ТУБ №15:
Целочисленные типы данных в Delphi

— Вызывали, гражданин начальник? — отчеканил Ви, подсаживаясь к вождю Ро, задумчиво помешивавшему угли догорающего костра.

— Ты мне эти свои замашки брось, Ви! Если я тебя по дружбе назначил жин… дар… жиндармэром племени…

— Инженером.

— Да, вот этим вот, инжирмэром… Набрался ж где-то таких слов!.. так это не означает, что тебе всё с рук сойдёт. С крокодилами шутить будешь про своих гранд-павлинов!

Ви хотел было опять подсказать правильное слово, но врождённое чувство такта и потянувшаяся к лежавшему рядом булыжнику рука вождя заставили его сдержаться. Вождь закрыл глаза, пытаясь успокоиться, выдержал паузу и продолжил.

— Есть проблема как раз по твоей части, Ви. Наши бул… алл… гах… тьфу, чтоб тебя!.. Счетоводы нашего племени жалуются, что совсем запутались с подсчётом коз. Вот, смотри, принесли мне свои… эти…

Вождь достал из складок надетой на нём шкуры несколько деревянных табличек с засечками и аккуратно проделанными в некоторых местах отверстиями.

— Дендрокарты, — подсказал Ви.

— Да, дыркокарты, — нетерпеливо подтвердил вождь. — У них, в общем, баланс не сходится.

— Так-с, у нас было три козы, верно? — Ви взял в руки одну из табличек и принялся её изучать.

— Верно, три козы, — подтвердил вождь.

— Потом охотники привели ещё двух коз, — Ви ткнул пальцем в табличку чуть пониже. — Значит, всего у нас стало…

Вождь посмотрел на Ви в ожидании ответа.

— Ну так сколько? — переспросил вождь, не дождавшись ответа.

— Вождь Ро, мы же с тобой уже проходили таблицу сложения. Не капризничай, напрягись, это же несложно!

Вождь потупил взгляд, потом снова взглянул на Ви, но на лице того по-прежнему читалась едва уловимая улыбка учителя, ожидающего ответа от своего ученика. Глубоко вздохнув, вождь закрыл глаза и принялся что-то бормотать, рисуя указательным пальцем в воздухе какие-то фигуры.

Пользуясь заминкой, Ви снял с лежавшей рядом с вождём ветки пригоршню спелых ягод, отправил их себе в рот и, стараясь не чавкать слишком уж громко, принялся разглядывать дендрокарту. Вдоль вытянутой таблички тянулись две параллельные засечки, которые пересекались примерно с десятком коротких засечек-строк. На некоторых пересечениях были проделаны аккуратные отверстия.

Иногда в тех строках, где было два отверстия, сбоку добавлялась ещё одна маленькая засечка. Эта идея, предложенная Ви, в своё время помогла ему занять высокую должность главного счетовода племени, а главного счетовода Кри, не справлявшегося больше со своими обязанностями, отправили охотиться на мамонтов.

Незавидная судьба предшественника несколько омрачала карьерный рост, но Ви успокаивал себя тем, что эта работа действительно куда лучше подходила Кри. Тот потом, чтобы вернуться в должность, пробовал предлагать разные идеи: то отказаться от имён и указывать друг на друга пальцем, то счетоводство сделать открытым и разрешить участвовать в нём всем, то мамонтов считать большими козами. Племя тыканье пальцем не оценило, а всеобщую открытую бухгалтерию запретил под страхом ссылки к крокодилам сам вождь Ро после того, как странный Ша сломал две дендрокарты. Точку в истории Кри поставил мамонт. Мало кому нравится, когда его называют козлом.

— Два, три, много… Много! — произнёс наконец вождь.

— Просто «много»? — переспросил Ви.

— Два, три, много… Очень много! — исправился вождь.

— Прямо в мамонта! — похвалил вождя Ви.

Вождь засветился от счастья.

— А теперь смотри, вождь Ро, — Ви придвинулся к вождю поближе и показал дендрокарту. — Видишь, они здесь не поставили признак переполнения, вот в этой строке?

— Ну-у-у, вижу, — неуверенно пробормотал вождь, косясь на табличку. — И что теперь делать?

— Ну есть у меня одна идейка, — с видом фокусника, который вот-вот достанет из уха кролика платок с девяткой пик, ответил Ви. — Надо разрядность повышать.

— Кого повышать? — не понял вождь. — У нас больше руководящих должностей нет.

— Больше разрядов для записи чисел использовать. А то у нас сейчас всего-то два бита…

— Погоди-погоди, больше разрядов — это ведь тарифную сетку менять придётся, зарплаты пересчитывать. А этого, Бита, мы же ещё на прошлой неделе крокодилам скормили, разве нет?

Ви загадочно улыбнулся и достал откуда-то маленькую коробочку. Раздался тихий щелчок, коробочка засветилась, Ви направил луч света на сушившуюся неподалёку огромную шкуру антилопы — и на ней появилась надпись

Целочисленные типы данных

Целочисленные типы данных — без преувеличения самые широко используемые типы данных в вычислительной технике. Как можно догадаться из названия, множества значений для этих типов данных представлены целыми числами. В зависимости от того, входят ли во множество значений отрицательные числа, выделяют знаковые и беззнаковые целочисленные типы.

На выбор способа представления данных того или иного типа в языке программирования высокого уровня большое влияние оказывают возможности тех устройств, на которых будут выполняться программы, написанные на этом языке. И это логично: трудно притворяться, что Вы сидите на велосипеде, если под Вами лодка, а в руках вёсла, ещё труднее — крутить педали, но совершенно невозможно — объяснить, ради чего все старания.

Для представления неотрицательных целых чисел (тех, которые используются для подсчёта чего-либо, — больших или равных 0) традиционно используется прямой код, т.е. используется математическая запись чисел в двоичной системе счисления (дополненная незначащими старшими нулями). По этой причине диапазон целых чисел, которые можно записать в переменную беззнакового целочисленного типа, — от 0 до $2^N - 1$, где N — размер типа в битах.

Когда возникает необходимость записывать отрицательные целые числа, как правило, выбирают между обратным и дополнительным кодом. На сегодняшний день большинство вычислительных устройств на аппаратном уровне использует дополнительный код, поэтому именно он оказался закономерным выбором для знаковых целочисленных типов в Delphi. Отличительной особенностью такого способа представления целых чисел является некоторая «несимметричность» множества значений относительно нуля: от -2^{N-1} до $2^{N-1} - 1$, где N — размер типа в битах. Впрочем, это куда проще пережить, чем наличие сразу двух нулей — +0 и –0 — в обратном коде.

Может показаться, что теперь проблема представления целых чисел в памяти компьютера решена, можно давать названия типам данных и переходить к операциям над ними, но нет, история только начинается…

Как типы данных размеры меняли

Дело в том, что у каждого вычислительного устройства, работающего в определённых условиях, есть такие размеры элементов данных, работа с которыми происходит более эффективно, чем с другими. Конечно, эта особенность находит своё отражение и в языках программирования высокого уровня. Проблема заключается в том, что аппаратное обеспечение развивается и со временем эти «родные» размеры типов данных могут становиться другими. Один из таких размеров принято называть разрядностью платформы — именно о нём идёт речь, когда говорят о 32- и 64-битных версиях Windows, например.

Как известно, язык Delphi своими корнями уходит в язык Pascal, а конкретно в один из диалектов — Object Pascal, — который был реализован в среде разработки Borland Pascal (урезанная версия которой известна под названием Turbo Pascal). Поскольку к моменту появления Delphi на этом диалекте языка было написано огромное количество программ, было совершенно логично сохранить в новом языке совместимость с предшественником, в том числе и в части поддерживаемых целочисленных типов данных, чтобы эти программы можно было перенести на новый язык с минимальными усилиями.

Основными целочисленными типами данных в Borland Pascal были следующие:

Название типа	Диапазон значений	Размер
Беззнаковые		
Byte	$0..2^8 - 1$	1 байт
Word	$0..2^{16} - 1$	2 байта
LongWord	$0..2^{32} - 1$	4 байта
Знаковые		
ShortInt	$-2^7..2^7 - 1$	1 байт
Integer	$-2^{15}..2^{15} - 1$	2 байта
LongInt	$-2^{31}..2^{31} - 1$	4 байта

Программы, написанные в Borland Pascal, предназначались для выполнения под управлением ОС MS-DOS в режиме процессора, в котором «родным» размером переменной были 16 бит, или 2 байта. Названия беззнаковых типов «Byte», «Word» и «LongWord» совпадали с общепринятыми терминами для обозначения соответствующих типов данных. За словом «Integer» (англ. целочисленный) закрепили самый эффективный с точки зрения производительности знаковый тип данных. Для редких же случаев, когда требовался знаковый тип большего или меньшего размера, использовали имена «ShortInt» (англ. короткий целочисленный) и «LongInt» (англ. длинный целочисленный). И жили они долго и счастливо…

Тем временем компания Intel начала выпуск процессоров, которые поддерживали новый режим — защищённый (protected mode), — позволивший поднять на новый уровень надёжность работы операционных систем и прикладного ПО. Уже начиная с Intel 80386 у процессоров, работающих в защищённом режиме, «родным» размером стали 32 бита, или 4 байта. В новых версиях Windows этот режим становится основным, и именно в это время ведётся разработка первой версии Delphi.

Для 32-битных версий Delphi основными целочисленными типами данных стали:

Название типа	Диапазон значений	Размер
Беззнаковые		
Byte	$0..2^8 - 1$	1 байт
Word	$0..2^{16} - 1$	2 байта
LongWord	$0..2^{32} - 1$	4 байта
Cardinal	$0..2^{32} - 1$	4 байта (*)
Знаковые		
ShortInt	$-2^7..2^7 - 1$	1 байт
SmallInt	$-2^{15}..2^{15} - 1$	2 байта
LongInt	$-2^{31}..2^{31} - 1$	4 байта
Int64	$-2^{63}..2^{63} - 1$	8 байт
Integer	$-2^{31}..2^{31} - 1$	4 байта (*)

Беззнаковые типы данных пережили переход на 32-битные рельсы без изменений. Для знаковых типов ситуация немного поменялась: к классическим ShortInt и LongInt добавились 2-байтовый SmallInt и 8-байтовый Int64. По сути, единственный классический целочисленный тип, который был изменён, — это Integer. Тот самый, который был самым эффективным в 16-битных программах. В большинстве случаев увеличение размера этого типа данных никак не влияло на работу программ, не считая того, что после компиляции в Delphi они без усилий со стороны программистов начинали работать с большими диапазонами значений, чем раньше, причём с той же высокой эффективностью. В тех же редких случаях, когда требовался именно 2-байтовый знаковый тип, Integer просто заменялся программистами на SmallInt. Но зачем вообще было трогать Integer?

Разработчики Delphi прекрасно понимали: если пришлось менять разрядность однажды, значит, это может произойти снова. Поэтому было принято решение два типа данных — один знаковый (Integer) и один беззнаковый (Cardinal) — заявить как типы данных, размер которых всегда будет «родным» для платформы, на которой выполняется программа. В документации написали, что Integer и Cardinal — это типы, обеспечивающие максимальную производительность, и что в данный момент оба имеют размер 4 байта, но в будущем размер может быть изменён. Размеры остальных типов были объявлены фиксированными.

«Твори бардак — мы здесь проездом!»

Но ни одна хорошая идея не остаётся безнаказанной. В 2001 году компания Intel решает перейти к выпуску 64-битных процессоров, однако, являясь безоговорочным лидером в производстве процессоров для потребительского сегмента, принимает решение расширить своё влияние и делает ставку на сервера. Процессоры Itanium, выпущенные Intel, существенным образом отличаются от их классических процессоров и… с треском проваливаются!

Этой оплошностью тут же воспользовалась их главный конкурент AMD, которая к тому времени уже буквально наступала на пятки Intel и даже успела в августе 2000 года опубликовать своё видение того, какими должны быть 64-битные процессоры. Инженеры AMD не стали вносить радикальных изменений, а лишь немного дополнили то, что уже было в 32-битных процессорах Intel, нарастив разрядность до 64 бит, и сделали небольшую работу над ошибками.

Может показаться, что для системы типов Delphi всё складывается идеально, но… В 64-битных процессорах, разработанных AMD (а Intel позже вынуждена была принять их разработку), размер данных по умолчанию для большинства команд остался… 32-битным! Команды же, работающие с «родным» для этих процессоров размером в 64 бита, записываются длиннее, что приводит к увеличению объёма кода, а это в свою очередь отрицательно сказывается на производительности. Сложилась уникальная ситуация: в зависимости от решаемой задачи, модели процессора и фазы луны более эффективным может оказаться как 32-, так и 64-битный целочисленный тип данных.

Ещё одной ложкой дёгтя стало то, что за прошедшие годы Delphi стала поддерживать разработку программ не только для Windows. А ведь то, какие типы данных используются при взаимодействии программы с операционной системой, тоже накладывает отпечаток на эффективность типов данных. Ведь даже у людей переговоры проходят намного эффективнее, когда для обоих собеседников родным является один и тот же язык.

Как обычно, технический прогресс не щадит никого и о том, как со всеми этими нюансами будут разбираться новички, никто не думает.

Разработчики Delphi, скорее всего, не жалели резких слов в адрес инженеров AMD и разработчиков операционных систем, но в последних версиях Delphi имеются следующие целочисленные типы:

Название типа		Диапазон значений	Размер
Беззнаковые фиксированного размера			
Byte		$0..2^8 - 1$	1 байт
Word		$0..2^{16} - 1$	2 байта
Cardinal		$0..2^{32} - 1$	4 байта
Uint64		$0..2^{64} - 1$	8 байт
FixedUInt		$0..2^{32} - 1$	4 байта
Знаковые фиксированного размера			
ShortInt		$-2^7..2^7 - 1$	1 байт
SmallInt		$-2^{15}..2^{15} - 1$	2 байта
Integer		$-2^{31}..2^{31} - 1$	4 байта
Int64		$-2^{63}..2^{63} - 1$	8 байт
FixedInt		$-2^{31}..2^{31} - 1$	4 байта
Беззнаковые с размером, зависящим от платформы			
NativeUInt	32-битные платформы	$0..2^{32} - 1$	4 байта
	64-битные платформы	$0..2^{64} - 1$	8 байт
LongWord	32-битные платформы и 64-битные Windows	$0..2^{32} - 1$	4 байта
	64-битные POSIX-платформы	$0..2^{64} - 1$	8 байт
Знаковые с размером, зависящим от платформы			
NativeInt	32-битные платформы	$-2^{31}..2^{31} - 1$	4 байта
	64-битные платформы	$-2^{63}..2^{63} - 1$	8 байт
LongInt	32-битные платформы и 64-битные Windows	$-2^{31}..2^{31} - 1$	4 байта
	64-битные POSIX-платформы	$-2^{63}..2^{63} - 1$	8 байт

Как видим, концепцию пришлось слегка изменить, но большинство типов по-прежнему устояло. LongInt и LongWord пришлось скорректировать в соответствии со сложившейся практикой их использования, а Integer и Cardinal сделать всё же фиксированного размера.

Чтобы в следующий раз ураган нововведений не смёл всю цивилизацию, были установлены защитные сооружения. Во-первых, ввели типы FixedInt и FixedUInt, которые уж точно не поменяют размеры, даже если байт станет в два раза больше. Во-вторых, для «родных» размеров ввели имена NativeInt и NativeUInt, подчёркивающие их назначение.

Резервной системой защиты от чрезвычайных ситуаций стали типы Int8, Int16, Int32, Int64, Uint8, Uint16, Uint32 и Uint64, имеющие фиксированный размер, закреплённый в их названиях. Эти типы должны устоять, даже если придётся изменить размеры FixedInt и FixedUInt. Они не являются встроенными в язык, а всего лишь объявлены как эквивалентные соответствующим встроенным типам. Подобные типы данных программисты на Delphi могли самостоятельно создавать и раньше, но разработчики Delphi решили стандартизировать их имена, чтобы уж наверняка. В конце концов, куда надёжнее не полагаться на то, что люди смогут сами собрать аптечку на случай ядерной войны, а выдавать готовые комплекты единого образца.

* * *

— Благодарю за внимание! — подытожил Ви.

Вождь, до того всеми силами пытавшийся удержать в голове свалившийся на него поток информации, облегчённо выдохнул, закрыл глаза и лёг на спину, раскинув руки:

— Сколько времени тебе нужно, чтобы решить проблему с подсчётом коз?

— Пары дней должно хватить, бухгалтеры племени вроде бы смышлёные малые, обучим!

— Действуй, жирнермэр! Даю добро.

— Будет исполнено, гражданин начальник! — снова отчеканил Ви и вприпрыжку помчался к бухгалтерам племени.

— Послал бог соплеменничка, — пробормотал вождь, глядя вслед удаляющемуся Ви. — И чем же это ты в этом своём будущем так провинился, что тебя именно к нам забросили?..