Типы трансляторов. Интерпретаторы и компиляторы

- Конечная цель создания программного продукта является достижение некоторого результата, способ получения которого закодирован в этой программе
- Этот результат может быть получен только при работе аппаратуры вычислительной системы, которой для работы передаётся программа, а также входные данные, требующиеся программе при её работе

Типы трансляторов. Интерпретаторы и компиляторы

Существует несколько вариантов взаимодействия с аппаратурой в целях достижения требуемого результата:

1. Не подразумевается никакой необходимости в системе программирования, кодирование программ ведётся непосредственно на машинном языке
2. Программирование ведётся на языке, не совпадающем с машинным языком данной вычислительной системы, что требует наличия системы программирования, в которую должны быть включены компоненты, ответственные за преобразование исходной программы к виду, в котором она может быть понята вычислительной системой

Типы трансляторов. Интерпретаторы и компиляторы

- Преобразование исходных программ выполняется системами программирования с помощью компонентов, называемых ***трансляторами***
- Задачей ***трансляторов*** является перевод (преобразование) исходной программы, написанной на некотором исходном (входном) языке, в другую программу на целевом (выходном) языке, эквивалентную первой

Типы трансляторов. Интерпретаторы и компиляторы

- Процесс перевода с исходного языка программы в целевой язык охватывает сразу три программы и называется ***трансляцией***:
  1. При трансляции вычислительная система выполняет программу транслятора (***транслирующая программа***)
  2. Транслятор преобразует последовательность предложений входного языка, удовлетворяющую набору синтаксических и семантических правил (***транслируемая программа***)
  3. Результатом работы транслятора является программа, построенная по синтаксическим правилам выходного языка с учётом семантики выходного языка (***результирующая программа***)
- Результирующая программа полностью эквивалентна исходной программе

Ассемблеры

- Для каждой вычислительной машины имеется язык программирования, близкий к машинному языку, называемый автокодом или ***языком ассемблера***
- Программы, которые обрабатывают тексты на таких языках, называются ***ассемблерами***
- Для языков ассемблера разработан стандарт *«Standard for Microprocessor Assembly Language» IEEE 694-1985*, в котором указано, что они должны обрабатываться ассемблерами на основе принципа «один-в-один»

Компиляторы

- Термин ***компилятор*** обычно используется вместо термина транслятор в тех случаях, когда исходным языком трансляции является язык программирования высокого уровня (Паскаль или Си++), а целевым языком является автокод, язык ассемблера или машинный язык некоторой вычислительной машины
- Вычислительная система, для которой ведётся компиляция, называется ***целевой вычислительной системой***, в понятие о которой входят:
  - аппаратура ЭВМ
  - операционная система, работающая на этой аппаратуре
  - набор динамических библиотек, которые необходимы для выполнения целевой программы

Схема преобразования программ компиляторами

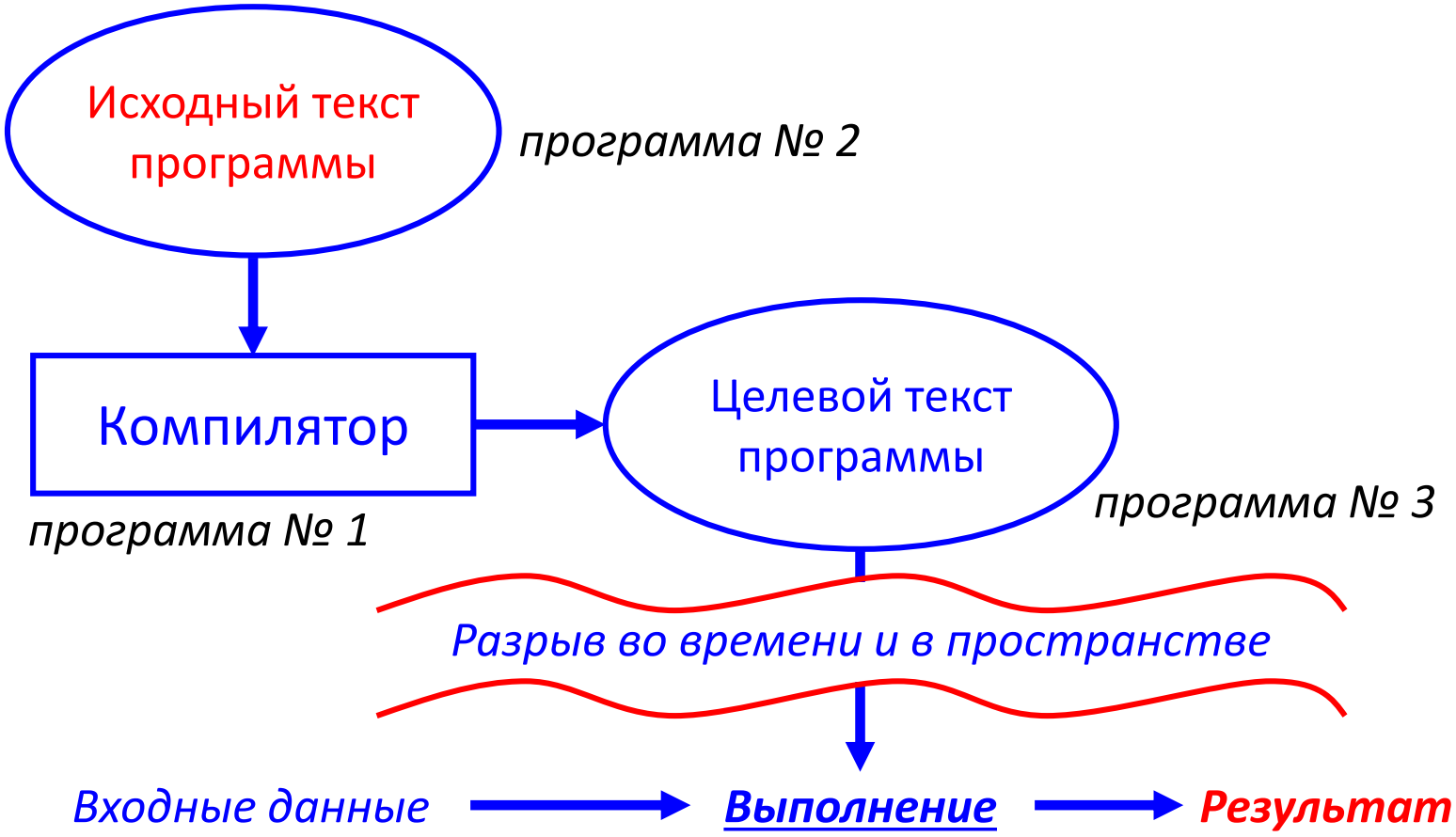
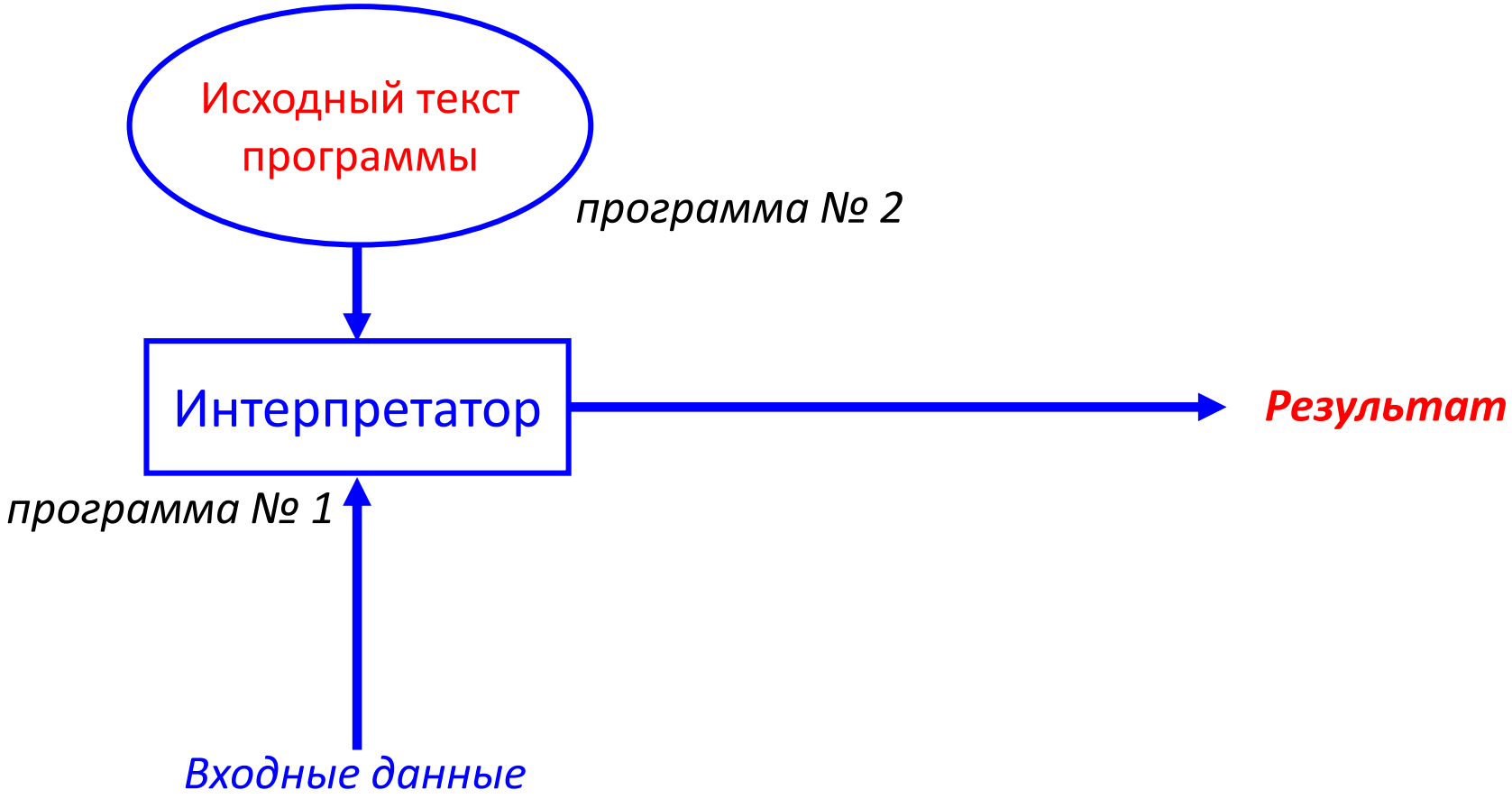


Схема получения результата программ интерпретаторами



Интерпретаторы

- *Интерпретация* выполняется программой, называемой *интерпретатором*
- При интерпретации программы она размещается в той области памяти вычислительной машины, которая предназначена для исходных данных выполняемых программ, интерпретатору также необходимы те же данные, что и при выполнении исходной программы
- В отличие от компилятора и ассемблера, *интерпретатор* исходного языка не просто обрабатывает текст исходной программы, а выполняет действия, которые этой программой предписываются

Принципиальное отличие интерпретатора от компилятора

- Интерпретатор ***не порождает целевую программу***, которая впоследствии должна выполняться, а выполняет её сам
- Итогом работы интерпретатора является результат, определяемый смыслом исходной программы, если исходная программа правильна синтаксически и семантически, либо сообщение об ошибке, в противном случае

Смешанная стратегия трансляции

- При работе интерпретаторов сначала производится преобразование исходной программы в некоторое внутреннее представление, которое затем программно интерпретируется
- Именно поэтому *интерпретаторы относят к трансляторам*
- Термин *транслятор* является самым общим и обозначает, как *компиляторы* и *ассемблеры*, так и *интерпретаторы*

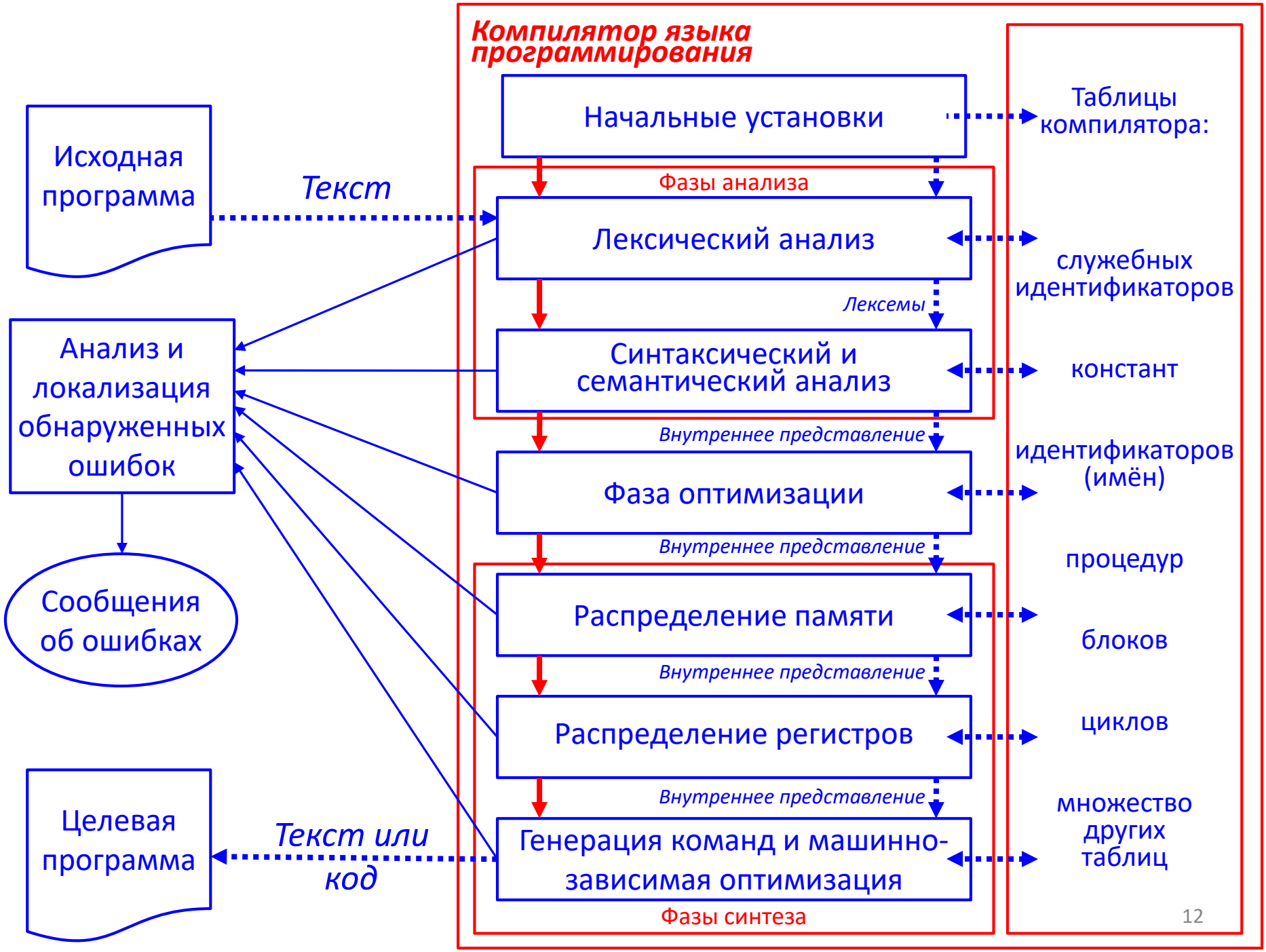

Компилятор языка программирования
Исходная программа
Текст
Начальные установки
Таблицы компилятора:
Фазы анализа
Лексический анализ
служебных идентификаторов
Лексемы
Синтаксический и семантический анализ
констант
Внутреннее представление
Анализ и локализация обнаруженных ошибок
Фаза оптимизации
идентификаторов (имён)
Внутреннее представление
процедур
Сообщения об ошибках
Распределение памяти
блоков
Внутреннее представление
Распределение регистров
циклов
Внутреннее представление
Целевая программа
Текст или код
Генерация команд и машинно-зависимая оптимизация
множество других таблиц
Фазы синтеза

Схема работы компилятора

- Информационные таблицы
  - Таблицы служебных идентификаторов
  - Таблица констант
  - Таблица имён
  - Таблица процедур
  - Таблица блоков
  - Таблица циклов
  - Множество других таблиц

Схема работы компилятора

1. Начальные установки
2. Фазы анализа программ
    2.1. Лексический анализатор
        - Выделение и обработка лексем
        - Замена сложных конструкций
            - Диграфы (‘*:=*’, ‘****’, ‘*->*’, ‘*&&*’)
              и триграфы (‘*??<*’ ≡‘*(*’, ‘*??>*’ ≡‘*)*’, ‘*??=*’ ≡‘*#*’)
        - Замена знаков операций
            - (‘*AND*’, ‘*not*’, ‘*mod*’)
        - Макрорасширение
        - Исключение примечаний

Схема работы компилятора

2. Фазы анализа программ
    2.2. Синтаксический и семантический анализаторы
        - Проверка программ на синтаксическую и семантическую правильность
        - Формирование внутреннего представления для каждой составной части программы

Внутреннее представление программ

- Внутреннее представление программы в трансляторе зависит от обработки, которой должна подвергнуться программа
- В ассемблерах внутреннее представление близко к окончательному виду программы
- Внутреннее представления в компиляторах
  - двусвязные или древовидные списки
  - линейные последовательности операторов

Схема работы компилятора

3. Фазы оптимизации программ

Стратегии оптимизации

- Повышение скорости работы
- Уменьшение размеров программы

«Машинно-зависимая» оптимизация

«Машинно-независимая» оптимизация

Независимость рассматривается в терминах некоторого класса вычислительных архитектур

Схема работы компилятора

4. Фазы синтеза программ

    4.1. Распределение памяти и регистров

    4.2. Генерация команд и машинно-зависимая оптимизация

- В интерпретаторах фазы синтеза заменяются программой, которая фактически выполняет (интерпретирует) внутреннее представление исходной программы

Однопроходный компилятор

- *Проход* – процесс последовательного чтения компилятором данных из внешней памяти, их обработки и записи результата во внешнюю память
- Количество проходов в разных компиляторах может исчисляться от одного до нескольких десятков
- Во время одного прохода может выполняться сразу несколько фаз компиляции, но некоторые фазы компиляции могут выполняться за несколько проходов

Однопроходный компилятор