Особенности явного вызова деструктора

Явный вызов деструктора требуется в достаточно редких ситуациях (см. раздел 12.4. Стандарта C++). Одним из примеров, где требуется явный вызов деструктора, является размещение объекта по конкретному адресу в памяти посредством new-выражения с фразой размещения. Такое использование явного размещения и явного разрушения значения объекта может быть нужным для управления выделенными аппаратными ресурсами и для **написания** программных средств управления памятью. Например:

```
void* operator new (size_t, void* p) { return p; }

struct A {
    int a;
     A(int a=0);
    ~A();
};
/* ... */
void g(A* p); …
void h() {
    //   редкий случай использования для специальных нужд:
    char* buf = new char[sizeof(A)];
    A* p = new(buf) A(111); // используется и инициализируется buf[]
    g(p);
    p->A::~A();   // очистка значения объекта явным вызовом деструктора
    delete[]buf; // освобождение памяти, которую занимал объект
}
```

Как только для объекта запущен деструктор, этот объект перестает существовать. Если деструктор запущен для объекта, чьё время жизни уже закончилось, то поведение программы не определено (*behavior is undefined*).

Пример: если для автоматического объекта, описанного в блоке, явно вызван деструктор, после чего произойдёт выход из этого блока таким образом, который обычно подразумевает неявный запуск процесса разрушения объекта (например, выход из блока по закрывающей фигурной скобке, или свертка стека при выбросе исключения), то поведение программы не определено.

Нотация явного вызова деструктора может использоваться для имени любого скалярного типа (см. Стандарт, раздел 5.2.4). Это допущение позволяет писать код, не зная, существует ли для данного типа деструктор. Например:

```
    typedef unsigned U; U* p;
    // ... p->U::~U();
```

Явный вызов деструктора всегда должен записываться с использованием операции доступа к члену класса (с помощью точки или стрелки), например *y.~Y()*. Выражение *~Y()* не является вызовом деструктора, это операция тильда, применяемая к временному объекту, созданному с помощью конструктора умолчания. Вот пример:

```
class  Y {
public:
       Y();
   virtual ~Y();
   Y& operator~();
   void g();
};

void Y::g() {
    Y bb = Y(); // конструирование и инициализация объекта  bb
    ~bb;        // операция дополнения, применённая к  bb
    bb.~Y();    // разрушение  bb
    this->~Y(); // разрушение  *this
    ~Y();       // создаём объект класса  Y  и берём от него дополнение
                // это не вызов деструктора

}
```