Second, a function or operator that returns an rvalue reference produces an *xvalue* when invoked:

```
int&& rf();  // rf() is an xvalue of type int.
S&& rg();    // rg() is an xvalue of type S.

S&& operator*(const S&, const S&);  // oddly defined operator

void testOperator()
{
    int i, j;
    i * j;  // i * j is a prvalue of type int.
    S a, b;
    a * b;  // a * b is an xvalue of type S.
}
```

Third, the Standard Library utility function `std::move` also produces *xvalues*, as it is nothing more than a function **defined** to return an rvalue reference to the type passed to it; see *The `std::move` utility* on page 731.

Finally, expressions that access subobjects of any non*lvalue* are *xvalues*, including **nonstatic data member** access, array subscripting, and dereferencing pointers to data members. Note that when any of these operations is applied to a *prvalue*, a temporary needs to be created from that *prvalue* to contain the subobject, so the subobject is an *xvalue*[7]:

```
struct C  // C() is a prvalue of type C.
{
    int d_i;
    int d_arr[5];
};

C&& h();               // h() is an xvalue of type C.
int C::* pd = &C::d_i;  // pointer to data member C::d_i

void testSubobjects()
{
    h().d_i;       // h().d_i is an xvalue of type int.
    C().d_i;       // C().d_i  "  "    "    "   "   "
    h().d_arr;     // h().d_arr is an xvalue of type int[5].
    C().d_arr;     // C().d_arr  "  "    "    "   "     "
    h().d_arr[0];  // h().d_arr[0] is an xvalue of type int.
    C().d_arr[0];  // C().d_arr[0]  "  "    "    "   "   "
```

[7]The identification of subobjects as *xvalues* rather than *prvalues* or, in some cases, *lvalues*, has been the subject of several core issues, all of which were accepted as **defect reports** between C++14 and C++20. Specifically, CWG issue 616 (**stroustrup07**) and CWG issue 1213 (**merrill10a**) deal with changes to the value categories of subobject expressions. Also note that compiler implementations of these clarifications took some time, with GCC not fully supporting them until GCC 9 (c. 2019).