Лицензия

ЧАСТЬ 2

2 Вещественные числа

Математически, множество вещественных (или действительных) чисел $\mathbf{R}$ образуется из множества целых $\mathbf{Z}$ путем присоединения к нему нецелых значений. В вычислительной технике и программировании под вещественными числами понимают подмножество рациональных чисел $\mathbf{Q}$, представляющих собой значения, которые можно выразить обыкновенной дробью m/n. Речь идет именно о подмножестве, поскольку под хранение данных такого типа отводится определенное фиксированное количество машинной памяти. Из-за этого ограничения большая часть рациональных чисел представляется приближенными с некоторой точностью значениями (в этом случае мы говорим о ***машинной точности***). Конечно, если нужна абсолютная точность, можно создать собственный тип данных, в которых числитель и знаменатель дроби хранятся в виде пары целочисленных значений, и описать набор операций над ними, но такие типы будут рассматриваться позднее. Отметим лишь, что библиотека численных методов NumLibABC, входящая в состав PascalABC.NET, включает класс для работы с обыкновенными дробями, реализованный именно описанным выше способом.

Формат объявления констант и переменных вещественного типа не отличается от формата для целочисленных типов.

2.1 Типы вещественных чисел

PascalABC.NET предоставляет три типа вещественных чисел: **single**, **real** (и его синоним **double**) и **decimal** (все цифры, хранимые в этом типе, точные).

Длина, байт	Тип и количество значащих цифр	Диапазон представляемых значений
4	**single** 7-8	-1.8×10^{308} .. 1.8×10^{308}
8	**real (double)** 15-16	-3.4×10^{38} .. 3.4×10^{38}
16	**Decimal** 28-29	-79228162514264337593543950335 .. 79228162514264337593543950335

Таблица 2.1. Вещественные типы данных

Тип **decimal** достаточно специфичен, поэтому его пока мы рассматривать не будем. В частности, для него не определено большинство стандартных математических функций.

Для вещественных типов определены стандартные константы MaxReal (MaxDouble) и MaxSingle, представляющие собой максимальные значения, которые может хранить соответствующих тип данных. Эти значения указаны в таблице 2.1. Кроме того, определены константы вида T.V, где T принимает значения типа **single**, **real** или **double**, а V описывает, что собой представляет константа.

`R.MaxValue` – максимальное значение типа R;
`R.MinValue` – минимальное значение типа R;
`R.Epsilon` – самое маленькое положительное значение типа R;
`R.NaN` – значение «не число» (Not a Number), которое получается, например, при делении 0/0;
`R.NegativeInfinity` – «минус бесконечность», которое получается при делении отрицательного числа на ноль, например, -2/0;
`R.PositiveInfinity` – «плюс бесконечность», которое получается при делении положительного числа на ноль, например, 2/0.

Имеются и другие константы, например, число π: `Pi = 3.141592653589793` и основание натурального логарифма e: `E = 2.718281828459045`.

2.2 Литералы вещественного типа

Литерал изображает в программном коде некоторое значение при помощи символов. Вещественное число, может иметь одну из следующих форм записи:

- m.n – привычный формат записи числа, имеющего целую часть m и дробную n, например, 3.14, -1000.2643, 0.00002432234;
- m.nEk – «научный формат» записи числа, в котором латинская буква E (или e) отделяет мантиссу m.n от десятичного порядка k. Мантисса может быть записана в виде целого числа – тогда литерал имеет вид mEk. Примеры записи литералов в научном формате:

```
3E6 // 3×10^6
5.7e-11 // 5.7×10^-11
-0.000234e-7 // -0.000234×10^-7 = -0.234×10^-10
```

2.3 Арифметические выражения вещественного типа

Правила составления и записи арифметических выражений вещественного типа такие же, как для целочисленных выражений. Любые части целочисленных выражений могут входить в вещественное выражение. Разница имеется лишь в некоторых операциях.

2.3.1 Деление и возведение в степень

Операция деления бинарная и записывается при помощи символа «слэш» (/). Операция q/r возвращает результат деления q на r вещественного типа. Например, 5/8 возвращает вещественное значение 0.625, несмотря на то, что литералы 5 и 8 изображают целые числа. Здесь произойдет автоприведение типа, причем для операции деления оба целочисленных операнда приводятся к типу **real**.

Операция деления имеет приоритет 2, такой же, как операции *, div, mod.

Операция возведения в степень также бинарная; ее знак операции ** представляет собой пару знаков операции умножения, следующих друг за другом без пробела, а приоритет равен 1. Запись вида a**b обозначает операцию a^b. Реализация этой операции в PascalABC.NET базируется на стандартных средствах платформы Microsoft .NET Framework (System.Math.Pow), что накладывает некоторые ограничение: при отрицательном основании степени показатель может быть только целочисленным. Это легко объяснить. Когда a и b вещественные, возведение в степень производится в соответствии с известным тождеством

$$a^b = e^{b \cdot \ln a}$$

Понятно, что при неположительном значении переменной a невозможно вычислить логарифм. Тем не менее, на случай такого некорректного возведения в степень предусмотрено получение результата со значением NaN (Not a Number).

```
begin // p020301
  var x := 5 ** 8;
  var b := 3.5 ** 2.9;
  Println(x, b, x ** b, 3 ** 0, 4 ** (-3), (-4.2) ** (-1.12))
end.
```

Приведенная программа выдаст следующий результат

```
390625 37.826601883412 3.2906421758964E+211 1 0.015625 NaN
```

Здесь также имело место автоприведение целочисленных литералов к типу **real**.

> В бинарных вещественных операциях и при обращении к функциям, требующим вещественные параметры, значения всех целых типов, кроме BigInteger, будут автоматически приведены к типу **real**.

2.3.2 Приведение типа

Приведение целого типа к вещественному в худшем случае приводит к потере точности в представлении числа.

```
begin
  var i := 1234567891011213141;
  var a := i + 0.0;
  Writeln(i:21, a:21);
end.
```

Получим следующий результат (подчеркнутые цифры потеряны):

```
1234567891011213141 1.23456789101121E+18
```

Автоприведение значения вещественного типа к целым типам запрещено и попытка его организовать приведет к выдаче сообщения об ошибке при компиляции. Это понятно: можно потерять дробную часть полностью. Программист обязан указать, как именно производить такое приведение: отбросить дробную часть или округлить значение до целого по правилам арифметики. Отбрасывает дробную часть функция Trunc(a), округляет – функция Round(a).

```
begin
  var (a, b, c, d) := (-12.87, 12.87, -5.2, 5.2);
  Println(Trunc(a), Round(a));
  Println(Trunc(b), Round(b));
  Println(Trunc(c), Round(c));
  Println(Trunc(d), Round(d))
end.
```

```
-12 -13
12 13
-5 -5
5 5
```

Разберитесь, почему результаты получились именно такими.

> Функция Round(a) реализует так называемое «банковское» округление. Если вещественное значение находится точно посередине между двумя целыми, то округление осуществляется к ближайшему четному целому значению. Например, `Round(2.5)=2, Round(3.5)=4`.

Тип можно приводить явно подобно тому, как это делалось для целых типов. Например, **single**(1 / 3), **real**(6.18) и т.п. Это требуется редко, поскольку автоприведение производится к типу **real**, а тип **single** не дает ничего, кроме потери точности.

2.3.3 Об арифметическом переполнении

В отличии от ситуации с переполнением в целочисленной арифметике, аппаратная часть компьютера умеет сообщать о переполнении при выполнении операций с плавающей точкой (так именуются в архитектуре компьютеров форматы представления вещественных данных). Соответственно, программа также умеет реагировать на такие переполнения. Эта реакция состоит в возвращении в качестве результата переполнения констант `NaN`, `NegativeInfinity` или `PositiveInfinity`. Значения таких констант могут быть операндами выражений.

```
begin
  var a := 0 / 0;
  var b := 2 / a;
  var c := 1 / 0;
  var d := 1 / c;
  var e := a * c;
  Println(a, b, c, d, e)
end.
```

```
NaN NaN Infinity 0 NaN
```

Проанализируем полученные результаты

0 / 0 дает NaN – это понятно, результат операции не является числом. 2 / NaN также дает NaN, что тоже понятно. 1 / 0 дает бесконечность (∞ – Infinity) и это не вызывает сомнений. 1 / ∞ – конечно же, ноль. NaN * ∞ – наверное, NaN тут лучшее, что можно придумать.

Существуют логические функции, позволяющие обнаруживать различные переполнения и в зависимости от результатов направлять дальнейший ход работы программы. Об этом – в следующих частях. Пока достаточно помнить, что переполнение при работе с вещественными числами само по себе не приводит к аварийному завершению программы.

2.3.4 Некоторые математические функции

Бибилиотека PascalABC.NET насчитывает большое количество математических функций, работающих с вещественным типом данных. Подробно об этом можно узнать в Справке по PascalABC.NET. Приведем часть из них. В описании функций a и b представляют вещественные значения, n – целочисленные значения.

`Abs(a)` – возвращает абсолютное значение a;
`ArcSin(a)` – возвращает арксинус a в радианах. Есть также `ArcCos` и `ArcTan`;
`Ceil(a)` – возвращает ближайшее большее a целочисленное значение;
`Cos(a)` – возвращает угла a, заданного в радианах. Есть также `Sin(a)`, `Tan(a)`;
`DegToRad(a)` – возвращает радианную меру угла a, заданного в градусах;
`Exp(a)` – возвращает e^a;
`Floor(a)` - возвращает ближайшее меньшее a целочисленное значение;
`Frac(a)` – возвращает дробную часть a;
`Int(a)` – возвращает целую часть a;
`Ln(a)` – возвращает натуральный логарифм a. Есть синоним `Log(a)`;
`Log10(a)` – возвращает десятичный логарифм a;
`Log2(a)` – возвращает логарифм a по основанию 2;
`LogN(a,n)` – возвращает логарифм a по основанию n;
`Max(a,b)` – возвращает максимальное из значений a и b;
`Min(a,b)` – возвращает минимальное из значений a и b;
`RadToDeg(a)` – возвращает значение угла a, переведенное из радиан в градусы;
`Random` – возвращает случайное число на интервале [0;1];
`Random(a,b)` – возвращает случайное число на интервале [a;b];
`Round(a)` – возвращает целое значение a;
`Round(a,n)` – возвращает значение a, округленное до n знаков после запятой;
`Sign(a)` – возвращает значение -1.0, 0.0, 1.0 в зависимости от знака числа;
`Sinh(a)` – возвращает гиперболический синус a. Есть также `Cosh(a)`, `Tanh(a)`;
`Sqr(a)` – возвращает a^2;
`Sqrt(a)` – возвращает $\sqrt{a}$;
`Trunc(a)` – возвращает целое значение a, отсекая дробную часть.

2.4 Точность машинной арифметики

Поскольку вещественные значения представляются в компьютере с некоторой конечной точностью, в ряде случаев приходится принимать определенные меры с тем, чтобы получить приемлемый результат. Наибольшее число проблем порождают операции, многократно увеличивающие результат вычитания близких по значению чисел. Предположим, мы хотим вычислить значение следующего выражения, равное 10^{20}

$$\frac{1}{\left(\frac{1}{3}+10^{-20}\right)-\frac{1}{3}}$$

Если запрограммировать выражение «как есть», получим следующий код:

```
begin
  Println(1 / ((1 / 3 + 1e-20) - 1 / 3))
end.
```

При выполнении этой программы выводится значение `Infinity`. Значение ⅓ будет вычисляться в представлении типа real, т.е. иметь до 16 верных цифр. В этих условиях сложение с величиной 1e-20, представляющей собой единицу в двадцатом после запятой разряде, не изменит вычисленного значения. Последующее вычитание ⅓ даст в результате ноль. Деление единицы на ноль, конечно же, даст бесконечность.

Если раскрыть скобки в знаменателе, это ничего не даст: приоритет операций сложения и вычитания одинаков, так что сначала будет выполнено сложение, а затем вычитание. Но если принудительно поменять местами члены выражения, можно получить правильный результат 1e+20.

```
begin
  Println(1 / (1 / 3 - 1 / 3 + 1e-20))
end.
```

Приведенный пример показывает, что при программировании даже внешне простых выражений можно получить неожиданные проблемы, если не учитывать особенностей работы с вещественными числами.

2.5 Вывод вещественных значений

Оператор Print(a) выводит результат с 15 значащими цифрами после запятой, либо с меньшим числом знаков, если число имеет точное машинное представление.

```
begin
  Println(1 / 8, 0.000001, 352 / 1231234, 2 * 124534 + 4563325351.8)
end.
```

```
0.125 1E-06 0.000285892040018388 4563574419.8
```

Если нужно управлять представлением выводимых значений, используется оператор базового Паскаля `Write(a:i:j)`, где a – выводимое значение вещественного типа, i – общее количество позиций, отводимых под вывод, j – количество позиций, отводимых под знаки в дробной части числа. При i=0 количество позиций под вывод определяется автоматически, при j=0 дробная часть не выводится. Отметим, что при выводе число округляется до j знаков после запятой.

```
begin
  Writeln(1 / 8:0:5, 0.000001:10:7, 352 / 1231234:10:5,
      2 * 124534 + 4563325351.8:16:3)
end.
```

```
0.12500 0.0000010   0.00029  4563574419.800
```

2.6 Ввод вещественных значений

PascalABC.NET при вводе значений типа **real** предлагает совмещать описание переменной с вводом ее значения:

```
var ИмяПеременной := ReadReal('Текст приглашения ко вводу');
```

Если приглашение не нужно, оператор выглядит еще проще

```
var ИмяПеременной := ReadReal;
```

Имеется также вариант оператора с `ReadlnReal`, но поскольку символьные данные тут не вводятся (о них – в части 6), достаточно писать `ReadReal`.

Если переменная была описана ранее, ключевое слово **var** указывать не нужно.

Как и при вводе данных типа **integer**, можно вводить значения двух и трех переменных.

```
var (имя1,имя2) := ReadReal2('Текст приглашения ко вводу');
var (имя1,имя2,имя3) := ReadReal3('Текст приглашения ко вводу');
```

Вещественные значения также можно вводить при помощи оператора базового Паскаля Read, указывая в списке ввода переменные вещественного типа. Формат оператора ввода не отличается от рассмотренного в части 1 для целочисленных типов.

2.7 Обращение к системным библиотекам .NET

В состав функций для работы с вещественными значениями PascalABC.NET не вошла часть функций, доступных в среде Microsoft .NET Framework. В частности, может быть полезной функция Atan2(y, x), возвращающая значение угла в радианах, тангенс которого равен отношению y / x. Другая интересная функция, IEEEReminder(x, y), возвращает значение x – y / Q, где Q – округленный до ближайшего целого результат деления x / y. Вызов этих функций производится в виде System.Math.Atan2(…) и System.Math.IEEEReminder(…) соответственно.