C++ 标准库 < sstream >

在 C++ 编程中，处理字符串和数字之间的转换是一项常见的任务。

`sstream` 是 C++ 标准库中的一个组件，它提供了一种方便的方式来处理字符串流（可以像处理流一样处理字符串）。

`<sstream>` 允许你将字符串当作输入/输出流来使用，这使得从字符串中读取数据或将数据写入字符串变得非常简单。

定义

`sstream` 是 C++ 标准库中的一个命名空间，它包含了几个类，用于处理字符串流，这些类包括：

- `istringstream`：用于从字符串中读取数据。
- `ostringstream`：用于将数据写入字符串。
- `stringstream`：是 `istringstream` 和 `ostringstream` 的组合，可以同时进行读取和写入操作。

语法

使用 `sstream` 的基本语法如下：

```
#include <sstream>

// 使用istringstream
std::istringstream iss("some data");

// 使用ostringstream
std::ostringstream oss;

// 使用stringstream
std::stringstream ss;
```

从字符串读取数据

下面是一个使用 `istringstream` 从字符串中读取整数和浮点数的例子：

```
#include <iostream>
#include <sstream>

int main() {
  std::string data = "10 20.5";
  std::istringstream iss(data);

  int i;
  double d;

  iss >> i >> d;

  std::cout << "Integer: " << i << std::endl;
  std::cout << "Double: " << d << std::endl;
```

```
  return 0;
}
```

输出结果：

```
Integer: 10
Double: 20.5
```

向字符串写入数据

下面是一个使用 `ostringstream` 将数据写入字符串的例子：

```
#include <iostream>
#include <sstream>

int main() {
  std::ostringstream oss;
  int i = 100;
  double d = 200.5;

  oss << i << " " << d;

  std::string result = oss.str();
  std::cout << "Resulting string: " << result << std::endl;

  return 0;
}
```

输出结果：

```
Resulting string: 100 200.5
```

使用stringstream进行读写操作

下面是一个使用 `stringstream` 同时进行读取和写入操作的例子：

```
#include <iostream>
#include <sstream>

int main() {
  std::string data = "30 40.5";
  std::stringstream ss(data);

  int i;
  double d;

  // 从stringstream读取数据
  ss >> i >> d;

  std::cout << "Read Integer: " << i << ", Double: " << d << std::endl;

  // 向stringstream写入数据
```

```
    ss.str(""); // 清空stringstream
    ss << "New data: " << 50 << " " << 60.7;

    std::string newData = ss.str();
    //使用str()来实现字符串的写入。
    std::cout << "New data string: " << newData << std::endl;

    return 0;
}
```

输出结果：

```
Read Integer: 30, Double: 40.5
New data string: New data: 50 60.7
```

例：

```
#include <iostream>
#include <sstream>
#include <string>
using namespace std;
int main(){
    stringstream input;
    input<<"hello world";
    string a,b;
    input>>a>>b;
    cout<<a<<endl;
    cout<<b<<endl;
    return 0;
}
```

字符串中的str()：

`std::stringstream` 类提供了一个名为 `str` 的成员函数，用于**获取和设置流中的字符串内容**。这个函数有两个重载版本，一个是无参的，用于获取流中的字符串内容；另一个是带一个 `std::string` 参数的，用于设置流中的字符串内容。

获取字符串内容

当你调用无参的 `str` 函数时，它会返回一个 `std::string`，包含当前流中的所有内容。

```
#include <iostream>
#include <sstream>

int main() {
    std::stringstream ss;
    ss << "Hello, " << "world!";
    std::string result = ss.str(); // 获取流中的内容
    std::cout << result << std::endl; // 输出: Hello, world!
    return 0;
}
```

设置字符串内容

当你调用带 `std::string` 参数的 `str` 函数时，它会设置流的内容为指定的字符串，并清除之前的内容。

```
#include <iostream>
#include <sstream>

int main() {
    std::stringstream ss;
    ss << "Initial content.";
    std::cout << "Before: " << ss.str() << std::endl; // 输出: Before: Initial
content.

    ss.str("New content."); // 设置新的内容
    std::cout << "After: " << ss.str() << std::endl; // 输出: After: New content.
    return 0;
}
```

结合使用 `str` 和 `clear` 方法

在重用 `std::stringstream` 对象时，通常会结合使用 `str` 方法和 `clear` 方法。`str` 方法用于设置新的字符串内容，而 `clear` 方法用于重置流的状态（例如，清除错误标志）。

```
#include <iostream>
#include <sstream>

int main() {
    std::stringstream ss;
    ss << "123 456";
    int a, b;
    ss >> a >> b;
    std::cout << "a: " << a << ", b: " << b << std::endl; // 输出: a: 123, b: 456

    ss.clear(); // 重置流的状态
    //重置流的状态意味着清除流的状态标志（如错误标志、结束标志等），但并不改变流的内部缓冲区的内
容。换句话说，重置状态让流恢复到一个干净的状态，但原有数据仍然保留。
    ss.str(""); // 清空流的内容

    ss << "789 1011";
    ss >> a >> b;
    std::cout << "a: " << a << ", b: " << b << std::endl; // 输出: a: 789, b: 1011
    return 0;
}
```

总结

- `str()`：无参版本用于获取流中的字符串内容。
- `str(const std::string &s)`：带参数版本用于设置流中的字符串内容，并清除之前的内容。

通过使用 `str` 方法，你可以方便地获取和设置 `std::stringstream` 对象中的字符串内容，从而实现灵活的字符串操作。

总结

`sstream` 是 C++ 标准库中一个非常有用的组件，它简化了字符串和基本数据类型之间的转换。通过上述实例，我们可以看到如何使用 `istringstream`、`ostringstream` 和 `stringstream` 来实现这些转换。掌握这些技能将帮助你在 C++ 编程中更加高效地处理字符串数据。