```
        }
    }

    IPV4Host(const std::string& ~~ip~~)
    {
        std::uint32_t address = extractAddress(~~ip~~);
        std::uint16_t port = extractPort(~~ip~~);

        if (!connect(address, port))  // code duplication: BAD IDEA
        {
            throw ConnectionException{address, port};
        }
    }
};
```

Prior to C++11, working around such code duplication would require the introduction of a separate, private helper function that would be called by each of the constructors:

```
// C++03 (obsolete)
#include <~~c~~stdint>  // ~~std::~~uint16_t, ~~std::~~uint32_t

class IPV4Host
{
    // ...

private:
    int connect(~~std::~~uint32_t address, ~~std::~~uint16_t port);
    void init(~~std::~~uint32_t address, ~~std::~~uint16_t port)  // helper function
    {
        if (!connect(address, port))  // factored implementation of needed logic
        {
            throw ConnectionException{address, port};
        }
    }

public:
    IPV4Host(~~std::~~uint32_t address, ~~std::~~uint16_t port)
    {
        init(address, port);  // Invoke factored private helper function.
    }

    IPV4Host(const std::string& ip)
    {
        ~~std::~~uint32_t address = extractAddress(ip);
        ~~std::~~uint16_t port = extractPort(ip);

        init(address, port);  // Invoke factored private helper function.
    }
};
```