



***ПРОГРАММИРОВАНИЕ НА
ЯЗЫКЕ ВЫСОКОГО
УРОВНЯ ПАСКАЛЬ***



Список литературы:

1. И.А. Полетаев, Д.И. Полетаев, О.А.Полетаева Программирование на языке высокого уровня Паскаль Учебное пособие – Псков, ППИ, 2010.
2. И.А. Полетаев, О.А.Полетаева Алгоритмический язык Паскаль Методические указания к лабораторным работам – СПб/Псков, СПбГПУ, 2002.
3. Турбо Паскаль 7.0. Самоучитель. – СПб.: Питер; К.: Издательская группа ВHV, 2002.

Основные сведения об алгоритмах

- **Алгоритм** – это формальное предписание, однозначно определяющее содержание и последовательность операций, переводящих совокупность исходных данных в искомый результат – решение задачи.

С алгоритмами связаны следующие области исследований :

1. **Анализ алгоритмов.** Предмет этой области состоит в том, чтобы для заданного алгоритма определить рабочие характеристики. Например, часто требуется, чтобы алгоритм был быстрым.
2. **Теория алгоритмов.** К этой области относятся вопросы существования или отсутствия эффективных алгоритмов вычисления определенных величин.
3. **Построение алгоритмов.** В этой области рассматриваются стандартные приемы и методы, используемые при создании алгоритмов.

Свойства алгоритма:

1. **Конечность (результативность, финитность).** Свойство так определять процесс преобразования исходных данных, чтобы он через конечное число шагов для любых допустимых исходных данных приводил к искомому результату.
2. **Определенность (детерминированность).** Предполагает такое составление алгоритма, которое не допускает различных толкований или искажения результата.
3. **Ввод (массовость, наличие входных данных).** Определяет возможность использования любых исходных данных из некоторого определенного множества для однотипных задач.
4. **Вывод (наличие выходных данных).** Алгоритм имеет одно или несколько выходных данных, имеющих определенную связь с входными данными.
5. **Эффективность.** Алгоритм считается эффективным, если его операторы достаточно просты для того, чтобы их можно было точно выполнить в течение конечного промежутка времени.
6. **Направленность.** Означает наличие способа однозначного перехода от одного действия к другому.
7. **Дискретность.** Свойство, означающее, что алгоритм разбивается на последовательные команды, возможность выполнения которых человеком или машиной (исполнителем) не вызывает сомнений.
8. **Понятность.** Означает, что все команды алгоритма должны быть понятны для конкретных исполнителей.

Формы записи алгоритмов

Основные формы записи алгоритмов:

- словесная (текстуальная) запись алгоритма;
- описание на алгоритмическом языке;
- структурная схема (графическая схема).

Другие формы записи алгоритмов :

- операторный способ;
- с использованием диаграммы Нэсси-Шнейдермана;
- с использованием Р-схемы;
- с помощью псевдокода.

Словесное описание алгоритма

Словесное описание алгоритма представляет собой текст, в котором на различном разговорном языке (например, на русском) по пунктам записана последовательность действий. Строгие требования к форме такой записи не предъявляются, но существуют определенные правила, выполнение которых облегчает понимание алгоритма. Все действия расписываются по шагам или нумеруются, чтобы было удобно ссылаться при необходимости по номеру шага или пункта. Начало алгоритма и его окончание принято отмечать словами «начало» и «конец». Можно указывать в одном пункте не одно действие, а группу простых действий.

Пример: Составить словесное описание алгоритма решения уравнения $x+2 = 3x-4$.

Для наглядности каждый пункт снабдим записью результата.

■ **Алгоритм 1.**

Шаг1. Начало.

Шаг2. Перенести слагаемое $3x$ в левую часть уравнения. $x-3x+2 = -4$

Шаг3. Перенести слагаемое 2 в правую часть $x-3x = -2-4$

Шаг4. Привести подобные слагаемые в левой части $-2x = -2-4$

Шаг5. Привести подобные слагаемые в правой части $-2x = -6$

Шаг6. Разделить обе части уравнения на -2 . $x = 3$

Шаг7. Записать результат. Ответ: $x = 3$

Шаг8. Конец.

Описание алгоритма на алгоритмическом языке.

Алгоритмический язык — это система обозначений и правил для единообразной и точной записи алгоритмов и их исполнения.

Алгоритмический язык имеет свой **словарь**, основу которого составляют слова, употребляемые для записи команд, входящих в **систему команд исполнителя** алгоритмов. Такие команды называются **простыми командами**.

Ограниченное число слов, смысл и способ употребления которых задан раз и навсегда, называются **служебными словами**. При записи алгоритмов они выделяются и могут записываться **в сокращенной форме**.

Использование служебных слов делает запись алгоритмов более наглядной, а формы представления алгоритмов — более единообразными. Команды при этом записываются последовательно.

Пример построения алгоритма на школьном алгоритмическом языке *КуМир*.

Пример: Правописание приставок на «з» и «с»

Алгоритм 2.

- нач
- если корень слова начинается со звонкой согласной
- I то на конце приставки написать «з»
- I иначе на конце приставки написать «с»
- все
- конец

Структурное схема алгоритма (графическая схема алгоритма).

Алгоритм наиболее удобно изображать графически с помощью **блок-схем** или **граф-схем**.

Блок-схемы — это набор элементов, называемых блоками, соединенных между собой линиями или стрелками. Линии называются **линиями потока**. Они отражают последовательность выполнения действий, определяемых каждым блоком.

Представление алгоритмов в виде блок-схем строго регламентировано, так как оно должно соответствовать стандартам. Основное направление потока информации идет сверху вниз и слева направо, здесь стрелки на линиях потока можно не указывать. В остальных случаях наличие стрелок обязательно. Блоки можно нумеровать. Порядковые номера проставляются в верхней левой части блока в разрыве его контура.

В настоящее время в вычислительной технике для записи алгоритмов используется «ГОСТ 19.701-90. Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения». Он входит в группу стандартов единой системы программной документации (ЕСПД). Этот стандарт введен взамен «ГОСТ 19.002-80. Схемы алгоритмов и программ. Правила выполнения. ГОСТ 19.003-80. Схемы алгоритмов и программ. Обозначения условные графические». Но стандарт от 1990 года не совсем полный, в частности, там отсутствуют размеры и отношения сторон блоков, изменены название и трактовка некоторых из них. Поэтому совместно с ГОСТ 19.701-90 рекомендуется использовать и ГОСТ 19.002-80, ГОСТ 19.003-80.

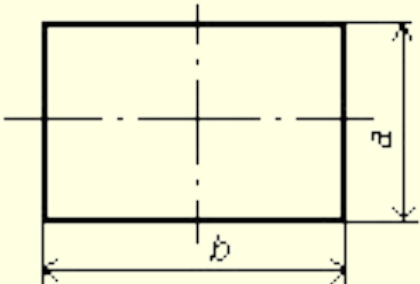
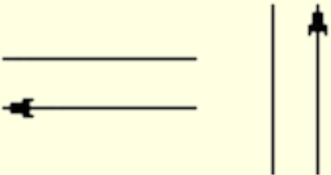
Схема программы.

Схемы программ отображают последовательность операций в программе.

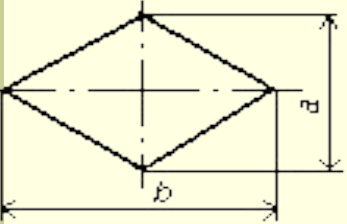
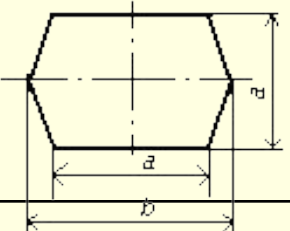
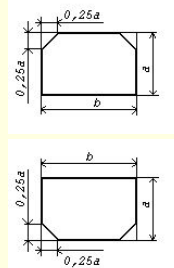
Схема программы состоит из:

- 1) **символов (т.е. блоков) процесса**, указывающих фактические операции обработки данных (включая символы, определяющие путь, которого следует придерживаться с учетом логических условий);
- 2) **линейных символов**, указывающих поток управления;
- 3) **специальных символов**, используемых для облегчения написания и чтения схемы.

Описание основных символов по ГОСТ 19.701-90 и по ГОСТ 19.003-80

Обозначение и размеры	Наименование и функция по ГОСТ 19.701-90	Наименование и функция по ГОСТ 19.003-80
	<p>Процесс. Символ отображает функцию обработки данных любого вида (выполнение операции или группы операций, приводящие к изменению значения, формы или размещения информации или к определению, по которому из нескольких направлений потока следует двигаться).</p>	<p>Процесс. Выполнение операций или группы операций, в результате которых изменяется значение, форма представления или расположение данных.</p>
	<p>Линия. Символ отображает поток данных или управления. При необходимости или для повышения удобочитаемости могут быть добавлены стрелки-указатели.</p>	<p>Линия потока. Указание последовательности перехода между символами. Направления линии потока сверху вниз и слева направо принимают за основные и, если линии потока не имеют изломов, стрелками можно не обозначать. В остальных случаях направление линии потока обозначать стрелкой</p>

Описание основных символов по ГОСТ 19.701-90 и по ГОСТ 19.003-80

	<p>Решение.</p> <p>Символ отображает решение или функцию переключательного типа, имеющую один вход и ряд альтернативных выходов, один и только один из которых может быть активизирован после вычисления условий, определенных внутри этого символа. Соответствующие результаты вычисления могут быть записаны по соседству с линиями, отображающими эти пути.</p>	<p>Решение.</p> <p>Выбор направления выполнения алгоритма или программы в зависимости от некоторых переменных условий.</p>
	<p>Подготовка.</p> <p>Символ отображает модификацию команды или группы команд с целью воздействия на некоторую последующую функцию (установка переключателя, модификация индексного регистра или инициализация программы).</p>	<p>Модификация.</p> <p>Выполнение операций, меняющих команды или группу команд, изменяющих программу.</p>
	<p>Граница цикла.</p> <p>Символ, состоящий из двух частей, отображает начало и конец цикла. Обе части символа имеют один и тот же идентификатор. Условия для инициализации, приращения, завершения и т.д. помещаются внутри символа вначале или в конце в зависимости от расположения операции, проверяющей условие.</p>	<p>(отсутствует)</p>

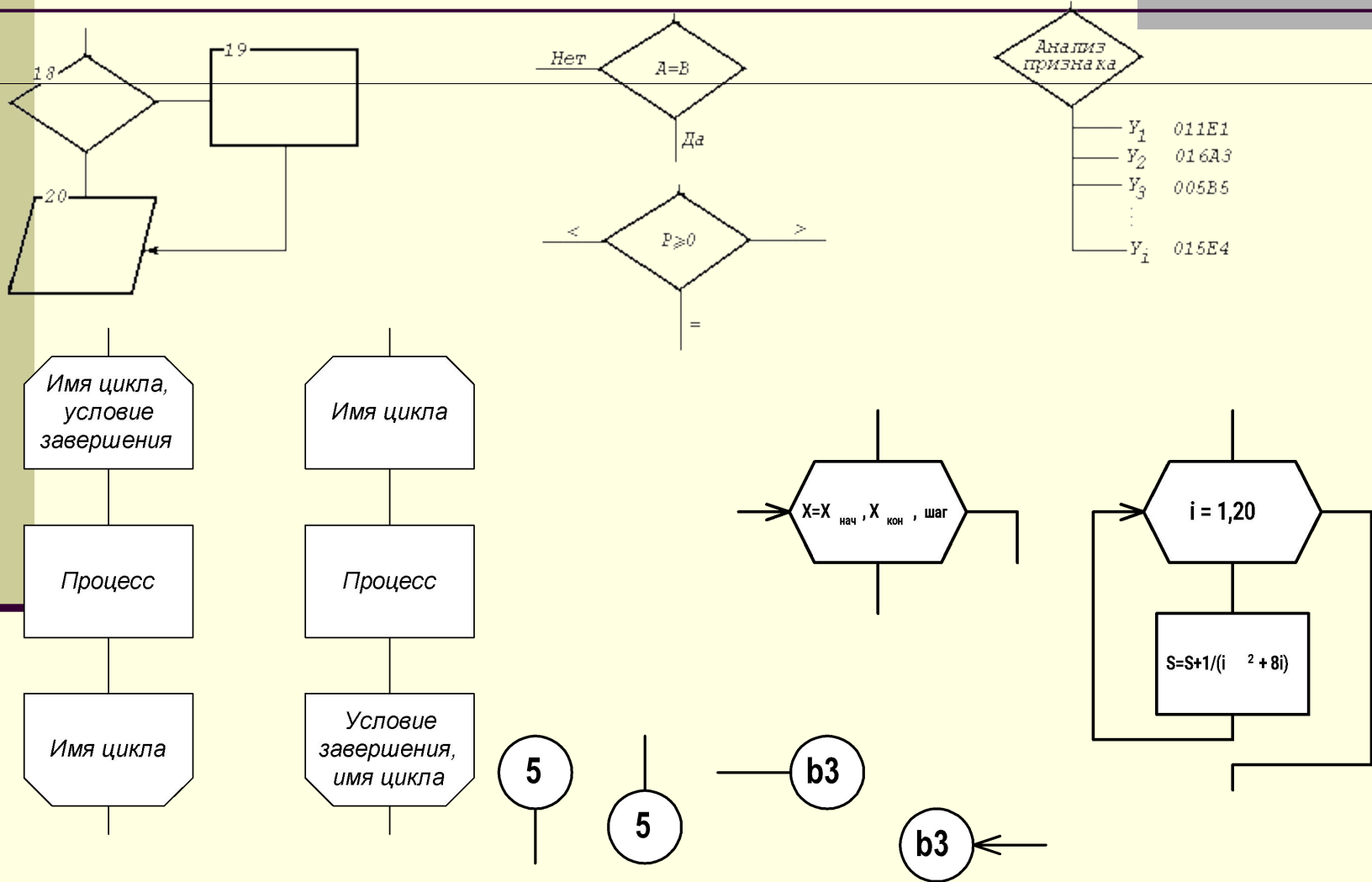
Описание основных символов по ГОСТ 19.701-90 и по ГОСТ 19.003-80

	<p>Данные. Символ отображает данные, носитель данных не определен.</p>	<p>Ввод-вывод. Преобразование данных в форму, пригодную для обработки (ввод) или отображения результатов обработки (вывод)</p>
	<p>Предопределенный процесс. Символ отображает предопределенный процесс, состоящий из одной или нескольких операций или шагов программы, которые определены в другом месте (в программе, модуле).</p>	<p>Предопределенный процесс. Использование ранее созданных и отдельно описанных алгоритмов или программ.</p>
	<p>Терминатор. Символ отображает выход во внешнюю среду и вход из внешней среды (начало или конец схемы программы, внешнее использование и источник или пункт назначения данных).</p>	<p>Пуск – останов. Начало, конец, прерывание процесса обработки данных или выполнения программы.</p>

Еще несколько правил

- Расстояния между параллельными линиями потока должно быть не менее 3 мм, между остальными символами(блоками) схемы - не менее 5 мм.
- Размер a должен выбираться из ряда 10, 15, 20 мм. Допускается увеличивать размер a на число, кратное 5. Размер b равен $1,5a$.
- Записи внутри символа(блока) или рядом с ним должны быть краткими. Сокращение слов и аббревиатуры, за исключением установленных государственными стандартами, должны быть расшифрованы в нижней части поля схемы или в документе, к которому эта схема относится.
- Координаты зоны символа(блока) или порядковый номер проставляют в верхней части символа в разрыве его контура.
- При большом количестве блоков или линий связи линии потока можно прерывать, используя «соединители», изображаемые в виде круга. Внутри круга ставятся цифры или комбинации букв и цифр, но одинаковые в начале и в конце обрыва линии потока.

Примеры фрагментов алгоритмов



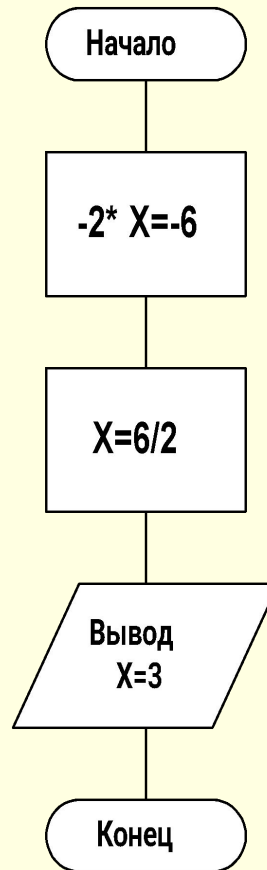
ОСНОВНЫЕ АЛГОРИТМИЧЕСКИЕ КОНСТРУКЦИИ.

Алгоритмические конструкции можно разделить на три основных типа: **линейная, разветвляющаяся и циклическая.**

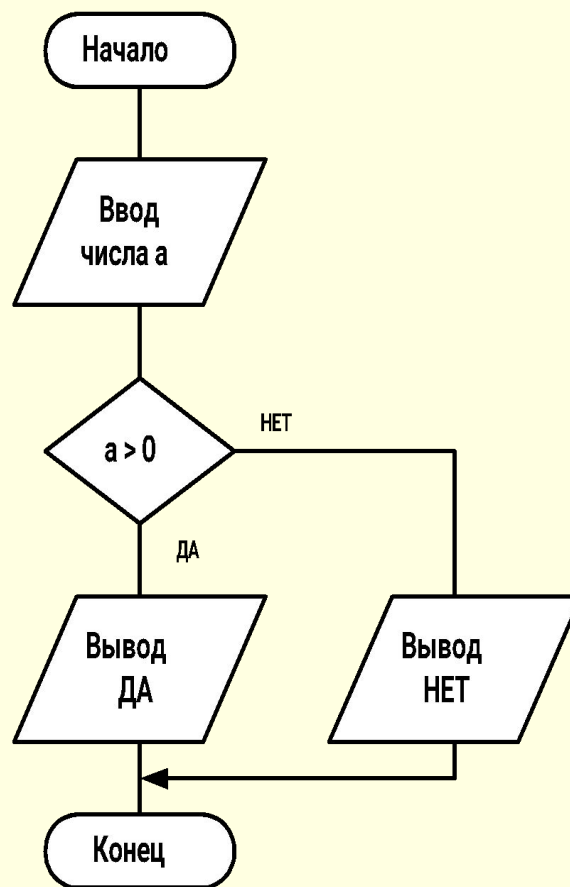
- Наиболее простым является **линейный алгоритм**, содержащий набор шагов, выполняемых один за другим. Линейный алгоритм состоит из блоков, соединенных последовательно.
- Более сложным является **разветвляющийся (ветвящийся) алгоритм**, имеющий несколько вариантов выполнения, реализуемых в зависимости от удовлетворения каких-либо логических условий.
- **Циклическая конструкция алгоритма** предусматривает наличие цикла. Цикл – это многократно выполняемый участок алгоритма. Циклический алгоритм при каждом исполнении предписывает многократное выполнение одной и той же последовательности действий.

запись алгоритма решения линейного уравнения

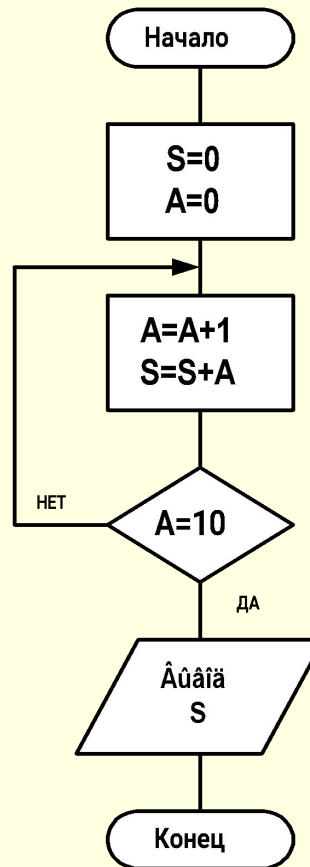
$x+2 = 3x-4$ в виде блок-схемы



Определить, является ли число **a** положительным



Вычислить сумму целых чисел от 1 до 10.



. Этапы разработки программного обеспечения

Профессиональное программирование подразумевает, что результатом труда, – программным продуктом, – будет пользоваться определенный круг людей, пользователей. На этапе разработки программы, в которой может участвовать группа людей, пользователей представляет **Заказчик**.

Для выполнения задачи создания и эксплуатации программного обеспечения ее разбивают на определенные этапы:

1. Постановка задачи.
2. Составление алгоритма.
3. Составление и ввод программы.
4. Отладка и тестирование программы.
5. Сопровождение программного продукта.

1. Постановка задачи.

Создание любой программы начинается с **постановки задачи**. Изначально задача ставится в терминах некоторой предметной области, и необходимо перевести ее в понятия и выражения, более близкие к программированию. Поскольку программист первоначально редко досконально разбирается в предметной области, а Заказчик – в программировании, то постановка задачи может стать весьма непростым итерационным процессом.

Постановка задачи заканчивается созданием технического задания, а затем и **внешней спецификации программы**, которая включает в себя:

1. Описание исходных данных и результатов (виды, представление, точность, ограничения и т.п.).
2. Описание задачи, реализуемой программой.
3. Способ обращения к программе.
4. Описание возможных особых и аварийных ситуаций и ошибок пользователя.
5. На этом этапе программа рассматривается как «черный ящик», для которого определяется выполняемая им функция, входные и выходные данные. Каким образом достигается выполнение функций, здесь не указывается.

2. Составление алгоритма.

3. Составление и ввод программы.

- **На втором этапе** разрабатываются алгоритмы, задаваемые спецификациями, и формируется (проектируется) общая структура программ. Здесь обычно применяется технология нисходящего проектирования с использованием **метода пошаговой детализации**. То есть сначала составляется укрупненный алгоритм в самом общем виде. Затем уточняются шаги (блоки) с более подробным описанием. На этом этапе описания производятся на языке, понятном человеку, используя определенную форму записи алгоритма. В программировании широко используется графическая форма записи в виде блок-схем или граф-схем.
- **Третий этап** как раз и является непосредственно программированием на языке, понятном ЭВМ. По своей сути третий этап является продолжением второго, так как программа тоже есть форма записи алгоритма с максимальной степенью детализации, – программная.

4. Отладка и тестирование программы.

5. Сопровождение программного продукта.

- **Четвертый этап** подразумевает устранение всех ошибок и недопониманий, возникших на предыдущих этапах.

Существуют самые разнообразные методы и рекомендации по тестированию и отладке.

Тестирование представляет собой процесс, посредством которого проверяется правильность функционирования программы и соответствие всем проектным спецификациям. В частности, для этих целей создается набор тестов.

Отладка – процесс исправления ошибок в программе. Так, при отладке исправляются синтаксические ошибки, алгоритмические, ошибки, обнаруженные при тестировании и другие.

- **Пятый этап** наступает, когда программный продукт сдан в эксплуатацию (или начались его продажи). Здесь так же возможно обнаружение не найденных на этапе тестирования ошибок, – их необходимо локализовать и исправить. Кроме этого, возможно изменение свойств программы для удобства эксплуатации: элементов интерфейса, некоторых функций и т.д. Кажется бы, пятый этап самый простой. Но ему отводится самая большая часть затрат времени и средств: до половины и более.

жизненный цикл программы

1. Возникновение и исследование идеи
2. Управление
3. Анализ требований
4. Проектирование
5. Программирование
6. Тестирование и отладка
7. Ввод в действие
8. Эксплуатация и сопровождение
9. Завершение эксплуатации