Центр компьютерного обучения «СПЕЦИАЛИСТ» при МГТУ им. Н.Э.Баумана

Основы программирования и баз данных

Модуль 3. Методологии и языки программирования

КАШКИН ЕВГЕНИЙ ВЛАДИМИРОВИЧ

Учебный центр компьютерного обучения «СПЕЦИАЛИСТ» при МГТУ им. Н.Э.Баумана

ekashkin@specialist.ru

Образование:

- 2001-2006 г.г. Политехнический колледж №19. Специальность «Вычислительные машины, комплексы, системы и сети»
- 2006-2009 г.г. АНО ВПО «Институт деловой карьеры». Специальность «Прикладная информатика в экономике»

Опыт работы:

Преподаватель ИДК, заместитель декана ИДК.
Старший преподаватель МГУПИ, общий стаж работы в образовательной сфере 10 лет из них в ВУЗах 4 года.

Работа в сфере программирования, в том числе создание проектов с применением языков высокого уровня C/C++/Pascal/Delphi/Visual C++ более 5 лет.

Цели занятия

- Стадии и этапы разработки программ. Проектирование. Реализация.
- Проблемы программирования;
- Методологии программирования. Классификация методологий программирования (структурное, объектно-ориентированное, логическое, функциональное, программирование в ограничениях).
- Структурное программирование. Базовые принципы (пошаговая детализация, модульное структурное программирование);
- Объектно-ориентированное программирование. Базовые принципы (абстрагирование; инкапсуляция; наследование, полиморфизм);
- Языки программирования. Классификация.

Стадии и этапы разработки программ.

Определяются стандартами:

- ГОСТ 34.601-90
- ISO/IEC 12207:2008 «System and software engineering — Software life cycle processes» (российский аналог — ГОСТ Р ИСО/МЭК 12207-2010 Информационная технология. Системная и программная инженерия. Процессы жизненного цикла программных средств)

Стандарт ГОСТ 34.601-90

Стандарт ГОСТ 34.601-90 предусматривает следующие стадии и этапы создания автоматизированной системы:

1. Формирование требований к АС
2. Разработка концепции АС
3. Техническое задание
4. Эскизный проект
5. Технический проект
6. Рабочая документация
7. Ввод в действие
8. Сопровождение АС.

Модели жизненного цикла

Модель жизненного цикла ПО — структура, определяющая последовательность выполнения и взаимосвязи процессов, действий и задач на протяжении жизненного цикла. Модель жизненного цикла зависит от специфики, масштаба и сложности проекта и специфики условий, в которых система создается и функционирует.

Модель ЖЦ ПО включает в себя:

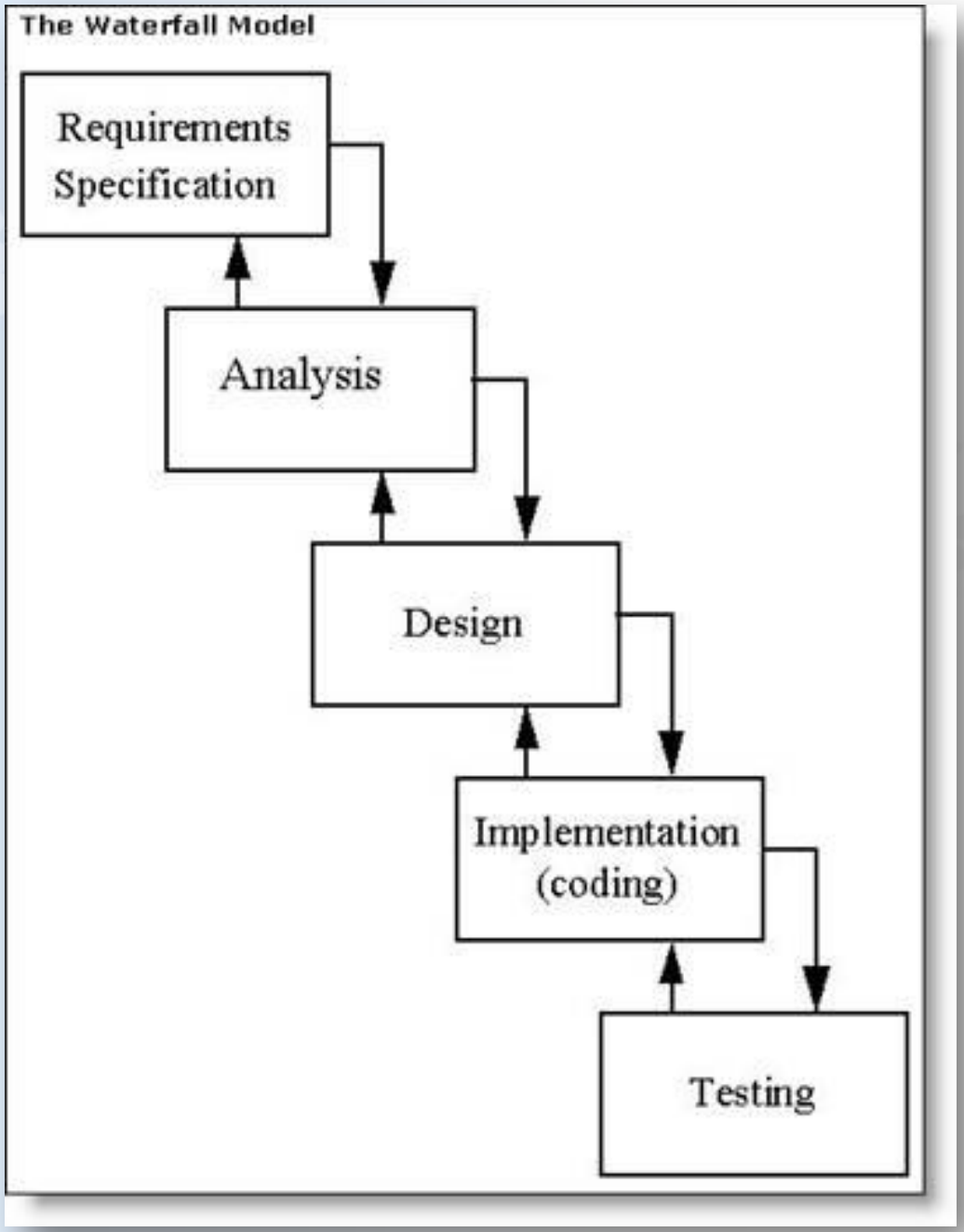
- Стадии;
- Результаты выполнения работ на каждой стадии;
- Ключевые события — точки завершения работ и принятия решений.

Водопадная (каскадная, последовательная) модель

Водопадная модель жизненного цикла была предложена в 1970 г. Уинстоном Ройсом. Она предусматривает последовательное выполнение всех этапов проекта в строго фиксированном порядке. Переход на следующий этап означает полное завершение работ на предыдущем этапе. Требования, определенные на стадии формирования требований, строго документируются в виде технического задания и фиксируются на все время разработки проекта. Каждая стадия завершается выпуском полного комплекта документации, достаточной для того, чтобы разработка могла быть продолжена другой командой разработчиков.

Этапы проекта в соответствии с каскадной моделью:

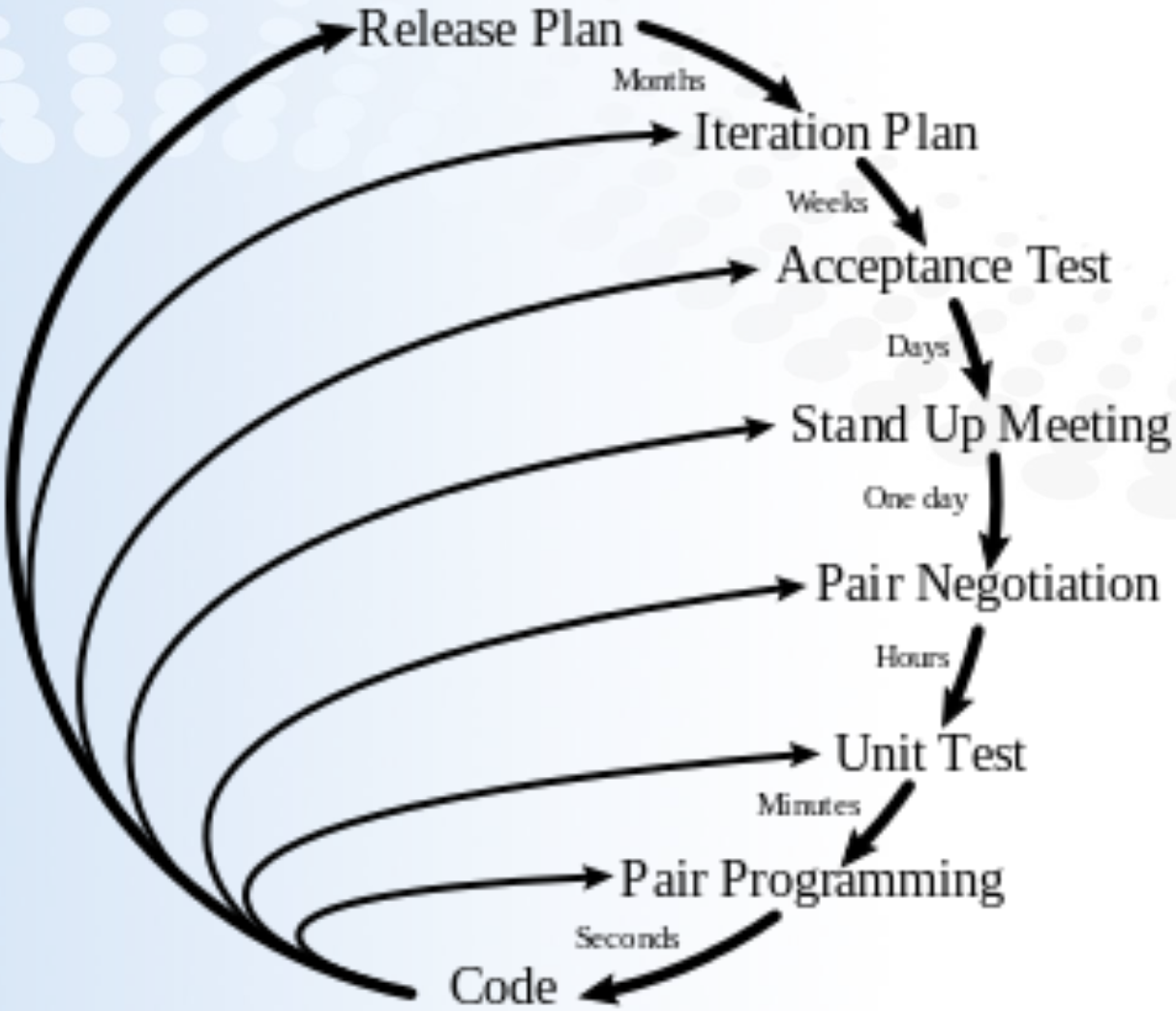
- Формирование требований;
- Проектирование;
- Реализация;
- Тестирование;
- Внедрение;
- Эксплуатация и сопровождение.


The Waterfall Model
Requirements Specification
Analysis
Design
Implementation (coding)
Testing

Итерационная модель

Модель предполагает разбиение жизненного цикла проекта на последовательность итераций, каждая из которых напоминает «мини-проект», включая все процессы разработки в применении к созданию меньших фрагментов функциональности, по сравнению с проектом в целом. Цель каждой *итерации* — получение работающей версии программной системы, включающей функциональность, определённую интегрированным содержанием всех предыдущих и текущей итерации. Результат финальной итерации содержит всю требуемую функциональность продукта. Таким образом, с завершением каждой итерации продукт получает приращение — *инкремент* — к его возможностям, которые, следовательно, развиваются *эволюционно*. Итеративность, инкрементальность и эволюционность в данном случае есть выражение одного и то же смысла разными словами со слегка разных точек зрения[3].

Planning/Feedback Loops

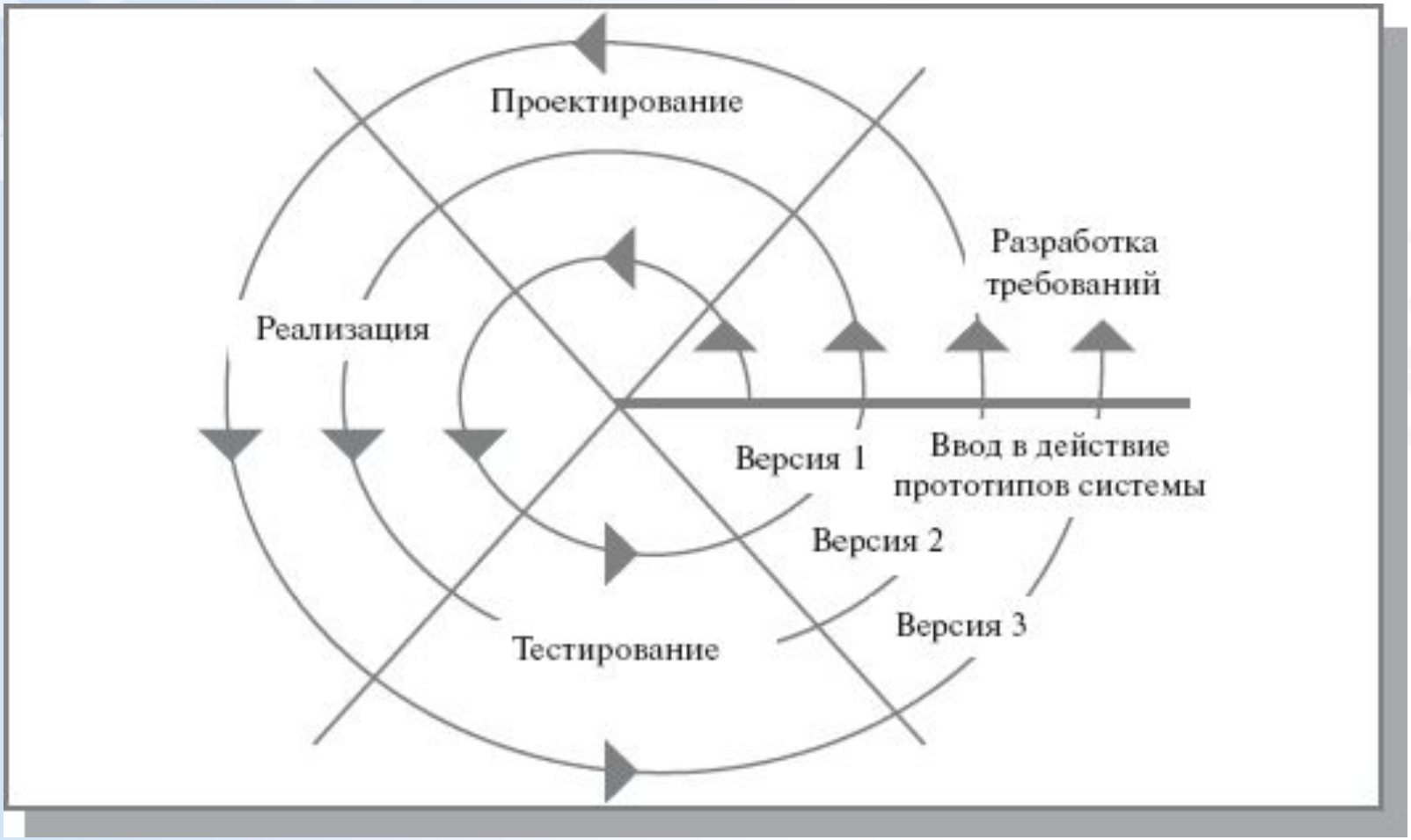


Спиральная модель

Спиральная модель была разработана в середине 1980-х годов Барри Боэмом. При использовании этой модели ПО создается в несколько итераций (витков спирали).

Каждая итерация соответствует созданию фрагмента или версии ПО, на ней уточняются цели и характеристики проекта, оценивается качество полученных результатов и планируются работы следующей итерации.

- На каждой итерации оцениваются:
- риск превышения сроков и стоимости проекта;
- необходимость выполнения ещё одной итерации;
- степень полноты и точности понимания требований к системе;
- целесообразность прекращения проекта.



Локальное программирование

Глобальное программирование

Локальное программирование

- **Проблемы системы типов**

При передачи значения одного типа данных с разным размером выделяемой памяти (int(2) -> double(8))

- **Проблемы с метаданными**

При использовании компиллировании программного кода помимо исполняемого когда они наполняются инструкциями по обработке

- **Проблемы выполнения**

Сложность с внедрением части кода написанного на другом языке в программу

Глобальное программирование

При работе с программными компонентами, написанными разными программистами и при помощи разных языков программирования в различных средах, и последующих попытках собрать эти компоненты в одну распределенную систему, программисту придется решить бесчисленное множество проблем глобального программирования.

Проблема именования (Naming)

Определение имен переменных разными разработчиками

Обработка ошибок (error handling)

При возникновении ошибок возвращаемый код не унифицирован

Безопасность (security)

Нет гарантий при передачи данных через Интернет

Контроль версий (versioning)

Несовместимость версий ПО

Масштабируемость (scalability)

Невозможность использования большого количества пользователей (Интернет)

Методологии программирования

Методология программирования — совокупность методов, применяемых на различных стадиях жизненного цикла программного обеспечения и имеющих общий философский подход.

Классификации:

- **Классификация по ядрам**
- **Классификация по топологической специфике**
- **Классификация по специфике реализации**

Классификация по ядрам

При подходе к методологии, как имеющей *ядро*, соответствующее способу описания алгоритма, и *дополнительные особенности*, можно выделить следующие пять основных ядер методологий:

- Методология императивного программирования
- Методология структурного программирования
- Методология ООП
- Методология функционального программирования
- Методология логическое программирование
- Методология программирования в ограничениях

Методология структурного программирования

Структу́рное программи́рование — методология разработки программного обеспечения, в основе которой лежит представление программы в виде иерархической структуры блоков. Предложена в 70-х годах XX века Э. Дейкстрой, разработана и дополнена Н. Виртом.

В соответствии с данной методологией любая программа представляет собой структуру, построенную из трёх типов базовых конструкций:

- **последовательное исполнение** — однократное выполнение операций в том порядке, в котором они записаны в тексте программы;
- **ветвление** — однократное выполнение одной из двух или более операций, в зависимости от выполнения некоторого заданного условия;
- **цикл** — многократное исполнение одной и той же операции до тех пор, пока выполняется некоторое заданное условие (условие продолжения цикла).

В программе базовые конструкции могут быть вложены друг в друга произвольным образом, но никаких других средств управления последовательностью выполнения операций не предусматривается.

Методология императивного программирования

- **Императи́вное программи́рование** — это парадигма программирования, которая описывает процесс вычисления в виде инструкций, изменяющих состояние программы. Императивная программа очень похожа на приказы, выражаемые повелительным наклонением в естественных языках, то есть это последовательность команд, которые должен выполнить компьютер.

Методология объектно-ориентированного программирования

В центре ООП находится понятие объекта. Объект — это сущность, которой можно посылать сообщения, и которая может на них реагировать, используя свои данные. Объект — это экземпляр класса. Данные объекта скрыты от остальной программы.

Методология функционального программирования

Функциона́льное программи́рование — раздел дискретной математики и парадигма программирования, в которой процесс вычисления трактуется как вычисление значений функций в математическом понимании последних.

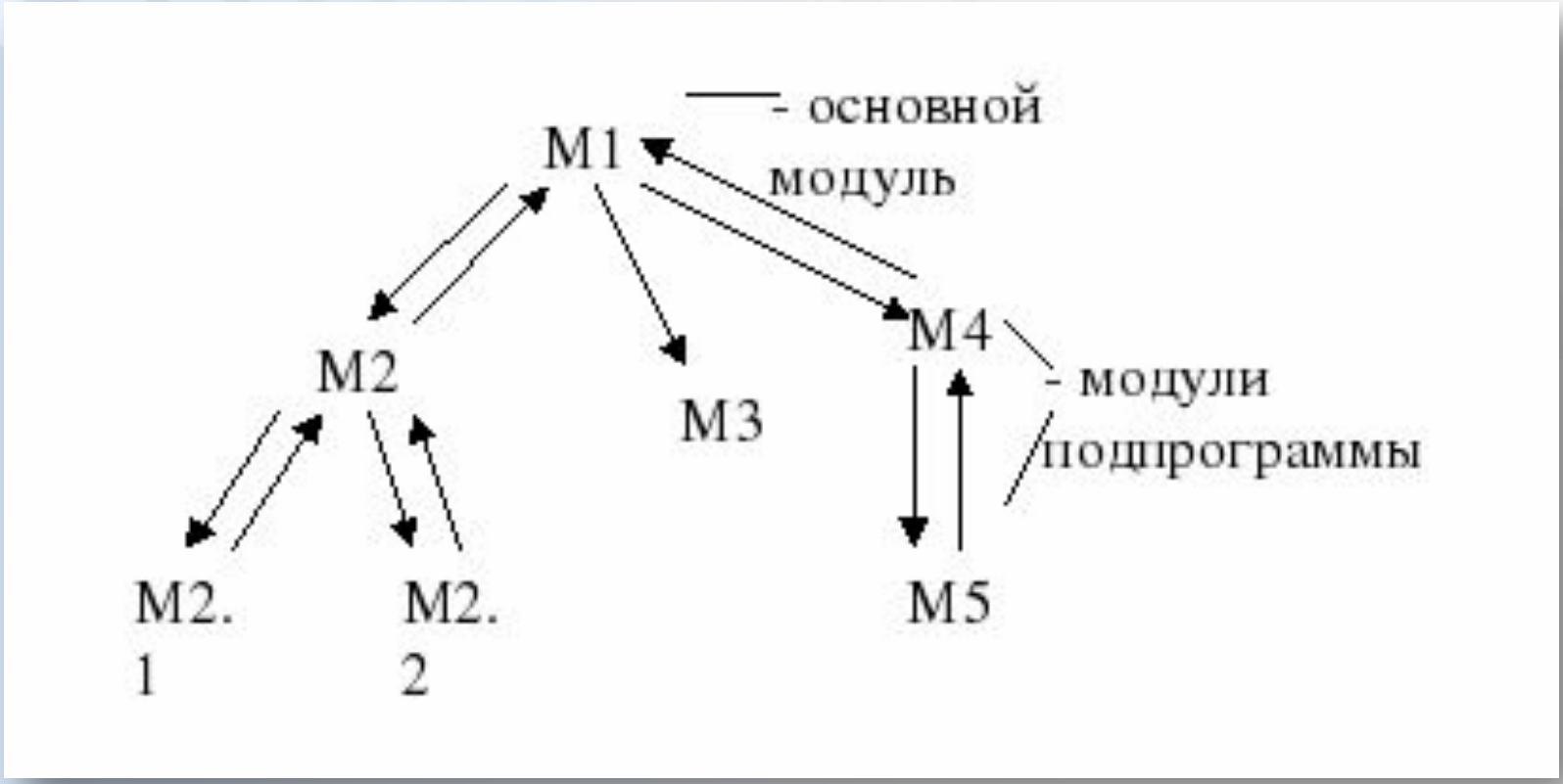
Методология логического программирования

Логи́ческое программи́рование — парадигма программирования, основанная на автоматическом доказательстве теорем, а также раздел дискретной математики, изучающий принципы логического вывода информации на основе заданных фактов и правил вывода. Логическое программирование основано на теории и аппарате математической логики с использованием математических принципов резолюций.

Методология программирования в ограничениях

Программирование в ограничениях (или *программирование ограничениями*) является парадигмой программирования, в которой отношения между переменными указаны в форме ограничений. Ограничения определяют не последовательность шагов для исполнения, а свойства искомого решения. Ограничения, которые используются в программировании в ограничениях, бывают различных видов: те, которые используются в задачах удовлетворения ограничений (например, «A или B истинно»), те, которые решаются симплекс-алгоритмом (например, «$x \leq 5$») и другие. Ограничения, как правило, встроены в язык программирования или осуществляются через отдельные программные библиотеки.

Структурное программирование.



Пошаговая детализация и нисходящее проектирование

Технология нисходящего проектирования с пошаговой детализацией является неотъемлемой частью создания хорошо структурированных программ. При написании программы с использованием этой технологии вся задача рассматривается как единственное предложение (вершина), выражающее общее назначение программы. Так как вершина редко отображает достаточное количество деталей, на основании которых можно написать программу, то поэтому надо начинать процесс детализации. Вершина разделяется на ряд более мелких задач в том порядке, в котором эти задачи должны выполнятся. В результате получим первую детализацию. Далее каждая из подзадач разбивается на подзадачи, принадлежащие второму уровню детализации. Программист завершает процесс нисходящей разработки с пошаговой детализацией, когда алгоритм настолько детализирован, чтобы его можно было бы преобразовать в программу.

Пример пошаговой детализации

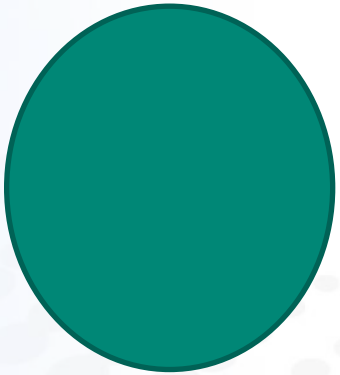
Объектно-ориентированное программирование

R1 – радиус круга;

X1, Y1 – координаты центра круга;

Color1 – цвет круга.

Draw1

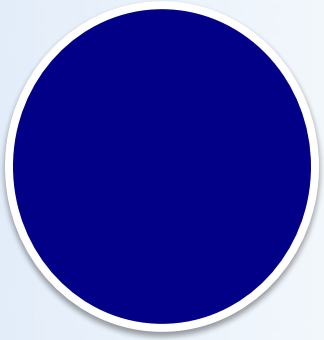


R2 – радиус круга;

X2, Y2 – координаты центра круга;

Color2 – цвет круга.

Draw2



R_n – радиус круга;

X_n, Y_n – координаты центра круга;

Color_n – цвет круга.

Draw_n

Абстракция

Абстрагирование — это способ выделить набор значимых характеристик объекта, исключая из рассмотрения незначимые. Соответственно, абстракция — это набор всех таких характеристик.

Инкапсуляция

Инкапсуляция — это свойство системы, позволяющее объединить данные и методы, работающие с ними в классе, и скрыть детали реализации от пользователя.

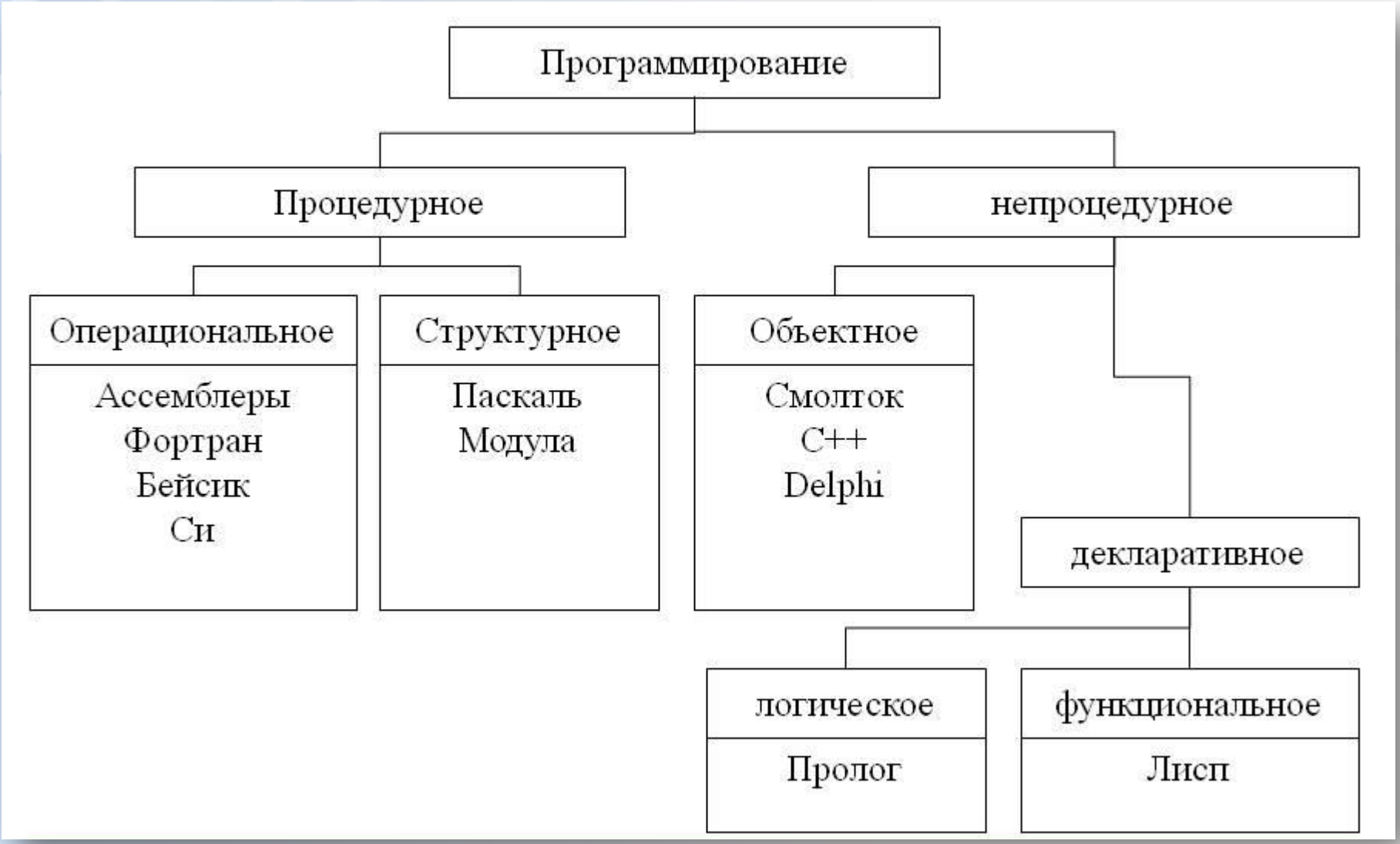
Наследование

Наследование — это свойство системы, позволяющее описать новый класс на основе уже существующего с частично или полностью заимствующейся функциональностью. Класс, от которого производится наследование, называется базовым, родительским или суперклассом. Новый класс — потомком, наследником или производным классом.

Полиморфизм

Полиморфизм — это свойство системы использовать объекты с одинаковым интерфейсом без информации о типе и внутренней структуре объекта.

Классификация языков программирования



Вопросы

- Какие модели жизненного цикла существуют? В чем их отличие?
- Какие методологии разработки программного обеспечения существуют?
- На какие группы разделяются языки программирования?
- В чем отличительные особенности объектно-ориентированного программирования?

Выводы

В рамках данного модуля были получены базовые знания в области методологий разработки программного обеспечения. Обозначены основные модели жизненного цикла программного обеспечения. Выявлены методологии разработки программного обеспечения, их достоинства и недостатки. Классифицированы современные языки программирования.