



Технологии программирования

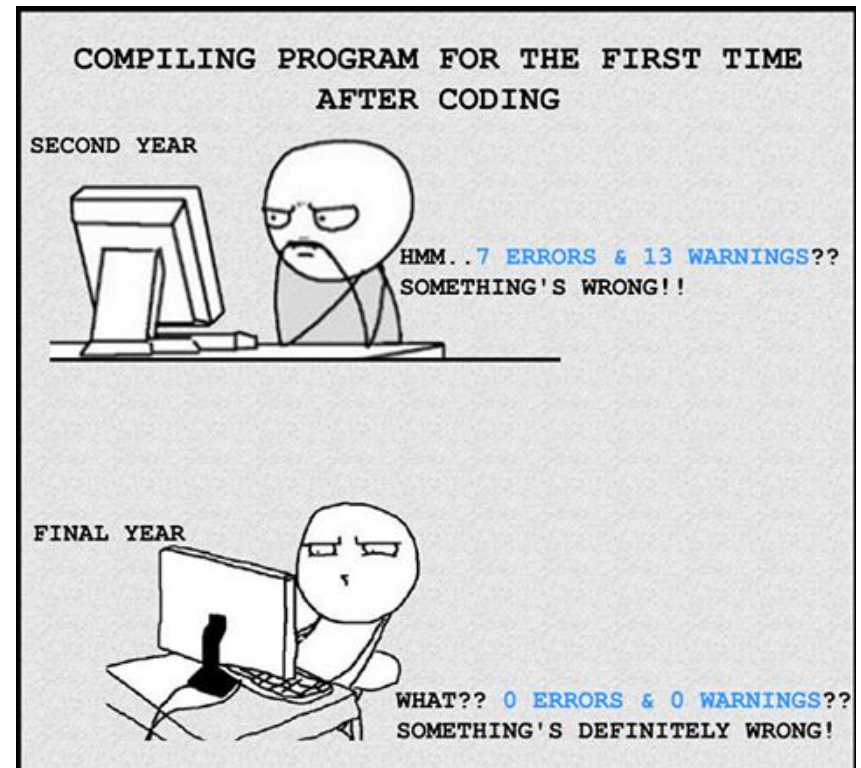
Лекция 1

Компиляторы

- *Компиляция* - трансляция программы, составленной на исходном языке высокого уровня трансляция программы, составленной на исходном языке высокого уровня, в эквивалентную программу на низкоуровневом языке трансляция программы, составленной на исходном языке высокого уровня, в эквивалентную программу на низкоуровневом языке, близком машинному коду трансляция программы, составленной на исходном языке высокого уровня, в эквивалентную программу на низкоуровневом языке, близком машинному коду (абсолютный код) трансляция программы, составленной на исходном языке высокого уровня, в эквивалентную программу на низкоуровневом языке, близком машинному коду (абсолютный

Но как это работает?

- Лексический анализ
- Синтаксический анализ
- Семантический анализ
- Оптимизация
- Генерация кода





Лексический анализ

- Оно же «токенизация»
- Разбор последовательности символов на распознанные сущности – лексемы, с последующим анализом и выдачей токенов

Выделение лексем

```
while (var < 5)
```

```
{
```

```
  □ var += 1;
```

```
}
```

Определение токенов

- Одному токену может соответствовать целое множество лексем – зависит от синтаксиса языка программирования

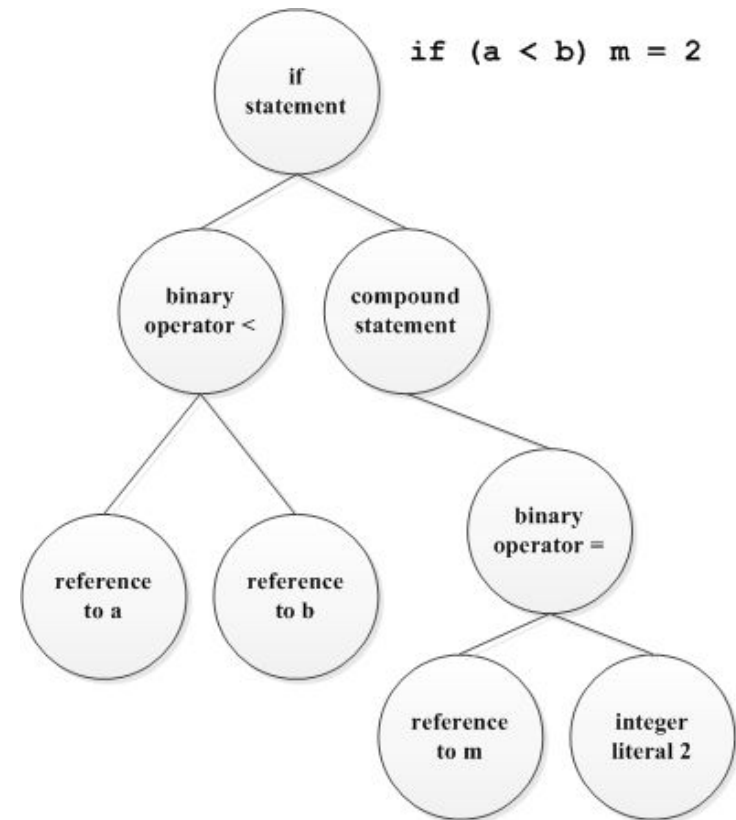
Token	Example lexeme
const	const
if	if
relop	<, <=
id	pi, count, age
num	3.14, 0
literal	"hello world"

Синтаксический анализ

- Генерация дерева синтаксического разбора
- *В следующих сериях... (курсе на 3)*
- *(сопоставление последовательности токенов формальной грамматике)*

Пример синтаксического разбора

- «Внутренние» вершины
 - операторы
- «Листья» - операнды
- Обычно граф.представление такое:
 - «снизу вверх»
 - сначала вычисляется левый «ребенок»
 - потом просматривается «родитель»
 - при необходимости вычисляется «правый ребенок»
 - вычисляется родитель «родитель»



Семантический анализ

- Проверка корректности
- Статическая проверка типов
- Вывод типов (выражения наподобие *auto*)
- Раскрытие «синтаксического сахара»
- и проч. проч. проч.

Оптимизация

- Перестроение дерева для генерации более эффективного машинного кода

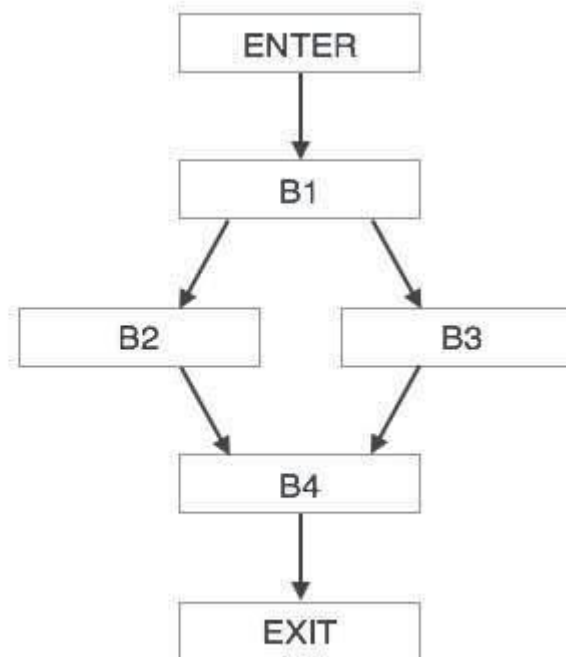
B1
w = 0;
x = x + y;
y = 0;
if(x > z)

B2
y = x;
x++;

B3
y = z;
z++;

B4
w = x + z;

Basic Blocks



Flow Graph

Генерация кода

- Генерируется машинный код
- На выходе – объектный файл
- Машинный код – уже машиннозависимый, т.е. для каждой архитектуры/поколения процессоров/моделей может быть разным



Оптимизация машинного кода

Продолжение следует...

*Но об этом через несколько лекций, в
теме “LLVM”...*