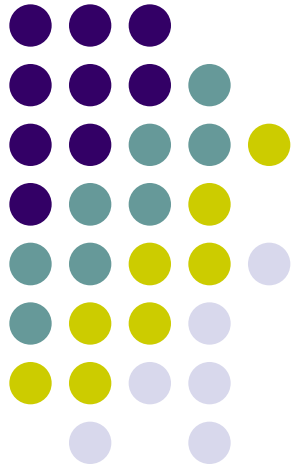
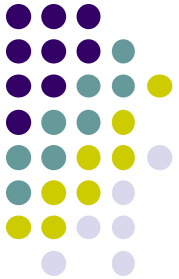
Основы программной инженерии

Барышникова Марина Юрьевна
МГТУ им. Н.Э. Баумана
Каф. ИУ-7
baryshnikovam@mail.ru

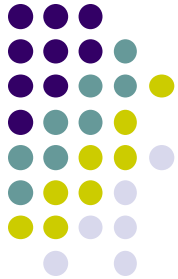


Лекция 2

Основные этапы разработки программ, их назначение и характеристики

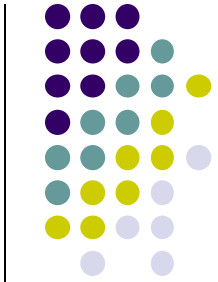


Основные этапы разработки программ



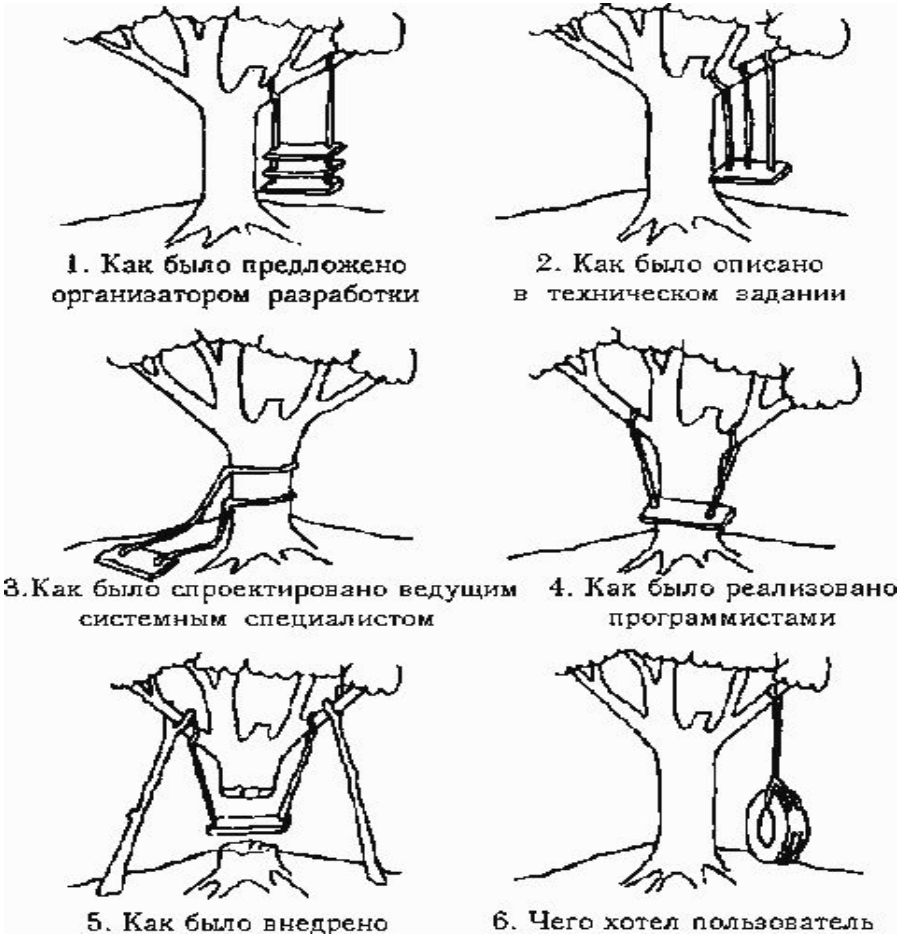
Этапы	
1. Постановка задачи	8. Выполнение
2. Выбор метода решения	9. Тестирование
3. Разработка алгоритма	10. Отладка
4. Написание программы на языке программирования	11. Документирование
5. Ввод программы в компьютер	12. Эксплуатация
6. Трансляция	13. Модификация
7. Компоновка	14. Снятие с эксплуатации

Постановка задачи



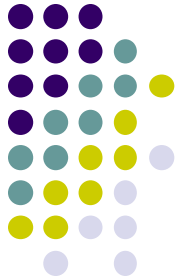
Цель этапа: определение функциональных возможностей программы, подготовка технического задания и внешней спецификации

К чему может привести непонимание между заказчиком и разработчиком ПО



1. Заказчик не может толком сформулировать требования, завышает их
2. Ответное предложение поставщика (разработчика) не вполне соответствует заявке заказчика
3. Аналитик предлагает ошибочную эскизную архитектуру
4. Программисты создают код с ошибками
5. Имеют место проблемы внедрения

Внешняя спецификация программы



Внешняя спецификация - достаточно полная и точная формулировка решаемой задачи

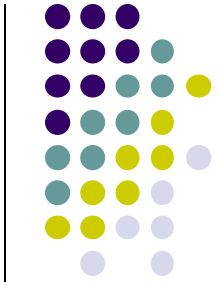
Формальная спецификация программы: {Q}S{R},

где Q – предусловие программы S,

R – постусловие программы S

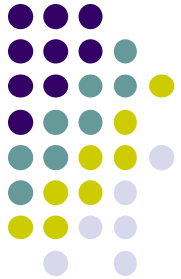
Если выполнение программы S началось в состоянии, удовлетворяющем Q, то имеется гарантия, что оно завершится через конечное время в состоянии, удовлетворяющем R

Состав внешней спецификации



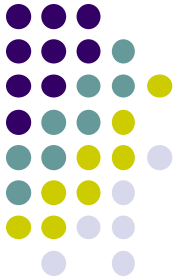
- *Описание исходных данных*. Должны быть точно описаны синтаксис (формат) и семантика (назначение, тип, допустимые значения, область изменения) всех исходных данных, которые вводит пользователь в программу
- *Описание выходных данных*. Дается точное описание семантики и синтаксиса всех результатов, формируемых программой, а также сообщений оператору об ошибках, о ходе вычислительного процесса, о запросах и т.д. Указывается реакция программы на некорректность исходных данных
- *Описание функций преобразования информации*, выполняемых программой, с точки зрения пользователя
- *Дополнительные сведения о программе*: ограничения на используемую память, длину программы, время ее работы; идеи относительно внутреннего проектирования функций (если это необходимо). В этот раздел также включают описание способа обращения к программе

Разработка алгоритма



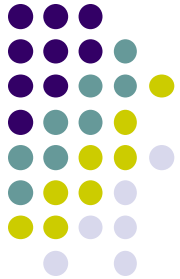
Алгоритм – это полное и точное описание на некотором языке конечной последовательности правил, указывающих исполнителю действия, которые он должен выполнить, чтобы за конечное время перейти от (варьируемых) исходных данных к искомому результату

Свойства алгоритмов



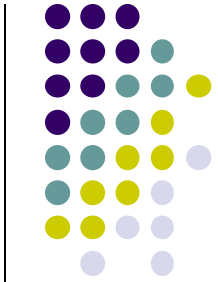
- Дискретность – возможность разбиения на шаги
- Понятность – ориентация на конкретного исполнителя
- Определенность – однозначность толкования инструкций
- Конечность – возможность получения результата за конечное число шагов
- Массовость – применимость к некоторому классу объектов
- Эффективность – оптимальность времени и ресурсов, необходимых для реализации алгоритма

Процесс алгоритмизации



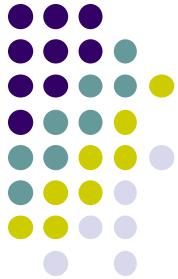
- разложение всего вычислительного процесса на отдельные шаги – возможные составные части алгоритма, что определяется внутренней логикой самого процесса и системой команд исполнителя;
- установление взаимосвязей между отдельными шагами алгоритма и порядка их следования, приводящего от известных исходных данных к искомому результату;
- полное и точное описание содержания каждого шага алгоритма на языке выбранной алгоритмической системы;
- проверка составленного алгоритма на предмет, действительно ли он реализует выбранный метод и приводит к искомому результату

Способы описания алгоритмов



- словесно-формульный (на естественном языке, вербальный);
- структурный или блок-схемный (графический);
- с использованием специальных алгоритмических языков (нотаций);
- с помощью сетей Петри;
- программный

Словесно-формульный способ



Пусть необходимо найти значение выражения:
y=2a-(x+6)

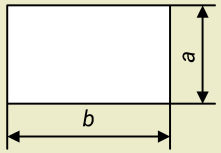
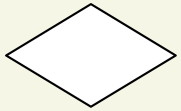
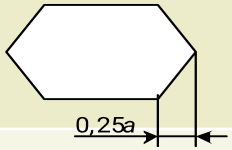
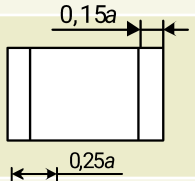
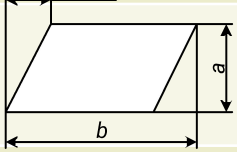
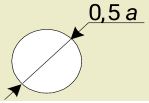
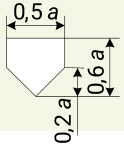
Словесно-формульным способом алгоритм решения этой задачи может быть записан в следующем виде:

1. Ввести значения *a* и *x*
2. Сложить *x* и *6*
3. Умножить *a* на *2*
4. Вычесть из *2a* сумму (*x+6*)
5. Вывести y как результат вычисления выражения

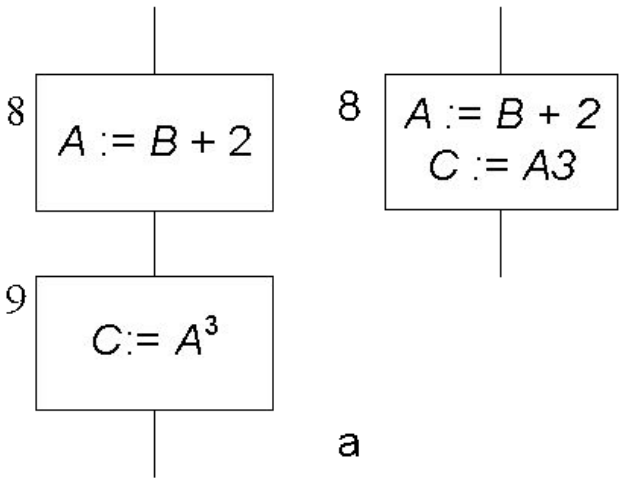
При словесно-формульном способе алгоритм записывается в виде текста с формулами по пунктам, определяющим последовательность действий

Стандарты графических изображений блоков

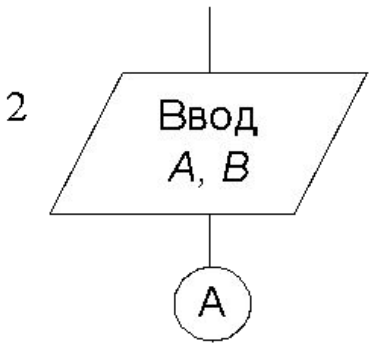
a = 10, 15, 20 мм; b = 1,5a

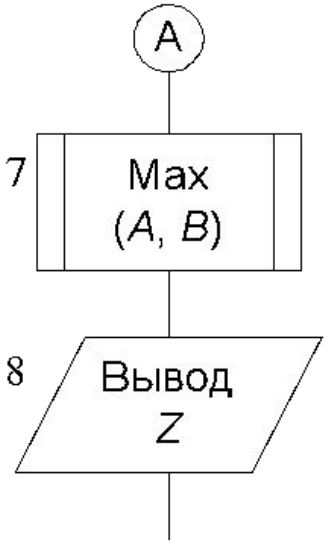
Наименование символа	Обозначение и размеры	Функция
Процесс (вычислительный блок)		Выполнение операции или группы операций, в результате которых изменяются значение, форма представления или расположение данных
Решение (логический блок)		Выбор направления выполнения алгоритма в зависимости от некоторых условий
Модификация (заголовок цикла)		Выполнение операций по управлению циклом – повторением команды или группы команд алгоритма
Пуск-останов (начало-конец)		Начало или конец выполнения программы или подпрограммы
Предопределенный процесс (вызов подпрограммы)		Вызов и использование ранее созданных и отдельно описанных алгоритмов (подпрограмм)
Ввод/вывод		Общее обозначение ввода или вывода данных в алгоритме безотносительно к внешнему устройству
Соединитель		Указание прерванной связи между блокам в пределах одной страницы
Межстраничный соединитель		Указание прерванной связи между блоками, расположенными на разных листах

Применение блок-схем

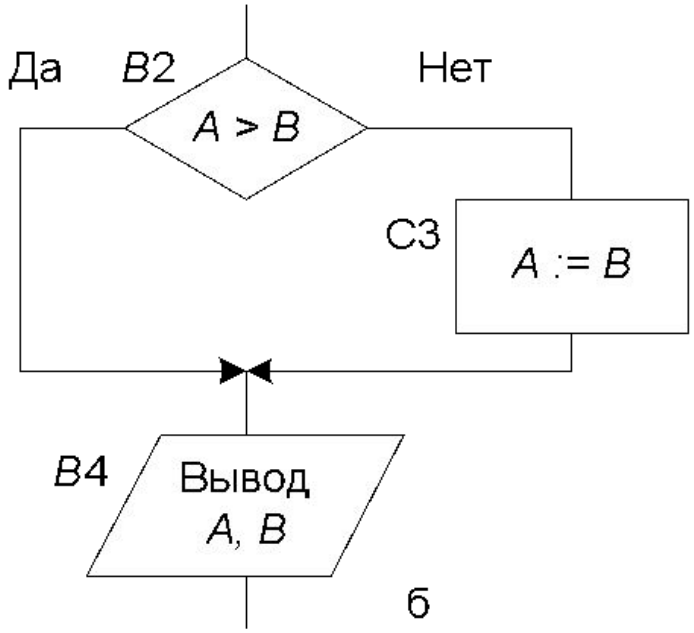


а

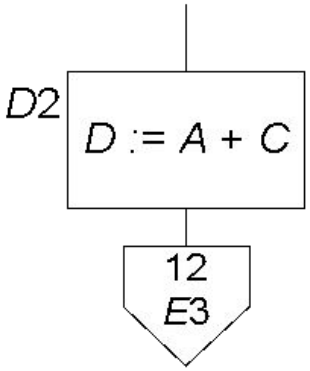




в

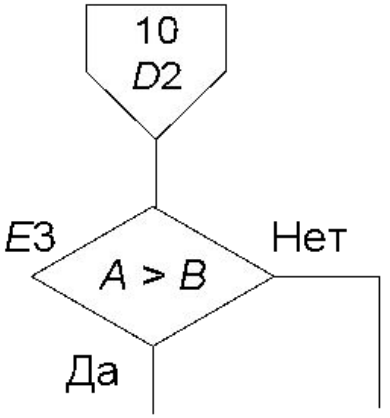


б



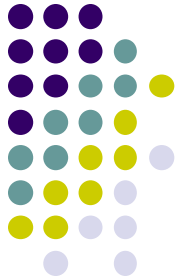
Лист 10

г



Лист 12

Пример псевдокода для функции печати массива



Вход: a – массив целых чисел,

n – количество элементов в массиве

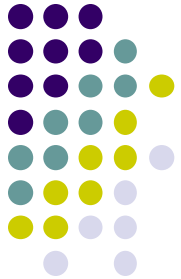
Выход: -

пока не конец массива **делать**

печатать очередной элемент массива

все пока

Технология разработки алгоритмов



Качества хорошего алгоритма:

- правильно решает поставленную задачу
- легок для понимания
- прост для доказательства правильности
- удобен для модификации

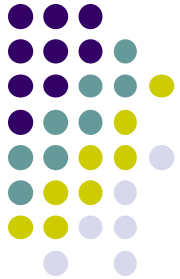
Конструирование и оформление алгоритмов осуществляется в рамках структурного подхода, в основе которого лежит теорема о структурировании: алгоритм решения любой практически вычислимой задачи может быть представлен с использованием трех элементарных базисных управляющих структур:

а) следования;

б) ветвления;

в) цикла с предусловием

Базисные управляющие структуры

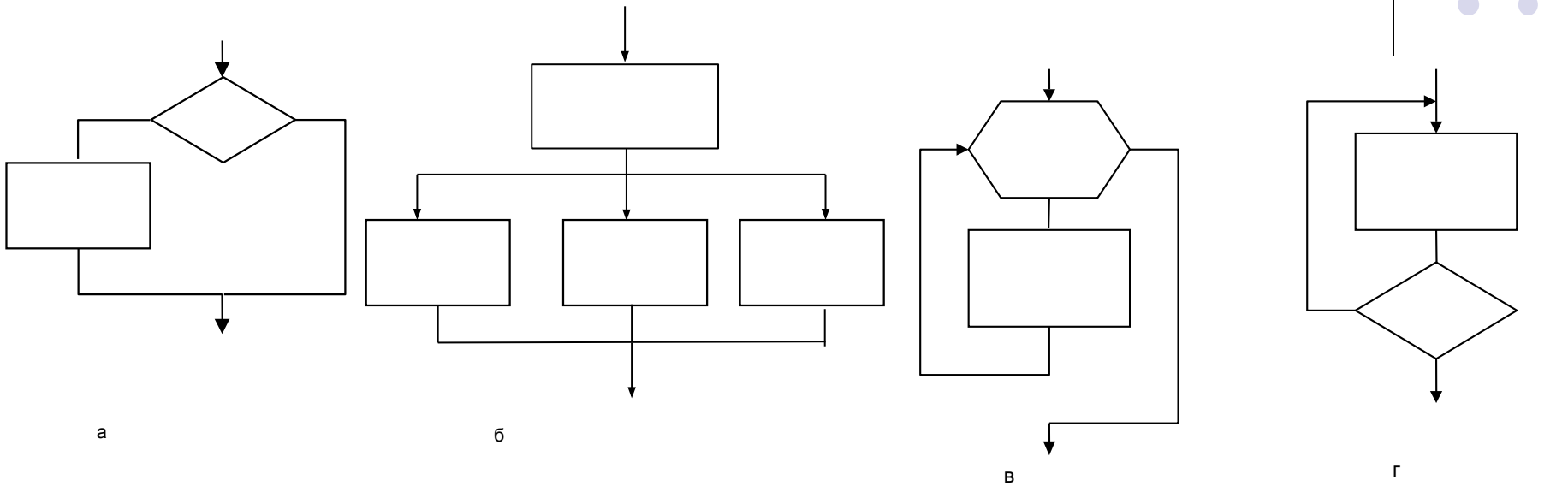


где P – условие, S – оператор

а) следование; б) ветвление; в) цикл с предусловием

Базисный набор управляющих структур является функционально полным

Дополнительные управляющие структуры



а) структура сокращенного ветвления;

б) структура выбора;

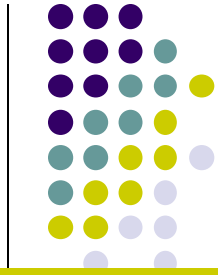
в) структура цикла с параметром;

г) структура цикла с постусловием

Любой алгоритм может быть построен посредством композиции базисных и дополнительных структур:

- их последовательным соединением, т.е. образованием последовательных конструкций;
- их вложением друг в друга, т.е. образованием вложенных конструкций

Написание программы на языке программирования

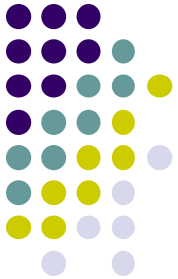


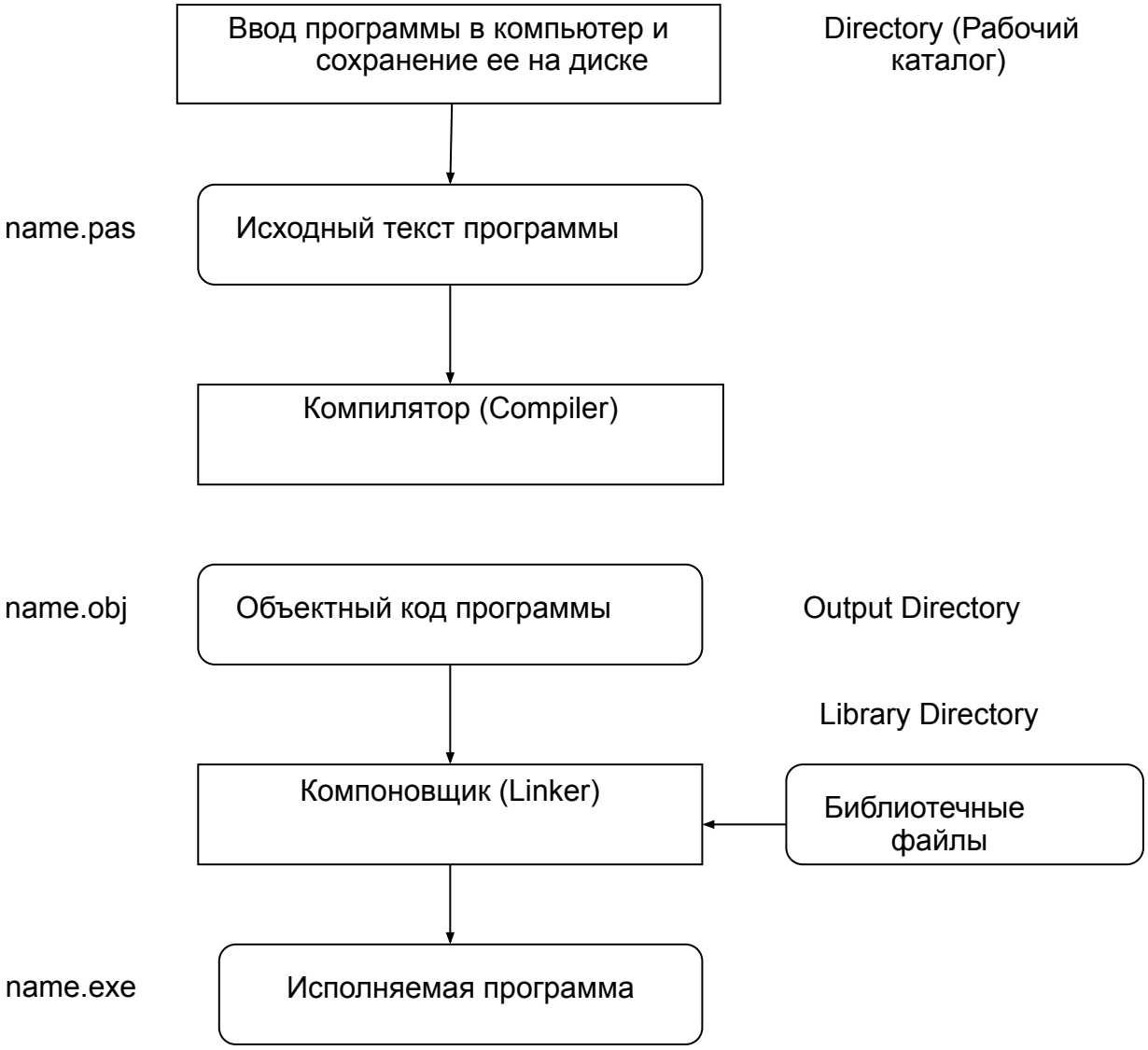
Язык программирования — формальная знаковая система, предназначенная для записи компьютерных программ. Язык программирования определяет набор лексических, синтаксических и семантических правил, задающих внешний вид программы и действия, которые выполнит исполнитель (компьютер) под ее управлением

- Программа – логически упорядоченная последовательность команд, необходимых для решения определенной задачи
- Текст программы – полное законченное и детальное описание алгоритма на языке программирования

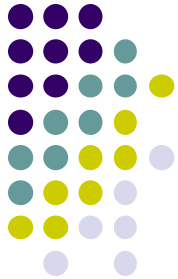
Программа – алгоритм, записанный на языке прогр

Порядок прохождения задач через ЭВМ





Трансляция программы



- **Компиляция** - преобразование объектов (данных и операций над ними) с входного языка в объекты на другом языке для всей программы в целом с последующим выполнением полученной программы в виде отдельного шага
- **Интерпретация** - анализ отдельного объекта на входном языке с одновременным выполнением (интерпретацией)

Трансляция — это преобразование программы с одного языка программирования в семантически эквивалентный текст на другом языке

Компиляция программы

Компилятор – это программа, предназначенная для трансляции исходного текста программы с высокоуровневого языка в объектный код. Входной информацией для компилятора является описание алгоритма или программа на языке программирования

Результатом компиляции является объектный файл с необходимыми внешними ссылками для компоновщика. Программа уже переведена в машинные инструкции, однако еще не полностью готова к выполнению

Компоновка программы

Компоновщик – модуль системы программирования или самостоятельная программа, которая собирает результирующую программу из объектных модулей и стандартных библиотечных модулей

Компоновка - это процесс сборки программы из объектных модулей, в котором производится их объединение в исполняемую программу и связывание вызовов внешних функций и их внутреннего представления (кодов), расположенных в различных объектных модулях. При этом могут объединяться один или несколько объектных модулей программы и объектные модули, взятые из библиотечных файлов и содержащие стандартные функции и другие инструкции

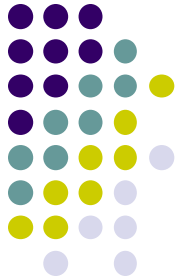
Результатом компоновки является исполняемый файл, т.е. файл, который может быть обработан или выполнен компьютером без предварительной трансляции

Выполнение программы

- *Исполняемый файл* — это файл, содержащий программу в том виде, в котором она может быть исполнена компьютером
- *Формат исполняемого файла* — это соглашение о размещении в нём машинных команд и вспомогательной информации

В операционной системе Windows основным является формат исполняемых файлов PE (от англ. portable executable — переносимый исполняемый). Файлы этого формата обычно имеют расширение «.exe» или «.dll». При этом непосредственно выполнить можно только файлы с расширением «.exe»

Тестирование программы

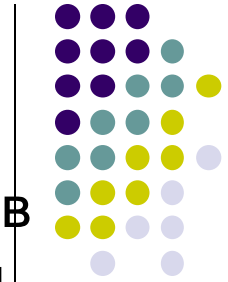


Тестирование – это процесс исполнения программы с целью обнаружения ошибок

Требования к тестам:

- Простота: тест должен представлять собой простой набор исходных данных, позволяющих легко просчитать и получить верный результат
- Полнота: в результате тестирования каждый оператор программы должен выполниться хотя бы один раз
- Не избыточность: не избыточный тест – это такой тест, в котором удаление хотя бы одного тестового набора данных превращает его в неполный

Отладка программы



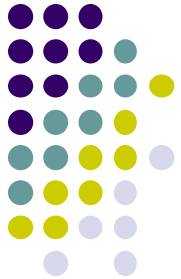
Отладка — этап разработки компьютерной программы, в ходе выполнения которого обнаруживают, локализуют и устраняют ошибки

Ошибка – это расхождение между вычисленным, наблюдаемым и истинным, заданным или теоретически правильным значением

Классификация программных ошибок:

- синтаксические (нарушение грамматических правил языка программирования);
- семантические (нарушение порядка следования параметров функций, неправильное построение выражений);
- прагматические или логические (заключаются в неправильной логике алгоритма, нарушении смысла вычислений и т. п.)

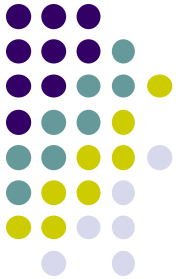
Примеры синтаксических ошибок



- пропуск необходимого знака пунктуации;
- несогласованность скобок или пропуск нужных скобок;
- неверное написание зарезервированных слов;
- отсутствие объявлений идентификаторов

Синтаксические ошибки обнаруживаются компилятором

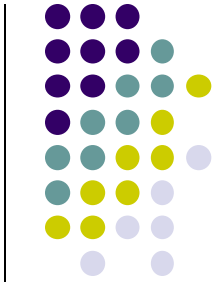
Примеры семантических ошибок



- некорректное использование переменных (до инициализации, использование индексов, выходящих за границы массивов и т.п.);
- ошибки вычисления (некорректное использование целочисленной арифметики, незнание приоритетов выполнения операций, деление на 0, извлечение корня из отрицательного числа и т.п.);
- ошибки межмодульного интерфейса (игнорирование системных соглашений при передаче параметров, нарушение области действия локальных и глобальных переменных и т.п.)

Семантические ошибки также обнаруживаются компилятором

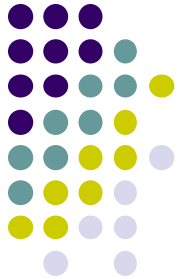
Примеры логических ошибок



- ошибки алгоритма;
- ошибки накопления погрешностей результатов вычисления (некорректное отбрасывание дробных цифр числа, некорректное использование приближенных методов вычисления, игнорирование использования разрядной сетки ЭВМ и т.п.)

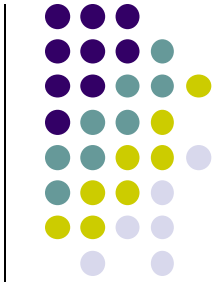
Логические ошибки обнаруживаются программистом

Последовательность обнаружения ошибок



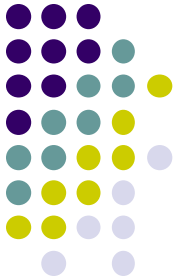
- ошибки трансляции (компиляции):
  - ошибки соответствия синтаксису языка
  - ошибки компоновки (ошибки связи);
  - ошибки данных;
- ошибки выполнения
- ошибки логики

Методы отладки



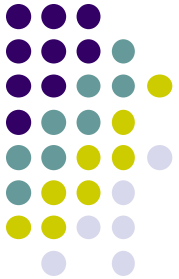
- Отладка за столом:
  - просмотр;
  - проверка;
  - прокрутка;
- Программный способ отладки (так называемая отладочная печать):
  - эхо–печать входных данных;
  - печать в ветвях программы;
  - печать в узлах программы;
- Аппаратный способ (встроенные интегрированные средства отладки):
  - выполнение по шагам
  - просмотр переменных в окне наблюдения;
  - локализация места ошибки при выполнении программы до курсора

Трудоемкость этапов



Этапы	Трудозатраты	Ошибки	
		Появление	Выявление
Постановка задачи Математическая формулировка Выбор метода решения	10%	40-46%	50%
Составление алгоритма Написание программы на языке программирования	20% 15%	35-38%	
Ввод программы в компьютер Выполнение программы	5%	5-10%	
Тестирование Отладка	40%		45%
Документирование Эксплуатация Модернизация	10%		3%

Эксплуатация ПО



Эксплуатация системы выполняется в предназначенной для этого среде в соответствии с пользовательской документацией

Сопровождение ПО

внесение изменений в ПО в целях исправления ошибок, повышения производительности или адаптации к изменившимся условиям работы или требованиям