Лабораторная работа №9
«Форматтер исходных текстов»

Скоробогатов С. Ю., Коновалов А. В.

10 мая 2016

1 Цель работы

Целью данной работы является приобретение навыков использования генератора синтаксических анализаторов bison.

2 Исходные данные

Форматтер исходных текстов — это инструментальное средство, выполняющее вставку пробельных символов в исходные тескты программ, записанных на некотором языке, для улучшения удобочитаемости этих исходных текстов.

Форматтеры бывают *сильными* и *слабыми*. Слабый форматтер, в отличие от сильного, не может вставлять в исходный текст программы переводы строки.

Например, рассмотрим неотформатированную программу на языке Pascal:

```
var i:integer;
a,b,c:string
begin
a:='a';b:='b';
for i:=1 to 10 do begin
Println(a);
c:=a;a:=b;b:=c+b
end
end.
```

Слабый форматтер расставит в программе отступы и пробелы между лексемами:

```
var i: integer;
  a, b, c: string
begin
  a := 'a'; b := 'b';
  for i := 1 to 10 do begin
    Println(a);
    c := a; a := b; b := c+b
  end
end.
```

Сильный форматтер добавит переводы строки после ключевых слов **var** и **begin** и, кроме того, разобьёт строки, содержащие сразу несколько операторов:

```
var
  i: integer;
  a, b, c: string
begin
  a := 'a';
  b := 'b';
  for i := 1 to 10 do
  begin
    Println(a);
    c := a;
    a := b;
    b := c+b
  end
end.
```

3 Задание

В данной лабораторной работе требуется разработать слабый или сильный форматтер исходных текстов для одного из следующих языков:

1. **var**-блок языка Pascal (включая безымянные записи);
2. составной оператор языка Java (без вложенных анонимных классов, объявлений переменных обобщённых типов, **break**'ов на метку и оператора **switch**);
3. XML;
4. определение функции в языке Scheme;
5. определение процедуры в языке Visual Basic;
6. РЕФАЛ-5;
7. составной оператор языка Go (с опциональными точками с запятой);
8. составной оператор языка Pascal;
9. объявление класса в языке C++98 (со вложенными классами, тела методов — пустые);
10. РБНФ;

Выполнение лабораторной работы состоит из следующих этапов:

1. Создание лексического анализатора с помощью flex.
2. Разработка синтаксического анализатора с помощью bison и соединение его с лексическим анализатором.

3. Вставка в синтаксический анализатор семантических действий, выполняющих порождение отформатированного исходного текста программы.

4. Тестирование работоспособности получившегося форматтера на наборе примеров.

Форматтер должен работать до обнаружения первой синтаксической ошибки в тексте программы.

Слабый форматтер должен сохранять комментарии в тексте программы.

Сильный форматтер должен принимать **параметр командной строки**, ограничивающий длину строчек отформатированной программы. Например, если указано значение 80, то длины строчек на выходе не должны превышать в длину 80 символов. Единственной причиной выхода за пределы ограничения может быть только наличие токена, длина которого плюс текущий отступ в сумме превышает указанное ограничение.

Форматирование должно осуществляться в соответствии с традициями соответствующего языка программирования. Если таких традиций несколько (как, например, для языков с Си-подобным синтаксисом), то можно выбрать любую.