Основы программирования и баз данных

Модуль 3.

МЕТОДОЛОГИИ И ЯЗЫКИ ПРОГРАММИРОВАНИЯ

- Стадии и этапы разработки программ. Проектирование. Реализация.
- Проблемы программирования;
- Методологии программирования. Классификация методологий программирования:
  - структурное
  - объектно-ориентированное
  - логическое
  - функциональное
  - программирование в ограничениях

Модуль 3.
МЕТОДОЛОГИИ И ЯЗЫКИ ПРОГРАММИРОВАНИЯ
(продолжение)

- 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. Проектирование. Реализация

- **Разработка программного обеспечения** (*software engineering*, *software development*) — это процесс, направленный на создание и поддержание работоспособности программного обеспечения.
- Разработка программного обеспечения имеет дело с проблемами стоимости и надёжности.
- Некоторые программы содержат миллионы строк исходного кода, которые, как ожидается, должны правильно исполняться в изменяющихся условиях.

Материал из Википедии — свободной энциклопедии

Трудоемкость разработки программ и программного обеспечения

Программа – завершенный продукт, пригодный для запуска своим автором на системе, на которой она была разработана.

Программный комплекс – набор взаимодействующих программ, согласованных по функциям и форматам и вместе составляющим полное средство для решения больших задач.

Для этого каждая программа должна:

- удовлетворять точно определенным интерфейсам,
- использовать заранее оговоренный бюджет ресурсов (объем памяти, устройства ввода/вывода, процессорное время),
- должна быть оттестирована во взаимодействии с прочими компонентами во всех возможных сочетаниях.

Компонент программного комплекса стоит, по крайней мере, в **3** раза дороже, чем автономная программа с теми же функциями.

Программный продукт – такая программа, что:

- любой человек может ее запускать, тестировать, исправлять и развивать,
- может использоваться в различных средах и со многими наборами данных.
- написана максимально обобщенно, тщательно оттестирована и обеспечена подробной документацией.

Такой продукт стоит, по меньшей мере, в **3** раза дороже, чем отлаженная программа с такой же функциональностью.

Системный программный продукт – цель большинства системных программных проектов – отличается от обычной программы во всех перечисленных выше отношениях.

И стоит, соответственно, в **9** раз дороже.

Фредерик Брукс
«Мифический человеко-месяц
или
как создаются программные системы»

Стадии и этапы разработки программ. Проектирование. Реализация (продолжение)

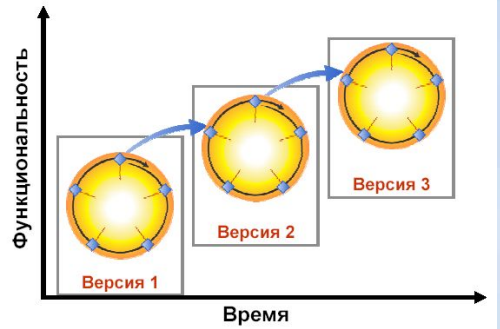
- Процесс разработки состоит из множества видов деятельности:
  - Сбор и анализ требований
  - Спецификация
  - Проектирование
  - Кодирование
  - Тестирование
  - Документирование
  - Сопровождение
- В различных моделях разработки их порядок или состав изменяются.

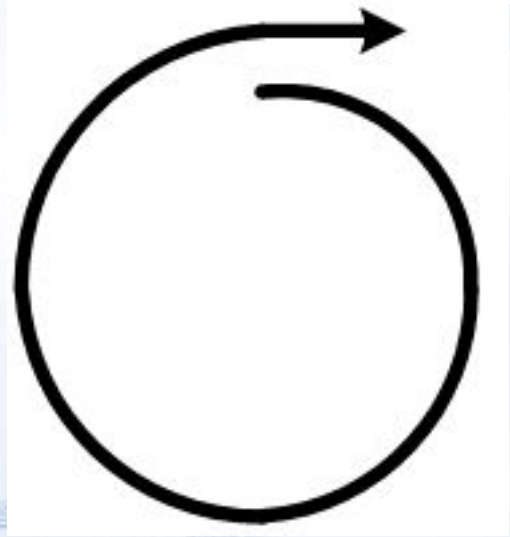
Стадии и этапы разработки программ. Проектирование. Реализация (продолжение)

- Модель водопада:

Стадии и этапы разработки программ. Проектирование. Реализация (продолжение)

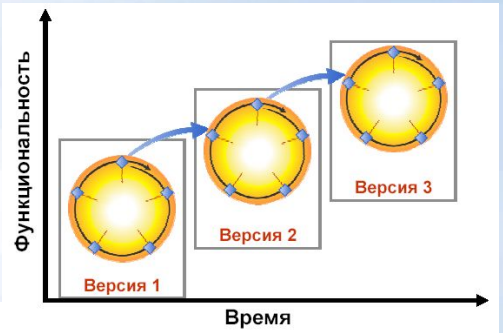
- Спиральная модель:





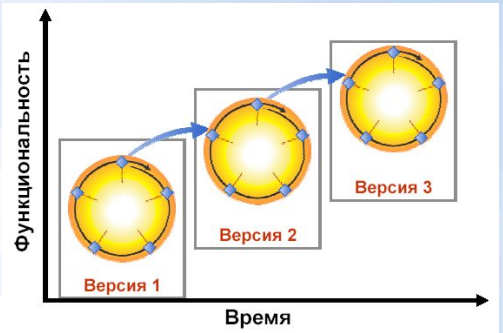
Стадии и этапы разработки программ. Проектирование. Реализация (продолжение)

- Итерационная модель:



Стадии и этапы разработки программ. Проектирование. Реализация (продолжение)

- Модель Microsoft Solution Framework:



Стадии и этапы разработки программ. Проектирование. Реализация (продолжение)

- **Стадии** разработки программного обеспечения как ***показатель степени готовности*** программного продукта и количества реализованных функций:

1. ***Альфа*** — Стадия добавления новых функциональных возможностей.

   Программы на данной стадии могут применяться только для ознакомления с будущими возможностями.

2. ***Бета*** — Стадия активного тестирования и отладки.

   Программы этого уровня могут быть использованы другими разработчиками программного обеспечения для испытания совместимости. Тем не менее программы этого этапа могут содержать достаточно большое количество ошибок.

3. ***Гамма*** — Стадия-кандидат на то, чтобы стать стабильной.

   Программы этой стадии прошли комплексное тестирование, благодаря чему были исправлены все найденные критические ошибки. Но в то же время, существует вероятность выявления еще некоторого числа ошибок, незамеченных при тестировании.

4. ***Стабильная версия — Релиз***

   Стабильная версия программы, прошедшая все предыдущие стадии, в которых исправлены основные ошибки, и готовая к применению.

Материал из Википедии — свободной энциклопедии

Проблемы программирования (продолжение)

- недостаточное взаимопонимание

Методологии программирования. Классификация методологий программирования

- **Парадигма программирования —** некоторый цельный набор методов и рекомендаций, определяющих стиль написания программ.
- Парадигма программирования представляет и определяет то, как программист видит выполнение программы.
- Например,
  - в объектно-ориентированном программировании программист рассматривает программу как набор взаимодействующих объектов,
  - в функциональном программировании программа представляется в виде цепочки вычисления функций.

Материал из Википедии — свободной энциклопедии

Методологии программирования. Классификация методологий программирования

- Примеры методологий программирования:
  - структурное программирование
  - объектно-ориентированное программирование
  - логическое программирование
  - функциональное программирование
  - программирование в ограничениях

Материал из Википедии — свободной энциклопедии

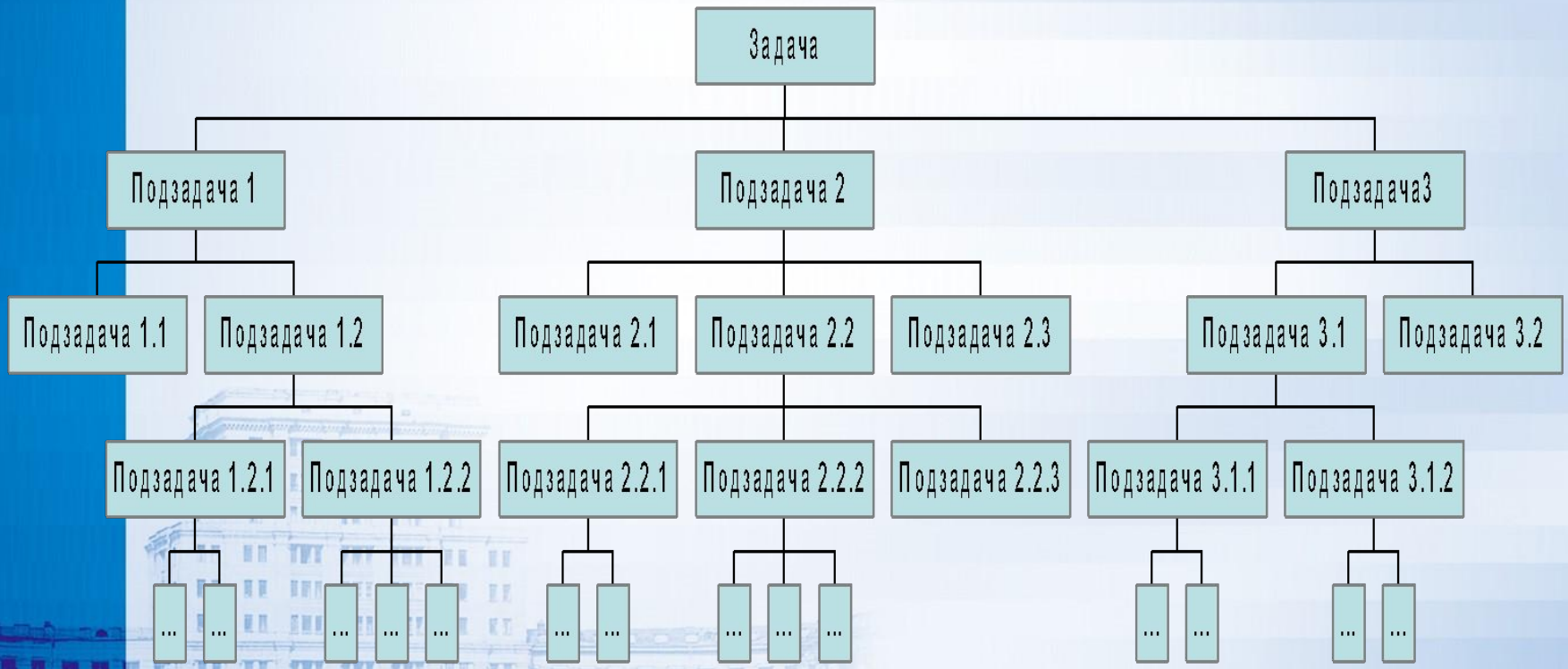
Структурное программирование.

- Базовые принципы
  - пошаговая детализация
  - модульная организация программы
  - типовые структуры управления процессом исполнения программы

Материал из Википедии — свободной энциклопедии

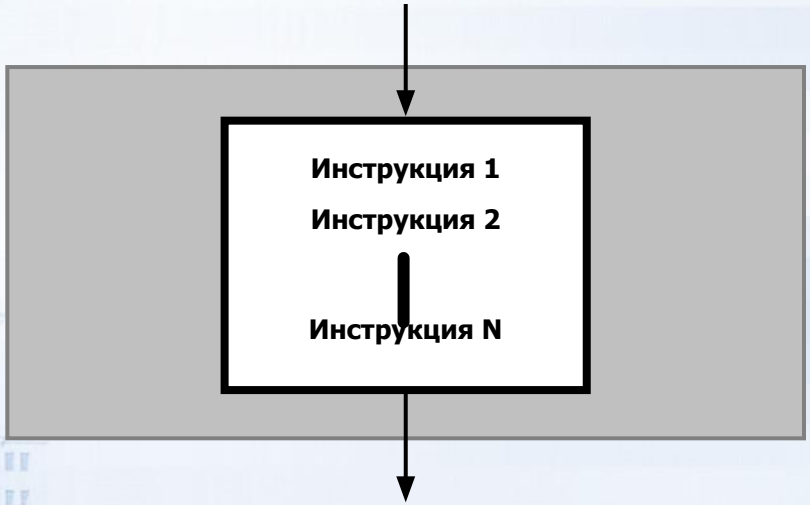
Структурное программирование (продолжение)

- Функциональная декомпозиция



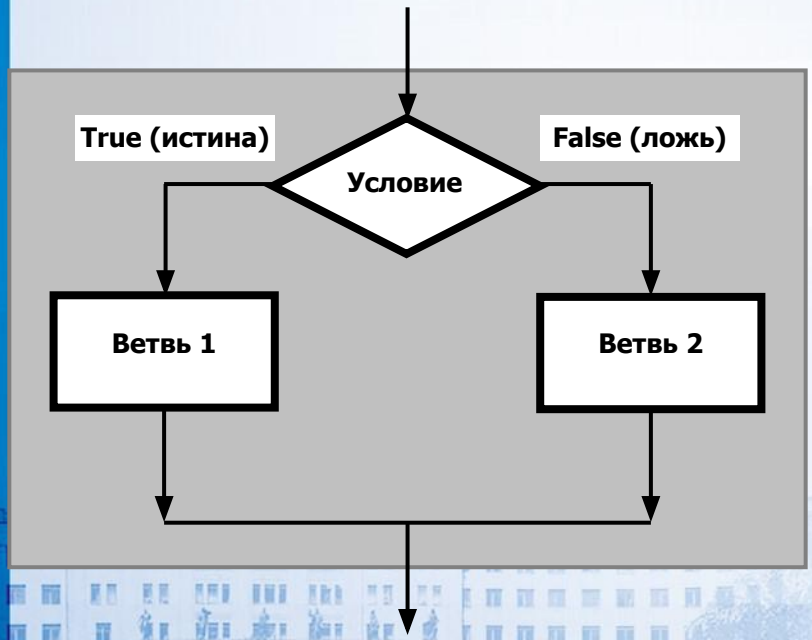
Структурное программирование (продолжение)

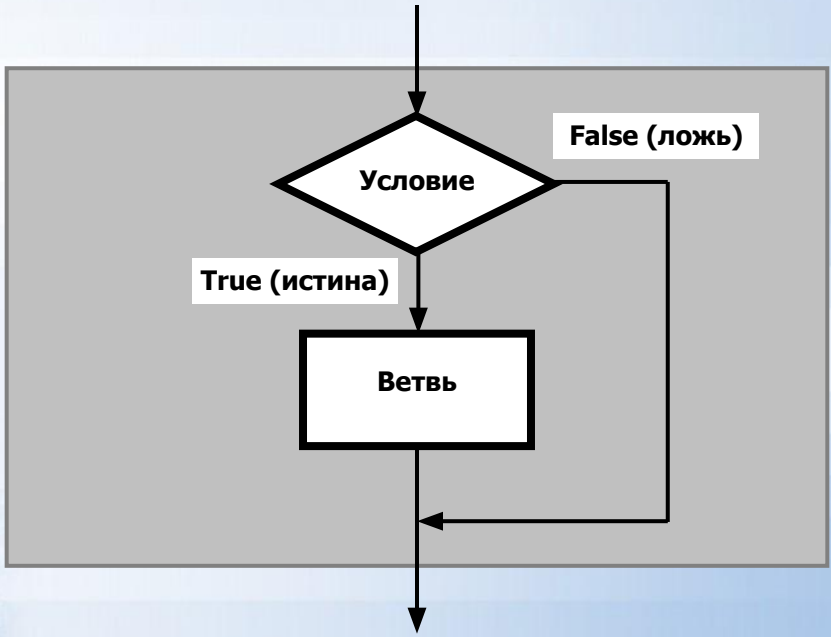
- Типовые структуры
  - **1. Следование (последовательность)**



Структурное программирование (продолжение)

- Типовые структуры
  - **2. Ветвление (выбор)**
  - **2а. Обход**

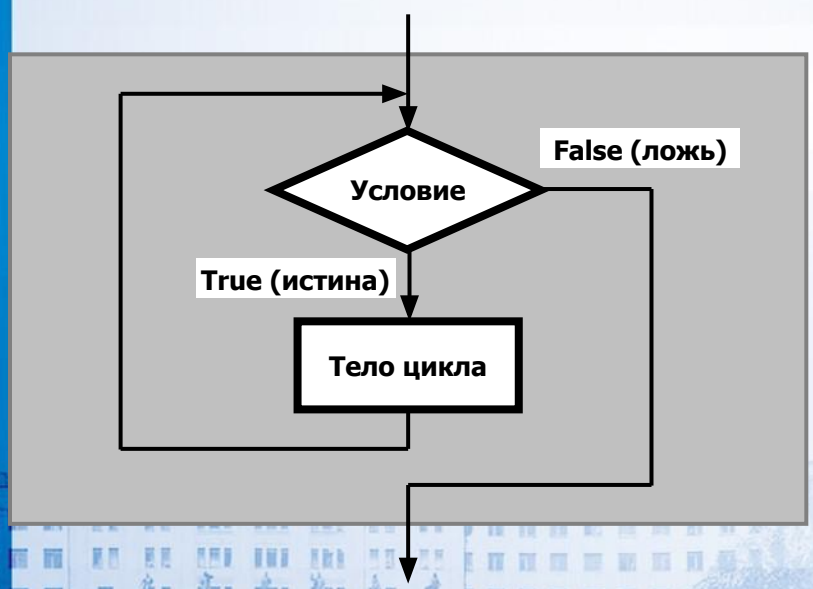


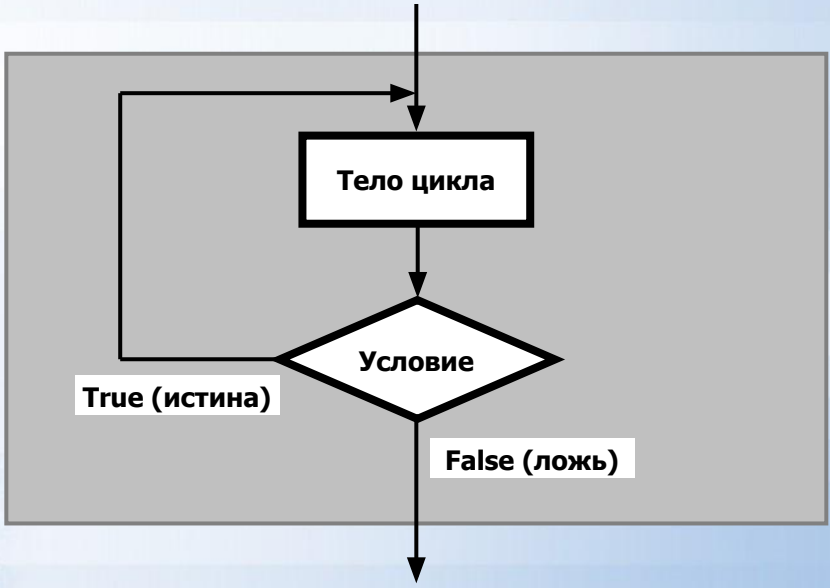


Структурное программирование (продолжение)

- Типовые структуры
  - **3. Циклы**

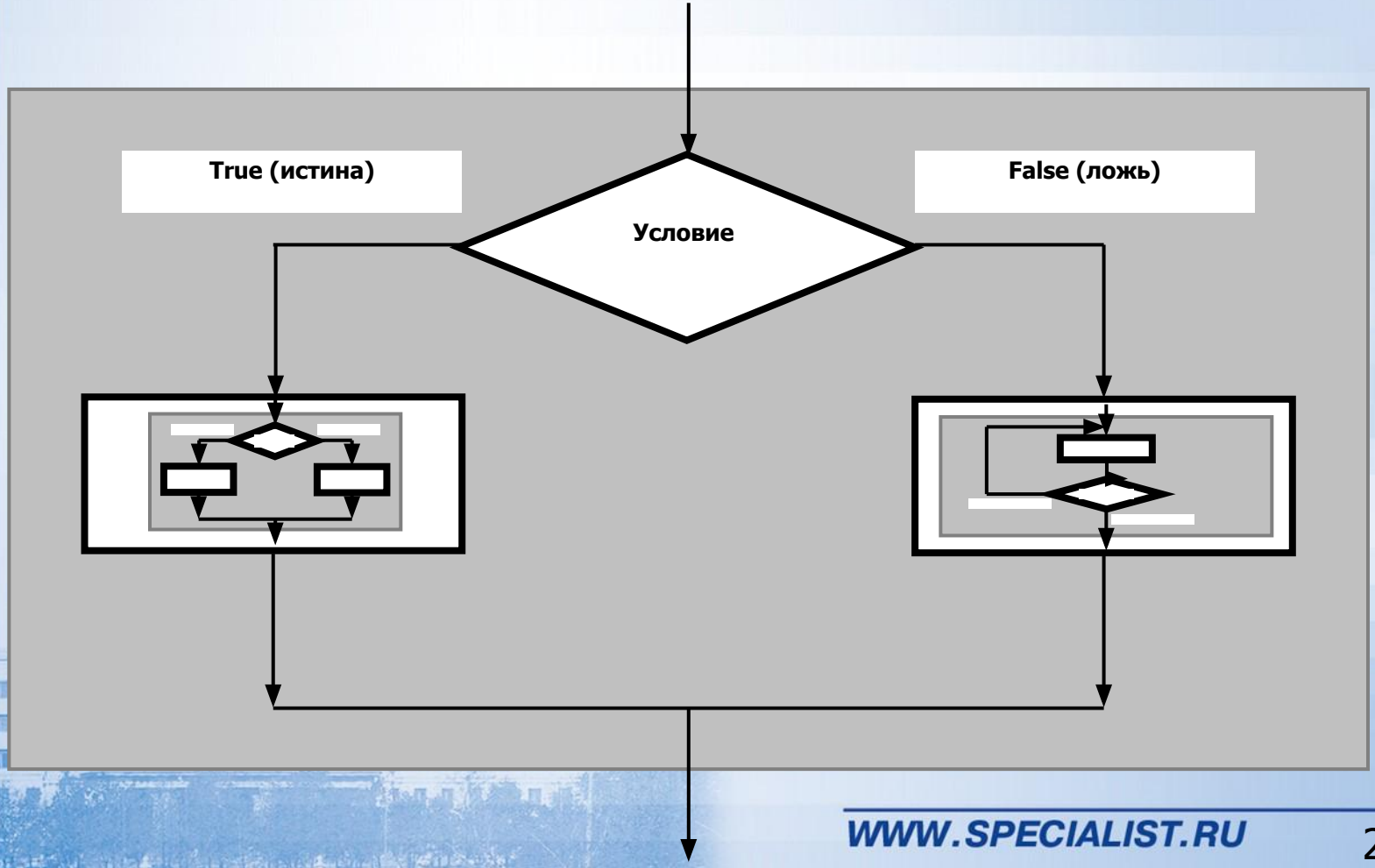
3а. С предусловием **3б. С постусловием**





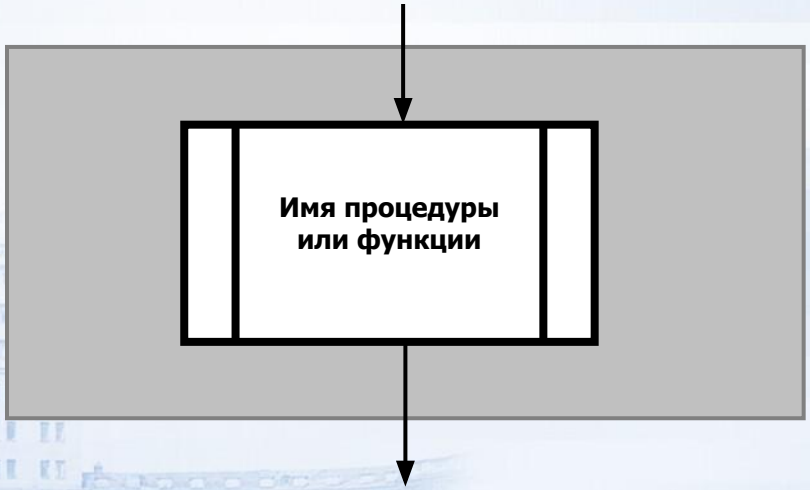
Структурное программирование (продолжение)

- Типовые структуры могут вкладываться друг в друга



Структурное программирование (продолжение)

- Типовые структуры
  - **4. Предопределенный процесс (процедура, функция)**



Объектно-ориентированное программирование

- **Объектно-ориентированное программирование** (*ООП*) — парадигма программирования, основанная на представлении предметной области в виде системы взаимосвязанных абстрактных объектов и их реализаций.
- Основной проблемой процедурного программирования является то, что данные и функции их обработки не были связаны.
- С появлением ООП появилась новая структура данных — **класс**. Это тип данных, внешне похожий на структуру (в языке С) или запись (в Pascal), в котором кроме данных (*свойств*) также содержатся функции их обработки (*методы*).

Материал из Википедии — свободной энциклопедии

Объектно-ориентированное программирование (продолжение)

- **Базовые понятия в ООП**:
  - Класс представляет собой *тип данных*, имеющий в составе:
    - **Свойства** - атрибуты объекта (параметры его состояния)
    - **Методы** - действия, которые можно выполнять над объектом или которые сам объект может выполнять.
  - Каждый объект является *экземпляром* некоторого класса объектов.
  - Один класс отличается от других именем и, обычно, набором поддерживаемых *интерфейсов* (доступных методов).
  - Интерфейсы, в свою очередь, представляют собою набор *сообщений*, которые можно посылать объекту.

Материал из Википедии — свободной энциклопедии

Объектно-ориентированное программирование (продолжение)

- **Базовые принципы**
  - **Абстракция данных**
    - ***Объекты*** представляют собою ***модели*** реальных сущностей из предметной области.
      - Работать с ними намного удобнее, чем с низкоуровневым описанием всех возможных свойств и реакций объекта.
  - **Инкапсуляция**
    - Любой ***класс*** должен рассматриваться как ***чёрный ящик***.
      - Пользователь класса должен видеть и использовать только интерфейс (от английского interface — внешнее лицо, т. е. список декларируемых свойств и методов) класса и может не вникать в его внутреннюю реализацию.

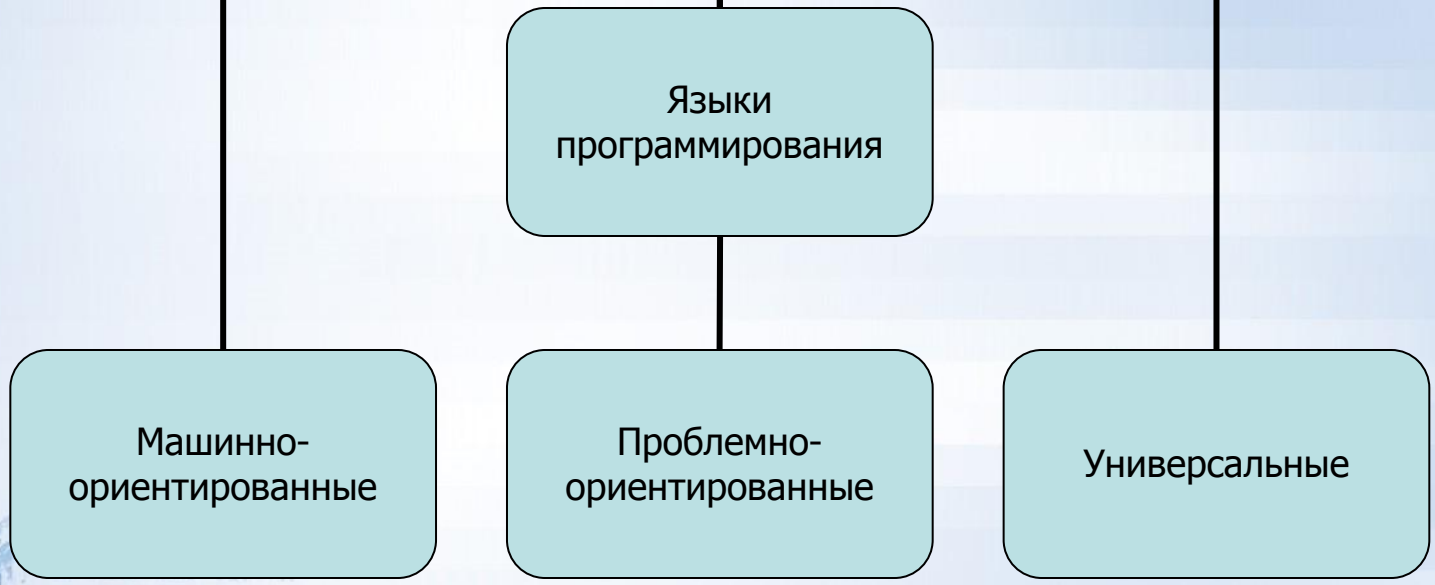
Материал из Википедии — свободной энциклопедии

Объектно-ориентированное программирование (продолжение)

- **Базовые принципы**
  - **Наследование**
    - Порождение одного класса от другого с сохранением всех свойств и методов класса-предка и добавлением, при необходимости, новых свойств и методов.
      - Наследование призвано отобразить такое свойство реального мира, как иерархичность.
  - **Полиморфизм**
    - Классы-потомки могут изменять реализацию унаследованных методов класса-предка, сохраняя неизменным его интерфейс.
      - Это позволяет обрабатывать объекты классов-потомков как однотипные объекты, не смотря на то, что реализация методов у них может различаться.

Материал из Википедии — свободной энциклопедии

Языки программирования. Классификация



Языки программирования. Классификация (продолжение)

- Список наиболее известных процедурных языков программирования
  - Fortran
  - Cobol
  - Basic
  - Pascal
  - C
  - Modula-2
  - Ada
  - Perl

Материал из Википедии — свободной энциклопедии

Языки программирования. Классификация (продолжение)

- Список наиболее известных объектно-ориентированных языков программирования:
  - Smalltalk
  - C++
  - Object Pascal (Delphi)
  - Java
  - C#
  - Python
  - PHP
  - VB.NET

Материал из Википедии — свободной энциклопедии

Проблемы программирования

- сложность современных задач:
  - сетевые многокомпонентные клиент-серверные и многоуровневые межплатформенные приложения, взаимодействующие с базами данных, со многими пользователями и с другими приложениями
- сложность отладки (устранения ошибок)
- необходимость сопровождения и модернизации
- сокращение времени на разработку
- безопасность