

ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО  
ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ  
УФИМСКИЙ ГОСУДАРСТВЕННЫЙ АВИАЦИОННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Кафедра ТК

курс лекций по дисциплине

# Методы построения трансляторов

Тема: Общая схема работы компилятора

Преподаватель: к.т.н., доцент Карамзина А.Г.

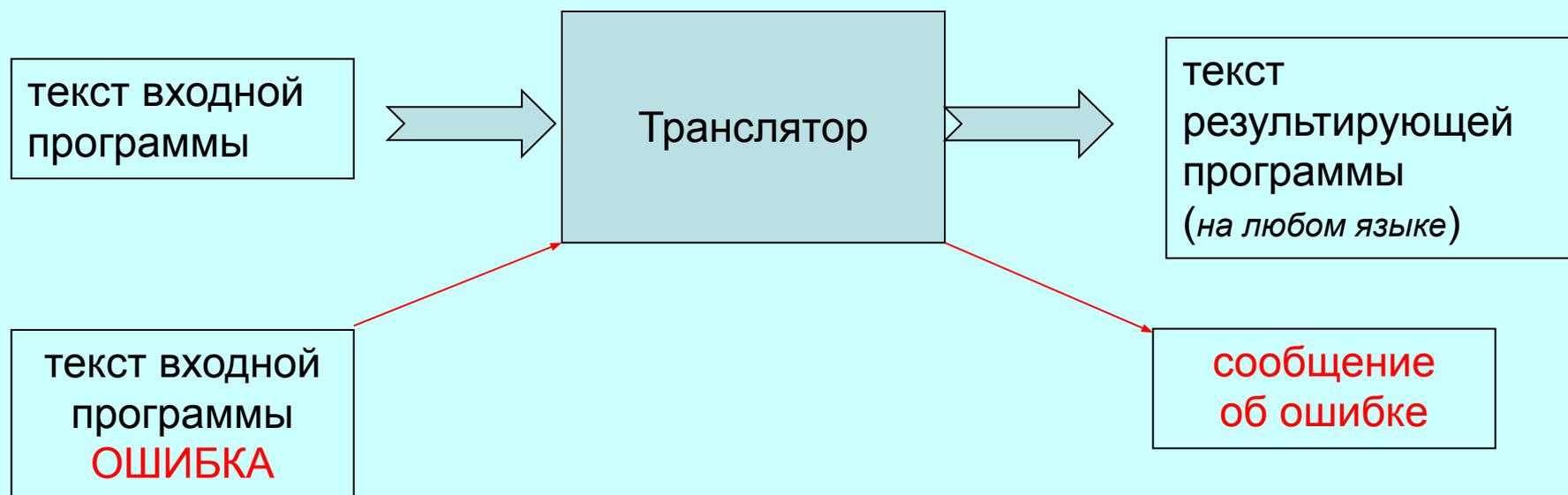
# Тема № 1

## Общая схема работы компилятора

- Определение и назначение транслятора
- Определение и назначение компилятора
- Определение и назначение интерпретатора
- Этапы трансляции
- Многопроходные и однопроходные компиляторы

## Определение и назначение транслятора

**Транслятор** – это программа, которая переводит входную программу на исходном (*входном*) языке в эквивалентную ей выходную программу на результирующем (*выходном*) языке.



## Определение и назначение компилятора

**Компилятор** – это транслятор, который осуществляет перевод исходной программы в эквивалентную ей объектную программу на языке машинных команд или на языке ассемблера.

*Файл, в который записана результирующая программа, обычно называется «объектным файлом».*

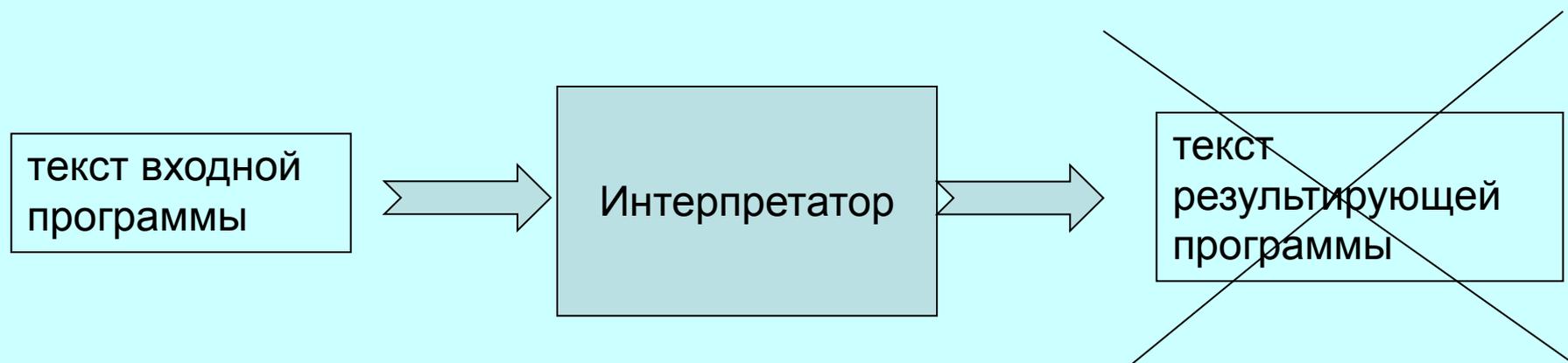
*Даже в том случае, когда результирующая программа порождается на языке машинных команд, между объектной программой (объектным файлом) и исполняемой программой (исполняемым файлом) есть существенная разница.*

*Порожденная компилятором программа не может непосредственно выполняться на компьютере, так как она не привязана к конкретной области памяти, где должны располагаться ее код и данные.*

программный модуль, являющийся результатом компиляции исходного модуля, представляющий собой последовательность машинных команд, готовый к объединению с другими объектными модулями.

## Определение и назначение интерпретатора

**Интерпретатор** – это программа, которая воспринимает входную программу на исходном языке и сразу выполняет ее.





Компилятор в целом с точки зрения формальных языков выполняет две основные функции:

-является распознавателем для языка исходной программы

получив на вход цепочку символов входного языка, проверяет ее принадлежность языку и выявляет правила, по которым эта цепочка была построена - генератор цепочек – автор входной программы;

-является генератором для языка результирующей программы

строит на выходе цепочку выходного языка по определенным правилам, предполагаемым языком машинных команд или языком ассемблера распознавателем этой цепочки является вычислительная система, под которую создается результирующая программа.

## Основные фазы компиляции

- **Лексический анализ (сканер)** преобразовывает литеры программы на исходном языке в токены исходного языка. На вход лексического анализатора подается исходный текст программы, а выходная информация передается компилятором на этапе синтаксического анализа.

- **Синтаксический анализ (разбор)** выполняется на этапе анализа. Она выполняет проверку корректности текста исходной программы, обрабатывая токены. На этой же фазе компиляции производится выделение структуры программы.

- **Семантический анализ** — это часть компилятора, проверяющая правильность текста исходной программы. Кроме непосредственной проверки, она выполняет преобразования текста программы (такие, как добавление функций, оптимизации). В различных реализациях компиляторов семантический анализ входит в фазу синтаксического анализа или фазу генерации кода.

```
program m4;
var c:boolean;
    a,b:real;
begin
    if a>b then #:=a-b else
a:=b*0.3;
end
```

```
program m4;
var c:boolean;
    a,b:real;
begin
    if a:=b else a:=a-b then
a:=b*0.3;
end.
```

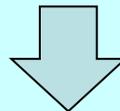
```
program m4;
var c:boolean;
    a,b:real;
begin
    if a>b then c:=a-b else
a:=n*0.3;
end.
```

ает  
мы)  
ной  
тки  
на  
в  
на  
сть  
я  
го  
ен  
ка  
в  
но  
к

## Основные фазы компиляции

**Подготовка к генерации кода** – это фаза, на которой компилятором выполняются предварительные действия, непосредственно связанные с синтезом текста результирующей программы, но еще не ведущие к порождению текста на выходном языке. Обычно в эту фазу входят действия, связанные с идентификацией элементов языка, распределением памяти и т. п.

**Генерация кода** – это фаза, непосредственно связанная с порождением команд, составляющих предложения выходного языка и в целом текст результирующей программы. Это основная фаза на этапе синтеза результирующей программы.



Кроме непосредственного порождения текста результирующей программы, генерация обычно включает в себя также оптимизацию – процесс, связанный с обработкой уже порожденного текста. Иногда оптимизацию выделяют в отдельную фазу компиляции, так как она оказывает существенное влияние на качество и эффективность результирующей программы.

**Таблицы идентификаторов** (*таблицы символов*) – это специальным образом организованные наборы данных, служащие для хранения информации об элементах исходной программы, которые затем используются для порождения текста результирующей программы.

Таблица идентификаторов в конкретной реализации компилятора может быть одна, или же таких таблиц может быть несколько.

Элементами исходной программы, информацию о которых нужно хранить в процессе компиляции, являются переменные, константы, функции и т. п. – конкретный состав набора элементов зависит от используемого входного языка программирования.

Понятие «таблицы» вовсе не предполагает, что это хранилище данных должно быть организовано именно в виде таблиц или других массивов информации.

## Многопроходные и однопроходные компиляторы

Реальные компиляторы  
программы за несколько

**Проход** – это процесс  
внешней памяти, их  
память.

- оперативная память компьютера,
- накопители на магнитных дисках,
- накопители на магнитных лентах и т. п.

Чаще всего один проход включает  
в себя выполнение одной или нескольких фаз компиляции.  
Результатом промежуточных проходов  
является **внутреннее представление** исходной программы,  
результатом последнего прохода  
– **резльтирующая объектная программа**.

Однопроходные  
же пор

Реаль

В современных системах программирования  
нередко первый проход компилятора (лексический анализ кода)  
выполняется параллельно с редактированием  
кода исходной программы, то есть выполняется  
**лексический анализ «на лету»**, который является  
функцией текстового редактора системы программирования,  
закрывающийся в поиске и выделении лексем входного языка  
в тексте программы непосредственно  
в процессе ее создания разработчиком.