

Лабораторная работа №1

«Раскрутка самоприменимого компилятора»

Скоробогатов С.Ю.

20 августа 2013

1 Цель работы

Целью данной работы является ознакомление с раскруткой самоприменимых компиляторов на примере модельного компилятора.

2 Исходные данные

В качестве модельного выберем компилятор Р5 языка Pascal, разработанный С. Муром [1]. Входным языком компилятора является язык Pascal, соответствующий стандарту ISO 7185, а целевым языком – псевдокод, который может быть исполнен специальным интерпретатором.

Исходный текст компилятора Р5 составлен на языке Pascal и удовлетворяет стандарту ISO 7185. Тем самым, компилятор является самоприменимым.

Исходные данные для выполнения лабораторной работы в операционной системе Linux представлены следующим набором файлов:

pcom.pas – исходный текст компилятора Р5;

pcom – бинарная версия компилятора Р5, полученная путём компиляции исходного текста компилятора с помощью gpc (GNU Pascal Compiler);

pint – интерпретатор псевдокода, предназначенный для выполнения программ;

iso7185.pdf – текст стандарта ISO 7185:1990 [2];

hello.pas – программа, предназначенная для проверки работоспособности компилятора.

3 Использование pcom и pint

Бинарная версия компилятора Р5 берёт исходный текст компилируемой программы из стандартного потока ввода и записывает порождаемый псевдокод в файл с именем prt. Тем самым, для компиляции программы hello.pas нужно выполнить команду

```
./pcom <hello.pas
```

Интерпретатор `pint` считывает псевдокод из файла с именем `prd`, поэтому перед его запуском требуется переименовать файл `prg` в `prd`:

```
mv prr prd
./pint
```

Бинарную версию компилятора P5 можно применить к его исходному тексту:

```
./pcom <pcom.pas
```

После этого для компиляции программы `hello.pas` можно использовать компилятор P5, представленный в псевдокоде. Для этого нужно запустить компилятор с помощью `pint`:

```
mv prr prd
./pint <hello.pas
```

Более того, можно ещё раз откомпилировать `pcom.pas` с помощью компилятора P5, представленного в псевдокоде. Для этого нам потребуется выполнить команду

```
./pint <pcom.pas
```

Отметим, что последняя команда работает достаточно длительное время. После её выполнения можно убедиться, что файлы `prd` и `prg` совпадают с точностью до пробельных символов.

4 Задание

Выполнение лабораторной работы заключается в осуществлении одного шага раскрутки самоприменимого компилятора P5 и состоит из нескольких этапов:

1. добавление во входной язык компилятора P5 новых возможностей (см. таблицу 1) путём редактирования его исходного текста, в результате чего должен получиться файл `pcom2.pas`;
2. компиляция `pcom2.pas`, которая может осуществляться как бинарной версией компилятора, так и версией, представленной в псевдокоде (бинарная – быстрее);
3. проверка работоспособности `pcom2.pas` на небольшой программе, в которой обязательно должны использоваться новые возможности языка;
4. внесение изменений в `pcom2.pas`, связанных с использованием новых возможностей языка, и сохранение новой версии исходного текста компилятора в файле `pcom3.pas`;
5. завершение шага раскрутки путём компиляции `pcom3.pas` с помощью полученного на этапе 2 псевдокода компилятора.

Список литературы

- [1] Scott A. Moore. *The P5 compiler*. – URL: <http://www.moorecad.com/standardpascal/p5.html>.
- [2] *ISO 7185:1990: Information technology – Programming languages – Pascal*. – Geneva, Switzerland: International Organization for Standardization, 1990.

Таблица 1: Варианты изменений входного языка компиляторов P5 и BeRo Tiny Pascal

1	Обеспечить возможность использования символа подчёркивания в идентификаторах.
2	P5: не разрешать комментариям, начинающимся с «(*», заканчиваться на «}», а комметариям, начинающимся с «{», заканчиваться на «*)». BeRo Tiny Pascal: Разрешать комментариям, начинающимся с «(*», заканчиваться на «}», а комметариям, начинающимся с «{», заканчиваться на «*)».
3	P5: Сделать так, чтобы можно было использовать идентификаторы любой длины, но при этом символы идентификатора, начиная с одиннадцатого, не учитывались. BeRo Tiny Pascal: Выводить сообщение об ошибке при превышении длины идентификатора 40 символов.
4	Добавить однострочный комментарий, начинающийся с символа «?». Т.е. суффикс строки программы, расположенный после символа «?», должен считаться комментарием.
5	Сделать так, чтобы целочисленные константы, выходящие за границы допустимого интервала, считались равными нулю.
6	Заменить запись оператора присваивания на «:=».
7	Заменить запись операции «<>» на «!=».
8	P5: Сделать так, чтобы символы в строке программы, расположенные справа от 80-й позиции, не учитывались. BeRo Tiny Pascal: Сделать так, чтобы символы в строке программы, расположенные справа от 120-й позиции, не учитывались.
9	Разрешить использовать знак «..» вместо ключевого слова «to» при записи цикла for. При этом использование слова «to» не запрещается.
10	Добавить в строковые литералы Escape-последовательности «\a», «\b», «\t», «\\»
11	Добавить в язык шестнадцатиричные константы вида 0x12ABcd.
12	Добавить синонимы ~, & и для операторов not, and и or соответственно. При этом операторы not, and и or остаются допустимыми.
13	Сделать идентификаторы чувствительными к регистру.