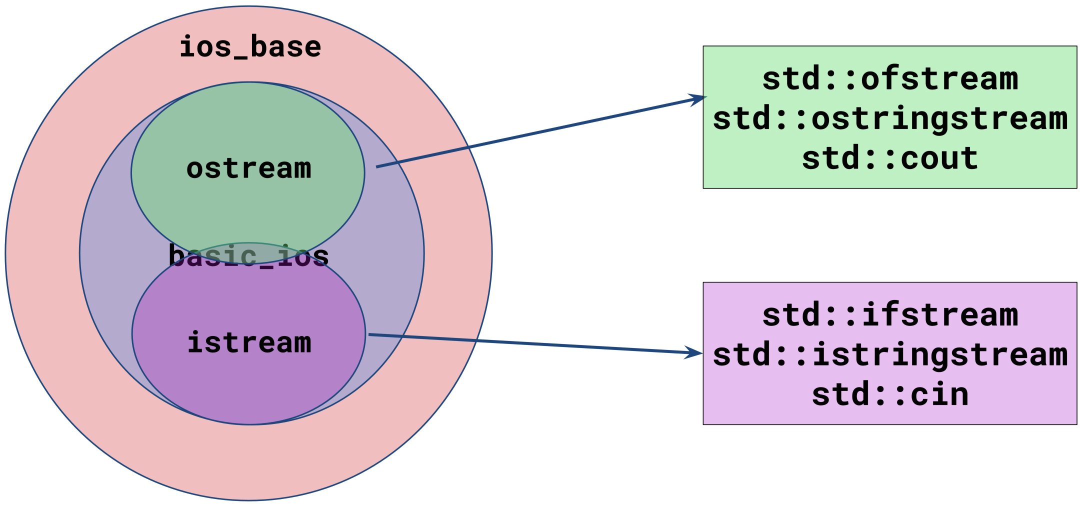
lecture4

what:streams
what more:
1.streams what
2.strings streams
3.cout and cin
4.output streams
5.input streams

1.streams: a general input/output facility for c++
streams help us read and write
distinction:
`fcout<<data` 文件写入
`fin>>data` 文件读取
`cout<<data` 控制台输出
`cin>>data` 控制台输入
console:控制台
streams allow for a universal way of dealing with external data



2.strings streams

[2025Fall-04-Streams, p.47]()

> a way to treat strings as streams

```
void foo() {
    /// partial Bjarne Quote
    std::string initial_quote = “Bjarne Stroustrup C makes it easy to shoot
yourself in the foot\n”;
    /// create a stringstream
    std::stringstream ss(initial_quote);
    /// data destinations
    std::string first; std::string last; std::string language,
extracted_quote;
        ss >> first >> last >> language >> extracted_quote;
        std::cout << first << “ ” << last << “ said this: ”<< language << “
“ << extracted_quote << std::endl; }
```

3.cin and cout

for example:

```
std::cout<<std::flush
std::cout<<std<<std::endl;
此处的endl，是cout<<'\n'<<std::flush
```

key:
cout的输出条件：
手动刷新:flush/endl
程序结束输出(reach the end of program)
缓冲区满(buffer is full)
在读取cin前自动刷新（绑定流）

对比：cerr与clog
std::cerr:输出错误信息，无缓冲，立刻
std::clog:日志输出，有缓冲

caveat:警告
针对缓冲模式：
行缓冲：\n刷新
全缓冲：达到一定大小刷新
无缓冲：立刻输出

std::ios::sync_with_stdio(false):关闭同步流（解除c++流和c流的同步）

interactive:交互式
使用上面，是否行缓冲取决于输出设备类型
console：interactive-->行缓冲
txt:non-interactive-->全缓冲，全程序运行结束才输出

注意：多用'\n'代替endl

output file streams=>use std::ofstream
some functions:
is_open()
open()
close()
fail()

区分：
std::ofstream ofs("file.txt") 写入数据到文件
std::ifstream ifs("file.txt") 从文件输入

关于getline():
语法：std::getline(input_stream,string_variable)
如：getline(cin,str)

cin读：读到空格，\n,tab就停了，但是\n还在buffer区域
getline读：读到\n为止，但对\n，读取，但不留存，也就是把\n从buffer区域踢出去

不要把cin和getline混用！
修复方法：
cin>>str
getline(cin,str)
getline(cin,str)