Концепции языков программирования

Введение

Зачем изучать концепции языков программирования

- Большая свобода в выражении мыслей
- Знания, необходимые для правильного выбора подходящего языка программирования
- Способность легче осваивать новые языки
- Понимание значения реализации
- Общий прогресс в компьютерных областях

Области применения программирования

- Научные приложения
  - Частые вычисления с плавающей точкой: Фортран
- Бизнес-приложения
  - Составление отчетов; десятичные числа и символы: КОБОЛ
- Искусственный интеллект
  - В основном, манипуляции с символами, а не числами: ЛИСП
- Системное программирование
  - Ввиду постоянного использования необходима эффективность: C
- Веб-приложения
  - Разнородная коллекция языков: языки разметки (XHTML), сценариев (PHP), общего назначения (Java)

Критерии оценки языков

- Читабельность: насколько легко читать и понимать программы
- Легкость написания программ
- Надежность: соответствие спецификациям (т.е. поведение согласно спецификациям)
- Стоимость: конечная общая стоимость

Критерии оценки языков

Читабельность

- Общая простота
  - Обозримое количество конструкций
  - Ограниченное количество способов выполнения операций
  - Минимальная перегрузка операторов
- Ортогональность
  - Небольшое число примитивных конструкций, комбинируемых небольшим числом способов
  - Каждая возможная комбинация допустима
- Структуры управления
  - Наличие общепринятых структур управления (например, цикл while)
- Типы данных и структуры
  - Наличие адекватных средств для определения структур данных
- Синтаксис
  - Идентификаторы: гибкость создания
  - Специальные слова и методы формирования составных операторов

Критерии оценки языков

Легкость написания программ

- Простота и ортогональность
  - Небольшое число конструкций, примитивов и правил их комбинирования
- Поддержка абстракции
  - Возможность определять и использовать сложные структуры и операции, игнорируя детали
- Выразительность
  - Набор относительно удобных средств определения операций
  - Пример: включение оператора `for` во многие современные языки

Критерии оценки языков

Надежность

- Проверка типов
  - Проверка ошибок, связаных с типами
- Обработка исключительных ситуаций
  - Перехват ошибок исполнения и принятие мер по исправлению ситуации
- Именование
  - Наличие двух или более различных методов для обращения к одному и тому же участку памяти
- Легкость чтения и написания программ
  - Язык, который не поддерживает «естественные» способы формулировки алгоритма, требует применения «неестественных» подходов, что понижает надежность

Критерии оценки языков

Стоимость

- Обучение программистов языку
- Написание программ (приспособленность к конкретным приложениям)
- Компиляция программ
- Выполнение программ
- Реализация языка: доступность бесплатных компиляторов
- Надежность: низкая надежность влечет большие траты
- Поддержка программ

Критерии оценки языков

Дополнительные критерии

- Переносимость
  - Легкость перенесения программ между двумя разными реализациями
- Универсальность
  - Пригодность для широкого спектра приложений
- Стандартизация
  - Полнота и точность официального описания языка

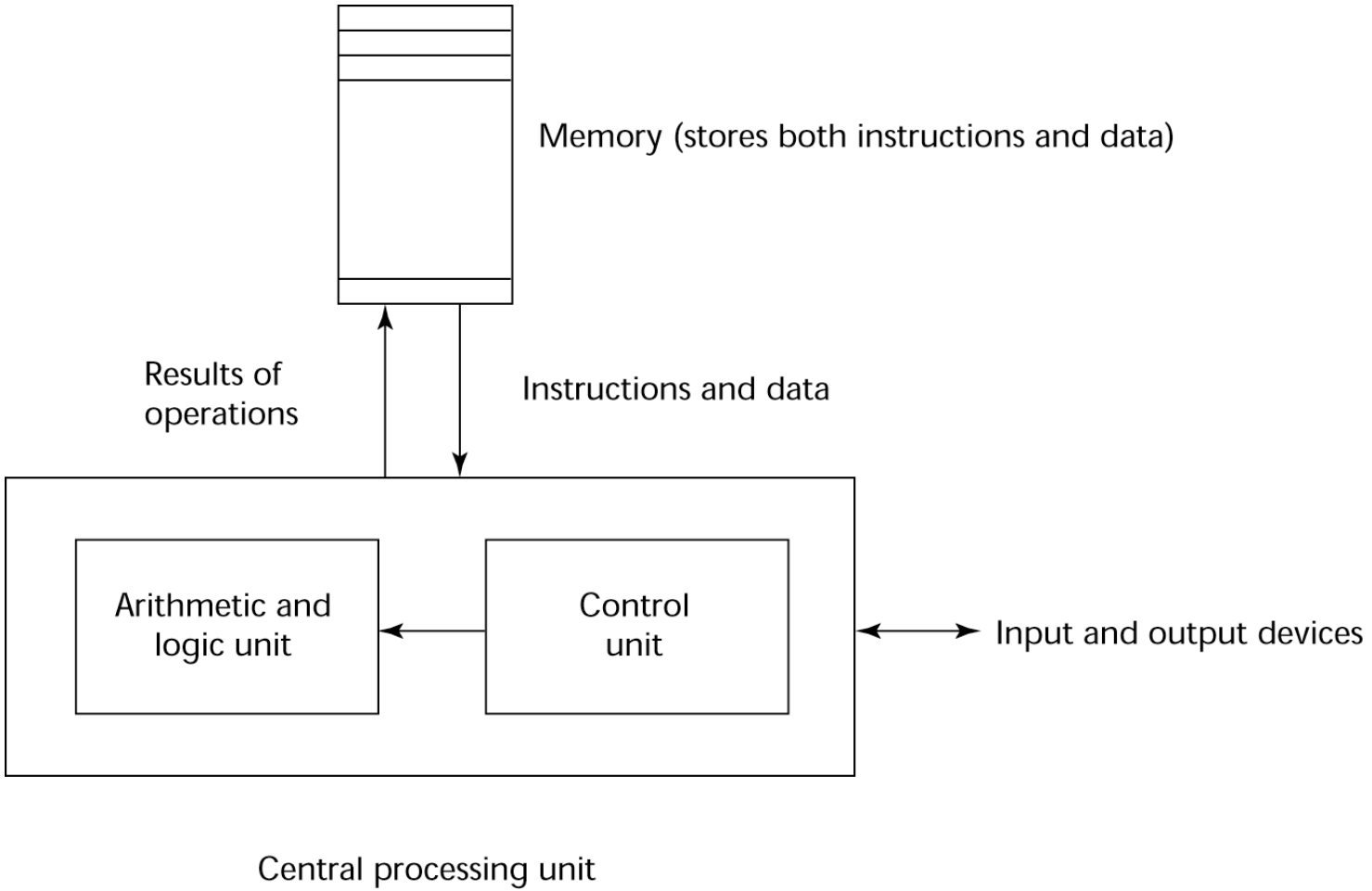
Что влияет на дизайн языка

- Архитектура ЭВМ
  - Языки создаются для наиболее распространенной компьютерной архитектуры, называемой архитектурой фон Неймана
- Методологии программирования
  - Новые методологии разработки программного обеспечения (например объектно-ориентированная разработка программного обеспечения) порождают новые парадигмы программирования и, следовательно, новые языки программирования

Влияние компьютерной архитектуры

- Известная компьютерная архитектура: фон Нейманн
- Императивные языки, наиболее распространены благодаря архитектуре фон Неймана
  - Данные и программы хранятся в памяти
  - Память отделена от ЦПУ
  - Инструкции и данные передаются из памяти в ЦПУ
  - Основа императивных языков
    - Переменные являются моделью ячеек памяти
    - Операторы присваивания являются моделью передачи данных
    - Итерации эффективны

Архитектура фон Нейманна



Влияние методологий программирования

- 1950–1960: Простые приложения; приоритет эффективности компьютеров
- Конец 1960-ых: Повышается значение эффективности программистов; читабельность, улучшенные структуры управления
  - структурное программирование
  - проектирование сверху вниз и пошаговое улучшение
- Конец 1970-ых: От процедур к данным
  - абстракция данных
- Середина 1980-ых: Объектно-ориентированное программирование
  - абстракция данных + наследование + полиморфизм

Категории языков

- Императивные
  - Основные черты: переменные, операторы присвоения и итерации
  - Примеры: C, Паскаль
- Функциональные
  - Вычисления осуществляются применением функций к параметрам
  - Примеры: ЛИСП, Scheme
- Логические
  - Основанные на правилах (правила задаются в произвольном порядке)
  - Пример: Пролог
- Объектно-ориентированные
  - Абстракция данных, наследование, позднее связывание
  - Примеры: Java, C++
- Языки разметки
  - Новые; не являются языками программирования в собственном смысле слова, используются для разположения информации в веб-документах

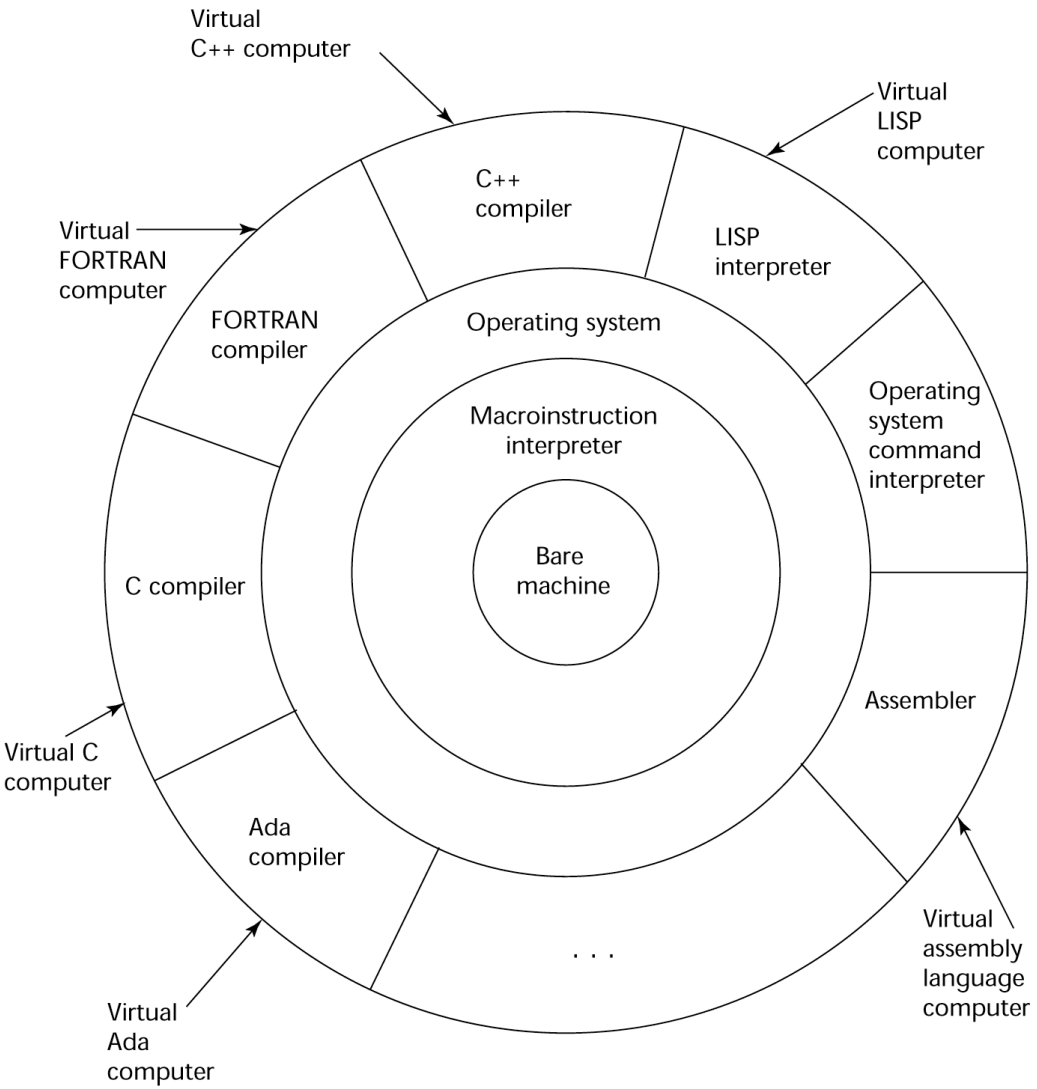
Компромиссы при проектировании языка

- Надежность и стоимость выполнения
  - Конфликтующие критерии
  - Пример: Java требует, чтобы все ссылки на элементы массива проходили проверку на соответствие индексов, но это приводит к повышению стоимости выполнения
- Легкость чтения и написания программ
  - Также конфликтующие критерии
  - Пример: APL предоставляет множество мощных операторов (и большое число новых символов), позволяя записывать сложные вычисления в виде компактной программы за счет плохой читабельности
- Легкость написания программ (гибкость) и надежность
  - Также конфликтующие критерии
  - Пример: Указатели в C++ — мощное и очень гибкое средство, но приводит к понижению надежности программ

Методы реализации

- Компиляция
  - Программы переводятся на машинный язык
- Интерпретация в чистом виде
  - Программы интерпретируются другой программой, называемой интерпретатором
- Гибридные реализации
  - Компромисс между компиляторами и интерпретаторами в чистом виде

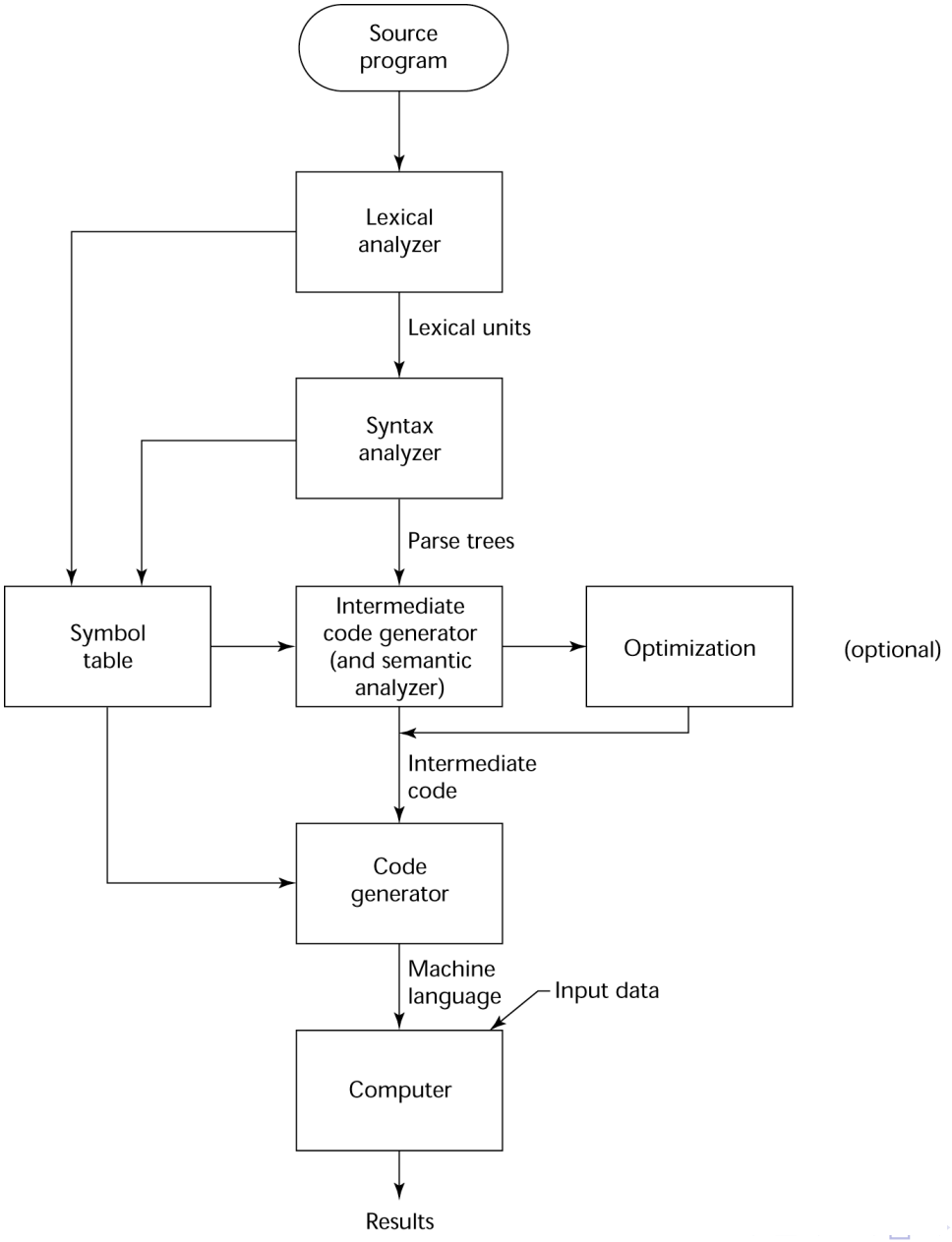
Методы реализации



Компиляция

- Перевод высокоуровневых программ (исходный язык) на язык машинных кодов (машинный язык)
- Медленная трансляция, быстрое исполнение
- Процесс компиляции состоит из нескольких фаз:
  - лексический анализ: составление лексических единиц из символов исходной программы
  - синтаксический анализ: получение из лексических единиц деревьев парсинга, которые представляют синтаксическую структуру программы
  - семантический анализ: порождение промежуточного кода
  - порождение машинного кода

Процесс компиляции



Компиляция: дополнительные термины

Загрузочный модуль (исполнимый образ): пользовательский и системный код вместе взятые

Связывание и загрузка : процесс сбора системных программ и связывания их с пользовательской программой

Исполнение машинного кода

- Цикл выбор – выполнение (на архитектуре фон Неймана)

инициализировать программный счетчик
repeat бесконечно
    выбрать инструкцию, на которую указывает счтечик
    увеличить значение счетчика
    декодировать инструкцию
    выполнить инструкцию
end repeat

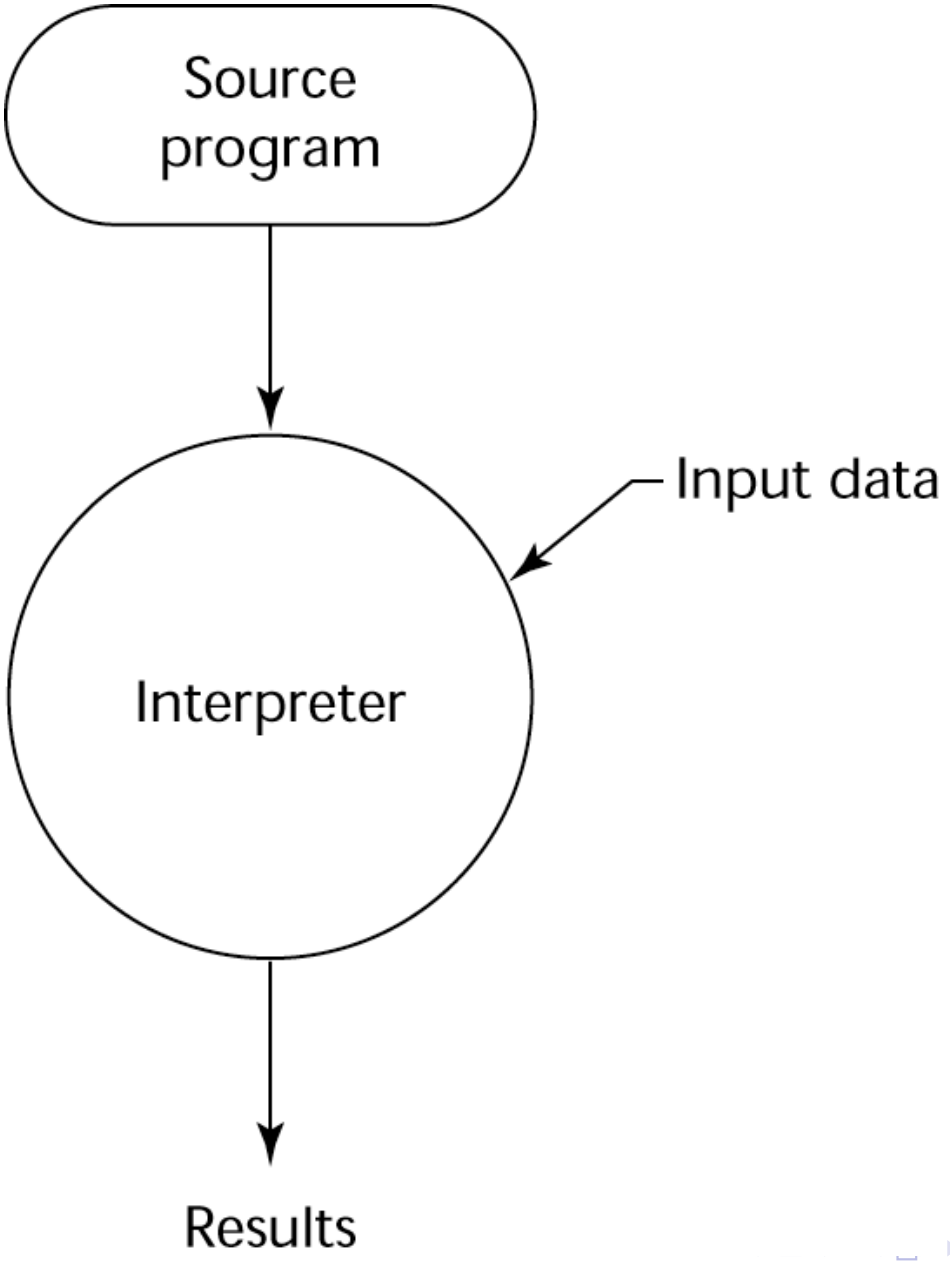
Узкое место архитектуры фон Нейманна

- Скорость связи между памятью компьютера и его процессором определяет скорость компьютера
- Программные инструкции часто могут выполняться на гораздо более высокой скорости, чем скорость этой связи; таким образом, скорость связи является узким местом
- Узкое место архитектуры фон Нейманна является главным фактором, отрицательно влияющим на скорость работы компьютеров

Интерпретация в чистом виде

- Никакой трансляции
- Более легкая реализация программ (ошибки исполнения могут быть немедленно локализованы)
- Более медленное исполнение (от 10 до 100 раз более медленное по сравнению с откомпилированными программами)
- Часто требует больше пространства
- Не часто используется с языками высокого уровня
- Важное исключение — некоторые языки сценариев для Веба (например, JavaScript)

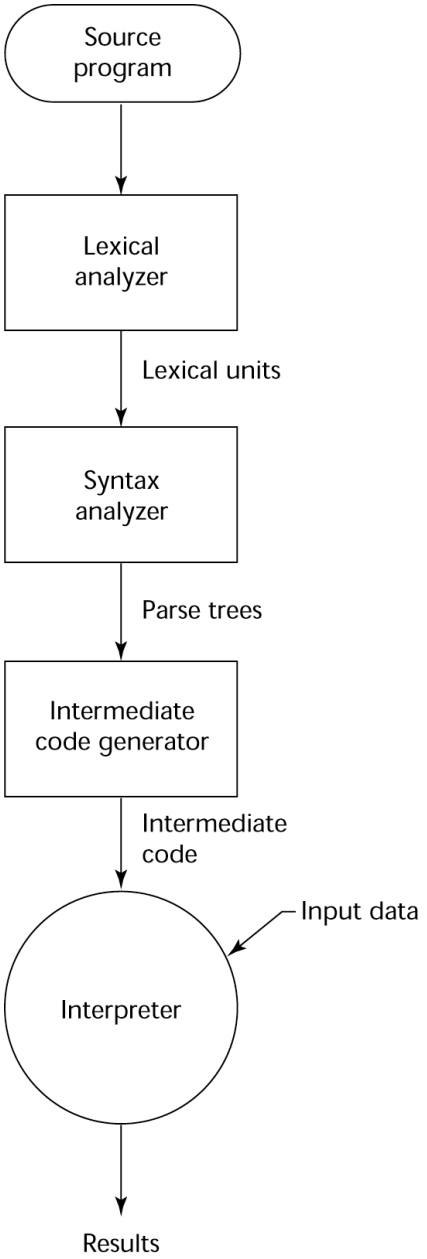
Процесс интепретации в чистом виде



Гибридные реализации

- Компромисс между компиляторами и интерпретаторами в чистом виде
- Программа на языке высокого уровня переводится на промежуточный язык, допускающий более легкую интерпретацию
- Быстрее, чем интерпретация в чистом виде
- Примеры
  - Программы на Перле частично компилируются перед интерпретацией с целью обнаружения ошибок
  - Первые реализации Java были гибридными; промежуточная форма — байт-код — обеспечивает возможность исполнения программы на любой машине, снабженнной интерпретатором байт-кода и системой исполнения (вместе они называются виртуальной Java-машиной)

Процесс гибридной интерпретации



Реализация Just-in-Time

- Сначала программы переводятся на промежуточный язык
- Затем промежуточный язык компилируется в машинный код
- Версия в машинном коде используется при дальнейших вызовах
- Системы JIT широко применяются для Java-программ
- Языки .NET реализованы с использованием JIT-системы

Препроцессоры

- Макросы (инструкции) препроцессора широко используются для указания на то, что код из другого файла должен быть включен
- Препроцессор обрабатывает программу непосредственно перед компиляцией и раскрывает встроенные макросы пропроцессора
- Известный пример: препроцессор C
  - раскрывает `#include`, `#define` и другие подобные макросы

Среды программирования

- Набор инструментов для разработки программного обеспечения
- UNIX
  - Старая операционная система и набор инструментов
  - В наши дни часто снабжена графичечкой оболочкой (например, CDE, KDE или GNOME), которая выполняется поверх UNIX
- Borland JBuilder
  - Интегрированная среда разработки для Java
- Microsoft Visual Studio.NET
  - Большая, сложная визуальная среда
  - Поддерживает программирование на C#, Visual Basic.NET, JScript, J# и C++

Заключение

- Сравнительное изучение языков программирования полезно по ряду причин:
  - Дает возможность лучше использовать различные конструкции
  - Позволяет выбирать языки с большим основанием
  - Облегчает освоение новых языков
- Важнейшие критерии оценки языков программирования:
  - Легкость чтения, написания, надежность, стоимость программ
- Особое влияние на проектирование языков оказывают архитектура компьютера и методологии разработки программ
- Основные методы реализации языков программирования: компиляция, интерпретация в чистом виде и гибридная реализация