

Направление подготовки 09.03.03 «Прикладная информатика»
Профиль «Прикладная информатика в топливно-энергетическом комплексе»
Методическое обеспечение дисциплины Б1.В.16 «Программная инженерия»



**Филиал федерального государственного бюджетного образовательного учреждения
высшего образования
«Национальный исследовательский университет «МЭИ»
в г. Смоленске**

**МЕТОДИЧЕСКИЕ МАТЕРИАЛЫ ДЛЯ ОБЕСПЕЧЕНИЯ
ОБРАЗОВАТЕЛЬНОГО ПРОЦЕССА**

Направление подготовки: **09.03.03 «Прикладная информатика»**

Профиль: **«Прикладная информатика в топливно-энергетическом комплексе»**

Уровень высшего образования: **бакалавриат**

Нормативный срок обучения: **4 года**


Форма обучения: **очная**

Год набора: **2023**

Смоленск


*Направление подготовки 09.03.03 «Прикладная информатика»
Профиль «Прикладная информатика в топливно-энергетическом комплексе»
Методическое обеспечение дисциплины Б1.В.16 «Программная инженерия»*

Методические материалы составил:

канд. техн. наук, доцент кафедры
информационных технологий в экономике и управлении  А.Ю. Пучков

«20» января 2023 г.

Заведующий кафедрой информационных технологий в экономике и управлении:


_____ подпись _____ д-р техн. наук, профессор М.И. Дли
Ф.И.О

«08» февраля 2023 г.

МЕТОДИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ЛАБОРАТОРНЫХ РАБОТ ПО ДИСЦИПЛИНЕ

Лабораторная работа № 1. Структурное моделирование существующих бизнес-процессов организации

Цель занятия: изучить на практике методологию структурного моделирования для описания бизнес-процессов организации.

Теоретические сведения. Методическое обеспечение структурного моделирования бизнес-процессов.

Модели бизнес-процессов позволяют получить целостную структурированную картину внешних и внутренних процессов организации.

Структурная модель – модель, включающая в себя упорядоченный по определенному принципу набор процессов (групп процессов) с указанием основных связей между ними. Основное назначение структурной модели заключается в отображении построения бизнес-процесса организации, а также раскрытии информации об основных группах процессов и их взаимосвязях. Структурная модель не показывает последовательность выполнения процессов во времени. Структурные модели можно использовать локально (не в системе процессов организации) для схематичного описания состава процессов, декомпозированных на следующий уровень.

Структурные модели процессов, как правило, применяются для:

- описания, анализа бизнес-модели организации и определения возможных направлений ее реорганизации;
- разработки системы бизнес-процессов организации по принципу «сверху вниз»;
- системного описания процессов, которые необходимо автоматизировать (например, в ERP-системе);
- описания состава процессов, декомпозированных на следующий уровень (несистемное, фрагментарное использование).

Наиболее популярной нотацией, используемой для моделирования бизнес-процессов, является нотация IDEF0. Правила и система обозначений IDEF0 позволяют отобразить структуру и функции любой системы.

Бизнес-процесс в обозначениях IDEF0 – прямоугольник (блок), его связи с элементами внешней среды или другими процессами – это стрелки. Внутри прямоугольника (блока) вписывается название функции/процесса и его номер.

Стрелки в IDEF0 могут быть:

- входящие – вводные, которые показывают «вход» процесса, ставят определенную задачу.
- исходящие – выводные результат деятельности, показывающие «выход» процесса.
- управляющие (сверху вниз) – механизмы управления (положения, инструкции и пр.).
- механизмы (снизу вверх) – что используется для того, чтобы произвести необходимую работу.

Основой метода являются следующие понятия:

1) В терминах IDEF0 система представляется в виде комбинации блоков и интерфейсных дуг. Блоки используются для представления функций системы; функции должны иметь имена, выраженный грамматической формой глагола.

2) Интерфейсная дуга – элемент, описывающий данные, управление, механизмы, оказывающие влияние на функцию, изображенную блоком. В зависимости от того, к какой стороне блока направлена дуга, она, соответственно, носит название «входная», «выходная», «управляющая». Изобразительным элементом, представляющим дугу, является стрелка. Место соединения дуги с блоком определяет тип интерфейса.

3) Управляющие выполнением функции данные входят в блок сверху, информация, которая подвергается воздействию функции, показана с левой стороны блока; результаты выхода показаны с правой стороны. Механизм, который выполняет функцию, представляется дугой, входящий в блок снизу.

Стрелки или дуги играют роль интерфейса и означают либо предметы (материальные объекты), либо информационные объекты – данные.

Большинство CASE-средств поддерживает методологию функционального моделирования IDEF0.

Задание к лабораторной работе:

Выполнение заданий 1 – 3 предполагает оформление отчета в текстовом редакторе Word (Times New Roman, 12 п., одинарный), в котором отражаются этапы выполнения заданий и результаты (ответы на вопросы, скриншоты построенной диаграммы IDEF0, количественный анализ диаграммы и т.д.)

Задание 1. Изучение методологии функционального моделирования IDEF0.

Изучить основы методологии IDEF0. В отчете отразить назначение диаграммы IDEF0 и описать основные компоненты данной диаграммы (блоки и дуги).

Задание 2. Для заданной предметной области построить функциональный блок и интерфейсные дуги в нотации IDEF0.

1.1. Изучить предметную область и определить входы (input), выходы (output), управления (control) и механизмы (mechanism) для разрабатываемой диаграммы бизнес-процесса.

1.2. Построить диаграмму IDEF0 для заданной предметной области.

1.3. Отобразить на диаграмме IDEF0 декомпозицию функций.

Принцип декомпозиции применяется при разбиении сложного процесса на составляющие его функции. При этом уровень детализации процесса определяется непосредственно разработчиком модели. Декомпозиция позволяет представлять модель системы в виде иерархической структуры отдельных диаграмм, что делает ее менее перегруженной и легко усваиваемой.

Задание 3. Провести количественный анализ построенной диаграммы и сделать выводы о правильности ее построения.

Для проведения количественного анализа диаграммы необходимо определить следующие показатели модели:

- количество блоков на диаграмме (N);
- уровень декомпозиции диаграммы (L);
- число стрелок, соединяющихся с *i*-м блоком диаграммы (A).

Необходимо стремиться к тому, чтобы количество блоков на диаграммах нижних уровней было бы ниже количества блоков на родительских диаграммах, т. е. с увеличением уровня декомпозиции убывал бы коэффициент N/L. По мере декомпозиции модели функции должны упрощаться, следовательно, количество блоков должно убывать.

Коэффициент сбалансированности диаграммы определяется по следующей формуле:

$$K_b = \left| \frac{\sum_{i=1}^N A_i}{N} - \max_{i=1}^N (A_i) \right|.$$

Необходимо стремиться, чтобы K_b был минимален для диаграммы. Помимо анализа графических элементов диаграммы необходимо рассматривать наименования блоков. Для оценки имен составляется словарь элементарных (тривиальных) функций моделируемой системы.

После формирования словаря и составления пакета диаграмм системы необходимо рассмотреть нижний уровень модели. Если на нем обнаружатся совпадения названий блоков диаграмм и слов из словаря, то это говорит, что достаточный уровень декомпозиции достигнут. Коэффициент, количественно отражающий данный критерий, можно записать как $L * C$ – произведение уровня модели на число совпадений имен блоков со словами из словаря. Чем ниже уровень модели (больше L), тем ценнее совпадения.

Контрольные вопросы

1. Дайте определение понятию «структурная модель».
2. Каково основное назначение структурного моделирования бизнес-процессов.
3. Перечислите и поясните базовые показатели
4. Что обозначают стрелки в нотации IDEF0 и какие их виды существуют?
5. Что такое интерфейсная дуга?

Лабораторная работа № 2. Моделирование потоков данных автоматизируемых бизнес-процессов и задач

Цель занятия: изучить основы моделирования потоков данных автоматизируемых бизнес-процессов и задач.

Теоретические сведения. Диаграмма потоков данных DFD и принципы ее построения.

Диаграмма потоков данных (Data Flow Diagrams) DFD – это граф, на котором показано движение значений данных от их источников через преобразующие их процессы к их потребителям в других объектах. Цель DFD – продемонстрировать, как каждый процесс преобразует свои входные данные в выходные, а также выявить отношения между этими процессами.

DFD используются для описания движения документов и обработки информации, как дополнение к IDEF0.

Для построения DFD традиционно используются две различные нотации, соответствующие методам Йордона-ДеМарко и Гейна-Сэрсона, отличающиеся друг от друга графическим изображением символов. В соответствии с DFD-методом модель системы определяется как иерархия диаграмм потоков данных, описывающих асинхронный процесс преобразования информации от ее ввода в систему до выдачи потребителю.

Основными компонентами диаграмм потоков данных являются:

- внешние сущности;
- системы и подсистемы;
- процессы;

- накопители данных;
- потоки данных.

Внешняя сущность – это материальный объект или физическое лицо, являющиеся источником или приемником информации (заказчики, персонал, поставщики, клиенты, склад). Внешние сущности находятся за пределами границ анализируемой системы. Внешняя сущность обозначается прямоугольником, расположенным над диаграммой и бросающим на нее тень для того, чтобы можно было выделить этот символ среди других обозначений.

При построении модели сложной системы она может быть представлена в самом общем виде на так называемой контекстной диаграмме в виде одной системы как единого целого либо может быть декомпозирована на ряд подсистем.

Процесс представляет собой преобразование входных потоков данных в выходные в соответствии с определенным алгоритмом. Физически процесс может быть реализован различными способами: это может быть подразделение организации (отдел), выполняющее обработку входных документов и выпуск отчетов, программа, аппаратно-реализованное логическое устройство и т.д.

Накопитель данных – это абстрактное устройство для хранения информации, которую можно в любой момент поместить в накопитель и через некоторое время извлечь, причем способы помещения и извлечения могут быть любыми.

Накопитель данных может быть реализован физически в виде микрофиши, ящика в картотеке, таблицы в оперативной памяти, файла на магнитном носителе и т.д.

Поток данных определяет информацию, передаваемую через некоторое соединение от источника к приемнику. Реальный поток данных может быть информацией, передаваемой по кабелю между двумя устройствами, пересылаемыми по почте письмами, магнитными лентами или дискетами, переносимыми с одного компьютера на другой, и т.д.

Главная цель построения иерархии DFD состоит в том, чтобы сделать описание системы ясным и понятным на каждом уровне детализации, а также разбить его на части с точно определенными отношениями между ними. Для достижения этого целесообразно пользоваться следующими рекомендациями:

- размещать на каждой диаграмме от 3 до 6-7 процессов; верхняя граница соответствует человеческим возможностям одновременного восприятия и понимания структуры сложной системы с множеством внутренних связей, нижняя граница выбрана по соображениям здравого смысла: нет необходимости детализировать процесс диаграммой, содержащей всего один или два процесса;

- не загромождать диаграммы несущественными на данном уровне деталями;

- декомпозицию потоков данных осуществлять параллельно с декомпозицией процессов; эти две работы должны выполняться одновременно, а не одна после завершения другой;

- выбирать ясные, отражающие суть дела имена процессов и потоков, при этом стараться не использовать аббревиатуры.

Задание к лабораторной работе:

Выполнение заданий 1 – 2 предполагает оформление отчета в текстовом редакторе Word (Times New Roman, 12 п., одинарный), в котором отражаются этапы выполнения заданий и результаты (ответы на вопросы, описание методологии DFD, скриншот созданной диаграммы DFD и т.д.)

Задание 1. Изучить методологию моделирования потоков данных DFD.

Изучить основы методологии DFD. В отчете отразить назначение диаграммы DFD и описать основные компоненты данной диаграммы.

Задание 2. Для заданной предметной области построить диаграмму DFD.

Декомпозируйте созданную контекстную диаграмму с помощью выбранного CASE-средства.

Контрольные вопросы

1. Дайте определение понятию «диаграмма DFD».
2. Какие нотации для построения диаграммы DFD вы знаете?
3. Перечислите основные компоненты диаграммы потоков данных.
4. Каким образом может быть физически реализован накопитель данных?
5. Какова главная цель построения иерархии диаграммы DFD?

Лабораторная работа № 3. Объектное моделирование автоматизируемых бизнес-процессов и задач (часть 1)

Цель занятия: познакомиться с основными понятиями и назначением языка Unified Modeling Language (UML), а также рассмотреть основные типы UML-диаграмм и способы их построения.

Теоретические сведения. Унифицированный язык объектно-ориентированного программирования UML и виды диаграмм UML.

Унифицированный язык объектно-ориентированного моделирования Unified Modeling Language (UML) явился средством достижения компромисса между этими подходами. Существует достаточное количество инструментальных средств, поддерживающих с помощью UML жизненный цикл информационных систем, и, одновременно, UML является достаточно гибким для настройки и поддержки специфики деятельности различных команд разработчиков.

UML представляет собой объектно-ориентированный язык моделирования, обладающий следующими основными характеристиками:

- является языком визуального моделирования, который обеспечивает разработку репрезентативных моделей для организации взаимодействия заказчика и разработчика ИС, различных групп разработчиков ИС;

- содержит механизмы расширения и специализации базовых концепций языка.

Словарь UML включает три вида строительные блоки:

- диаграммы;
- сущности;
- связи.

Сущности – это абстракции, которые являются основными элементами модели, связи соединяют их между собой, а диаграммы группируют представляющие интерес наборы сущностей.

Диаграмма – это графическое представление набора элементов, чаще всего изображенного в виде связного графа вершин (сущностей) и путей (связей). Язык UML включает 13 видов диаграмм.

С помощью языка UML можно строить различные виды диаграмм, которые подразделяются на 3 типа: структурные, поведенческие и диаграммы взаимодействия.

К структурным диаграммам относятся: диаграмма классов (Classdiagram), диаграмма объектов (Objectdiagram), диаграмма компонентов (Component diagram), диаграмма развертывания (Deployment diagram), диаграмма профилей (Profile diagram), диаграмма

композиционной/составной структуры (Compositestructurediagram), диаграмма пакетов (Package diagram).

Поведенческие UML-диаграммы: диаграмма кооперации (Collaborationdiagram), диаграмма состояний (Statemachinediagram), диаграмма активности (или деятельности) (Activitydiagram), диаграмма вариантов использования (или прецедентов) (Usecasediagram).

Диаграммы взаимодействий также относятся к поведенческим диаграммам, однако некоторые авторы выносят их в отдельную классификацию. Диаграммы взаимодействий: диаграмма последовательности (Sequencediagram), диаграмма коммуникации (Communicationdiagram), диаграмма обзора взаимодействий (Interactionoverviewdiagram), диаграмма синхронизации (Timingdiagram).

Задание к лабораторной работе:

Выполнение заданий 1 – 2 предполагает оформление отчета в текстовом редакторе Word (Times New Roman, 12 п., одинарный), в котором отражаются этапы выполнения заданий и результаты (ответы на вопросы, определения языка UML, классификация UML-диаграмм и т.д.).

Задание 1. Изучение унифицированного языка моделирования UML.

Используя различные русскоязычные и иностранные источники (печатные и электронные), рассмотреть существующие определения UML. В отчете представить и пояснить несколько найденных определений.

Задание 2. Найти и проанализировать различные типы UML-диаграмм.

Рассмотреть различные подходы к классификации UML-диаграмм. На основе собранной информации составить таблицу, состоящую из 3 столбцов: «№», «Наименование UML-диаграммы», «Назначение».

Представить подробное описание следующих типов UML-диаграмм:

- диаграмма классов;
- диаграмма деятельности;
- диаграмма состояний;
- диаграмма последовательности;
- диаграмма кооперации;
- диаграмма вариантов использования.

Контрольные вопросы

1. Что представляет собой язык UML и для чего его используют?
2. Какие существуют виды строительных блоков для диаграмм UML?
3. Дайте определение понятию «сущность» в терминах языка UML.
4. Какие виды диаграмм UML существуют? Перечислите названия конкретных диаграмм любого выбранного вида.

Лабораторная работа № 4. Объектное моделирование автоматизируемых бизнес-процессов и задач (часть 2)

Цель занятия: получить практические навыки построения UML-диаграмм на примере диаграммы классов.

Теоретические сведения. Назначение UML-диаграммы классов; основы ее построения.

Диаграммы классов показывают набор классов, интерфейсов, а также их связи.

Диаграммы этого вида чаще всего используются для моделирования объектно-ориентированных систем. Они предназначены для статического представления системы. Большинство элементов UML имеют уникальную и прямую графическую нотацию, которая дает визуальное представление наиболее важных аспектов элемента.

Диаграммы классов оперируют тремя видами сущностей UML:

- структурные;
- поведенческие;
- аннотирующие.

Структурные сущности – это «имена существительные» в модели UML. В основном, статические части модели, представляющие либо концептуальные, либо физические элементы. Основным видом структурной сущности в диаграммах классов является класс.

Поведенческие сущности – динамические части моделей UML. Это «глаголы» моделей, представляющие поведение модели во времени и пространстве. Основной из них является взаимодействие – поведение, которое заключается в обмене сообщениями между наборами объектов или ролей в определенном контексте для достижения некоторой цели. Сообщение изображается в виде линии со стрелкой, почти всегда сопровождаемой именем операции.

Аннотирующие сущности – это поясняющие части UML-моделей, иными словами, комментарии, которые можно применить для описания, выделения и пояснения любого элемента модели. Главная из аннотирующих сущностей – примечание. Это символ, служащий для описания ограничений и комментариев, относящихся к элементу либо набору элементов. Графически представлен прямоугольником с загнутым углом; внутри помещается текстовый или графический комментарий.

Класс – это описание набора объектов с одинаковыми атрибутами, операциями, связями и семантикой. Графически класс изображается в виде прямоугольника, разделенного на 3 блока горизонтальными линиями:

- имя класса;
- атрибуты (свойства) класса;
- операции (методы) класса.

Каждый класс должен обладать именем, отличающим его от других классов. Имя – это текстовая строка. Имя класса может состоять из любого числа букв, цифр и знаков препинания (за исключением двоеточия и точки) и может записываться в несколько строк. На практике обычно используются краткие имена классов, взятые из словаря моделируемой системы. Каждое слово в имени класса традиционно пишут с заглавной буквы (верблюжья конвенция), например Sensor (Датчик) или TemperatureSensor (ДатчикТемпературы).

Атрибут (свойство) – это именованное свойство класса, описывающее диапазон значений, которые может принимать экземпляр атрибута. Класс может иметь любое число атрибутов или не иметь ни одного. В последнем случае блок атрибутов оставляют пустым. Атрибут представляет некоторое свойство моделируемой сущности, которым обладают все объекты данного класса. Имя атрибута, как и имя класса, может представлять собой текст. На практике для именованного атрибута используются одно или несколько коротких существительных, выражающих некое свойство класса, к которому относится атрибут.

Операция (метод) – это реализация метода класса. Класс может иметь любое число операций либо не иметь ни одной. Часто вызов операции объекта изменяет его атрибуты.

Графически операции представлены в нижнем блоке описания класса.

Допускается указание только имен операций. Имя операции, как и имя класса,

должно представлять собой текст. На практике для именованной операции используются короткие глагольные конструкции, описывающие некое поведение класса, которому принадлежит операция. Обычно каждое слово в имени операции пишется с заглавной буквы, за исключением первого, например move (переместить) или isEmpty (проверка на пустоту).

Можно специфицировать операцию, устанавливая ее сигнатуру, включающую имя, тип и значение по умолчанию всех параметров, а применительно к функциям – тип возвращаемого значения.

Существует четыре типа связей в UML:

- зависимость;
- ассоциация;
- обобщение;
- реализация.

Эти связи представляют собой базовые строительные блоки для описания отношений в UML, используемые для разработки хорошо согласованных моделей.

Задание к лабораторной работе:

Выполнение заданий 1 – 3 предполагает оформление отчета в текстовом редакторе Word (Times New Roman, 12 п., одинарный), в котором отражаются этапы выполнения заданий и результаты (ответы на вопросы, определения классов и отношений, скриншот построенной UML-диаграммы класса и т.д.).

Задание 1. Изучение понятия «класс» при построении UML-диаграммы классов.

Используя различные русскоязычные и иностранные источники (печатные и электронные), определить назначение диаграммы классов. Рассмотреть основные компоненты класса, привести примеры классов для заданной предметной области с указанием и пояснением атрибутов и методов для каждого класса.

Задание 2. Изучение отношений между классами.

С помощью различных источников определить основные виды отношений между классами при построении UML-диаграмм. Дать определения всем типам отношений, описать графическое представление каждой связи. Привести конкретные примеры каждого вида отношений между классами.

Задание 3. Построение диаграммы классов для заданной предметной области.

С помощью выбранного CASE-средства построить диаграмму классов для заданной предметной области. В отчете представить описание функционала информационной системы, для которой моделируется диаграмма классов, скриншот построенной диаграммы. Описание классов диаграммы свести в таблицу, состоящей из 4 столбцов: «№», «Название класса», «Атрибуты», «Методы». Описание отношений между классами представить также в виде таблицы, имеющей следующие столбцы: «№», «Название связи», «Тип связи».

Контрольные вопросы

1. С какой целью строится UML-диаграмма классов и для чего она предназначена?
2. Какие существуют виды сущностей для построения диаграммы классов?
3. Что такое класс? Перечислите компоненты (блоки) класса.
4. Что такое методы, для чего они отображаются на диаграмме классов и каково их графическое представление?
5. Перечислите виды отношений между классами при построении UML-диаграммы классов.

Лабораторная работа № 5. Основы объектно-ориентированного программирования

Цель занятия: изучить основы объектно-ориентированного программирования на примере языка программирования C#.

Теоретические сведения. Применение объектно-ориентированного программирования к разработке программных проектов.

Объектно-ориентированное программирование или ООП (object-oriented programming) – методология программирования, основанная на представлении программы в виде совокупности объектов, каждый из которых является реализацией определенного типа, использующая механизм пересылки сообщений и классы, организованные в иерархию наследования.

Центральный элемент ООП – абстракция. Данные с помощью абстракции преобразуются в объекты, а последовательность обработки этих данных превращается в набор сообщений, передаваемых между этими объектами. Каждый из объектов имеет свое собственное уникальное поведение. С объектами можно обращаться как с конкретными сущностями, которые реагируют на сообщения, приказывающие им выполнить какие-то действия.

ООП характеризуется следующими принципами (по Алану Кею):

- все является объектом;
- вычисления осуществляются путем взаимодействия (обмена данными) между объектами, при котором один объект требует, чтобы другой объект выполнил некоторое действие; объекты взаимодействуют, посылая и получая сообщения; сообщение – это запрос на выполнение действия, дополненный набором аргументов, которые могут понадобиться при выполнении действия;
- каждый объект имеет независимую память, которая состоит из других объектов;
- каждый объект является представителем класса, который выражает общие свойства объектов данного типа;
- в классе задается функциональность (поведение объекта); тем самым все объекты, которые являются экземплярами одного класса, могут выполнять одни и те же действия;
- классы организованы в единую древовидную структуру с общим корнем, называемую иерархией наследования; память и поведение, связанное с экземплярами определенного класса, автоматически доступны любому классу, расположенному ниже в иерархическом дереве.

Основные определения ООП:

Абстрагирование (abstraction) – метод решения задачи, при котором объекты разного рода объединяются общим понятием (концепцией), а затем сгруппированные сущности рассматриваются как элементы единой категории. Абстрагирование позволяет отделить логический смысл фрагмента программы от проблемы его реализации, разделив внешнее описание (интерфейс) объекта и его внутреннюю организацию (реализацию).

Инкапсуляция (encapsulation) – техника, при которой несущественная с точки зрения интерфейса объекта информация прячется внутри него.

Наследование (inheritance) – свойство объектов, посредством которого экземпляры класса получают доступ к данным и методам классов-предков без их повторного определения. Наследование позволяет различным типам данных совместно использовать один и тот же код, приводя к уменьшению его размера и повышению функциональности.

Полиморфизм (polymorphism) – свойство, позволяющее использовать один и тот же интерфейс для различных действий; полиморфной переменной, например, может соответствовать несколько различных методов. Полиморфизм перекраивает общий код, реализующий некоторый интерфейс, так, чтобы удовлетворить конкретным особенностям отдельных типов данных.

Класс (class) – множество объектов, связанных общностью структуры и поведения; абстрактное описание данных и поведения (методов) для совокупности похожих объектов, представители которой называются экземплярами класса.

Объект (object) – конкретная реализация класса, обладающая характеристиками состояния, поведения и индивидуальности, синоним экземпляра.

C# (произносится Си-Шарп) – это новый язык программирования от компании Microsoft. Он входит в новую версию Visual Studio – Visual Studio.NET.

Для каждого типа данных C# существует соответствующий тип данных в CLR (Common Language Runtime). Это, в частности, означает, что каждый тип имеет два названия – полное (из CLR, его можно использовать в любом языке .NET) и сокращенное, которое используется в C#.

Для создания Windows-приложения на C# необходимо выполнить следующие действия:

1. запустить Visual Studio.NET; для создания нового пустого проекта C# зайти в меню File, выбрать New и затем Project;

2. в появившемся окне New Project выбрать тип приложения – Windows Application.

Работу с классами рассмотрим на следующем примере. Создайте новое консольное приложение для C# и введите следующий текст:

```
using System;
using System.Collections.Generic;
using System.Text;

namespace proba2
{
    class Worker//Началокласса
    {
        public int age = 0;
        public string name;
    } //Конецкласса
    class Test
    {
        [STAThread]
        static void Main(string[] args)
        {
            Worker wrk1 = new Worker();
            wrk1.age = 34;
            wrk1.name = "Иванов";
            Console.WriteLine(wrk1.name + ", " + wrk1.age);
        }
    }
}
```

Программа выведет на экран «Иванов, 34».

Атрибут STAThread указывает, что потоковой моделью COM для приложения

является однопоточное подразделение. Данный атрибут должен находиться в точке входа любого приложения, использующего Windows Forms; если он отсутствует, компоненты Windows могут работать неправильно, программа может «зависнуть». Если этот атрибут не указан, приложение использует модель многопоточного подразделения, которая не поддерживается для Windows Forms.

В строках

```
class Worker
{ public int age = 0;
public string name;
}
```

определен класс Worker. Внутри этого класса существует две переменные: целая age для возраста и name строкового типа для имени. В отличие от C/C++, можно задавать некоторое начальное значение прямо сразу после объявления переменной:

Начальное значение задавать не обязательно – это видно по переменной name.

Перед переменными пишется ключевое слово public. Значение у него такое же, как и в C++ - а именно это означает, что эта переменная (или функция) будет видна вне класса. Если не написать перед переменной никакого модификатора доступа, или написать private, то переменная не будет видна снаружи класса и ее смогут использовать только функции этого же класса (т. е. она будет для «внутреннего использования»).

Для вызова в программе вне класса функции private необходимо указывать имя класса, а для вызова функций public надо указывать имя экземпляра класса. Далее в строке

```
Worker wrk1 = new Worker();
```

используется созданный класс. А именно присваиваются некоторые значения для возраста и имени, и выводят их потом на экран.

Задание к лабораторной работе:

Выполнение заданий 1 – 3 предполагает оформление отчета в текстовом редакторе Word (Times New Roman, 12 п., одинарный), в котором отражаются этапы выполнения заданий и результаты (ответы на вопросы, описание и скриншот блок-схемы алгоритма для индивидуального задания, тексты программ, результаты тестирования, скриншот работы консольного приложения, скриншот оконного интерфейса для разработанного приложения и т.д.).

Задание 1. Составление блок-схемы алгоритма для разработки приложения.

Используя любое выбранное CASE-средство построить блок-схему алгоритма работы приложения для индивидуального задания. Список вариантов заданий представлен в таблице 1. В отчете представить текст задания, а также скриншот результата выполненного задания.

Таблица 1 – Варианты индивидуальных заданий для выполнения лабораторной работы № 5

Вариант	Индивидуальное задание
1	Даны катет и гипотенуза прямоугольного треугольника. Найдите площадь треугольника. Выведите результат на экран.
2	Введите значения трех сопротивлений r1, r2, r3 (соединение параллельное). Найдите сопротивление соединения. Выведите результат на экран.
3	Введите длину ребра куба. Найдите объем куба и площадь его поверхности. Выведите результат на экран.
4	Введите внутренний и внешний радиусы кольца. Найдите площадь кольца. Выведите результат на экран.
5	Введите радиус и высоту цилиндра. Вычислите объем цилиндра и площадь

	боковой поверхности. Выведите результат на экран.
6	Даны три стороны треугольника. Найдите площадь треугольника. Выведите результат на экран.
7	Введите радиус окружности. Найдите длину окружности и площадь круга. Выведите результат на экран.
8	Дана сторона равностороннего треугольника. Найдите площадь треугольника. Выведите результат на экран.
9	Введите значения трех сопротивлений r_1 , r_2 , r_3 (соединение последовательное). Найдите сопротивление соединения. Выведите результат на экран.
10	Введите длину, ширину и высоту параллелепипеда. Найдите объем параллелепипеда. Выведите результат на экран.

Задание 2. Разработка консольного приложения для индивидуального задания.

Для вариантов заданий, представленных в таблице 1 необходимо написать программу для консольного приложения C#. Обязательно предусмотреть установку цвета фона, символов, очистку окна, а также необходимые и понятные пояснения в процессе работы программы. Провести тестирование программы, задавая некорректные данные, слишком большие или малые значения параметров. На основании тестирования скорректировать типы данных и добавить, при необходимости, проверки.

Задание 3. Разработка оконного интерфейса для индивидуального задания.

Разработать оконный интерфейс для индивидуальных заданий из таблицы 1. За основу программы взять программный код, разработанный при выполнении задания 2 лабораторной работы №5. В отчете отразить скриншот оконного интерфейса, а также результаты работы программы.

Контрольные вопросы

1. Дайте определение понятию «объектно-ориентированное программирование».
2. Какими принципами характеризуется объектно-ориентированное программирование согласно утверждениям Алана Кея?
3. Дайте определения понятиям «инкапсуляция» и «полиморфизм». Объясните значения данных понятий своими словами.
4. Что такое C#? Какой разработчик предоставляет данный язык?
5. Какие действия необходимо выполнить для создания Windows-приложения на C#.

РЕКОМЕНДУЕМАЯ ЛИТЕРАТУРА

- 1 Волк, В. К. Практическое введение в программную инженерию : учебное пособие / В. К. Волк. – Санкт-Петербург : Лань, 2019. – 100 с. [электронный ресурс]: <https://e.lanbook.com/book/119634>
- 2 Маран, М. М. Программная инженерия : учебное пособие / М. М. Маран. – Санкт-Петербург : Лань, 2018. – 196 с. [электронный ресурс]: <https://e.lanbook.com/book/106733>
- 3 Барков, И. А. Объектно-ориентированное программирование : учебник / И. А. Барков. – Санкт-Петербург : Лань, 2019. – 700 с. [электронный ресурс]: <https://e.lanbook.com/book/11966>
- 4 Исследование операций в задачах программной инженерии : учебное пособие / Н. А. Соловьев, Е. Н. Чернопрудова, Н. А. Тишина, А. Ф. Валеев. – Санкт-Петербург : Лань, 2019. – 164 с. [электронный ресурс]: <https://e.lanbook.com/book/121486>
- 5 Пантелеев, Е. Р. Методы научных исследований в программной инженерии : учебное пособие для вузов / Е. Р. Пантелеев. – 2-е изд., стер. – Санкт-Петербург : Лань, 2021. – 136 с. [электронный ресурс]: <https://e.lanbook.com/book/152439>

МЕТОДИЧЕСКОЕ ОБЕСПЕЧЕНИЕ РАСЧЕТНО-ГРАФИЧЕСКОЙ РАБОТЫ ПО ДИСЦИПЛИНЕ

Целью расчетно-графической работы является получение практических навыков по освоению инженерных методик разработки программных средств в MSVisualStudio; освоение методики расчета трудоемкости программного проекта и составление календарного плана его реализации; оформлению программной документации в соответствии с требованиями стандартов.

1. ЗАДАНИЕ

Общее количество тем, представленных в приложении А, следует разбить на подгруппы по списочному количеству студентов в следующем порядке: БЭИС, ИТЭК, ПИЭ. В каждой подгруппе выбрать свою тему в соответствии с порядковым номером N фамилии студента в журнале своей группы. Например, если всего в БЭИС 15 студентов, то вариант N=1 из группы ИТЭК должен взять 16-ю тему. Задания разделены на три класса, в зависимости от предполагаемой оценки. В зависимости от объема выполненных пунктов задания, глубины их проработки выставляется оценка за расчетное задание.

Предложенные тематики задания следует рассматривать как общее направление, сложность и глубину проработки которой студент может выбирать самостоятельно, проконсультировавшись с преподавателем. Оценка будет зависеть от глубины проработки темы и качества ее программной реализации.

Задание на оценку «удовлетворительно».

1. В соответствии с вариантом N выбрать тему разработки, которая приведена в приложении А.

2. Для указанной темы изучить математические аспекты метода и конкретизировать постановку задачи, после этого написать техническое задание (ТЗ) на разработку в соответствии с требованиями ЕСПД.

3. Разработать схему алгоритма программы (блок-схему) решения поставленной задачи.

4. Разработать функциональную модель для решаемой задачи в соответствии со стандартом IDEF0.

4. Разработать программу с для решения задачи на заданном алгоритмическом языке и провести проверку ее тестирование.

5. Оформить отчет в соответствии с требованиями.

Задание на оценку «хорошо».

1. В соответствии с вариантом N выбрать тему разработки, которая приведена в приложении А.
2. Для указанной темы изучить математические аспекты метода и конкретизировать постановку задачи, после этого написать техническое задание (ТЗ) на разработку в соответствии с требованиями ЕСПД.
3. Разработать схему алгоритма программы (блок-схему) решения поставленной задачи. При изображении блок-схемы руководствоваться требованиями ЕСПД.
4. Провести анализ поставленной задачи с точки зрения объектно-ориентированного подхода и выполнить разработку модели, включающей в себя диаграмму классов, диаграмму деятельности, а также, при необходимости, другие диаграммы на языке UML.
5. Разработать программу для решения поставленной задачи с использованием рассмотренных в п. 4 классов на заданном алгоритмическом языке.
6. Выполнить тестирование программы.
7. В соответствии с требованиями ЕСПД составить описание программы.
8. Оформить отчет в соответствии с требованиями.

Задание на оценку «отлично».

1. В соответствии с вариантом N выбрать тему разработки, которая приведена в приложении А.
2. Для указанной темы изучить математические аспекты метода и конкретизировать постановку задачи, после этого написать техническое задание (ТЗ) на разработку в соответствии с требованиями ЕСПД.
3. Разработать схему алгоритма программы (блок-схему) решения поставленной задачи. При изображении блок-схемы руководствоваться требованиями ЕСПД.
4. Построить модели для решаемой задачи:
 - функциональную модель в соответствии со стандартом IDEF0;
 - модель, применяемую при объектно-ориентированном проектировании включающую в себя диаграмму классов, диаграмму деятельности, а также, при необходимости, другие диаграммы на языке UML.
5. Разработать две программы, реализующие функциональную и объектно-ориентированную модели, представленные в п.4. на заданном алгоритмическом языке.
6. Выполнить тестирование программ.
7. Провести сравнение двух разработанных программ по каким-либо критериям (не меньше трех), например: быстродействие, требуемая память, расширяемость, надежность. ованиями.

2. ТРЕБОВАНИЯ К ОФОРМЛЕНИЮ И СТРУКТУРА ОТЧЕТА ПО РГР

Оформление отчета: размер шрифта 12, шрифт TimesNewRoman, межстрочный интервал – одинарный, поля – со всех сторон по 2 см

В данном РГР отчет должен содержать следующие элементы:

- титульный лист;
- содержание;
- математическое описание и блок-схема алгоритма;
- техническое задание;
- модели предметной области;
- описание программы (на оценку «хорошо»);
- тестирование программы;

- результаты сравнения программ (на оценку «отлично»);
- список использованных источников;
- приложения.

Более подробно требования к оформлению отчета по РГР изложены в «Требования_к_оформлениюРПЗ_2016.docx».

РЕКОМЕНДУЕМАЯ ЛИТЕРАТУРА

1 Волк, В. К. Практическое введение в программную инженерию : учебное пособие / В. К. Волк. – Санкт-Петербург : Лань, 2019. – 100 с. [электронный ресурс]: <https://e.lanbook.com/book/119634>

2 Маран, М. М. Программная инженерия : учебное пособие / М. М. Маран. – Санкт-Петербург : Лань, 2018. – 196 с. [электронный ресурс]: <https://e.lanbook.com/book/106733>

3 Барков, И. А. Объектно-ориентированное программирование : учебник / И. А. Барков. – Санкт-Петербург : Лань, 2019. – 700 с. [электронный ресурс]: <https://e.lanbook.com/book/11966>

4 Исследование операций в задачах программной инженерии : учебное пособие / Н. А. Соловьев, Е. Н. Чернопрудова, Н. А. Тишина, А. Ф. Валеев. – Санкт-Петербург : Лань, 2019. – 164 с. [электронный ресурс]: <https://e.lanbook.com/book/121486>

5 Пантелеев, Е. Р. Методы научных исследований в программной инженерии : учебное пособие для вузов / Е. Р. Пантелеев. – 2-е изд., стер. – Санкт-Петербург : Лань, 2021. – 136 с. [электронный ресурс]: <https://e.lanbook.com/book/152439>

ПРИЛОЖЕНИЕ А

Темы расчетных заданий

Номер темы выбрать в соответствии с номером варианта N.

1. Программа нахождения минимума функции двух переменных методом Ньютона-Рафсона.
2. Программа для вычисления определенного интеграла функции одной переменной методом Монте-Карло.
3. Программа для безусловной оптимизации функций одной переменной методом золотого сечения.
4. Программа безусловной оптимизации функций двух переменных методом по координатного спуска.
5. Программа для решения систем линейных уравнений 3-го порядка методом Гаусса.
6. Программа для обращения матриц 3-го порядка.
7. Программа для нахождения определителя матрицы порядка ≤ 4 .
8. Программа для вычисления определенного интеграла методом трапеций.
9. Программа для решения нелинейного уравнения методом половинного деления.
10. Программа для вычисления сложных процентов и их графической иллюстрации.
11. Программа для решения дифференциального уравнения 2-го порядка методом Рунге-Кутты 4 порядка и графического отображения результата.
12. Программа для решения дифференциального уравнения 3-го порядка методом Эйлера и графического отображения результата.
13. Программа для вычисления определенного интеграла методом прямоугольников и графического отображения результата.
14. Программа для решения дифференциального уравнения 2-го порядка усовершенствованным методом Эйлера и графического отображения результата.
15. Программа для нахождения собственных значений матрицы 3-го порядка.
16. Программа для решения нелинейного уравнения методом простых итераций.
17. Программа для интерполяции функции двух переменных на равномерной сетке.
18. Программа для вычисления определенного интеграла методом Симпсона.
19. Программа для линейной аппроксимации заданной функции по методу наименьших квадратов и графической иллюстрации ее решения.
20. Программа для решения систем линейных уравнений 3-го порядка методом простых итераций.
21. Программа нахождения минимума функции двух переменных методом Хука-Дживса.
22. Программа нахождения минимума функции двух переменных методом метод наискорейшего спуска.
23. Программа нахождения минимума функции двух переменных методом Нелдера-Мида.
24. Программа нахождения минимума функции двух переменных методом дробления шага.

25. В тексте, хранящемся в файле, найти возможные номера телефонов и имена их владельцев.
26. Программа для решения дифференциального уравнения 3-го порядка усовершенствованным методом Эйлера.
27. Программа для решения дифференциального уравнения 1-го порядка методом Адамса.
28. Программа для прогнозирования значений числового ряда по методу скользящего среднего.
29. Программа для прогнозирования значений числового ряда по методу экспоненциального сглаживания.
30. Программа для интерполяции значений числового ряда с графическим отображением результата (график функции и интерполяции).
31. Программа, реализующая масштабирование изображения, загружаемого из файла.
32. Программа для решения дифференциального уравнения 1-го порядка методом Адамса-Бэшфоркса-Моултона.
33. Программа, реализующая перемещение произвольной фигуры по экрану. Фигура считывается из файла.
34. Программа для решения дифференциального уравнения 3-го порядка методом Рунге-Кутты третьего порядка.
35. Трехмерная матрица содержит k страниц, на каждой странице расположена матрица размера 3×3 . Найти определители для каждой страницы и расположить страницы в порядке возрастания их определителей.
36. Дан текстовый файл, имеющий структуру «Фамилия И.О. – улица – номер дома – квартира – номер телефона». Используя регулярные выражения вывести на экран фамилии всех абонентов, проживающих на заданной улице и подсчитать количество этих абонентов.