Default `class`/`union` Member Initializers

Non**`static`** ~~class~~ **data members** might specify a default initializer, which is used ~~for~~ constructors that don't initialize the **member** explicitly.

Description

The traditional means for initializing non**`static`** data members and base class objects within a class is a constructor's **member initializer list**:

```
struct B
{
    int d_i;

    B(int i) : d_i(i) { }     // Initialize d_i with i.
};

struct D : B
{
    char d_c;

    D() : B(2), d_c('3') { }  // Initialize base B with 2 and d_c with '3'.
};
```

Starting with C++11, non**`static`** data members — except for **bit fields** — can also be initialized using a **default member initializer**, by using **copy initialization**, **copy list initialization**, or **direct list initialization**; see Section 2.1."Braced Init" on page 215:

```
struct S0
{
    int   d_i = 10;     // OK, uses copy initialization
    char  d_c = {'a'};  // OK, uses copy list initialization
    float d_f{2.f};     // OK, uses direct list initialization
};
```

Note that although **braced initialization** is supported, **direct initialization** with a parenthesized list is not:

```
struct S1
{
    char d_c('a');  // Error, invalid syntax
};
```