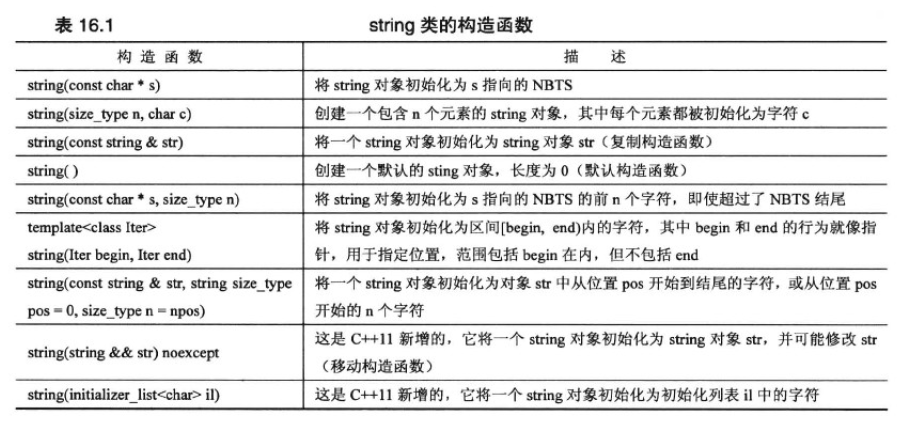
**Chapter 16 String类和STL库**

1. **String类**

丰富的接口：构造函数，输入，比较，size，查找…

1. **构造函数**



1. **输入**
2. Cin.getline（info, 100, ‘;’） //读到；为止，且不读入分号，CString头文件
3. Getline(cin, info, ‘;’) //读到；为止，且不读入分号，string头文件
4. 由于是DMA，所以会自动处理string内存的大小
5. 限制因素：可使用的内存量
6. 到达文件尾。输入流的eofbit将被设置。Fail()和eof()都返回true
7. 遇到分界字符（默认是\n）。删除分界字符且不存储它
8. 读取的字符数达到最大允许值（string :: npos和可供分配的字节数中较小的一个），设置输入流的failbit。Fail()返回true
9. String版本的operator>>()函数会不断读取，直到遇到了空白符并将其保留。通常分空白符包括空格，换行，制表符。
10. **功能**
11. 六个运算符都已经被重载，可以直接用于string串
12. 查找子串第一次出现的位置；查找某字符第一次/最后一次出现的位置；查找第一个/最后一个不包含在参数里的字符
13. **智能指针模板类**
14. 智能指针模板的作用在于出现下面的情况后**自动调用析构函数，防止内存泄漏**auto\_ptr; unique\_ptr<float>pd; shared\_ptr<double>mr
15. 复制构造函数里：\*ps = new std::string (str) 对象过期，不会自动调用析构函数，内存泄漏
16. 某函数里：\*ps = new std::string (str) … throw exception 此时前面的函数被自动结束，内存会泄漏
17. 本质是每次在函数里new了以后return之前容易忘掉delete，所以调用只能指针模板类，创建指针是创建一个智能指针模板类的对象，然后指针释放时调用类里的析构函数
18. **使用**
19. 头文件memory
20. 指向某类的指针替换为指向该类的智能指针对象
21. 所有的智能指针模板类都定义了explicit构造函数，因此禁止隐式转换
22. 非new的内存不能使用只能智能指针
23. **Attention**

**避免两个指针指向同一个内存**

1. 定义赋值运算符，执行深度复制，两个指针指向不同的对象
2. 建立ownership概念，对于特定的对象，只能有一个智能指针可以操作它。Auto\_ptr和unique\_ptr使用此策略（auto\_ptr在C++11几乎不使用）
3. 创建智能更高的指针，跟踪引用特定对象的智能指针数量，即引用计数reference counting。赋值时计数+1，删除时计数-1，仅当最后对象过期时才会调用delete。Shared\_ptr采用此策略
4. 同样的策略也适用于复制构造函数
5. **标准模板库STL**

∈泛型编程generic programming

容器，迭代器，函数，算法…

可以构造各种容器和执行容器上的各种操作

每个标准模板库都包含一个参数用于选择分配器，分配器是新建和删除新建内存的方式，如果不写这个参数，默认使用new和delete

1. **模板类vector**

对应数组

存储了一组可随机访问的值

可以创建vector对象，进行vector对象的互相赋值，使用[]运算符来访问vector中的元素

使用DMA动态内存分配（DMA的特点是不用考虑内存问题，系统会进行自动处理

1. **迭代器iterator**

Iterator是一个广义指针，可以是指针，也可以是可对其执行类似指针操作的对象

1. **Iterator的部分功能**

超尾

push\_back()操作：把某元素添加到最末尾

Erase操作：（begin[0], begin[4]），其实本意是[ begin[0], begin[4] )

For\_each()

Random\_shuffle()

Sort()升序排列

1. **泛型编程generic programming**

**特点：**

关注的是算法

共同点是抽象和创建可重用代码

旨在编写独立于数据类型的代码

模板类使得能够按泛型定义函数或者类，STL通过通用算法更近了一步

迭代器使得算法独立于使用容器

1. **为什么使用迭代器**

为两种不同的数据表示实现find函数，double数组，double链表

算法上看：两个find函数不同

逻辑上二者相同：将值一次与容器中的每个值进行比较，知道找到匹配的为止

迭代器要能实现解除引用，赋值比较等功能

不同的容器一般会定义适合其自己的迭代器，以希望达到适用于不同的情况的容器可以使用同一个函数的效果

1. STL定义了**五种迭代器，并根据所需的迭代器类型对算法进行了描述**
2. 输入迭代器

来自容器的信息被视为输入

只读

支持++运算

Single-pass

单向迭代器，只能递增，不能倒退

1. 输出迭代器

用于将信息从程序传送给容器

只能接触引用让程序能够修改容器的值，不能读取（只写）

Single-pass

1. 正向迭代器

只支持++运算

递增后，仍然可以使用前面的迭代器解除引用以访问容器中的值

可读可写，也可以使其只读

1. 双向迭代器

同时支持两种（前缀和后缀）递减--运算符

并且具有双向迭代器所有特性

1. 随机访问迭代器

包括了双向迭代器的全部功能

支持随机访问功能

1. 
2. **概念concept，改进refinement，模型model**
3. **将指针用作迭代器**

迭代器是广义指针，指针满足所有迭代器的要求

Copy() //copy array to vector

ostream\_iterator

istream\_iterator

1. **改进：关联（附录G）**

将值与键关联在一起，用键来查找值，通常是用过某种树来实现的

1. Set

类似于集合，set中键唯一，因此不能存储对个相同的值

Set中，值类型与键相同

使用模板参数来制定要存储的值类型

1. Multiset

与set类似，但是有可能多个键的值相同

1. Map
2. Multimap

同一个键可能与多个值对应

1. **算法**
2. **STL，string类，函数和容器方法**

STL不包含string类，但设计时考虑了string类