2.4：const 限定符

有时候我们希望定义一种变量，其值不能改变，这时候需要用到 const。

① 初始化

Const 的类型须初始化。可用给一个对象去初始化另外一个对象，是不是 const 无所谓。

② 默认状态下，const 对象只在文件内有效。

如需在多个文件下使用，须用 extern const int x=0;

可在另外一个文件下对其进行声名即可： extern const int x;

③ Const 引用

可以把引动绑定到 const 对象上，称之为对常量的引用。对常量的引用不能更改其绑定的对象。

常量引用是对 const 的引用。

Int i=43;

Const int &r1=i; //正确，这是一个 const int &绑定到一个普通的 int 上.

④ 常量引用仅对其参与的操作做出了限定，其引用的是否为常量，并未做出限定，但是如果引用的是非常量对象，也不能通过常量引用的方式对其值做出改变，详见 P56。

⑤ Const 和指针

和引用一样，指针也可指向常量或非常量。

指向常量的指针不能用于修改其所指对象的值（这个对象可能是非常量）。

要想存放常量对象的地址，只能使用指向常量的指针。

因为指针本身是一个对象，所以可以把其定义为常量型指针，常量指针必须初始化，而且一旦初始化完成，其值就不能再改变，值是指存放在指针中的那个地址而非指针指向的对象的值。

Const double pi=3.14;

Const double *const pip=π //pip 是指向常量对象的常量指针。

⑥ 顶层 const 指针是否是常量与指针所指的对象是否是常量是一个独立的问题，如果指针本身是一个常量，认为其是一个顶层 const，如果其所指的对象是常量则认为其是一个底层 const。

⑦ 顶层 const 的拷贝不受限制，但是底层 const 的拷贝的对象必须具有相同的底层 const 资格。一般来说：非常量可以赋值给常量，反之则不行。

这里看的不是很懂，以后再返回来看吧。

⑧ Constexpr 和常量表达式

常量表达式指的是指值不会改变且在编译过程中就能确定值的表达式，则面值属于常量表达式，用常量表达式初始化的 const 对象也是常量表达式。

C++允许将变量声名成 constexpr 型，必须用常量表达式初始化。

一个 constexpr 指针的初始化必须是 nullptr 或者是 0，或者是存储于某个固定地址中的对象

2.5：处理类型。

① 类型别名

两种方式：

Typedef double wages;

Using wages=double;

这两种方法都可以定义类型别名，类型别名的好处就是可以便于阅读理解和使用。

② 试图把别名代入程序去理解语句含义在一些情况下是错误的。

Eg: typedef　　char * pstring;

　Const　pstring　cstr=0;

Pstring 是一个指针类型，加了 const 之后就是一个常量指针，这个常量指针指向 char 类型。

如果用替换的方式来理解就是：const char * cstr; 这样表示 cstr 是一个 char 的指针，指向一个 char 型常量。这样理解是不对的，一定要注意 typedef 的基本数据类型。

　Const pstring *ps.

这个的理解应该是 ps 是一个指针，他指向的是一个 char 型常量指针，ps 并不是常量指针，所以这里并没有初始化。

③ Auto 类型说明符

可以根据右端表达式的类型来推断此 auto 所声名的变量的类型，在不知道要声名什么类型的时候可以这样做，这是 c++11 之后的新特性。

④ Decltype

有时希望从表达式的类型推断出要定义的变量的类型，但是不想用该表达式的值来初始化变量，c++11 中有 decltype 的类型说明，可以返回操作数的数据类型，

2.6：类的定义和操作。

① C++11 规定，可以对类内数据成员提供一个类内初值，在创建对象时就依此初始化对象数据成员，这可以省掉构造函数。

② 头文件一般写的是只能被定义一次的实体，如类，const，constexpr 等。

③ 头文件保护符。

#ifndef xxx

#define xxx

Code----------

#endif

这种头文件保护符是从 c 语言继承过来的，第一次包含此头文件时，预处理器执行后面的操作知道遇见#endif 为止，此时 xxx 的值已经变为已定义，第二次包含时遇到 ifndef 结果就会为假，这些东西就会被忽略过去。

写头文件保护是一个很好的习惯，无论是否需要，一般都把其加上。