ТУБ №28:
Тип диапазон в Delphi

Программисты ошибаются. Много. Если бы за каждую допущенную в программе ошибку, которую заметили пользователь, 1% от зарплаты программистов шёл пользователю, программисты бы обнищали, а пользователи бы озолотились. Наверняка Вы и сами без труда приведёте примеры программ, которые при относительно небольшой сложности просто-таки кишат неожиданными отказами.

Вдвойне плохо, если ошибка допускается в программе, которая управляет чем-то опасным для жизни и здоровья людей: самолётом, атомной электростанцией, Вашим начальником. Здесь последствия могут быть плачевными и стоить людям жизней, а программисту — лет, проведённых за решёткой.

Всё это давным-давно заставило программистов придумать в языках программирования механизмы, позволяющие выявлять ошибки как можно проще. Проще выявить — проще устранить, меньше шансов, что какая-то останется незамеченной. Один из таких механизмов — это типизация, в особенности та, которая называется строгой. Та самая, которая не позволяет Вам записать, скажем, в целочисленную переменную значение символьного типа.

В языках Pascal и Delphi пошли дальше и придумали специальные типы — диапазоны. *Тип диапазон* — это тип данных, который получается из другого типа путём наложения ограничений на диапазон его значений.

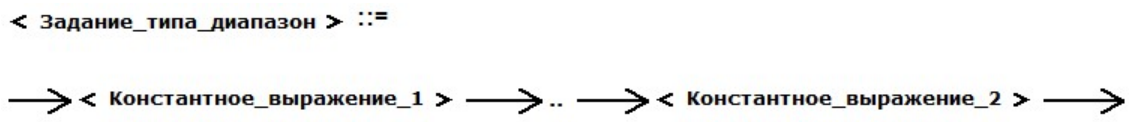
Давайте разбираться. Предположим, нужно написать программу, которая работает с датами и временем. Для хранения номера дня в месяце, количества часов, минут и секунд напрашивается использование целочисленных типов. Но так ли хороша эта идея?

Минимальный целочисленный тип занимает 1 байт и может иметь одно из 256 различных значений, от 0 до 255 для беззнакового типа Byte и от –128 до 127 для знакового типа ShortInt. Очевидно этого достаточно для хранения вышеперечисленных величин, но есть проблемка: скажем, номер дня в месяце на самом деле принимает значения от 1 до 31, ну максимум 33, если Вы из будущего и Земля стала слишком медленно двигаться вокруг солнца. С часами, минутами и секундами та же история: весь диапазон значений целочисленных типов им не нужен.

Это не проблема, если Вы планируете просто хранить даты и время. Но что, если Вы начинаете выполнять вычисления? Например, определять, какой день был через миллиард секунд после полуночи 1 января 1918 г. Здесь начинаются вычисления с умножениями, делениями, сложениями и вычитаниями, которые, к тому же, должны учесть, что между 23:59:59 31 января 1918 г. и 00:00:00 14 февраля 1918 г. прошла всего лишь одна секунда, что были годы високосные и не очень и т.п. Даты и время — вообще кладезь неожиданностей и источник бесчисленного множества ошибок [1].

Представляете, как непросто будет тестировать Вашу программу, проверяя, чтобы ни при каких обстоятельствах не случилось 38 марта или –7 июня?

Встречайте тип диапазон! Синтаксис задания диапазонов таков:



Поскольку диапазоны объявляются путём ограничения какого-либо типа, который называется *базовым типом*, оба константных выражения должны иметь один и тот же тип. Например:

```
type
  TDayOfMonth = 1..31;
  TCapitalLetter = 'A'..'Z';

var
  HoursNow: 0..23;
```

Базовый тип должен обладать свойством перенумерованности. Это логично, т.к. если значения типа не могут быть однозначно расположены по возрастанию, то может быть проблематично задать диапазон, т.к. он предполагает не только определённые границы, но и некоторую последовательность значений между ними. Поскольку диапазон получается из перенумерованного типа путём выбора некоторого подмножества его подряд идущих значений, полученный тип также обладает свойством перенумерованности.

Внутреннее представление диапазонов такое же, как и у базовых типов. Фактически значения типа диапазон ведут себя, как и значения базового типа — над ними определены те же *операции*, те же *процедуры и функции* и т.д. — но компилятор может выполнять дополнительный контроль за правильностью использования этих значений, а именно — за тем, чтобы получаемые значения соответствовали ограничениям типа диапазон.

Частично эти проверки компилятор выполняет во время компиляции — там, где это технически осуществимо. Но это возможно не всегда, поэтому существует также возможность выполнять аналогичные проверки во время выполнения программы. Т.к. во время выполнения программы компилятора уже нет, то эти проверки включаются в машинный код самой программы.

Для управления этой возможностью имеется директива компилятора {$R}. По умолчанию она установлена в {$R–}, что означает для компилятора «не добавлять в машинный код программы проверки диапазонов (range checking)». Если установить её значение в {$R+}, компилятор автоматически добавит ко всем вычислениям в программе контроль за получаемыми результатами для всех типов данных, в том числе и диапазонов. Тогда, если при вычислениях получается величина, которая не попадает в диапазон значений того типа данных, который будет использоваться для хранения результатов (целочисленного, символьного, диапазона и т.п.), во время работы программы будет сгенерирована ошибка.

Следует понимать, что подобные проверки выполняются уже во время работы программы, а значит, замедляют её выполнение. По этой причине по умолчанию они отключены. Назначение этих проверок — преимущественно для отладки и контроля программы во время разработки и доработки. В версии программы, которая поступает конечному пользователю, как правило, имеет смысл подобные проверки не включать.

[1] https://habr.com/ru/post/313274/

Дополнительные вопросы

1. Почему базовым типом для диапазона не может быть тип Comp?

2. Почему при хранении даты и времени для секунд может быть недостаточно диапазона 0..59?