Указания по оформлению текстов программ на языке Паскаль

21 ноября 2007 г.

1 Общие замечания по оформлению

1.1 Комментарии

- В начале текста программы должен помещаться комментарий с описанием постановки задачи, решаемой программой.
- Хорошим тоном в программировании считается вставка комментариев в текст программы, поясняющих неочевидные особенности её работы.

1.2 Расположение операторов на строке

На одной строке не должно располагаться более одного оператора.

Пример (расположение операторов на строках):
Следующий фрагмент программы:

```
T := A; A := B; B := T;
```

оформлен неправильно, поскольку на одной строке здесь расположено сразу три оператора. Правильным оформлением данного фрагмента будет следующее:

```
T := A;
A := B;
B := T;
```

*

Если оператор слишком длинный, для удобства чтения его рекомендуется разбивать на несколько строк. При этом продолжение записи оператора на последующих строках дополняется слева двумя пробелами.

Пример (расположение оператора на нескольких строках):

```
RA_Edge := Sqrt(
  Sqr(RB_Edge) + Sqr(RC_Edge) -
  2 * RB_Edge * RC_Edge * Cos(RA_Angle));
```

*

2 Операторы

2.1 Операторы, следующие друг за другом

Если последовательно записанные операторы реализуют конструкцию «следование» структурного программирования, то они должны быть записаны друг за другом на отдельных строках, причём слева от каждого оператора необходимо вставить одинаковое количество пробелов.

Пример (следование операторов):
Пусть задан следующий фрагмент программы:

```
T := A;
A := B;
B := T;
```

Данному фрагменту соответствует блок-схема, изображённая на рис. 1. Поскольку все три оператора следуют друг за другом, в тексте программы они

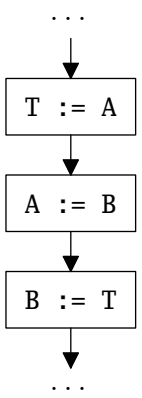

Рис. 1: Структура «следование»

располагаются один под другим, как в приведённом фрагменте. *

2.2 Составной оператор

Составной оператор, а также блок операторов программы «**begin** ... **end.**» оформляется следующим образом:

- Ключевые слова **begin** и **end** записываются на отдельных строках, больше в эти строки ничего не помещается.
- Все операторы между ключевыми словами **begin** и **end** дополняются слева двумя пробелами (т. е., выравниваются на 2 пробела правее слов **begin** и **end**).
- Если внутри составного оператора или блока операторов программы встречается другой составной оператор, то его операторные скобки **begin**

и **end** записываются на одном уровне по горизонтали вместе с другими операторами внешнего блока (т. е., также выравниваются 2 пробелами), а операторы внутри него выравниваются ещё на 2 пробела правее ключевых слов **begin** и **end** внутреннего составного оператора (см. примеры далее).

Пример (блок операторов программы):

```
{
  Вычисление площади прямоугольника с заданным
  длинами сторон
}
program Square;
var
  A, B, S: Real;
begin
  Write('Введите длины сторон прямоугольника: ');
  Read(A, B);
  S := A * B;
  WriteLn('Площадь прямоугольника равна ', S)
end.
```

*

2.3 Условный оператор

- На отдельной строке записывается ключевое слово **if**, за ним — условие и ключевое слово **then**.
- Оператор, являющийся веткой «**then**» записывается, начиная со следующей строки.
- Если оператор-ветка «**then**» не является составным оператором, то он дополняется слева двумя пробелами, таким образом, смещаясь по горизонтали на 2 пробела вправо относительно ключевого слова **if** строкой выше.

Пример (условный оператор без составного оператора):

```
  if X < 0 then
    X := 0;
```

*

- Если оператор-ветка «**then**» является составным оператором, то он записывается, начиная со следующей строки, не дополняясь слева пробелами (тем не менее, операторы внутри составного оператора должны иметь отступ относительно ключевых слов **begin** и **end** по правилам записи составного оператора. Таким образом, ключевые слова **begin** и **end** составного

оператора должны находиться на одном уровне с ключевым словом **if** условного оператора, веткой «**then**» которого он является.

Пример (условный оператор с составным оператором):

```
if A > B then
begin
  T := A;
  A := B;
  B := T
end;
```

*

- Если условный оператор имеет также ветку «**else**», то ключевое слово **else** записывается на отдельной строке после оператора-ветки «**then**». Сам оператор, являющийся веткой «**else**», записывается на следующей после **else** строки, причём правила его записи идентичны правилам записи оператора-ветки «**then**».

Примеры (условный оператор с веткой «иначе»):

1. Ветка «иначе» является простым оператором:

```
if A < B then
  Min := A
else
  Min := B;
```

2. Ветка «иначе» является составным оператором:

```
if D < 0 then
  WriteLn('Корней нет')
else
begin
  SD := Sqrt(D);
  A2 := 2 * A;
  X1 := (- B - SD) / A2;
  X2 := (- B + SD) / A2;
  WriteLn(
    'I корень: ', X1, ' ',
    'II корень: ', X2)
end;
```

*

- Если условие оператора **if** является слишком длинным, то, аналогично общему правилу записи операторов, данное условие рекомендуется разбить на несколько строк с добавлением отступа в 2 пробела на последующих строках.

Пример (условный оператор с длинным условием):

```
if (A > 0) and (B > 0) and (C > 0) and
  (A + B > C) and (A + C > B) and (B + C > A) then
  WriteLn(
    'Можно построить треугольник со сторонами ',
    A, ', ', B, ' и ', C)
else
  WriteLn(
    'Треугольник со сторонами ',
    A, ', ', B, ' и ', C, ' построить нельзя');
```

*

Замечание: Необходимо уделять особое внимание случаям, когда после условного оператора стоит другой оператор. △

Пример (оператор после условного оператора):

```
if X < 0 then
  X := - X;
S := S + X;
```

Блок-схема данного фрагмента приведена на рис. 2. Здесь условный опера-

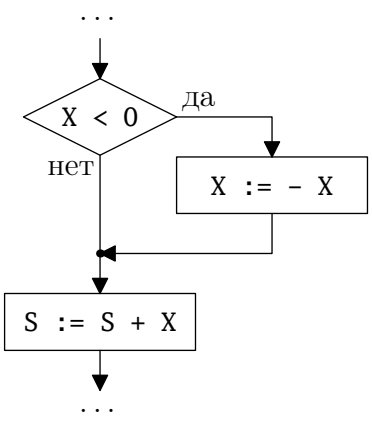

Рис. 2: Следование после развилки

тор реализует структурную конструкцию «развилка», в то время как условный оператор вместе с оператором присваивания «`S := S + X`» образуют конструкцию «следование» (если условный оператор представить в виде укрупнённого блока, данный фрагмент программы можно изобразить в виде блок-схемы на рис. 3). Именно поэтому данный фрагмент оформляется так, как показано в данном примере: оператор «`S := S + X`» находится под условным оператором (под ключевым словом **if**, см. правило оформления следующих друг за другом операторов в разделе 2.1). Отсюда следует, что данное оформление этого фрагмента:

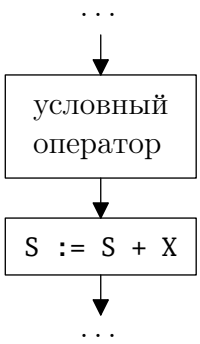

Рис. 3: Следование после развилки, укрупнённая блок-схема

```
if X < 0 then
  X := - X;
  S := S + X;
```

является неправильным, поскольку оператор «`S := S + X`» не относится к ветке «**then**» условного оператора, в отличие от следующего фрагмента:

```
if X < 0 then
begin
  X := - X;
  S := S + X
end;
```

блок-схема которого приведена на рис. 4.

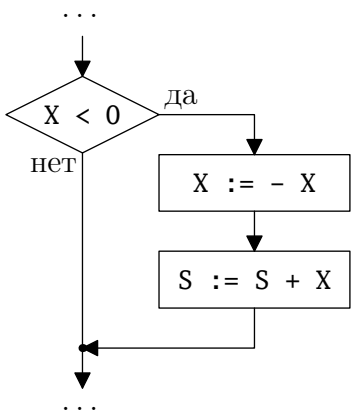

Рис. 4: Следование внутри развилки

*

2.4 Операторы циклов с предусловием и со счётчиком

Операторы циклов «**while**» и «**for**» оформляются по таким же правилам, как и оператор «**if**».

Пример (вложенные циклы — печать таблицы умножения):

```
  for I := 1 to 10 do
  begin
    for J := 1 to 10 do
      Write(I * J : 3);
    WriteLn
  end;
```

*

2.5 Оператор цикла с постусловием

- Оператор цикла с постусловием оформляется аналогично блоку «**begin** ... **end**»: ключевые слова **repeat** и **until** с условием располагаются на отдельных строках друг под другом, операторы внутри тела цикла располагаются с отступом на 2 пробела вправо относительно ключевых слов **repeat** и **until**.

- Аналогично условию в операторе «**if**», длинное условие цикла «**repeat** ... **until**» может быть продолжено на следующих строках с отступом в 2 пробела относительно ключевого слова **until**.

Пример (цикл с постусловием):

```
{
  Вычисление e в степени X при помощи ряда Тейлора
}
const
  Eps = 1e-7;
var
  X, A, S: Real;
  K: Integer;
begin
  Write('Введите значение аргумента: ');
  Read(X);
  S := 0;
  A := 1;
  K := 1;
  repeat
    S := S + A;
    A := A * X / K;
    K := K + 1
  until Abs(A) < Eps;
  WriteLn('e в степени ', X, ' равно ', S)
end.
```

*

2.6 Оператор выбора

- Заголовок оператора («**case** ... **of**») и завершающее ключевое слово **end** располагаются на отдельных строках друг под другом.
- Каждое перечисление констант выбора (с двоеточием на конце) или ключевое слово **else**, определяющие альтернативу выбора, располагаются на отдельной строке с отступом в 2 пробела вправо от заголовка оператора.
- Оператор, исполняемый по соответствующей ветви для данной альтернативы, располагается на следующей после неё строке. Если этот оператор — составной, то он записывается ровно под записью альтернативы. Иначе он записывается с отступом в 2 пробела вправо от записи альтернативы (таким образом, с выравниванием на 4 пробела вправо от заголовка оператора).
- Как и в случае с другими операторами, длинное выражение выбора (стоящее между ключевыми словами **case** и **of**) рекомендуется разбивать на несколько строк с отступом в 2 пробела на последующих строках.

Пример (оператор выбора):

```
case N of
  1, 3, 5:
    WriteLn('Один, три или пять.');
  2, 4:
    WriteLn('Два или четыре.');
  6 .. 10:
    WriteLn('От шести до десяти.');
  else
  begin
    WriteLn('Вне диапазона от одного до десяти.');
    BError := True
  end
end;
```

∗

Замечание: Ранее приведённое в разделе 2.3 замечание относительно операторов, следующих за условными оператором, в равной мере относится ко всем операторам циклов и выбора: необходимо уделять особое внимание оформлению данных фрагментов программ. △