待解决问题：

1. 对象相加，P19
2. 养成命名习惯，看库函数方法
3. unique\_ptr
4. 总结文件的空白（空格、tab等）、换行、文件结束符号，方便操作文件
5. 模板和类的区别？模板里元素必须同一类型？
6. 标准库类型和内置的差别
7. const深入理解
8. 与C不同类型或名词

* int main(int argc ,char \*\*aegv，char \* envp)，argc参数个数，包括执行文件，以空格分；argv是输入的参数，是一个数组；envp为环境变量，其参数是命令行或launch文件输入的参数
* cerr输出错误信息，与cout 的区别是不缓冲
* 字面值常量(P37):0x1FF，024（八进制），L‘a’。
* String是一种库类型，表示可变化的字符序列
* 定义在函数体内部的内置类型变量将不会被初始化
* ：：作用运算符，如：std：：cout
* lvalue、rvalue：左、右值
* 引用（复合类型）：int a=1；int &b=a；int &c=a；可多次引用，其中a、b、c本身不是对象，对象是指a指向的地址上的内容
* void\*指针，能做指针比较、函数输入输出、赋给另外一个void\*指针，不能就直接操作其所指对象，因为无法确定对象类型。
* const限定符：它的值（对象不改变），只能也必须在初始化时赋值。

常量只能赋值给常量，常量引用是指引用绑定的对象不变了，但对象可变

* 类型别名

typedef double wages;//wages是double的同义词

typedef wages base,\*p;//base是double的同义词，p是double\*的同义词

* 结构struct具备了类的功能，称为类

<https://www.cnblogs.com/starfire86/p/5367740.html>

* 多了一些与机器无关的类型，c中类型是与计算机单片机等存储、工作方式直接相关的
* sizeof (type),sizeif expr：计算类或对象所占空间大小
* throw{}：抛出异常；try{}：包含可能触发异常语句；catch{}：捕获异常，并处理异常，可以用来打印错误信息

1. C++11新标

* int \*p=nullptr，可以被转换成任意其他类型指针
* constexpr int sz = size()//只有当size为constexpr函数时，才是一条正确的声明。另外，其需绑定有固定地址的变量，如函数体之外的变量
* 别名声明

using SI = sales\_item;//SI是sales\_item的同义词

* 列表初始化

int a = {d};//使用花括号，当d为double时，会报警

* auto类型说明符：编译器根据运算结果确定变量类型，一般忽略顶层const，保留底层const

auto i=0,\*p=&i; //正确；i、p类型相同

auto sz = 4，pi = 3.14;//错误，两者类型不同

* decltype类型指示符

decltype(f()) sum = x;//sum的类型为f（）的返回类型，并不调用f

decltype（（variable））双层括号的结果肯定是引用，decltype（variable）单层括号的结果只有当variable本身就是一个引用时才是引用

* range for：遍历变量（如字符串变量）中基础元素，可以利用其实现下标合法，不越界。
* vector（容器）是一种模板，要求模板类型都一样，如：vector <类型>，可用{}做列表初始化，允许用push\_back动态添加元素至尾部
* 迭代器的两个新函数cbegin和cend返回const\_iterator类型
* begin（数组名）、end（数组名）分别返回数组的首地址的指针和尾元素的下一个地址的指针，可以模仿迭代器
* 商一律向0取整
* sizeof不实际求运算对象的值，故可直接使用作用域符号直接访问类成员，而无需通过对象

1. 自己总结

* #define常量是通过包含头文件用，const常量是通过extern来引用
* const int sz = get\_size()//其具体值在运行时获得，不是常量表达式
* 编辑：使用IDE写代码的过程，像写文件一样，在硬盘（存储器）进行

编译：将代码变成机器可执行的文件（二进制等），存放于硬盘（存储器）

运行：将可执行文件从硬盘（存储器）调入运行内存（内存）执行，会占用运存空间

* 对象是指会在内存上开辟存储空间的，程序是在运存上运行，内存有限，会经常开辟、释放内存
* 一般前面是参数，后面是被操作变量，如：typedef int zhang；
* 一定要用名称（变量名、名称）和对象（内存上数据）的思想思考问题，如在传递参数时把传参看成被传参数赋值给参数，那么引用和指针会改变被传参数
* 内置类是库自带类型，类类型是自己定义的类
* 转义：n在C语言中有特殊意义：一个字母，那么可以通过\来转义，变成换行的意思
* strin和vector是容器，可称容器的迭代器，可用迭代器像下标方式一样访问容器元素，返回的是迭代器类型
* int \*ptr；其中ptr自己占用一个地址，为一个对象，\*ptr解引用才是它指向的对象
* 越界出错，可能取到一个未知的地址，取得一个未知的数
* 常见、未知错误：溢出（下标、值范围）、未定义（指针未定义）
* 左结合： 优先级相同时，从左到右结合，右结合：优先级相同时，由右向左结合。如：=是右结合
* 指针是指P，指针本身
* assert函数：先计算表达式 expression ，如果其值为假（即为0），那么它先向stderr打印一条出错信息，然后通过调用 abort 来终止程序运行
* set类元素都是唯一的，可以方便排序，

<https://blog.csdn.net/byn12345/article/details/79523516>

* new新建的是一个指针实例
* inline宏函数

<https://blog.csdn.net/K346K346/article/details/52065524>

* 运算符重载operator

<https://blog.csdn.net/zhangzheng_1986/article/details/81080407>

* 在C++中，直接=的话，会将两个值类型指向同一个地址这样的话，修改任何一个对象，另一个对象也会跟着变化？？Clone执行了浅拷贝~~~，拷贝后是一个新的对象，对新对象操作不影响原对象。非std类才需这样？
* 实例名（参数），调用operator（）类函数
* 如下代码，这不同于赋值操作，改变temp就改变image
* static\_cast < type-id > ( exdivssion ) 该运算符把exdivssion转换为type-id类型，但没有执行时类型检查来保证转换的安全性。

<https://www.cnblogs.com/yxwkf/p/4090364.html>

* reserve减少重新分配次数，局部运算中，内存会被释放，那么多给点内存也没事。

<https://blog.csdn.net/zhangchen1003/article/details/49070341>

* 只有全局变量一直保存在内存中，局部变量在不断建立和释放
* iterator可以把迭代器指针像数一样操作
* erase：删除容器中position所指位置的元素。返回值是指向被删元素之后的那个元素(即下一个元素)的迭代器。

<https://www.cnblogs.com/wangkundentisy/p/9023977.html>

* front（）函数：取的是内容

<https://blog.csdn.net/laobai1015/article/details/51568456>

* 对于类中的类指针，可以使用new创建指针实例类中类指针实例=new 类名（参数）；对于类中的类实例，可以使用类中类实例=类名（参数）
* 1

1. 类型决定了数据所占的比特数，以及该如何解释这些比特内容
2. 字节-8比特，字-32比特，float-字，double-双字
3. 负数以补码形式存储，补码=原码按位取反+1。有符号、无符号数相互转换就是直接读取内存中的字节。默认情况下有符号变为无符号。
4. 未经初始化的变量和指针都会引发不可预知的错误。（为什么不由编译器检查呢）
5. 指针的值是地址，指针的对象是所指地址上内容
6. P141隐式转换：小整数类型的运算对象被提升成较大的整数类型（如short int转换成int，有符号转换成无符号），所有运算的对象最终会转换成同一类型（显式也一样）。被转换类型所占空间不应大于转换类型所占空间。尽可能避免损失精度。
7. 整型提升：转换后的类型能容纳原类型的所有值，如：P142，先转为int，否则，转为unsigned int
8. 强制转换：static\_cast:类型转换，不能改变常量属性；const\_cast：改变常量属性，不能做类型转换
9. 迭代语句通常称为循环语句
10. 1