**06.09.19**

**Программа** – это данные, предназначенные для управления конкретными компонентами системы, обработки информации в целях реализации определенного алгоритма.

**Программное обеспечение** – это совокупность программ, системы обработки информации и программных документов, необходимых для эксплуатации.

**Свойства программного обеспечения:**

1. Необходимость документирования.

2. Эффективность программного обеспечения

3. Надёжность.

4. Возможность сопровождения.

**Системная программа** – программа, предназначенная для поддержания работоспособности системы, обработки информации или повышения эффективности её использования.

**Прикладная программа** – это программа, предназначенная для решения задачи или класса задач в определенной области применения Системы Обработки Информации (СОИ).

**Промежуточное ПО** – это совокупность программ, осуществляющих управление вторичными ресурсами, ориентированными на решение определенного класса задач.

**Управляющая программа** – это системная программа, реализующая набор функций управления, которые включают в себя управление ресурсами и взаимодействие с внешней средой СОИ, востоковедение работы системы после проявления неисправности в технических средствах.

**Система программирования** – это система образуемая языком программирования, компилятором или интерпретатором программ, представленными на этом языке соответствующей документацией, а так же вспомогательными средствами для подготовки программ к форме, пригодной для выполнения.

**Программный модуль** – это программа или функционально завершенный фрагмент программы, предназначенный для хранения, трансляции, объединения с другими программными модулями и загрузки в ОП.

**Исходный модуль** – программный модуль на исходном языке, обрабатываемый транслятором и представляемый для него как целая, достаточная для проведения трансляция.

**Трансляция** – это преобразование программы, предоставленной на одном языке программирования в программу на другом языке программирования в определённом смысле, равносильную первой.

**Автокод** – это символьный язык программирования, по своей структуре предложение которого в основном подобны командам и обрабатываемым данным конкретного машинного языка.

**Объектный модуль** - программный модуль, получаемый в результате трансляции исходного модуля.

**Загрузочный модуль** – программный модуль, представленный в форме пригодной для загрузки в ОП для выполнения.

**Интерпретация** – реализация смысла в виде некоторого синтаксически законченного текста, представленная на каком то языке.

**09.09.19 – 10.09.19**

Концепция разработки программного модуля. Понятие жизненного цикла программы.

Жизненный цикл программного обеспечения - это непрерывный процесс, который начинается с момента принятия решения о необходимости его создания и заканчивается в момент его полного изъятия из эксплуатации.

Основным нормативным документом, регламентирующим жизненный цикл программного обеспечения, является международный стандарт ISO/IEC 12207.

Структура жизненного цикла базируется на трех группах процессов:

1. Основные процессы (приобретение, поставка, разработка, эксплуатация и сопровождение).

2. Вспомогательные процессы (документирование, управление конфигурацией, обеспечение качества, верификация, аттестация, оценка, аудит, решение проблем).

3. Организационные процессы (управление проектами, создание инфраструктуры проекта, определение, оценка и улучшение самого жизненного цикла, обучение).

Разработка включает в себя: все работы по созданию программного обеспечения и его компонент в соответствии с заданными требованиями, включая оформление проектной и эксплуатационной документации, подготовку материалов необходимых для проверки работоспособности и соответствующего качества программных продуктов, материалов, необходимых для организации обучения персонала. Разработка программного обеспечения включает в себя анализ, проектирование и реализацию-программирование.

Эксплуатация включает в себя: работы по внедрению компонентов программного обеспечения в эксплуатацию, конфигурирование базы данных и рабочих мест пользователей, обеспечение эксплуатационной документацией, проведение обучения персонала, непосредственно эксплуатацию в том числе локализацию проблем и устранение причин их возникновения, модификацию программного обеспечения в рамках установленного регламента, подготовку предложений по совершенствованию, развитию и модернизации системы.

Управление проектом связано с вопросами планирования и организации работ, создания коллективов разработчиков и контроля за сроками и качеством выполняемых работ. Техническое и организационное обеспечение проекта включает: выбор методов и инструментальных средств для реализации проекта, определение методов, описание промежуточных состояний разработки проекта, разработку методов и средств испытаний ПО, обучение персонала. Обеспечение качества проекта связано с проблемами проверки соответствию требованиям на данном этапе проектирования и тестирования ПО. В процессе реализации проекта важное место занимают вопросы идентификации, описания и контроля конфигурации отдельных компонентов и всей системы в целом.

Управление конфигурацией является одним из вспомогательных процессов, поддерживающих основные процессы жизненного цикла, прежде всего процессы разработки и сопровождения ПО. При создании проектов сложных ИС, состоящих из многих компонентов, каждый из которых может иметь разновидности или версии, возникает проблема учета их связей и функций. Создание унифицированной структуры и обеспечение развития всей системы. Управление конфигурацией позволяет организовать, систематически учитывать и контролировать внесение изменений в ПО на всех стадиях жизненного цикла.

Общие принципы и рекомендации конфигурационного учета и планирования управления конфигурациями отражены в проекте стандарта ISO 1220 – 2.

Модели жизненного цикла:

1. Каскадная модель. Основной характеристической особенностью каскадной модели является разбиение всей разработки на этапы, причем переход с одного этапа на следующий происходит только после того, как будет полностью завершена работа на текущем. (схема : 1.анализ, 2.проектирование, 3.реализация, 4.внедрение, 5.сопровождение). Каждый этап завершается выпуском полного комплекта документации.

2. Спиральная модель. Делает упор на начальные этапы: анализ и проектирование. На этих этапах реализуемость технических решений проверяется путем создания прототипов. Каждый виток спирали соответствует созданию фрагмента или версии ПО. На нем уточняются цели и характеристики проекта, определяется его качество, и планируются работы следующего витка спирали. САНЯ. При итеративном способе разработки недостающую работу можно будет выполнить на следующей итерации. Главная задача – можно быстро показать пользователям работоспособный продукт, тем самым активизируя процесс уточнения и дополнения требований. Основная проблема данного метода – не нарушать временные ограничения на каждый из этапов жизненного цикла.

12.09.19

Современные методы разработки ПО.

Метод нисходящего проектирования (метод пошаговой детализации, метод иерархического проектирования, top-down подход). Суть метода заключается в определении спецификаций компонентов системы путем последовательного выделения в ее составе отдельных составляющих и их постепенной детализации до уровня, обеспечивающего однозначное понимание того, что и как необходимо разрабатывать и реализовывать. Реализация метода нисходящего проектирования тесно связана с другим понятием программирования – модульное проектирование. Модуль – последовательность логически взаимосвязанных фрагментов задачи, оформленных как отдельная часть программы.

Свойства модулей:

1. На модуль можно ссылаться.

2. По завершении работы модуль должен возвращать управление тому модулю, который его вызвал.

3. Модуль должен иметь один вход и один выход.

4. Модуль должен иметь небольшой размер.

Преимущества модульного принципа:

1. Большую программу могут разрабатывать одновременно несколько исполнителей, что позволяет сократить сроки разработки.

2. Появляется возможность создавать и многократно использовать в дальнейшем библиотеки наиболее употребляемых программ.

3. Упрощается процедура загрузки больших программ в ОП, когда требуется ее сегментация.

4. Возникает много естественных контрольных точек для наблюдения за осуществлением хода разработки программ, а в последующем за ходом исполнения программ.

5. Обеспечивается более эффективное тестирование программ, проще осуществляется проектирование и последующая отладка.

16.09.19

Тестирование программного обеспечения это процесс исследования ПО с целью получения информации о качестве продукта.

**Виды тестирования по целям:**

Функциональная - направленна на проверку того какие функции программного продукта реализованы и того на сколько верно они реализованы. Состав функционального тестирования: требования, анализ, тест кейсы, ручное тестирование, автоматизированное тестирование, исследовательское тестирование и отчетность.

Не функциональное тестирование - проверяет корректность работы нефункциональных требований. Этот вид тестирования включает в себя тестирование производительности: нагрузочная, стрессовая. Тестирование стабильности, конфигурационная, тестирование пользовательского интерфейса, тестирование удобства использования, тестирование защищенности, инсталяционное тестирование, тестирование совместимости, тестирование надежности, тестирование локализации .

**По степени автоматизации:** ручное и автоматизированное.

**По позитивности сценария** бывают позитивным (проверка программного продукта на соответствие ожидаемому поведению) и негативный (проверяет будет ли программный продукт работать в случае, когда поведение пользователя отличается от ожидаемого.

**По доступу к коду программного продукта** делится на тестирование белого ящика (тестирование программного продукта с доступом к коду) и тестирование черного ящика(тестирование без доступа к коду) и тестирование серого ящика(тестирование основанное на ограниченном знании внутренней структуры программного продукта.

**По уровню:** 1) модульное (unit) проверка корректной работы отдельных единиц системы программного продукта 2) интеграционное тестирование - проверка взаимодействия между несколькими единицами программного продукта 3) системная - проверка работы всей системы на соответствие заявленным требованиям к программному продукту.

**По исполнителю:** альфа тестирование и бета тестирование.

**По формальности:** тестирование по тестам, исследовательское тестирование и свободное тестирование.

**По важности:** дымное тестирование, тестирование критического пути, расширенное тестирование.

16.09.19

Методы проектирования алгоритмов.

Методы проектирования алгоритмов можно классифицировать по признакам:

1. Степень автоматизации проектных работ. По степени автоматизации проектирования алгоритмов можно выделить: методы традиционного проектирования и методы автоматизированного проектирования.

2. Принятая методология процесса разработки.

Проектирование алгоритмов основывается на различных подходах:

1. Структурное проектирование.

2. Информационное моделирование предметной области и связанных с ней приложений.

3. Объектно-ориентированное проектирование программных продуктов.

В зависимости от объекта структурирования различают:

1. Функционально-ориентированные методы.

2. Методы структурирования данных.

Структурный подход используют:

1. Диаграммы потоков данных.

2. Интегрированную структуру данных предметной области.

3. Диаграммы декомпозиции.

4. Структурные схемы.

Даталогические модели имеют логический и физический уровни. Физический уровень соответствует организации хранения данных в памяти компьютера, а логический уровень реализован в виде:

1. Концептуальной модели БД.

2. Внешних моделей данных.

Обьектно-ориентированный подход основан на:

1. Выделении классов объектов.

2. Установлении характерных свойств объектов и методов их обработки.

3. Создание иерархии классов, наследование свойств объектов и методов их обработки.

Объектно-ориентированный анализ - это анализ предметной области и выделение объектов, определение свойств и методы обработки объектов, установление их взаимосвязей.

Объектно-ориентированное проектирование соединяет процесс объектной декомпозиции и представления с использованием моделей данных, проектируемой системы на логическом и физическом уровнях в статике и динамике.

18.09.19

Современные языки программирования.

1. JavaScript - прототипно-ориентированный сценарный язык программирования.

2. Java - строго типизированный объектно-ориентированный язык программирования.

3. PHP - является распространенным интерпретируемым языком общего назначения с открытым исходным кодом. Язык для web-разработки.

4. Python - высокоуровневый язык общего назначения ориентированный на повышение производительности разработчика, читаемости кода и на разработку web-приложений.

5. C# - объектно-ориентированный язык программирования с синтаксисом ближе к C++, имеет статистическую типизацию, поддерживает полиморфизм, перезагрузку операторов, делегаты, атрибуты, события, свойства, обобщенные типы и методы, итераторы, анонимные функции.

6. C++ - компилируемый статистически-типизированный язык общего назначения.

7. Ruby - ориентирован на разработку web-приложений, обладает независимой от ОС реализацией многопоточности, строго динамической типизацией, сборщиком мусора.

8. CSS - формальный язык описания внешнего вида документа, написанного с использованием языка разметки.

9. C - компилируется статистически-типизированный язык программирования общего назначения для разработки низкоуровневых приложений.

10. Shell - скрипты используются для автоматизации обновления ПО.

11. R - язык для статистической обработки данных и работы с графикой, также свободная программная среда вычислений с открытыми исходными кодом.

12. Perl - высокоуровневый интерпретируемый динамический язык общего назначения. Особенности языка это работа с текстом и с выражениями.

13. Scala - мульти-парадигмальный язык программирования, спроектированный кратким и типобезопасным, для простого и быстрого создания компонентного ПО.

14. Go - компилируемый многопоточный язык, разрабатывался как язык системного программирования.

15. SQL - языке структурированных запросов.

16. Husckell - стандартизированный чистый функциональный язык программирования общего назначения.

17. Swift - открытый мульти-парадигмальный компилируемый язык программирования общего назначения.

18. Matlab - высокоуровневый интерпретируемый язык программирования, основанный на матрицах структуры данных. Программы, написанные на нем, бывают 2х типов: функции и скрипты.

19. VisualBasic.

20. Delphi.

21. Groovy - объектно-ориентированный язык работающий для платформы java как дополнение.

22. Assembler - низкоуровневый машино-ориентированный язык программирования.

23. D - мульти-парадигмальный язык программирования похожий на C++.

20.09.19

Понятие блох схемы по ГОСТу.

При описании алгоритмов используются блок схемы. Согласно ГОСТу 19.701-90 под схемой понимается графическое представление определения, анализа или метода решения задачи. Символы, приведенные в государственном стандарте, могут использоваться в следующих типах схем:

1. Схемы данных определяют последовательность обработки данных и их носителей.

2. Схемы программ отображают последовательность операций в программе.

3. Схемы работы системы отображают управление операциями и потоки данных в системе.

4. Схемы взаимодействия программ отображают путь активации программ и их взаимодействия с соответствующими данными.

5. Схемы ресурсов системы отображают конфигурацию блоков данных и обрабатывающих блоков.

Элементы графической нотации:

1. Символы данных указывают на наличие данных, вид носителя или способ ввода/вывода данных.

2. Символы процесса указывают операции, которые следует выполнить над данными.

3. Символы линий указывают потоки данных между процессами или носителями данных, а также потоки управления между процессами.

4. Специальные символы используются для обеспечения написания и чтения схем.

Каждую категорию символов делят на основные и специфические. Основной символ используется в тех случаях, когда точный вид процесса или носителя данных неизвестен или отсутствует необходимость в описании фактического носителя данных. Специфический символ используется в тех случаях, когда известен точный вид процесса или носителя данных и это необходимо отразить на схеме.

**Правила и рекомендации построения схем:**

1. Допускается зеркально отображать символы и поворачивать их вокруг оси.

2. Большинство символов запускают задание внутри них текстовых пояснений.

3. Количество пересечений линий следует минимизировать.

4. Если две или более линии объединены в одну, то место объединения должно быть смещено.

5. Несколько выходов из символа решения сделает показывать одним из способов:

1) Несколькими линиями от данного стала к другим символам.

2) Одной линией от данного символа, которая потом разветвляется.

6. Вместо одного символа с соответствующим текстом могут быть использованы несколько символов с перекрытием изображения, каждый из которых может содержать дополнительный текст.

7. Если направление стрелки не указано, то направление потока считается сверху вниз, или слева направо.