1. 类型转换
   1. 静态转换 static\_cast
   2. 使用方式 static\_cast< 目标类型>（原始数据）
   3. 可以进行基础数据类型转换
   4. 父与子类型转换
   5. 没有父子关系的自定义类型不可以转换
   6. 动态转换 dynamic\_cast
   7. 不可以转换基础数据类型
   8. 父子之间可以转换
      1. 父转子 不可以
      2. 子转父 可以
      3. 发生多态 都可以
   9. 常量转换 const\_cast
   10. 不能对非指针或者非引用进行转换
   11. 重新解释转换 reinterpret\_cast
       1. 最不安全，最鸡肋 不推荐
2. 异常
   1. try 试图执行 try{}中的内容
   2. 在可能出现异常的地方 抛出异常 throw
   3. try下面 catch捕获异常
   4. catch( 捕获类型 ) …代表 所有其他类型
   5. 如果不想处理异常，继续向上抛出 throw
   6. 如果没有任何处理异常的地方，那么成员调用terminate函数，中断程序
   7. 自定义异常类 ，可以抛出自定义的对象 ，捕获自定义的异常
3. 栈解旋
   1. 从try开始 到 throw 抛出异常之前 所有栈上的对象 都会被释放 这个过程称为栈解旋
   2. 栈上对象构造顺序与析构顺序相反
4. 异常的接口声明
   1. 如果想抛出特定的类型异常 ，可以利用异常的接口声明
   2. void func() throw ( int) 只能抛出 int类型
   3. throw() 不抛出任何类型异常
5. 异常变量生命周期
   1. 如果 MyException e，会多开销一份数据 ,调用拷贝构造
   2. 如果 MyExcepiton \*e ， 不 new提前释放对象 new 自己管理delete
   3. 推荐 MyException &e 容易些 而且 就一份数据
6. 异常的多态使用
   1. 利用多态来实现 printError同一个接口调用
   2. 抛出不同的错误对象，提示不同错误
7. 使用系统标准异常
   1. #incldue <stdexcept>
   2. throw out\_of\_range（”aaa”） 。。。
   3. catch(out\_of\_range & e) cout << e.what();
8. 编写自己的异常类
   1. 自己的异常类 需要继承于 exception
   2. 重写 虚析构 what（）
   3. 内部维护以错误信息 字符串
   4. 构造时候传入 错误信息字符串，what返回这个字符串
   5. string 转 char \* .c\_str();
9. 标准的输入流
   1. cin.get 缓冲区中读取一个字符
   2. cin.get(两个参数) 不读换行符
   3. cin.getline () 读取换行 并且扔掉
   4. cin.ignore 忽略 （N） N代表忽略字符数
   5. cin.peek 偷窥 偷看1个字符然后放回去
   6. cin.putback 放回 把字符放回缓冲区
10. 输入流案例
    1. 判断用户输入的是字符串还是数字 利用偷窥 或者 放回
    2. 让用户输入指定范围内的数字，如果不正确 重新输入
       1. cin.fail() 看标志位 0正常 1不正常
       2. cin.clear()重置标志位
       3. cin.syne() 清空缓冲区
11. 标准输出流
    1. 流对象的成员函数
       1. int number = 99;
       2. *cout*.*width*(20);
       3. *cout*.*fill*('\*');
       4. *cout*.*setf*(*ios*::*left*); //设置格式 输入内容做对齐
       5. *cout*.*unsetf*(*ios*::*dec*); //卸载十进制
       6. *cout*.*setf*(*ios*::*hex*); //安装16进制
       7. *cout*.*setf*(*ios*::*showbase*); // 强制输出整数基数 0 0x
       8. *cout*.*unsetf*(*ios*::*hex*);
       9. *cout*.*setf*(*ios*::*oct*);
       10. *cout* << number << *endl*;
    2. 控制符

int number = 99;

*cout* << *setw*(20)

<< *setfill*('~')

<< *setiosflags*(*ios*::*showbase*) //基数

<< *setiosflags*(*ios*::*left*) //左对齐

<< *hex* // 十六进制

<< number

<< *endl*;

1. 文件操作
   1. 写文件
      1. ofstream ofs
      2. open 指定打开方式
      3. isopen 判断是否打开成功
      4. ofs << “数据”
      5. ofs.close
   2. 读操作
      1. ifstream ifs
      2. 指定打开方式 ios：：in
      3. isopen判断是否打开成功
      4. 三种方式读取数据