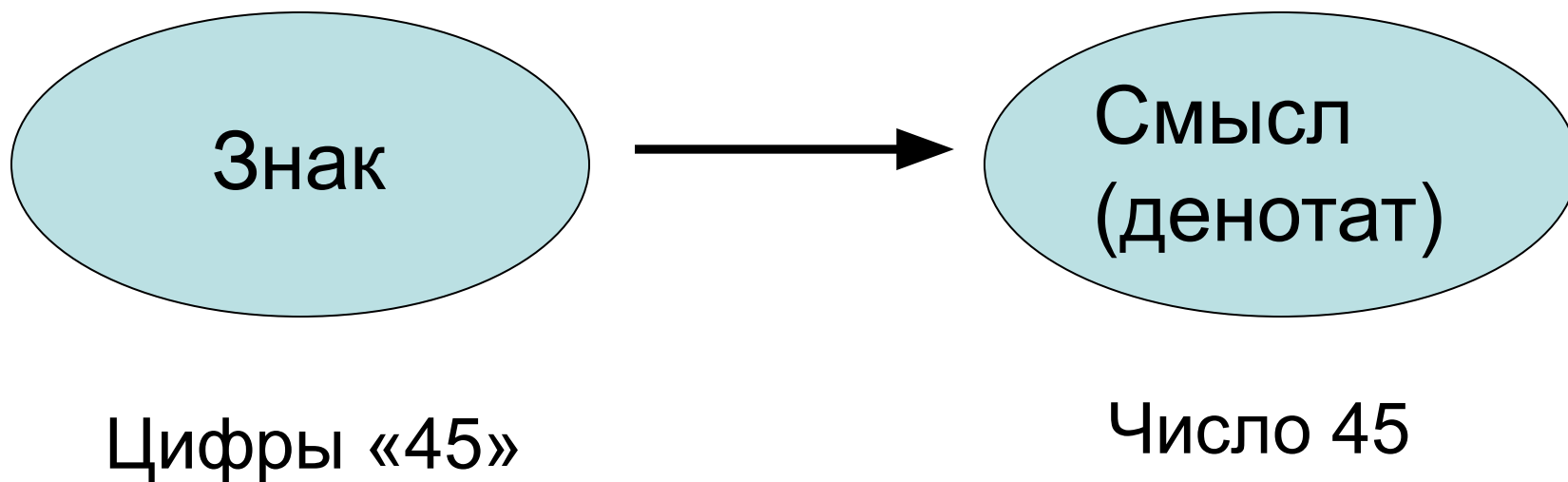


# Языки программирования

Язык – знаковая система



Семантическая функция  $Val(\text{«45»}) = 4 * 10 + 5$

# Языки программирования

- Лексика
  - Орфография
  - Морфология
- Синтаксис
  - Грамматика
  - Пунктуация
- Семантика
  - Прагматика
- Стил

# Лексика

*Лексема* – элементарная (относительно синтаксиса) единица языка

Примеры:

- Числа: 123.4e2, 12, 0x25
- Знаки: +, !=, [, <<, <
- Идентификаторы: i, Pi2, PersonID
- Ключевые слова: **while**, **if**
- Строки: “Hello, World”, “while + 1”
- Символы: ‘a’

# Лексика - пример

*Идентификатор* – последовательность букв и цифр, начинающаяся с буквы

Вопросы:

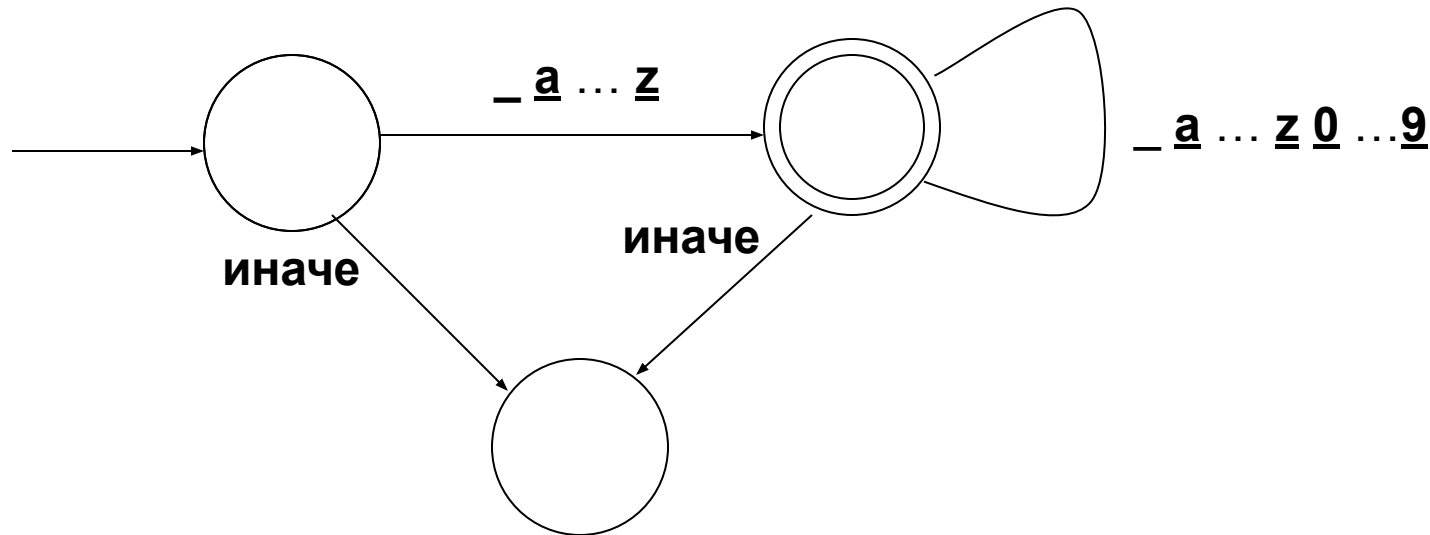
- Кириллица? `Инд2`
- Регистр? `PersonID = PeRSonID`
- `_`? `student_count, __FILE__, _1`
- Длина? `TheBestApproximationReachedSoFar`
- Другие символы? `IsLegal?`
- Пробелы? `Min X`

# Лексика – формальное описание

- Регулярные выражения

$(\_|\underline{\mathbf{a}}|\dots|\underline{\mathbf{z}})(\_|\underline{\mathbf{a}}|\dots|\underline{\mathbf{z}}|\underline{\mathbf{0}}|\dots|\underline{\mathbf{9}})^*$

- Конечные автоматы



# Форма Бэкуса-Наура - БНФ

- *Нетерминал* – определяемое понятие
- Терминал – неопределяемый символ
- *Метасимволы* – ( ) ::= [ ] \*

## Правило грамматики

*Нетерминал* ::= последовательность терминалов и нетерминалов

# Пример БНФ

*буква ::= а*

*буква ::= б*

...

*буква ::= з*

*цифра ::= 0*

...

*цифра ::= 9*

*букра ::= буква*

*букра ::= цифра*

*букры ::=*

*букры ::= буква*

*букры*

*идент ::= буква*

*букры*

# Регуляризованная БНФ - РБНФ

## *Альтернатива*

*разное* ::= вариант<sub>1</sub>

...

*разное* ::= вариант<sub>n</sub>

## Эквивалентно

*разное* ::= вариант<sub>1</sub> | ... | вариант<sub>n</sub>

## Пример

*буква* ::= \_ | a | ... | z



# Регуляризованная БНФ - РБНФ

*Необязательный элемент* – возможное отсутствие

*можетбыть ::=*

*можетбыть ::= нечто*

Эквивалентно

*можетбыть ::= [ нечто ]*

Пример

*букры ::= [ букра букры ]*

# Регуляризованная БНФ - РБНФ

*Итерация* – повторение ноль или более раз (звезда Клини)

*много ::=*

*много ::= нечто много*

Эквивалентно

*много ::= (нечто)\**

Пример

*букры ::= (букра)\**

# Регуляризованная БНФ - РБНФ

*Ненулевая итерация* – повторение один или более раз (плюс Клини)

*много ::= нечто*

*много ::= нечто много*

Эквивалентно

*много ::= (нечто)<sup>+</sup>*

Пример

*букра (букра)<sup>\*</sup> эквивалентно (букра)<sup>+</sup>*

*(букра)<sup>\*</sup> эквивалентно [(букра)<sup>+</sup>]*

# Пример РБНФ

*буква ::= —*

*буква ::= а*

...

*буква ::= z*

*цифра ::= 0*

...

*цифра ::= 9*

*букра ::= буква*

*букра ::= цифра*

*букры ::=*

*букры ::= буква*

*букры*

*идент ::= буква*

*букры*

# Пример РБНФ

*буква ::= \_|a...|z*

*цифра ::= 0|...|9*

*букра ::= буква  
| цифра*

*букры ::= (букра)\**

*идент ::= буква  
букры*

# Пример РБНФ

• *буква* ::= \_ | a | ... | z

• *цифра* ::= 0 | ... | 9

*идент* ::= *буква*  
(*буква* | *цифра*)\*

# Лексика

- Разделители
  - Пробелы, переводы строк, табуляции
  - Значащие позиции: с 7 по 72
  - Комментарии: /\*...\*/ // до конца строки
  - Вложенные комментарии
- Максимальность лексемы: a+++++b, <<
- Нормализация
  - 1.23 = 0.123e+1
  - ZERO = ZEROS = ZEROES = 0
  - Count = COUNT = count

# Лексика – национальные версии (Алгол 60)

```
проц НОД(x,y,z);  
  знач x,y; цел x,y,z;  
  начало  
    цел проц ОСТ(A,B); знач A,B; цел A,B;  
    ОСТ := A – (A % B) * B;  
    начало  
      цел u;  
      для u := ОСТ(x,y) пока u ≠ 0 цикл  
        начало y := x; x := u конец;  
    конец;  
  z := x  
конец
```



# Лексика – национальные версии (проблемы)

- Для «правильного» перевода нужно менять не только лексику, но и синтаксис, структуру фраз
- Русские имена могут не допускаться окружающей обстановкой
- Использование «иноязычных» библиотек
- Изображение данных:
  - числа: десятичная точка или десятичная запятая
  - даты: 09/01/04 или 04/01/09
- Неудобство набора текста
  - опасность совпадения разных букв по начертанию

# Лексика

Результат – поток лексем

- *Тип лексемы*: идентификатор, строка, число...
- *Значение лексемы*: изображение, значение числа,.....

# Синтаксис

Правила построения фраз из лексем

- *Контекстно-свободный* - структура фразы не зависит от окружения
- *Контекстно-зависимый*

Пример (Algol-68): *.Ax := 2*

- Описание переменной с инициализацией, если A – тип
- Присваивание 2 по адресу (.Ax), если .A - операция

# Контекстно-свободный синтаксис

Пример (РБНФ)

$выр ::= пере\text{м}$

|  $конст$

|  $(\pm \mid -) выр$

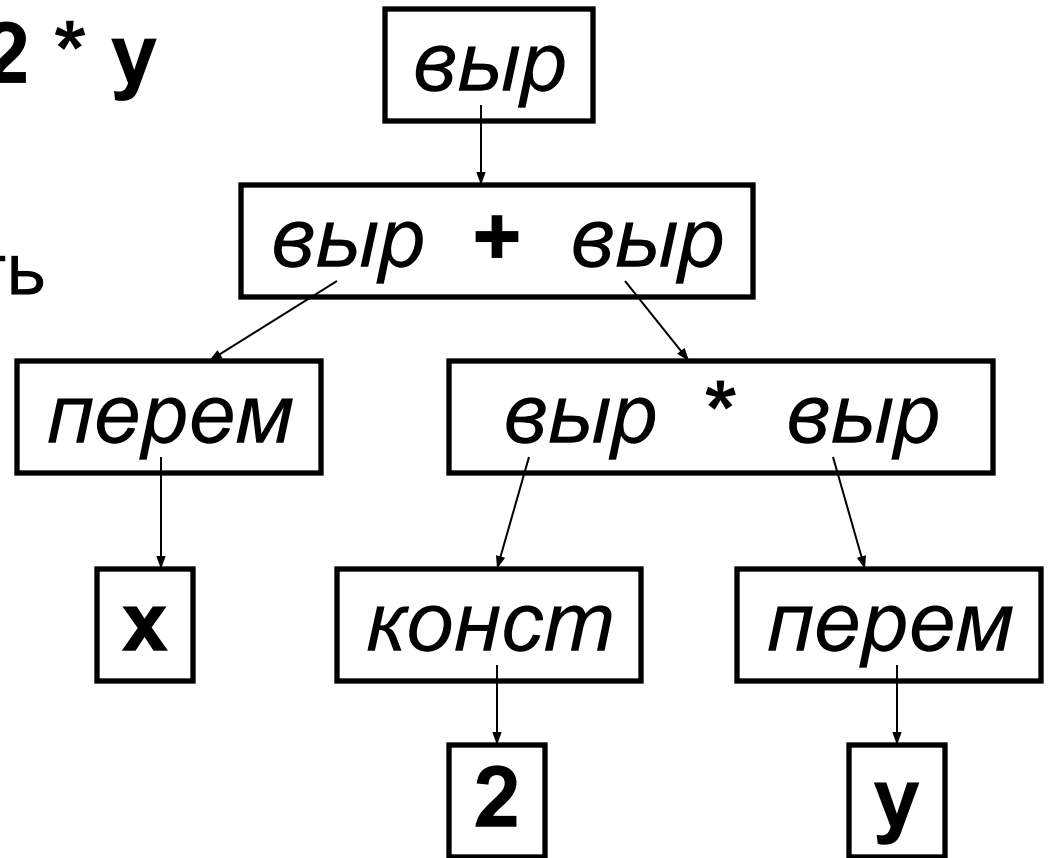
|  $выр (\equiv \mid \leq \mid \leq= \mid \leq> \mid \pm \mid - \mid * \mid /) выр$

|  $( выр )$

# Синтаксический вывод - дерево разбора

Выражение:  $x + 2 * y$

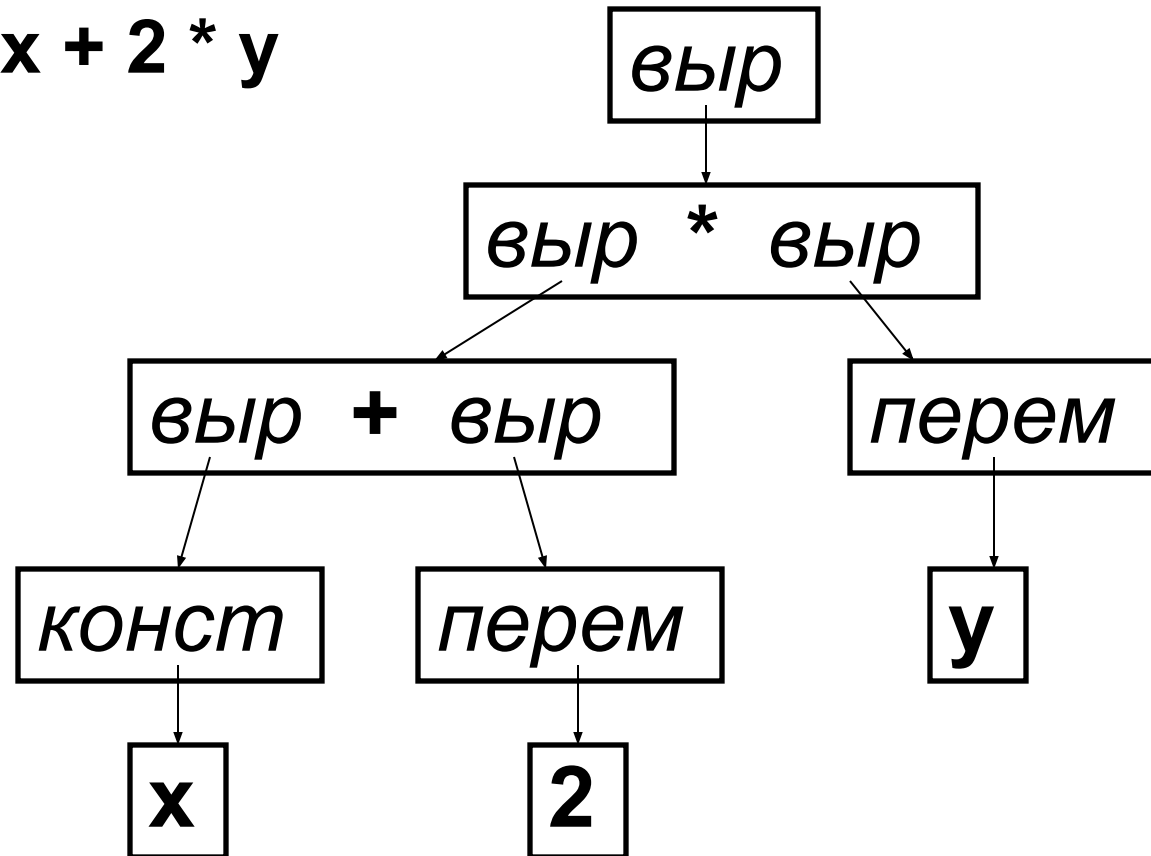
Задача: найти последовательность правил вывода для заданной цепочки терминалов



$((x) + ((2) * (y)))$

# Синтаксический вывод (неоднозначность)

Выражение:  $x + 2 * y$



$((x + 2) * y)$

# Синтаксический вывод (избыточность)

Допускается «лишнее»

Пример:

- $A < B + C < D$
- $+ - + 2$
- $X + - Y$

# Контекстно-свободный синтаксис

Пример – улучшенный вариант

$выр ::= прост-выр [(\equiv | \leq | \leq\equiv | \leq\>)]$   
 $прост-выр$

$прост-выр ::= [\pm | \mp] слаг ((\pm | \mp) слаг)^*$

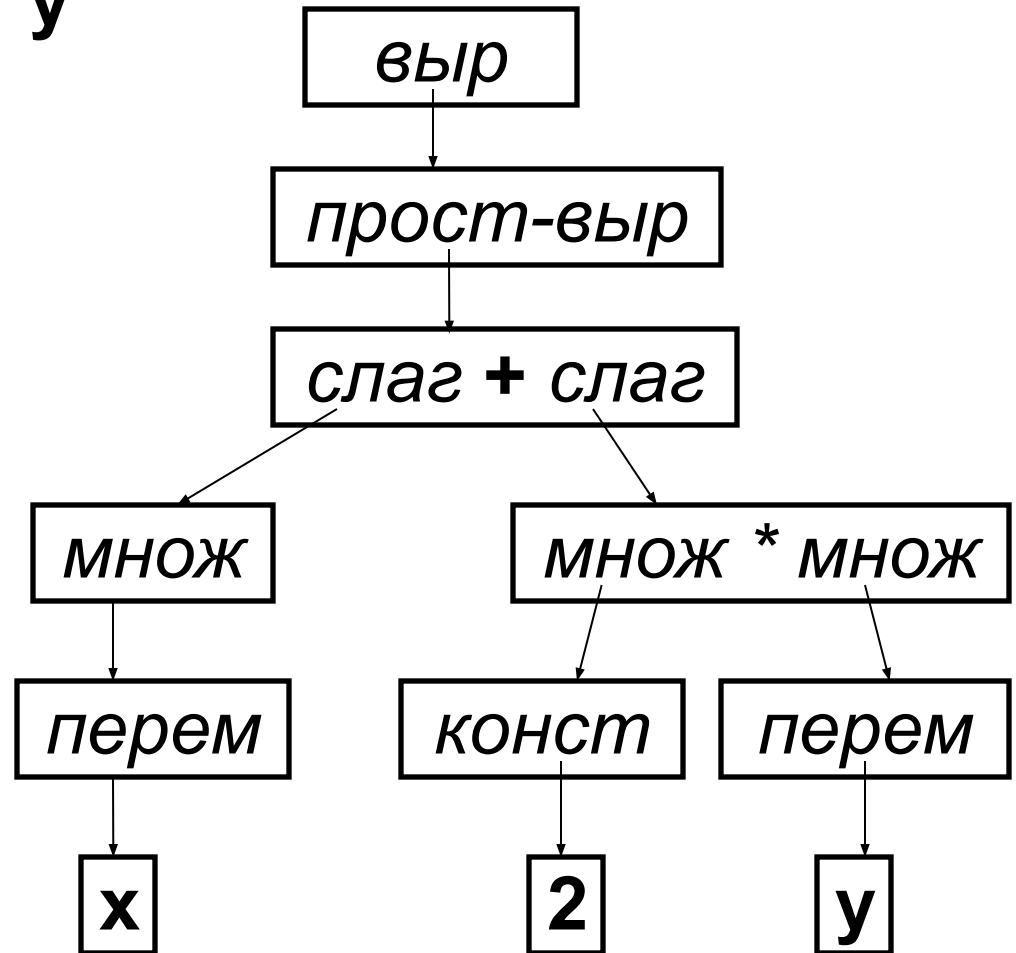
$слаг ::= множ ((\ast | \_ ) множ)^*$

$множ ::= (перемен | конст | (\_ выр \_))$



# Синтаксический вывод - дерево разбора

Выражение:  $x + 2 * y$



$(x + (2 * y))$

# Неоднозначность if

```
if (x > 0)
    if (x < 2)
        x = x+1;
else
    x = x-1;
```

```
if (x > 0)
    if (x < 2)
        x = x+1;
else
    x = x-1;
```

# Неоднозначность if

Решение проблемы:

```
if (x > 0)
```

```
    if (x < 2)
```

```
        x = x+1;
```

```
    fi
```

```
else
```

```
    x = x-1;
```

```
fi
```

# Синтаксические диаграммы

Структурированный ориентированный граф с одним входом и одним выходом, вершинами которого являются *нетерминалы* и **терминалы**

*Допускает* цепочку терминалов на пути от входа к выходу с «заходом» в диаграммы нетерминалов.

# Синтаксические диаграммы

- *Вход:*



- *Выход:*

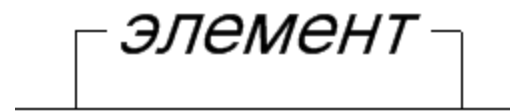


- *Обязательный:* — элемент1 — ... — элементn —

- *Необязательный*

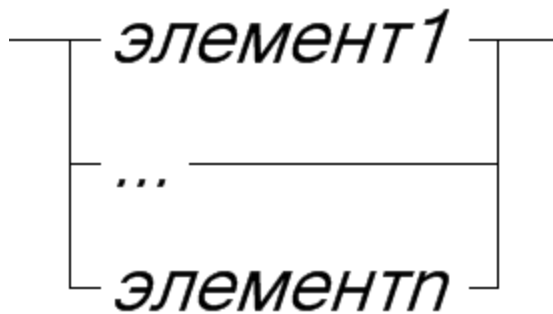


- *Игнорируемый:*



# Синтаксические диаграммы

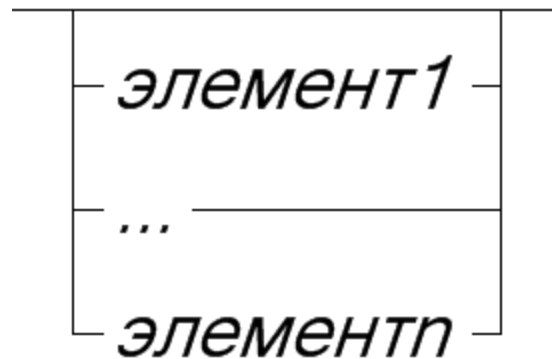
- *Выбор:*



- Выбор с умолчанием :



- *Необязательный выбор:*



# Синтаксические диаграммы

- *Повторение:*

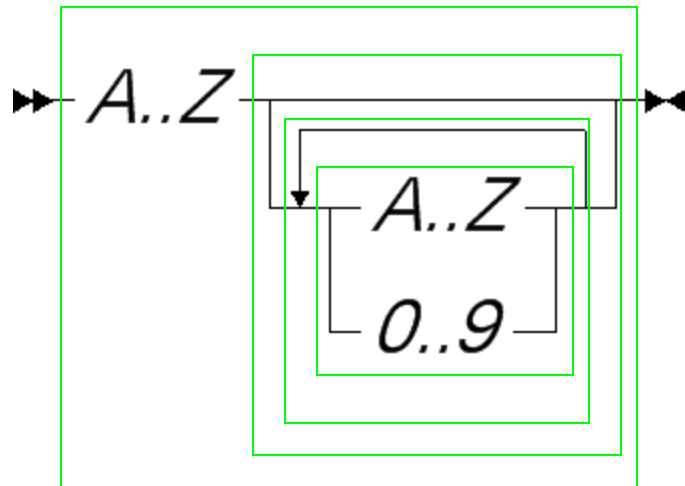


- *Повторение  
через  
разделитель:*



# Синтаксические диаграммы

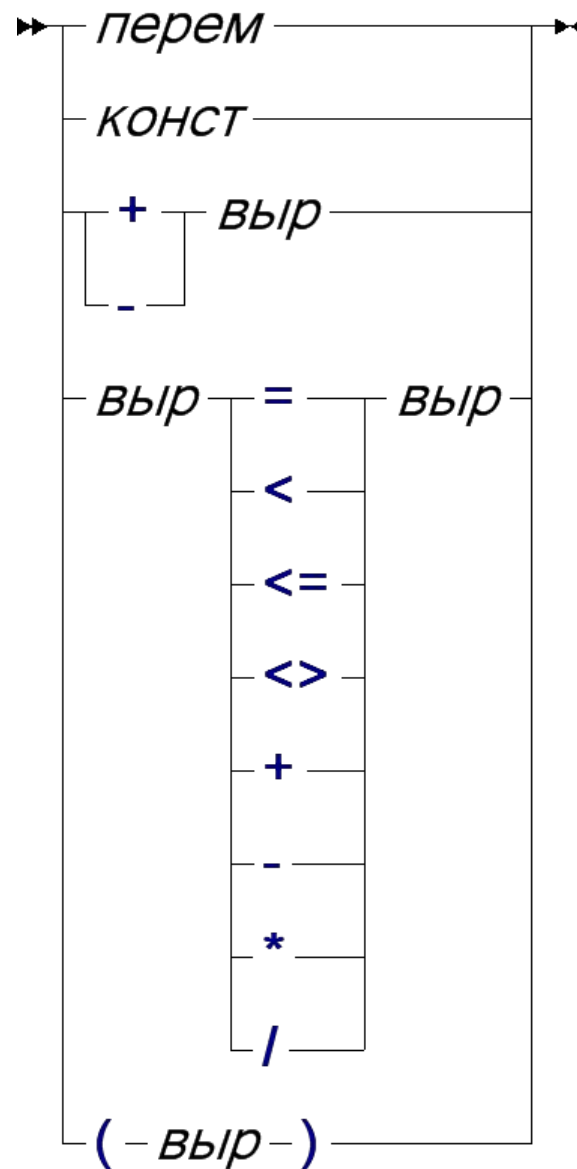
- *идент ::= A..Z [(A..Z | 0..9)\*]*





# Синтаксические диаграммы - пример

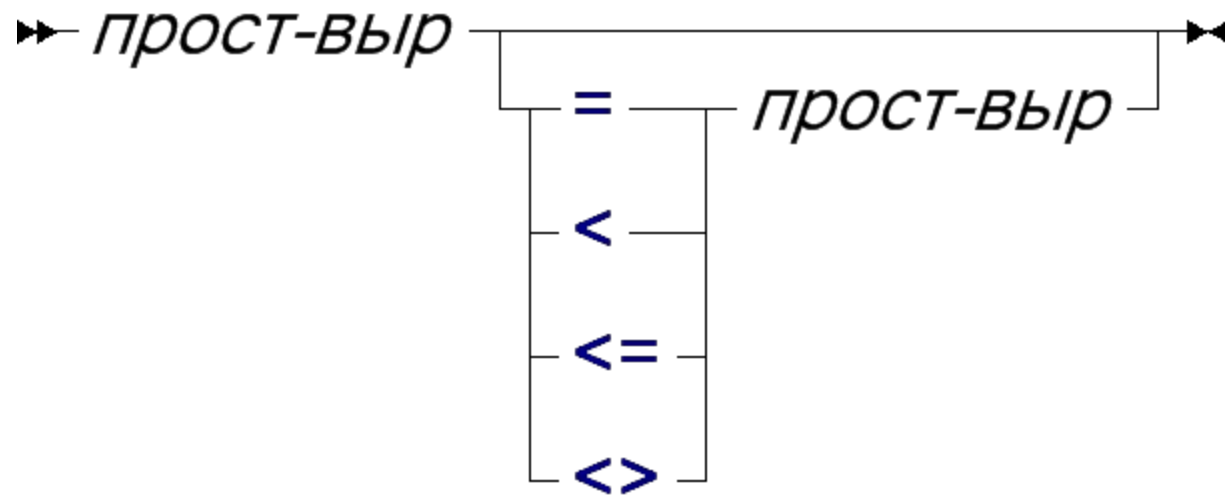
«Плохая» грамматика



# Синтаксические диаграммы - пример

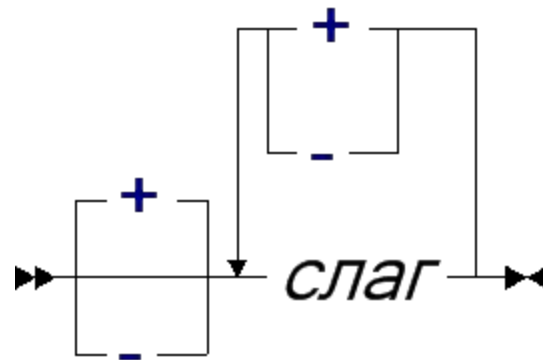
Улучшенная грамматика

- $выр ::= прост-выр [(= | < | <= | <>) прост-выр]$

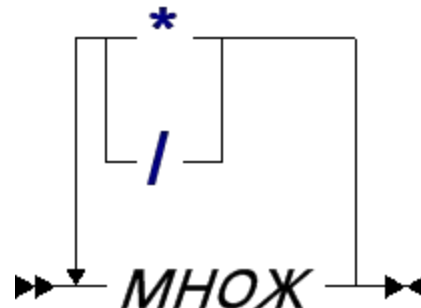


# Синтаксические диаграммы - пример

- $\text{прост-выр} ::= [\pm \mid \mp] \text{слаг} ((\pm \mid \mp) \text{слаг})^*$

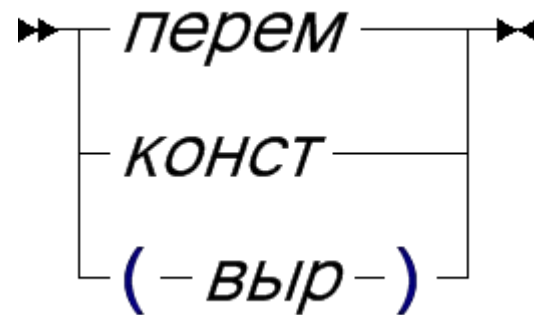


- $\text{слаг} ::= \text{множ} ((\ast \mid /) \text{множ})^*$



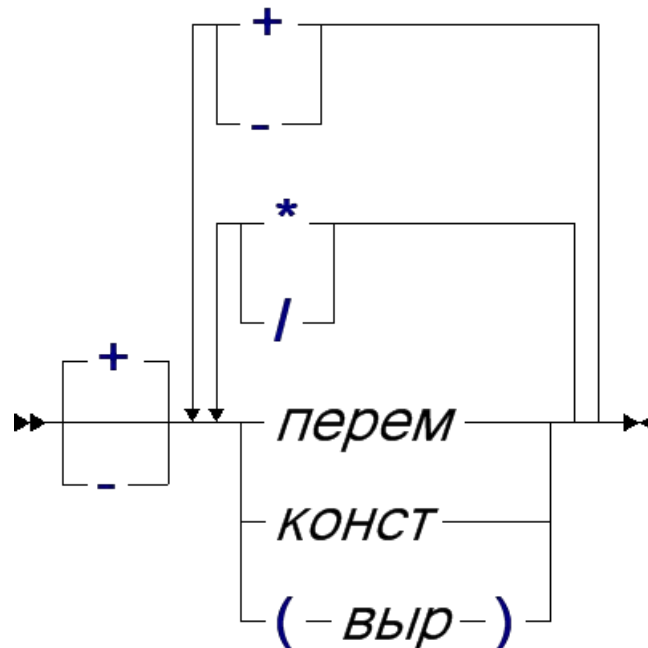
# Синтаксические диаграммы - пример

- *множ ::= перемен | конст | (выр )*



# Синтаксические диаграммы – ПОНЯТНОСТЬ ПОЛЬЗОВАТЕЛЮ

- Критерии
  - чтобы не было слишком большим (умещалось на странице)
  - чтобы не было слишком много диаграмм



# Устойчивость синтаксиса

- Случайные ошибки и опечатки должны обнаруживаться
- Разные конструкции должны визуально различаться
- Примеры:

C

```
for (i = 0; i < N - 1; i++);  
    A[i] = A[i - 1];  
  
for (float x = 0.0; x <= 1.2; f = +0.1)  
    s = + f(x);  
  
y = a[0]/2 + a[1]//3 + a[2]/5 + a[3]/7  
    + a[4]/11 + a[5]/13 + a[6]/17 + a[7]/19;
```

Fortran:

```
10 DO I = 1.1, N  
    S = S * I  
    CONTINUE
```

Algol-68:

```
.for i from 10 .to N .do  
    print(" ")  
.od;
```

# Контекстно-зависимый анализ

- Идентификация – сопоставление определений объектов с их использованиями
- Статический анализ типов – определение (вывод) типов объектов и выражений и проверка типовой правильности.

# Семантика

Что делает данная программа?

- Функциональная семантика – функция, реализуемая программой
- Операционная семантика – последовательность (содержательных) действий выполняемая программой
- Аксиоматическая семантика – следствие постусловий из предусловий



# Стиль

- *Лесенка* - иногда обязательна (Оссам), иногда поддерживается автоматически.

```
int l1 = busy_class(cl, d*lessons_per_day + t1);  
if (t1==t || l1==-1 || lessons[l1]-> share[0].teacher  
!= tch) continue; if (t1 < t-1 || t1>t+1) ++  
not_sequence; else {++ total_class_overload;  
sum += B_CLASS_OVERLOAD; }
```

Неправильно

# СТИЛЬ

- *Лесенка*

```
int l1 = busy_class(cl, d*lessons_per_day + t1);
```

```
if (t1 == t  
    || l1 == -1  
    || lessons[l1]->share[0].teacher != tch)  
    continue;
```

```
/* Не скупитесь на пробелы */
```

```
if (t1 < t-1 || t1 > t+1)  
    ++ not_sequence;  
else  
{  
    ++ total_class_overload;  
    sum += B_CLASS_OVERLOAD;  
}
```

# СТИЛЬ

- Лесенка else if

плохо

```
if (x >= 1000)
```

```
....
```

```
else
```

```
  if (x > 0)
```

```
  ...
```

```
  else
```

```
    if (x == 0)
```

```
    ...
```

```
  else
```

```
    if (x > -1000)
```

```
    ...
```

```
    else // x <= -1000
```

```
    ...
```

# Стиль

- Лесенка else if

```
if (x >= 1000)
    ....
else if (x > 0)
    ...
else if (x == 0)
    ...
else if (x > -1000)
    ...
else // x <= -1000
    ...
```

# Стиль

- Содержательные, мнемоничные идентификаторы

```
int n1, n2;
```

```
for (int index_of_outer_loop = 0;
```

```
    index_of_outer_loop < n1;
```

```
        index_of_outer_loop ++)
```

```
    for (int intIndexJ = 0; intIndexJ < n2; intIndexJ ++)
```

```
        ...
```

Неправильно

# Стиль

- Содержательные идентификаторы

```
int PersonCount, ExamCount;  
for (int p = 0; p < PersonCount; p++)  
    for (int j = 0; j < ExamCount; j ++)  
        ...
```

- Длина идентификатора пропорциональна размеру области его действия

# Стиль

- Неиспользование умолчаний

```
int cnt = 0;
```

```
unsigned char line[128]
```

```
FILE * file;
```

```
...
```

```
while ( fgets(line, 127, file) != NULL)
```

```
    cnt ++;
```

# Комментарии

Совсем без комментариев – **плохо**

```
int max = 0;
for (int i = 0; i < n; i++)
    if (M[i] > max)
        max = M[i];
```



# Комментарии

С плохими комментариями – ещё хуже

```
/* начальник приказал написать  
   комментарии к каждой строчке  
   – ему же хуже будет :-[ */
```

```
int max = 0; // присвоить 0
```

```
// перебираем i=0..n-1
```

```
for (int i = 0; i < n; i++)
```

```
    if (M[i] > max) // сравниваем с max
```

```
        max = M[i]; // обновляем, если надо
```

# Комментарии

*Комментарии облегчают понимание*

```
/*  
 * Нахождение максимума max в массиве M  
 */  
int max = 0; // предполагается, что все M[i] > 0  
for (int i = 0; i < n; i++)  
    if (M[i] > max)  
        max = M[i];
```

# Прагматика

Использование конструкций языка  
согласно их предназначению

```
while (n<0)
{
    n = -n;
    break;
}
```

```
if (n<0)
    n = -n;
```

```
n = (n<0 ? -n : n);
```

# Преимственность

- Fortran -> Fortran IV -> Fortran 77 -> Fortran 90
- K&R C -> ANSI C -> C++ -> C#
- Обратная совместимость