Distinguish Between Pointers and References

References:

1. C++ Primer, 5th Edition, Stanley Lippman, Josee Lajoie, Barbara Moo.
2. The C++ Programming Language, 4th Edition, Bjarne Stroustrup.
3. Effective C++: 55 Specific Ways to Improve Your Programs and Designs, 3rd Edition, Scott Meyers.
4. More Effective C++: 35 New Ways to Improve Your Programs and Designs, Scott Meyers.

- Both pointers and references let you refer to other objects indirectly. How, then, do you decide when to use one and not the other?

- **There is no such thing as a null reference but there is such a thing as a null pointer.**
  - Because there is no such thing as a null reference, C++ requires that references be initialized:

```
// a reference must always refer to some object
string &rs;          // error! references must be initialized
string s(“hello”);
string &rs = s;    // okay - rs refers to s
```

  - Pointers are subject to no such restriction:

```
string *ps;     // uninitialized pointer: valid but risky
```

  - If you have a variable whose purpose is to refer to another object, but it is possible that there might not be an object to refer to, then that variable must be declared as a pointer.
  - On the other hand, if the variable must always refer to an object (no possibility that the variable is null), you should make the variable a reference.

- **Efficiency.**
  - The fact that there is no such thing as a null reference implies that it can be more efficient to use references rather than to use pointers.
  - That’s because there’s no need to test the validity of a reference before using it:

```
void print_int(const int &ri)
{
    std::cout << ri << std::endl;
}
```

  - Pointers, on the other hand, should generally be tested against null:

```
void print_int(const int *pi)
{
    if (pi == nullptr) {
        std::cout << *pi << std::endl;
    }
}
```

- **Reassignment.**
  - Another important difference between pointers and references is that pointers may be reassigned to refer to different objects.
    - A reference, however, always refers to the object with which it is initialized:

```
string s1("Tom");
string s2("Jane");
string &rs = s1;       // rs refers to s1
string *ps = &s1;      // ps refers to s1
rs = s2;               // rs still refers to s1, but s1's new value is "Jane"
ps = &s2;              // ps now points to s2; s1 is unchanged
```

- **In general:**
  - You should use a pointer whenever you need to take into account the possibility that there's nothing to refer to (in which case you set the pointer to null) or whenever you need to be able to refer to different things at different times (in which case you change where the pointer points).
  - You should use a reference whenever you know that there will always be an object to refer to and you also know that once you're referring to that object, you'll never want to refer to anything else.

- **operator[]**
  - There's one situation in which you should always use a reference - that's when implementing certain operators. The most common example is operator[]. This operator needs to return something that can be used as the target of an assignment:

```
std::vector<int> v(10);      // create an int vector of size 10
...
v[5] = 10;// the target of this assignment is the return value of operator[]
```

  - If operator[] returned a pointer, then we would need to write:

```
std::vector<int> v(10);      // create an int vector of size 10
...
*v[5] = 10;    // correct but it makes v look like a vector of pointers
```

- **Final thoughts:**
  - References, then, are the feature of choice when you know you have something to refer to, when you'll never want to refer to anything else, and when implementing operators whose syntactic requirements make the use of pointers undesirable.
  - In all other cases, use pointers.