程序考试经验交流

蔡聪辉

2015年1月13日

特别感谢赵鸿泽师兄（MFC），乔鑫同学（C++编程）对复习材料的补充

程考主要考什么？

- 同学A：算法？(好像不是)
- 同学B：C++ STL库的灵活使用？（好像有点是）
- 同学C：MFC界面编程?（最后一题分值好像有点重）
- 同学D：基本编程能力？（废话.....）
- 实验室师兄：很简单的，考的就是基本的C++语法
- 众人顿悟：好像很有道理的样子.....

我们需要掌握什么？

- C++ 基础语法 （必备）
- C++ STL库的使用 （非必备，但可以提高编码速度和正确率）
- MFC界面编程（重要）
- 必须注意算法效率吗？ 答：否，哪种方法最熟练能够快速正确地解决问题，那就用哪种方法
- 必须使用面向对象的编程方式吗？答：否，能解决问题的方法就是好方法，一般面向过程就能解决问题，除非题目特殊要求
- 简单最好, 理论上暴力法的运行时间都可以在程考的忍受范围内

知己知彼之程序考试

- 编程环境 VS2012
- 编程时长 3小时
- 编程题型 2道简单编程题 + 1道MFC界面编程题
- 考前准备时间检查下编程环境，如果有问题，申请换机器
- 用简单的方法快速做完前两道题目，不要在一个细节处纠结太久
- 程序考试的目标是通过
- 使用VS的调试功能帮助解决异常或者程序错误
- 适当地给代码加注释，按照功能点完成

程考C++的主要知识点

- 字符，字符串（主要是string对象）操作，使用STL库帮助解决问题，<cctype>头文件有单个字符处理函数,tolower(int c), toupper(int c)等
- 控制台的输入输出操作 cout，cin
- 文件对象的读写 <fstream> 头文件封装了文件的IO操作
- 进制之间的互相转换，如十进制转十六进制等 atoi，itoa
- 熟悉string ,vector, map, set 等类型的使用，能够使用iterator 和 reverse_iterator对容器对象进行遍历
- 能够用C++语法解决基础算法问题(在忘记API的情况下依然可以解决问题)

MFC界面编程的主要知识点

- 能够建立一个基于单文档或者对话框的可运行基本程序
- 熟悉MFC的消息处理机制，能够正确添加事件处理程序
- 能够在OnDraw（对话框是OnPaint）函数中实现绘制代码
- 熟悉CString, string, char*之间的区别和转换
- 能够将界面上的值与属性变量的值进行同步或获取控件的值
- 读写文件，对象序列化（基于单文档的程序）
- 常见题型，棋盘类（如扫雷，五子棋等），家庭收支管理（需要学会多种控件的使用）

如何进行MFC编程的复习

- 看材料，先学会建立一个基本的对话框或单文档程序
- 依样画葫芦（模仿别人的代码实现同样的程序，熟悉MFC消息处理机制）
- 多研究，多思考（不要追求题量，看一两道题就可以基本解决所有可能出现的MFC类型的题目，棋盘类问题是经典）
- 多讨论，多交流（有想法跟大家分享，多个人一起交流，可以事半功倍的效果）
- 不要纠结于细节，重点掌握界面绘制功能，MFC编程的评价标准是首先程序能运行，按照功能点得分

C++程考编程小技巧

Tips #1:
Simple things should be simple.

```
vector<int> vi;
```

How to print the contents?

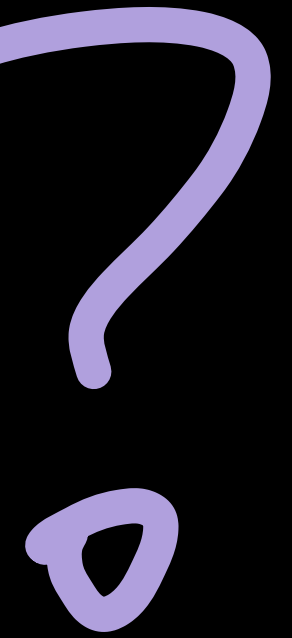
```
vector<int> vi;
for(vector<int>::iterator
   it = vi.begin();
   it != vi.end();
   ++it)
      cout << *it;
```

```
vector<int> vi;
for(auto it = vi.begin();
  it != vi.end();
  ++it)
    cout << *it;
```

```
vector<int> vi;
for(auto& e : vi)
  cout << e;
```

```
string year_str = "2015";

int year = 2015;
```



```
string year_str = "2015";
```

for loop *10 ?

```
int year = 2015;
```

```
string year_str = "2015";
```

sscanf ?

```
int year = 2015;
```

```
string year_str = "2015";
```

string → char* → atoi ?

```
int year = 2015;
```

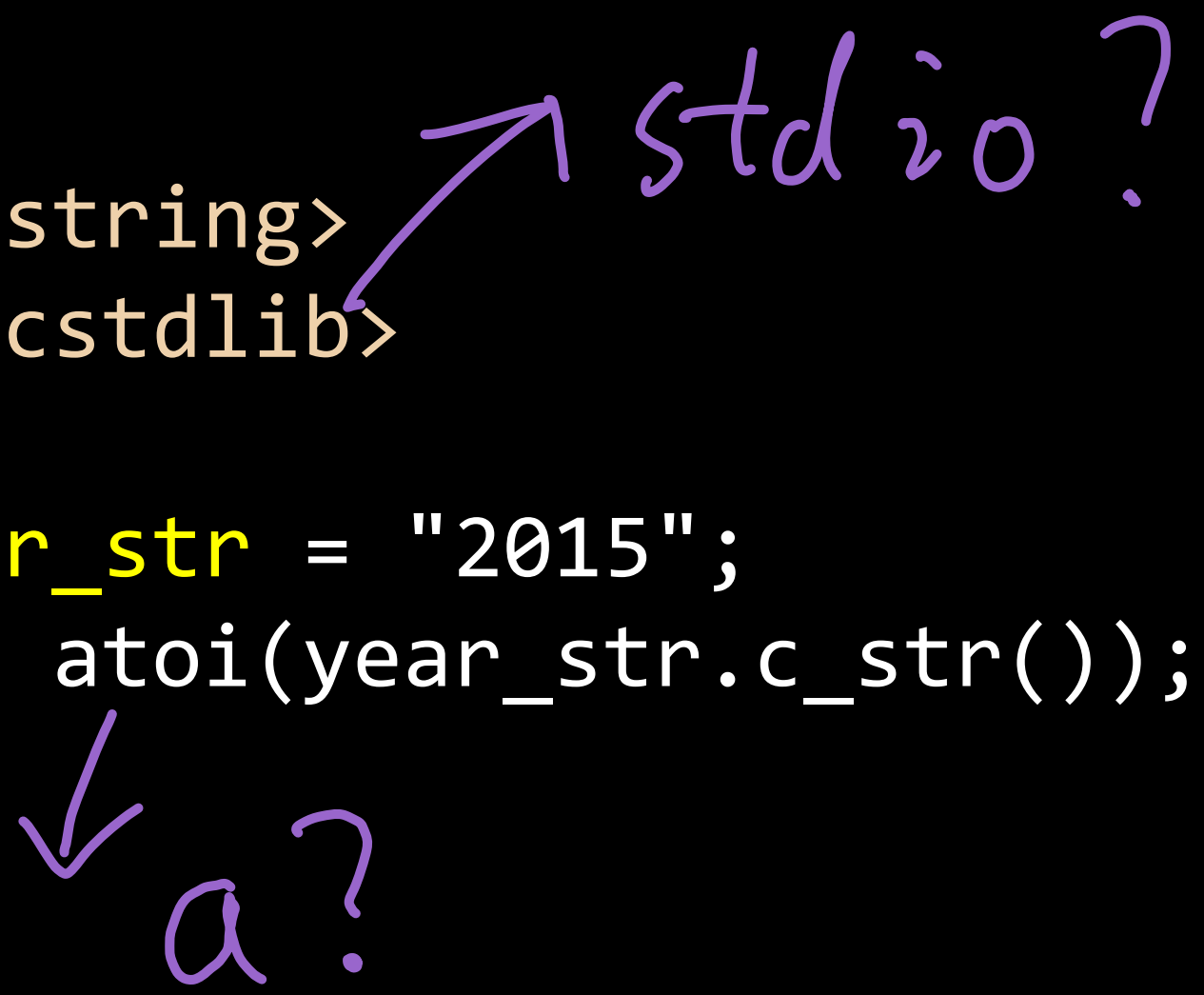
```
string year_str = "2015";
```

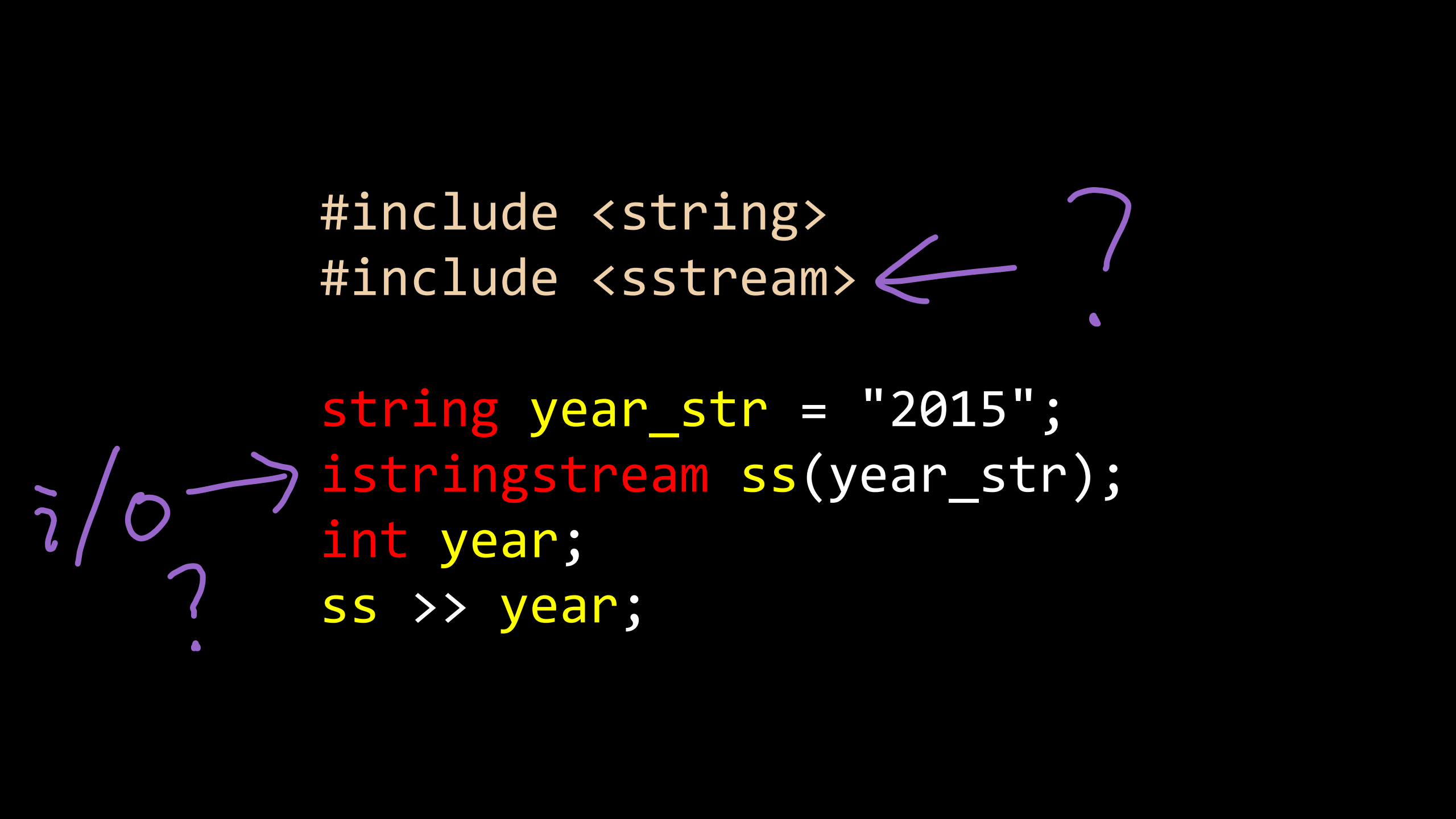
string stream?

```
int year = 2015;
```

```
#include <string>
#include <cstdlib>

string year_str = "2015";
int year = atoi(year_str.c_str());
```




#include <string>
#include <sstream>
string year_str = "2015";
istringstream ss(year_str);
int year;
ss >> year;
i/o
?
?

```
string year_str = "2015";
```

stoi

```
int year = 2015;
```

```
#include <string>

string year_str = "2015";
int year = stoi(year_str);
```

#include <string>

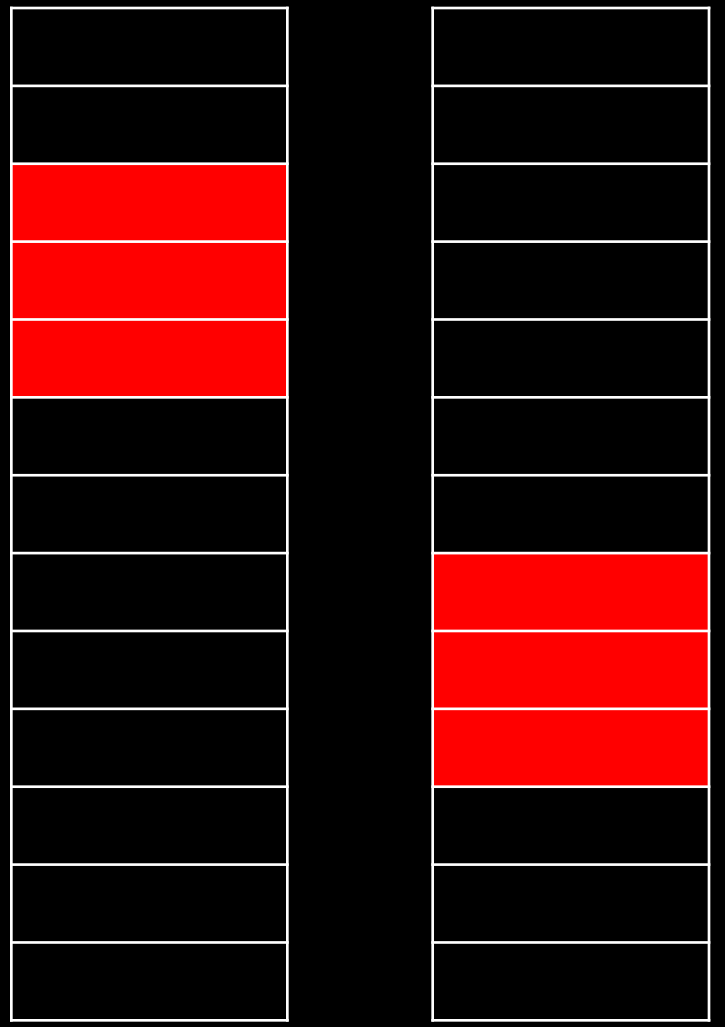
- stol() Convert string to long int
- stof() Convert string to float
- stod() Convert string to double

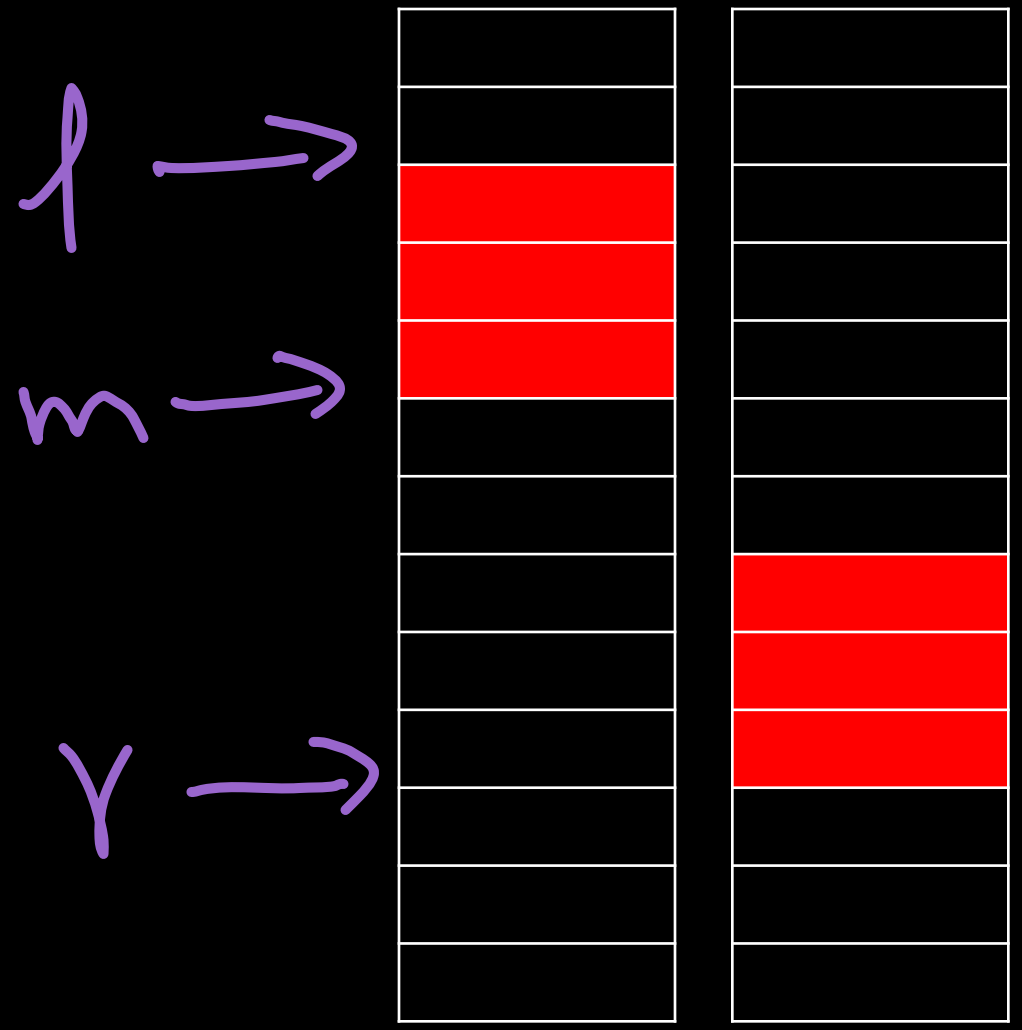
- to_string() Convert numeric value to string

Simple things should be simple.

don’t complicate them

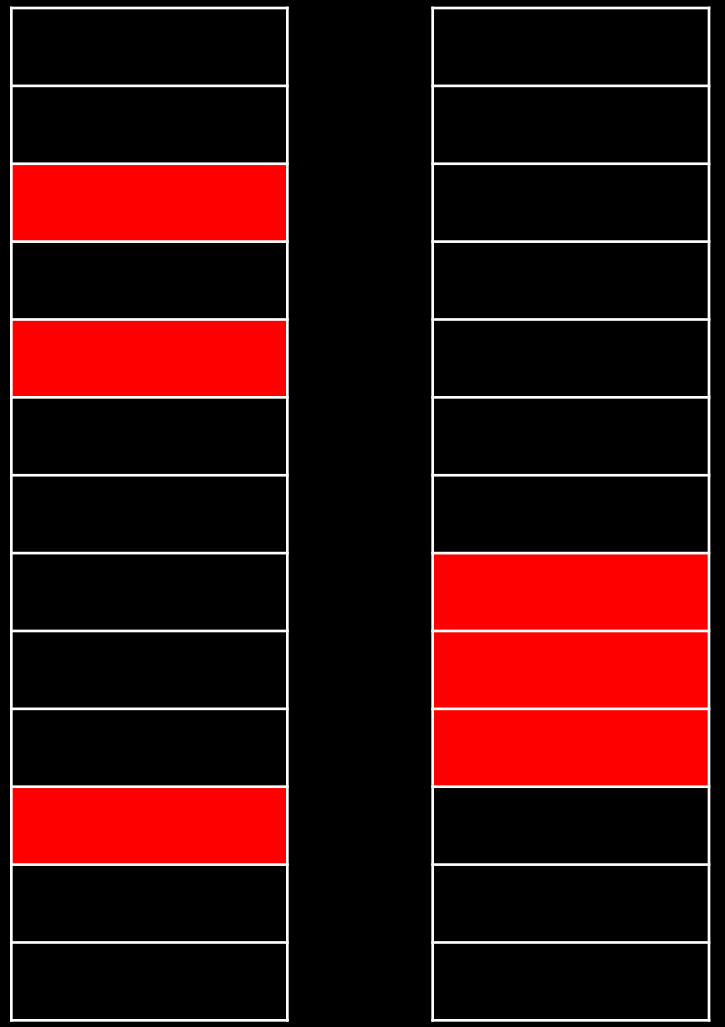
Tips #2:
Never reinvent wheels.





std::
rotate(l, m, r)

reverse *3



?

std::

stable_partition

① (begin, m, p)

② (m, end, p)

Never reinvent wheels.

use STL algorithms instead

Query

- all_of Test condition on all elements in range
- any_of Test if any element in range fulfills condition
- none_of Test if no elements fulfill condition
- for_each Apply function to range
- find Find value in range
- find_if Find element in range
- adjacent_find Find equal adjacent elements in range
- count Count appearances of value in range
- count_if Return number of elements in range satisfying condition
- is_permutation Test whether range is permutation of another
- search Search range for subsequence

Modification

- copy Copy range of elements
- copy_if Copy certain elements of range
- swap Exchange values of two objects
- swap_ranges Exchange values of two ranges
- transform Transform range
- replace Replace value in range
- replace_if Replace values in range
- generate Generate values for range with function
- remove Remove value from range
- remove_if Remove elements from range
- unique Remove consecutive duplicates in range
- reverse Reverse range
- rotate Rotate left the elements in range

Partition & Sorting

- is_partitioned Test whether range is partitioned
- partition Partition range in two
- stable_partition Partition range in two - stable ordering
- partition_point Get partition point

- sort Sort elements in range
- stable_sort Sort elements preserving order of equivalents
- partial_sort Partially sort elements in range
- is_sorted Check whether range is sorted
- nth_element Sort element in range

Binary search & Merge

- lower_bound Return iterator to lower bound
- upper_bound Return iterator to upper bound
- equal_range Get subrange of equal elements
- binary_search Test if value exists in sorted sequence

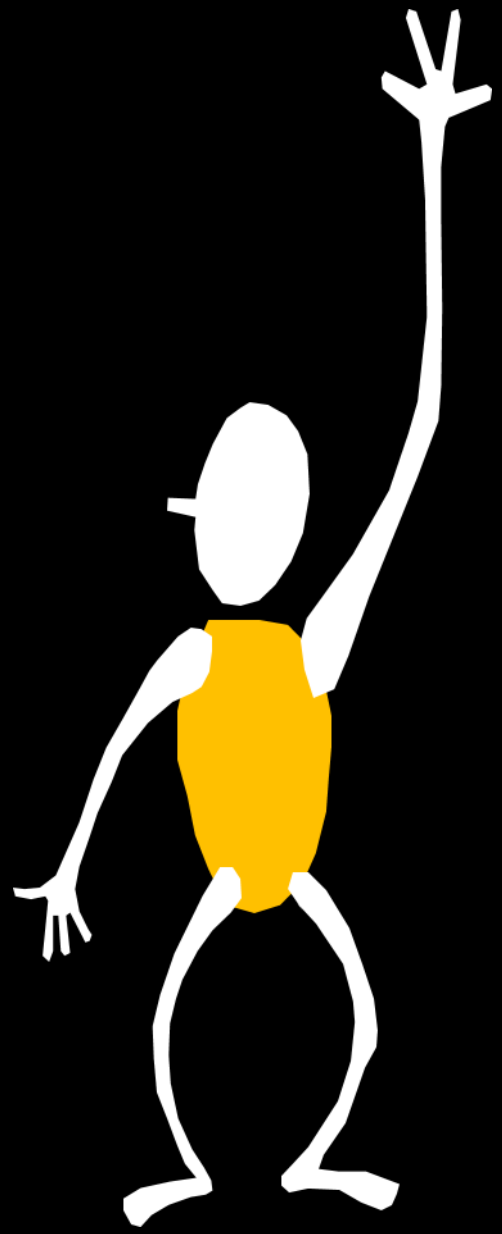
- merge Merge sorted ranges
- inplace_merge Merge consecutive sorted ranges
- includes Test whether sorted range includes another sorted range
- set_union Union of two sorted ranges
- set_intersection Intersection of two sorted ranges
- set_difference Difference of two sorted ranges
- set_symmetric_difference Symmetric difference of two sorted ranges

Min/max & Other

- `min` Return the smallest
- `max` Return the largest
- `min_element` Return smallest element in range
- `max_element` Return largest element in range

- `next_permutation` Transform range to next permutation
- `prev_permutation` Transform range to previous permutation

Feel free to ask questions



Bonus slide: lambda

```
std::array<int,7> foo {1,2,3,4,5,6,7};
std::partition(foo.begin(), foo.end(),
     [](int x){
         return x % 2 == 1;
     });

auto comp = [](Person x, Person y){ return x.rank < y.rank; };
std::set<Person, decltype(comp)> sp(comp);
```