Семантика перемещения — конспект темы

Copy elision и NRVO

Copy elision и NRVO — механизмы оптимизации, которые применяет компилятор, чтобы избежать лишних действий.

Copy elision защищает программу от лишних вызовов конструктора копирования:

- при возврате из функции временного объекта;
- при инициализации объекта временным объектом.

NRVO (Named Return Value Optimization) помогает избегать лишних копирований. Как видно из названия, речь идёт о возвращении из функции некой именованной переменной.

Функция move и другие перемещения

Семантика перемещения, или **move-семантика** позволяет не размножать объекты там, где не нужно.

Чтобы сказать компилятору, что объект в будущем не понадобится, и его можно не копировать, а перенести, используют функцию `move`. Находится она в библиотеке `<utility>`:

```
#include <utility>
...
int main() {
    vector<WhiteElephant> crowd_of_elephants;
    WhiteElephant heavy_elephant = BuyElephant();
    SmallWalk(heavy_elephant);
    // я отпускаю слона на волю
    crowd_of_elephants.push_back(move(heavy_elephant));
}
```

Копировать нельзя переместить

При копировании вызываются специальные методы — конструктор копирования или оператор присваивания. Если вместо копирования надо осуществить перемещение, компилятору нужны другие методы.

Перед вами конструктор перемещения и перемещающий оператор присваивания:

```
NoncopyableInt(NoncopyableInt&&) = default;
NoncopyableInt& operator=(NoncopyableInt&&) = default;
```

Специальное слово `default` говорит, что компилятор должен сам решить, как будет происходить перемещение.

rvalue-ссылка относится к объекту, к адресу которого вы не можете получить прямой доступ. Такие ссылки бывают на временные объекты.

Тайные техники передачи аргументов

В C++ функции с одинаковыми названиями компилятор различает благодаря списку аргументов. При вызове с аргументом компилятор ищет функцию с нужным именем и максимально близким типом аргумента.

Контейнер `array` поддерживает перемещение, но оно не такое эффективное, как у вектора. В векторе достаточно переместить указатель на данные.

Вектор неперемещаемых элементов переместить можно. Но `array` не хранит указатель на данные, и для реализации перемещения нужно переместить каждый из элементов. Отсюда следуют два вывода:

- Для перемещения `array` его элементы должны быть перемещаемыми.
- Сложность перемещения `array` — линейная от количества элементов, что значительно хуже константной сложности перемещения вектора.

Выбирая между передачей по значению, по константной ссылке и по rvalue-ссылке, программист будет руководствоваться двумя принципами:

- эффективностью кода или минимальным количеством копирований;
- удобством для того, кто будет этот метод вызывать.

Когда программист пишет своё приложение, он знает, с какими объектами собирается работать. Если объекты перемещаемые, стоит выбрать передачу

аргумента по значению. Если неперемещаемые — по константной ссылке. Такой механизм позволяет сохранить эффективность кода и избежать дублирования.