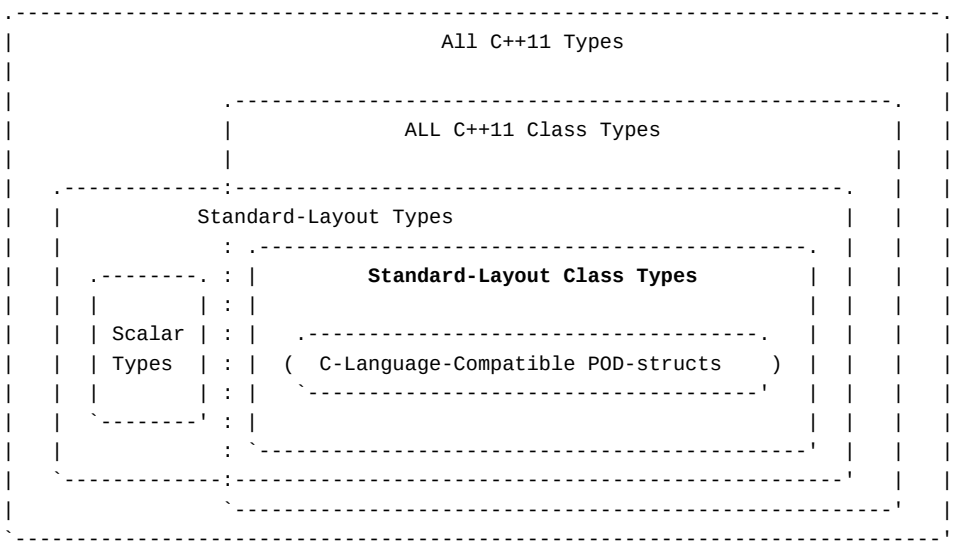
8. The type has no nonstatic data members or direct base classes that are not themselves of standard-layout type:

```
// Type                                    Is standard layout?
struct Y7 { int i; public:  int j; };  // yes, same access control
struct N7 { int i; private: int j; };  // no, not same access control
struct S7a { Y7 d; };                  // yes, standard-layout member
struct S7b { N7 d; };                  // no, non-standard-layout member
struct S7c : Y7 { };                   // yes, standard-layout base class
struct S7d : N7 { };                   // no, non-standard-layout base class
```

Standard-layout class special properties

Although standard-layout types can be classes or scalars, the C++ type category known as **standard-layout class types** identifies an important category that is a subset of other important categories — each affording its own specific set of supported properties:



Importantly, for a class type to be considered **standard layout**, it does not need to be trivial or look like a C type in any way:

```
#include <cstddef>   // std::size_t
class String  // non-trivial standard-layout class type
{
    char*       d_array_p;   // scalar (standard layout)
    std::size_t d_length;    //    "        "       "
    std::size_t d_capacity;  //    "        "       "
```