> 水平有限，如有错误，欢迎指出

实现 `split` 函数

我们希望实现两个函数，签名如下：

```
// vector 类似 Java 的 ArrayList，需要引入 <vector> 头文件，具体可以参考 cppreference
std::vector<std::string> split(const std::string &s, const char delim);
std::vector<std::string> split(const std::string &s, const std::string &delim);
```

基本实现

```
std::vector<std::string> split(const std::string &s, const char delim) {
    std::vector<std::string> ans;
    // stringstream 可以包装字符串，让我们能在字符串上进行 I/O 操作，使用上跟 cin、cout 相似
    // 需要引入 <sstream> 头文件
    std::istringstream iss(s);
    std::string token;
    // getline 的第三个参数为定界符（delimiter），注意这个 delimiter 会被丢弃
    while (std::getline(iss, token, delim)) {
        // vector::push_back 表示在 vector 的末尾添加一个元素
        ans.push_back(token);
    }
    return ans;
}
```

```
std::vector<std::string> split(const std::string &s,
                               const std::string &delim) {
    std::vector<std::string> ans;
    // begin 和 end 确定一个左闭右开区间 [begin, end)
    // std::string::npos 是一个非法的索引，定义为：
    //     static const size_t npos = -1;
    // 因为 size_t 是一个无符号类型，所以实际上 npos 是 size_t 类型的最大值
    int begin = 0, end = std::string::npos;
    do {
        // PPT 上这里有问题！
        end = s.find(delim, begin);
        if (end == std::string::npos) {
            // end == std::string::npos 说明从 begin 开始的子字符串
            // 再找不到 delim，这样剩下的部分就是一个整体
            // substr 只有一个参数时跟 Java 的类似，都是从指定的位置开始一直到
            // 字符串末尾
            ans.push_back(s.substr(begin));
        } else {
            // [begin, end) 的长度为 end - begin
            // 建议大家在表示范围是使用这样的方式，有兴趣的可以看看 Dijkstra 的
            //     Why numbering should start at zero
            // 传送门在文末
            ans.push_back(s.substr(begin, end - begin));
            begin = end + delim.length();
        }
    } while (end != std::string::npos);
    return ans;
}
```

如果运行测试代码（提供的 `str_split_test.cpp`），会发现对于 `":root"` `"root:"` `root::0:/bin/bash` 这样的输入会输出 `["root", ""]` `["", "root"]` `["root", "", "x", "0", "/bin/bash"]`，我们希望把空字符串丢弃。

改进的实现

```
std::vector<std::string> split2(const std::string &s,
                                const std::string &delim) {
    std::vector<std::string> ans;
    int begin = 0;
    // 空字符串实际上是因为我们用 [begin, end) 划定了一个空区间，那么我们丢弃空字符串的基本思路就是
    // 检查 [begin, end) 是否为空区间
    while (begin < s.length()) {
        int end = s.find(delim, begin);
        if (end == std::string::npos) {
            ans.push_back(s.substr(begin));
            // 注意这里，同前文所述，如果在剩余的子字符串中没有找到 delim，
            // 我们就将剩余的部分看作一个整体，下一步应该考虑如何退出循环
            // 在这里我们用 break，你也可以将 begin 设置为某个 >= s.length() 的值
            break;
        } else {
            // 这部分就是我们检查空区间的逻辑，对于左闭右开区间 [begin, end)，
            // begin == end 就表示空区间
            if (begin == end) {
                begin = end + 1;
            } else {
                ans.push_back(s.substr(begin, end - begin));
                begin = end + delim.length();
            }
        }
    }
    return ans;
}
```

> 有兴趣的同学可以看看这篇：How to split a string in C++

> 《Why numbering should start at zero》的传送门

> 关于我们上课时提到的字符编码问题，这篇文章很不错。

> 关于编码风格的问题，可以看看 Google 的 C++ 风格指南