12.07.23

北島

Effective C++

Chapter 1:

Item 2: Prefer consts, enums and inlines to #defines

why use less #define?

1. may not get entered into the symbol table
2. not in symbol table, leading to crop up in a symbolic debugger.
3. reduce resource waste, because preprocessor may always make marco appear many times even it has existed

so we use const more than #define

But here is something to remember:

1. const char* const ps;

   but we prefer: const std::string ps("string");

   for a ptr to const string

2. in-class const; some one may use:

   static const double scd; (declartion)

and its definition is

const double ClassName::scd; (to use scd's add)

but we better use enum hack:

```
class ClassName {
  ---
  enum {scd=123};
  }
```

it more like #define, no address..

3. For function-like macros, prefer inline functions to #define.

Item 3: Use const whenever possible

Tips: Read const char const *p; like this

常量指针指向常量

in functions, when parameter is pointing to what is constant, below both is OK:

```
void f1(const int* pi);
void f2(int const* pi);
```

in STL:

const vector<int>::iterator iter;

iter is a const pointer, like: vector const *it;

vector<int>::const_iterator citer;

citer is a pointer to const, like: const vector *it;

```
class Rational { ... };
const Rational operator*(const Rational& lhs, const Rational& rhs);
```

this const is to avoid mistakes like:

```
if (a * b = c) ...                    // oops, meant to do a comparison!
```