ТУБ №21:
Алфавит и лексемы языка Delphi

Отправная точка в изучении любого языка программирования — его алфавит и лексическая структура, т.е. виды лексем, которые могут встретиться в тексте программы.

Алфавит языка Delphi

Как Вам уже должно быть известно, под алфавитом языка понимают множество символов, которые могут использоваться для составления программ на данном языке. В случае Delphi выделяют следующие группы символов:

- буквы;
- цифры;
- специальные символы.

К буквам относят заглавные и строчные буквы латинского алфавита, а также символ подчёркивания «_». Включение этого символа в состав букв позволяет в дальнейшем упростить описание языка.

Следует отметить, что как в Pascal, так и в Delphi заглавные и строчные буквы считаются эквивалентными. Например, следующие записи компилятор будет считать одинаковыми:

```
count     Count     COUNT     CoUnT     cOUNt
```

Во многих языках программирования эти записи будут считаться различными (про такие языки ещё говорят, что они регистрочувствительны). В частности, в них возможно использовать, например, `count` для названия переменной, а `COUNT` — для названия константы.

Это может показаться удобным, но даже к моменту появления языка Pascal (а это, напомним, 1970 г.) не было секретом, что ни к чему хорошему такой стиль именования не приводит, не говоря уже про наши дни. Кроме того, далеко не все компьютеры того времени умели работать со строчными буквами: многие и них всё ещё поддерживали только заглавные. Представьте себе, сколько проблем создала бы программа на регистрочувствительном языке, в которой есть строчные буквы, при попытке работы с ней на таком компьютере!

В итоге Pascal остался нечувствительным к регистру, и это свойство также оказалось полезным и для подготовки молодых специалистов: плохой стиль именования сделан невозможным уже на уровне языка. Закономерным решением было сохранить это решение и в потомке Pascal — языке Delphi.

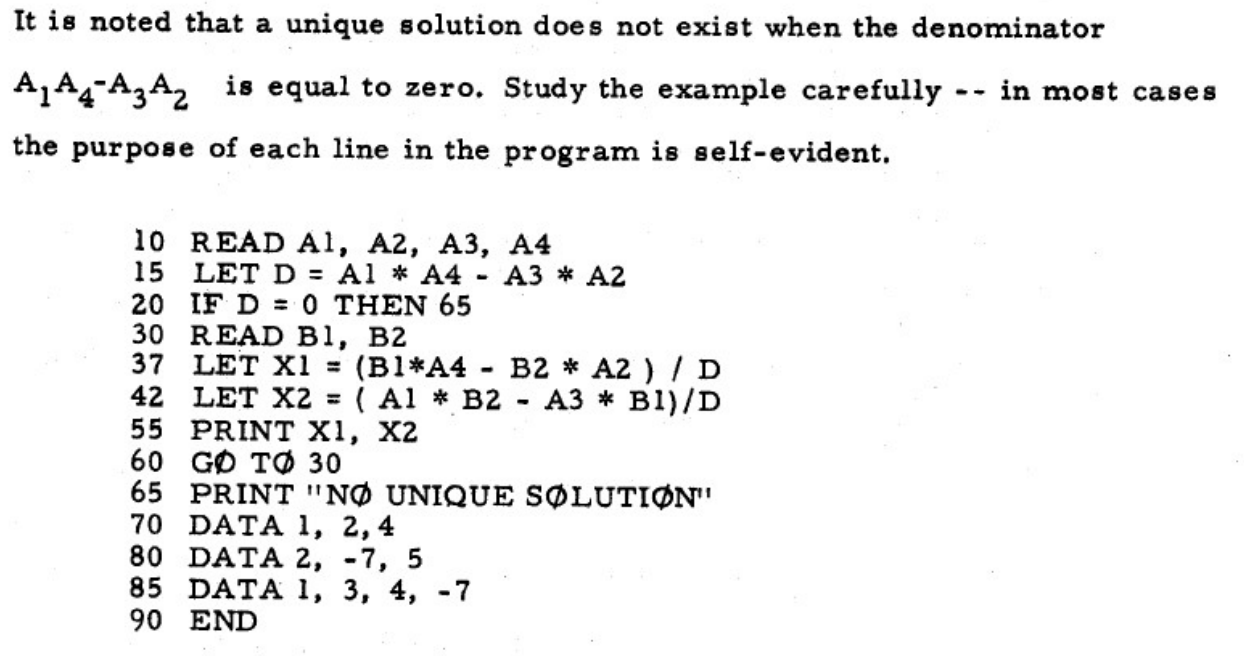
Исключением из этого правила (только подтверждающим правило) являются так называемые строковые литералы — вид констант, позволяющий задать в программе требуемый фрагмент текста. Соответственно, следующие записи задают различные значения (что, впрочем, не влияет на отношение к ним компилятора):

```
'count'   'Count'   'COUNT'   'CoUnT'   'cOUNt'
```

Следующая группа символов алфавита Delphi — цифры от 0 до 9. Казалось бы, что ещё можно добавить к этой информации? На самом деле можно.

Ещё в далёкие 50-е гг. XX века программисты заметили, что буква «O» и цифра «0» уж очень похожи друг на друга. Критически настроенный читатель скажет, что поздновато они спохватились, — и будет совершенно прав. Но именно оттуда берёт свои корни традиция при записи текста программы на бумаге перечёркивать нули. Некоторые современные шрифты сразу содержат перечёркнутый ноль: «Ø».

Кстати, не сразу это обозначение прижилось в таком виде. Например, в оригинальном документе 1964 года, описывающем язык Basic, всё в точности наоборот:



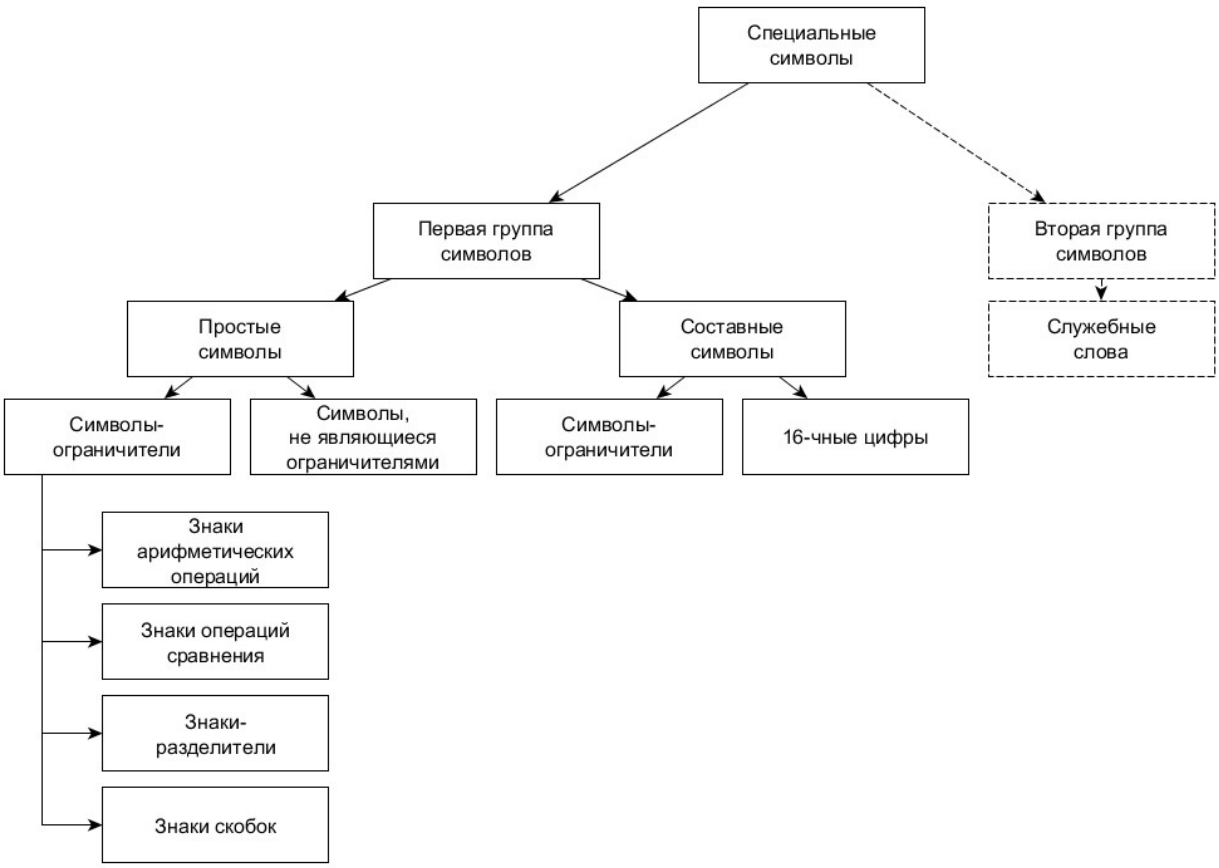
It is noted that a unique solution does not exist when the denominator $A_1A_4-A_3A_2$ is equal to zero. Study the example carefully -- in most cases the purpose of each line in the program is self-evident.

```
10  READ A1, A2, A3, A4
15  LET D = A1 * A4 - A3 * A2
20  IF D = 0 THEN 65
30  READ B1, B2
37  LET X1 = (B1*A4 - B2 * A2 ) / D
42  LET X2 = ( A1 * B2 - A3 * B1)/D
55  PRINT X1, X2
60  GØ TØ 30
65  PRINT "NØ UNIQUE SØLUTIØN"
70  DATA 1, 2,4
80  DATA 2, -7, 5
85  DATA 1, 3, 4, -7
90  END
```

Последняя группа символов алфавита языка Delphi — специальные символы. Они в свою очередь подразделяются на две группы, одна из которых включает в себя большую часть символов, а другая — только служебные слова.

Включение служебных слов в алфавит языка выполняют не всегда. Технически эти символы являются более сложной часть текста программы — лексемой — и обрабатываются компилятором несколько иначе. Тем не менее, некоторые источники приводят и такую классификацию, поэтому на рисунке вторая группа символов выделена пунктирными линиями.

Далее перечислим символы каждой группы с краткими пояснениями, какое применение они имеют в языке Delphi. Конкретные способы их применения имеет смысл рассматривать при изучении соответствующих тем.



Перечислим простые символы.

Символы-ограничители:
- Знаки арифметических операций:
  - + (сложение, объединение множеств, сцепление строк);
  - − (вычитание, разность множеств);
  - * (умножение, пересечение множеств);
  - / (деление);
- Знаки операций сравнения:
  - < (меньше);
  - > (больше);
  - = (равно);
- Знаки-разделители:
  - . (десятичная точка, разделитель полей, признак конца модуля);
  - , (разделитель в перечислениях);
  - : (разделитель объявляемого элемента и его определения, метки и оператора);
  - ; (разделитель операторов);
  - ' (ограничитель строковых литералов);
- Знаки скобок:
  - ( ) (выделение подвыражений);
  - [ ] (выделение индексов массивов);
  - { } (ограничители комментариев);

Символы, не являющиеся ограничителями:
- @ (операция взятия адреса);
- # (признак символьного литерала, задающего код символа);
- $ (признак 16-чного целочисленного литерала);
- ^ (обозначение указателя, операция разыменования указателя);
- ␣ (пробел, разделяет другие символы).

Имеется также ряд составных символов-ограничителей:

> <= (меньше или равно);
> >= (больше или равно);
> <> (не равно);
> := (присваивание);
> .. (обозначение диапазона)

По историческим причинам для многих символов алфавита есть также эквивалентные замены. Например:

> (* *) эквивалентны { }
> (. .) эквивалентны [ ]

Причина существования таких замен (они ещё называются диграфами) та же, что и обычно — историческая: ранние компьютеры часто не позволяли использовать символы квадратных и фигурных скобок, поэтому были предусмотрены замены. Позднее от них можно было бы отказаться, но имеющиеся программы на языке Pascal, в которых использовались диграфы, пришлось бы исправлять — а это лишние расходы на оплату работы программистов, которая не добавляла бы ничего полезного в саму программу. Разумеется, на это никто бы не пошёл и компании-разработчики предпочли бы не переходить на Delphi, если бы пришлось вносить такие исправления. Поэтому в Delphi поддержка диграфов сохранена.

К 16-чным цифрам относятся первые 6 букв латинского алфавита: A, B, C, D, E, F. Такой смысл эти буквы имеют только в составе так называемых 16-чных целочисленных литералов.

Последняя группа символов — *служебные слова*. Это сочетания букв, которые имеют специальное значение в языке программирования. Как правило, они используются для обозначения некоторых операций над данными, для разделения программы на составные части. Другие, абсолютно эквивалентные названия, которые можно встретить, — *зарезервированные слова* или *ключевые слова*.

Примеры ключевых слов языка Delphi (неполный список):

```
absolute abstract and as array begin case class const constructor
destructor div do downto dynamic else end external file for forward
function if implementation in inline interface interrupt label mod nil not
object of or packed procedure program record resourcestring repeat set shl
shr then to type unit until uses var virtual while with xor
```

Служебные слова в Pascal/Delphi нельзя использовать в качестве имён констант, переменных и т.п. Аналогичная ситуация наблюдается и в подавляющем большинстве современных языков программирования. Некоторые ранние языки программирования, такие как PL/I (1964 г.), не имели такого ограничения, однако это оказалось сомнительным решением, к тому же значительно усложняющим компилятор, поэтому в наши дни подобные вольности можно встретить в языках программирования крайне редко.

Лексическая структура языка Delphi

Одно из самых базовых понятий, относящихся к языкам программирования, — понятие лексемы.

Лексема — неделимая последовательность символов алфавита, которая имеет в программе определённый смысл.

Аналогом лексемы в естественных языках является слово. Оно, как и лексема в языке программирования, может состоять из одного или нескольких символов алфавита, но в отличие от букв имеет лексическое значение, т.е. связанный с этим набором букв смысл.

В языках Pascal/Delphi выделяют следующие основные виды лексем (первые два уже были рассмотрены ранее):

- простые и составные специальные символы;
- ключевые слова;
- комментарии;
- пробельные последовательности;
- идентификаторы;
- литералы.

Приведённая классификация — не единственная возможная, однако именно в таком виде она отражает то, как обрабатывает текст программы компилятор. Комментарии и пробельные последовательности, как правило, не влияют на смысл программы, поэтому часто их не относят к лексемам, однако технически они таковыми являются.

Комментарий — лексема, которая служит для внесения пояснений в текст программы.

В Delphi, как и в большинстве языков программирования, имеется два вида комментариев: однострочные и многострочные. *Однострочные комментарии* начинаются с двух символов `//` и продолжаются до конца строки:

```
// Решение квадратного уравнения
d := b * b + 4 * a * c;
x1 := (-b + Sqrt(d)) / (2 * a);
x2 := (-b - Sqrt(d)) / (2 * a);

// Отображение результатов
WriteLn('x1 = ', x1:0:5, ' x2 = ', x2:0:5);
```

Многострочные комментарии записываются с помощью фигурных скобок и распространяются на произвольное количество строк в тексте программы. При этом диграфы `(* *)` эквивалентны фигурным скобкам:

```
{ Это пример записи многострочного комментария. Многострочные
  Комментарии часто используются для внесения в текст программы
  объёмных пояснений, а также для записи информации об авторе,
  условиях распространения исходных кодов программы и т.п.

  Внутри многострочного комментария можно использовать любые символы,
  кроме ограничивающих его. Например, внутри фигурных скобок можно
  использовать диграфы (* заменяющие фигурные скобки *), а внутри }
(* этих диграфов — { фигурные скобки }.

  Комментарии не могут быть вложенными. *)
{ Разумеется, многострочный комментарий может содержать одну строку }
(* Независимо от того, какие символы-ограничители использованы *)
```

Комментарии (как однострочные, так и многострочные) игнорируются компилятором и могут встречаться везде, где допускается использование пробельных символов и их последовательностей.

В зависимости от расположения и назначения выделяют следующие виды комментариев:

- *вводный комментарий* — комментарий, который записывается в начале текста программы и содержит общие сведения о ней: назначение, используемые методы вычислений, длительность работы программы, необходимые ресурсы памяти, дата разработки и последнего обновления, авторы, правила использования и т.п.
- *комментарий-заголовок* — комментарий, который записывается перед подпрограммами (или большими фрагментами программы) и содержит описание соответствующих подпрограмм (фрагментов программ), выполняемых ими действий, положенных в основу их работы методов и т.п.
- *строчный комментарий* — комментарий, описывающий небольшой фрагмент программы.

К пробельным символам относят собственно символ пробела (часто на письме его обозначают символом «␣»), а также символ табуляции (набирается с клавиатуры клавишей Tab), символ перехода на новую строку и т.п. Последовательности из одного или нескольких таких символов образуют лексему и выступают в качестве разделителей между другими лексемами.

Как и комментарии, пробельные последовательности сами по себе не влияют на смысл программы. Существуют языки программирования, в которых это не так, но чаще всего это либо ранние версии языков из 50-х гг. XX века (Fortran), либо языки, авторы которых посчитали хорошей идеей сделать пробельные последовательности значимыми для смысла программы (Python). В большинстве языков программирования это не прижилось, т.к., во-первых, пробельные символы могут быть легко искажены при обработке различными программами (в т.ч. при передаче по сети в неправильном режиме), а во-вторых, такая особенность повышает сложность компилятора, что негативно сказывается на скорости его работы.

Следующий вид лексем — идентификаторы. Чаще всего идентификаторы — это имена переменных, констант, типов или подпрограмм, реже — меток.

Идентификатор — последовательность символов алфавита языка, используемая для обозначения какого-либо элемента программы или его атрибутов.

Важным свойством идентификатора является его способность идентифицировать элемент программы, т.е. однозначно выделять этот элемент из множества ему подобных. В повседневной жизни роль идентификаторов выполняют, например, имена. Однако нужно иметь в виду, что не всегда имя справляется с задачей идентификации. Например, в беседе в кругу друзей слова «Вася» может быть достаточно, чтобы всем было понятно, о ком идёт речь. Вместе с тем одного только этого слова будет недостаточно при подаче документов в университет: Василиев может оказаться много, и тогда понадобится ещё хотя бы фамилия, иногда — отчество, а бывает и так, что этого тоже недостаточно, если среди поступающих есть полные тёзки.

В Pascal/Delphi идентификатором может быть любая последовательность букв и цифр, начинающаяся с буквы. При этом следует помнить, что в алфавите языка к группе букв относится также символ подчёркивания. Таким образом, следующие последовательности символов могут быть идентификаторами:

```
h      x1      Count      UserName      cbItemSize      day_of_week      _11
```

тогда как такие последовательности символов корректными идентификаторами не являются:

```
2x      User.Name      vasya-pupkin
```

Идентификаторы, используемые в программе, могут быть *стандартными (предопределёнными)* или *определёнными программистом*. К стандартным идентификаторам относятся те, которые входят в описание языка. В основном это имена стандартных типов данных, а также определённых для работы с ними подпрограмм (процедур и функций).

Хороший стиль программирования предполагает использование идентификаторов, которые отражают назначение соответствующего элемента программы.

Литерал — запись в исходном коде программы (лексема), которая представляет конкретное фиксированное значение.

Выделяют литералы различных типов данных. Например, *целочисленные литералы*, т.е. литералы, значениями которых являются целые числа, в Delphi могут записываться двумя способами: в 10-чной и в 16-чной системе счисления. В первом случае используется привычная математическая запись:

```
1321548      21114      0      45      -18      +53
```

Во втором случае (для записи в 16-чной системе счисления) сначала записывается символ «$», а затем одна или несколько 16-чных цифр (в том числе это могут быть латинские буквы от A до F):

```
$0      $5F      $BE8D      $C001C0DE      $FFFF0000
```

Вещественные литералы также имеют две формы записи: в форме с фиксированной точкой и в форме с плавающей точкой. В первом случае используется привычная форма записи:

```
5.3      -96.5324      +7589      -0.956      0
```

Во втором случае отдельно задаются мантисса и порядок числа:

```
1.2E-5      -75.9e8      0.2E+3      -3e8      +57.2E-4
```

Символьные литералы также могут записываться двумя способами. Один из способов заключается в явно указании символа между апострофами:

```
'a'      '@'      '%'      ' '      '5'
```

Второй способ применяется, как правило, тогда, когда по какой-то причине невозможно или нецелесообразно задавать символ его явным указанием. Например, крайне проблематично было бы явно задавать символы табуляции, перевода строки и так называемые управляющие (непечатаемые) символы. В этом случае вместо указания самого символа можно указывать его код, причём как в 10-чной, так и в 16-чной системе счисления:

```
#13     #$0A     #32     #$FF
```

Иногда в тексте программы необходимо задавать не только отдельные символы, но и сразу целые последовательности из них. В этом случае используются так называемые строковые литералы. Они, как и символьные, записываются между апострофами, но могут состоять более чем из одного символа:

```
'Hello, world!'     'Пример строкового литерала'
```

Все приведённые примеры являются различными видами литералов.

Дополнительные вопросы

1. Постройте синтаксическую диаграмму, отражающую правила записи идентификаторов.

2. Предложите РБНФ-описание для правил записи вещественных литералов.