Отличие C от C++

Тема 2.1 От «C» к C++

Существуют два диаметрально противоположенных, но одинаково распространенных мнения, которые можно выразить как "C++ это C с классами" и "C++ и C --- разные языки программирования". В общем-то, не важно, какого мнения придерживаться, но интересно иное - в каких случаях какой из этих языков (или вариантов языка) предпочтительнее. Однако стоит все же рассмотреть некоторые принципиальные различия:

Различие 1. Объекты

- В C++ появились классы и объекты. Технически класс C++ - это тип структуры в C, а объект - переменная такого типа. Разница только в том, что в C++ есть еще модификаторы доступа и полями могут быть не только данные, но и функции (функции-методы).
- Функция-метод - это обычная функция C, у которой первый параметр - это указатель на структуру, данные которой она обрабатывает: this. Если сравнить, как выглядят функции-методы в C++ и функции с параметром-указателем на структуру в C, то мы обнаружим, что всего лишь изменилась форма записи. В C++ получается короче, так как this и имя типа во многих случаях писать не обязательно (подразумевается по умолчанию).
- Модификаторы доступа - это слова public, private и protected. В C программисту приходилось полагаться лишь на свою внимательность: public - значит с этими полями делаю, что хочу; private - значит к этим полям обращаюсь только с помощью методов этой структуры; protected - то же, что public, но еще можно обращаться из методов унаследованных структур.

Различие 2. Наследование

- То, что в C++ - наследование, в C - это просто структура в структуре. При программировании в стиле C++ применяются такие ключевые слова, как "класс Circle порожден от класса Point" или "класс Circle наследуется от класса Point и является производным от него". На практике все это заключается в том, что структура Point - это первое поле структуры Circle.
- При этом реальных усовершенствований два. Первое - поля Point считаются так же и полями Circle, в результате доступ к ним записывается короче, чем в C. Второе - в обоих структурах можно иметь функции-методы, у которых имена совпадают с точностью до имени структуры. Например, Point::paint и Circle::paint . Следствие - не надо изобретать имена вроде Point_paint и Circle_paint, как это было в C, а префиксы Point:: и Circle:: в большинстве случаев можно опускать.

Различие 3. new и delete

В C++ появились две новые операции: new и delete. В первую очередь это - сокращения для распространенных вызовов функций malloc и free:

В стиле C:

```
Point *p = (Point*)
malloc(sizeof(Point));
free(p);
```

В стиле C++:

```
Point *p = new Point;
delete p;
```

При вызове new автоматически вызывается конструктор, а при вызове delete - деструктор (см. следующий пункт). Так что нововведение можно описать формулой: new = malloc + конструктор, delete = free + деструктор.

Различие 4. Виртуальные функции

При программировании на C часто бывает так, что имеется несколько вариантов одной и той же структуры, для которых есть аналогичные функции. Например, есть структура, описывающая точку (Point) и структура, описывающая окружность (Circle). Для них обоих часто приходится выполнять операцию рисования (point). Так что, если у есть блок данных, где перемешаны точки, окружности и прочие графические примитивы, то перед программистом стоит задача быстро вызвать для каждого из них свою функцию рисования.

Обычное решение - построить таблицу соответствия "вариант структуры - функция". Затем берется очередной примитив, определяется его тип, и по таблице вызывается нужная функция. В C этот метод применять довольно нудно из-за того, что во-первых, надо строить эту таблицу, а во-вторых, внутри структур заводить поле, сигнализирующее о том, какого она типа, и следить за тем, чтобы это поле содержало правильное значение.

В C++ всем этим занимается компилятор: достаточно обозначить функцию-метод как virtual, и для всех одноименных функций будет создана таблица и поле типа, за которыми следить будет опять-таки компилятор. Программисту останется только пользоваться ими: при попытке вызвать функцию с таким именем, будет вызвана одна из серии одноименных функций в зависимости от типа структуры.

Вывод

Обычно к разрабатываемому программному обеспечению предъявляются какие-то требования, связанные с его качеством. Эти требования не могут быть настолько жесткими, чтобы совсем исключать вероятность наличия ошибок в программе, но чем они сильнее, тем лучше использовать старый и проверенный во многих разработках C. В остальных же проектах может быть обратная ситуация: сложность разработки на C вызовет увеличение сроков, связанное с отладкой и выявлением ошибок в коде у самих программистов, поэтому в данной ситуации гуманнее использовать C++. Именно поэтому программисту нужно знать особенности каждого языка и делать правильный выбор на основе поставленной задачи.