
# Основы программирования и баз данных

В.Г.Тетерин – Microsoft Solution Developer (Visual C++)

## Модуль 3.
# МЕТОДОЛОГИИ И ЯЗЫКИ ПРОГРАММИРОВАНИЯ

## 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

## 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.

# 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.

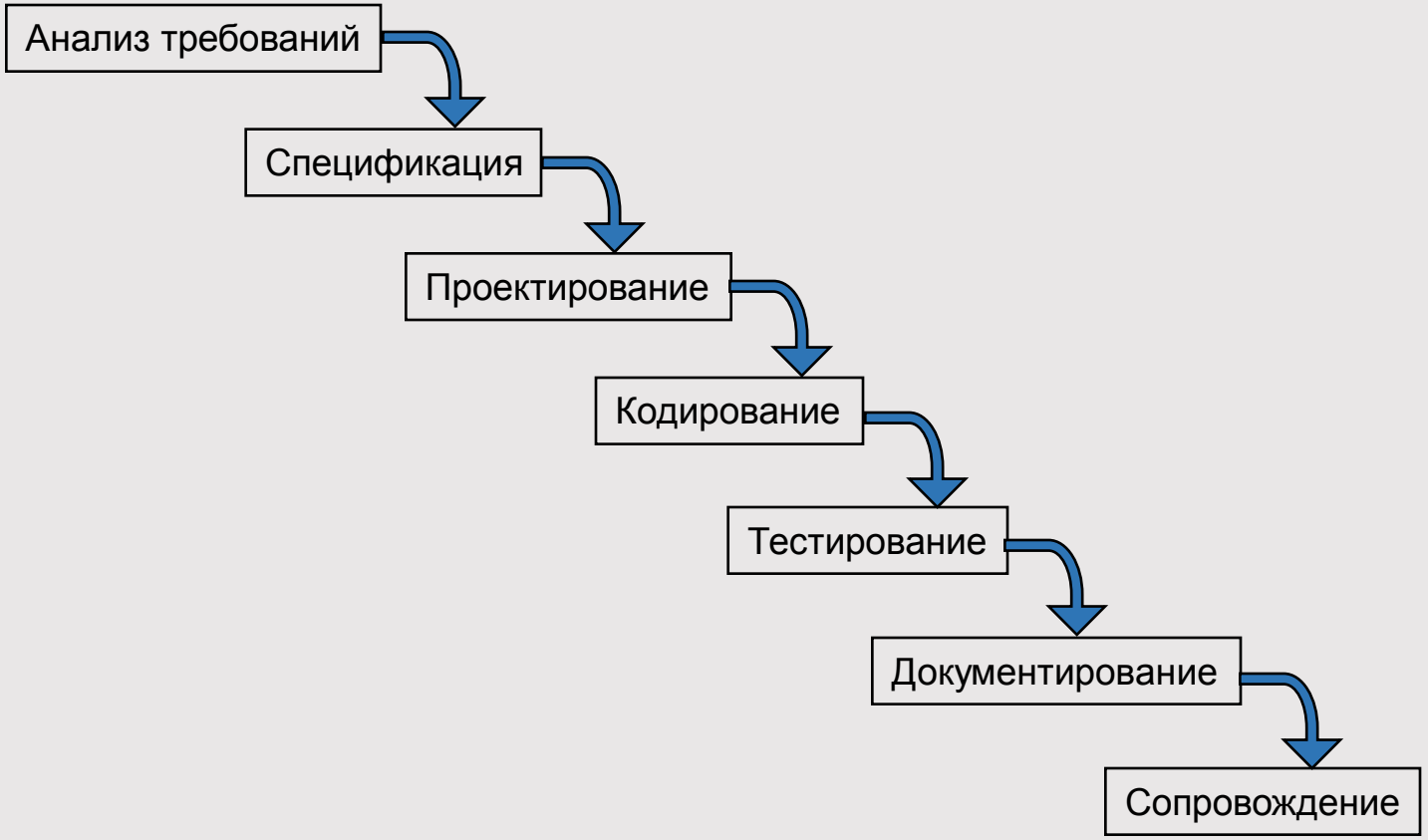
**Фредерик Брукс**
**«Мифический человеко-месяц**
**или**
**как создаются программные системы»**

# 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.

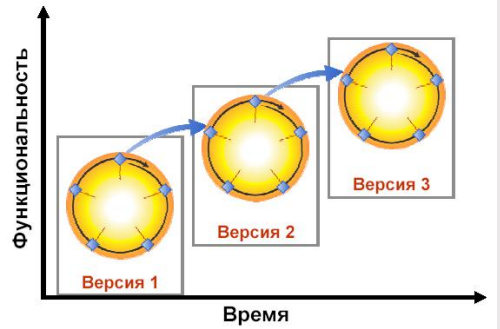
# Стадии и этапы разработки программ. Проектирование. Реализация (продолжение)

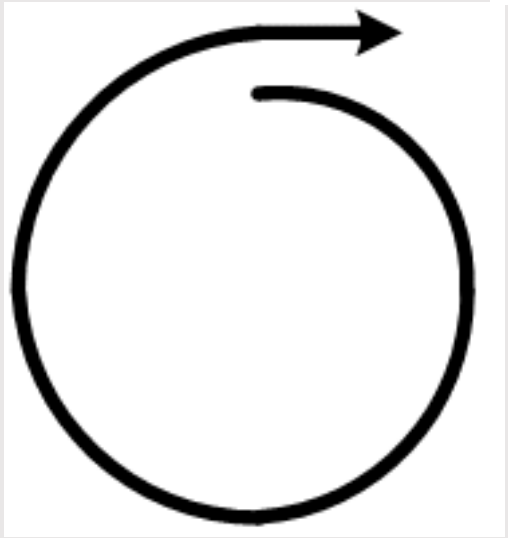
- Модель водопада:



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

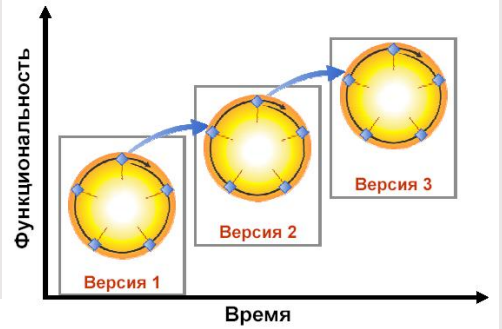
- Спиральная модель:



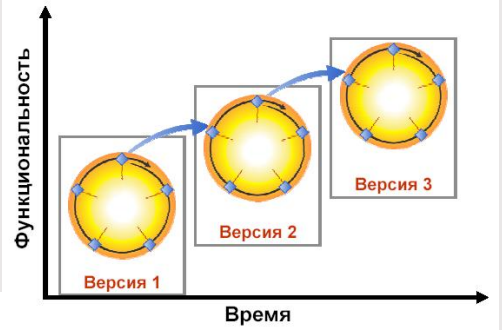
# Стадии и этапы разработки программ. Проектирование. Реализация (продолжение)

- Итерационная модель:




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

- Модель Microsoft Solution Framework:




# 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

# Проблемы программирования (продолжение)

- недостаточное взаимопонимание с заказчиком






# 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.

# Структурное программирование.

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

Материал из Википедии — свободной энциклопедии

# Структурное программирование (продолжение)

- **Структурное программирование** - методология разработки программного обеспечения, предложенная в 70-х года XX века *Э. Дейкстрой* и разработанная и дополненная *Н. Виртом*.

- **История**

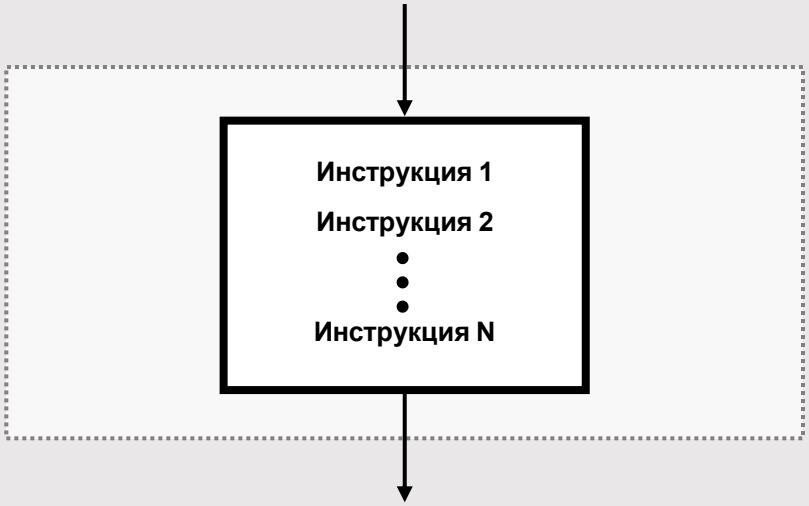
- Методология структурного программирования появилась как *следствие* возрастания сложности решаемых на компьютерах задач и соответственного *усложнения программного обеспечения*.

  - В 70-е годы XX века объёмы и сложность программ достигли такого уровня, что "интуитивная" разработка программ, которая была нормой в более раннее время, перестала удовлетворять потребностям практики.

  - Программы становились слишком сложными, чтобы их можно было нормально сопровождать, поэтому потребовалась какая-то формализация процесса разработки и структуры программ.

# Структурное программирование (продолжение)

- Типовые структуры управления
- В соответствии с данной методологией любая программа представляет собой структуру, построенную из ***трёх типов базовых конструкций***:
- **1. Последовательное исполнение**
  - однократное выполнение операций в том порядке, в котором они записаны в тексте программы;

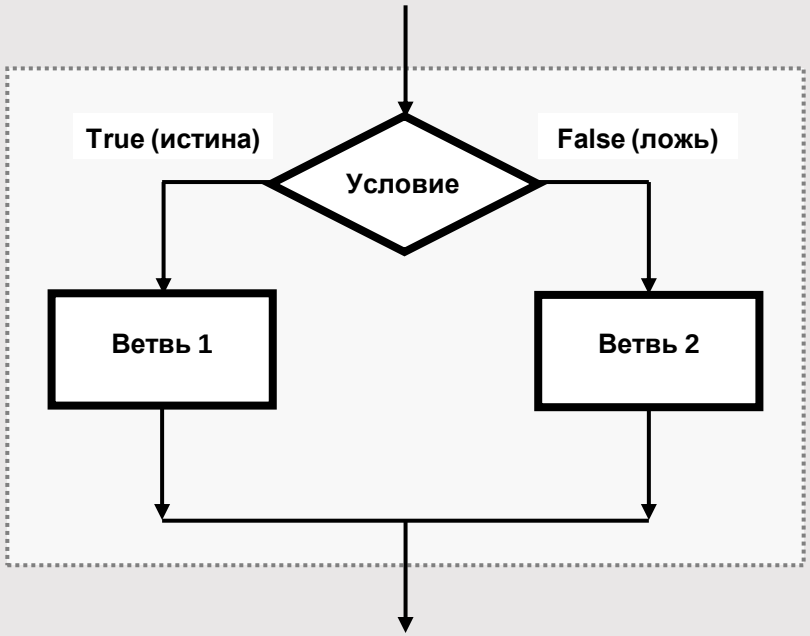


# Структурное программирование (продолжение)

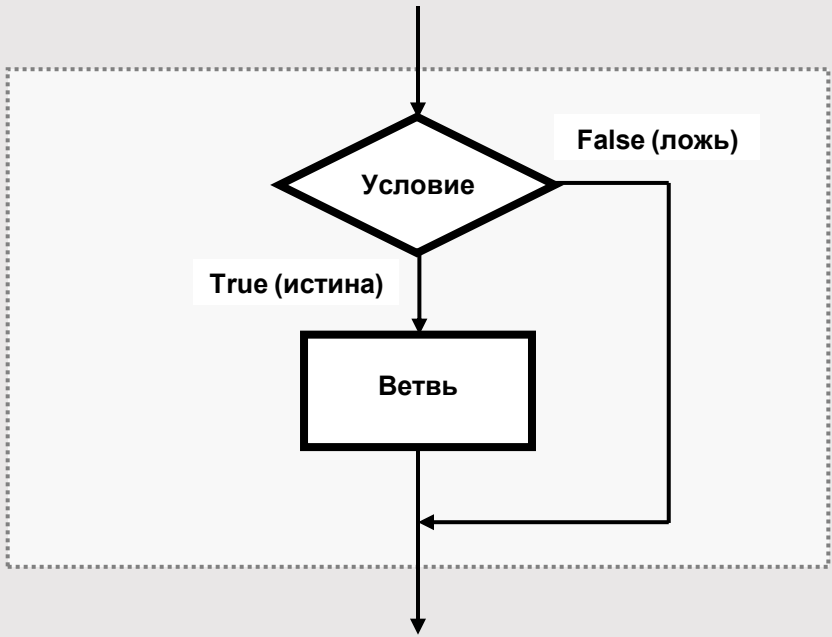
## 2. Ветвление

- однократное выполнение одной из двух или более операций, в зависимости от выполнения некоторого заданного условия;

**Ветвление (выбор)**



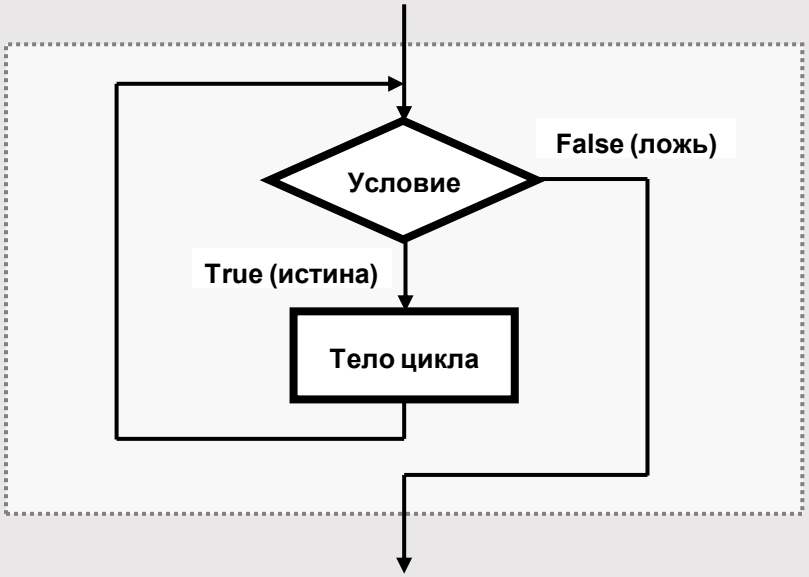

**Обход**



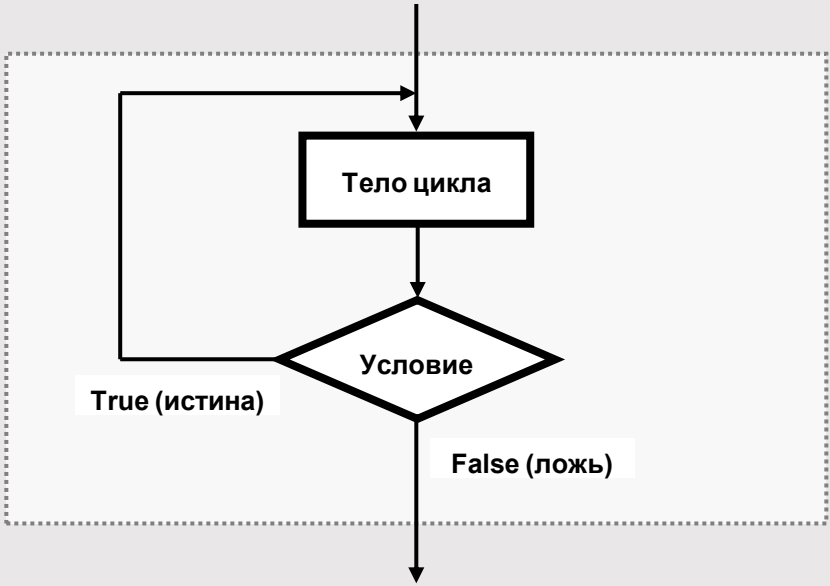
# Структурное программирование (продолжение)

- **3. Цикл**
  - многократное исполнение одной и той же операции до тех пор, пока выполняется некоторое заданное условие (условие продолжения цикла).

## 3а. С предусловием

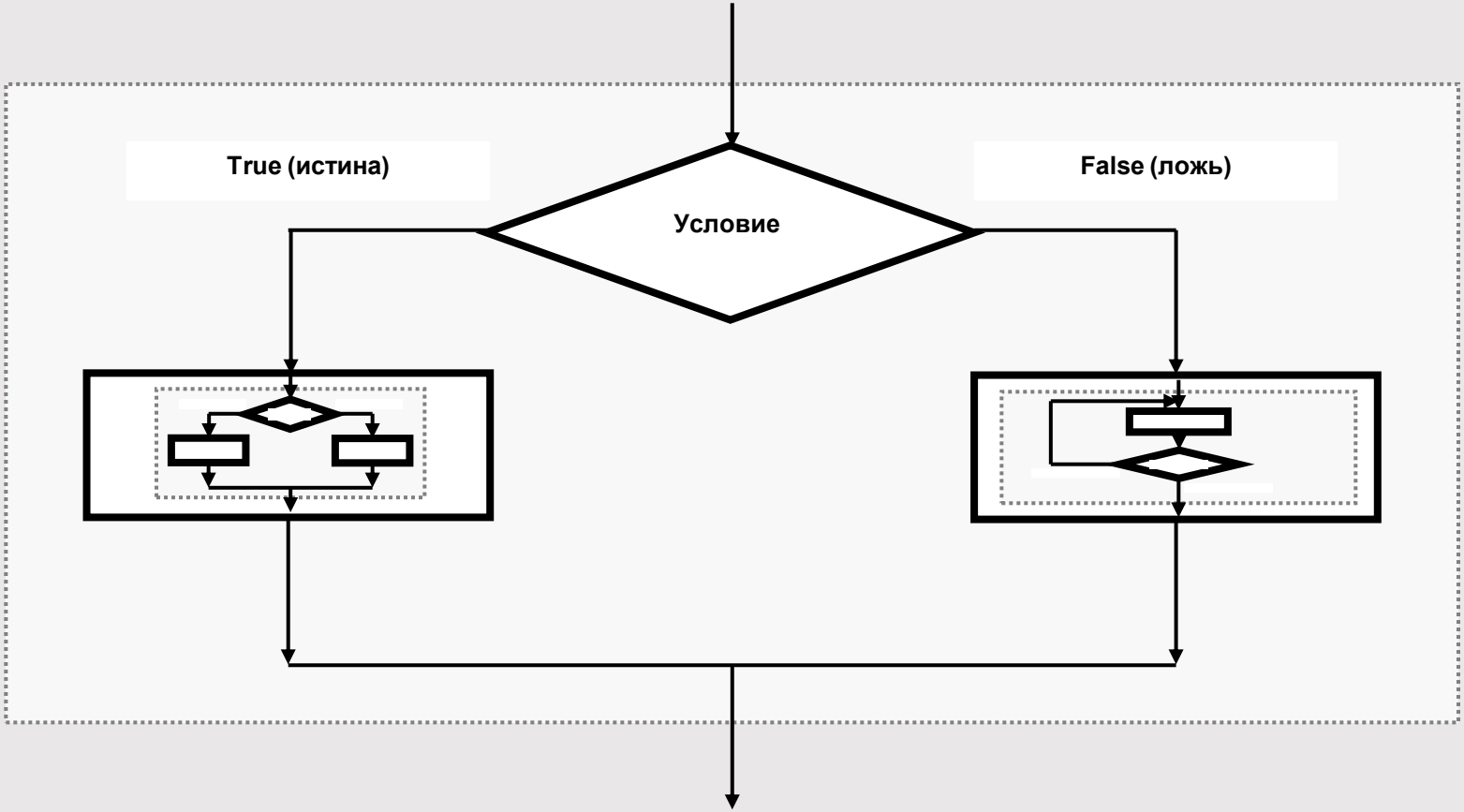



## 3б. С постусловием



# Структурное программирование (продолжение)

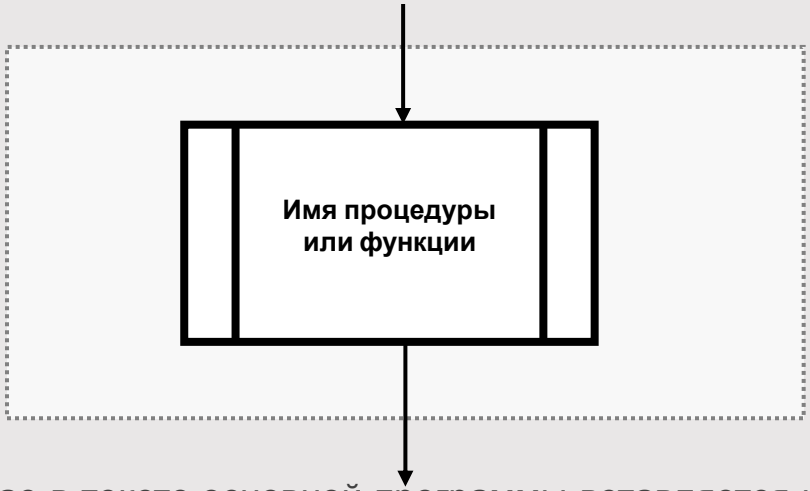
- В программе базовые конструкции могут быть вложены друг в друга произвольным образом,




- но никаких других средств управления последовательностью выполнения операций не предусматривается.

# Структурное программирование (продолжение)

- Повторяющиеся фрагменты программы, либо представляющие собой логически целостные вычислительные блоки, могут оформляться в виде так называемых **подпрограмм** (процедур или функций).
  - **4. Предопределенный процесс (процедура, функция)**




  - В этом случае в тексте основной программы вставляется инструкция **вызова подпрограммы**.
  - При исполнении такой инструкции выполняется вызванная подпрограмма, после чего происходит возврат к инструкции, следующей за командой вызова подпрограммы.

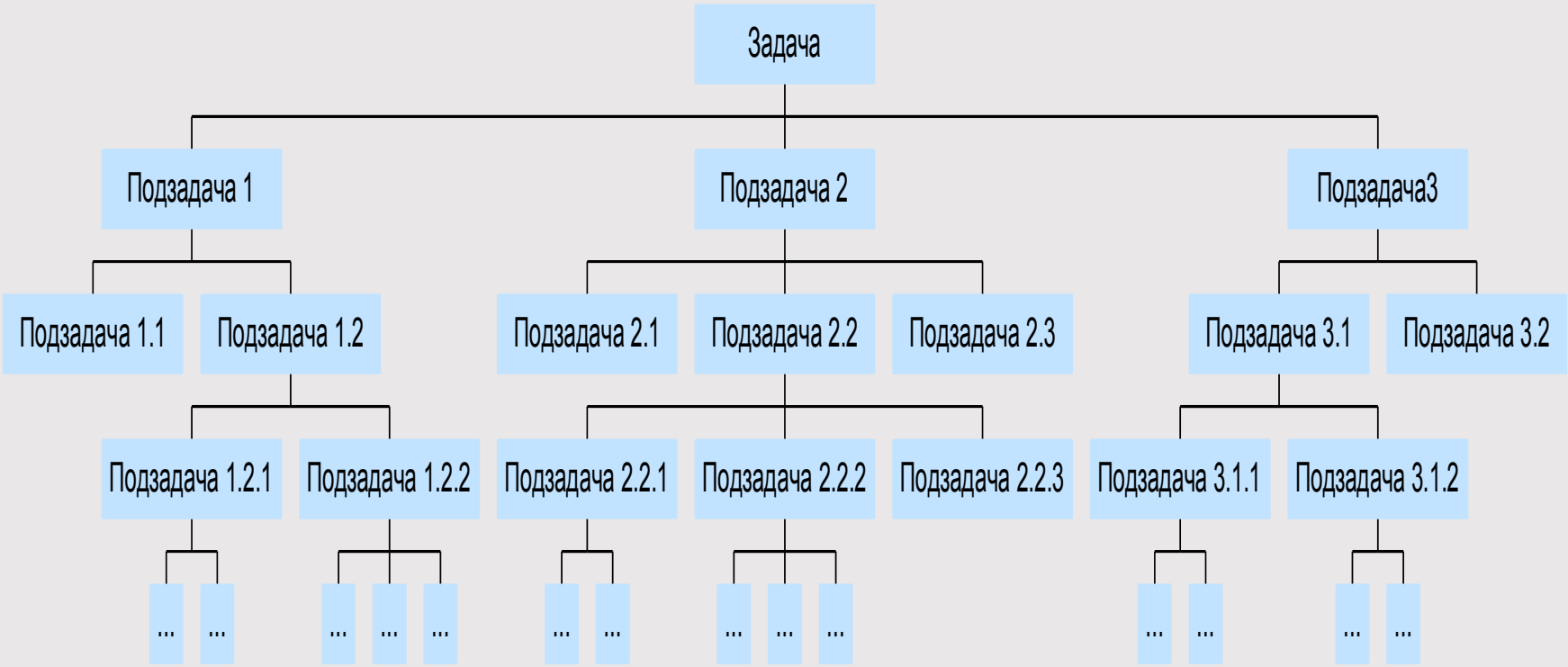
# Структурное программирование (продолжение)

## Пошаговая детализация

- Разработка программы ведётся пошагово, методом "сверху вниз".
- Сначала пишется текст основной программы, в котором вместо каждого связного логического фрагмента текста вставляется вызов подпрограммы, которая будет выполнять этот фрагмент.
  - Вместо настоящих, работающих подпрограмм, в программу вставляются "заглушки", которые ничего не делают. Полученная программа проверяется и отлаживается.
- После того, как программист убедится, что подпрограммы вызываются в правильной последовательности (то есть общая структура программы верна), подпрограммы-"заглушки" последовательно заменяются на реально работающие,
  - причём разработка каждой подпрограммы ведётся тем же методом, что и основной программы.
- Разработка заканчивается тогда, когда не останется ни одной "заглушки", которая не была бы удалена.
  - Такая последовательность гарантирует, что на каждом этапе разработки программист одновременно имеет дело с обозримым и понятным ему множеством фрагментов и может быть уверен, что общая структура всех более высоких уровней программы верна.
- При сопровождении и внесении изменений в программу выясняется, в какие именно процедуры нужно внести изменения, и они вносятся, не затрагивая непосредственно не связанные с ними части программы.
  - Это позволяет гарантировать, что при внесении изменений и исправлении ошибок не выйдет из строя какая-то часть программы, находящаяся в данный момент вне зоны внимания программиста.

# Структурное программирование (продолжение)

- Функциональная декомпозиция



# Структурное программирование (продолжение)

- Методология структурной разработки программного обеспечения была признана "***самой сильной формализацией 70-х годов***".

## Достоинства структурного программирования:

1. Структурное программирование позволяет значительно **сократить число вариантов** построения программы по одной и той же спецификации, что значительно **снижает сложность программы**.
2. В структурированных программах **логически связанные операторы находятся визуально ближе**, а **слабо связанные дальше**, что позволяет обходиться без блок схем и других графических форм изображения алгоритмов.
3. Упрощается процесс **тестирования** и отладки структурированных программ.

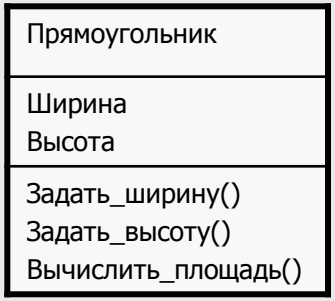
# Объектно-ориентированное программирование

- **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 C) или запись (в Pascal), в котором кроме данных (*свойств*) также содержатся функции их обработки (*методы*):

# 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.
  - описание класса

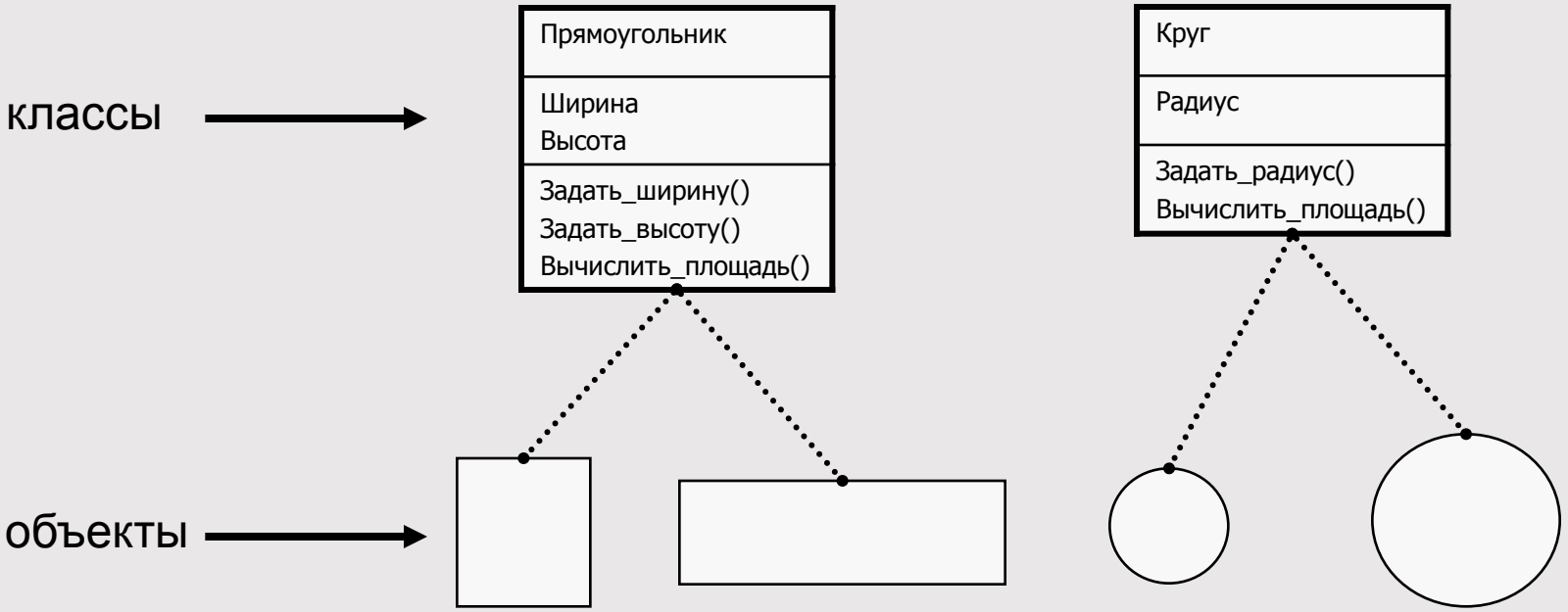

примеры классов




# Объектно-ориентированное программирование (продолжение)

## Базовые понятия в ООП:

- Каждый объект является *экземпляром* некоторого класса объектов.



# Объектно-ориентированное программирование (продолжение)

- **Базовые понятия в ООП**:
  - 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.


```
Прямоугольник A;

A . Задать_ширину(40);

A . Задать_высоту(20);

S = A . Вычислить_площадь();
```

# Объектно-ориентированное программирование (продолжение)

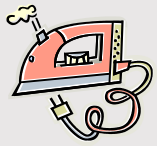
- **Базовые принципы**
  - **Абстракция данных -** Объекты представляют собою модели реальных сущностей из предметной области.
    - Работать с ними намного удобнее, чем с низкоуровневым описанием всех возможных свойств и реакций объекта.

| Человек |
| --- |
| Фамилия, имя, отчество<br>Год рождения<br>Пол |
| Сообщить_имя()<br>Слушать()<br>Говорить() |

| Прибор |
| --- |
| Напряжение<br>Мощность<br>Цена |
| Включить()<br>Выключить() |





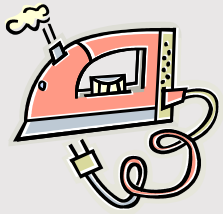



# Объектно-ориентированное программирование (продолжение)

## Базовые принципы

- **Инкапсуляция -** Любой класс должен рассматриваться как “чёрный ящик”.
  - Пользователь класса должен видеть и использовать только интерфейс класса (от английского ***interface*** — внешнее лицо, т. е. список декларируемых *свойств* и *методов*) и может не вникать в его внутреннюю реализацию.

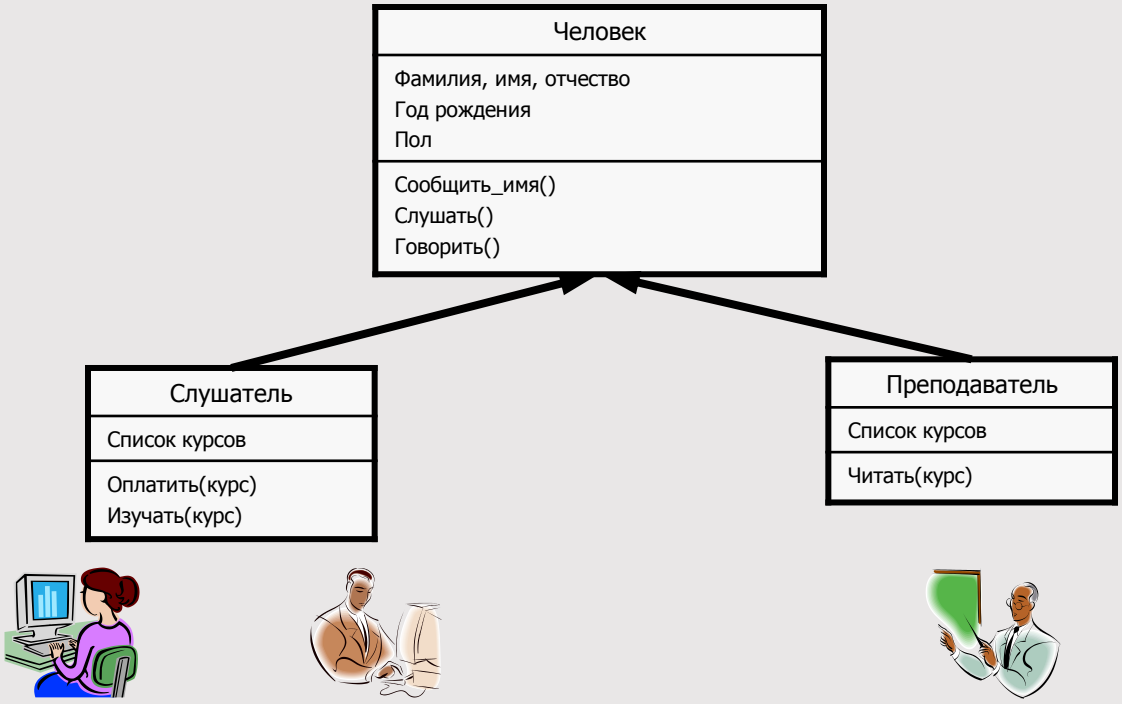
```
Прибор.Включить();
```



# 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.

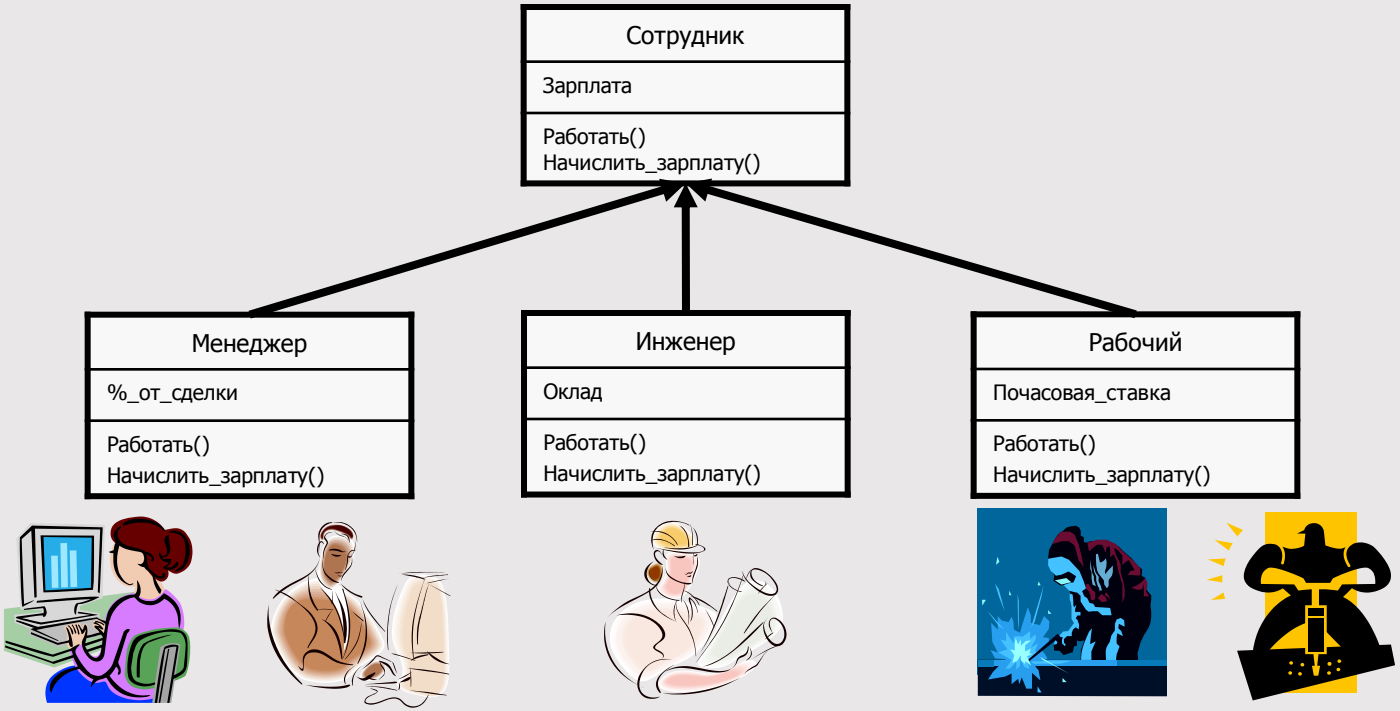



- Технически наследование обеспечивает повторное использование кода.

# Объектно-ориентированное программирование (продолжение)

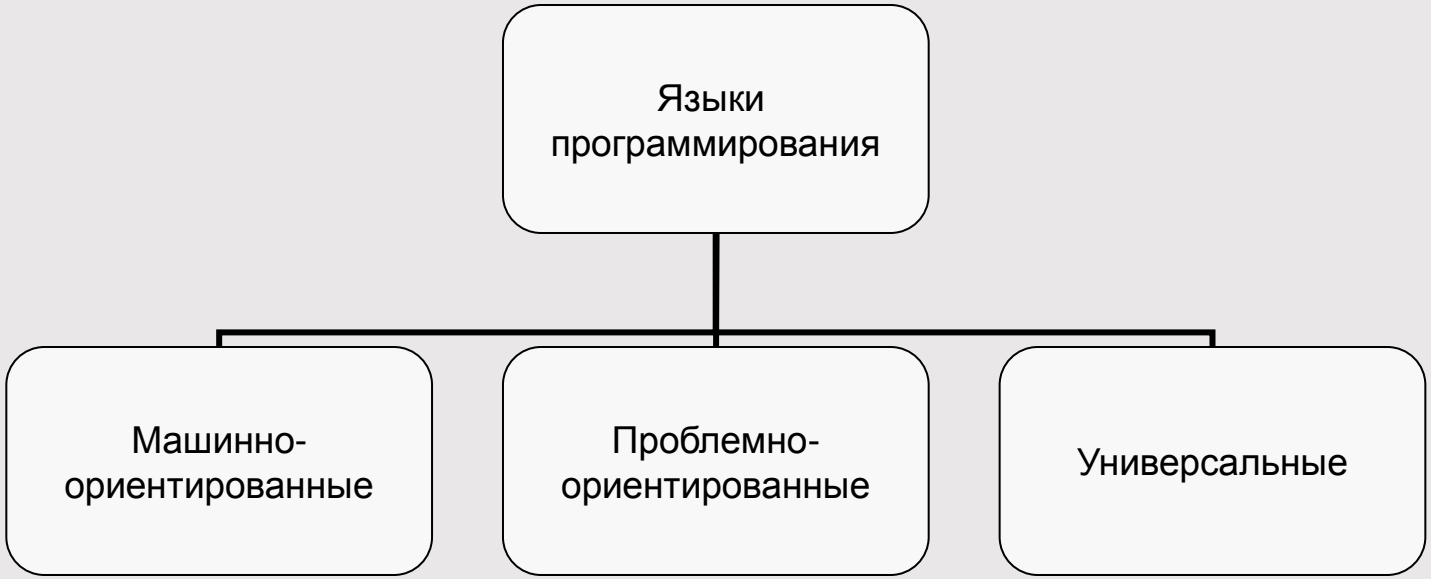
## Базовые принципы

- **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.

# 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#
  - VB.NET
  - PHP
  - Python
  - Ruby
  - Swift

**Материал из Википедии — свободной энциклопедии**

# Итоги

- В этом модуле Вы изучили:
  - **Стадии и этапы разработки программ**
  - **Методологии программирования:**
    - ***Структурное программирование*** и его базовые принципы:
      - пошаговая детализация, модульная организация программы, типовые структуры управления процессом исполнения программы
    - ***Объектно-ориентированное программирование*** и его базовые принципы:
      - абстрагирование, инкапсуляция, наследование, полиморфизм
  - **Классификацию** языков программирования


- В.Г.Тетерин – Microsoft Solution Developer (Visual C++)
- teterin@specialist.ru

# Вопросы?