

# 现代C++：标准库与容器

COMP250205：计算机程序设计

李昊

[hao.li@xjtu.edu.cn](mailto:hao.li@xjtu.edu.cn)

西安交通大学计算机学院

# C++标准库: `string`

# string: 标准库内置的字符串类型

包装C风格字符串，并通过一系列重载的操作符，提供极佳的可操作性和较佳的性能

string库



```
#include <string>
#include <iostream>
int main()
{
    using namespace std;
    string s1;
    string s2 = s1; // string s2(s1);
    string s3 = "hiya"; // string s3("hiya");
    string s4(10, 'c');

    cout << s3 << s4 << endl;
    return 0;
}
```

声明一个string对



s2是s1的副本



使用字符串常量初始化



长度为10，字符都是c



直接cout字符串



# string上的操作 - 读写

## 使用cin、cout读写string对象

忽略开头空白，  
直到下一个空白



```
int main()
{
    string s;
    cin >> s;
    cout << s << endl;
    return 0;
}
```

连续cin，从缓冲区里连续读入字符串



```
int main()
{
    string s1, s2;
    cin >> s1 >> s2;
    cout << s1 << s2 << endl;
    return 0;
}
```

# string上的操作 - 读取整行

使用`getline(cin, string)`获取完整行（不包括  
`\n`）

持续读一整行，直到EOF →

```
int main()
{
    string line;
    while (getline(cin, line)) {
        cout << line << endl;
    }
    return 0;
}
```

# string上的操作 - 判断长度

判断string对象的长度: `empty()` 和 `size()` 方法

判断是否string为空



输出string对象的长度



```
int main()
{
    string line;
    getline(cin, line);
    if (line.empty()) {
        cout << "An empty line.";
        cout << endl;
    }
    else {
        cout << "A line with " << line.size();
        cout << " characters." << endl;
    }
    return 0;
}
```

# string上的操作 - 比较

## 对string对象进行比较

**false**, 相同字符, 长大于短 →  
**false**, 不同字符, 比ASCII →  
**true** →

```
#include <string>
#include <iostream>
int main()
{
    string s1 = "Hello";
    string s2 = "Hello, world!";
    string s3 = "Hiya";

    cout << s1 > s2 << endl;
    cout << s1 > s3 << endl;
    cout << s3 > s2 << endl;

    return 0;
}
```

# string上的操作 - 相加

## string对象相加

连接两个string对象 →

String对象可以与字符常量或  
字符串常量相加（连接） →

按照结合律，进行两次加法 →

错误！按照结合律，两个字  
符串常量相加是不被允许的 →

```
int main() {  
    string s1 = "hello";  
    string s2 = "world";  
    string s3 = s1 + s2;  
  
    string s4 = s1 + "," + s2 + '\n';  
  
    string s5 = s1 + "!" + "\n";  
  
    string s6 = "hello," + "world!" + s1;  
  
    cout << s6;  
    return 0;  
}
```



# string上的操作 - QUIZ

编写一段程序读入两个字符串，比较其是否相等并输出结果；如果不相等，输出较大的字符串

改写程序，比较是否等长，如果不等长，输出较长的字符串

编写一段程序，读入多个字符串，以#结束。以空格连接这些字符串，输出连接后的字符串

# 处理string对象 - 访问单个字符

与字符数组类似，**string**对象提供**下标访问**

s1[0]返回第1个元素 →

s1[5]仍然是空字符 →

```
int main() {  
    string s1 = "hello";  
  
    if (s1[0] != 'h' && s1[0] != 'H') {  
        cout << "Something weird happened" << endl;  
    }  
  
    if (s1[5] == '\0') {  
        cout << "an empty char" << endl;  
    }  
    return 0;  
}
```

# 处理string对象 - 字符处理函数

**cctype**头文件定义了对单个字符的判断处理函数

函数原型	意义
<code>isalpha/isdigit</code>	如果字符是字母/数字时为真
<code>isupper/islower</code>	如果是大写/小写字母时为真
<code>toupper/tolower</code>	转换为大写/小写字母

# 使用字符处理函数 - QUIZ

从键盘接收一段字符，数出其中大写字母、小写字母和数字的个数

使用size()控制循环次数 →

使用下标访问字符，使用字符处理函数判断 →

```
int main() {
    string str;
    getline(cin, str);
    int u = 0, l = 0, d = 0;
    for (int i=0; i<str.size(); ++i) {
        if (isupper(str[i]))
            u++;
        if (islower(str[i]))
            l++;
        if (isdigit(str[i]))
            d++;
    }
    cout << u << l << d << endl;
    return 0;
}
```

# 处理string对象 - 循环遍历

与字符数组一样使用**for/while+**下标方式访问

更现代的做法：范围**for**循环

访问序列中每一个元素的变量      序列对象（数组、string等）

```
for (declaration: expression)  
    statement
```

The diagram shows the syntax of a range for loop. The word 'for' is in blue, and the rest of the code is in red. A box surrounds the entire code snippet. Three red arrows point from the text above to the code: one from '访问序列中每一个元素的变量' to 'declaration', one from '序列对象（数组、string等）' to 'expression', and one from '循环体' to 'statement'.

↑  
循环体

# 使用范围for循环 - 1

使用**c**对**string**对象进行范围**for**循环



```
string s = "Hello, World!";  
for (char c : s) {  
    cout << c << ' ';  
}
```

使用**num**对数组对象进行范围**for**循环



```
int arr[] = {1, 2, 3, 4, 5};  
for (int num : arr) {  
    cout << num << ' ';  
}
```

# 使用范围for循环 - 2

把s中的每个字符改成大写 →

```
string s = "Hello, World!";  
for (char c : s) {  
    c = toupper(c);  
}  
cout << s << endl;
```

这里的c为什么不能修改s?

# 引用

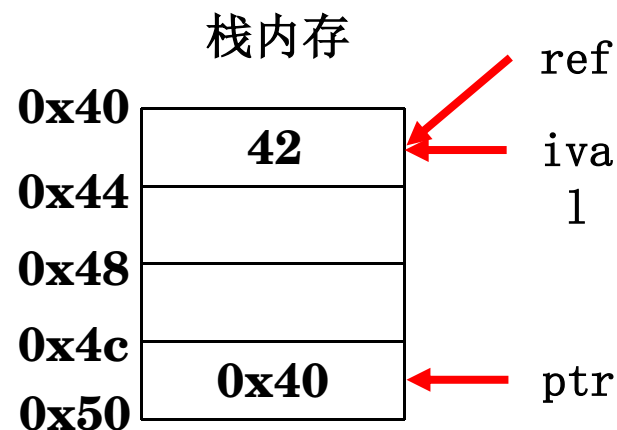
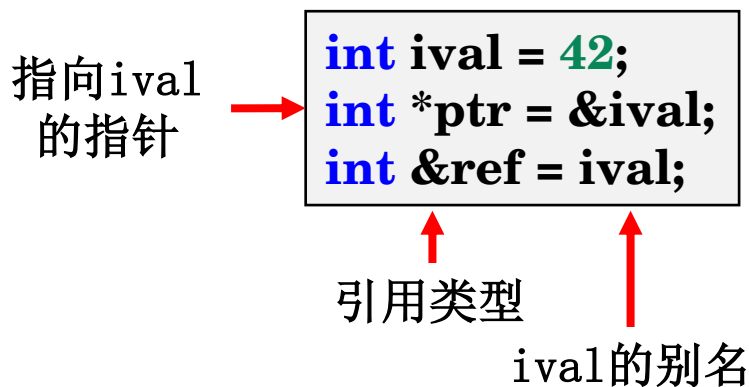


# 引用：另外一种访问数据的方法

变量名：以变量名访问数据

指针：以地址访问数据

引用：是变量的别名

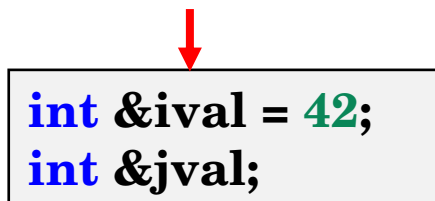


# 引用必须与一个变量绑定

引用不是对象：只是一个名字，没有独立的存储

引用必须初始化绑定到变量，不存在空引用

错误！引用需要绑定到一个变量（左值）



```
int &ival = 42;  
int &jval;
```

错误！没有绑定到任何一个变量

# 三种访问数据的方法

---

	按名访问	动态生存周期	传递参数需要复制
变量名	是	否	是
指针	否	是	否
引用	是	否	否

# 使用范围for循环 - 3

`for (char &c : s)`  
把s中的每个字符改成大写



```
string s = "Hello, World!";  
for (char c : s) {  
    c = toupper(c);  
}  
cout << s << endl;
```

这里的c为什么不能修改s?

使用引用访问每一个元素

# C++标准库: `vector`

# vector: 标准库内置的数组类型

**类模板**。包装动态数组，并通过一系列重载的操作符，提供**极佳的可操作性和较佳的性能**

<b>vector库</b>	→	<b>#include &lt;vector&gt;</b>
		<b>#include &lt;iostream&gt;</b>
		<b>int main()</b>
		<b>{</b>
		<b>using namespace std;</b>
声明一个 <b>vector</b> 对	→	<b>vector&lt;int&gt; v1;</b>
<b>v2</b> 是 <b>v1</b> 的副本	→	<b>vector&lt;int&gt; v2 = v1;</b>
初始化 <b>v3</b>	→	<b>vector&lt;int&gt; v3 = {1, 2, 3};</b>
长度为 <b>10</b> ，数字都是	→	<b>vector&lt;int&gt; v4(10, 42);</b>
<b>42</b>		
使用范围 <b>for</b> 循环打印	→	<b>for (int i : v3) cout &lt;&lt; i &lt;&lt; " ";</b>
		<b>return 0;</b>
		<b>}</b>

# 向vector中添加元素

使用**push\_back**向vector**尾部**添加元素

声明一个**vector**对象 →  
将**word**添加到**text**尾部 →  
使用引用，避免复制 →

```
#include <vector>
#include <string>
#include <iostream>
int main()
{
    using namespace std;
    string word;
    vector<string> text;
    while (cin >> word) {
        text.push_back(word);
    }
    for (string &w : text) cout << w << " ";
    return 0;
}
```

# 从vector中删除元素

使用`pop_back`从vector尾部删除元素

使用`clear`将vector清空

访问最后一个元素 →  
删除最后一个元素 →

清空所有元素 →

```
int main() {  
    std::vector<int> vec = {1, 2, 3, 4, 5};  
    while (vec[vec.size() - 1] > 3) {  
        vec.pop_back();  
    }  
    for (int i : vec) cout << i << " ";  
    cout << endl;  
    vec.clear();  
    for (int i : vec) cout << i << " ";  
    cout << endl;  
    return 0;  
}
```



# vector上的操作

函数原型	意义
<code>v.push_back(e)</code>	在尾部添加一个元素e
<code>v.pop_back()</code>	删除最后一个元素
<code>v.empty()</code>	如果不含有任何元素，返回真
<code>v.size()</code>	返回v中元素的个数
<code>v[n]</code>	访问第n个元素
<code>v1 == v2</code>	元素数量相等且每个元素相等为真
<code>v1 = v2</code>	用v2中元素的拷贝替换v1中的元素

# 使用vector - 1

有一组学生成绩，以60分为及格线，统计及格与不及格人数

一共分为2段成绩

不处理大于100的错误输入

判断成绩区间

范围for循环

```
int main() {  
    vector<unsigned> scores(2, 0);  
    unsigned grade;  
    while (cin >> grade) {  
        if (grade <= 100) {  
            if (grade < 60) ++scores[0];  
            else ++scores[1];  
        }  
    }  
    for (unsigned g : scores) {  
        cout << g << " ";  
    }  
    cout << endl;  
    return 0;  
}
```

# 使用vector - 2

有一组学生成绩，以10分为一分数段，统计每个分数段各有多少学生

一共分为11段成绩 →

不处理大于100的错误输入 →

利用整数除法计算下标 →

范围for循环 →

```
int main() {  
    vector<unsigned> scores(11, 0);  
    unsigned grade;  
    while (cin >> grade) {  
        if (grade <= 100) {  
            ++scores[grade/10];  
        }  
    }  
    for (unsigned g : scores) {  
        cout << g << " ";  
    }  
    cout << endl;  
    return 0;  
}
```

# 使用vector - 3

有一组学生成绩，如果某个学生成绩为不及格，  
则补录一个开平方乘以10的成绩作为补考成绩

```
int main() {  
    vector<unsigned> scores{11, 22, 33, 44, 55, 66, 77};  
  
    for (unsigned g : scores) {  
        if (g < 60) {  
            scores.push_back(sqrt(g)*10);  
        }  
    }  
    for (unsigned g : scores) cout << g << " ";  
    cout << endl;  
    return 0;  
}
```

范围for循环 →

补录成绩 →

错误！在范围for循环内部改变容器大小

# 迭代器

# 容器的通用访问方法：迭代器

只有顺序/连续存储的容器才可以用下标访问容器

迭代器：所有容器均提供统一通用的遍历方法

迭代器与指针使用方法类似，支持比较、算术加减

**auto**表示所有可以被推断出的类型

获得**string**的头迭代器

只要迭代器不等于尾迭代器

使用解引用方式访问/修改元素

```
#include <iostream>
#include <string>
using namespace std;

int main() {
    string s = "some string";
    auto it = s.begin();
    for (; it != s.end(); ++it) {
        *it = toupper(*it);
    }
    cout << s << endl;
    return 0;
}
```

# 使用迭代器 - 1

有一组学生成绩，以10分为一分数段，统计每个分数段各有多少学生

使用迭代器访问对应元素 →

```
int main() {  
    vector<unsigned> scores(11, 0);  
    auto beg = scores.begin();  
    unsigned grade;  
    while (cin >> grade) {  
        ++*(beg+grade/10);  
    }  
    for (auto c : scores) cout << c << " ";  
    cout << endl;  
    return 0;  
}
```

# 使用迭代器 - 2

在26个字母当中找到给定的字母

当前头迭代器 →  
当前尾迭代器 →  
当前中心迭代器 →  
是否找到了元素? →  
移动迭代器 →

```
int main() {  
    string text =  
    "abcdefghijklmnopqrstuvwxyz";  
    char ch = 's';  
    auto beg = text.begin();  
    auto end = text.end();  
    auto mid = text.begin() + (end-beg)/2;  
    while (mid != end && *mid != ch) {  
        if (ch < *mid) end = mid;  
        else beg = mid + 1;  
        mid = beg + (end - beg)/2;  
    }  
    cout << *mid << endl;  
    return 0;  
}
```



# 内容总结

## C++的标准库

提供了极佳的易用性和较好的性能

引用：与变量名绑定的别名，提供另一种访问模式

string和vector：提供对字符串和动态数组的封装

迭代器：容器的通用访问方式