C++ 标准库 <string>

C++ 标准库（Standard Template Library, STL）是 C++ 的核心组成部分之一，提供了丰富的数据结构和算法。

<string> 是 C++ 标准库中用于处理字符串的头文件。

在 C++ 中，字符串是由字符组成的序列。<string> 头文件提供了 std::string 类，它是对 C 风格字符串的封装，提供了更安全、更易用的字符串操作功能。

要在 C++ 程序中使用 <string> 库，首先需要包含这个头文件：

```
#include <iostream>
#include <string>
```

基本语法

std::string 类的基本语法如下：

声明字符串变量：

```
std::string str;
```

初始化字符串：

```
std::string str = "Hello, World!";
```

使用 + 连接字符串：

```
std::string str1 = "Hello, ";
std::string str2 = "World!";
std::string result = str1 + str2;
```

常用成员函数

std::string 类提供了许多成员函数来操作字符串，以下是一些常用的成员函数：

- size()：返回字符串的长度。
- empty()：检查字符串是否为空。
- operator[]：通过索引访问字符串中的字符。
- substr()：获取子字符串。
- find()：查找子字符串在主字符串中的位置。
- replace()：替换字符串中的某些字符。

实例

下面是一个使用 <string> 库的简单实例，包括输出结果：

实例

```
#include <iostream>
#include <string>

int main() {
    // 声明并初始化字符串
    std::string greeting = "Hello, World!";
    std::cout << "Greeting: " << greeting << std::endl;

    // 使用 size() 获取字符串长度
    std::cout << "Length of the greeting: " << greeting.size() << std::endl;

    // 使用 empty() 检查字符串是否为空
    std::cout << "Is the greeting empty? " << (greeting.empty() ? "Yes" : "No") << st
d::endl;

    // 使用 operator[] 访问特定位置的字符
    std::cout << "Character at position 7: " << greeting[7] << std::endl;

    // 使用 substr() 获取子字符串
    std::string sub = greeting.substr(7, 5);
    std::cout << "Substring from position 7 with length 5: " << sub << std::endl;

    // 使用 find() 查找子字符串
    std::cout << "Position of 'World' in the greeting: " << greeting.find("World") <
< std::endl;

    // 使用 replace() 替换字符串中的部分内容
    // 替换 'World' 为 'C++'
    std::string modified = greeting;
    std::string::size_type pos = modified.find("World");
    if (pos != std::string::npos) {
        modified.replace(pos, 5, "C++"); // 从位置 pos 开始，替换 5 个字符为 "C++"
    }
    std::cout << "Modified greeting: " << modified << std::endl;

    return 0;
}
```

输出结果:

```
Length of the greeting: 13
Is the greeting empty? No
Character at position 7: W
Substring from position 7 with length 5: World
Position of 'World' in the greeting: 7
Modified greeting: Hello, C++!
```

std::string 成员函数汇总表

下面是一个常见的 std::string 成员函数的汇总:

函数名	描述	示例代码
size()	返回字符串的长度（字符数）。	std::cout << str.size();
length()	与 size() 相同，返回字符串的长度。	std::cout << str.length();
empty()	判断字符串是否为空。	std::cout << (str.empty() ? "Yes" : "No");
operator[]	访问字符串中指定位置的字符。	std::cout << str[0];
at()	访问字符串中指定位置的字符（带边界检查）。	std::cout << str.at(0);
substr()	返回从指定位置开始的子字符串。	std::string sub = str.substr(0, 5);
find()	查找子字符串在字符串中的位置。	std::cout << str.find("sub") << std::endl;
rfind()	从字符串末尾开始查找子字符串的位置。	std::cout << str.rfind("sub") << std::endl;
replace()	替换字符串中的部分内容。	str.replace(pos, length, "new_substring");
append()	在字符串末尾添加内容。	str.append(" more");
insert()	在指定位置插入内容。	str.insert(pos, "inserted");
erase()	删除指定位置的字符或子字符串。	str.erase(pos, length);
clear()	清空字符串。	str.clear();
c_str()	返回 C 风格的字符串（以 null 结尾）。	const char* cstr = str.c_str();
data()	返回指向字符数据的指针（C++11 及之后的版本）。	const char* data = str.data();
compare()	比较两个字符串。	int result = str.compare("other");

函数名	描述	示例代码
find_first_of()	查找第一个匹配任意字符的位置。	size_t pos = str.find_first_of("aeiou");
find_last_of()	查找最后一个匹配任意字符的位置。	size_t pos = str.find_last_of("aeiou");
find_first_not_of()	查找第一个不匹配任意字符的位置。	size_t pos = str.find_first_not_of("aeiou");
find_last_not_of()	查找最后一个不匹配任意字符的位置。	size_t pos = str.find_last_not_of("aeiou");

实例

```
#include <iostream>
#include <string>

int main() {
    std::string str = "Hello, World!";

    // size()
    std::cout << "Length: " << str.size() << std::endl;

    // empty()
    std::cout << "Is empty? " << (str.empty() ? "Yes" : "No") << std::endl;

    // operator[]
    std::cout << "First character: " << str[0] << std::endl;

    // at()
    std::cout << "Character at position 7: " << str.at(7) << std::endl;

    // substr()
    std::string sub = str.substr(7, 5);
    std::cout << "Substring from position 7 with length 5: " << sub << std::endl;

    // find()
    size_t pos = str.find("World");
    std::cout << "Position of 'World': " << pos << std::endl;

    // replace()
    str.replace(pos, 5, "C++");
    std::cout << "Modified string: " << str << std::endl;

    // append()
    str.append(" How are you?");
    std::cout << "Appended string: " << str << std::endl;

    // insert()
    str.insert(7, " Beautiful");
```

```
    std::cout << "String after insert: " << str << std::endl;

    // erase()
    str.erase(7, 10);
    std::cout << "String after erase: " << str << std::endl;

    // clear()
    str.clear();
    std::cout << "String after clear: " << (str.empty() ? "Empty" : "Not empty") << s
td::endl;

    // c_str()
    str = "Hello, C++!";
    const char* cstr = str.c_str();
    std::cout << "C-style string: " << cstr << std::endl;

    // compare()
    int cmp = str.compare("Hello, C++!");
    std::cout << "Comparison result: " << cmp << std::endl;

    // find_first_of()
    size_t pos_first_vowel = str.find_first_of("aeiou");
    std::cout << "First vowel at position: " << pos_first_vowel << std::endl;

    // find_last_of()
    size_t pos_last_vowel = str.find_last_of("aeiou");
    std::cout << "Last vowel at position: " << pos_last_vowel << std::endl;

    // find_first_not_of()
    size_t pos_first_non_vowel = str.find_first_not_of("aeiou");
    std::cout << "First non-vowel at position: " << pos_first_non_vowel << std::endl;

    // find_last_not_of()
    size_t pos_last_non_vowel = str.find_last_not_of("aeiou");
    std::cout << "Last non-vowel at position: " << pos_last_non_vowel << std::endl;

    return 0;
}
```

输出结果:

```
Length: 13
Is empty? No
First character: H
Character at position 7: W
Substring from position 7 with length 5: World
Position of 'World': 7
Modified string: Hello, C++!
Appended string: Hello, C++! How are you?
String after insert: Hello,  BeautifulC++! How are you?
String after erase: Hello, C++! How are you?
String after clear: Empty
C-style string: Hello, C++!
Comparison result: 0
```

```
First vowel at position: 1
Last vowel at position: 4
First non-vowel at position: 0
Last non-vowel at position: 10
```