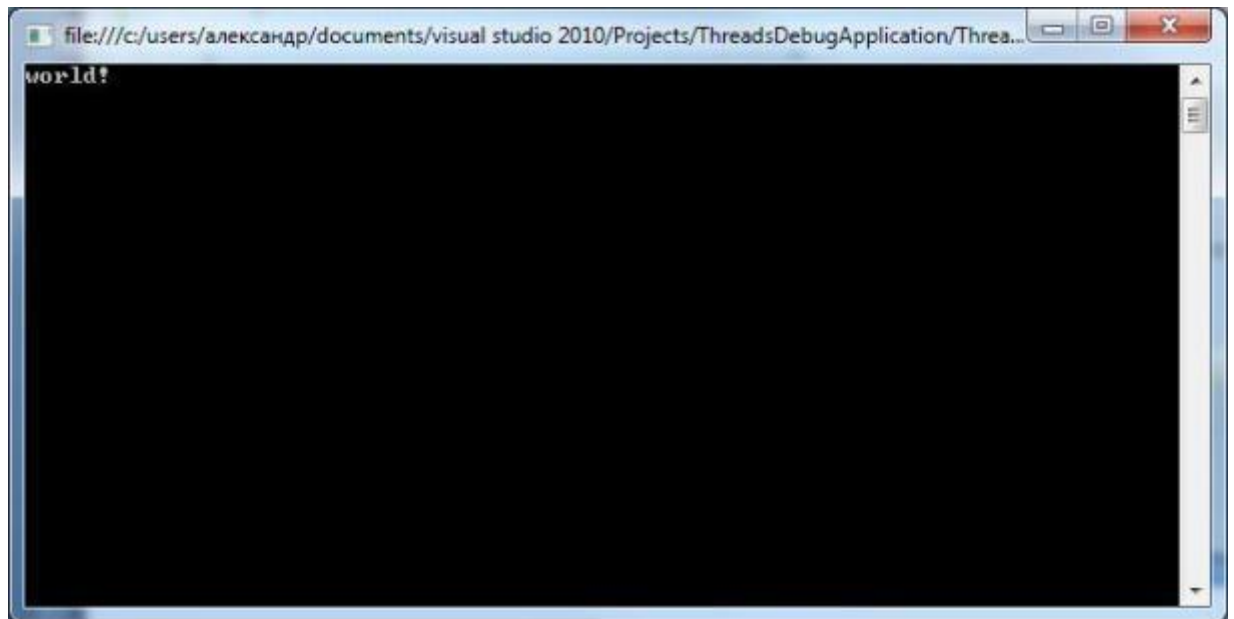
Чтобы перейти на место выполнения конкретного потока, дважды щелкните левой кнопкой мыши на нужном потоке:



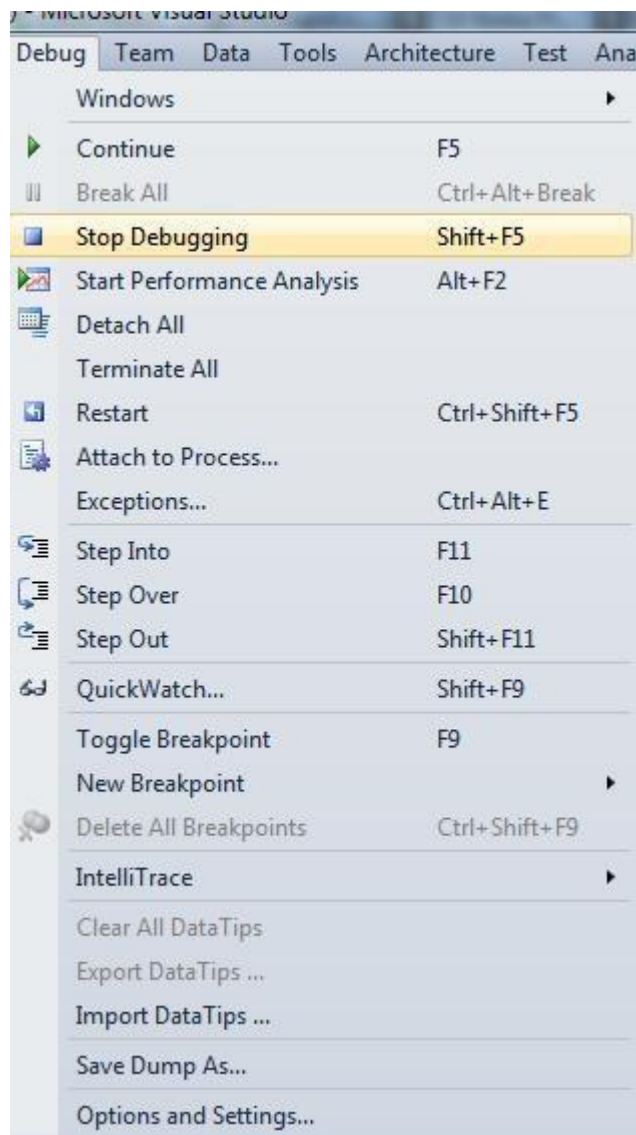
Для того что бы заморозить выполнение потока, щелкните на нужном потоке выберите пункт **"Freeze"**, в нашем случае это поток с именем **"Thread1"**:



После выполнения программы выведется результат без "замороженного" потока:



Для завершения процесса отладки используется пункт из меню "Debug" - "Stop Debugging" или сочетание клавиш "Shift+F5":



Задание 2. Перепишите программу на 4 треда. Выполните отладку.

Критерии оценивания:

Оценка 5 «отлично» работа выполнена полностью и правильно, сделаны правильные выводы

Оценка 4 «хорошо» работа выполнена правильно с учетом 1-2 несущественных ошибок, исправленных самостоятельно по требованию преподавателя

Оценка 3 «удовлетворительно» работа выполнена правильно не менее чем на половину или допущены 3-4 существенные ошибки

Оценка 2 «неудовлетворительно» допущены 5 и более существенные ошибки в ходе работы, которые обучающийся не может исправить даже по требованию преподавателя

Практическое занятие № 10

Тема раздела : Инструментарий тестирования и анализа качества программных средств

Цель работы: инспекция кода

Материально-техническое и комплексно-методическое обеспечение: Для проведения практической работы используется следующее обеспечение: персональный компьютер, подключённый к Интернету,

Время выполнения: 180 минут

Форма отчетности по занятию: файл с выполненной работой

Задание 1. Изучить теоретические сведения

Теоретические сведения

Интеграция программ выполняется посредством *поэтапного* или *инкрементного* подхода.

Поэтапная интеграция состоит из этапов, перечисленных ниже:

«Модульная разработка»: проектирование, кодирование, тестирование и отладка каждого класса.

«Системная интеграция»: объединение классов в одну огромную систему.

«Системная дезинтеграция»: тестирование и отладка всей системы.

Проблема поэтапной интеграции в том, что, когда классы в системе впервые соединяются вместе, неизбежно возникают новые проблемы и их причины могут быть в чем угодно. Поскольку у вас масса классов, которые никогда раньше не работали вместе, виновником может быть плохо протестированный класс, ошибка в интерфейсе между двумя классами или ошибка, вызванная взаимодействием двух классов. Все классы находятся под подозрением.

Неопределенность местонахождения любой из проблем сочетается с тем фактом, что все эти проблемы вдруг проявляют себя одновременно. Это заставляет вас иметь дело не только с проблемами, вызванными взаимодействием классов, но и другими ошибками, которые трудно диагностировать, так как они взаимодействуют.

Поэтому поэтапную интеграцию называют еще «интеграцией большого взрыва»

Поэтапную интеграцию нельзя начинать до начала последних стадий проекта, когда будут разработаны и протестированы все классы. Когда классы, наконец, будут объединены и проявится большое число ошибок, программисты тут же ударятся в паническую отладку вместо методического определения и исправления ошибок.

Для небольших программ — нет, а для крошечных — поэтапная интеграция может быть наилучшим подходом. Если программа состоит из двух-трех классов, поэтапная интеграция может сэкономить ваше время, если вам повезет. Но в большинстве случаев инкрементный подход будет лучше.

При **инкрементной интеграции** вы пишете и тестируете маленькие участки программы, а затем комбинируете эти кусочки друг с другом по одному. При таком подходе — по одному элементу за раз — вы выполняете перечисленные далее действия:

1. Разрабатываете небольшую, функциональную часть системы. Это может быть наименьшая функциональная часть, самая сложная часть, основная часть или их комбинация. Тщательно тестируете и отлаживаете ее. Она послужит скелетом, на котором будут наращиваться мускулы, нервы и кожа, составляющие остальные части системы.

2. Проектируете, кодируете, тестируете и отлаживаете класс.

3. Прикрепляете новый класс к скелету. Тестируете и отлаживаете соединение скелета и нового класса. Убеждаетесь, что эта комбинация работает, прежде чем переходить к добавлению нового класса. Если дело сделано, повторяете процесс, начиная с п. 2.

Инкрементный подход имеет массу преимуществ перед традиционным поэтапным подходом независимо от того, какую инкрементную стратегию вы используете:

Ошибки можно легко обнаружить Когда во время инкрементной интеграции возникает новая проблема, то очевидно, что к этому причастен новый класс. Либо его интерфейс с остальной частью программы неправилен, либо его взаимодействие с ранее интегрированными классами приводит к ошибке. В любом случае вы точно знаете, где искать проблему.

В таком проекте система раньше становится работоспособной Когда код интегрирован и способен выполняться, даже если система еще не пригодна к использованию, это выглядит так, будто это скоро произойдет. При инкрементной интеграции программисты раньше видят результаты своей работы, поэтому их моральное состояние лучше, чем в том случае, когда они подозревают, что их проект может никогда не сделать первый вдох.

Вы получаете улучшенный мониторинг состояния При частой интеграции реализованная и нереализованная функциональность видна с первого взгляда. Менеджеры будут иметь лучшее представление о состоянии проекта, видя, что 50% системы уже работает, а не слыша, что кодирование «завершено на 99%».

Вы улучшите отношения с заказчиком Если частая интеграция влияет на моральное состояние разработчиков, то она также оказывает влияние и на моральное состояние заказчика. Клиенты любят видеть признаки прогресса, а инкрементная интеграция предоставляет им такую возможность достаточно часто.

Системные модули тестируются гораздо полнее Интеграция начинается на ранних стадиях проекта. Вы интегрируете каждый класс по мере его готовности, а не ожидая одного внушительного мероприятия по интеграции в конце разработки. Программист тестирует классы в обоих случаях, но в качестве элемента общей системы они используются гораздо чаще при инкрементной, чем при поэтапной интеграции.

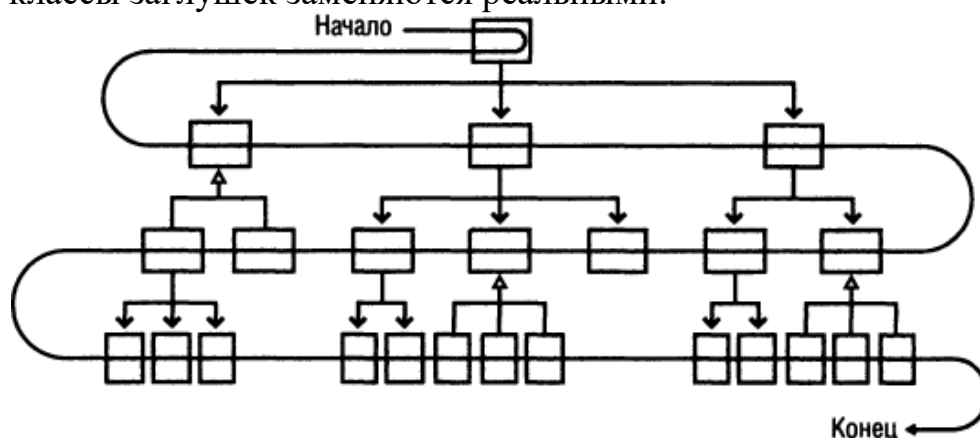
Вы можете создать систему за более короткое время Если интеграция тщательно спланирована, вы можете проектировать одну часть системы в то время, когда другая часть уже кодируется. Это не уменьшает общее число человеко-часов, требуемых для полного проектирования и кодирования, но позволяет выполнять часть работ параллельно, что является преимуществом в тех случаях, когда время имеет критическое значение.

При поэтапной интеграции вам не нужно планировать порядок создания компонентов проекта. Все компоненты интегрируются одновременно, поэтому вы можете разрабатывать их в любом порядке — главное, чтобы они все были готовы к часу X.

При инкрементной интеграции вы должны планировать более аккуратно. Большинство систем требует интеграции некоторых компонентов перед интеграцией других. Так что планирование интеграции влияет на планирование конструирования — порядок, в котором конструируются компоненты, должен обеспечивать порядок, в котором они будут интегрироваться.

Нисходящая интеграция

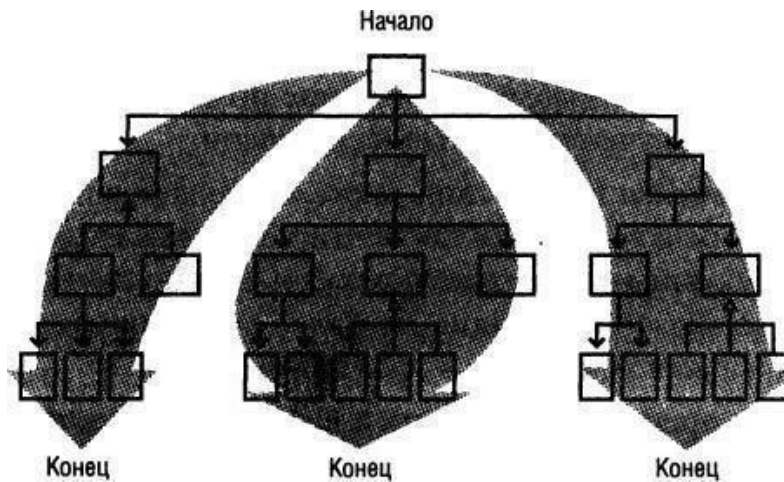
При нисходящей интеграции класс на вершине иерархии пишется и интегрируется первым. Вершина иерархии — это главное окно, управляющий цикл приложения, объект, содержащий метод `main()` в программе на Java, функция `WinMain()` в программировании для Microsoft Windows или аналогичные. Для работы этого верхнего класса пишутся заглушки. Затем, по мере интеграции классов сверху вниз, классы заглушек заменяются реальными.



При нисходящей интеграции вы создаете те классы, которые находятся на вершине иерархии, первыми, а те, что внизу, — последними.

Хорошей альтернативой нисходящей интеграции в чистом виде может стать подход вертикальным секционированием.

Рис. 4 Вертикальное секционирование



При этом систему реализуют сверху вниз по частям, возможно, по очереди выделяя функциональные области и переходя от одной к другой.

Восходящая интеграция

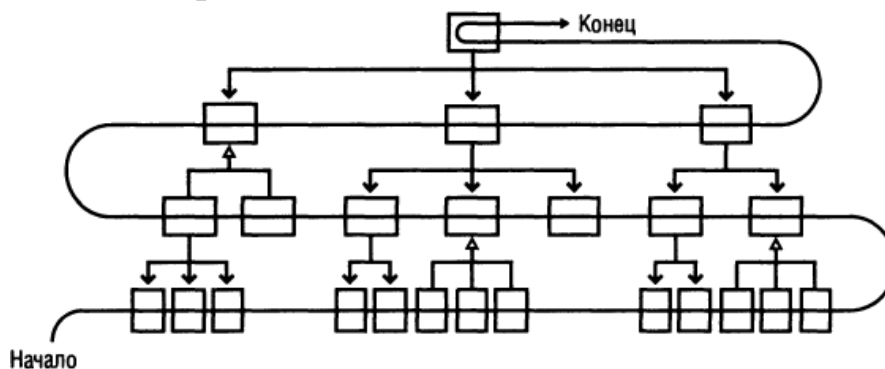
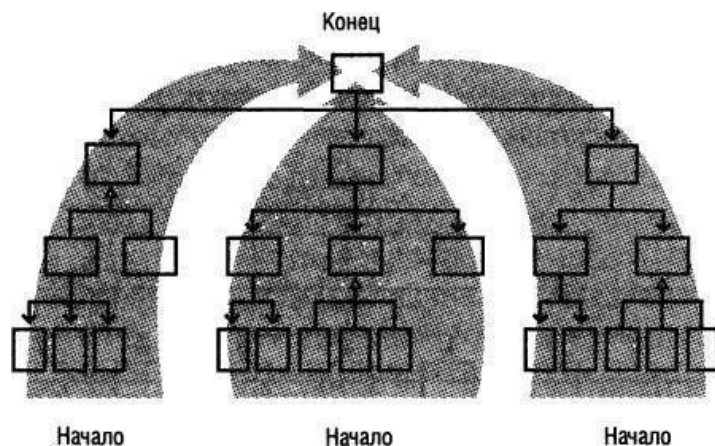


Рис.5 Восходящая интеграция

При восходящей интеграции вы пишете и интегрируете сначала классы, находящиеся в низу иерархии. Добавление низкоуровневых классов по одному, а не всех одновременно — вот что делает восходящую интеграцию инкрементной стратегией. Сначала вы пишете тестовые драйверы для выполнения низкоуровневых классов, а затем добавляете эти классы к тестовым драйверам, пристраивая их по мере готовности. Добавляя класс более высокого уровня, вы заменяете классы драйверов реальными.



Как и нисходящую, восходящую интеграцию в чистом виде используют редко — вместо нее можно применять гибридный подход, реализующий секционную интеграцию.

Сэндвич-интеграция

Проблемы с нисходящей и восходящей интеграциями в чистом виде привели к тому, что некоторые эксперты стали рекомендовать сэндвич-подход.

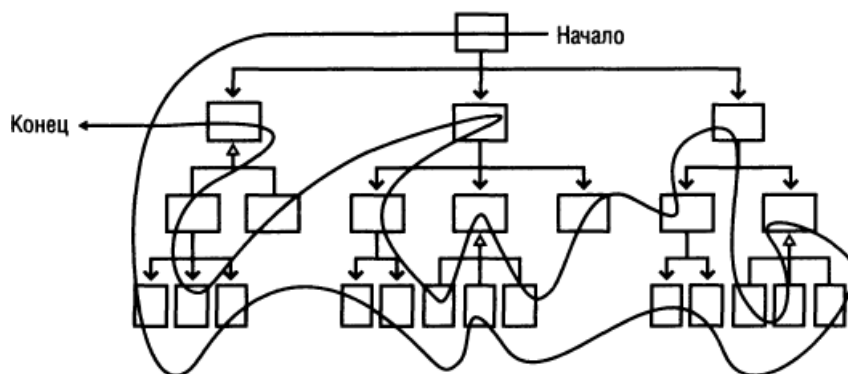


Рис. 7 Сэндвич-интеграция

Сначала вы объединяете высокоуровневые классы бизнес-объектов на вершине иерархии. Затем добавляете классы, взаимодействующие с аппаратной частью, и широко используемые вспомогательные классы в низу иерархии.

Напоследок вы оставляете классы среднего уровня.

Риск-ориентированная интеграция

Риск-ориентированную интеграцию, которую также называют «интеграцией, начиная с самых сложных частей» (hard part first integration), похожа на сэндвич-интеграцию тем, что пытается избежать проблем, присущих нисходящей или восходящей интеграциям в чистом виде. Кроме того, в ней также есть тенденция к объединению классов верхнего и нижнего уровней в первую очередь, оставляя классы среднего уровня напоследок. Однако суть в другом.

При риск-ориентированной интеграции вы определяете степень риска, связанную с каждым классом. Вы решаете, какие части системы будут самыми трудными, и реализуете их первыми.

Функционально-ориентированная интеграция

Еще один поход — интеграция одной функции в каждый момент времени. Под

«функцией» понимается не нечто расплывчатое, а какое-нибудь поддающееся определению свойство системы, в которой выполняется интеграция.

Когда интегрируемая функция превышает по размерам отдельный класс, то «единица приращения» инкрементной интеграции становится больше отдельного класса. Это немного снижает преимущество инкрементного подхода в том плане, что уменьшает вашу уверенность об источнике новых ошибок. Однако если вы тщательно тестировали классы, реализующие эту функцию, перед интеграцией, то это лишь небольшой недостаток. Вы можете использовать стратегии инкрементной интеграции рекурсивно, сформировав сначала из небольших кусков отдельные свойства, а затем инкрементно объединив их в систему.

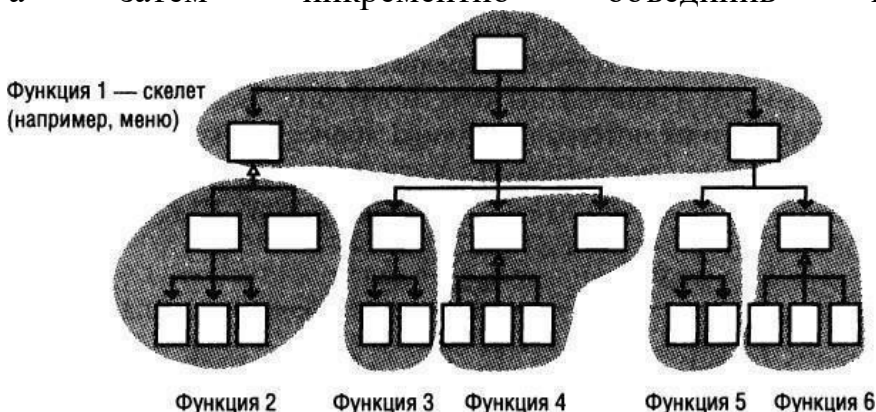
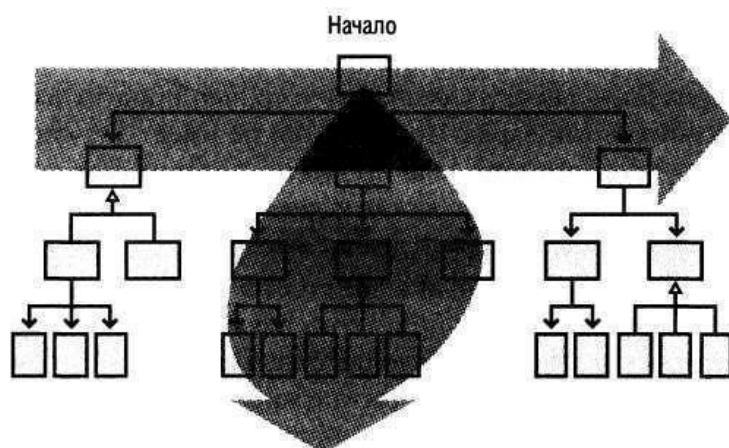


Рис. 8 Функционально-ориентированная интеграция

Обычно процесс начинается с формирования скелета, поскольку он способен поддерживать остальную функциональность. В интерактивной системе такой изначальной опцией может стать система интерактивного меню. Вы можете прикреплять остальную функциональность к той опции, которую интегрировали первой.

Т-образная интеграция

Последний подход, который часто упоминается в связи с проблемами нисходящей и восходящей методик, называется «Т-образной интеграцией». При таком подходе выбирается некоторый вертикальный слой, который разрабатывается и интегрируется раньше других. Этот слой должен проходить сквозь всю систему от начала до конца и позволять выявлять основные проблемы в допущениях, сделанных при проектировании системы.



Реализовав этот вертикальный участок (и устранив все связанные с этим проблемы), можно разрабатывать основную канву системы (например, системное меню для настольного приложения). Этот подход часто комбинируют с риск-ориентированной и функционально-ориентированной интеграциями.

Задание 2. Оформить внешнюю спецификацию.

Задание 3. Составить в виде блок-схемы алгоритм решения задачи.

Задание 4. Реализовать программу согласно индивидуальному заданию. Закодировать сообщение, используя код Полибия

ВНИМАНИЕ! ДОЛЖНО ДЫТЬ ДВА МОДУЛЯ: НА ШИФРОВАНИЕ И НА ДЕШИФРОВКУ

Древнегреческий историк и полководец Полибий придумал сложный способ шифрования, названный в честь его квадрат Полибия. «Квадрат Полибия» представляет собой квадрат 6х6 (5х5), столбцы и строки которого нумеруются цифрами от 1 до 6 (5). Все буквы алфавита вписывались в квадрат по одной на клетку. (можно в одну клетку поместите буквы е-э, и-й, ж-з, р-с, ф-х, ш-щ). Буквы расположены в алфавитном порядке.

В результате каждой букве соответствует пара чисел, и зашифрованное сообщение превращается в последовательность пар чисел.

Расшифровывается путем нахождения буквы стоящей на пересечении строки и столбца.

Суть в том, что каждой букве предназначена своя пара цифр одна по горизонтальной линии, а другая по вертикальной. Данный вид кодирования изначально применялся для греческого алфавита, но затем был распространен на другие языки.

	1	2	3	4	5	6
1	А	Б	В	Г	Д	Е
2	Ё	Ж	З	И	Й	К
3	Л	М	Н	О	П	Р
4	С	Т	У	Ф	Х	Ц
5	Ч	Ш	Щ	Ъ	Ы	Ь
6	Э	Ю	Я	,	.	-

Индивидуальные задания

№ варианта	Задание
1	Если бы да кабы, да во рту росли грибы.
2	Конь о четырех ногах, а спотыкается.
3	Дают — бери, а бьют — беги.
4	Кто рано встает, тому Бог подает.
5	Взялся за гуж — не говори, что не дюж.
6	На Бога надейся, а сам не плошай.
7	Поспесишь — людей насмешишь.
8	Слово — не воробей, вылетит — не поймаешь.
9	Обещать — дело господское, исполнять — холопское.
10	Дурная голова ногам покоя не дает.
11	Мягко стелет, да жестко спать.
12	Бей своих, чтобы чужие боялись.
13	Не плюй в колодец — пригодится напиться.
14	Куй железо, пока горячо.
15	Хлеба ни куска, и стол — доска
16	Хлеб наш насущный: хоть черный, да вкусный
17	Хлеб ногами топтать — народу голодать
18	Без ума проколотишься, а без хлеба не проживешь

19	Блюди хлеб на обед, а слово – на ответ
20	Весною сутки мочит, а час сушит.
21	Вешний ледок, что чужой избы порог.
22	Вешняя пора - поел да и со двора.
23	Кто спит весною - плачет зимою.
24	Прилетела бы чайка, а весна будет.
25	Матушка – весна всем красна.
26	Солнце светит, солнце сияет – вся природа воскресает.
27	Готовь сани с весны, а колёса с осени.
28	Летний день на вес золота, он дороже зимней недели.
29	Лето - отрадная пора, с курами ложись, а с петухами на работу вставай.
30	Летом дома сидеть — зимой хлеба не иметь.

Задание 5. Выполнить отладку и тестирование модулей программы.

Задание 6. Выполнить инкрементную интеграцию модулей с использованием одного из подходов.

Критерии оценивания:

Оценка 5 «отлично» работа выполнена полностью и правильно, сделаны правильные выводы

Оценка 4 «хорошо» работа выполнена правильно с учетом 1-2 не существенных ошибок, исправленных самостоятельно по требованию преподавателя

Оценка 3 «удовлетворительно» работа выполнена правильно не менее чем на половину или допущены 3-4 существенные ошибки

Оценка 2 «неудовлетворительно» допущены 5 и более существенные ошибки в ходе работы, которые обучающийся не может исправить даже по требованию преподавателя

Практическое занятие № 11, № 12

Тема раздела : Инструментарий тестирования и анализа качества программных средств

Цель работы: Тестирование интеграции

Материально-техническое и комплексно-методическое обеспечение: Для проведения практической работы используется следующее обеспечение: персональный компьютер, подключённый к Интернету,

Время выполнения: 240 минут

Форма отчетности по занятию: файл с выполненной работой

Последовательность выполнения работы

1. Выполнить задание
2. Сохранить файл
3. Оформить отчет о выполненной работе

Задание № 1. Разработать приложение , на формах состоящее из трех модулей:

- 1) главный модуль, считывающий из текстового файла координаты точек на плоскости;
- 2) модуль, содержащий функции расчета расстояния между двумя точками;
- 3) модуль, содержащий функцию, определяющую треугольник с максимальной площадью.

Задание № 2. Описать этапы нисходящего проектирования разработанного приложения.

Задание № 3. Описать этапы восходящего проектирования разработанного приложения

Задание № 4. Протестировать приложение, составить отчет о дефектах

Критерии оценивания:

Оценка 5 «отлично» работа выполнена полностью и правильно, сделаны правильные выводы

Оценка 4 «хорошо» работа выполнена правильно с учетом 1-2 незначительных ошибок, исправленных самостоятельно по требованию преподавателя

Оценка 3 «удовлетворительно» работа выполнена правильно не менее чем на половину или допущены 3-4 существенные ошибки

Оценка 2 «неудовлетворительно» допущены 5 и более существенные ошибки в ходе работы, которые обучающийся не может исправить даже по требованию преподавателя

**Информационное обеспечение реализации программы
Перечень используемых учебных изданий, Интернет-ресурсов,
дополнительной литературы**

4.1 Печатные издания

4.2 Электронные издания (электронные ресурсы)

1 Гниденко, И. Г. Технология разработки программного обеспечения : учебное пособие для среднего профессионального образования / И. Г. Гниденко, Ф. Ф. Павлов, Д. Ю. Федоров. — Москва : Издательство Юрайт, 2021. — 235 с. — (Профессиональное образование). — ISBN 978-5-534-05047-9. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: <https://urait.ru/bcode/472502>

2. Подбельский, В. В. Программирование. Базовый курс C# : учебник для среднего профессионального образования / В. В. Подбельский. — Москва : Издательство Юрайт, 2020. — 369 с. — (Профессиональное образование). — ISBN 978-5-534-11467-6. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: <https://urait.ru/bcode/456697>

Интернет-ресурсы

1. Образовательный портал INTUIT.RU <http://www.intuit.ru>
2. METANIT.COM. Сайт о программировании <https://metanit.com>
3. Журнал «Моя профессиональная карьера» - Режим доступа: <https://www.elibrary.ru/item.asp?id=45669781>
4. Журнал «Апробация» - Режим доступа <https://www.elibrary.ru/item.asp?id=23216950>
5. Журнал «Новая наука и стратегия развития» - Режим доступа: <https://www.elibrary.ru/item.asp?id=27424248>
6. Журнал «Вестник сыктывкарского университета. Серия 1: Математика. Механика. Информатика <https://www.elibrary.ru/item.asp?id=32546230>

Электронно-библиотечные системы:

Доступ авторизованных пользователей через Интернет
ЭБС «IPRbooks», ООО «Ай Пи Эр Медиа»

ЭБС «Электронная библиотека технического вуза», ООО
«Политехресурс»
ЭБС «Лань», ООО «Издательство Лань»
ЭБС «elibrary», ООО «РУНЭБ»
ЭБС «ЮРАЙТ»
ЭБС «Book.ru»