

## **Лекция 3.**

# **Основные парадигмы и технологии программирования**

---

# Парадигма

– совокупность ценностей, методов, подходов, технических навыков и средств, принятых в научном сообществе в рамках устоявшейся научной традиции в определенный период времени.

**Парадигма программирования** – это совокупность идей и понятий, определяющих стиль написания программ.

# Первые программисты



20-е годы 19 в. Чарльз Бэббидж высказал мысль о предварительной записи порядка действий машины для последующей автоматической реализации вычислений – программе, которую можно было записывать на перфокартах.

# Первые программисты



- 1843г. - Ада Августа Лавлейс издала перевод на английский язык статьи итальянского ученого Минембраа о вычислительных машинах со своими комментариями.

# В комментариях к статье излагались принципы, ставшие основой теории программирования:

- введено понятие цикла
- сформулирован принцип экономии рабочих ячеек
- сформулирован принцип хранимых данных
- обозначена связь рекуррентных формул с циклическими процессами вычислений
- высказана мысль о том, что машина может выполнять работу, превышающую возможности человека

# Парадигмы программирования

1. Императивная
  - a) Не процедурное (машинно-ориентированное (ассемблеры))
  - b) Процедурное (структурное) (Фортран, С, Паскаль)
2. Объектная (C++, Delphi)
3. Декларативная
  - a) Логическое (Пролог)
  - b) Функциональное (Лисп и диалекты)

# машинно-ориентированное программирование

На начальном историческом этапе возникновения в 40-х годах XX электронные вычислительные машины программировались только на языках машинного уровня.

Процессору посылались бинарные коды из системы команд данного процессора, как правило, вместе с данными для обработки. Обычно речь шла об операциях по перемещению данных из памяти в регистр или о простой арифметике над содержимым регистра



В августе 1944 года для релейной машины "Марк-1" под руководством **Грейс Хоппер** (женщина-программист, морской офицер ВМФ США) была написана первая подпрограмма для вычисления  $\sin(x)$ .



В 1949 году **Джон Моучли** (один из создателей ЭВМ ENIAC) разработал систему **Short Code**, которую можно считать предшественницей языков программирования высокого уровня, первый примитивный **интерпретатор**.

В 1951 году **Г. Хоппер** создала первый **компилятор A-0**. Ею же впервые был введен этот термин и термин «отладка».

# Ассемблер

Ассемблер – машинно-зависимый язык низкого уровня, т.е. он отражает особенности архитектуры конкретного типа вычислительных машин.

Относится ко второму поколению языков программирования (если первым считать машинные коды)

Ассемблер обеспечивает возможность применения символических имен в исходной программе и избавляет программиста от необходимости распределения памяти компьютера для команд, переменных и констант.

# Ассемблер

На ассемблере пишут программы или фрагменты программ, для которых критически важны:

- быстродействие
- объем используемой памяти

Этот язык часто применяют для программирования систем реального времени, для обеспечения работы информационно-измерительных комплексов, программирования низкоуровневых драйверов устройств.

# Машинно-ориентированное программирование

- характеризуется аппаратным подходом к организации работы компьютера, нацеленным на доступ к любым возможностям оборудования.
- В центре внимания – конфигурация оборудования, состояние памяти, команды, передачи управления, очередность событий, исключения и неожиданности, время реакции устройств и успешность реагирования

# Императивное (директивное) программирование

- Программа – это цепочка команд (директив), которая приводит к вычислению одной или нескольких искомым величин.
- Эти директивы совершенно однозначно и четко предписывают выполнение каждого шага алгоритма

# Процедурное программирование

- Императивное программирование стали называть *процедурным*, когда в процессе увеличения сложности моделируемых систем и размера получаемых программ возникла концепция подпрограмм, называемых также *процедурами* (procedure) или *функциями* (function)

# Структурное программирование

Программирование с использованием нескольких типов управляющих конструкций (структур), которые позволяют сильно повысить понимаемость логики работы программы конструкций называется *структурным*.

Доказано (Эдсгер В. Дейкстра, 1969), что  
**любая программа может быть  
построена на основе трех базовых  
конструкций:**

последовательное исполнение (линейная)

условная (ветвление)

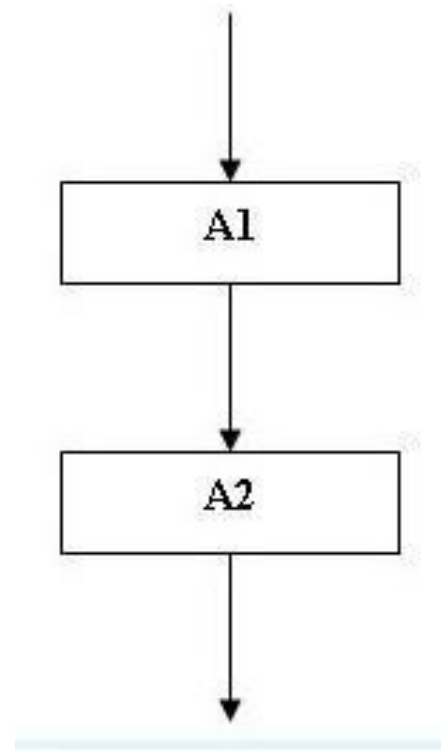
циклическая



# Основные конструкции структурного программирования

## 1. Линейная (функциональный блок).

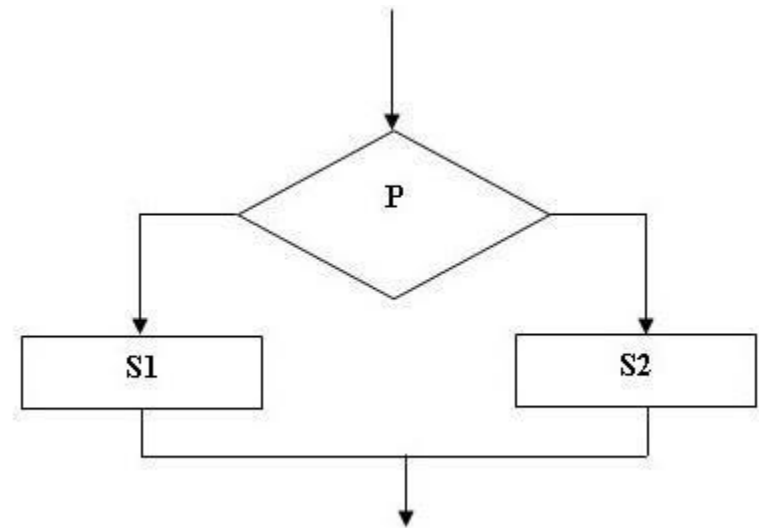
Операторы ввода, вывода и присваивания, следующие строго друг за другом.



# Основные конструкции структурного программирования

## 2. Условная или ветвление.

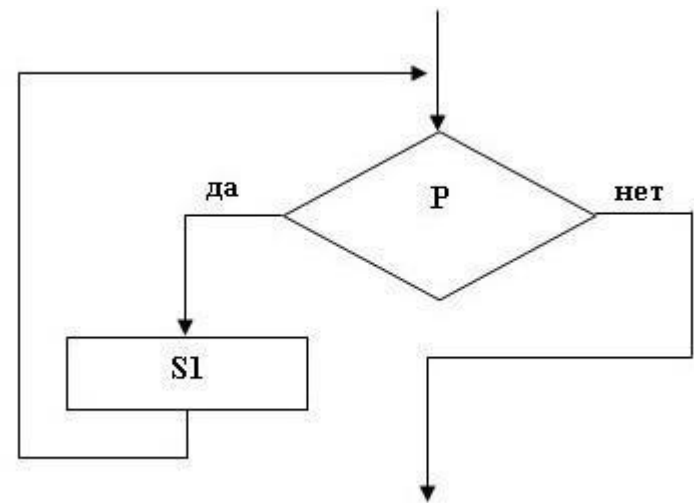
Предполагает проверку некоторого условия, в зависимости от которого выполняется то или иное действие



# Основные конструкции структурного программирования

## 3. Циклическая.

Обеспечивает многократное повторение набора операторов, пока не будет выполнено некоторое условие



# Стратегии разработки программ

1. **Нисходящее проектирование** (программирование «сверху-вниз»).

Разработка программы начинается с определения целей решения проблемы, после чего идет последовательная детализация, заканчивающаяся детальной программой.

2. **Восходящее проектирование** (программирование «снизу-вверх»)

Разработка программ, начинающаяся с разработки подпрограмм (процедур, функций), в то время когда проработка общей схемы не закончилась.

# Принципы структурного программирования

- Принцип **абстрактности**. Увеличение абстрактности программы с уменьшением деталей.
- Принцип **формальности** предполагает строгий методический подход к программированию
- Принцип **иерархического упорядочения**. Взаимосвязь между частями программы должна носить иерархический характер
- Принцип **модульности**. Программа разделяется на отдельные законченные фрагменты, модули, которые просты по управлению и допускают независимую отладку и тестирование.

# Первые языки высокого уровня. Фортран

В 1954 году публикуется сообщение о создании языка **FORTRAN** (FORmula TRANslation) (Фортран). Место рождения языка - штаб-квартира фирмы IBM в Нью-Йорке. Одним из главных разработчиков является **Джон Бэкус**. Он же стал автором НФБ (нормальная форма Бэкуса), которая используется для описания синтаксиса многих языков программирования.

Фортран был широко распространенным языком, особенно среди пользователей, которые занимались численным моделированием.

# Джон Бэкус



# Фортран

В 1977 году был принят новый стандарт языка Фортран-77.

Сейчас используется и стандарт Фортран-90.

Трансляторы: Watfor, Lap-Fortran, Fortran-77, MS-Fortran 5.0.

В настоящий момент Фортран наравне с C/C++ используется для параллельного программирования.



# *Бейсик.*

**BASIC (Beginner's All-Purpose Symbolic Instruction Code** – “универсальный символический код инструкций для начинающих”) разработан в Дартмутском университете в 1964 году под руководством **Джона Кемени** и **Томаса Курца**. Прямой наследник Фортрана.

Это простой язык, легко изучаемый, предназначенный для программирования несложных расчетных задач.

Компиляторы: GWBASIC, Turbo-Basic и Quick Basic.

**Quick Basic** - второе поколение систем программирования на языке Бейсик.

# VISUAL BASIC

Предоставляет возможность модульного и процедурного программирования, создания библиотек, компиляции готовых программ и прочее, что вывело его на уровень таких классических языков программирования, как Си, Паскаль, Фортран и др.

В 1991 году появилась первая версия языка VISUAL BASIC, которая, развиваясь, сильно потеснила Delphi-подобные системы.

Microsoft использовала Бейсик как основу для создания языка VBA, используемого для программирования макросов в офисных приложениях.

# Алгол

В период конца 50-х в Европе и в СССР популярен язык **ALGOL**. Как и FORTRAN, он ориентировался на математические задачи. В нем была реализована передовая для того времени технология программирования - **структурное программирование**.

Характерная черта первых языков программирования - **предметная ориентация**. Это значит, что каждый язык предназначался для решения какого-то определенного, достаточно ограниченного класса задач.

## PL/1.

PL/1 разработан в 1964-1965 годах фирмой IBM. PL/1 относится к числу универсальных языков, т. е. позволяет решать задачи из разных областей: численные расчеты, текстовая обработка, экономические задачи и т.д.

Из-за универсальности язык оказался слишком сложным и не вполне независимым от архитектуры машины, как следствие не получил повсеместного распространения.

# Паскаль.



Язык был разработан профессором кафедры вычислительной техники Швейцарского Федерального института технологии **Николаусом Виртом** в 1968 году как альтернатива существующим и все усложняющимся языкам программирования.

# Паскаль

1973 год - появление стандарта языка, а число трансляторов с этого языка в 1979 году перевалило за 80.

В начале 80-х годов - появление трансляторов MS-Pascal и Turbo-Pascal для ПЭВМ. С этого времени Паскаль становится одним из наиболее важных и широко используемых языков программирования

# Паскаль.

## Важнейшие особенности:

- воплощенная идея структурного программирования;
- концепция структуры данных как одного из фундаментальных понятий;

## Причины популярности:

- простота языка;
- развитые средства представления структур данных;
- Независимость от аппаратных средств
- наличие специальных методик создания трансляторов;

C.



Сотрудник фирмы Bell Labs **Денис Ритчи** создал язык Си в 1972 году во время совместной работы с Кеном Томпсоном, как инструментальное средство для реализации операционной системы Unix



## C.

В настоящее время любая инструментальная и операционная система не может считаться полной если в ее состав не входит компилятор языка Си.

Язык программирования Си был разработан как инструмент для программистов-практиков. В соответствии с этим главной целью его автора было создание удобного и полезного во всех отношениях языка.

Си является орудием системного программиста, требующим от программиста высокой дисциплины, но язык не строг в формальных претензиях и допускает краткие формулировки.

# C.

## Характеристики языка:

- соответствие современным требованиям. Его структура побуждает программиста использовать в своей работе нисходящее проектирование, структурное программирование и пошаговую разработку модулей.
- эффективность, программы на этом языке отличаются компактностью и быстротой исполнения.
- мобильность
- мощность
- гибкость
- универсальность

# Ада.

Язык Ада возник в результате международного конкурса языковых проектов проходившего в 1978-1979 годах под эгидой Министерства Обороны США. Целью конкурса - разработать единый язык программирования для так называемых встроенных систем. Имелись в виду прежде всего бортовые системы управления военными объектами

Интересно, что все языки, дошедшие до последних туров этого конкурса, были основаны на Паскале. В этой связи Аду можно предварительно охарактеризовать как развитый Паскаль. Авторы пошли в основном по пути расширения Паскаля новыми элементами. В результате получился существенно более сложный язык.

# Декларативная парадигма

- необходимость решения логических, интеллектуальных задач, а также задач, связанных с обработкой не только числовой информации, но и информации различных типов
- декларативные программы не предписывают выполнять определенную последовательность действий, в них лишь дается разрешение совершать их. Исполнитель сам находит способ достижения поставленной перед ним составителем программы (программистом) цели

# Функциональное программирование

- Функциональная программа состоит из совокупности определений функций, которые в свою очередь представляют собой вызовы других функций и предложений, управляющих последовательностью вызовов. При этом функции часто либо прямо, либо опосредованно вызывают сами себя (рекурсия).
- Каждая функция возвращает некоторое значение в вызвавшую его функцию, вычисление которой после этого продолжается; этот процесс повторяется до тех пор, пока начавшая процесс вычислений функция не вернет конечный результат пользователю

# Языки функционального программирования. Лисп.



Язык Лисп (Lisp - LISt Processing) был предложен **Джоном Маккарти** в 1960 году и ориентирован на разработку программ для решения задач не численного характера.

# Лисп

Изначально предназначен для программирования и разработки систем искусственного интеллекта. В основу языка легло т.н. лямбда-исчисление.

Программирование ведется с помощью функций. Причем функция понимается как правило, сопоставляющее элементам некоторого класса соответствующие элементы другого класса.

# Лисп.

На протяжении почти сорокалетней истории существования Лиспа появился ряд диалектов этого языка: Common LISP, Mac LISP, Inter LISP, Standard LISP и др. Различия между ними не носят принципиального характера и в основном сводятся к несколько отличающемуся набору встроенных функций и некоторой разнице в форме записи программ.



# Логическое программирование

- Основано на логике предикатов
- Программа рассматривается как набор логических фактов и правил вывода, а выполнение программы состоит в вычислении истинности (попытке доказательства) некоторого утверждения

# Языки логического программирования. Пролог.

Одной из уникальных особенностей языка ПРОЛОГ является то что его появление является результатом исследований ученых нескольких университетов и научных центров, а не разработка конкретного человека или группы людей.

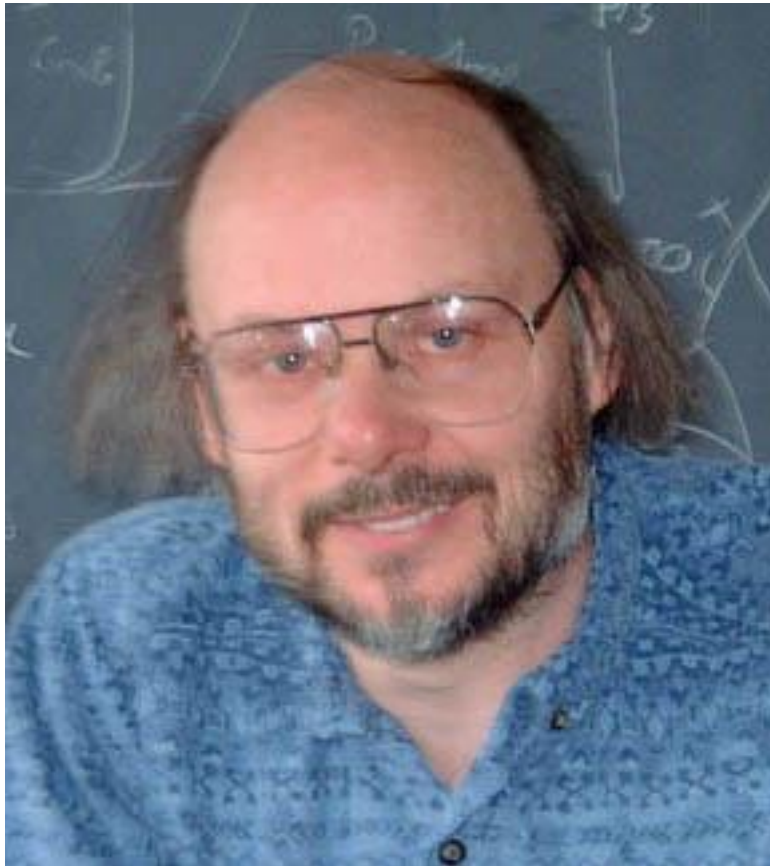
Пролог - язык логического программирования предназначенный для представления и использования знаний о некоторой предметной области.

Основой языка послужила математическая логика. В Прологе реализован декларативный подход, при котором достаточно описать задачу с помощью правил и утверждений относительно заданных объектов. Если это описание является достаточно точным, то ЭВМ может самостоятельно найти требуемое решение.

# Объектно-ориентированное программирование

- Объектно-ориентированное программирование (ООП) есть по сути императивное программирование, дополненное принципами
- инкапсуляции данных и методов в **объект** (принцип модульности)
- **наследования** (принципом повторного использования разработанного функционала).
- **полиморфизма**

# Языки объектно-ориентированного программирования. C++.



Язык C++ появился в начале 80-х годов. Создан **Бьерном Страуструпом** с первоначальной целью избавить себя и своих друзей от программирования на ассемблере, Си или различных других языках **высокого уровня**.

# C++

Больше всего C++ позаимствовал из языка Си, а также из непосредственного его предшественника языка BCPL.

В первую очередь C++ отличается от Си разной степенью внимания к типам и структурам данных. Это связано с появлением понятий класса, производного класса и виртуальной функции.

# C++.

Язык C++ является средством объектного программирования, широко применяемой методики проектирования и реализации программ, практически заменившую традиционное процедурное программирование.

Абстракция, реализация, наследование и полиморфизм являются необходимыми свойствами которыми обладает язык C++.

# Визуальное программирование

- Возможность автоматической генерации программного кода
- Особенно эффективно при создании интерфейсной части приложений (диалоговых окон, командных кнопок и т.п.)

# Параллельное программирование

- В отличие от программирования последовательных вычислений, концептуальную основу которого составляет понятие алгоритма, реализуемого по шагам строго последовательно во времени, в параллельном программировании программа порождает совокупность параллельно протекающих процессов обработки информации, полностью независимых или связанных между собой статическими или динамическими пространственно-временными или причинно-следственными отношениями.



# Параллельное программирование

- Определение параллелизма: анализ задачи с целью выделить подзадачи, которые могут выполняться одновременно;
- Выявление параллелизма: изменение структуры задачи таким образом, чтобы можно было эффективно выполнять подзадачи.
- Выражение параллелизма: реализация параллельного алгоритма в исходном коде с помощью систем параллельного программирования.

# Системы параллельного программирования

- OpenMP (разработан стандарт для языков Fortran, C и C++);
- PVM (Parallel Virtual Machine), (поддерживает языки Fortran, C, C++, имеет средства сопряжения с языками Perl, Java);
- CUDA (Compute Unified Device Architecture) – программно-аппаратное решение, позволяющее использовать видеопроцессоры NVIDIA для вычислений общего назначения;
- MPI – интерфейс передачи данных (message passing interface).

# Сценарное программирование

- Возникшее, с появлением глобальной сети Internet, Web-программирование и языки его реализующие, определяют формирование современной, сценарной технологии программирования.
- Первое из средств создания Web-страничек – это язык разметки гипертекстов HTML (HyperText Markup Language)
- Для создания более сложных и функциональных интерактивных сайтов сейчас имеется довольно разнообразный выбор инструментальных средств – ASP, PHP, Perl, Python, JavaScript и т.п

# Визуальные средства создания Web-сайтов

- Визуальные редакторы веб-интерфейсов, в которых содержание отображается в процессе редактирования и выглядит максимально близко похожим на конечную продукцию (Microsoft FrontPage, Web Page Maker, Adobe Dreamweaver и т.п.)
- Системы управления сайтом (CMS - Content Management System) – программное обеспечение, позволяющее управлять содержимым и структурой сайта; движки, в которых присутствует набор готовых шаблонов сайтов (Joomla, Drupal и т.п.)

# Язык разметки гипертекстов HTML

- разработан британским учёным Тимом Бернерсом-Ли в 1986—1991 годах в стенах ЦЕРНа



- (CERN от фр. *Conseil Européen pour la Recherche Nucléaire* (Европейский совет по ядерным исследованиям), лаборатория которого находится в Женеве (Швейцария))

Языки							Парадигмы и технологии
Ассемблер							Машинно-ориентированная
Fortran	Algol C	Pascal	Modula	Oberon			Императивная
	Basic		Ada				
	LISP	ML, Scheme	Prolog	Perl	Haskell		Декларативная
			VisualProlog				
				VisualBasic, C++, Delphi	Java, C#		Объектно-ориентированная
			Smalltalk		Ruby		
				Perl	Python		Сценарная
					PHP, ASP		
1950	1960	1970	1980	1990	2000		

# Основные этапы решения задач на компьютере

1. постановка задачи
2. математическое моделирование
3. алгоритмизация
4. программирование
5. трансляция программы
6. тестирование и отладка программы
7. исполнение отлаженной программы и анализ результатов

# Первый этап - постановка задачи

- словесное описание содержания задачи, общий подход к ее решению.
- Для нашей задачи – решение квадратного уравнения - можно предложить такое описание: даны коэффициенты уравнения - три целых числа, вычислить значения корней уравнения - одно или два числа, вывести их в качестве результата



# Второй этап - математическое моделирование

- Математические формулы и логические условия,
- Для нашей задачи этот этап – это известные формулы для вычисления дискриминанта и корней квадратного уравнения

# Третий этап - алгоритмизация

**Алгоритм** — понятное и точное предписание исполнителю выполнить конечную последовательность команд, приводящих от исходных данных к искомому результату.

# *Свойства алгоритма*

1. Дискретность.
2. Детерминированность.
3. Точность.
4. Понятность.
5. Результативность.
6. Конечность.
7. Массовость.

# Основные алгоритмические конструкции

- **последовательная** - каждый шаг алгоритма выполняется один раз, причем после каждого  $i$ -го шага выполняется  $(i + 1)$ -й шаг, если  $i$ -й шаг – не конец алгоритма.
  - **ветвящаяся** - последовательность выполнения шагов алгоритма зависит от входных данных.
  - **циклическая** - некоторая, подряд идущая группа шагов алгоритма может выполняться несколько раз в зависимости от входных данных.
- достаточно для записи любого алгоритма  
(1969 г. Э.В. Дейкстра, статья «Структуры данных и алгоритмы»)

# *Способы записи алгоритма*

- **словесная** (записи на естественном языке);
- **псевдокоды** (полуформализованные описания алгоритмов на условном алгоритмическом языке, включающие в себя как элементы языка программирования, так и фразы естественного языка, общепринятые математические обозначения и др.);
- **графическая** (изображения из графических символов, блок-схемы);
- **формальная** (машина Тьюринга или машина Поста);
- **программная** (тексты на языках программирования)

# Блок-схемы

Одна из универсальных форм записи алгоритмов, допускающая использование двух видов блоков: операторный и условный.

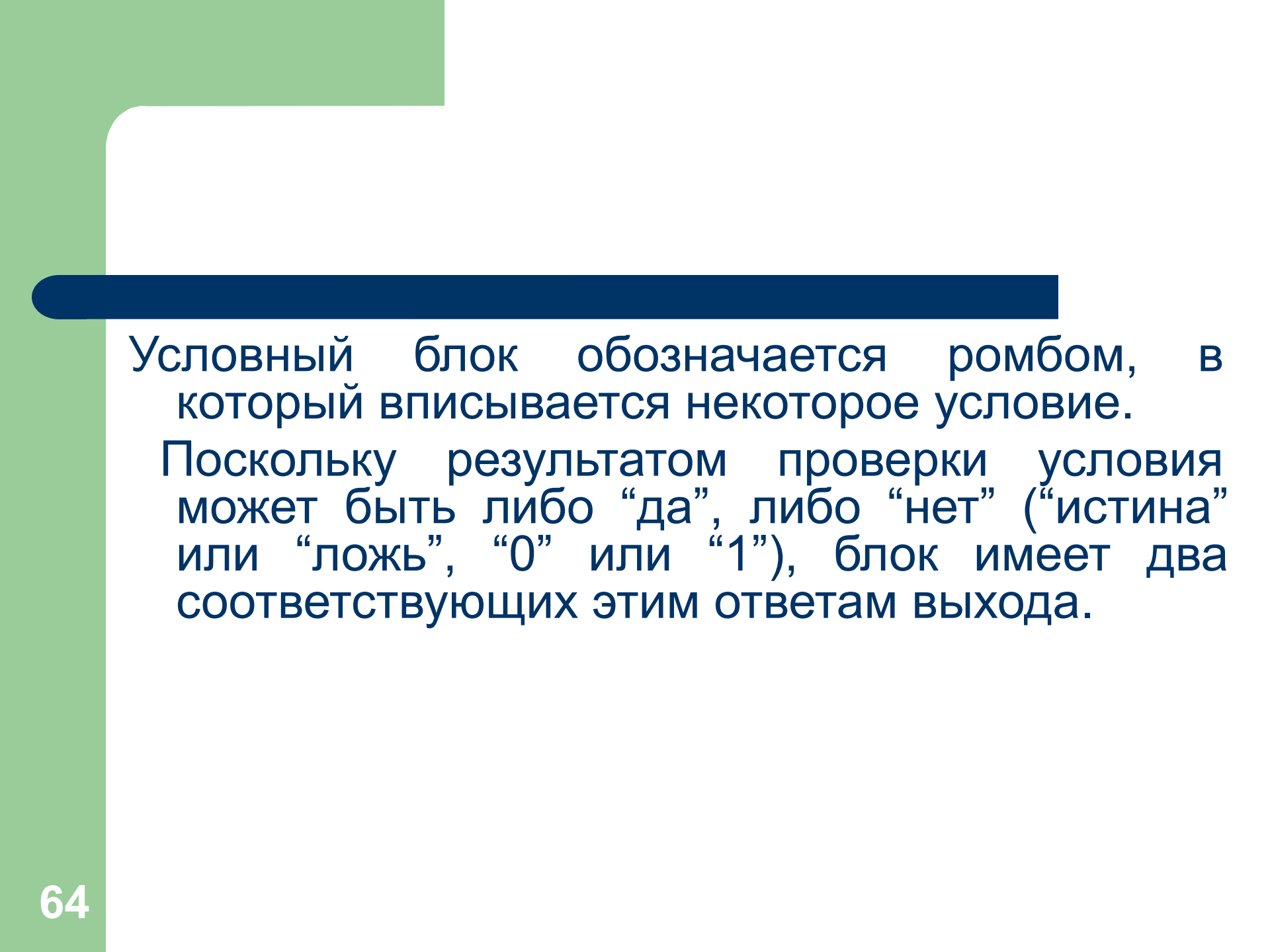
Операторный блок – прямоугольник.

Условный блок – ромб.

Внутри блоков записывают соответствующие действия или условия.

**Операторный блок** – это прямоугольник, в который вписывается некоторое действие или выражение.

Этот блок может иметь несколько входов и только один выход, что обеспечивает однозначность в определении последовательности выполняемых действий.

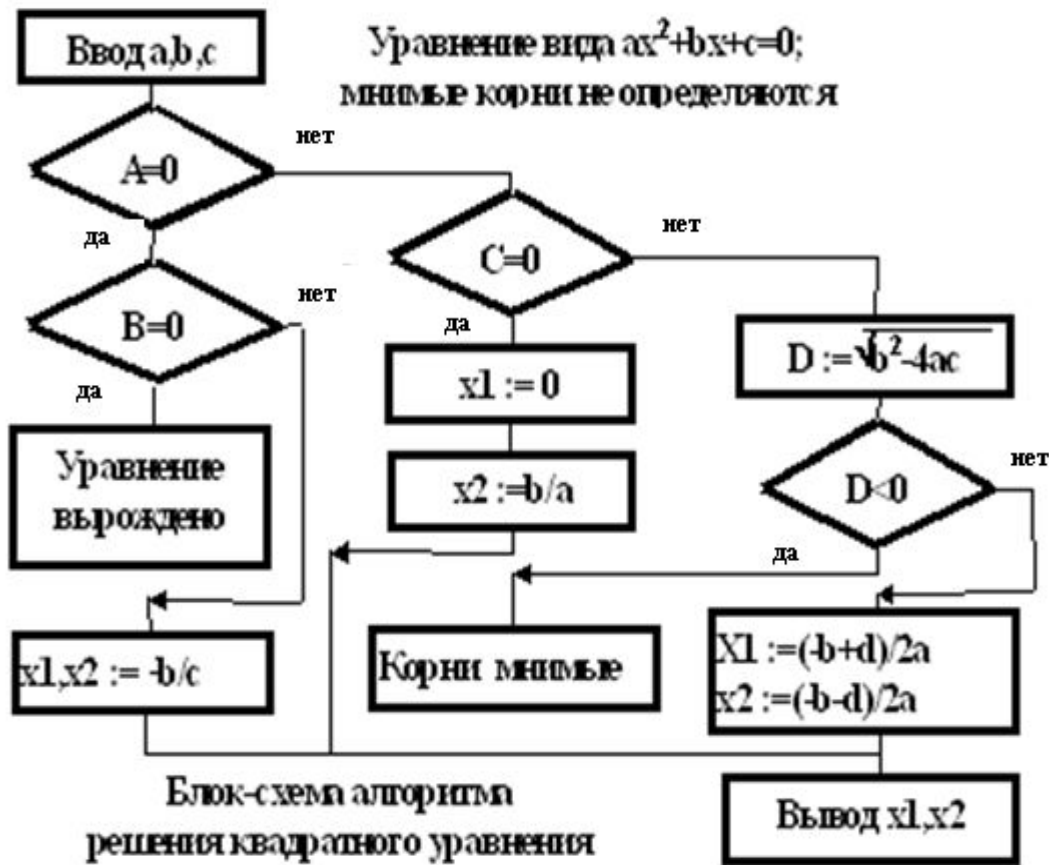


Условный блок обозначается ромбом, в который вписывается некоторое условие.

Поскольку результатом проверки условия может быть либо “да”, либо “нет” (“истина” или “ложь”, “0” или “1”), блок имеет два соответствующих этим ответам выхода.



# Блок-схема алгоритма поиска корней квадратного уравнения



## Четвёртый этап – *программирование*

- ***Программа*** – описание структуры алгоритма на языке программирования.
- Программирование включает в себя следующие виды работ: выбор языка программирования; уточнение способов организации данных; запись алгоритма на выбранном языке программирования

## Пятый этап – *трансляция программы*

- Трансляция - это перевод команд языка программирования в компьютерные двоичные коды.
- На этом этапе происходит проверка программы на ее соответствие правилам (синтаксису) языка программирования и при отсутствии синтаксических ошибок создается исполняемый файл программы

## Шестой этап – *тестирование и отладка программы*

- Тестирование - выполнение программы, проверка программы на наличие логических ошибок. Для этого нужно подобрать систему тестов (набор исходных данных с заранее известным результатом) и сравнить выдаваемые программой результаты с контрольными
- Отладка – проверка работы программы на контрольных примерах. Контрольные примеры – это различные (желательно все возможные) комбинации исходных данных. Контрольные примеры выбираются так, чтобы при работе были задействованы все ветви алгоритма

## **Седьмой этап – *исполнение отлаженной программы и анализ результатов***

- Программист запускает программу и задаёт исходные данные, требуемые по условию задачи.
- Постановщик задачи анализирует полученные результаты, на основании анализа принимаются решения, вырабатываются рекомендации, делаются выводы.

# Итоги

Мы рассмотрели

- Основные парадигмы и технологии программирования
- Этапы решения задачи средствами вычислительной техники
- Понятие алгоритма и его свойства
- Способы записи алгоритмов