Лабораторная работа № 1.1. Раскрутка самоприменимого компилятора

12 февраля 2024 г.

Сергей Виленский, ИУ9-62Б

Цель работы

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

Индивидуальный вариант

Компилятор **P5**. Сделать идентификаторы и ключевые слова чувствительными к регистру.

Реализация

Различие между файлами `pcom.pas` и `pcom2.pas`:

```
--- pcom.pas    2020-02-15 14:28:42.000000000 +0300
+++ pcom2.pas   2024-02-12 20:59:59.236616008 +0300
@@ -811,7 +811,6 @@
   { find lower case of character }
   function lcase(c: char): char;
   begin
-    if (c >= 'A') and (c<= 'Z') then c := chr(ord(c)-ord('A')+ord('a'));
     lcase := c
   end { lcase };
```

Различия между файлами `pcom2.pas` и `pcom3.pas`:

```
--- pcom2.pas   2024-02-12 20:59:59.236616008 +0300
+++ pcom3.pas   2024-02-12 20:36:56.311767635 +0300
@@ -1377,7 +1377,7 @@
                     until chartp[ch] <>  number
                   end;
                 if lcase(ch) = 'e' then
```

```
-                begin k := k+1; if k <= DIGMAX then digit[k] := ch;
+                begin k := k+1; if k <= digmax then digit[k] := ch;
                   nextch;
                   if (ch = '+') or (ch ='-') then
                     begin k := k+1; if k <= digmax then digit[k] := ch;
@@ -1480,7 +1480,7 @@
               until iscmte or (ch = ')') or eof(input);
               if not iscmte then nextch; goto 1
             end
-         else if ch = '.' then begin sy := LbRaCk; nextch end
+         else if ch = '.' then begin sy := lbrack; nextch end
           else sy := lparent;
           op := noop
        end;
```

Тестирование

Тестовый пример:

```
program hello(output);

var x, X: integer;

begin
    x := 1;
    X := 10;
    writeln(x, X);
end.
```

Вывод тестового примера на `stdout` при компиляции с помощью `pcom.pas`

```
P5 Pascal interpreter vs. 1.0

Assembling/loading program
Running program

P5 Pascal compiler vs. 1.0


     1       40 program hello(output);
     2       40
     3       40 var x, X: integer;
     3   ****          ^101
     4       48
     5       48 begin
     6        3     x := 1;
```

```
     7        7     X := 10;
     8        9     writeln(x, X);
     9       19 end.

Errors in program: 1

Error numbers in listing:
-------------------------
101  Identifier declared twice


program complete
P5 Pascal interpreter vs. 1.0

Assembling/loading program
Running program

         10         10

program complete
```

Вывод тестового примера на `stdout` при компиляции с помощью `pcom3.pas`

```
P5 Pascal interpreter vs. 1.0

Assembling/loading program
Running program

P5 Pascal compiler vs. 1.0


     1       40 program hello(output);
     2       40
     3       40 var x, X: integer;
     4       48
     5       48 begin
     6        3     x := 1;
     7        7     X := 10;
     8        9     writeln(x, X);
     9       19 end.

Errors in program: 0

program complete
P5 Pascal interpreter vs. 1.0

Assembling/loading program
```

```
Running program

          1         10

program complete
```

Вывод

В процессе выполнения данной лабораторной работы были преобретены навыки модификации кода компилятора **P5** дополнительным функционалом до новой самоприменяемой его версии и написания скриптов для удобной и чистой раскрутки компилятора, которые приведены ниже:

```
# comp2.sh
cp pcom prd
./pint < pcom2.pas
mv prr pcom2
rm prd
```

Файл `comp2.sh` предназначен для компиляции кода функционально модифицированного компилятора `pcom2.pas` при помощи псевдокода исходного компилятора `pcom` в псевдокод `pcom2`, который поддерживает компиляцию кода с новым функционалом, но код которого не может быть скомпилирован им же.

```
# comp3.sh
cp pcom2 prd
./pint < pcom3.pas
mv prr pcom3
rm prd
```

Файл `comp3.sh` предназначен для компиляции кода функционально и синтаксически модифицированного компилятора `pcom3.pas` при помощи псевдокода функционально модифицированного компилятора `pcom2` в псевдокод `pcom3` который поддерживает компиляцию кода с новым функционалом и является самоприменимым.

```
# run.sh
cp $1 prd
./pint < hello.pas
mv prr prd
./pint
rm pr{r,d}
```

Файл `run.sh` выполняет функцию компиляции файла `hello.pas` при помощи псевдокода компилятора из переданного первым аргументом файла, и интерпритации полученного псевдокода.

В конечном счете набор команд для раскрутки компилятора **P5**, компиляции и

интерпритации файла `hello.pas` на исходном и модифицированном компиляторе строится следующим образом:

```
# компилируем функционально модифицированный код компилятора
sh comp2.sh
# компилируем синтаксически модифицированный код компилятора
sh comp3.sh
# компилируем файл на исходном компиляторе и запускаем
sh run.sh pcom
# компилируем файл на модифицированном компиляторе и запускаем
sh run.sh pcom3
```

Как нетрудно заметить, описанный выше набор скриптов куда проще для восприятия человеком, что снижает риск допущения ошибки на этапе раскрутки компилятора, благодаря чему появляется возможность больше сосредоточиться на непосредственной модификации кода компилятора.

Ниже преведена T-диаграмма исполнения файла `hello.pas`, которая наглядно демострирует принцип раскрутки компилятора и запуска на нем программы основанной на дополненном функционале.

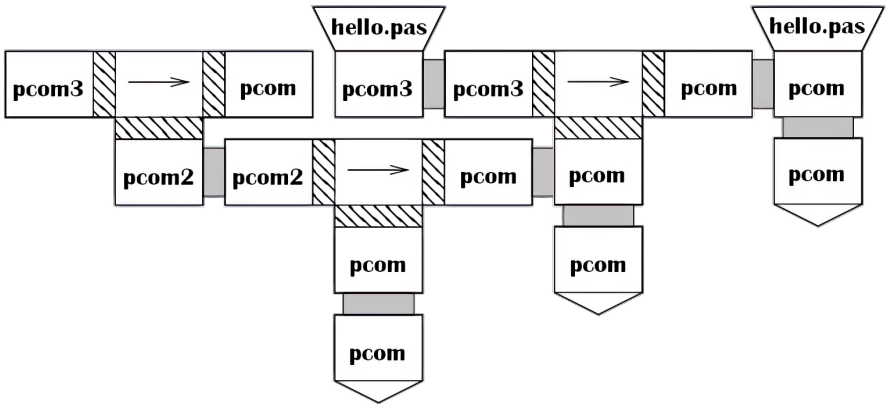

Figure 1: T-диаграмма исполнения `hello.pas`