Item 1: Distinguish between pointers and references

A reference must *always* refer to some object

char \*pc = 0; // set pointer to null

char& rc = \*pc; // make reference refer to dereferenced null pointer

Well, this is evil, pure and simple. The results are undefined.

Because a reference must refer to an object, C++ requires that references be initialized.

string& rs; // error! References must be initialized

string s("xyzzy");

string& rs = s; // okay, rs refers to s

Pointers are subject to no such restriction:

string \*ps; // uninitialized pointer valid but risky.

Nullability check is always issue for pointer but not for reference.

void printDouble(const double& rd)

{

cout << rd; // no need to test rd; it must refer to a double

}

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

void printDouble(const double \*pd)

{

if (pd) { // check for null pointer

cout << \*pd;

}

}

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("Nancy");

string s2("Clancy");

string& rs = s1; // rs refers to s1

string \*ps = &s1; // ps points to s1

rs = s2; // rs still refers to s1, but s1's value is now "Clancy"

ps = &s2; // ps now points to s2; s1 is unchanged

The most common example is operator[]. This operator typically needs to return something that can be used as the target of an assignment.

Int arr[10]={1,2,3,..}

Arr[3]=5;

Operator[] always returns the reference.

References are the feature of choice when you *know; you* have something to refer and will be neverwanted to refer to anything else. In all other cases, stick with pointers.

Item 2: Prefer C++-style casts.

There is a great difference, for example, between a cast that changes a pointer-to-const-object into a pointer-to-non-const-object (i.e., a cast that changes only the constness of an object) and a cast that changes a pointer-to-base-class-object into a pointer-to-derived-class-object (i.e., a cast that completely changes an object's type). Traditional C-style casts make no such distinctions. (This is hardly a surprise. C-style casts were designed for C, not C++.)