Типы и структуры данных

Информационные технологии
в профессиональной деятельности

Типы и структуры данных

Концепция типов данных развилась в языках программирования высокого уровня как естественное отражение того факта, что обрабатываемые программой двоичные данные могут иметь различные множества допустимых значений, храниться в памяти компьютера различным образом, занимать различные объемы памяти и обрабатываться с помощью различных команд процессора.

Особенности реализации понятия «тип данных» в современном программировании

Как правило, типы в языках программирования не всегда строго соответствуют подобным типам в математике. Например, тип данных "целое число" всех языков программирования не соответствует принятому в математике понятию целого числа, в математике указанный тип не имеет ограничений ни сверху, ни снизу (не является перечислимым), а в языках программирования эти ограничения есть. Как правило, в языках и системах имеется множество целых типов, отличающихся допустимым диапазоном значений (определяемым объемом занимаемой памяти).

Базовые типы данных языков программирования

Каждый язык программирования поддерживает один или несколько встроенных типов данных (базовых типов), кроме того, развитые языки программирования предоставляют программисту возможность описывать собственные типы данных, комбинируя или расширяя существующие.

Принято различать следующие основные типы данных:

Простые

Представляют собой базовые типы, характеризующиеся объемом памяти, выделяемой под переменную или константу данного типа, диапазоном значений, которые может принимать переменная и способом ее обработки машиной. Подробнее основные простые типы данных будут рассмотрены далее.

Перечисляемый тип

Может хранить только те значения, которые прямо указаны (перечислены) в его описании. Чаще всего значения перечислимого типа представляют собой лишь удобные для человека обозначения (например, названия дней недели), а компьютер интерпретирует их как целочисленные значения (например, названию дня недели соответствует его номер). В ряде языков определены также типы-диапазоны, которые могут принимать лишь ограниченный набор числовых значений (например, тип данных для номера дня в месяце может принимать значения от 1 до 31 включительно). Диапазоны удобны тем, что могут обеспечивать встроенный контроль значений переменных от переполнения. Обычно диапазоны записываются в следующем виде: 1..31

Числовые типы

Хранят числа, к которым могут применяться обычные арифметические операции. К числовым типам относятся:

Целочисленные типы данных

Целочисленные типы записываются со знаком + или -, или без знака, по обычным арифметическим правилам. Различают целочисленные типы со знаком, которые могут принимать как положительные, так и отрицательные значения (обычные названия типов – Integer или Int, Long Int); и без знака, которые могут принимать только неотрицательные значения (Word).

Вещественные типы данных

Вещественные типы могут записываться в одной из двух форм:

- обычная запись: 2.5 -3.14 2. В большинстве языков программирования целая часть отделяется от дробной символом точки;
- экспоненциальная ("научная")запись: в этой форме вещественное число представляется в виде m*10p, где m - мантисса или основание числа, принимающее значение 0.1 ≤|m|≤1, p - порядок числа, заданный целочисленной константой. Действительно, любое вещественное число можно представить в экспоненциальной форме: -153.5 -0.1535*103 99.005 0.99005*102

Вещественные типы данных обычной и двойной точности

В большинстве языков различаются вещественные типы обычной (Real или Float) и двойной (Double) точности. В последнем случае под число выделяется больший объем памяти и сохраняется большее число знаков в дробной части.

Символьный тип данных

Символьный тип (char). Хранит код одного символа. Могут использоваться различные кодировки. Во многих языках величины символьного типа записываются как символ, заключенный в апострофы или двойные кавычки. В разных языках под типом char может пониматься:

- набор печатных символов из алфавита, зафиксированного в описании языка (для большинства языков англоязычного происхождения этот алфавит соответствует кодовому набору ASCII);
- произвольная комбинация нулей и единиц, размещаемых в одном байте.

Логический тип данных

Логический тип (bool, boolean, logical). Имеет два значения: истина и ложь. К величинам этого типа могут применяться логические операции. Как правило, данный тип используется в операторах ветвления и циклах. В некоторых языках является подтипом числового типа, при этом ложь (false) равна 0, а истина (true) равна единице или любому ненулевому значению. Несмотря на то, что для хранения значений этого типа теоретически достаточно одного бита, обычно в реализациях переменные этого типа занимают один байт памяти.

Для всех типов данных, для которых определены операции сравнения, определены также и правила, по которым эти операции сравнения вырабатывают булевские значения. Над булевскими значениями возможны операции конъюнкции или логического умножения (обозначается & или AND), дизъюнкции или логического сложения (обозначается | или OR) и отрицания (~ или NOT).

Множество (set)

В основном определение типа совпадает с обычным математическим понятием множества. Допустимы стандартные операции с множествами и проверка на принадлежность элемента множеству. В некоторых языках рассматривается как составной тип.

Сложные типы

Массив (array). Массивом называют упорядоченный набор однотипных переменных (элементов). Каждый элемент имеет целочисленный порядковый номер, называемый индексом. Число элементов в массиве называют его размерностью. Массивы используются там, где нужно обработать сразу несколько переменных одного типа - например, оценки всех 20 студентов группы или координаты 10 точек на плоскости. Строку текста можно рассматривать как массив символов, а текст на странице - как массив строк. Пример одномерного массива — вектор в математике, двумерный массив — матрица.

Хранит строку символов. Аналогом сложения в строковой алгебре является конкатенация (прибавление одной строки в конец другой строки, сцепление строк). В языках, близких к бинарному представлению данных, чаще рассматривается как массив символов, в языках более высокой абстракции зачастую выделяется в качестве простого.

Запись (структура)

Как и массивы, записи относятся к составным типам данных. Запись состоит из фиксированного числа элементов, называемых полями. Каждое поле представляет собой объект простого типа данных - как правило, строку или число. Существенно то, что в одну запись могут входить поля различного типа, в отличие от массива, все элементы которого однотипны.

Файловый тип (file)

Хранит только однотипные значения, доступ к которым осуществляется только последовательно (файл с произвольным доступом, включенный в некоторые системы программирования, фактически является неявным массивом).

Класс (class)

Используется в объектно-ориентированном программировании

Указатель (Pointer)

Хранит адрес в памяти компьютера, указывающий на какую-либо информацию, как правило — указатель на переменную или массив.

Ссылка

Объект, указывающий на определенные данные, но не хранящий их.

Основные преимущества от использования типов данных

- Надежность
- Стандартизация
- Документирование

Надежность

Типы данных защищают от трех видов ошибок:

1. Некорректное присваивание. Пусть переменная объявлена как имеющая числовой тип. Тогда попытка присвоить ей символьное или какое-либо другое значение в случае статической типизации приведет к ошибке компиляции и не даст такой программе запуститься. В случае динамической типизации код программы перед выполнением потенциально опасного действия сравнит типы данных переменной и значения и также выдаст ошибку. Все это позволяет избежать неправильной работы и "падения" программы.

Типы данных защищают от трех видов ошибок:

2. Некорректная операция. Позволяет избежать попыток применения выражений вида "Строка" + 1. Поскольку все переменные в памяти хранятся как наборы битов, то при отсутствии типов подобная операция была бы выполнима (и могла дать странный результат). С использованием типов такие ошибки отсекаются.

Типы данных защищают от трех видов ошибок:

3. Некорректная передача параметров. Если функция "синус" ожидает, что ей будет передан числовой аргумент, то передача ей в качестве параметра строки может иметь непредсказуемые последствия. При помощи контроля типов такие ошибки также отсекаются на этапе компиляции.

Стандартизация

Благодаря соглашениям о типах, поддерживаемых большинством систем программирования, сложилась ситуация, когда программисты могут быстро менять свои рабочие инструменты, а программы не требуют больших переделок при переносе исходных текстов в другую среду. К сожалению, стандартизации по универсальным типам данных еще есть куда развиваться.

Документирование

Использование того или другого типа данных объясняет намерения программиста. Например, типы enum (перечислимый) и bool (логический) в конкретном языке программирования не обязательны и вместо них может быть int (целочисленный) — но оба они означают, что перед нами не целая величина, а одно из нескольких предопределенных значений.