基础11 C++中的异常处理以及swap & copy

①异常处理的手段 try-catch-throw

抛出异常：throw 异常；

作用：让调用者看到这个异常，层层通知，直到程序崩溃

接住异常：try{可能抛出异常的code} catch(异常类型){处理方式}

常为const &，防止拷贝出现新异常

如果异常类型不确定，也可用catch(…){处理方式}，并且在catch中仍可使用throw继续抛出

父类异常引用可接子类异常(详情见Base5.6)

当前类型未捕获异常，则需判段其它类型

接收顺序为子类，父类，catch{…}必须放最后

②标准库的异常类(部分)

<expection>

exception

父：bad_alloc <new> 内存不足时抛出；logic_error 逻辑错误；runtime_error 运行时错误；bad_cast <type_info> dynamic_cast引用转换失败时

子：length_error out_of_range std容器越界时；domain_error invalid_argument；range_error underflow_error overflow_error

<stdexception>

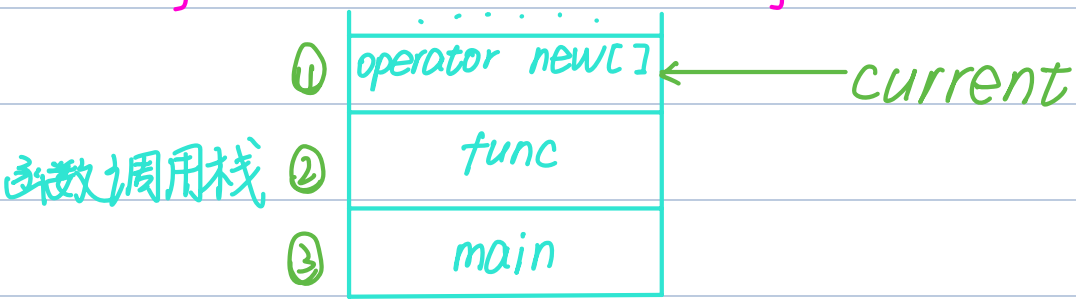
③栈展开 Stack Unwinding (作用十分有效，只能销毁局部对象)

```
void func(int n){
    int x=42;
```

```
③int main{
    int size=100;
```

① int *p = new int [n];
....}

try{
② func(size); } catch(...){}
}



① size很大，内存爆炸，p在申请内存时抛出异常，且p未被创建

② 由于①无异常处理，②也无异常处理，此时调用栈回到main函数

因此回收了p，x的内存

③ 异常被抛到main函数内，并且得到了处理

注：栈展开并非任何时刻发生，但在catch到异常时一定发生

④构造函数的try-catch

```
class Array{
  public: Array(std::size_t n)
          try : m_size(n), m_data(new int[n]){}
          catch(...){...}
  private: size_t m_size; int *m_data;}
```

try的作用范围

⑤异常安全保证

a. 不抛出保证(Nothrow guarantee)：保证一定不抛出异常

b. 强异常安全保证(Strong guarantee)：异常抛出，但程序状态不变

c. 弱异常安全保证(Weak guarantee)：状态改变，但都在有效态

⑥ 不抛出保证 noexcept 关键字可以让编译器进行更多的优化

如果函数声明了 但还是抛出异常，调用栈可能开解，

导致程序直接崩溃

注：移动构造和移动赋值如果不声明成 noexpect，编译器是不敢用的

noexcept 的两种特殊用法：

a. noexcept(bool)：相当于开关noexcept作用 ← 编译期值

b. noexcept(noexcept(std::swap(thing, other.thing)))

noexcept 表达式，可根据内部是否异常决定 nocept 的开关

⑦ swap & copy

涉及手动清理释放内存的类，在进行拷贝赋值的时候，非常容易因为代码的顺序而破坏异常安全性，而 swap& copy 操作解决了这个问题