Лекция 8.

Общая структура языков программирования

Сравнение ЯП с естественными яыками

Язык программирования — формальная знаковая система, предназначенная для записи (по определенным правилам) компьютерных программ.

Сравнение ЯП с естественными яыками

Язык программирования — формальная знаковая система, предназначенная для записи (по определенным правилам) компьютерных программ.

Сходство с естественными языками:

- Текст состоит из слов, составленных из символов.
- **Синтаксис**, определяющий структуру текста, отличается от **семантики**, определяющей смысл.
- Есть слова с предопределенным смыслом (if, do, while, ...). Можно определять собственные слова.

Синтаксис, грамматика и семантика

Глокая куздра штеко будланула бокра и курдячит бокрёнка

Синтаксис, грамматика и семантика

Глокая куздра штеко будланула бокра и курдячит бокрёнка

1. Тигрица быстро победила буйвола и грызет буйволенка.
2. Слон весело разбил бочку и катает бочонок.

Синтаксис, грамматика и семантика

Глокая куздра штеко будланула бокра и курдячит бокрёнка

1. Тигрица быстро победила буйвола и грызет буйволенка.
2. Слон весело разбил бочку и катает бочонок

Синтаксис, грамматика и семантика

Глокая куздра штеко будланула бокра и курдячит бокрёнка

1. Тигрица быстро победила буйвола и грызет буйволенка.
2. Слон весело разбил бочку и катает бочонок

Z=X+Y

1. Z=10, X=2, Y=8
2. Z=10, X=2, Y=120

Сравнение ЯП с естественными яыками

- ЯП не позволяют выражать чувства или мысли. Они определяют объекты, представимые в компьютере, и задачи, выполнимые над этими объектами.
- ЯП выигрывают в точности. Синтаксис и семантика ЯП должны быть определены совершенно строго.

ЯП влияет, независимо от нашего желания, на наш способ мышления.

Сравнение ЯП с естественными яыками

- ЯП не позволяют выражать чувства или мысли. Они определяют объекты, представимые в компьютере, и задачи, выполнимые над этими объектами.
- ЯП выигрывают в точности. Синтаксис и семантика ЯП должны быть определены совершенно строго.

ЯП влияет, независимо от нашего желания, на наш способ мышления.

Естественные языки используются в комментариях и служат основой для построения идентификаторов (имен переменных, подпрограмм, методов, …).

Языки программирования: основные определения

- **Язык** – множество строк.
- **Выражения** могут принадлежать или не принадлежать языку.

  “a:=123.45;” “вася - молодец”
- **Оператор** – минимальная единица программного кода, элементарные действия, из которых строятся более сложные. Могут быть одиночными и блочными операторы.

Языки программирования: основные определения

- **Синтаксис** программы = структура и форма записи ее текста. Синтаксис задается правилами, позволяющими определить, принадлежит ли заданное выражение языку или нет.

```
write("Привет!");
```

```
echo "Привет!";
```

Языки программирования: основные определения

- **Синтаксис** программы = структура и форма записи ее текста. Синтаксис задается правилами, позволяющими определить, принадлежит ли заданное выражение языку или нет.

```
write("Привет!");
```

```
echo "Привет!";
```

- Синтаксическая единица самого нижнего уровня – **лексема. Лексемы** – последовательности символов языка, которые имеют смысл только как единое целое.

  Выражение “2+3” – не лексема, идентификатор `TForm` – лексема.

Языки программирования: основные определения

- **Синтаксис** программы = структура и форма записи ее текста. Синтаксис задается правилами, позволяющими определить, принадлежит ли заданное выражение языку или нет.

```
write("Привет!");
echo "Привет!";
```

- Синтаксическая единица самого нижнего уровня – **лексема**. **Лексемы** – последовательности символов языка, которые имеют смысл только как единое целое.

  Выражение “2+3” – не лексема, идентификатор `TForm` – лексема.

- **Семантика** языка – описание смысла языковых выражений. Семантику составляют правила, определяющие, к выполнению каких действий приведет то или иное выражение.

Разница между синтаксисом и семантикой

Оператор присваивания в Pascal

```
x:=y+z;
```

Синтаксически это правильное выражение?

Разница между синтаксисом и семантикой

Оператор присваивания в Pascal

```
x:=y+z;
```

Синтаксически это правильное выражение?
Может ли оно быть семантически неверным?

Разница между синтаксисом и семантикой

Оператор присваивания в Pascal

x:=y+z;

Синтаксически это правильное выражение?
Может ли оно быть семантически неверным?

- Существуют формальные способы описания синтаксиса, позволяющие выделить отдельные синтаксические конструкции. Поэтому можно делать вывод о синтаксической корректности отдельной части выражения языка.
- Формальных правил описания семантики нет.

Лексика, синтаксис и семантика

Язык программирования определяет набор лексических, синтаксических и семантических правил:

- Лексика и синтаксис задают внешний вид программы.
- Семантика определяет действия, которые выполнит компьютер под её управлением.

Анализ текста программы

- **Лексический анализ** – разделение выражения на отдельные лексемы. Также анализатор может удалять из выражения комментарии, лишние разделители и т.д.
- **Синтаксический анализ** – проверка правильности выражений, составленных из лексем.

Самый распространенный способ формального описания синтаксиса языка программирования – расширенная форма Бэкуса-Наура (РБНФ). Предложена Джоном Бэкусом и модифицирована Питером Науром (для описания Алгола).

Позволяет описать **грамматику языка** – совокупность правил, записанных в виде РБНФ.

Джон Бэкус (*John Backus*, 3 декабря 1924 года — 17 марта 2007 года) — американский учёный в области информатики. Он был руководителем команды, разработавшей первый высокоуровневый язык программирования ФОРТРАН, изобретателем формы Бэкуса — Наура, одной из самых универсальных нотаций, используемых для определения синтаксиса формальных языков

Петер Наур (*Peter Naur*; 25 октября 1928, Фредериксберг — 3 января 2016, Херлев) — датский учёный в области информатики, один из пионеров компьютерной науки. Более всего известен как один из разработчиков первого языка структурного программирования Алгол 60 и, совместно с Бэкусом, как изобретатель формы Бэкуса — Наура.

Основы БНФ

Любой язык программирования имеет свой алфавит, который целиком состоит из терминальных и нетерминальных символов.

- **Терминальные символы** (=лексемы) — это отдельные символы или их последовательности, имеющие конкретные известные значения и являющиеся с точки зрения синтаксиса неразрывным целым, не сводимым к другим символам. Пишутся в одинарных кавычках.

Пример – цифры, буквы, зарезервированные слова ('if', 'do', 'begin', …).

- **Нетерминальные символы** – элементы языка, не имеющие заранее известного значения (формулы, команды и т.п.), которые по определенным правилам сводятся к комбинации терминальных и/или других нетерминальных символов. Правила должны быть такими, чтобы существовала возможность выведения из них выражения, полностью состоящего из терминальных символов, за конечное число шагов.

Основы БНФ

Нетерминальные символы имеют имена, заключаются в угловые скобки.

- Правило в РБНФ имеет вид «идентификатор ::= выражение», идентификатор – это имя нетерминального символа,
- выражение – это соответствующая правилам РБНФ комбинация терминальных и нетерминальных символов,
- операция **::=** определяет нетерминальный символ (*читается как "есть по определению"*) .

Основы БНФ

Набор возможных конструкций РБНФ:

- конкатенация символов,
- выбор,
- условное вхождение,
- повторение.

<Separator> ::='.' (определение нетерминального символа для записи вещественных чисел)
<Assignment> ::=<Var> ':=' <Expression> (описание синтаксиса оператора присваивания в Pascal)

Основы БНФ

Фигурные скобки означают повторение того, что в них стоит, ноль или более раз.

<Unsigned> ::= <Digit>{<Digit>} (определение целого числа без знака)
<Assignment> ::=<Var> ':=' <Expression> (описание синтаксиса оператора присваивания в Pascal)

<Digit> ::= '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9' (| или)
<for> ::= 'for' <Var> ':=' <Expression> ('to' | 'downto') <Expression> 'do' <Operator> (описание синтаксиса оператора for в Pascal)

В квадратные скобки заключается необязательная часть определения.
<if> ::= 'if' <Condition> 'then' <Operator> ['else' <Operator>] (описание синтаксиса условного оператора)

Основы БНФ

Задача. Описать с помощью БНФ синтаксис вещественного числа.

Основы БНФ

Задача. Описать с помощью БНФ синтаксис вещественного числа.

- Перед числом может стоять знак — плюс или минус.
- Затем идет одна или несколько цифр.
- Потом может следовать точка, после которой будет еще одна или несколько цифр. Затем может быть указан показатель степени "E" (большое или малое), после которого может стоять знак плюс или минус, а затем должна быть одна или несколько цифр".

Основы БНФ

Задача. Описать с помощью БНФ синтаксис вещественного числа.

- Перед числом может стоять знак — плюс или минус.
- Затем идет одна или несколько цифр.
- Потом может следовать точка, после которой будет еще одна или несколько цифр. Затем может быть указан показатель степени "E" (большое или малое), после которого может стоять знак плюс или минус, а затем должна быть одна или несколько цифр".

<Number> ::= [<Sign>] <Digit> {<Digit>}[<Separator> <Digit> {<Digit>}][<Exponent> [<Sign>] <Digit> {<Digit>}]

<Digit> ::= '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9'
<Sign> ::= '+' | '-'
<Separator> ::= '.'
<Exponent> ::= 'E' | 'e'

БНФ и анализ синтаксиса

Имея описание выражений с помощью БНФ, можно писать **программы-лексеры** для разбора синтаксиса выражений и **программы-парсеры**, понимающие грамматику выбранного языка программирования.

Лексер – программный модуль, осуществляющий разбор текста по заданным грамматическим правилам и генерирующий поток токенов , т.е. нетерминальных символов языка программирования.

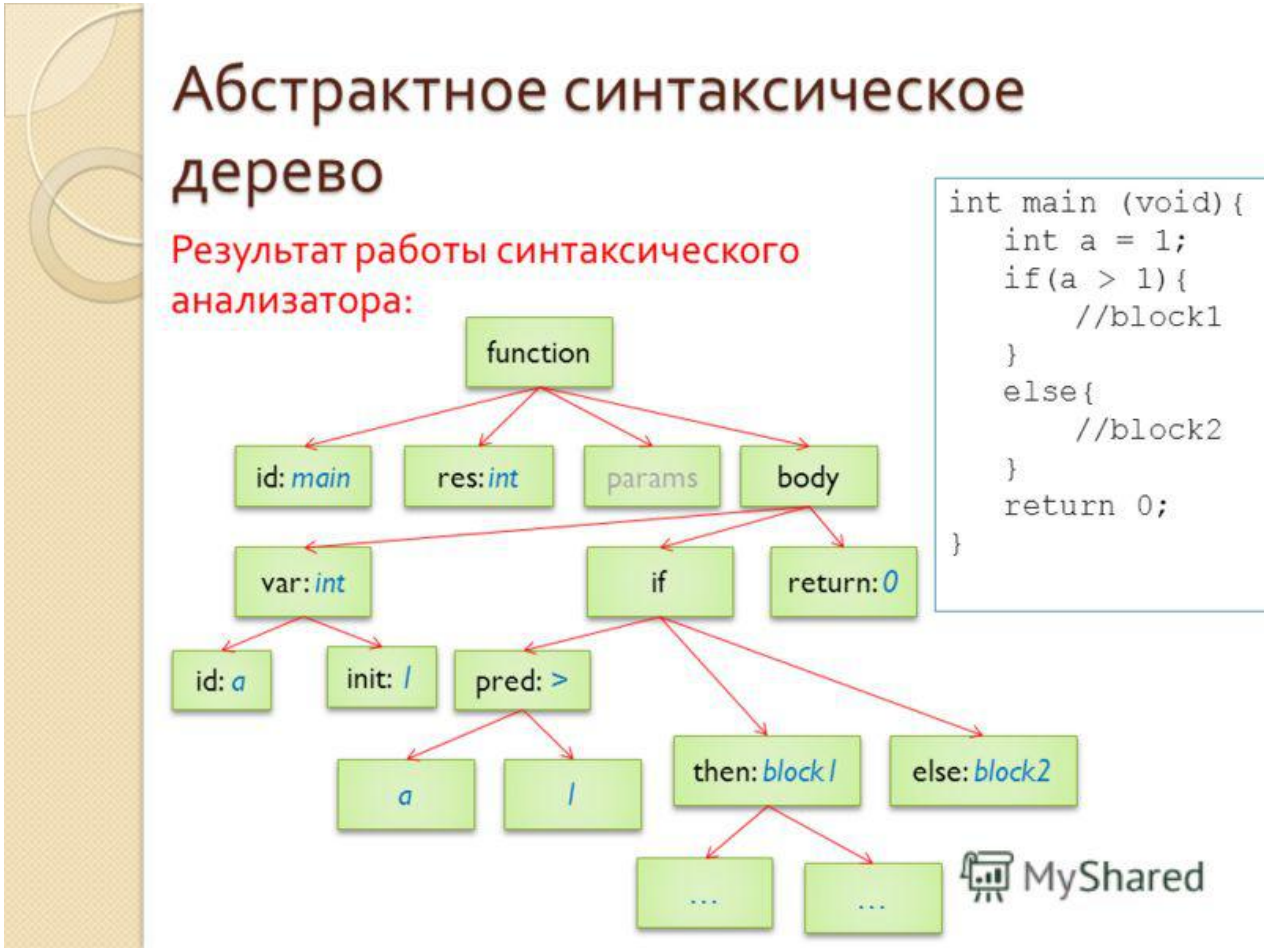
БНФ и анализ синтаксиса

Лексер должен выделять в коде сущности, описываемые грамматикой:

- функции, методы и процедуры, а также их вызовы;
- объекты и их вложенность;
- поля элементарных типов;
- комментарии;
- директивы;
- блоки кода и управляющие конструкции;
- различные атрибуты любой из перечисленных сущностей.

Парсер – программный модуль, который, основываясь на поступающих из потока лексера токенах, по заданным правилам строит **абстрактное синтаксическое дерево**, отражающее иерархию элементов программы и связи между ними.

Синтаксическая структура может быть задана в виде **абстрактного синтаксического дерева.**



Для выполнения написанной программы производится анализ и преобразование этого дерева компилятором или интерпретатором.

AST для JavaScript

https://astexplorer.net/

```
function foo(x) {
    if (x > 10) {
        var a = 2;
        return a * x;
    }

    return x + 10;
}
```