ТУБ №14:
Понятие типа данных

Самые везучие люди в этом мире — это математики! Вот надо им, к примеру, записать Очень Большое Число: берут себе листок бумаги и пишут, и никаких проблем. Не поместилось — взяли ещё один листок и дальше пишут. Прямо-таки бесконечные объёмы памяти! Записали одно число — то же самое с другим сделали. Компьютерщики в этом вопросе, конечно, полные неудачники: мало того, что оперативная память не бесконечная, так ещё и хранить в ней нужно всё и сразу, и данные программы, и её машинный код.

Представьте себе камеры хранения в магазине. В каких-то магазинах их больше, в каких-то — меньше, но всегда их количество конечно и в них можно размещать какие-то вещи. С битами и байтами в памяти всё точно так же, с одним отличием: биты и байты не бывают пустыми.

Что такое бит в оперативной памяти? Это небольшое устройство, которое всегда находится в одном из двух состояний: одно из них считается единичным, другое — нулевым. Специалисты по аппаратной части должны возмутиться и заявить, что там, на уровне проводов и электронов, всё не так однозначно, но когда программа попытается прочитать данные из памяти, там всегда будут комбинации из нулей и единиц.

В один бит можно записать информацию о выборе одного из двух предметов, явлений или значений: одному поставить в соответствие 0, другому — 1. Этого, конечно, маловато. Зато если взять два бита, количество вещей, поддающихся подсчёту (или нумерации) возрастает вдвое. Группу из 8 бит принято называть байтом. Количество бит было таким не всегда, но на сегодняшний день в подавляющем большинстве устройств используется именно такая группировка. В таком количестве бит можно записать 256 различных комбинаций нулей и единиц — не заоблачно много, но уже достаточно для многих задач.

Часто говорят, что байт — это единица адресации памяти. Чтобы понять, что это означает, представьте себе камеры хранения в магазине. Они пронумерованы и в каждой из них можно хранить некоторое ограниченное количество вещей. Если Вам нужно будет попросить кого-нибудь забрать одну из оставленных в камере хранения сумок, придётся назвать номер ячейки, — это и будет аналогом адреса байта в памяти.

Точно так же, как один и тот же посетитель магазина может занять только целое количество ячеек, любой элемент данных в программе занимает целое количество байт. Что будет внутри этих байтов, зависит от самих данных, и с этим связано понятие типа данных.

Под типом данных понимают три составляющие:

* множество значений;
* свойства этих значений;
* множество операций над этими значениями.

В языках программирования понятия типа вводится для нескольких целей. Во-первых, с названными тремя составляющими тесно связан способ записи данных соответствующего типа в памяти компьютера. Возьмём, например, фортепиано…

Попробуем описать тип данных, значениями которого будут клавиши фортепиано. В наше время у этого инструмента, как правило, 88 клавиш, из которых примерно 36 — чёрные, 52 — белые и ещё одна — та, на которую Вы всё время ошибаетесь при подсчёте. Эти 88 клавиш и будут множеством значений нашего типа.

Определившись с количеством различных значений, которые будут представляться типом данных, можно начинать говорить о том, сколько памяти потребуется для записи значения такого типа (или, другими словами, какого размера будут переменные этого типа). Одним битом можно записать только одно из двух значений: либо 0, либо 1. Если мы возьмём два бита, получится уже 4 различных значения: 00, 01, 10 и 11 — и этого, очевидно, тоже будет маловато. Три бита дают уже 8 различных значений: 000, 001, 010, 011, 100, 101, 110, 111.

Нетрудно заметить, что каждый новый бит удваивает количество комбинаций, т.е. $N = 2^k$, где k — количество бит, а N — количество различных комбинаций из нулей и единиц. Для 6 бит получим 64 различных комбинации, и этого всё ещё недостаточно. 7 бит дадут 128 комбинаций, и этого более чем достаточно: 88 комбинациям можно поставить в соответствие имеющиеся клавиши, а остальные 40 просто не использовать или, например, притвориться, что клавиш чуть больше, чем есть на самом деле. А что, а вдруг?

Как видим, от того, сколько различных значений входит в тип данных, зависит минимальное количество памяти, которое придётся отвести для хранения значений этого типа.

Внимательный читатель заметит, что объём памяти, выделяемой для переменной, всегда выражается целым количеством байт, так что хоть фортепиано, хоть дудочка — на практике имеем минимум 256 комбинаций, а если бы различных клавиш было больше 256, пришлось бы задействовать сразу 2 байта, а это 2^{16} = 65536 различных значений.

Над значениями нашего типа данных могут быть определены некоторые операции. Например, в музыке есть такое понятие, как интервал — расстояние между двумя звуками. При вычислении расстояния учитываются и чёрные, и белые клавиши. Расстояние между звуками двух соседних клавиш фортепиано — полутон, расстояние между звуками клавиш, расположенных через одну, — тон и т.д. Если два звука играют одновременно или последовательно один за другим, то от расстояния между ними во многом зависит их восприятие человеком: одни интервалы звучат напряжённо и тревожно, другие — более спокойно.

Исходя из этого над значениями нашего типа данных можно было бы определить, например, операцию нахождения расстояния между двумя значениями. Для значений `x` и `y` типа «Клавиша фортепиано» результатом такой операции могло бы быть, например, число полутонов.

У значений придуманного нами типа данных есть также одно интересное свойство — свойство перенумерованности. Говорят, что «значения типа данных обладают свойством перенумерованности», если между значениями этого типа данных установлен линейный порядок, т.е. все значения этого типа можно расположить в определённом порядке и для любых двух значений однозначно ответить на вопрос, какой из них предшествует другому. В нашем случае этот порядок определяется расположением соответствующих клавиш на клавиатуре.

Вторая цель, с которой во многие языки программирования вводят понятие типов данных — это обнаружение логических ошибок. Например, если у нас в программе есть тип данных для клавиш фортепиано и тип данных для зарплаты сотрудников, то сложение клавиши с зарплатой скорее всего не имеет никакого практического смысла, а попытка выполнить такую операцию — почти наверняка ошибка программиста. Может быть, «820 рублей и фа-диез второй октавы» — это и неплохое название для книги о музыкантах-контрабандистах, но в банкомате такое снять не получится.

Есть языки программирования, для которых клавиши фортепиано могут быть свободно конвертируемой валютой и, скажем, при сложении с рублями автоматически обмениваться на чёрном рынке на рубли (мы следим за тобой, JavaScript!). Никаких сообщений об ошибках при этом не выдаётся. Справедливости ради, это может быть весьма удобно, если Вы постоянно испытываете острую потребность распродать фортепиано на запчасти.

В языках программирования наподобие Delphi, ориентированных на то, чтобы разработанная программа работала максимально правильно и надёжно, компилятор использует информацию о типах данных, чтобы проверить правильность действий над этими данными, и если в программе имеются какие-то несоответствия, компилятор укажет на это, а программа не будет скомпилирована. В результате многие логические ошибки в программе удаётся обнаружить ещё до запуска программы, причём автоматически. А разве не замечательно, когда рутинную работу кто-то делает за вас?

Дополнительные вопросы

1. Предложите такой способ записи информации о клавише фортепиано, чтобы для заданных двух клавиш было легко определять интервал между ними.

2. Предложите такой способ записи информации о клавише фортепиано, чтобы для заданной клавиши было легко определить её цвет.

3. Предложите способ записи музыкального произведения для фортепиано. Учтите, что при исполнении большинства музыкальных произведений одновременно может быть нажато несколько клавиш.