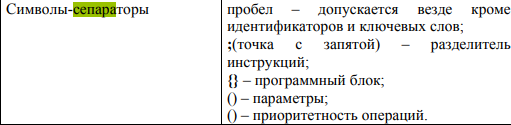
спецификация языка программирования

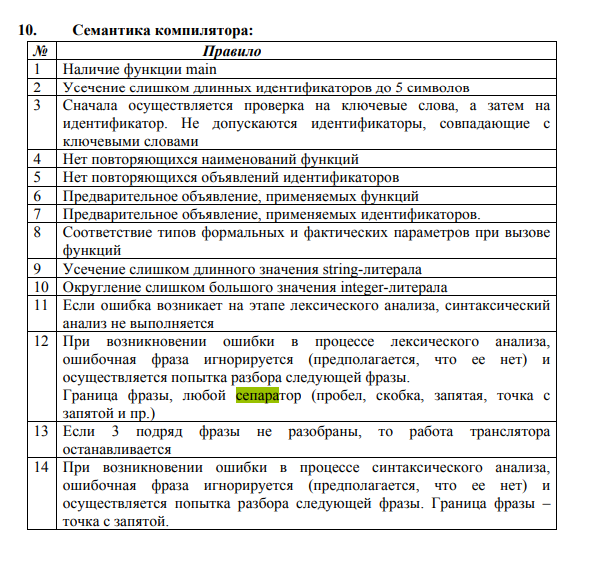
1.1

* строго типизируемый

Компилятор должен выполнять проверки типов, т.е. определять удовлетворяют ли типы набору правил языка программирования. Реализация языка является строго типизированной, если компилятор гарантирует, что полученная в результате программа не будет иметь ошибок, связанных с типами данных.(125)

* сепаратор





1.4

Какие есть кодировки ? тут Windows-125

1.7

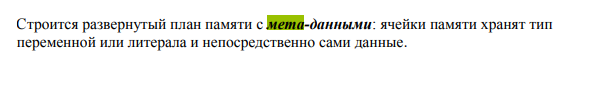
[] – один символ

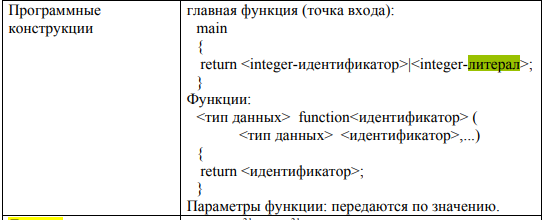
\* - повторение (0 или более раз)

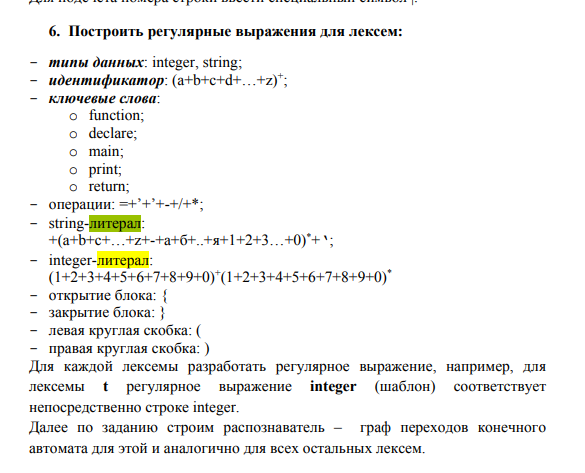
+ - повторение (1 или более раз)

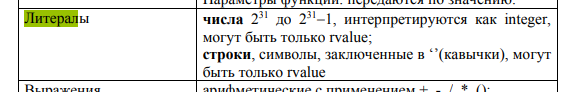
1.8

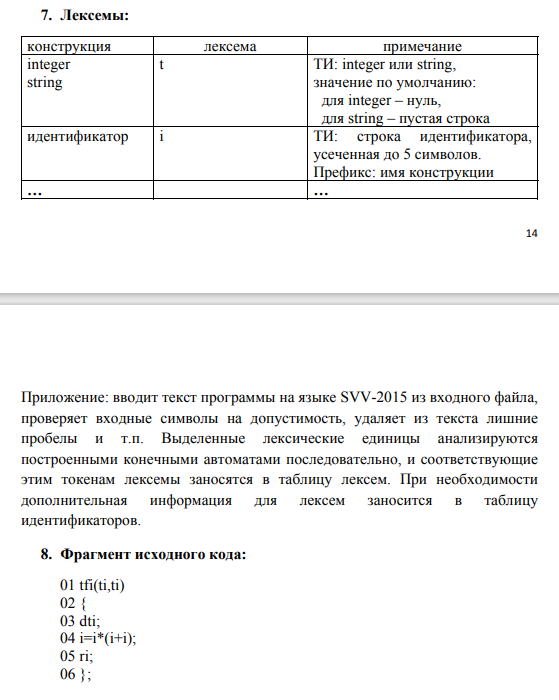
Литералы











1.9

## Область видимости **идентификаторов**

Когда мы объявляем переменную явно, мы можем указать дополнительную информацию об этой переменной. Например, тип переменной, что у переменной должна быть некоторая область видимости – т.е., что к ней нельзя обратиться из другого модуля или программного блока. 10

Область видимости имен

Классическим языком с точки зрения определения области видимости является Си. В Си область видимости задают не только функции, но и другие конструкции языка.

1.11

Инструкция языка программирования

1.16

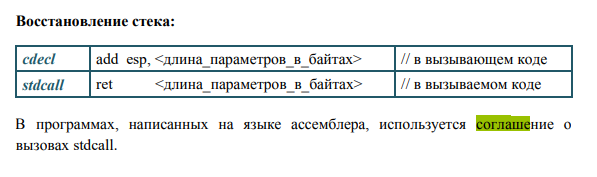
семантическая проверка кода(ЛЕКЦИЯ 23 --- 102 стр )

1.21

Директива

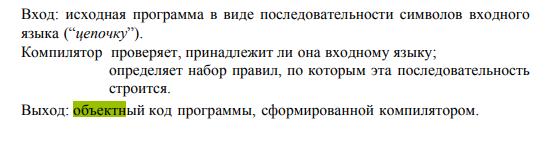
1.22

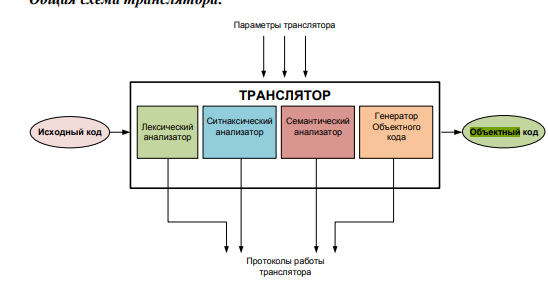
Соглашения о вызовах

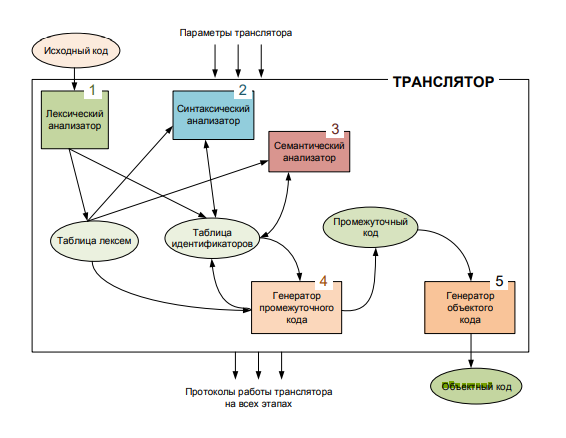


1.23

Объектный код







2.0

Транслятор

2.1

Исходный код

Лексический анализатор

Синтаксический анализатор

Семантический анализатор

Генерция кода

Протоколы работы транслятора

Ассемблированный код

3.3

символы табуляции,

3.8

е файл протокола. – это где что

4.2

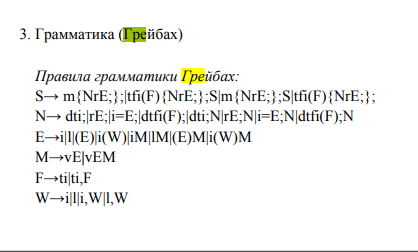
Нетерминал

Терминал

4.3.

Магазинный автомат

4.4. структура грмтики грейбах



В руле.хуй

Если видишь такую хуету:

fi(a, b, c);

То ассемблер еблан, не поймет что это

Тогда надо преобразовать в abc@3, чтобы он сначала видел параметры, а уже потом вызывал функцию. Тройка показывает сколько параметров в функции, то есть от символа собаки сутулой он отступит 3 параметра и занесет их в стек

\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\

**Пособие для ИрОчки**

1. Сначала наш входной текст фильтруется и разбивается на отдельные слова (файл div.cpp)
2. Затем лексический анализатор проверяет каждое слово и создаёт его лексему, на выходе получаются таблицы лексем и идентификаторов.
3. Далее синтаксический анализатор проверяет таблицу лексем на правильность написания (проверяет последовательность лексем)
4. Семантический анализатор еще раз пробегается по таблице лексем, но он обращается по этим лексемам к таблице идентификаторов (создан просто чтобы выявить ошибки)
5. В конце generation.cpp пробегается по таблице лексем и на их основе создаёт файл с кодом на ассемблере.

В файле лт.х изменить лет и тд с листика

В пояснительную сдвиги написать

procedure count(byte n)

{

create byte i = 1;

iterate (i < n)

{

output i;

output ' ';

i = i + 1;

}

outputln i;

}

byte function incr(byte n)

{

create byte res = n + 1;

back res;

}

main

{

outputln 'Считаем до 10: ';

count(10);

create byte t = 45 % 2;

outputln ' ';

output '45 четное? : ';

where (t ~ 0)

{

outputln 'правда';

}

else

{

outputln 'ложь';

}

outputln ' ';

create byte i = 5;

i = incr(i);

output 'incr(5) = ';

outputln i;

outputln ' ';

create string str = '29';

output 'Строка 29 в число + 2: ';

create byte num = strtobyte(str) + 2;

outputln num;

create byte len = slen(str);

output 'Длина строки: ';

outputln len;

create byte octo = o126;

output 'o126 = ';

outputln octo;

create byte n = 5;

create byte res = (1 + n << 2) \* 3 >> 2 / 2;

outputln res;

}