Понятие и свойства алгоритма

Лекция 10

Этапы решения задачи на компьютере

1. Постановка задачи:

- **сбор информации о задаче;**
- **формулировка условия задачи;**
- **определение конечных целей решения задачи;**
- **определение формы выдачи результатов;**
- **описание данных (их типов, диапазонов величин, структуры и т. п.).**

2. Анализ и исследование задачи, модели:

- **анализ существующих аналогов;**
- **анализ технических и программных средств;**
- **разработка математической модели;**
- **разработка структур данных.**

3. Разработка алгоритма:

- выбор метода проектирования алгоритма;
- выбор формы записи алгоритма (блок-схемы, псевдокод и др.);
- выбор тестов и метода тестирования;
- проектирование алгоритма.

4. Программирование:

- выбор языка программирования;
- уточнение способов организации данных;
- запись алгоритма на выбранном языке программирования.

С первого раза ничего не работает…

5. Тестирование и отладка:

- синтаксическая отладка;
- отладка семантики и логической структуры;
- тестовые расчеты и анализ результатов тестирования;
- совершенствование программы.

6. Анализ результатов решения задачи и уточнение в случае необходимости математической модели с повторным выполнением этапов 2…5.

7. Сопровождение программы:

- **доработка программы для решения конкретных задач;**
- **составление документации к решенной задаче, к математической модели, к алгоритму, к программе, к набору тестов, к использованию.**

Какие несчастные создания этим занимаются?

Системный программист (system programmer) - занимается разработкой, эксплуатацией и сопровождением системного программного обеспечения, поддерживающего работоспособность компьютера и создающего среду для создания и выполнения программ.

Прикладной программист (application programmer) — осуществляет разработку и отладку программ для решения функциональных задач (т. е. задач по реализации функций управления в рамках информационной системы — например, управление деятельностью предприятия, управление поставками, планирование выпуска продукции).

Программист-аналитик (analyst) — программист, анализирующий и проектирующий комплекс взаимосвязанных программ.

Постановщик задач — разработчик формальных постановок задач, требующих реализации на ЭВМ.

Системный администратор (system administrator) — человек, который обеспечивает организационную поддержку работы локальной сети и имеющегося программного обеспечения.

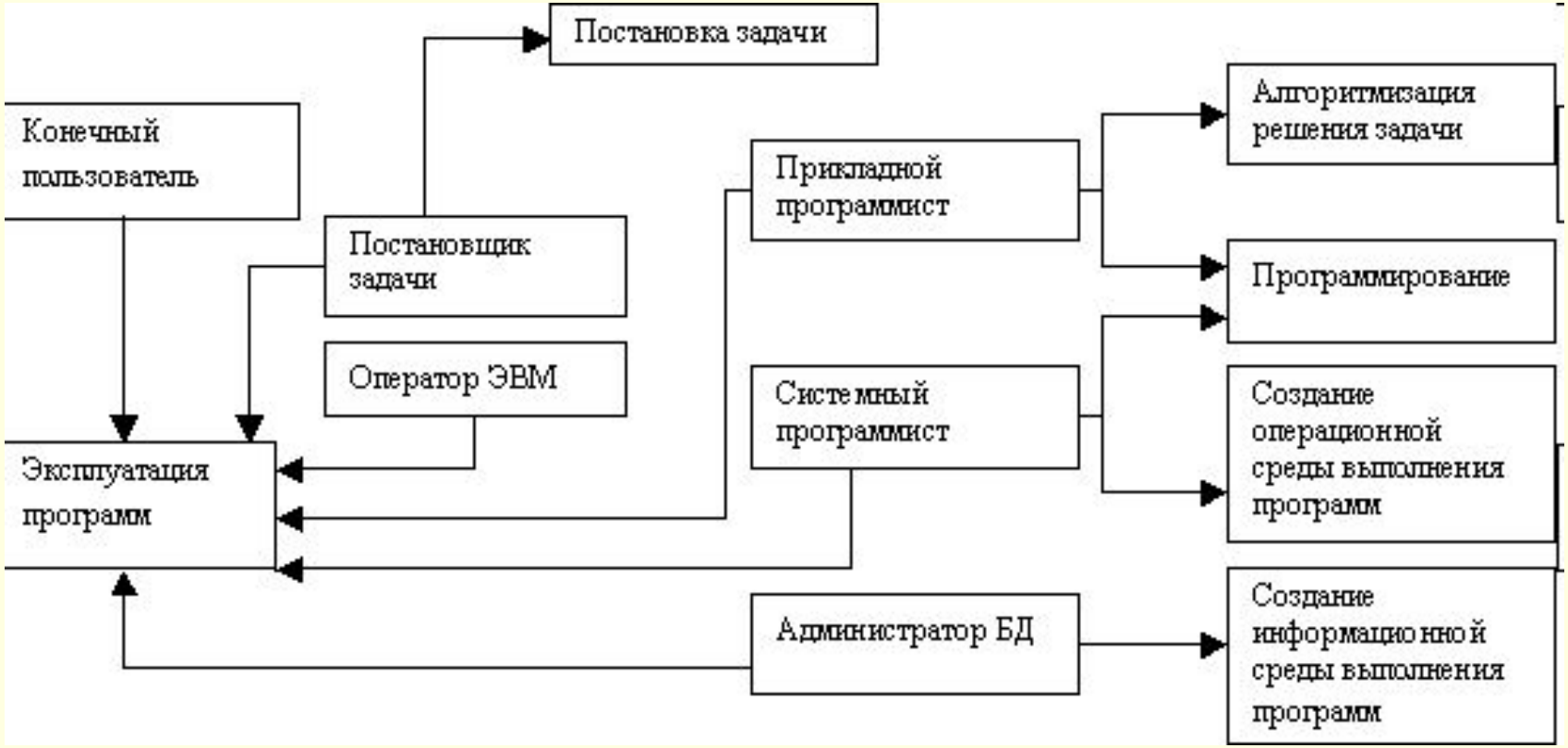
Основным потребителем программ является конечный пользователь (end user), который, как правило, не является специалистом в области программирования.

Ужасы из жизни ламеров

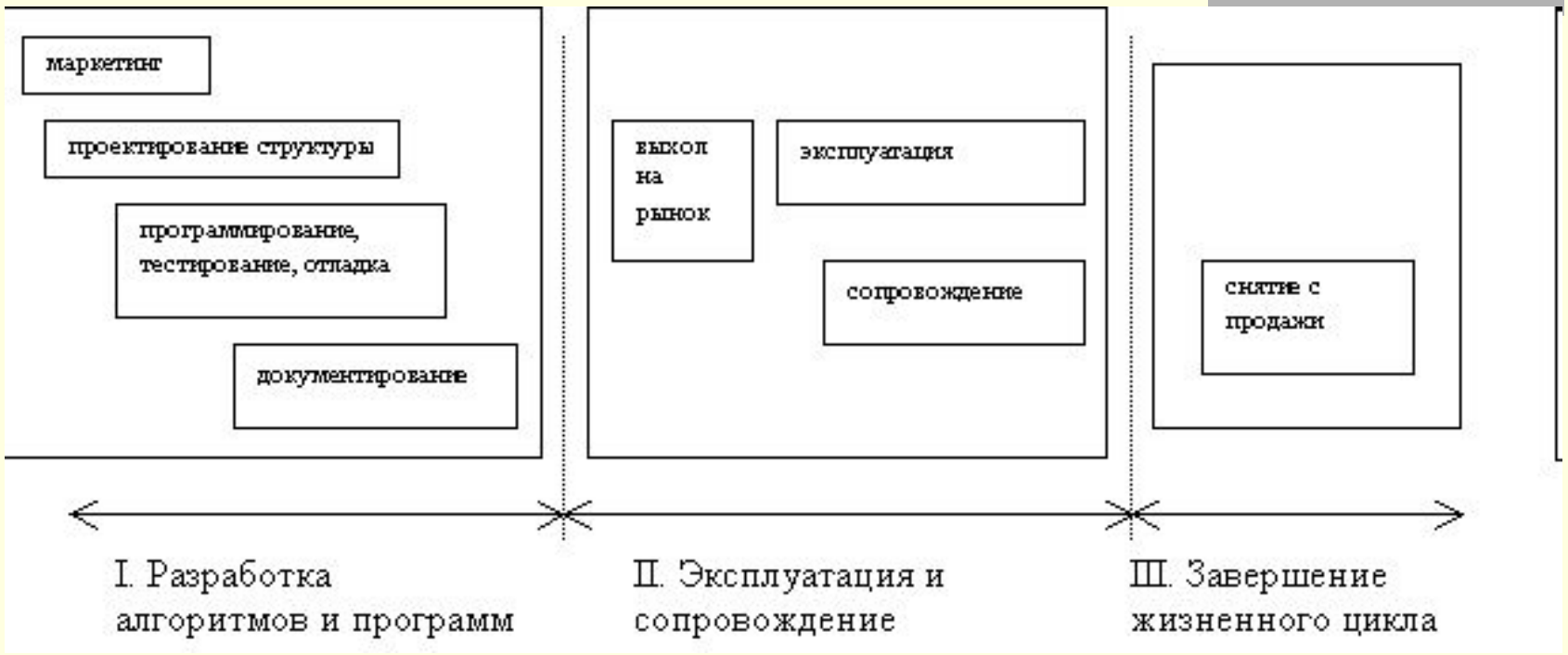
В отдел технического обслуживания позвонила женщина, которая сказала , что не может запустить свой компьютер. "Я все нажимаю на ножную педаль , а ничего не выходит", - сказала она технику .
- На ножную педаль ?
- Да , -ответила она, - Такая маленькая белая ножная педаль .
Этой педалью оказалась мышка.

1) Со слов менеджера:
- Не включается компьютер - зову админа. Админ приходит, воздевает руки к небу, бормочет про себя невнятные слова, поворачивает мой стул 10 раз вокруг своей оси, пинает компьютер - тот начинает работать. Вновь воздевает руки к небу, что-то бормочет, уходит.
2) Со слов сисадмина:
- Прихожу к юзеру - этот дурак так вертелся на стуле, что у него шнур питания на ножку намотался и выскочил из компа. Матерюсь про себя, распутываю, запихиваю комп ногой подальше под стол, включаю, ухожу..

Взаимодействие соучастников решения задачи на компьютере



Жизненный цикл коммерческого программного продукта



Для большинства современных программ длительность ЖЦ составляет 2-3 года

Определение алгоритма

Само слово “алгоритм” (algorithm) происходит от algorithmi - латинской формы написания имени аль-Хорезми, под которым в средневековой Европе знали величайшего математика из Хорезма (город в современном Узбекистане) Мухаммеда бен Мусу, жившего в 783-850 годах.

В своей книге “Об индийском счете” он сформулировал правила записи натуральных чисел с помощью арабских цифр и правила действий над ними столбиком.

Первоначально под алгоритмом и понимали только правила выполнения четырёх арифметических действий над многозначными числами.

В дальнейшем алгоритмом стали называть точное предписание, определяющее последовательность действий, обеспечивающую получение требуемого результата из исходных данных.

Правила построения алгоритма

- Выделить величины, являющиеся исходными для задачи.
- Указать, что является результатом решения задачи.
- Разбить процесс решения задачи на такие этапы, которые известны исполнителю и которые он может выполнить однозначно без всяких пояснений.
- Указать порядок выполнения этапов.
- Указать признак окончания процесса решения задачи.

Пример: решаем квадратное уравнение $ax^2+bx+c=0$

1. **Входные величины**: a,b,c
2. **Результат**: значения x1, x2 или сообщение «Корни комплексные»
3. **Этапы**: вычислить дискриминант; сравнить с 0; если меньше 0 – вывести сообщение «Корни комплексные», иначе вычислить x1, x2
4. **Порядок выполнения**: дискриминант считать до вычисления корней
5. **Признак окончания**: получен то или иной результат

Свойства алгоритма

- ***Однозначность алгоритма*** - единственность толкования исполнителем правила построения действий и порядок их выполнения. Чтобы алгоритм обладал этим свойством, он должен быть записан командами из системы команд исполнителя.
- ***Конечность алгоритма*** – обязательность завершения каждого из действий, составляющих алгоритм, и завершимость выполнения алгоритма в целом за конечное время.
- ***Детерминированность*** **–** будучи понятным, алгоритм не должен содержать предписаний, смысл которых может восприниматься неоднозначно, то есть одна и та же команда, будучи понятна разным исполнителем, после исполнения каждым из них должна давать одинаковый результат. Кроме того, в алгоритмах недопустимы также ситуации, когда после выполнения очередной команды алгоритма исполнителю неясно, какая из команд алгоритма должна выполняться на следующем шаге.

Результативность **–при точном исполнении всех предписаний алгоритма процесс должен прекратиться за конечное число шагов и при этом должен получиться определённый результат. Вывод о том, что решения не существует - тоже результат.**

Массовость **– свойство алгоритма, обеспечивать решение не одной конкретной задачи, а некоторого класса задач данного типа.**

Эффективность **– для решения задачи должны использоваться ограниченные ресурсы компьютера (процессорное время, объём оперативной памяти и т. д.).**

Пример неоднозначного алгоритма: Встреть девушку на поляне с цветами.

Пример неконечного алгоритма – инструкция к шампуню:

1. Нанести на волосы
2. Смыть водой
3. Повторить, начиная с п.1

Пример недетерминированного алгоритма: пойди туда не знаю куда принеси то не знаю что

Пример нерезультативного алгоритма:

1. Сложить c=a+b
2. Пока c<d повторять с п.1
3. Конец

Пример немассового алгоритма:

1. Вычислить c=10+20
2. Вычислить d=c*10
3. Вывести d
4. Конец

Пример неэффективного алгоритма:

1. Создать файл размером 100Гб
2. Повторить, начиная с п.1
4. Конец

Правила построения алгоритма

- ***Первое правило*** - при построении алгоритма, прежде всего необходимо задать множество объектов, с которыми будет работать алгоритм. Алгоритм приступает к работе с некоторым набором данных (входные), и в результате своей работы выдает данные (выходные). Таким образом, алгоритм преобразует входные данные в выходные.
- ***Второе правило*** - для работы алгоритма требуется память. В памяти размещаются входные данные, с которыми алгоритм начинает работать, промежуточные данные и выходные данные, которые являются результатом работы алгоритма. Память является дискретной, т.е. состоящей из отдельных ячеек. Поименованная ячейка памяти носит название переменной.

- ***Третье правило*** - **дискретность**. Алгоритм строится из отдельных шагов (действий, операций, команд). Множество шагов, из которых составлен алгоритм, конечно.
- ***Четвертое правило*** - **детерминированность**. После каждого шага необходимо указывать, какой шаг выполняется следующим, либо давать команду остановки.
- ***Пятое правило*** - **сходимость** (результативность). Алгоритм должен завершать работу после конечного числа шагов. При этом необходимо указать, что считать результатом работы алгоритма.

Способы записи алгоритмов

-естественный язык;

-псевдокод (pseudocode);

-блок-схема (flowchart)

Естественные языки *неоднозначны*, поэтому их сложно использовать для точной записи алгоритмов («проблема глобуса»).

Роботу на складе школьных принадлежностей кладовщик дал команду:
- Выброси отсюда все глобусы.
- Что такое глобус? – спросил робот.
- Это округлый предмет, соединенный с подставкой более тонким стержнем – ответил кладовщик.

ЭТО БЫЛИ ЕГО ПОСЛЕДНИЕ СЛОВА….

Псевдокод

Формальный язык с ограниченным словарём (часто на основе английского языка), промежуточный между естественным языком и языком программирования

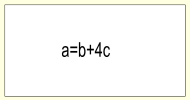
Псевдокод удобен тем, что позволяет программисту сосредоточиться на формулировке алгоритма, не задумываясь над синтаксическими особенностями конкретного языка программирования.

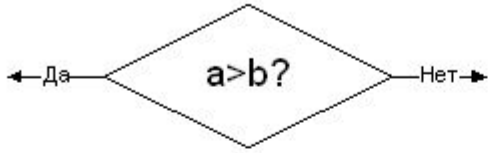
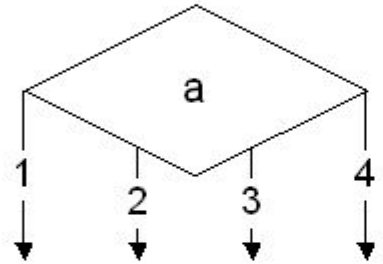
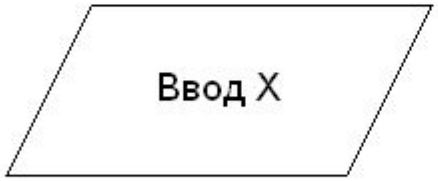
```
D=b^2-4*a*c

IF D<0
 Output “Корни комплексные»
ELSE
 {x1=… x2=… }
```

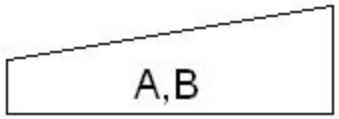
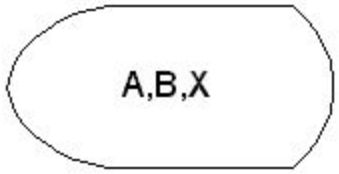
Блок-схемы

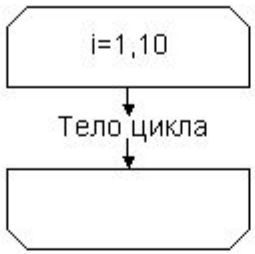
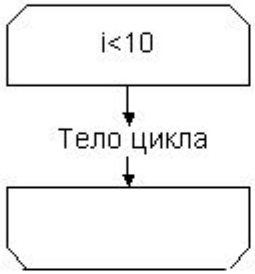
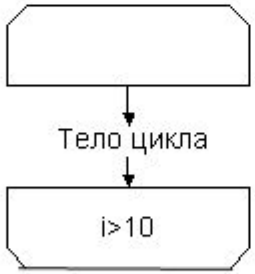
Удобны и понятны, но громоздки

Блок-схемы оформляются по ГОСТ 19.701-90: Блоки имеют пропорции 1:1,5, кроме блоков начала и конца, имеющих пропорции 1:4. Например, если ширина блока составляет 5 см, его высота должна быть 3,3см. Блоки, кроме начала или конца, нумеруются в разрыве линии слева вверху.

Пример обозначения	Что это?
	**Начало или конец программы, процедуры, функции**
	**Процесс. Любое вычисление.**
	**Вызов процедуры или функции**

	Выбор, проверка условия
	Множественное ветвление
	Ввод-вывод данных без уточнения устройства ввода-вывода

	Файл прямого доступа.
	Ввод с клавиатуры
	Вывод на экран

	Цикл с заданным числом повторений
	Цикл с предусловием
	Цикл с постусловием