参考STL allocator进行实现，采用模板变成，支持任意类型。

private:  
 union Slot\_ {  
 value\_type element;  
 Slot\_\* next;  
 };  
  
 typedef char\* data\_pointer\_; //

typedef Slot\_ slot\_type\_; //Slot要么是value\_type，要么是Slot\*，大小为两个的较大值  
 typedef Slot\_\* slot\_pointer\_;  
  
 slot\_pointer\_ currentBlock\_; //block list头指针  
 slot\_pointer\_ currentSlot\_; //当前可用的第一个slot的位置，即可以存放value\_type数据的位置  
 slot\_pointer\_ lastSlot\_; //最后一个slot的位置  
 slot\_pointer\_ freeSlots\_; //空闲链表头指针

将对象的构造分为两个部分：1，分配内存 2，对象构造。分别对应于类的成员函数allocate和construct。

内存池由一块一块的block组成，block之间通过链表相连。当内存池的内存不够时，通过allocateBlock()申请新的block加入。将每个block按照内存池中的存储类型value\_type划分成一个一个的slot。每个对象内存占用block中的一个slot

内存池创建时，并不分配内存，当有内存需求，即调用allocate时，如果发现内存池中没有内存，则调用allocateBlock分配一个block。

当对象析构之后，并不释放占用的内存空间，而是将其加入freeslot空闲链表中。表明可以在这部分内存构造对象。

有内存请求时，首先判断freeslot是否有空闲内存块存在，如果没有的话，就看分配的block中是否还有空闲块，如果还没有，就新建一个block。

**在内存池运行期间，内存池的大小只会增长不会减小。直到把内存池销毁之后，才会把所有的block释放。**

实现的时候用到的重要的知识：

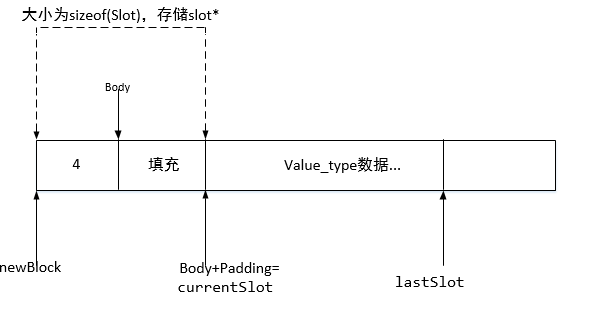
1、基于STL allocater的架构

2、union结构体

3、placement new

4、reinterpret\_cast进行底层的转换，不同类型的指针之间，指针和int之间等。

每个block的结构



每个block的第一个slot存储一个slot\*指针，指向下一个block

每个slot要么是value，要么是next指针，当空闲时，存储next指针，组成freeslot链表

当存储对象时，存储value。

关于union继续看