```
    int    y;      // size 4; alignment 16
    char   b[44];  // padding
    double z;      // size 8; alignment 64
    char   c[56];  // padding (optional)
};  // size 128; alignment 64
```

Again, if more than one alignment specifier pertains to a given data member, the strictest applicable alignment value is applied:

```
struct T4
{
    alignas(2) char
        c1 alignas(1),  // size 1; alignment 2
        c2 alignas(2),  // size 1; alignment 2
        c4 alignas(4);  // size 1; alignment 4
};                      // size 8; alignment 4
```

Strengthening the alignment of a user-defined type

The **alignas** specifier can also be used to specify alignment for UDTs, such as a **class**, **struct**, **union**, or **enum**. When specifying the alignment of a UDT, the **alignas** keyword is placed *after* the ~~type specifier~~ (e.g., **class**) and just before the name of the type (e.g., `C`):

```
class  alignas( 2) C { };  // OK, aligned on a  2-byte boundary; size = 2
struct alignas( 4) S { };  // OK, aligned on a  4-byte boundary; size = 4
union  alignas( 8) U { };  // OK, aligned on an 8-byte boundary; size = 8
~~enum   alignas(16) E { };  // OK, aligned on a 16-byte boundary; size = 4~~
```

Notice that, for each of **class**, **struct**, and **union** in the example above, the **sizeof** objects of that type increased to match the alignment~~; in the case of the **enum**, however, the size remains that of the default **underlying type**, e.g., 4 bytes, on the current platform. When **alignas** is applied to an enumeration `E`, the Standard does not indicate whether padding bytes are added to `E`'s object representation, affecting the result of **sizeof**(`E`)~~.[2]

Again, specifying an alignment that is less than what would be without the **alignas** specifier is ill formed:

```
struct alignas(2) T0 { int i; };
    // Error, alignment of T0 (2) is less than that of int (4).
struct alignas(1) T1 { C c; };
    // Error, alignment of T1 (1) is less than that of C (2).
```

[2]~~The implementation variance resulting from this lack of clarity in the Standard was captured in CWG issue 2354 (**miller17**). The outcome of the core issue was to completely remove permission for **alignas** to be applied to enumerations; see **iso18a**. Therefore, conforming implementations will eventually stop accepting the **alignas** specifier on enumerations in the future.~~