头文件：

- < iomanip>
  - srand((unsigned)time(NULL))

- <initializer_list>
  - initializer_list<T>

- <limits>
  - (std::numeric_limits<T>::max)()
  - (std::numeric_limits<T>::max)()

1. **关于默认构造函数**
   - vector<A>(0);这个不会调用 A 的默认构造函数！（如果 A 没有默认构造函数，会出现"尝试引用被删除的函数"的错误）
   - vector<A>(1);这个会调用 A 的默认构造函数！
   - vector<A>();不会调用 A 的默认构造函数！（如果 A 没有默认构造函数，也不会出现"尝试引用被删除的函数"的错误）
   - 调用默认构造函数的方法　vector<A> v; 或者显式调用　vector<A> v=vector<A>(); 这两者都不会调用 A 的默认构造函数，也不会出现"尝试引用被删除的函数"的错误
   - 当调用 vector<T>的默认构造函数时，vector<T>的默认构造函数不会调用类型 T 的默认构造函数
2. **若类类型 b 含有为类类型 a 的数据成员**

   如果对 b 进行默认初始化　若 b 的数据成员 a1 含有类内初始值，那么用该初始值初始化 a1,否则调用类类型 a 的默认构造函数
3. **指针的数组如何定义（动态数组）**

   (int*)* a;首先 a 是一个指针，指针指向的元素是 int*
4. **extern：**
   - 当头文件 a.h 中定义了一个变量，并在 b.cpp,c.cpp 中引入了该头文件，那么会产生重复定义的错误（无论是否使用宏来避免重复声明）。解决方案：在 a.h 中在变量前添加 extern，将其变为实例化声明，这样使用宏就能避免重复的声明，然后在任一个 cpp 中对其进行定义，cpp 文件只要包含该声明变量的头文件即可
   - 也可以在一个 cpp 中定义，其余 cpp 文件用 extern 进行声明

6、嵌套 try 语句，当内层 try 的代码抛出异常且被内层 try 的 catch 捕获后，外层的 try 会继续执行剩余语句

```
int main() {
    STACK<int> S(1);
    try {
        if (true){
            try {
                S.PUSH(1);
                S.PUSH(2);
```

```
                S.PUSH(3);
            }
            catch (out_of_range error) {
                cout << "Inside:" << error.what() << endl;
            }
        }
        cout << "here" << endl;
    }
    catch (out_of_range error) {
        cout << error.what() << endl;
    }
    catch (logic_error error) {
        cout << error.what() << endl;
    }
    system("pause");//定义在 iostream 中
    return 1;
}
```

5. 含有动态指针的类，要自定义析构函数以及其他五个函数

6. delete 销毁一个对象，delete[]销毁一个序列。

- 对于基本类型的数组来说，两者没什么区别
- 如果数组包含的是类的对象指针，则两者有区别。delete 只会引起第一个元素析构，而 delete[]会依序调用所有元素的析构

7. 头文件互相包含引发的大量未声明错误

```
//a.h
#pragma once
#include "b.h"
class A{
};

//b.h
#pragma once
//加上这句"class A;"
#include "a.h"
class B{
    A a;
};

//main.cpp
include "b.h"
int main(){
    return -1;
}
```

三个文件如下，类型 B 中包含有 A 的对象，因此需要包含 A 对象的头文件。
但是 a.h 中引入了 b.h。在编译时，相当于把 class B 的定义放到了 class A 之前，因此 class B 中类型 A 是未定义的。
解决方法是：加上标记为红色的这句，在 B 定义前，前置声明一下 A 即可。或者直接删掉 a.h 文件中的#include "b.h"，头文件相互包含本身就不合理！！！