

# 第 1 章 使用字符串

刘 卉

huiliu@fudan.edu.cn



# 前言

- 通过使用字符串了解C++的基础知识
  - 变量的声明和初始化
  - 输入
  - C++标准库中的string类型

# 1.1 输入

```
// ask for a person's name, and greet the person
#include <iostream>
#include <string>
int main()
{
    // ask for the person's name
    std::cout << "Please enter your first name: ";
    // read the name
    std::string name;    // define name
    std::cin >> name;    // read into name
    // write a greeting
    std::cout << "Hello, " << name << "!" << std::endl;
    return 0;
}
```

C:\Users\comet\Documents\C++\Examples\chapter01\greet.exe

Please enter your first name: Hui  
Hello, Hui!



## □ 变量: 有名字的对象

- 对象的类型隐式地说明了它的接口——可作用于该类型对象的一系列操作.
- `string`类型变量在定义时, 已经被隐式地初始化了.

`std::string name;` // `string`类将`name`初始化为空字符串

## □ 标准库使用'>>'和"`std::cin`"进行输入

`std::cin >> name;`

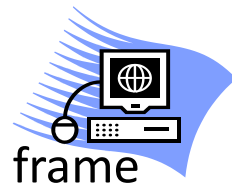
- 忽略输入中的前导空白符, 把其它字符读取到`name`中, 直至遇到空白符/文件结束标志.



## 1.2 为名字装框输出

Please enter your first name: Estragon

```
*****  
*                                     *  
* Hello, Estragon! *  
*                                     *  
*****
```



```
// ask for a person's name, and generate a framed greeting
```

```
#include <iostream>
```

```
#include <string>
```

```
int main() {
```

```
    std::cout << "Please enter your first name: ";
```

```
    std::string name;
```

```
    std::cin >> name;
```

```
    // build the message that we intend to write
```

```
    const std::string greeting = "Hello, " + name + "!";
```

```
    // build the second and fourth lines of the output
```

```
    const std::string spaces(greeting.size(), ' ');
```

```
    const std::string second = "* " + spaces + " *";
```

```
    // build the first and fifth lines of the output
```

```
    const std::string first(second.size(), '*');
```

```
    // write it all
```

```
    (next page)
```

```
    return 0;
```

```
}
```

```
Please enter your first name: Estragon
```

```
*****
```

```
*                                     *
```

```
* Hello, Estragon! *
```

```
*                                     *
```

```
*****
```

```
int main()
{
    .....
    // write it all
    std::cout << std::endl;
    std::cout << first << std::endl;
    std::cout << second << std::endl;
    std::cout << "*" << greeting << " *" << std::endl;
    std::cout << second << std::endl;
    std::cout << first << std::endl;
    .....
}
```

Please enter your first name: Estragon

```
*****
*                                     *
* Hello, Estragon! *
*                                     *
*****
```

## □ 实现策略:

- 读取名字→组成问候语→建立每行输出

□ `const std::string greeting = "Hello, " + name + "!";`

- 1) 显式的初始化: 定义变量时给定初值.

- 2) 用'+'连接string对象和字符串常量(或string对象).

- 操作符'+'对不同类型的操作数有不同含义→操作符'+'被重载

- 重载时, 操作符的结合性不会发生变化



```
const std::string greeting = "Hello, " + name + "!";
```

### 3) const变量

- const变量必须在定义时初始化.
- 保证变量在它的生存期内, 值不会改变.
- 明确指出变量不会改变, 让程序更容易理解.

#### □ const变量 vs 常量

- 初始化const变量的值可以不是常量.
- 不能把greeting定义为常量.

- `const std::string spaces(greeting.size(), ' ');`
  - 在定义中使用圆括号初始化变量: 要求系统根据'()'中的表达式来构造变量.  
e.g. `int i(100); //int i = 100;`
  - `greeting.size()`: 调用string类的成员函数, 返回string变量的有效字符个数.
  - `' '`: 空格字符常量.
  - 使用一个整数值n和一个字符值c构造一个string对象→c的n份复制.

## □ string类的操作

- `std::string s` 定义一个`std::string`类的变量`s`, 并初始化为空.
- `std::string t=s` 定义`t`并用`s`初始化`t`, `s`是`string`对象或字符串常量.  
`std::string t(s)`
- `std::string z(n,c)` 定义`z`并用字符`c`的`n`份复制初始化`z`.
- `os << s` 把`s`不做任何格式变化写入`os`, 运算结果是`os`(链式输出).
- `is >> s` 读取单词, 存入`s`, 运算结果是`is`(链式输入).
- `s + t` `s`或`t`都可以(但不能同时)是字符串常量或`char`型值.