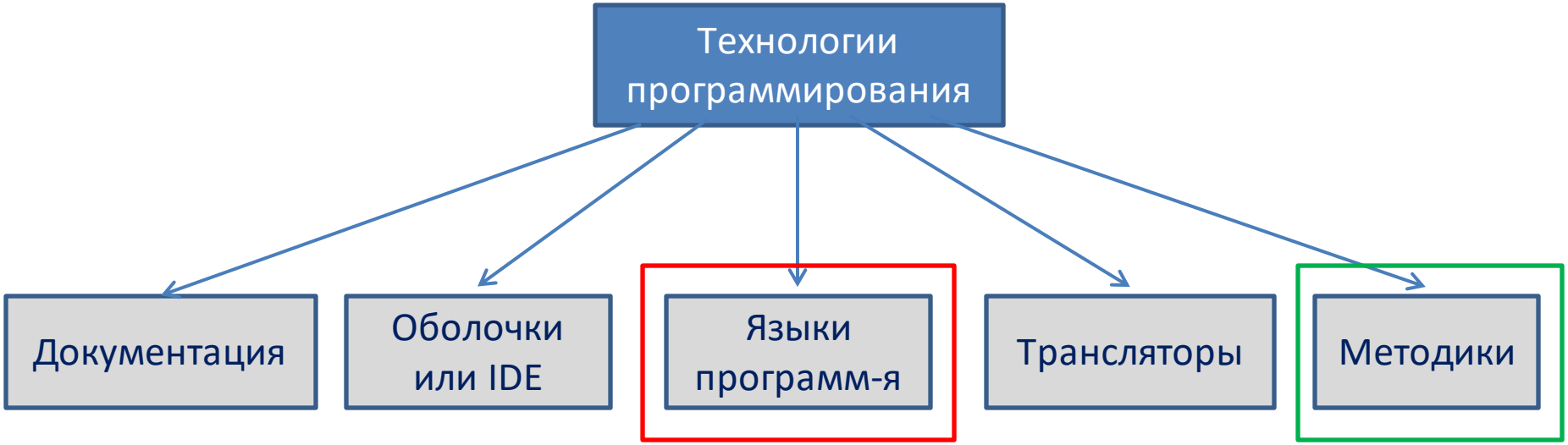
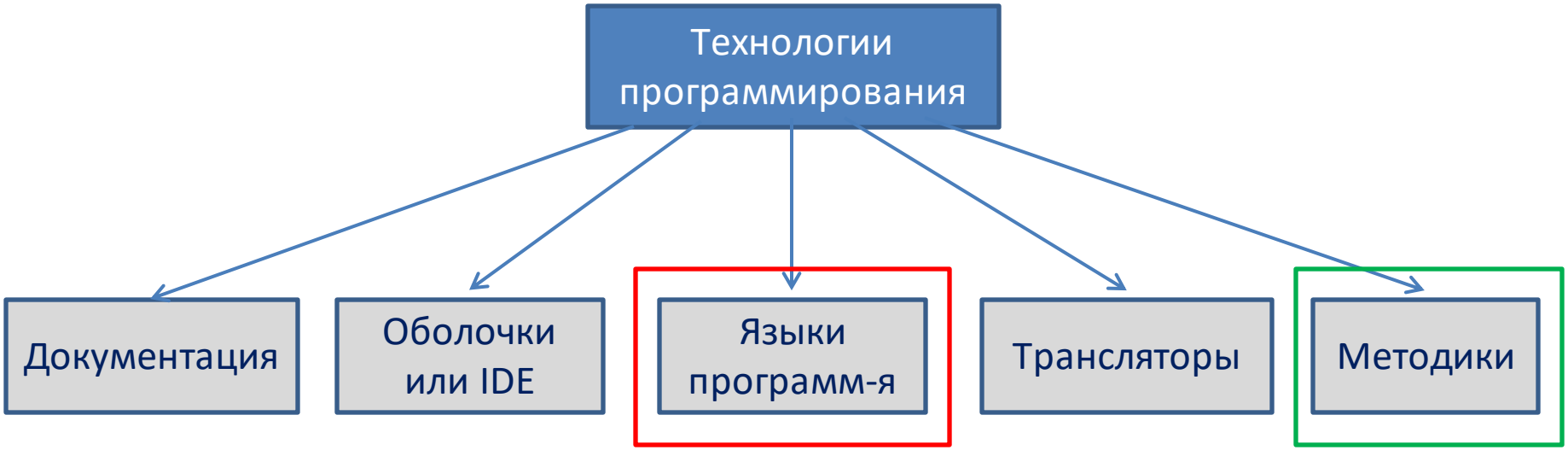
Лекция 4.

Технологии программирования – цели и этапы развития

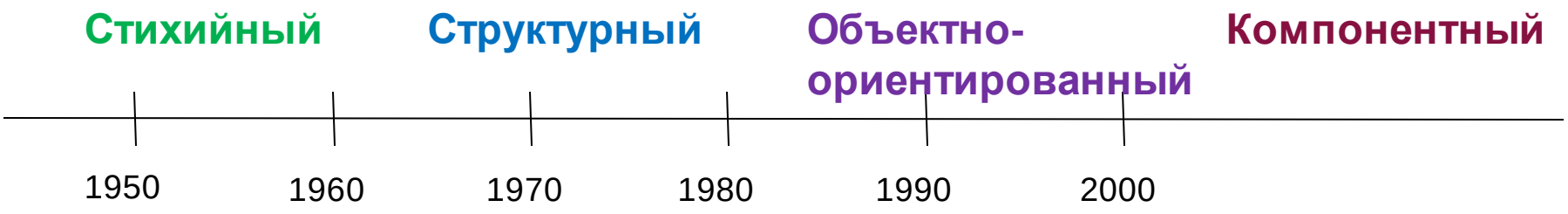

Технологии программирования
Документация
Оболочки или IDE
Языки программ-я
Трансляторы
Методики



Технология – это набор правил, методик и инструментов, позволяющих наладить <u>производственный процесс</u> выпуска продукта.

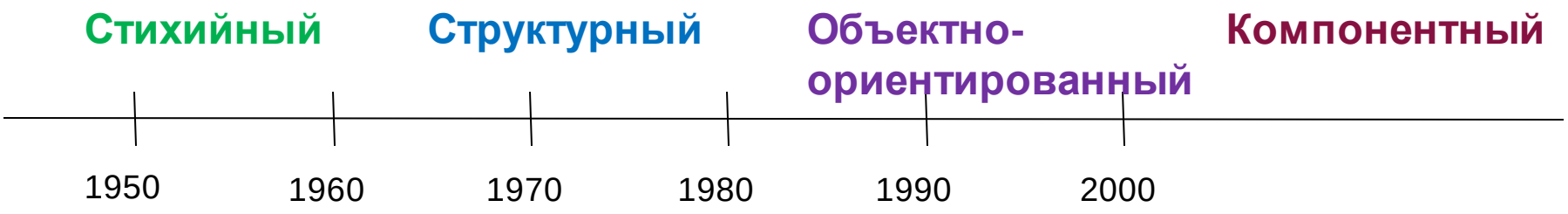
Технологии программирования

Методы, используемые на разных этапах разработки, составляют **подход к программированию**.



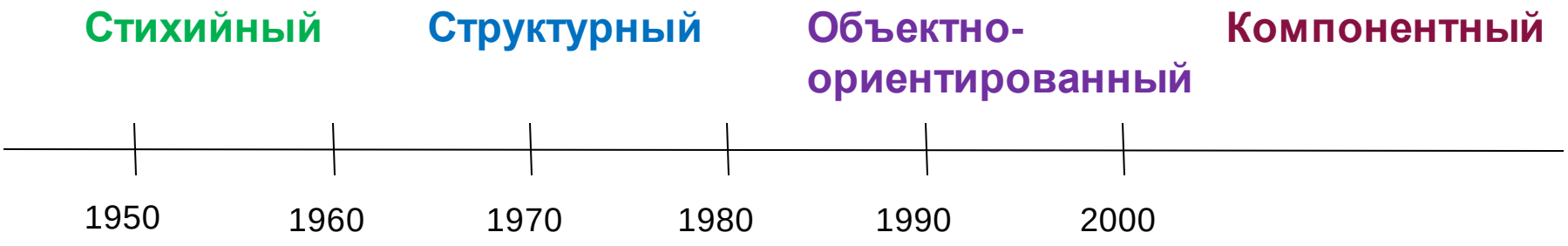
Технологии программирования

Методы, используемые на разных этапах разработки, составляют **подход к программированию**.



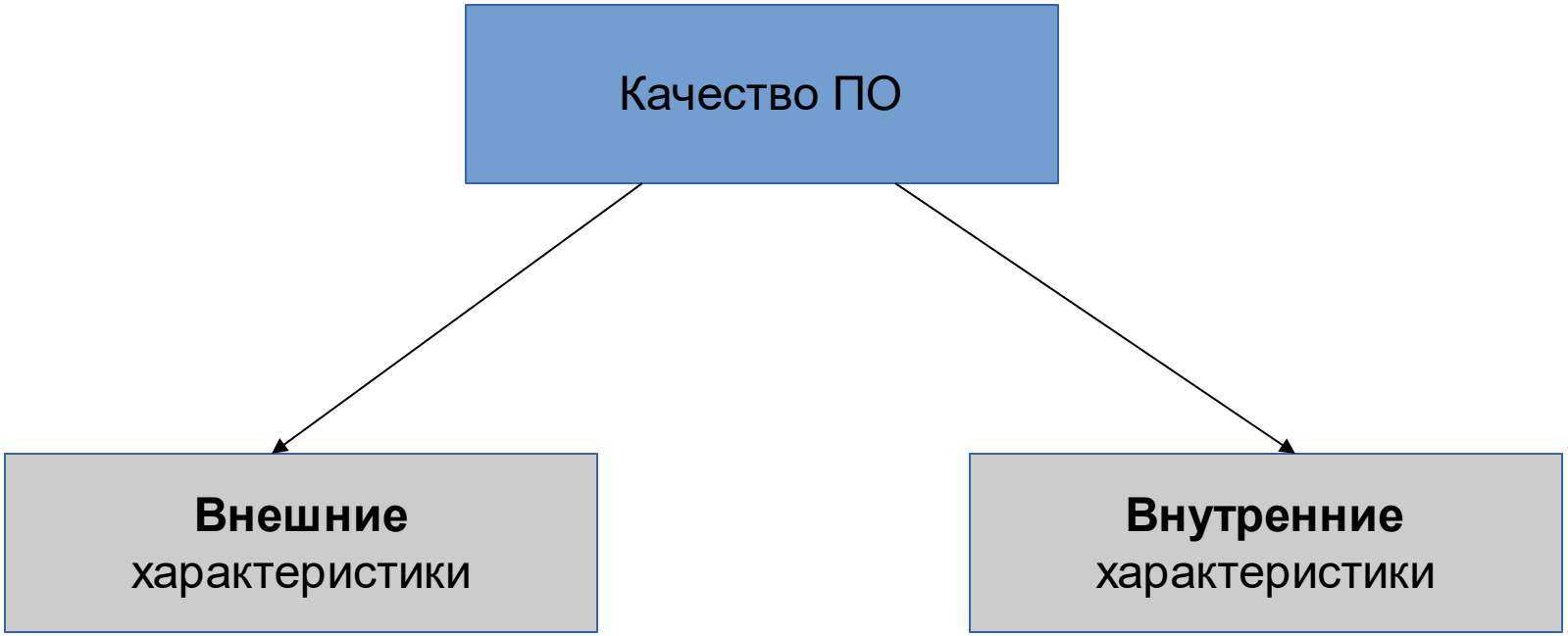
Какие общие цели и задачи у всех подходов?

Подходы к программированию



Цель:

Эффективное производство ПО требуемого качества



Для пользователя программы

Для разработчика программы

Внешние характеристики качества ПО – это свойства, которые осознает пользователь программы.

Внешние характеристики качества ПО – это свойства, которые осознает пользователь программы.

- **Корректность** — отсутствие/наличие дефектов в спецификации, проекте и реализации системы
- **Практичность** — легкость изучения и использования системы
- **Эффективность** — степень использования системных ресурсов (память, процессор)

Внешние характеристики качества ПО – это свойства, которые осознает пользователь программы.

- **Корректность**
- **Практичность**
- **Эффективность**
- **Надежность** — способность системы выполнять необходимые функции в предопределенных условиях
- **Целостность** — способность системы предотвращать неавторизованный или некорректный доступ к своим программам и данным.

Внешние характеристики качества ПО – это свойства, которые осознает пользователь программы.

- **Корректность**
- **Практичность**
- **Эффективность**
- **Надежность**
- **Целостность**

- **Адаптируемость** — возможность использования системы без ее изменения в тех областях или средах, на которые она не была ориентирована непосредственно
- **Живучесть** — способность системы продолжать работу при вводе недопустимых данных или в напряженных условиях.

SAP Interface Monitor

SAP Interface Monitor

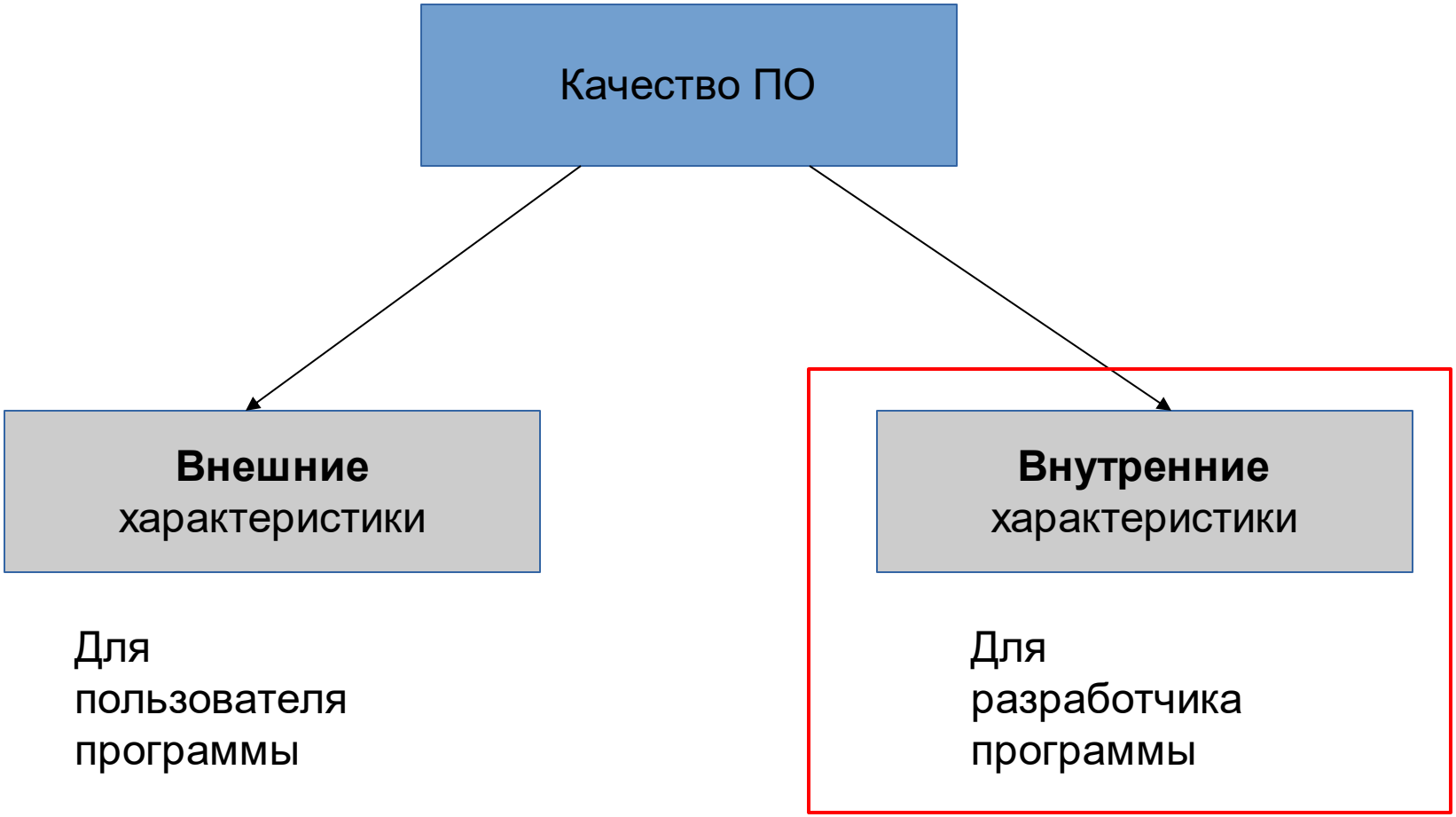
Summary	Descr.
Inbound	
DUMMY01	DUMMY In RFC Moni
Outbound	
DUMMY02	DUMMY Out RFC Moni

Interface	Proc.	Time	By	Ended	Log no.	Interface Description	Processed by	Σ Counter
DUMMY01	10.02.2019	20:02:22	DEVELOPER			DUMMY In RFC Moni	Jorge Sancho Royo	1
DUMMY02	10.02.2019	20:02:22	DEVELOPER			DUMMY Out RFC Moni	Jorge Sancho Royo	1
								2

DUMMY01 DUMMY In RFC Moni - 10.02.2019 20:02:22

Parameters	Descr.
Processing data	
INPUT	Input parameter
Returning data	
OUTPUT	

INPUT
dummy input string


Качество ПО
Внешние характеристики
Для пользователя программы
Внутренние характеристики
Для разработчика программы

Разработка программы

Написание начальной версии	**Изменения**

Внутренние характеристики качества ПО

Каждый дурак может написать программу, которую понятна компьютеру.

Хороший программист пишет программу, которая понятна человеку.

Мартин Фаулер

Внутренние характеристики качества ПО

- **Удобство сопровождения** — легкость изменения программной системы
- **Возможность повторного использования** частей системы в других системах
- **Легкость чтения** исходного кода системы
- **Понятность** и на уровне общей организации, и на детальном уровне отдельных операторов
- **Тестируемость**

Технологии программирования: эффективность

Для повышения эффективности разработки программ нужно упрощать процесс программирования.

Цель – снижение трудоемкости программирования

Технологии программирования: эффективность

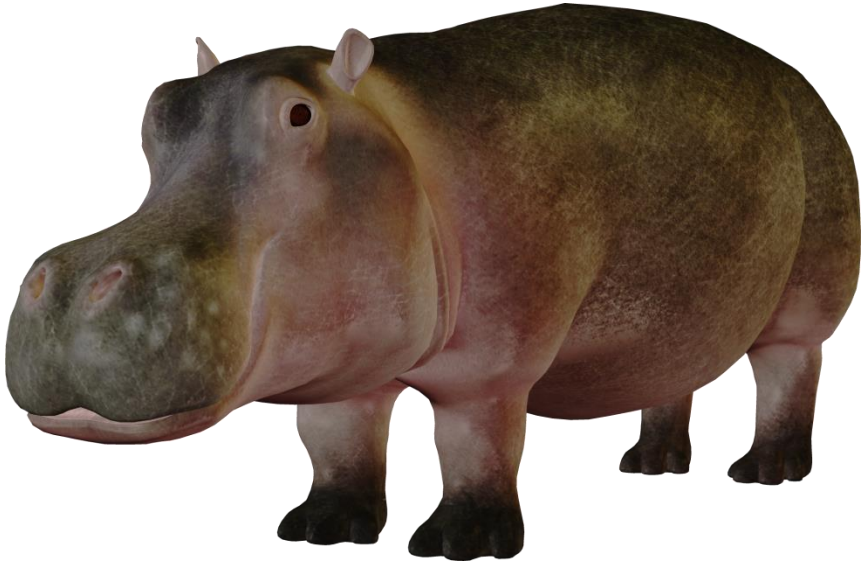
Для повышения эффективности разработки программ нужно упрощать процесс программирования.

Цель – снижение трудоемкости программирования.

Задачи для достижения цели:

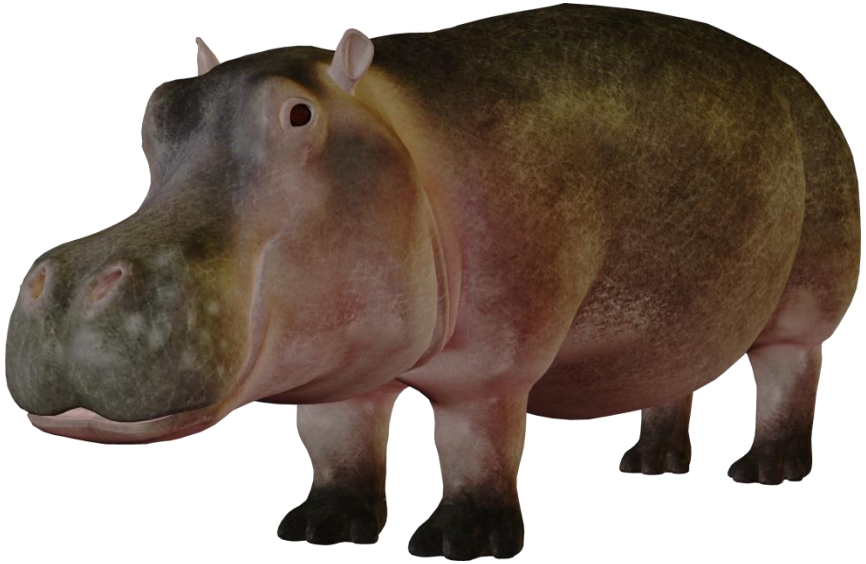
1. **Упростить разработку** (процесс написания программы).
2. **Повысить надежность ПО** (минимизация ошибок).
3. **Повторно использовать имеющийся код**.

Задача – съесть бегемота



Сложная проблема

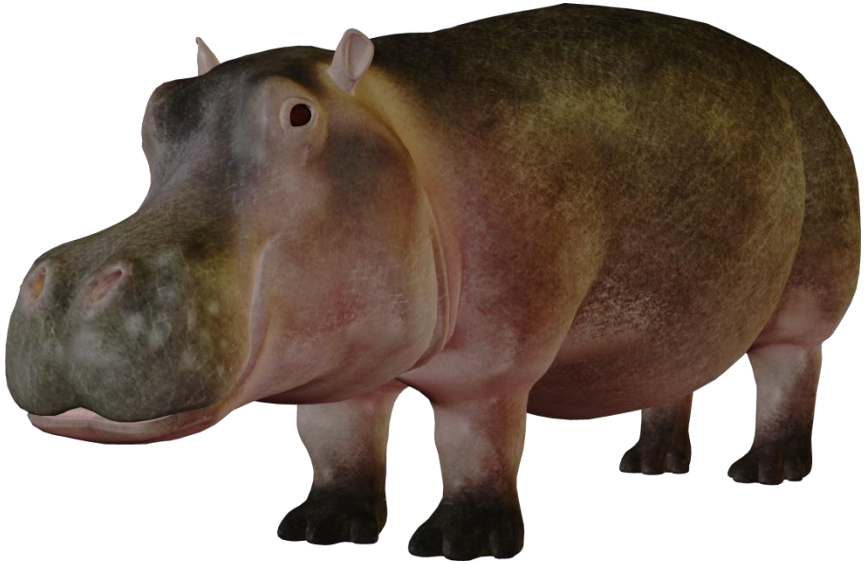
Задача – съесть бегемота



Простые фрагменты

Сложная проблема

Задача – съесть бегемота



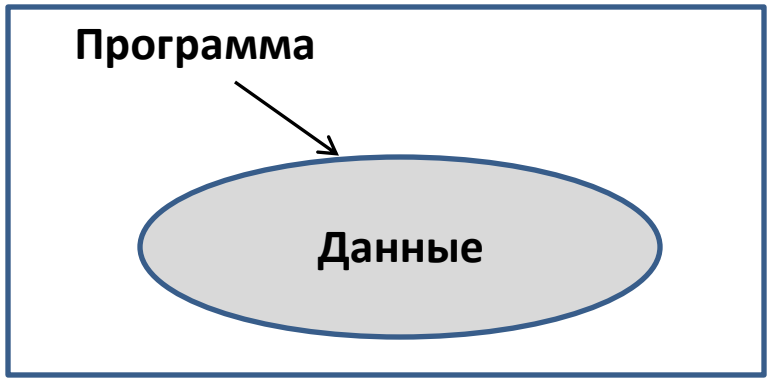
Методика проектирования ПО

Простые фрагменты

Сложная проблема

«Стихийное» программирование

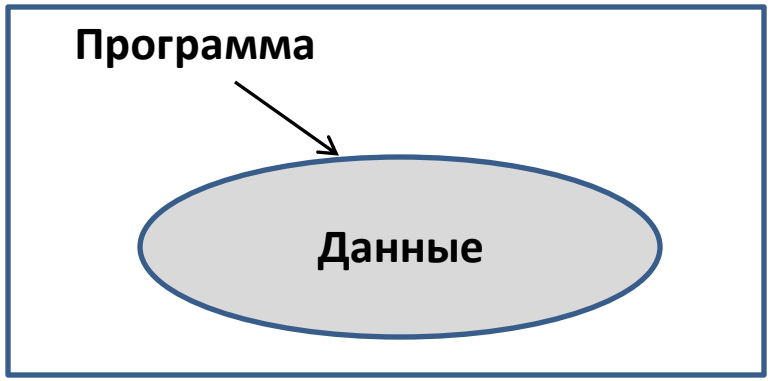
Программирование = искусство (нет технологий)



Монолитная структура первых программ

«Стихийное» программирование

Программирование = искусство (нет технологий)

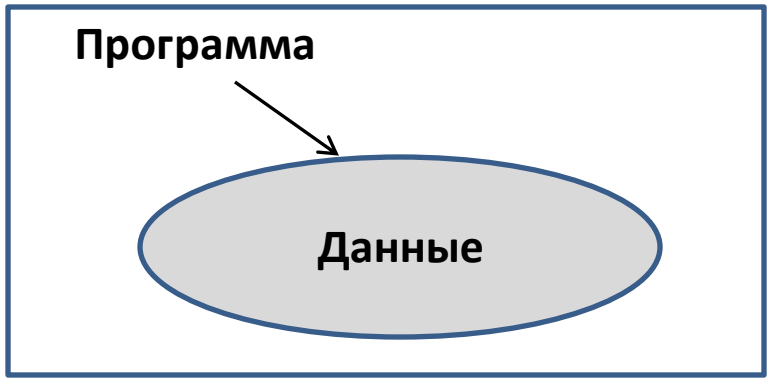


Монолитная структура первых программ

Программы очень сложные - программист должен одновременно мысленно отслеживать последовательность выполняемых операций и местонахождение данных.

«Стихийное» программирование

Программирование = искусство (нет технологий)



Монолитная структура первых программ

Программы очень сложные - программист должен одновременно мысленно отслеживать последовательность выполняемых операций и местонахождение данных.

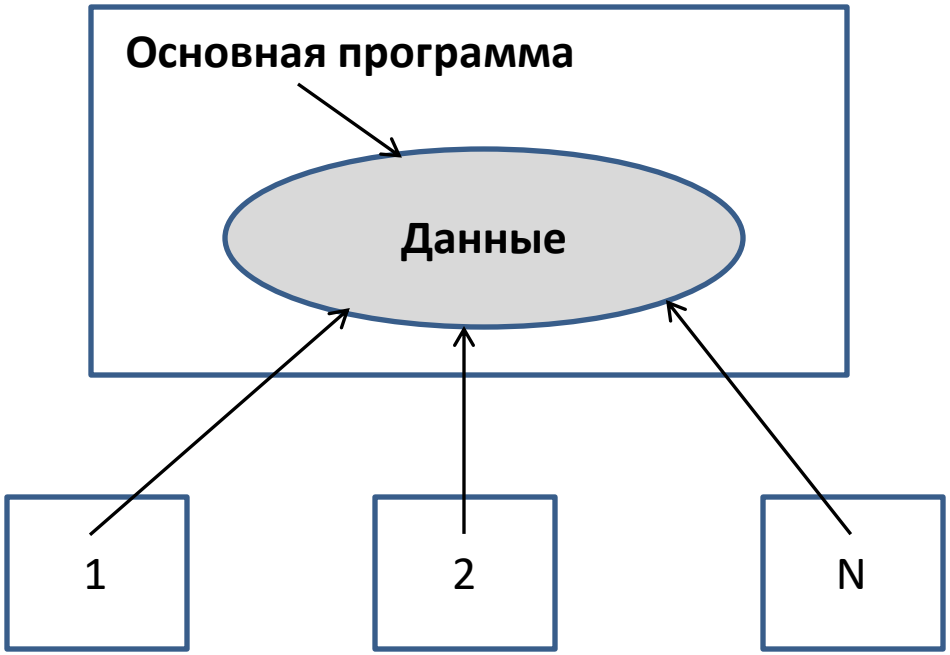
Как снижали сложность программ?

Упрощение программирования + повторное использование кода

- **Появление ассемблера** (символические имена данных и операций).
- **Создание языков высокого уровня**. Снижение уровня детализации операций, написание программ в терминах предметной области.
- Введение в языки **подпрограмм**.

Упрощение программирования + повторное использование кода

- Появление ассемблера (символические имена данных и операций).
- Создание языков высокого уровня. Снижение уровня детализации операций, написание программ в терминах предметной области.
- Введение в языки **подпрограмм**.



Программы с глобальной областью данных

Подпрограммы (библиотечные и собственные)

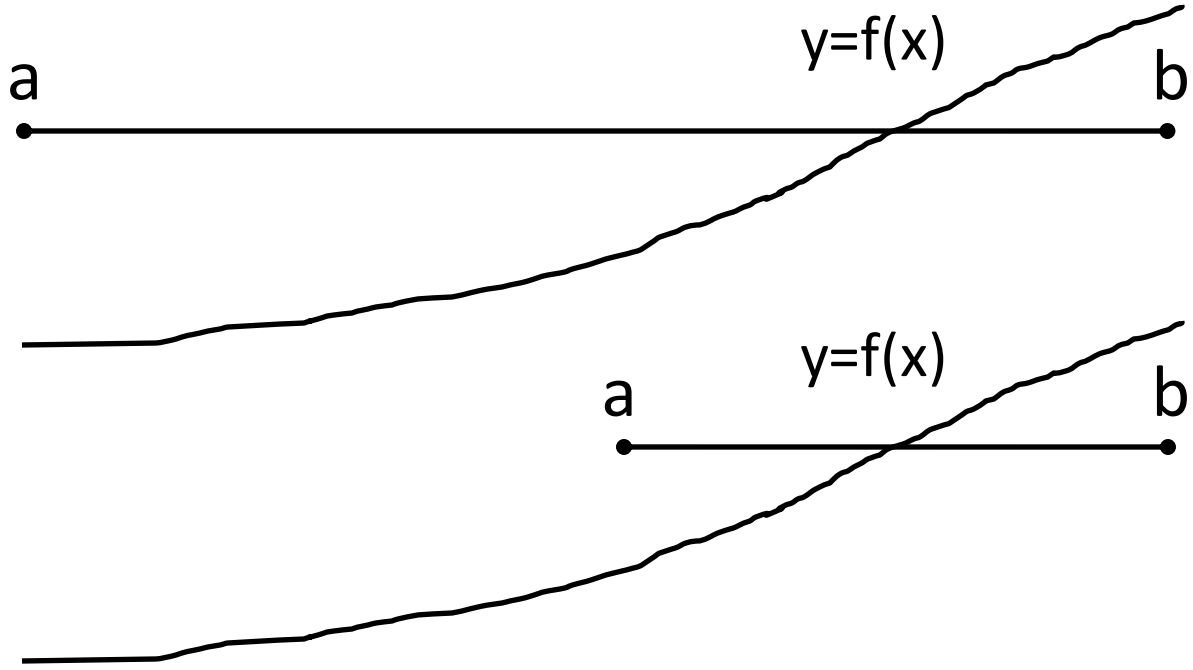
Повышение надежности

Проблема. Увеличивается количество подпрограмм – растет вероятность искажения части глобальных данных какой-то подпрограммой.

Повышение надежности

Проблема. Увеличивается количество подпрограмм – растет вероятность искажения части глобальных данных какой-то подпрограммой.

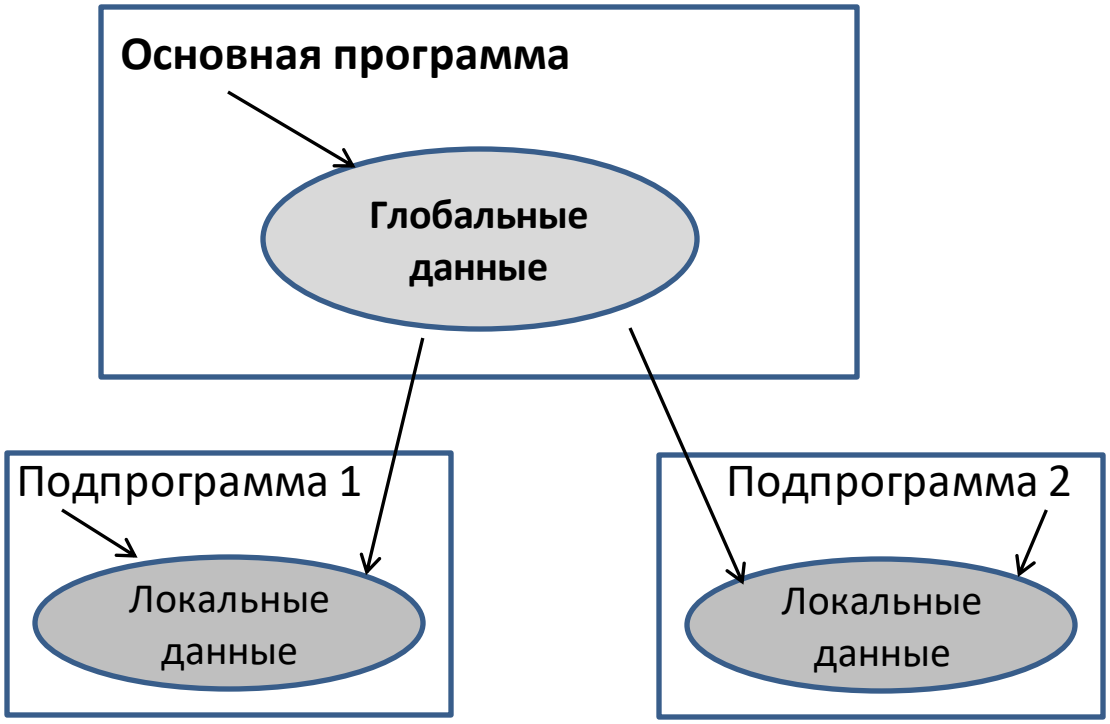
Пример. Нахождение решения уравнения дихотомией.



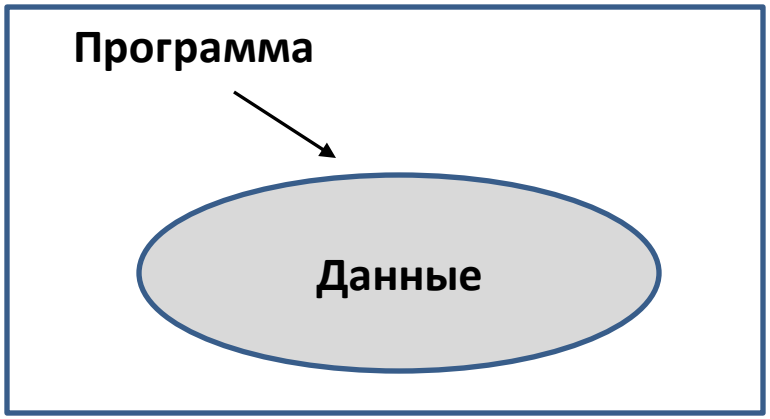
Повышение надежности

Проблема. Увеличивается количество подпрограмм – растет вероятность искажения части глобальных данных какой-то подпрограммой.

Решение. Подпрограммы с локальными данными.

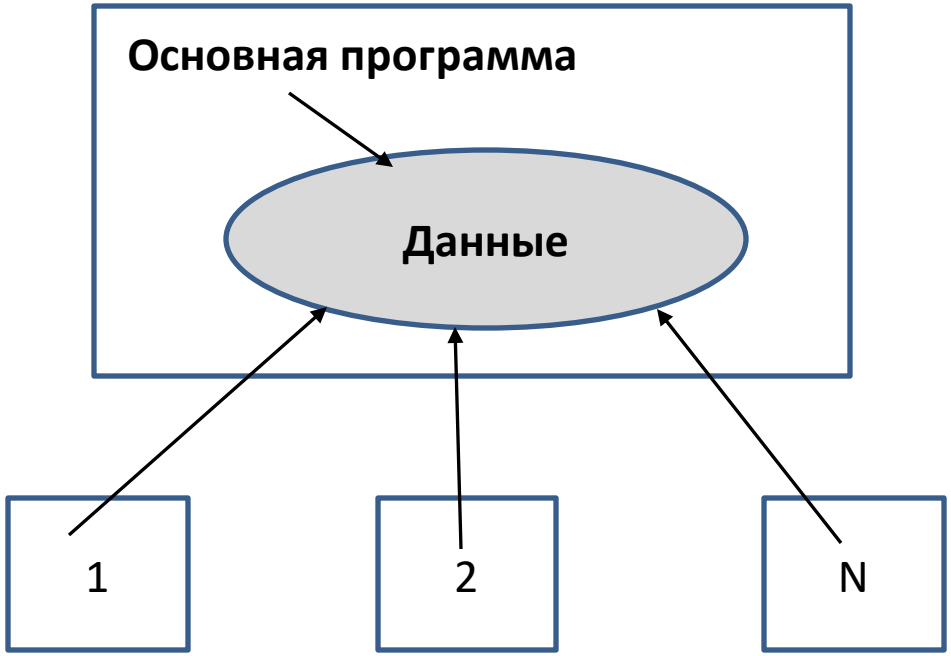


Эволюция структуры программ



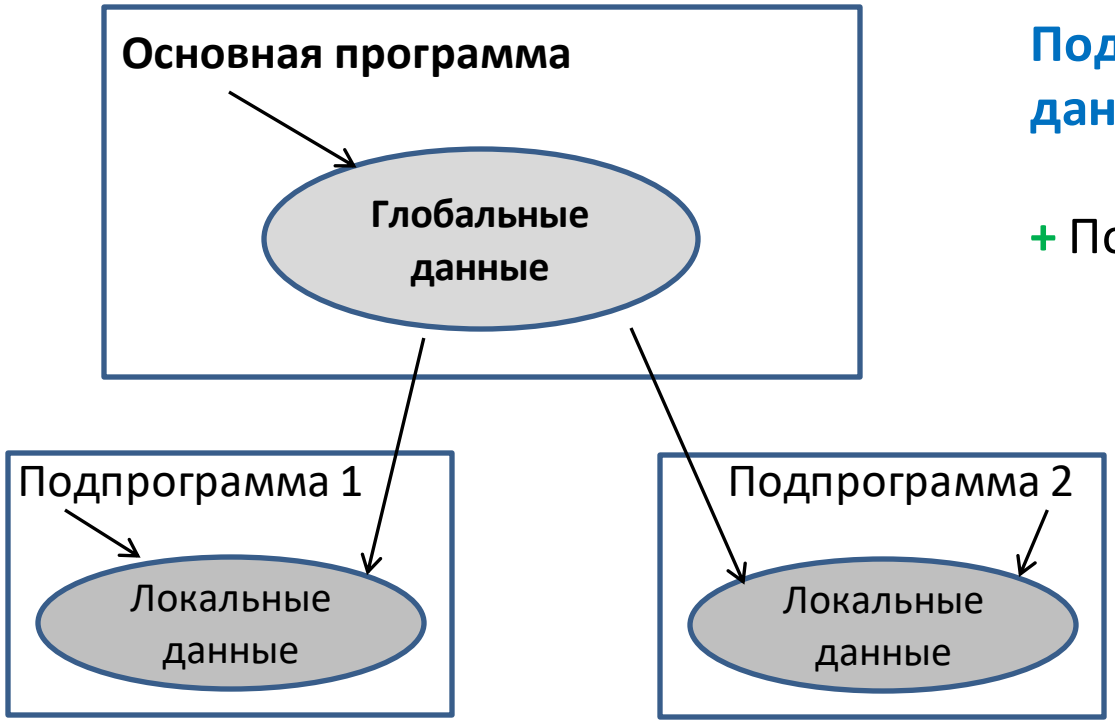
Монолитная структура

- Трудно отлаживать и повторно использовать код



Подпрограммы, глобальные данные

+ Декомпозиция (упрощение)
+ Повторное использование кода
- Растет вероятность испортить данные



Подпрограммы с локальными данными

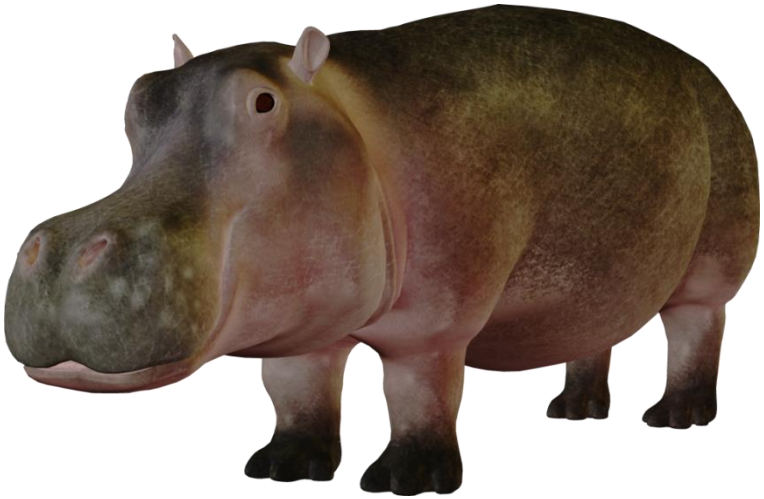
+ Повышение надежности кода

Процедурный подход

- Реши, какие требуются процедуры.
- Используй наилучшие доступные алгоритмы.

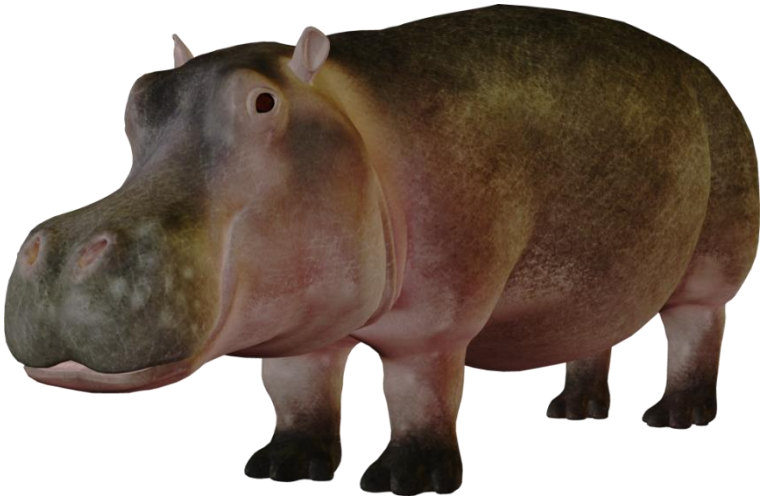
Кризис стихийного программирования

Разработка «снизу-вверх» - вначале проектировали и реализовывали простые подпрограммы, из которых пытались строить сложную систему.



Кризис стихийного программирования

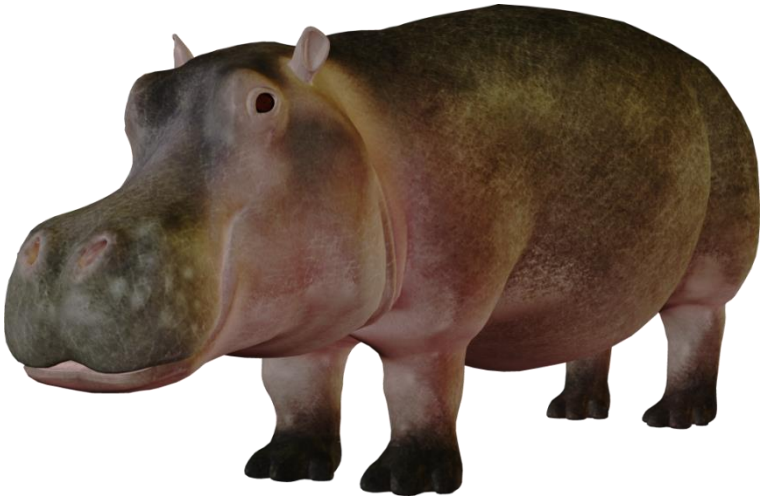
Разработка «снизу-вверх» - вначале проектировали и реализовывали простые подпрограммы, из которых пытались строить сложную систему.



- Не было четких моделей для описания подпрограмм и методов их проектирования.
- Интерфейсы подпрограмм получались сложными.
- При сборке программы выявлялись ошибки, для их исправлений изменялись подпрограммы.

Кризис стихийного программирования

Разработка «снизу-вверх» - вначале проектировали и реализовывали простые подпрограммы, из которых пытались строить сложную систему.



- Не было четких моделей для описания подпрограмм и методов их проектирования.
- Интерфейсы подпрограмм получались сложными.
- При сборке программы выявлялись ошибки, для их исправлений изменялись подпрограммы.

Нужна технология для создания сложных программ за приемлемое время.

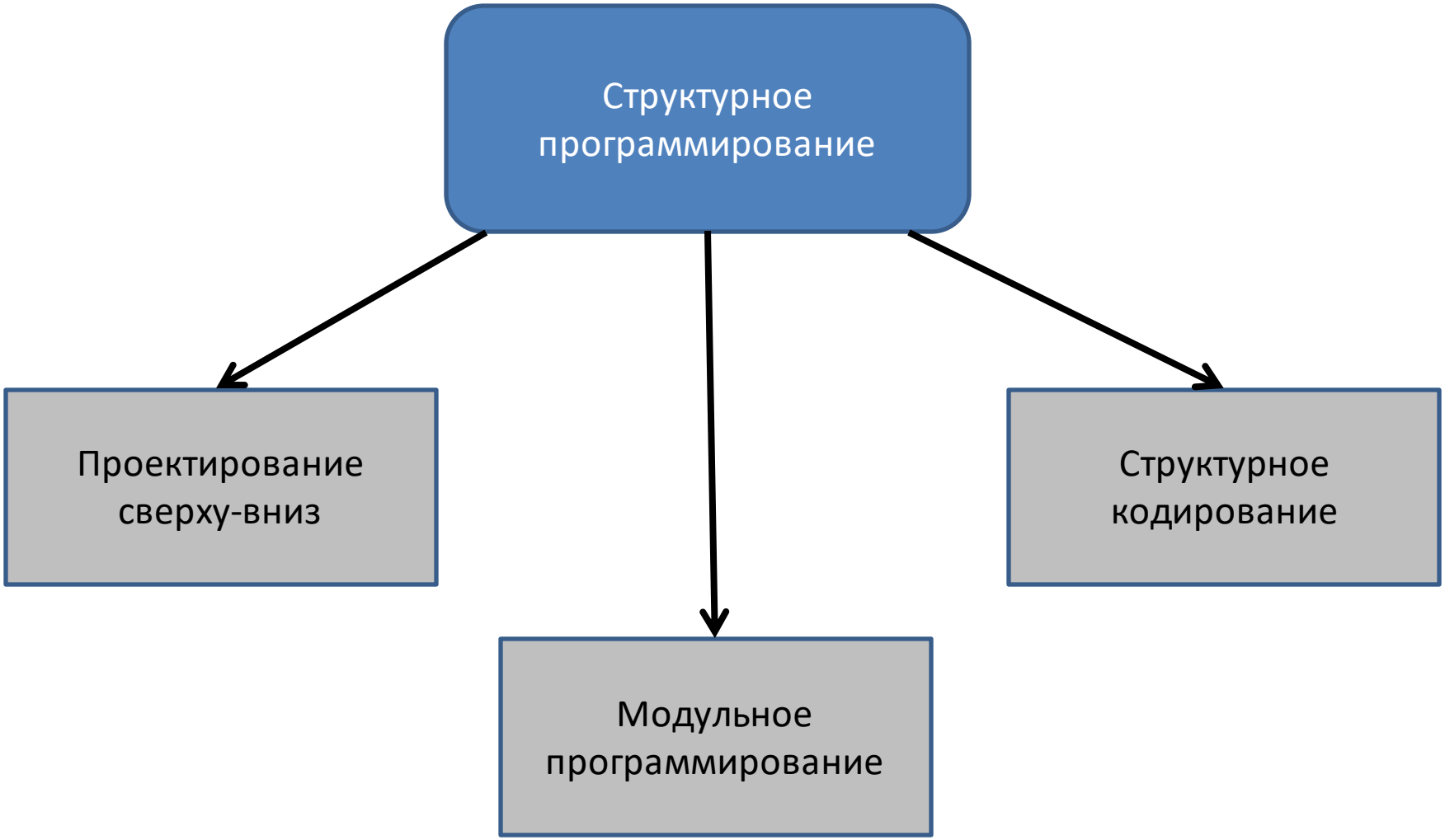
Причина кризиса – запутанная структура программ.

Структурное программирование – методология разработки ПО для создания логически простых и понятных программ.

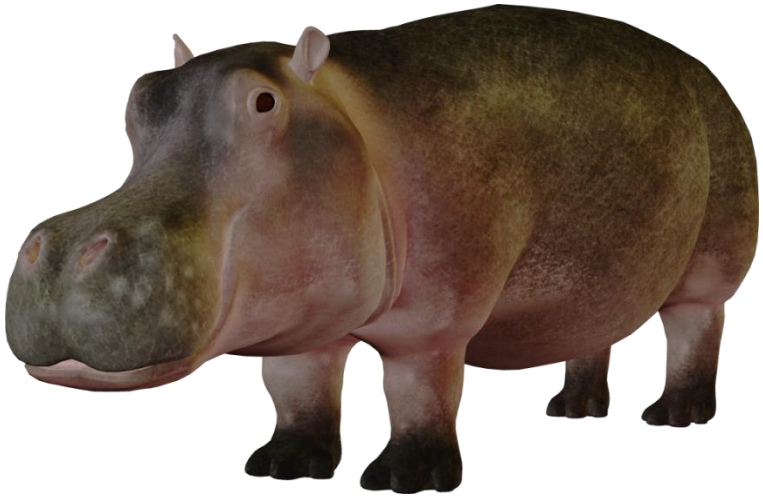
Эдсгер Дейкстра (1930-2002)

Статья «Structured programming» (1969 год) на конференции НАТО по разработке ПО

Никлаус Вирт (1934 г.р.)
Создатель языка Pascal


Структурное программирование
Проектирование сверху-вниз
Модульное программирование
Структурное кодирование

Проектирование «сверху вниз»



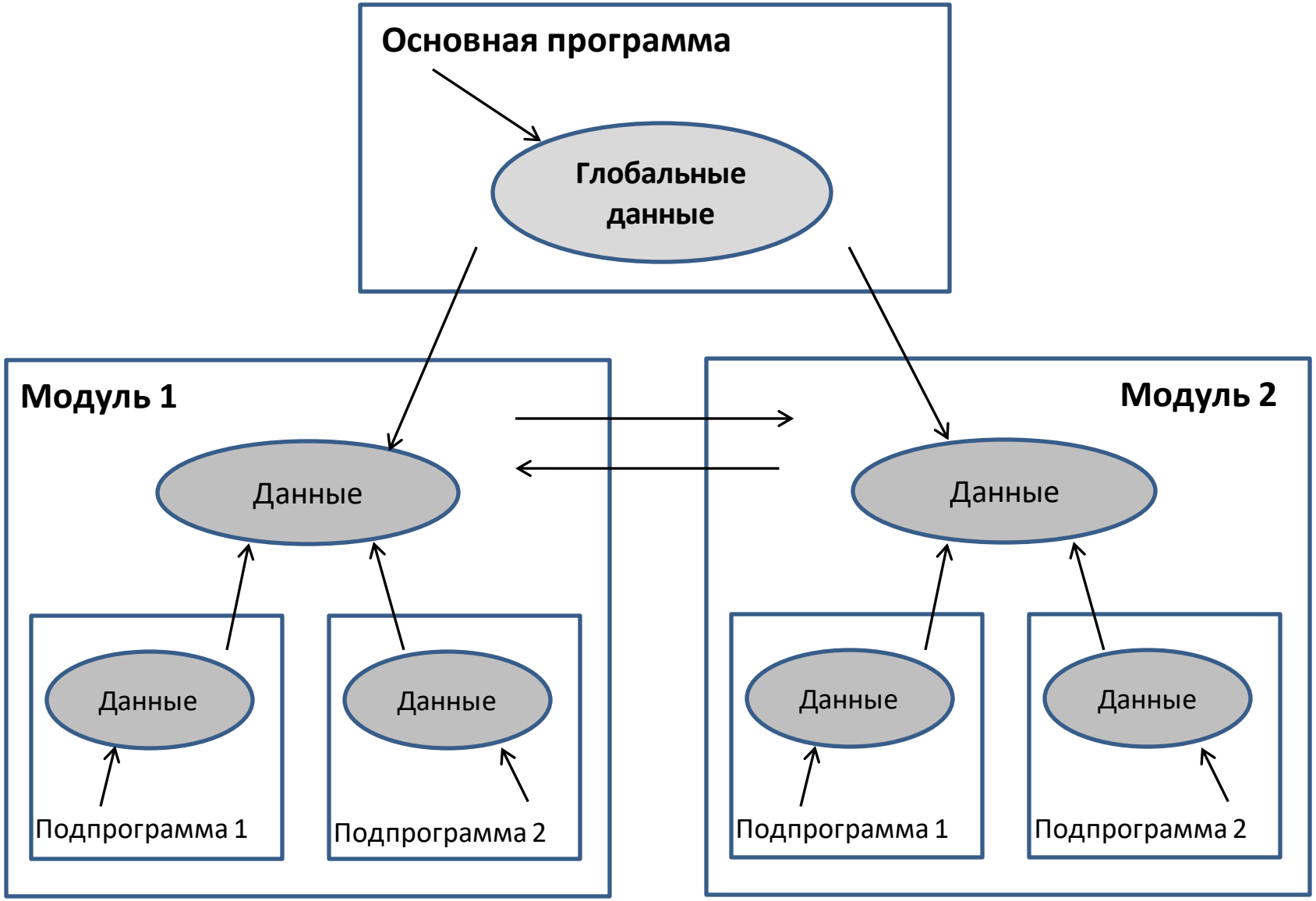
- Сначала напишите скелет программы на естественном языке.
- Вначале определение задачи в общих чертах, затем мелкие детали.
- Разрабатывайте тестовые данные заранее.
- Прежде чем начать программировать, разработайте проект.

Модульное программирование

Разделение программы на логические части (**модули**) и последовательное программирование каждой части.

Обычно в отдельно компилируемые модули (библиотеки подпрограмм) выделяют группы подпрограмм, использующие одни и те же глобальные данные.

Модульное программирование



Модульное программирование

Основные цели

- Корректность модуля
- Независимость модуля:
  - источника входных данных;
  - места назначения выходных данных;
  - от предыстории.
- Возможность формировать из модулей большие программы без каких-либо знаний о внутренней работе модуля.

Модульное программирование

Модуль – это замкнутый блок, вход и выход которого четко определены.

Основные цели

- Корректность модуля
- Независимость модуля:
  - источника входных данных;
  - места назначения выходных данных;
  - от предыстории.
- Возможность формировать из модулей большие программы без каких-либо знаний о внутренней работе модуля.

Модульный подход

- Реши, какие требуются модули.
- Разбей программу так, чтобы скрыть данные в модулях.

Структурное кодирование

Позволяет получать программы, более удобные для тестирования, модификации и использования.

Любую программу можно написать с использованием только следующих логических структур:

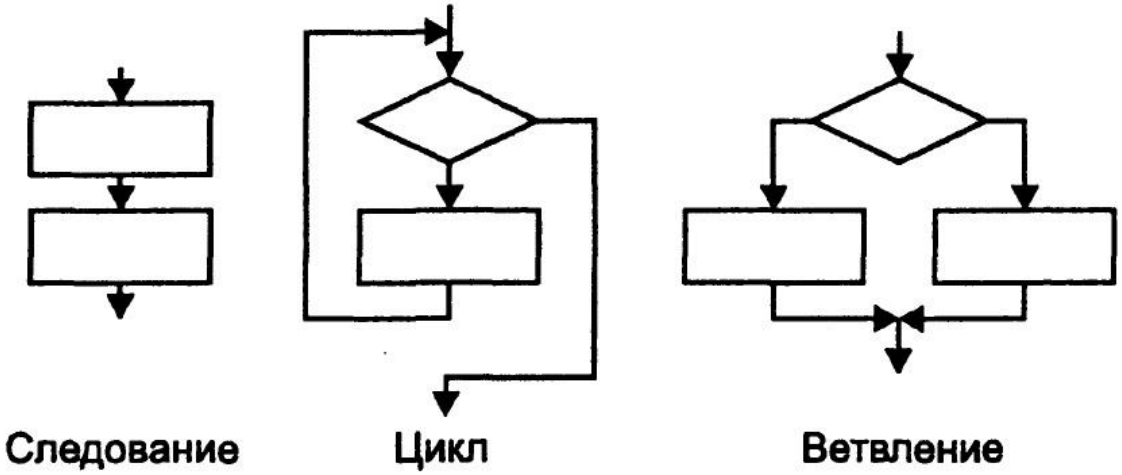
- **Последовательности двух или более операторов.**
- **Выбора одного из двух операторов.**
- **Повторения оператора, пока выполняется некоторое условие.**

Структурное кодирование

Позволяет получать программы, более удобные для тестирования, модификации и использования.

Любую программу можно написать с использованием только следующих логических структур:

- **Последовательности двух или более операторов.**
- **Выбора одного из двух операторов.**
- **Повторения оператора, пока выполняется некоторое условие.**



Базовые конструкции структурного программирования

Форматирование исходного кода

```
/*! jQuery v1.12.2 | (c) jQuery Foundation | jquery.org/license */
!function(a,b){"object"==typeof module&&"object"==typeof module.exports?module.exports=a.
document?b(a,!0):function(a){if(!a.document)throw new Error("jQuery requires a window
with a document");return b(a)}:b(a)}("undefined"!=typeof window?window:this,function(a,b
){var c=[],d=a.document,e=c.slice,f=c.concat,g=c.push,h=c.indexOf,i={},j=i.toString,k=i.
hasOwnProperty,l={},m="1.12.2",n=function(a,b){return new n.fn.init(a,b)},o=
/^[\s\uFEFF\xA0]+|[\s\uFEFF\xA0]+$/g,p=/^-ms-/,q=/-([\da-z])/gi,r=function(a,b){return b.
toUpperCase()};n.fn=n.prototype={jquery:m,constructor:n,selector:"",length:0,toArray:
function(){return e.call(this)},get:function(a){return null!=a?0>a?this[a+this.length]:
this[a]:e.call(this)},pushStack:function(a){var b=n.merge(this.constructor(),a);return b.
prevObject=this,b.context=this.context,b},each:function(a){return n.each(this,a)},map:
function(a){return this.pushStack(n.map(this,function(b,c){return a.call(b,c,b)}))},slice
:function(){return this.pushStack(e.apply(this,arguments))},first:function(){return this.
eq(0)},last:function(){return this.eq(-1)},eq:function(a){var b=this.length,c=+a+(0>a?b:0
);return this.pushStack(c>=0&&b>c?[this[c]]:[])},end:function(){return this.prevObject||
this.constructor()},push:g,sort:c.sort,splice:c.splice},n.extend=n.fn.extend=function(){
var a,b,c,d,e,f,g=arguments[0]||{},h=1,i=arguments.length,j=!1;for("boolean"==typeof g&&(
j=g,g=arguments[h]||{},h++),"object"==typeof g||n.isFunction(g)||(g={}),h===i&&(g=this,h
--);i>h;h++)if(null!=(e=arguments[h]))for(d in e)a=g[d],c=e[d],g!==c&&(j&&c&&(n.
isPlainObject(c)||(b=n.isArray(c)))?(b?(b=!1,f=a&&n.isArray(a)?a:[]):f=a&&n.isPlainObject
(a)?a:{},g[d]=n.extend(j,f,c)):void 0!==c&&(g[d]=c));return g},n.extend({expando:"jQuery"
+(m+Math.random()).replace(/\D/g,""),isReady:!0,error:function(a){throw new Error(a)},
noop:function(){},isFunction:function(a){return"function"===n.type(a)},isArray:Array.
isArray||function(a){return"array"===n.type(a)},isWindow:function(a){return null!=a&&a==a
.window},isNumeric:function(a){var b=a&&a.toString();return!n.isArray(a)&&b-parseFloat(b
)+1>=0},isEmptyObject:function(a){var b;for(b in a)return!1;return!0},isPlainObject:
function(a){var b;if(!a||"object"!==n.type(a)||a.nodeType||n.isWindow(a))return!1;try{if(
a.constructor&&!k.call(a,"constructor")&&!k.call(a.constructor.prototype,"isPrototypeOf"
```

Форматирование кода

- Визуализация логической структуры программы с помощью отступов.

Форматирование кода

- Логическая структура программы через отступы.

```
with open('Crimes.csv', 'rt') as f:
    content = csv.reader(f)
    i = 0
    for row in content:
        if i > 0:
            str_date = row[2]
            if time.strptime(str_date).tm_year == 2015:
                Primary_Type = row[5]
                if Primary_Type in type_dict:
                    type_dict[Primary_Type] += 1
                else:
                    type_dict[Primary_Type] = 1
        i+=1
print(type_dict)
values = list(type_dict.items())
values.sort(key=lambda item: item[1])
print(values)
```

Форматирование кода

- Логическая структура программы через отступы.
- Определенный стиль форматирования и именования.

  CamelCase, under_score

Форматирование кода

- Логическая структура программы через отступы.
- Определенный стиль форматирования и именования.
- Ограничение длины выражений, дополнительные пробелы для операндов и пустые строки для выделения программных блоков.

```
a=a+b
a = a + b
```

Форматирование кода

- Логическая структура программы через отступы.
- Определенный стиль форматирования и именования.
- Ограничение длины выражений, дополнительные пробелы для операндов и пустые строки для выделения программных блоков.
- Осмысленные названия переменных – код должен быть самодокументируемым и понятным даже без комментариев.

Самодокументируемый код

Форматирование кода

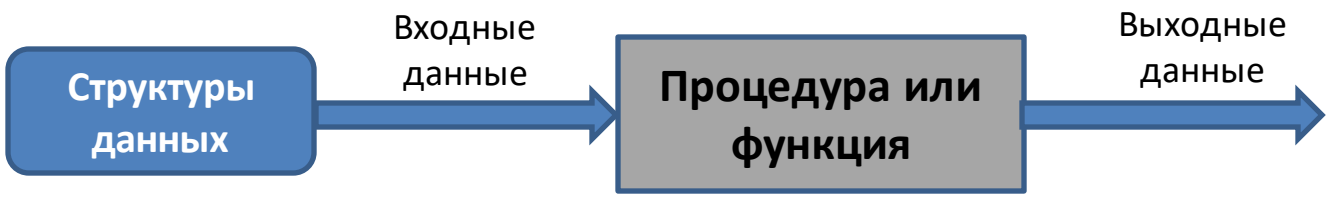
- Логическая структура программы через отступы.
- Определенный стиль форматирования и именования.
- Ограничение длины выражений, дополнительные пробелы для операндов и пустые строки для выделения программных блоков.
- Осмысленные названия переменных – код должен быть самодокументируемым и понятным даже без комментариев.

- Комментарии должны пояснять цель и задачу кода, а не ее решение.
  - *«Получение информации о текущем сотруднике»* - комментарий цели.
  - *«Обновление объекта employeeRecord»* – комментарий в терминах решения проблемы

Пример кода

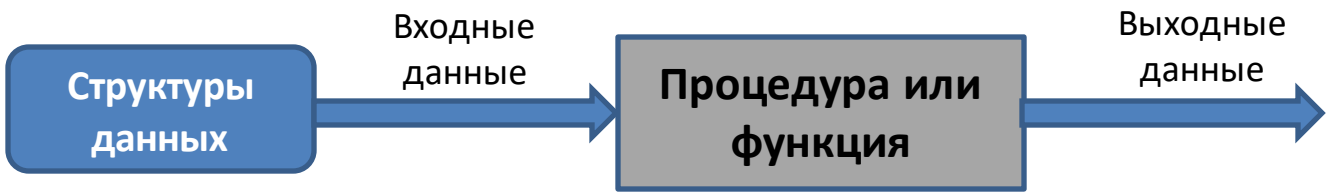
Структура программ при процедурном и модульном подходах

- Приложение – последовательность событий или программ, воздействующих на отдельные структуры данных.
- Сложные программы делятся на простые фрагменты с помощью подпрограмм.
- Данные и операции – логически разные сущности.



Структура программ при процедурном и модульном подходах

- Приложение – последовательность событий или программ, воздействующих на отдельные структуры данных.
- Сложные программы делятся на простые фрагменты с помощью подпрограмм.
- Данные и операции – логически разные сущности.

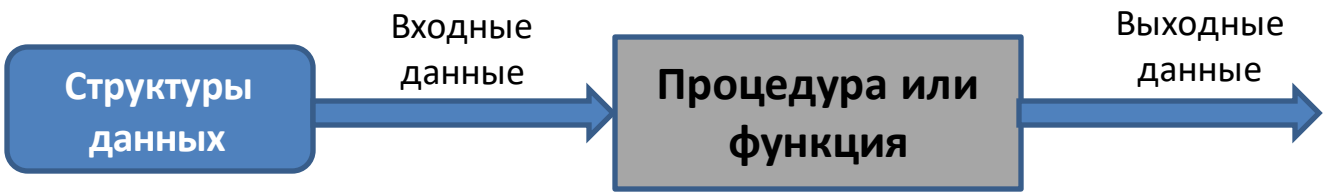


- Переменные (данные) могут иметь внутреннюю структуру.

```
Type
  Person = record
    Name: string;
    Address: string;
    Age: integer;
  end;
```

Структура программ при процедурном и модульном подходах

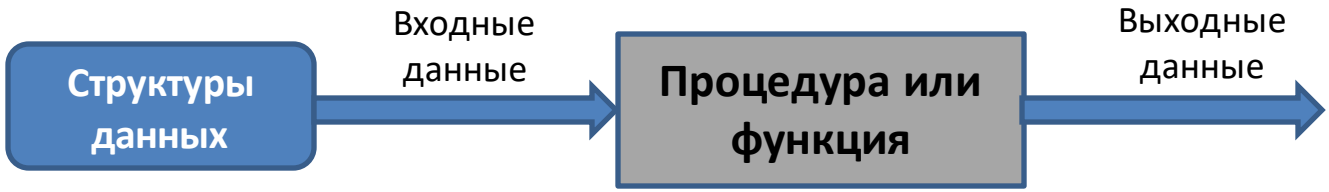
- Приложение – последовательность событий или программ, воздействующих на отдельные структуры данных.
- Сложные программы делятся на простые фрагменты с помощью подпрограмм.
- Данные и операции – логически разные сущности.



- Переменные (данные) могут иметь внутреннюю структуру.
- Тип переменной определяет множество ее допустимых значений.

Структура программ при процедурном и модульном подходах

- Приложение – последовательность событий или программ, воздействующих на отдельные структуры данных.
- Сложные программы делятся на простые фрагменты с помощью подпрограмм.
- Данные и операции – логически разные сущности.



- Переменные (данные) могут иметь внутреннюю структуру.
- Тип переменной определяет множество ее допустимых значений.
- Данные локализуются в модулях и защищаются от воздействий (инкапсулируются) с помощью интерфейсов.