

Краткое руководство по языку программирования Pascal

Р.В. Приходченко

May 29, 2013

Contents

1	Введение	2
1.1	Приблизительная последовательность действий при написании программы	2
1.2	Сообщения компилятора	4
2	Паскаль	5
2.1	Ключевые слова	5
2.2	Комментарии	6
2.3	Типы данных	6
2.4	Блок (Составной оператор)	7
2.5	Операторы управления выполнением программы	7
2.6	Циклы	9
2.7	Процедуры и функции	9
2.8	Множества	10
3	Отчёт по программе	10
4	литература	11

руководство распространяется в соответствии с условиями

Attribution-ShareAlike



(Атрибуция — С сохранением условий) CC BY-SA
Копирование и распространение приветствуется.

1 Введение

1.1 Приблизительная последовательность действий при написании программы

все действия этого параграфа происходят в терминале (консоль) кроме пункта 4; пункты нужно выполнять последовательно

1. создать каталог. Это необходимо сделать только один раз перед началом нового проекта (программы). Каждый проект хранится в отдельном каталоге. Каталог проекта будет содержать исходный код программы, исполняемые файлы, руководство пользователя и другие файлы необходимые для работы программы.

пример:

21119 - номер группы

petroff - фамилия

projabc3 - название проекта

```
mkdir -p 21119/petroff/projabc3
```

2. перейти в каталог проекта

```
cd 21119/petroff/projabc3
```

3. запустить любимый текстовый редактор (например emacs или gvim)
abc3.pas - название программы
& (амперсанд) - интерпретатор (bash) не дожидается завершения команды, выполнение программы (emacs) происходит в фоновом режиме (в терминале можно вводить команды не останавливая emacs)

```
emacs abc3.pas &
```

4. в текстовом редакторе (emacs) написать хорошую, правильную программу

5. компиляция программы

abc3.pas - название программы

```
fpc abc3.pas
```

Однако лучше использовать гламурную компиляцию; для этого нужно в терминале ввести команду (не забудьте написать команду в одну строчку, а также поменять типографские кавычки на одинарные кавычки)

```
function fpcc() { fpc "$1" 2>&1 | grep -Ei --color 'error|fatal|warning|note|'; }
```

и запускать

fpcc abc3.pas

или можно создать файл `~/bin/fpcc.sh` с таким содержимым:

```
#!/bin/sh
fpc $1 2>&1 | grep -Ei --color 'error|fatal|warning|note|'
```

и запускать `~/bin/fpcc.sh abc2.pas`

6. внимательно прочитать сообщения компилятора; при наличии ошибок или предупреждений перейти к пункту 4 (о сообщениях компилятора см. ниже)
7. запустить программу
abc3 - название исполняемого файла (без расширения «.pas»)

./abc3

8. если программа получилась негодной, перейти к пункту 4
9. если для демонстрации программы необходимо построить график то получаем текстовый файл с несколькими колонками разделёнными запятыми (без лишних сообщений), например так:

```
program abc5;
uses math;
const
  h: real = 1.0e-1;
var
  a,b,c : real;
begin
  a:=0.0;
  b:=5.0;
  c:=a;
  repeat
    writeln(c, ', ', sin(c));
    c := c + h;
  until (c>b);
end.
```

./abc3 > data.txt

для построения графика можно воспользоваться программой R или gnuplot (в них можно строить даже трёхмерные поверхности)

(a) R

запускаем в терминале R

```
gr <- read.table("data.txt", sep="," , head=FALSE)
plot(gr, type="l")
```

(b) gnuplot

запускаем в терминале gnuplot

```
plot "data.txt" with line
```

выход «Ctrl d»

1.2 Сообщения компилятора

Компилятор показывает сообщения об ошибках с номером строки и номером символа в круглых скобках.

Например (6,4) - ошибка в строке 6, номер символа 4.

Однако если отсутствует ; (точка с запятой) в конце оператора то компилятор укажет на следующую строку (пропущенную точку с запятой, скорее всего, нужно добавить строкой выше). Если вы воспользовались гламурной компиляцией (смотри выше) то ключевые слова будут подсвечены цветом.

Если в процессе компиляции появляются сообщения со словами «error» или «fatal», то в программе присутствует ошибка, которую необходимо исправить. Например ошибки синтаксиса и операции с различными типами:

```
abc3.pas(6,4) Fatal: Syntax error, "." expected but ";" found
abc3.pas(7,4) Error: Incompatible types: got "String" expected "Real"
abc3.pas(10) Fatal: There were 1 errors compiling module, stopping
Fatal: Compilation aborted
```

Если в процессе компиляции появляются сообщения со словами «warning» или «note», то в программе присутствует недостаток, который желательно исправить. Например неиспользуемая переменная и неинициализированная переменная (объявили переменную, в неё ничего не записали, попытались вывести её значение на экран):

```
abc3.pas(3,7) Note: Local variable "c" not used
abc3.pas(10,16) Warning: Variable "b" does not seem to be initialized
```

успешно откомпилированная программа должна содержать примерно такую строку:

```
10 lines compiled, 0.0 sec
```

2 Паскаль

примерный вид программы

```
program abc3;    // название программы (начинается с буквы)

uses math;      // подключение модулей
                // (в данном случае для математических функций)

const           // список констант
    MAX : integer = 100;

type
    mass : array [1..MAX] of integer;

var             // описание типов переменных
    a : integer;
    s : string;
    m : mass;

begin           // начало программы

    readln(s);    // программа
    m[2] := 7;    // программа
    m[MAX-8] := 3; // программа
    a := 5;       // программа
    writeln(s, a); // программа

end.            // конец программы (end с точкой)
```

2.1 Ключевые слова

ключевые слова не допустимо использовать для названия переменных, констант, процедур и функций. список ключевых слов:

absolute, and, array, asm, begin, boolean, break, case, char, const, continue, div, do, downto, else, end, for, function, goto, if, implementation, in, interrupt, is, label, mod, not, or, org, otherwise, print, procedure, program, read, real, record, repeat, shl, shr, step, string, then, to, type, unit, until, uses, var, while, with, xor

2.2 Комментарии

текст заключённый между фигурными скобками - комментарии к программе Пример:

```
{ Место для комментария  
  Комментарий может занимать несколько строк }
```

текст после двух слэшей также является комментарием

```
// Место для комментария  
// Комментарий может занимать только одну строку
```

2.3 Типы данных

1. real числа с плавающей запятой
 $\pm 1.17549435082 * 10^{-38} .. \pm 6.80564774407 * 10^{38}$
2. integer целые -32768 .. 32767
3. char символьный
4. boolean логический
5. перечисления

пример:

```
var          // объявления переменных  
  r: Real;    // переменная вещественного типа  
  i: Integer; // переменная целого типа  
  c: Char;    // переменная-символ  
  b: Boolean; // логическая переменная  
  s: String;  // переменная строки  
  t: Text;    // переменная для объявления текстового файла  
  
  e: (apple, pear, banana, orange, lemon);  
      // переменная типа-перечисления  
  x: 1..10;   // переменная типа-перечисления  
  y: 'a'..'z'; // переменная типа-перечисления  
  
  set1: set of 1..10; // множество  
  set2: set of 'a'..'z'; // множество
```

```

r = record // определение записи
    x: integer;
    y: char;
end;
f = Text; // определение файла

```

2.4 Блок (Составной оператор)

Блок используется если можно использовать только один оператор, а хочется несколько. Блок ограничивается ключевыми словами begin и end.

Например if (a>b) then оп1 else оп2;

вместо оп1 (или оп2) может быть только один оператор но часто нужно выполнить несколько.

```

if a > b then begin
    оп3;
    оп4;
    оп5;
end;
else
    оп2;

```

2.5 Операторы управления выполнением программы

```

if a > b then // условный оператор
    writeln('Условие выполнилось')
else // иначе - секция может отсутствовать
    writeln('Условие не выполнилось');

case i of // условный оператор множественного выбора
    0: write('ноль');
    1: write('один');
    2: write('два')
    else write('неизвестное число') // иначе - секция
                                     // может отсутствовать
end; // окончание case
// один из случаев когда нет begin но есть end

```

Для множественных условий предпочтительно использовать оператор

case (потому что компилятор в большинстве случаев создаст более оптимальный код).

операторы сравнения

< меньше
> больше
<= меньше или равно
>= больше или равно
= равно
<> не равно

логические операторы

or или
and и
not не

нежелательно делать так

```
if (b=5) then ...
```

лучше так (на 3 курсе вам скажут что паскаль не нужен и обучат C, а привычки останутся)

```
if (5=b) then ...
```

совсем неправильно делать так

```
program abc5;
var
  a,b : real;
begin
  a:=7.0;
  b:=1.0-((1.0/3.0)*(a-1.0)/2.0);
  if (b=0.0) then
    writeln('zero')
  else
    writeln ('no zero');
  writeln(b);
end.
```

более правильный вариант

```
program abc5;
const epsilon : real = 1.0e-10;
var
  a,b : real;
begin
  a:=7.0;
  b:=1.0-((1.0/3.0)*(a-1.0)/2.0);
  if (b < epsilon) then
    writeln('zero')
  else
    writeln ('no zero');
  writeln(b);
end.
```

в общем случае проверить равенство двух чисел (a,b) с плавающей запятой можно так, причём epsilon нужно выбирать исходя из числа разрядов, а также сложности и количества выполняемых действий (например r1 имеет погрешность $\pm 1\Omega$, r2 $\pm 2\Omega$, посчитайте погрешность сопротивления при соединении резисторов параллельно по двум формулам $r = \frac{1}{\frac{1}{r_1} + \frac{1}{r_2}}$ и $r = \frac{r_1 r_2}{r_1 + r_2}$)

```
if (abs(a-b) < epsilon) then ...
\\ или если хочется странного
if (abs(a-b) < epsilon * (abs(a)+abs(b))) then ...
```


2.6 Циклы

```
n:=5;

a:=1;
while (a < n+1) do begin // цикл с предусловием
    writeln('a=', a);
    a := a+1;
end;

for b := 1 to 5 do begin // итерационный цикл
    writeln('b=', b);
    // внутри цикла for нельзя менять счётчик (b)
end;
// пользоваться счётчиком (b) после цикла некорректно

c:=1;
repeat // цикл с постусловием
    writeln('c=', c);
    c := c + 1;
until (c > 5);
```

В результате работы на экран будут выведены числа 1,2,3,4,5 в столбик.

2.7 Процедуры и функции

Процедуры отличаются от функций тем, что функции возвращают какое-либо значение, а процедуры — нет.

```
program abc5;

var i : integer;

function next(k: integer): integer;
begin
    next := k + 1
end;

begin
```

```

    i := 1;
    writeln(next(i));
end.

```

2.8 Множества

```

var { секция объявления переменных }
    d : set of char;
begin { начало блока }
    d:=['a','b'];
    i:=7;
    if i in [5..10] then writeln('принадлежит множеству');

```

3 Отчёт по программе

1. Задание. Описать задание как вы его поняли
2. Словесно-формульный алгоритм. Описать как работает алгоритм и рассмотреть сложные моменты
3. Блок-схема. Агромадный рисунок с кружочками, стрелочками и многоугольниками
4. Программа. Можно оставить пункт пустым: продемонстрировать работающую программу
5. Руководство пользователя. Что нужно вводить и как получить результат
6. Проверка. Если в программе вычисляется $y:=\sqrt{1/x}$ нужно проверить как работает программа при $x=0.0$; $x=-9.0$ и обычных числах например $x=25.0$
7. Улучшения. Большинство программ можно улучшить; нужно описать эти изменения например в программе присутствует ввод целого числа, но пользователь может ввести «пять», « 5» (пробел 5 [вообще то так можно делать]), «=5», «50» (буква О очень похожа на цифру 0), «5,4» (вместо 5.4 если спрашивают число с плавающей запятой). Всё это можно исправить если создать функцию, например «readint», которая будет запрашивать ввод данных, предварительно обрабатывать их (например с помощью val), а в случае неправильного числа запрашивать ввод повторно

8. Лицензия. Указать название лицензии (в случае EULA привести полный текст лицензии)

- (a) BSD (делайте с программой что хотите: копируйте, изменяйте, распространяйте, продавайте)
- (b) GPL (делайте с программой что хотите: копируйте, изменяйте, распространяйте, продавайте. Но оставьте первоначального автора и лицензию GPL)
- (c) EULA (лицензионное соглашение с конечным пользователем) - договор между владельцем (автором) компьютерной программы и ~~работ~~ пользователем её копии. Студенту желающему сдать работу, и выбравшему в качестве лицензии EULA, требуется написать конечное соглашения пользователя в котором для примера, но не для бездумного копирования, используется в качестве основы, следующее описание, в котором описываются ограничения включающие, но не ограничивающиеся, запрещением просмотра исходного кода (только под NDA - соглашение о неразглашении), запрещением распространения, запрещением несанкционированного и несогласованного с высшим руководством запуска программы, запрещением продажи без покупки дистрибьюторских прав, банальные зонды и прочие соглашения почти не нарушающие конституцию и права человека, если будет доказано что пользователь действительно и неоспоримо на момент договора и в течении всего времени на которое распространяется действие договора время, являлся человеком, причём без возможности получения прямой либо косвенной выгоды в том числе либо материально либо нематериальной выгоды включая использование данного соглашения без изменения его сути и содержания, ограничиваясь только 10 (десятью) страницами мелкого, трудно читаемого текста.

4 литература

1. Е.Р. Алексеев, О.В. Чеснокова, Т.В. Кучер
Free Pascal и Lazarus: Учебник по программированию
Библиотека ALT Linux
2. **Quick Reference Guide for Pascal language**
mikroElektronika SOFTWARE AND HARDWARE SOLUTIONS
FOR THE EMBEDDED WORLD
3. **Паскаль (язык программирования)**
Материал из Википедии — свободной энциклопедии