Лабораторная работа № 1.1. Раскрутка самоприменимого компилятора

Коновалов А.В., Скоробогатов С.Ю.

29 июня 2016

1 Цель работы

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

2 Исходные данные

Вариант P5

В качестве модельного выберем компилятор P5 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 P5 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 P5;
- `pcom` — исполнимая версия компилятора P5, 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.

Вариант BeRo/BeRo64

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

На кафедре ИУ9 данный компилятор был портирован на платформы Linux x64 и macOS.

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` — исполнимая версия компилятора, полученная путём раскрутки;
- `hello.pas` — программа, предназначенная для проверки работоспособности компилятора.

Исходные данные для выполнения лабораторной работы в операционных системах Linux и macOS представлены следующим набором файлов:

- `btpc64.pas`/`btpc64macOS.pas` — исходный текст портированного компилятора BeRo Tiny Pascal;
- `btpc64`/`btpc64macOS` — исполнимая версия компилятора, полученная путём раскрутки;
- `hello.pas` — программа, предназначенная для проверки работоспособности компилятора.

3 Использование компилятора

Использование `pcom` и `pint`

Исполнимая версия компилятора P5 берёт исходный текст компилируемой программы из стандартного потока ввода и записывает порождаемый псевдокод в файл с именем `prr`. 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 `prr` в `prd`:

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

Исполнимую 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[3]. После её выполнения можно убедиться, что файлы `prd` и `prr` совпадают с точностью до пробельных символов. Сравнить их на идентичность можно при помощи комадны

```
diff -uw prd prr
```

Использование Btpc на Windows

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 записывает в стандартный вывод 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
```

Использование Btpc64 на Linux и macOS

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:

```
./btpc64 <hello.pas >hello
```

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

```
cat hello
```

Для выполнения одного шага раскрутки используется команда

```
./btpc64 <btpc64.pas >btpc64_new
```

После её выполнения можно убедиться, что файлы `btpc64` и `btpc64_new` идентичны при помощи команды

```
cmp btpc.exe btpc_new.exe
```

4 Задание

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

Вариант P5

1. добавление во входной язык компилятора P5 новых возможностей (см. Варианты заданий) путём редактирования его исходного текста, в результате чего должен получиться файл `pcom2.pas` — следует **сначала** скопировать `pcom.pas` в `pcom2.pas`, **а потом** вносить в него правки;
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` (**сначала** нужно скопировать `pcom2.pas` в `pcom3.pas`, **затем** вносить изменения);
5. завершение шага раскрутки путём компиляции `pcom3.pas` с помощью полученного на этапе 2 псевдокода компилятора;
6. разница между файлами `pcom.pas` и `pcom2.pas` должна демонстрировать изменения, внесённые в логику работы компилятора;
7. разница между файлами `pcom2.pas` и `pcom3.pas` должна демонстрировать новые возможности языка.

Вариант с Btpc/Btpc64

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

Просмотр различий между файлами

На операционной системе Windows для просмотра различий между файлами рекомендуется использовать встроенную утилиту `fc`:

```
fc pcom.pas pcom2.pas
fc btpc.pas btpc2.pas
```

На операционных системах Linux и macOS для просмотра различий между файлами рекомендуется использовать встроенную утилиту `diff` с ключом `-u`:

```
diff -u pcom.pas pcom2.pash
diff -u btpc64.pas btpc64-2.pas
```

5 Индивидуальный вариант

Компилятор **BeRo**. Заменить операторы `div` и `mod` на `//` и `%` соответственно (однострочные комментарии перестанут поддерживаться).

6 Отчёт по лабораторной работе № 1.1

Отчёт выполняется в разметке Markdown по следующему шаблону:

```
% Лабораторная работа № 1.1. Раскрутка самоприменимого компилятора
% 3 февраля 2023 г.
% Вася Пупкин, ИУ9-63Б
# Цель работы
‹переписываете цель работы из задания›
# Индивидуальный вариант
‹переписываете индивидуальный вариант›
# Реализация
Различие между файлами `pcom.pas` и `pcom2.pas`:
```
  diff
xxxxx
```
Различие между файлами `pcom2.pas` и `pcom3.pas`:
```
  diff
xxxxx
```
# Тестирование
Тестовый пример:
```
  pascal
xxxx
```
Вывод тестового примера на `stdout`
```
xxxx
```
# Вывод
‹пишете, чему научились›
```

В отчёте приведён лишь необходимый минимум. Можно писать больше и интереснее — интересные и вдумчивые отчёты поощряются дополнительным баллом.

Шаблон отчёта

Ваш отчёт будет конвертирован в PDF при помощи pandoc следующей командой:

```
pandoc \
  --pdf-engine=xelatex \
  -V 'mainfont:liberation Serif' \
  -V 'monofont:liberation Mono' \
  "$SOURCE" -o "$PDF"
```

---

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 ↩

3. На кафедре ИУ9 разработана более быстрая реализация интерпретатора псевдокода P5: https://github.com/bmstu-iu9/P5-interpreter ↩

Язык реализации: Markdown

Код решения

Из файла

Отправить