ТУБ №20:
Логические типы данных в Delphi

Часто в программы бывает нужно хранить данные, которые представляют собой ответ на простой вопрос: «Заселён ли студент в общежитие?», «Студент “платник” или “бюджетник”?», «Разрешён ли доступ пользователю к ресурсу?», «Хочет ли пользователь получать бесплатную рассылку по SMS?», «Зашёл пользователь на сайт со смартфона или с компьютера?», «Есть ли в ноутбуке пользователя поддержка запахов?» Такие вопросы предполагают выбор из двух вариантов: да/нет, истина/ложь.

Критично настроенный читатель скажет, что правильно не «ложь», а «клади», но мы здесь не лингвистику обсуждаем, так что будем смотреть на этот вопрос ширше.

В ранних версиях некоторых языков (например, C и Basic) для хранения подобных данных применяли обычно целочисленный тип: «нет» (ложь) обозначали числом 0, а «да» (истину) — ненулевыми значениями, чаще всего числом 1. Недостаток такого подхода заключался в том, что компилятор рассматривал такие переменные, как целочисленные, ввиду чего не препятствовал записи в них произвольных целочисленных значений, что не очень-то способствовало выявлению ошибок.

Одним из первых языков, в которых для этих целей появился самостоятельный тип — логический, — стал Algol 60 (появился в 1960 г.). Уже через 2 года аналогичный тип появляется в Fortran IV — новой версии одного из самых популярных языков программирования того времени. В свой новый Pascal (1970 г.) добавляет его и Никлаус Вирт. В языке Basic (1964 г.) полноценный логический тип появится при перерождении в Visual Basic в 1991 году. Язык C (1972 г.) будет сопротивляться прогрессу вплоть до выхода в 1999 году стандарта C99.

Таким образом, наличие отдельного логического типа данных в Delphi было просто неизбежным. Как и в самой первой версии Pascal, логический тип здесь называется Boolean. Его размер — 1 байт, а множество значений — соответственно значения «Истина» и «Ложь», для обозначения которых предусмотрено две стандартных (встроенных в язык) константы, False и True.

Название типа	Размер	Описание
Boolean	1 байт	Основной логический тип
ByteBool	1 байт	Добавлены для совместимости с Windows и языком C
WordBool	2 байта	
LongBool	4 байта	

Кроме Boolean, начиная с поздних версий Borland Pascal имеются также типы ByteBool, WordBool и LongBool. Их появление в языке вызвано тем, что основным языком разработки Windows стал язык C, вследствие чего логические величины при взаимодействии с этой ОС задаются целыми числами, причём используются знаковые целочисленные типы. Более того, в некоторых случаях «логические» значения в Windows могут иметь, например, три разных значения. Например, функция GetMessage объявлена, как возвращающая значение типа BOOL, и может вернуть значения TRUE (число 1), FALSE (число 0) и –1.

ByteBool, WordBool и LongBool позволяют работать с подобными величинами, как с логическими значениями, имея контроль со стороны компилятора, и в то же время корректно интерпретировать странные значения, приходящие от ОС. Для использования внутри программы предпочтительным является тип Boolean, остальные предназначены для того, чтобы скрывать несовершенства внешнего мира.

Внутреннее устройство переменной логического типа данных не уточняется, однако на практике False представляется значением $00000000_{(2)}$, а True может записываться как $00000001_{(2)}$ либо как $11111111_{(2)}$. В общем случае следует предполагать, что любая ненулевая комбинация битов эквивалентна True.

Над данными логических типов определены логические операции и операции сравнения:

Операция	Вид	Описание	Тип результата
Логические операции			
not	Одноместная	Инверсия (НЕ)	Логический
and	Двухместная	Логическое умножение (И)	Логический
or	Двухместная	Логическое сложение (ИЛИ)	Логический
xor	Двухместная	Логическое исключающее ИЛИ	Логический
Операции сравнения			
=	Двухместная	Равно	Логический
<>	Двухместная	Не равно	Логический
<	Двухместная	Меньше	Логический
>	Двухместная	Больше	Логический
<=	Двухместная	Меньше или равно	Логический
>=	Двухместная	Больше или равно	Логический

В отличие от поразрядных (побитовых) операций над целочисленными типами данных логические операции применяются не к отдельным разрядам, а к логическим значениям в целом. Операции сравнения считают False < True.

В языках Pascal и Delphi поддерживается две *модели вычисления логических операций* `or` и `and`: полная и сокращённая. По умолчанию применяется *сокращённое вычисление логических операций*: вычисление выражения прекращается, как только становится очевидным его результат. Рассмотрим, например, такое выражение:

```
(x > 5) and (y < 10) and (z > 42)
```

Обратите внимание: если первая часть выражения, `x > 5`, оказалась ложной, то продолжать сравнивать остальные переменные не имеет смысла, т.к. итоговым результатом всё равно будет False. Аналогичная ситуация возникает и с операцией `or`:

```
(x > 5) or (y < 10) or (z > 42)
```

Как только хотя бы одно из сравнений дало истинный результат, сразу же можно говорить о том, что и всё выражение целиком будет истинным.

Сокращённое вычисление логических операций позволяет повысить скорость работы программ и используется по умолчанию. Тем не менее, иногда возникает необходимость выполнять обработку выражения до конца. На этот случай предусмотрена возможность переключения компилятора на *полное вычисление логических операций*, при котором все необходимые действия для вычисления выражения будут выполняться, даже если результат вычислений уже очевиден.

Каких-либо особых встроенных функций у логического типа нет:

Имя	Вид	Описание	Тип результата
Определённые для всех перенумерованных типов			
Ord(x)	Функция	Возвращает порядковый номер значения аргумента во множестве значений типа	Integer
Pred(x)	Функция	Возвращает значение, следующее во множестве значений типа перед значением аргумента	Тип x
Succ(x)	Функция	Возвращает значение, следующее во множестве значений типа после значения аргумента	Тип x
Определённые для всех типов			
SizeOf(x)	Функция	Возвращает размер типа в байтах	Integer

Функция Ord возвращает 0 для False и 1 для True. Из списка встроенных функций можно сделать вывод о том, что значения логического типа обладают свойством перенумерованности.

Отдельно следует отметить, что сравнение логического значения с константами True и False, хоть и допускается языком, считается дурным тоном. Например, нежелательной считаются записи

```
bIsWindowVisible = True
bIsWindowVisible = False
```

Вместо них предпочтительнее использовать соответственно

```
bIsWindowVisible
not bIsWindowVisible
```

Такая запись, во-первых, более компактна, а во-вторых, получается приближенной к естественному языку. К тому же, результат сравнения имеет логический тип, а переменная `bIsWindowVisible` и так логического типа, поэтому сравнение, например, с True оказывается просто избыточным.

Дополнительные вопросы

1. Для хранения одного из двух возможных значений достаточно одного бита. Почему тип Boolean имеет размер 1 байт?

2. Даны переменные A и B (целого или вещественного типов). Как записать выражение, результатом вычисления которого будет большее из двух значений, используя только операции и встроенные функции?