## Constructors Calling Other Constructors

The use of the name of the class in the initializer list of that class's constructor enables delegating initialization to another constructor of the same class.

### Description

A **delegating constructor** is a constructor of a **user-defined type (UDT)** — i.e., **class**, **struct**, or **union** — that invokes another constructor defined for the same UDT as part of its initialization of an object of that type. The syntax for invoking another constructor is to specify the name of the type as the only element in the **member initializer list**:

```
#include <string>  // std::string

struct S0
{
  int         d_i;
  std::string d_s;

  S0(int i)         : d_i(i)       {} // nondelegating constructor
  S0()              : S0(0)        {} // OK, delegates to S0(int)
  S0(const char* s) : S0(0), d_s(s) {} // Error, delegation must be on its own
};
```

Multiple delegating constructors can be chained together, one calling exactly one other, so long as cycles are avoided; see *Potential Pitfalls — Delegation cycles* on page 50. ~~Once a *target* — i.e., invoked via delegation — constructor returns, the body of the delegator is invoked;~~

```
#include <iostream>  // std::cout

struct S1
{
    S1(int, int)           { std::cout << 'a'; }
    S1(int)      : S1(0, 0) { std::cout << 'b'; }
    S1()         : S1(0)    { std::cout << 'c'; }
};

void f()
{
    S1 s;  // OK, prints "abc" to stdout
}
```