补充：string类函数的妙用

`std::to_string` 是 C++11 引入的一个标准库函数，用于将数值类型（整数、浮点数等）转换为其字符串表示形式。它被声明在 `<string>` 头文件中。以下是 `std::to_string` 函数的声明：

```
namespace std {
    string to_string(int value);
    string to_string(long value);
    string to_string(long long value);
    string to_string(unsigned value);
    string to_string(unsigned long value);
    string to_string(unsigned long long value);
    string to_string(float value);
    string to_string(double value);
    string to_string(long double value);
}
```

每个重载函数接收一个不同类型的数值作为参数，并返回该数值的字符串表示形式。需要注意的是，`std::to_string` 生成的浮点数字符串可能会有较多的小数位数，如果需要控制小数位数，可以使用其他方法如 `std::ostringstream` 或 `printf` 风格的格式化。

具体代码实现：

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

int main() {
    double number = 12345.6789;
    std::string str = std::to_string(number);
    std::cout << "The string representation is: " << str << std::endl;
    return 0;
}
```

在 C++ 中，如果你需要控制浮点数转换为字符串时的小数位，可以使用 `std::ostringstream` 或 `printf` 风格的格式化来实现。以下是两种方法的具体代码实现：

使用 `std::ostringstream`

```
#include <iostream>
#include <sstream>
#include <iomanip> // for std::setprecision

int main() {
    double number = 12345.6789;
    std::ostringstream oss;

    // 设置小数位数，比如保留两位小数
    oss << std::fixed << std::setprecision(2) << number;
```

```
    std::string str = oss.str();
    std::cout << "The string representation with controlled precision is: " <<
str << std::endl;

    return 0;
}
```

使用 `printf` 风格的格式化

```
#include <iostream>
#include <cstdio> // for sprintf

int main() {
    double number = 12345.6789;
    char buffer[50];

    // 使用 sprintf 来格式化字符串，保留两位小数
    std::sprintf(buffer, "%.2f", number);

    std::string str(buffer);
    std::cout << "The string representation with controlled precision is: " <<
str << std::endl;

    return 0;
}
```

解释

- `std::ostringstream`：通过流操作符 `<<` 和 `std::setprecision` 来设置浮点数的精度。`std::fixed` 确保以固定小数点格式输出。
- `printf` **风格**：使用 `sprintf` 函数将浮点数格式化为字符串。`%.2f` 指定了输出两位小数。

这两种方法都可以让你灵活地控制浮点数转换为字符串时的小数位数。选择哪种方法可以根据你的编程风格和需求来决定。