УДК 734

Колбаса Е.

ПРЕОБРАЗОВАНИЯ И ПРИВЕДЕНИЯ ТИПОВ ДАННЫХ В C/C++

БНТУ, Минск
Научный руководитель: Дробыш А.А.

Основная цель любой программы состоит в обработке данных. Данные с которыми работают машинные команды, хранятся в оперативной памяти. Компилятору для формирования команд необходимо точно знать, сколько места занимают данные, как именно они закодированы и какие действия с ними можно выполнять. Все это задается при описании данных с помощью типа.

В C++ различают явное и неявное преобразование типов данных. *Неявное* преобразование типов данных выполняет компилятор C++, а *явное* преобразование данных выполняет сам программист.

В выражение могут водить операнды различных типов. Если операнды имеют одинаковый тип, то результат операции будет иметь тот же тип. Если операнды разного типа, перед вычислениями выполняется неявное преобразование типов. Обычно более короткие типы приводятся к более длинным, что обеспечивает сохранение значимости и точности.

Для стандартных арифметических типов направление неявных преобразований можно представить в виде цепочки:

(char, short) -> int -> unsigned int -> long -> unsigned long -> long long -> unsigned long long -> float -> double -> long double

Это не значит что преобразование int в double выполняется строго последовательно, как в цепочке. Просто менее объемный тип преобразуется в более объемный. Типы char и short, как знаковые, так и без знаковые перед выполнением операции обязательно преобразуются в int (это отмечено скобками). Кроме того, к типу int по умолчанию приводится перечисляемый тип enum.

Типом результата выражения обычно является самый «объемный», из типов операндов.

Широкие символьные типы приводятся к минимальному из целых типов, который включает все их допустимые значения. Однако в выражениях с операцией присваивания встречается и обратный вариант, когда более «объемный» тип присваивается менее «объемному», например float присваивается типу int. Подобные преобразования связаны с потерей информации (отбрасывается дробная часть), поэтому называются *сужающими приведениями* (*преобразованиями*). Компилятор обязательно предупреждает о таких операциях.

При преобразовании других типов данных в тип bool, любое не равное нулю значение трактуется как true, а нуль как false. При обратных преобразованиях, true преобразуется в целочисленную константу 1, а значение false – в 0.

Операция явного преобразования типа в стиле С может записываться в двух эквивалентных формах:

- (тип) выражение;
- выражение (тип).

Явное преобразование в стиле С оставлено в С++ только для совместимости. Использовать которое не рекомендуется так, как оно слишком универсально, а потому чревато плохо диагностируемыми ошибками.

Для выполнения явных преобразований типа в С++ существует целая группа операций:

- const_cast<тип> (объект);
- dynamic_cast<тип> (объект);
- reinterpret_cast<тип> (объект);
- static_cast<тип> (объект).

Оператор const_cast используется для явного переопределения модификаторов const и/или volatile (Квалификатор volatile позволяет предоставлять доступ к областям памяти, которые используются асинхронными процессами, например обработчиками прерываний.). Новый тип должен совпадать

с исходным типом, за исключением изменения его атрибутов const или volatile. Чаще всего оператор const_cast используется для удаления атрибута const.

Оператор dynamic_cast проверяет законность выполнения заданной операции приведения типа. Если такую операцию выполнить нельзя, то выражение устанавливается равным нулю.

Оператор static_cast выполняет не полиморфное приведение типов. Например, его можно использовать для приведения указателя на базовый класс к типу указателя на производный класс. Его можно также использовать для любого стандартного преобразования. При этом никакие проверки во время работы программы не выполняются.

Оператор reinterpret_cast переводит один тип в совершенно другой. Например, его можно использовать для перевода указателя в целый тип. Оператор reinterpret_cast следует использовать для перевода типов указателей, которые несовместимы по своей природе.

Однако следует помнить, что только оператор const_cast может освободить от «обета постоянства», то есть ни один из остальных операторов этой группы (dynamic_cast, static_cast, reinterpret_cast) не может «снять» с объекта атрибут const.

УДК 271

Колбаса Е., Выскварко В.

ПРОБЛЕМЫ ИНФОРМАТИЗАЦИИ ОБЩЕСТВА

БНТУ, Минск
Научный руководитель: Липень С.Г.

Информатизация призвана стать основанием кардинальной трансформации качества и уровня жизнедеятельности человека. Информатика как единство науки, техники и индустрии воздействует на общество, стимулируя процессы компьютеризации, интеллектуализации и т.д., что вызывает далеко не однозначные социальные последствия.