

Лабораторная работа №1

«Раскрутка самоприменимого компилятора»

Коновалов А. В.

29 июля 2016

1 Цель работы

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

2 Исходные данные

В качестве модельного выберем компилятор BeRo Tiny Pascal, разработанный Бенжамином Рузо (Benjamin Rosseaux) [1]. Входным языком компилятора является язык Pascal, совместимый с диалектами Delphi 7 и FreePascal ≥ 3.0 , а целевым языком — исполнимый код Win32.

Исходный текст компилятора составлен на языке Pascal, совместимом с подмножеством диалектов Delphi 7 и FreePascal ≥ 3.0 , при этом сам реализован на этом подмножестве. Тем самым, компилятор является самоприменимым.

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

btpc.pas — исходный текст компилятора BeRo Tiny Pascal;

btpc.exe — бинарная версия компилятора, полученная путём раскрутки;

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

3 Использование компилятора

Компилятор берёт исходный текст со стандартного ввода и в случае успешной компиляции записывает порождённый двоичный код в стандартный вывод. Тем самым, для компиляции программы `hello.pas` нужно выполнить команду:

```
btpc <hello.pas >hello.exe
```

При наличии синтаксической ошибки в коде компилятор в стандартный вывод записывает вместо двоичного кода сообщение об ошибке. Признаком того, что компиляция прошла неудачно является малый размер целевого файла (в данном примере `hello.exe`) — менее 100 байт. Для того, чтобы посмотреть размер файла, можно выполнить команду `dir`. Для просмотра сообщения об ошибке нужно выполнить команду:

```
type hello.exe
```

Для выполнения одного шага раскрутки используется команда

```
btpc <btpc.pas >btpc_new.exe
```

После её выполнения можно убедиться, что файлы `btpc.exe` и `btpc_new.exe` идентичны при помощи команды

```
fc /b btpc.exe btpc_new.exe
```

4 Задание

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

1. добавление во входной язык компилятора **btpc** новых возможностей (см. таблицу 1) путём редактирования его исходного текста, в результате чего должен получиться файл **btpc2.pas** (следует сначала скопировать **btpc.pas** в **btpc2.pas** (на Linux: **btpc64.pas** в **btpc64-2.pas**), а потом вносить в него правки);
2. компиляция **btpc2.pas**, в результате которой должен получиться файл **btpc2.exe**;
3. проверка работоспособности **btpc2.exe** на небольшой программе, в которой обязательно должны использоваться новые возможности языка;
4. внесение изменений в **btpc2.pas**, связанных с использованием новых возможностей языка, и сохранение новой версии исходного текста компилятора в файле **btpc3.pas**;
5. завершение шага раскрутки путём компиляции **btpc3.pas** с помощью полученного на этапе 2 файла **btpc2.exe**;
6. разница между файлами **btpc.pas** и **btpc2.pas** (отображаемая командой `fc btpc.pas btpc2.pas`, на Linux: `diff btpc64.pas btpc64-2.pas`) должна демонстрировать изменения, внесённые в логику работы компилятора;
7. разница между файлами **btpc2.pas** и **btpc3.pas** (отображаемая командой `fc btpc2.pas btpc3.pas`, на Linux: `diff btpc64-2.pas btpc64-3.pas`) должна демонстрировать новые возможности языка.

Список литературы

- [1] <https://github.com/BeRo1985/berotinyascal>.

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

1	Обеспечить возможность использования символа @ 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 35 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 110-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 и ключевые слова чувствительными к регистру.
14	Заменить операторы div и mod на // и % соответственно.
15	Добавить в язык двоичные константы вида 0b10010.
16	Обеспечить возможность использования в числовых литералах незначимый знак _.
17	Заменить ключевые слова begin и end на {{ и }} соответственно. При этом ключевые слова begin и end остаются допустимыми.
18	Разрешить возможность не писать ключевое слово then после условия в блоке if.
19	Разрешить использовать ключевое слово to вместо знака .. при записи типа диапазона. При этом использование знака .. не запрещается.
20	P5: Сделать так, чтобы символы .. и : были взаимозаменяемыми. BeRo Tiny Pascal: Сделать так, чтобы символы .. и : перестали быть взаимозаменяемыми.