C++标准库中的std::string类

C++标准库中的std::string类提供了大量用于处理字符串的成员函数和自由函数。这些接口覆盖了从字符串的构造、修改、查询到各种类型转换等多种功能。以下是一些常用的接口，但请注意，这并不是一个完整的列表，因为C++标准库会随着新的C++标准的发布而不断更新和扩展。

构造函数和析构函数

- string(): 默认构造函数
- string(const string& str): 复制构造函数
- string(const char* s): 从C字符串构造
- string(const char* s, size_t n): 从C字符串的一部分构造
- string(size_t n, char c): 用指定数量的某个字符构造
- ~string(): 析构函数

字符串修改

- void assign(const string& str): 分配新的字符串
- void assign(const char* s): 从C字符串分配
- void assign(const char* s, size_t n): 从C字符串的一部分分配
- void assign(size_t n, char c): 分配指定数量的某个字符
- void append(const string& str): 追加字符串
- void append(const char* s): 追加C字符串
- void append(const char* s, size_t n): 追加C字符串的一部分
- void append(size_t n, char c): 追加指定数量的某个字符
- void push_back(char c): 字符串尾追加字符
- void pop_back(): 删除字符串尾字符
- void resize(size_t n): 改变字符串大小
- void resize(size_t n, char c): 改变字符串大小，并用指定字符填充新位置
- void swap(string& str): 交换内容
- void replace(size_t pos, size_t len, const string& str): 替换子字符串
- void replace(iterator i1, iterator i2, const string& str): 替换子字符串
- void erase(size_t pos, size_t len): 删除子字符串
- iterator erase(iterator it): 删除单个字符或子字符串
- void clear(): 清空字符串内容

字符串查询

- size_t find(const string& str, size_t pos=0) const: 查找子字符串
- size_t find(const char* s, size_t pos=0) const: 查找C字符串
- size_t find(const char* s, size_t pos, size_t n) const: 查找C字符串的一部分
- size_t find(char c, size_t pos=0) const: 查找字符

- size_t rfind(const string& str, size_t pos=npos) const: 从末尾查找子字符串
- size_t rfind(const char* s, size_t pos=npos) const: 从末尾查找C字符串
- size_t rfind(const char* s, size_t pos, size_t n) const: 从末尾查找C字符串的一部分
- size_t rfind(char c, size_t pos=0) const: 从末尾查找字符
- size_t find_first_of(const string& str, size_t pos=0) const: 查找任意一个给定字符串中的字符
- size_t find_first_not_of(const string& str, size_t pos=0) const: 查找第一个不在给定字符串中的字符

子字符串

- string substr(size_t pos, size_t len) const: 获取子字符串

迭代器

- iterator begin(): 返回指向字符串开头的迭代器
- const_iterator begin() const: 返回指向字符串开头的常量迭代器
- iterator end(): 返回指向字符串末尾的迭代器
- const_iterator end() const: 返回指向字符串末尾的常量迭代器
- reverse_iterator rbegin(): 返回指向字符串逆序开头的逆序迭代器
- const_reverse_iterator rbegin() const: 返回指向字符串逆序开头的常量逆序迭代器
- reverse_iterator rend(): 返回指向字符串逆序末尾的逆序迭代器
- const_reverse_iterator rend() const: 返回指向字符串逆序末尾的常量逆序迭代器

容量

- size_t size() const: 返回字符串中字符的数量
- size_t length() const: 返回字符串的长度
- bool empty() const: 检查字符串是否为空
- size_t capacity() const: 返回当前容量
- void reserve(size_t n):

输入/输出操作

- std::basic_istream& operator>>(std::basic_istream& is, string& str): 从输入流读取字符串。
- std::basic_ostream& operator<<(std::basic_ostream& os, const string& str): 将字符串写入输出流。

字符串比较

- `int compare(const string& str) const noexcept`: 与另一个字符串比较。
- `int compare(size_t pos, size_t len, const string& str) const`: 与另一个字符串的一部分比较。
- `int compare(const char* s) const`: 与C字符串比较。
- `int compare(size_t pos, size_t len, const char* s) const`: 与C字符串的一部分比较。

数据访问

- `char& operator[](size_t pos)`: 返回指定位置的字符的引用。
- `const char& operator[](size_t pos) const`: 返回指定位置的字符的常量引用。
- `char& at(size_t pos)`: 返回指定位置的字符的引用，带有边界检查。
- `const char& at(size_t pos) const`: 返回指定位置的字符的常量引用，带有边界检查。
- `const char* data() const noexcept`: 返回指向字符串数据的指针。
- `const char* c_str() const noexcept`: 返回一个以null终止的C字符串。

内存管理

- `void shrink_to_fit()`: 请求减少内存容量以适应当前大小。
- `void reserve(size_t n)`: 为字符串分配至少能容纳n个字符的内存。
- `void resize(size_t n)`: 改变字符串的大小。
- `void resize(size_t n, char c)`: 改变字符串的大小，并用指定的字符填充新位置。

非成员函数重载运算符

- `bool operator==(const string& lhs, const string& rhs)`: 比较两个字符串是否相等。
- `bool operator!=(const string& lhs, const string& rhs)`: 比较两个字符串是否不相等。
- `bool operator<(const string& lhs, const string& rhs)`: 比较一个字符串是否小于另一个。
- `bool operator<=(const string& lhs, const string& rhs)`: 比较一个字符串是否小于或等于另一个。
- `bool operator>(const string& lhs, const string& rhs)`: 比较一个字符串是否大于另一个。
- `bool operator>=(const string& lhs, const string& rhs)`: 比较一个字符串是否大于或等于另一个。

其他非成员函数

- `void swap(string& lhs, string& rhs) noexcept`: 交换两个字符串的内容。
- `string to_string(int value)`: 将整数转换为字符串。
- `string to_string(unsigned value)`: 将无符号整数转换为字符串。

- `string to_string(long value)`: 将长整数转换为字符串。
- `string to_string(unsigned long value)`: 将无符号长整数转换为字符串。
- `string to_string(long long value)`: 将长长整数转换为字符串。
- `string to_string(unsigned long long value)`: 将无符号长长整数转换为字符串。
- `string to_string(float value)`: 将浮点数转换为字符串。
- `string to_string(double value)`: 将双精度浮点数转换为字符串。
- `string to_string(long double value)`: 将长双精度浮点数转换为字符串。 这些函数提供了对字符串进行操作的各种方法，从基本的字符操作到复杂的内存管理和输入/输出操作。由于`std::string`是一个模板类，它还支持与宽字符和窄字符相关的其他特化，例如`std::wstring`用于宽字符。