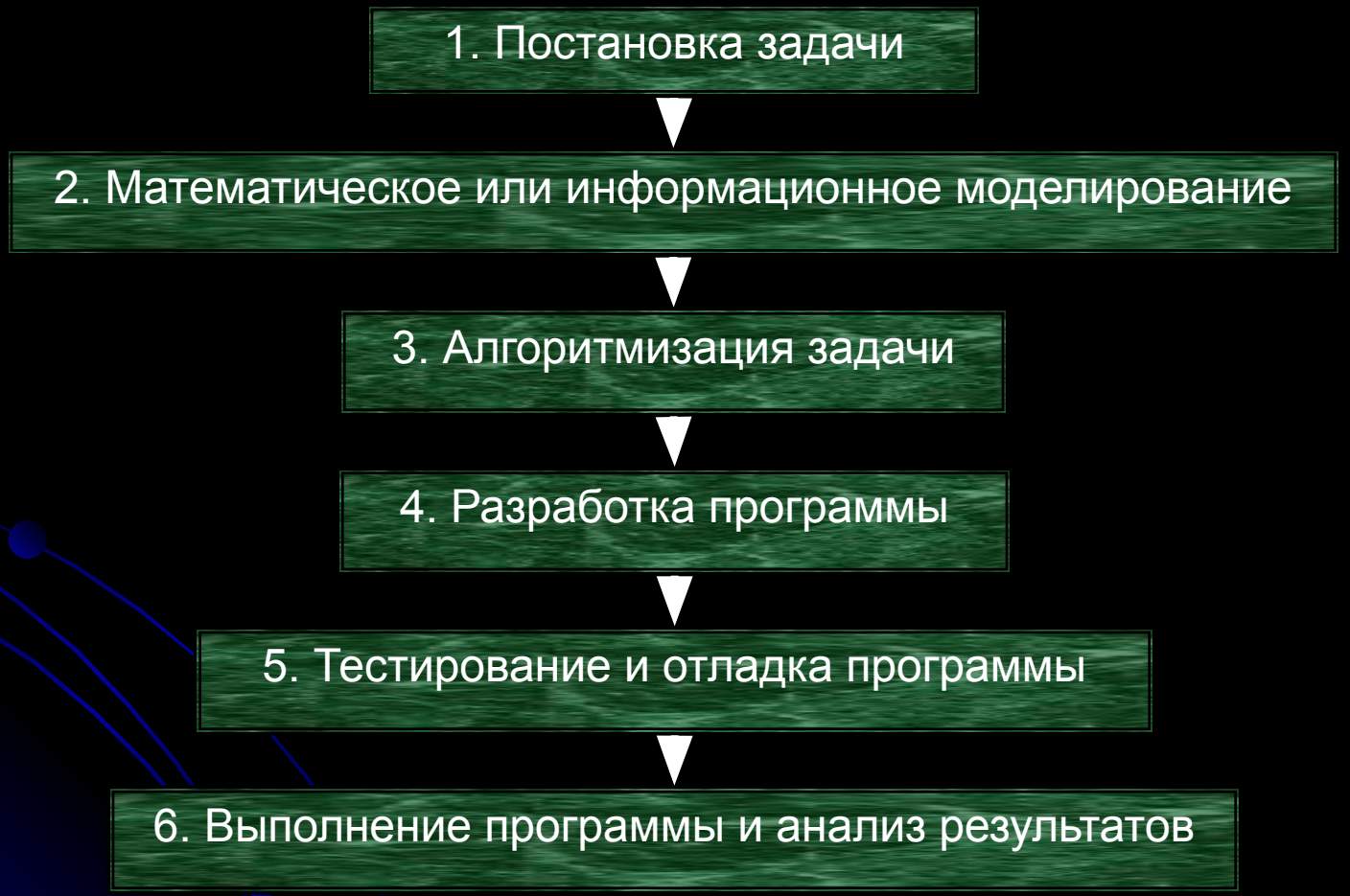
Глава 2. Алгоритмы и основы языка программирования TURBO PASCAL 7.0

Лекция 6. Алгоритмы

2.6.1. Общие сведения об алгоритме
2.6.2. «Величина» и алгоритм
2.6.3. Структурное программирование

2.6.1. Общие сведения об алгоритме

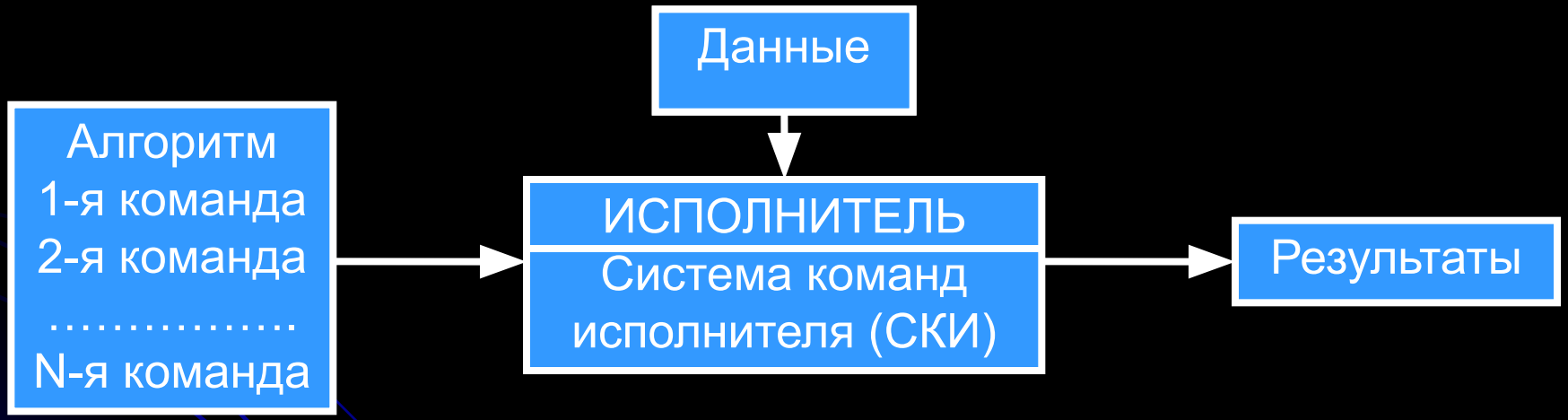
Процесс решения задачи на ЭВМ



2.6.1. Общие сведения об алгоритме

Алгоритм и обработка данных

Алгоритм — понятная и точная последовательность команд исполнителю выполнить преобразование исходных (входных) данных в желаемый результат (выходные данные) за конечное число шагов.



Исполнитель — это тот объект (или субъект), для управления которым составляется алгоритм.

2.6.1. Общие сведения об алгоритме

Свойства алгоритмов

Дискретность **— алгоритм определяется последовательностью шагов (этапов или действий), причем выполнение очередного шага возможно только после выполнения предыдущих и основывается на их результатах.**

Детерминированность **(определенность) — однозначность действий исполнителя на каждом шаге обработки данных.**

Элементарность **— простота и строгая определенность правила получения данных на каждом шаге из предыдущих результатов.**

Результативность **(конечность)** – конечное число шагов для получения результата.

Массовость – пригодность алгоритма для решения определенного класса задач из некоторого множества.

2.6.1. Общие сведения об алгоритме

Математическая модель алгоритма и его сложность

Работу алгоритма можно представить в виде ряда преобразований исходных данных Q_0 , выраженных посредством алфавита A, в окончательные результаты Q_k за k шагов (действий):

Исполнение любого алгоритма требует обеспечения определенных ресурсов:

Память → Эффективность ← Время

Временная сложность алгоритма — это функция, которая для всех конкретных однотипных задач с объемом входных данных n ставит в соответствие максимальное время, затрачиваемое алгоритмом на ее решение.

Полиноминальный алгоритм

Экспоненциальный алгоритм

$T(n)$

Трудноразрешимый

2.6.1. Общие сведения об алгоритме

Исполнитель алгоритма

Исполнитель алгоритма — это субъект или устройство, способные правильно интерпретировать описание алгоритма и выполнить содержащийся в нем перечень действий.

Исполнитель алгоритма считается заданным, если для него установлены параметры:

- система команд — совокупность элементарных действий алгоритма, которые способен выполнить исполнитель;
- формы представления входной и выходной информации;
- система допустимых внутренних состояний;
- язык представления алгоритма.

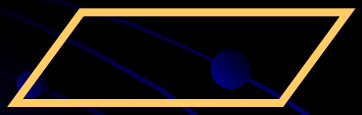
Для написания алгоритмов исполнителями которых являются компьютеры разработали формальные языки со строгим синтаксисом и полной смысловой определенностью.

2.6.1. Общие сведения об алгоритме

Графическое изображение алгоритма (блок-схема)

Начало и конец алгоритма

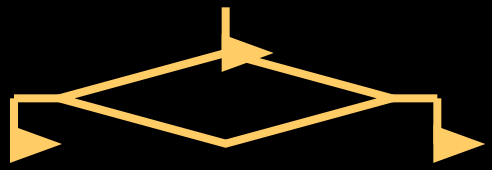
Блок обработки данных



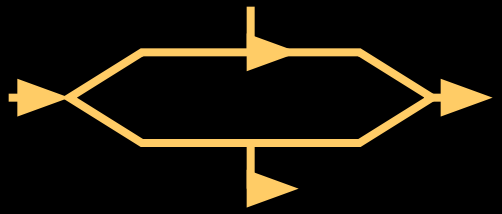
Ввод-вывод данных

Соединительная линия

Переход на следующую страницу



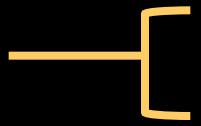
Блок ветвления по условию



Циклическая конструкция

Подпрограмма

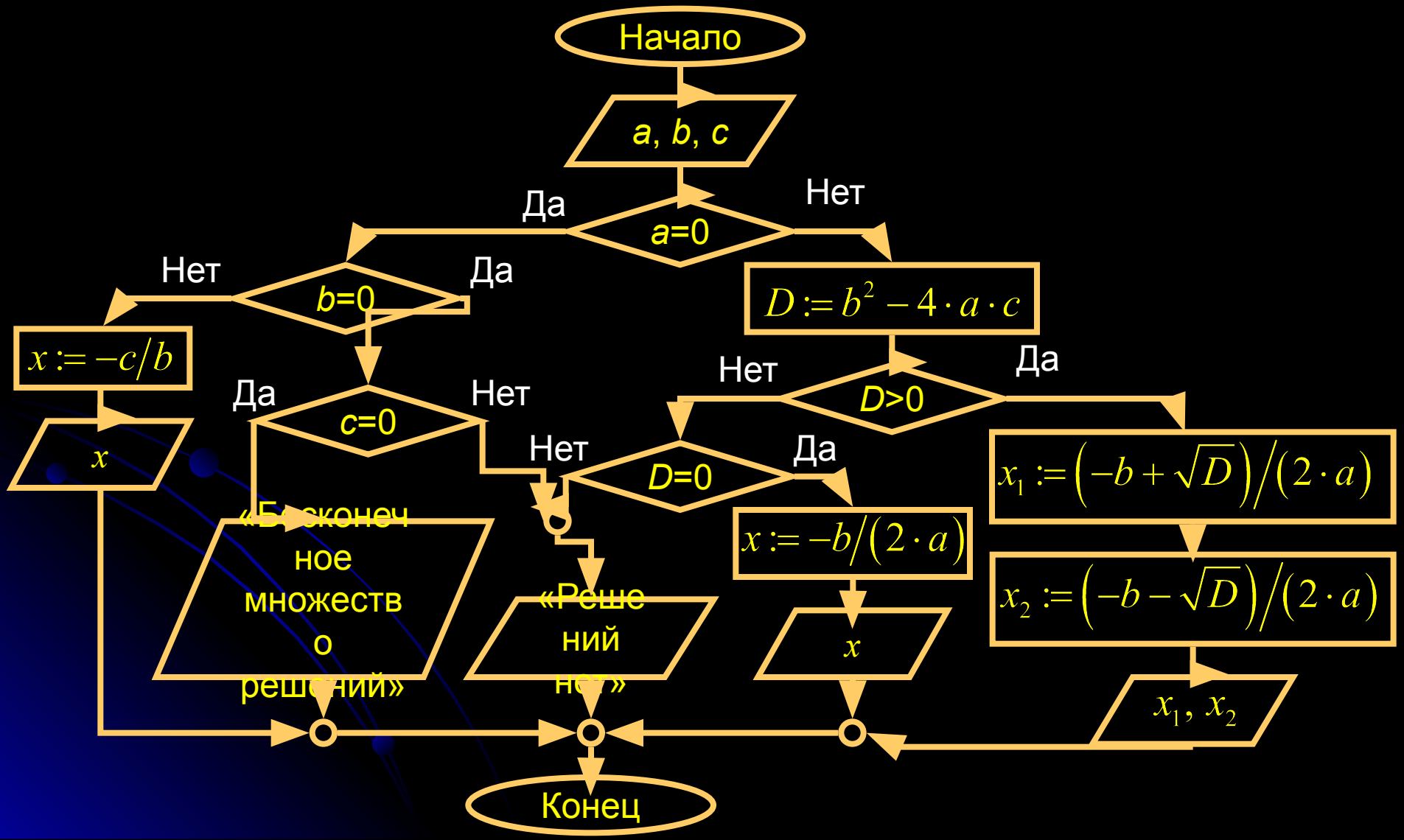
Объединение



Комментарии

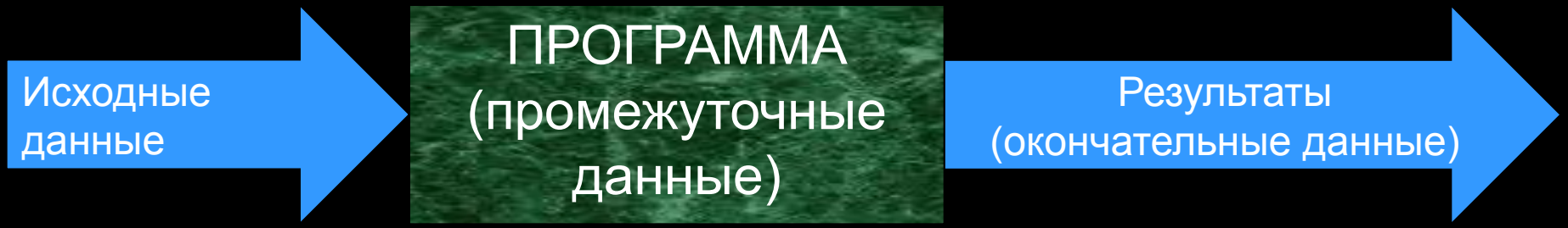
2.6.1. Общие сведения об алгоритме

Пример алгоритма решения квадратного уравнения



2.6.2. «Величина» и алгоритм

Данные и величина



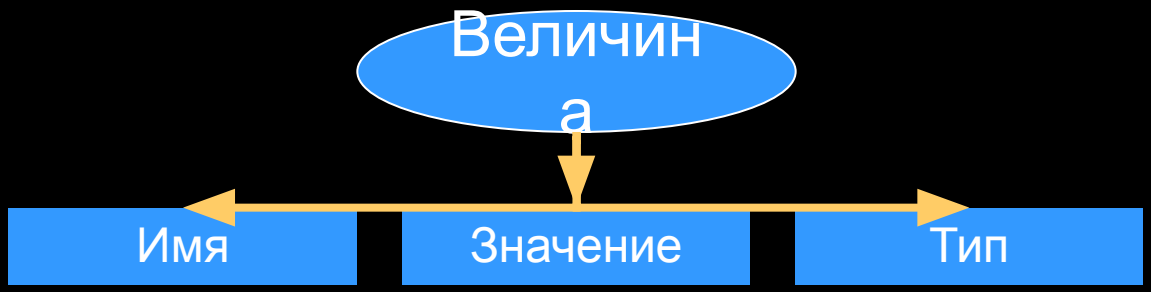
Данные — это сведения, характеризующие какой-то объект, процесс или явление, представленные в определенной форме и предназначенные для дальнейшего использования.

Величина — это отдельный информационный объект, отдельная единица данных.

Всякая величина занимает свое определенное место в памяти ЭВМ — ячейку памяти.

2.6.2. «Величина» и алгоритм

Величина



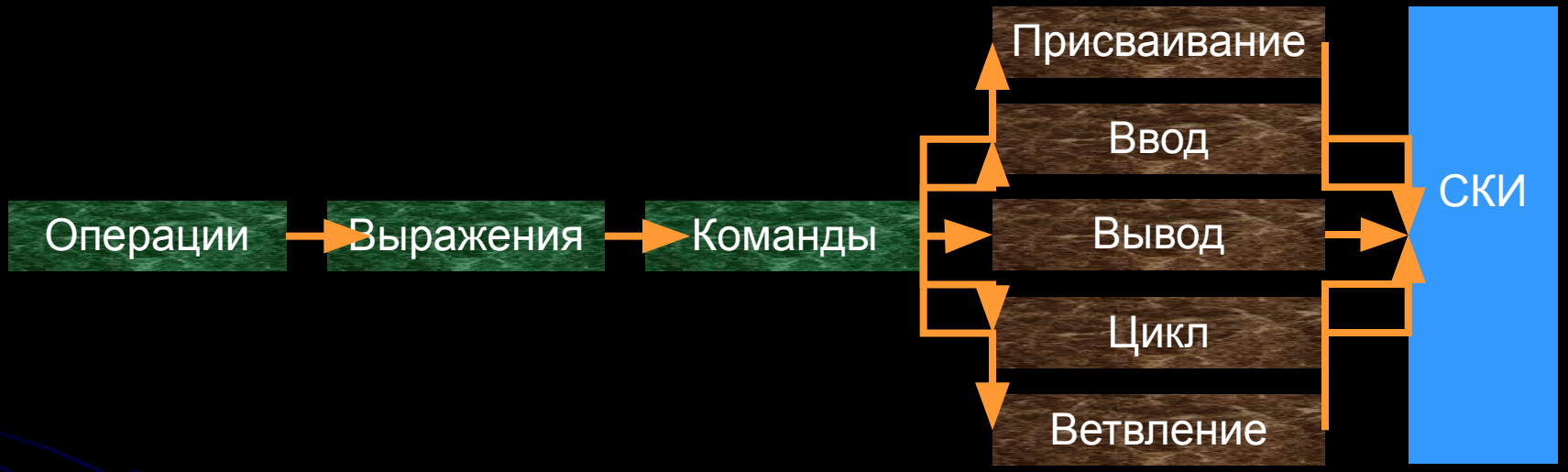
2.6.2. «Величина» и алгоритм

Свойства основных типов данных

Тип	Значения	Операции	Внутреннее представление
Целый	Целые положительные и отрицательные числа в некотором диапазоне	Арифметические операции с целыми числами (+, –, ×, DIV, MOD). Операции отношений (<, ≤, =, ≥, >, ≠).	Формат с фиксированной точкой
Вещественный	Любые (целые и дробные) числа в некотором диапазоне	Арифметические операции (+, —, ×, /). Операции отношений.	Формат с плавающей точкой
Логический	True (истина) False (ложь)	Логические операции. Операции отношений.	1 бит: 1 — true; 0 —false
Символьный	Любые символы компьютерного алфавита.	Операции отношений. Операции конкатенации.	Коды таблицы символьной кодировки. 1 символ — 1 байт

2.6.2. «Величина» и алгоритм

Действия над величинами



Операция — простейшее законченное действие над данными.

Выражение — запись в алгоритме (программе), определяющая последовательность операций для вычисления некоторой величины.

Команда — входящее в запись алгоритма типовое предписание исполнителю выполнить некоторое законченное действие.

2.6.2. Структурное программирование

Структурное программирование — это проектирование, написание и тестирование программы в соответствии с жестким соблюдением определенных правил.

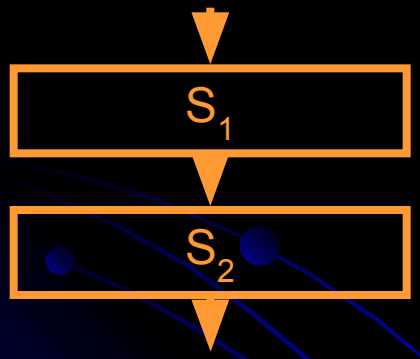
1. Метод нисходящего проектирования. Его еще называют методом «сверху вниз» или «от общего к частному». Он предполагает разбиение задачи на несколько более простых частей или подзадач. Их выделяют таким образом, чтобы проектирование подзадач было независимым. При этом составляют план решения всей задачи, пунктами которого и являются выделенные части. Затем производят детализацию каждой подзадачи. Число шагов детализации может быть произвольным, пока не станет ясно, как программировать данный фрагмент алгоритма.
2. Структурное программирование. Реализация идеи структурного программирования основывается на том факте, что правильная программа любой сложности может быть представлена логической структурой, представляющей собой композицию трех базовых структур, определяющих правила обработки данных: следования (линейная), разветвления (условного перехода) и повторения (цикла).
3. Сквозной структурный контроль. Он представляет собой регулярные проверки и согласования результатов работы исполнителей-программистов различных структур. Его необходимость определяется желанием разработчиков снизить стоимость разрабатываемых программ. Обязательным условием этого является раннее обнаружение и исправление возникающих ошибок и не состыковок.

2.6.2. Структурное программирование

Базовые структуры языков программирования

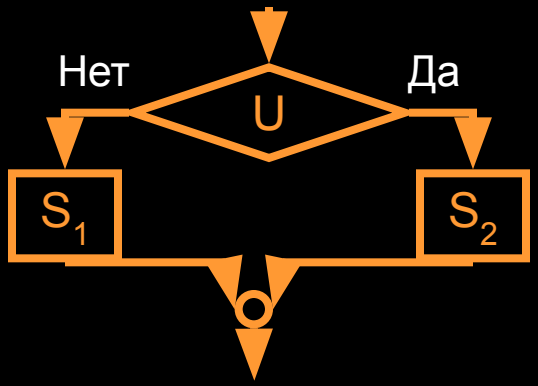
Структурная теорема: любой алгоритм может быть сведен к структурному алгоритму.

Линейный поток управления



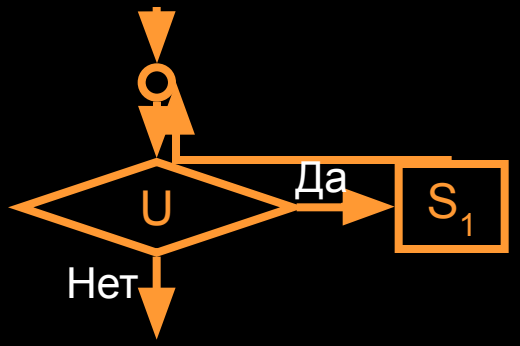
1. Выполняется каждый блок.
2. Каждый блок выполняется не более одного раза.

Ветвящейся поток управления



2. Каждый блок выполняется не более одного раза.

Циклический поток управления



1. Выполняется каждый блок.

2.6.2. Структурное программирование

Преимущества структурных алгоритмов

1. Понятность и простота восприятия алгоритма.
2. Проверяемость.
3. Модифицируемость.

Этапы структурного подхода к разработке алгоритмов

1. Описание общего замысла алгоритма.
2. Формализация задачи.
3. Разработка обобщенной схемы алгоритма.
4. Разработка отдельных блоков алгоритма.
5. Стыковка блоков.
6. Определение возможности использования стандартных блоков.
7. Разработка блоков логического контроля.
8. Оптимизация схемы алгоритма.
9. Уточнение параметров.
10. Оценка машинного ресурса.