

第三章

字符

在琴弦上轻奏音乐。

- f. s. 艾略特, 《荒原》, 1922 年

编程的抽象

C++

埃里克·尼伯茨

3.1 在 C 和 C+语言中使用字符串

+ 3.2 字符和<cctype>

库 3.3 c 风格的字符串和<cstring>

在 C+中使用字符串作为抽象值

+ 3.5 字符串操作 3.6 修改内容

字符串的操作 3.7 编写字符串应用 3.8 The

c++标准库简介

- 编写的类和函数的集合

核心语言，也是 c++ ISO 标准本身的一部分。

c++标准库的特性在内部声明

STD 命名空间

- 容器:vector、queue、stack、map、set 等。

- 通用:算法、函数式、迭代器、内存等。

- 字符串

- 流和输入/输出:iostream, fstream, sstream 等。

- 本地化

- 语言支持

- 线程支持库

- 数值库



在 C 和 c++语言中使用字

- 现在文本数据和数值数据一样重要。几乎
你用任何现代语言编写的任何程序都是可能的
在某些时候使用字符串数据，即使它只是用来显示
对用户或对结果进行标记的说明。

从概念上讲，字符串只是一个字符序列，
这正是 C 语言中字符串的实现方式。

- 作为一种新设计的语言，特别是扩展 C 的语言
采用面向对象的编程范式，c++
支持更高层次的字符串视图，如对象。
- 显示 C 和 c++在字符串上使用的不同策略
不同编程范式之间的差异。

字符

- C 和 c++ 都使用 ASCII(美国标准代码信息交换)作为它们的字符编码表示。因此，数据类型 `char` 适合放在单个八位字节。
 - 只有 256 个可能的字符，ASCII 是不够的。代表世界上使用的许多字母。在大多数现代语言，ASCII 已经被取代 `Unicode`，允许更大数量的字符。
 - 尽管 ASCII 编码的弱点是在设计 c++ 时，明确改变了定义。考虑到保留 C 作为一个子集的决定，`char` 是不可能的。
- c++ 库定义了 `wchar_t` 类型来表示“宽”字符”允许更大的范围。的细节 `Wchar_t` 类型超出了本文的范围。我们将坚持用传统的 `char` 类型。

十二月Hx十月Char

00 000 null (nul1)	
1 1 001 soh	(航向开始)
2 2 002 STX	(正文开头)
3 3 003 etx	(文末)
4 4 004 eot	传输结束)
5 5 005 Enq	(查询)
6 6 006 ack	(承认)
7 7 007 bel	(钟)
8 8 010 b	(退格)
9 9 011 标签	(水平选项卡)
10 a 012 lf	(NL换行, 新行)
11 b 013 vt	(垂直制表符)
12 c 014 ff	(NP表单提要, 新页面)
13 d 015 cr	(回车)
14 E 016 s0	(移出)
15 f 017 si	(转变)
16 10 020 dle	(数据链路转义)
17 11 021 dc1	(设备控制1)
18 12 022 dc2	(设备控制2)
19 13 023 dc3	(设备控制3)
20 14 024 dc4	(设备控制4)
21 15 025 nak	(否定应答)
22 16 026 syn	(同步闲置)
23 17 027 ETb	(trans结束。块)
24 18 030 即可	(取消)
25 19 031 em	(介质结束)
26 1a 032 sub	(替换)
27 1b 033 esc	(逃)
28 1c 034 fs	(文件分隔符)
29 1d 035 gs	(集团分隔符)
30 e 036 rs	(记录分隔符)
31 f 037 us	(单位分离器)

十二月Hx十月Html 空空的

32 20 040 Spac	
33 21 041 !! "	
34 22 042 "	
35 23 043 ##36 24 044	
$\$ 37 25 045 %	
38 26 046 && 39 27 047	
'	
40 28 050 #40;(41 29	
051)	
42 2a 052 *	*
43 2b 053 +	+
44 2c 054 ,	,
45 2d 055 -	-
46 2e 056 .	.
47 2f 057 /	/
48 30 060 00 49 31	
061 11 50 32 062	
22 51 33 063 33	
52 34 064 44 53 35	
065 55 54 36 066	
66 55 37 067 77	
56 38 070 88 57 39	
071 99	
58 3a 072 ::	
59 3b 073 ;	;
< 61 3d 075 =	=
62 3e 076 >	
63 3f 077 ??	












12月Hx 10月Html Chr

e 64 40 100 @	
65 41 101 A	一个
66 42 102 BB	
67 43 103 C	C
68 44 104 D	D
69 45 105 E	E
70 46 106 F	F
71 47 107 G	G
72 48 110 H	H
73 49 111 I	我
74 4a 112 J	J
75 4b 113 K	K
76 4c 114 L	l
77 4d 115 M米	
78 4e 116 NN	
79 4f 117 O	0
80 50 120 P	p
81 51 121 Q	o
82 52 122 R	R
83 53 123 S	年代
84 54 124 T	T
85 55 125 U	U
86 56 126 V	V
87 57 127 W	W
88 58 130 X	X
89 59 131 Y	Y
90 5a 132 Z	Z
91 5b 133 [l
92 5c 134 \	—
93 5d 135]	
94 5e 136 ^	^
95 5f 137 _	—

12月Hx 10月Html Chr

96 60 140 `	,
97 61 141 a	a
98 62 142 bb	
99 63 143 c	c
100 64 144 dd	
101 65 145 ee	
102 66 146 ff	
103 67 147 gg	
104 68 150 hh	
105 69 151 i我	
106 6a 152 jj	
107 6b 153 kk	
108 6c 154 ll	
109 6d 155 m米	
110 6E 156 nn	
111 6f 157 oo	
112 70 160 pp	
113 71 161 q问	
114 72 162 rr	
115 73 163 s年代	
116 74 164 tt	
117 75 165 uu	
118 76 166 vv	
119 77 167 wW	
120 78 170 xx	
121 79 171 yY	
122 7a 172 zZ	
123 7b 173	
124 7c 174 |我	
125 7d 175	
126 7e 176 ~	~
127 7f 177 ▽	

来源:www.LookupTables.com

128		144	E	160	d	176		192	l	208	我	224	8	240	3
129	我	145		161	我	177		193	H	209	〒	225	3.	241	±
130	e	146	E	162	o	178		194	T	210	T	226	Γ	242	艾尔
131	ㄣ	147	o	163	u	179		195	卜	211	F	227	π	243	6
132		148	o	164	n	180	└	196	—	212	l	228	W	244	f
133		149	o	165	N	181	└	197	+	213	F	229	b	245	J
134		150	u	166	▪	182	└	198	卜	214		230	ㄣ	246	÷
135	5	151	u	167	.	183	F	199	我	215		231	l	247	Σ
136	e	152	y	168	心	184	└	200	J	216	丰	232	Φ	248	°
137	ë	153	O	169	└	185	└	201	ㄣ	217	J	233	C	249	.
138	e	154	U	170	└	186		202	莱托	218	└	234	Ω	250	.
139	我	155	☼	171	1	187	7	203	特遣部队	219		235	δ	251	N
140	←	156	E	172	4	188	J	204	└	220		236	8	252	一个
141		157	𐄌	173	我	189	J	205	=	221		237	中	253	2
142	一个	158	B	174	«	190	└	206	+	222		238	τ	254	
143	一个	159	f	175	»	191	1	207	±	223		239		255	

来源:www.LookupTables.com

单字符

使用单字符类型的变量:

```
char ch;  
Char ch = 'a';char ch  
= 97;
```

练习:下面的等价语句是什么:

```
Char ch = 32;
```

```
Char ch = ' ';
```

- 一些字符是特殊的控制字符:

```
字符 ch = 9;// '\t' (Tab) char ch = 10;//  
'\n'(换行)char ch = 13;// '\r'(返回)
```

用于换行, Mac 使用\r, Unix 使用\n, Windows 使用
使用\r\n。

<cctype> (ctype.h)接口

Bool isdigit(char ch)

确定指定的字符是否是数字。

Bool isalpha(char ch)

确定指定的字符是否是字母。

Bool isalnum(char ch)

确定指定的字符是字母还是数字。

Bool 较低(char ch)

确定指定的字符是否是小写字母。

Bool isupper(char ch)

确定指定的字符是否为大写字母。

Bool isspace(char ch)

确定指定的字符是否为空白(空格和制表符)。

Char tolower(Char ch)

将 ch 转换为其等效的小写形式(如果有的话)。如果没有, ch 将原原不动地返回。

c 风格字符串的遗产

- 在 C 语言中，我们可以表示字符串，即。，字符序列，as 字符类型元素的普通数组：

```
char 装运箱  
[10];  
Char cstr[] = "hello";
```

- 如果在序列的前后加上双引号字符，就得到了所谓的 C 字符串字面量(const **char**[]类型)。字符存储在一个字节数组中，以一个 ASCII 值为 0 的空字节结束。例如，C 字符串“hello, world”中的字符是这样排列的：

h	和	l	l	o	,		在	o	r	l	d	\
0	1	2	3	4	5	6	7	8	9	10	11	0

C 字符串中的字符位置由索引标识从 0 开始，延伸到比字符串长度小 1 的索引。

c 风格字符串的遗产

- 问题: “a” 和 “a” 是一样的吗?

“a” 是一个字符串字面量, 包含一个 “a” 和一个空终止符 '\0' . 因此是一个 2 字符数组。

调用 C 字符串函数

<cstring> (string.h)接口提供了很多函数
可以用来操作 C 字符串。例如，如果你想
确定 C 字符串的长度 `cstring`，可以使用
下面的函数 `strlen`:

```
Char cstr[] = "hello";  
Int len = strlen(cstr);
```

cstr 有
足够的空间
容纳“世界”？

- 如果你想给一个 C 字符串 `cstr` 赋值，你可以使用
下面的函数 `strcpy`:

```
strcpy(装运箱, “世
```

- 然而，你不能直接给一个 C 字符串赋值
除了初始化:

```
CSTR = "世界";  
Cstr[] = "世界";
```



<cstring> (string.h)接口

复制:

memcpy	内存复制块(函数)
memmove	移动内存块(函数)
strcpy	复制字符串(函数)
strncpy	从字符串(函数)中复制字符

连接:strcat

strcat	串接字符串(函数)
strncat	从字符串(函数)中追加字符

比较:

memcmp	比较两个内存块(函数)
strcmp	比较两个字符串(函数)
strcoll	使用locale (function)比较两个字符串
strncmp	比较两个字符串的字符(函数)
strxfrm	使用locale (function)转换字符串

搜索:memchr

strchr	在内存块中查找字符(函数)
strcspn	查找字符串(函数)中的字符
strpbrk	
strrchr	查找字符串中最后出现的字符(函数)
strspn	查找子字符串(函数)
strstr	
strtok	将字符串拆分为令牌(函数)

<cstring> (string.h)示例

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

因为<字符串>

```
Int main() {
```

```
    char 装运箱  
    [80];
```

```
    strcpy (cstr, “这些”);strcat(装  
    运箱,  
    “字符串”);Strcat (cstr,  
    “ ”、
```

```
    " Length = " << strlen(装运
```

```
}
```

<cstring> (string.h) 练习

练习:输出是什么?

```
#include <iostream> #include
<cstring>使用 std::cout;
使用 std:: endl;

Int main() {char
    Cstr [] = "hello";Cout << CSTR
    endl
        << sizeof CSTR << endl
        << strlen(cstr) << endl;
    Strcpy (cstr, " hello world" );Cout <<
    CSTR <<
    endl
        << sizeof CSTR << endl

}
```

Sizeof x 回报

实际内存
变量 x 的大小。



使用字符串作为抽象值

- 从第一个程序中显示的文本开始，屏幕上的消息“hello, world”，你一直在使用字符串与用户通信。

到目前为止，你还不知道 `c++` 是如何表示的字符串，或者你可能如何操作组成字符串的字符。

- 与此同时，你不知道这些事情没有影响你有效使用字符串的能力吗，因为你已经能够把弦整体地想象成如果它们是原始类型。

对于大多数应用程序，你所拥有的字符串的抽象视图坚持到现在正是正确的一种。在里面，字符串是非常复杂的对象，其细节是最好隐藏起来。

使用字符串作为抽象值

作为设计的结果，c++必须保留旧的字符类型和字符串模型继承自它的前身，后者是通过在 c++标准库中包含 C 标准库，如<cctype> (ctype.h);**因为<字符串> (string.h)。**

- 作为一种新设计的语言，特别是扩展 C 的语言采用面向对象的编程范式，c++支持更高层次的字符串视图，如对象。

一个 c++ 库 <string> 提供了一个方便的抽象通过将 string 转换为字符串来处理字符串**类，其方法隐藏了底层的复杂性。**

类是支持对象的数据类型的术语面向编程范式。所应用的操作类的实例称为方法。

锻炼

下列语句的意思是什么？

`#include <cstring>` //使用的 C 字符串库

在 c++中`#include <string.h>` // C 字符串库，
C 中使用，c++中可接受`#include`

`"string.h"` //有些编译器可能仍然会找到 C
字符串库，除非你用

你自己，这会引起冲突`#include <string>` // C++
字符串库#包含 `"string.h"` //不正确，除非你
自己定义的 `string.h` `#include <string.h>` // most
即使你自己定义了 `string.h`，也可能会报错



Hello Name 程序

图3-1 “Hello World” 程序的交互式版本

```
/*  
 * 文件:HelloName.cpp  
 * -----  
 * 这个程序通过询问扩展了经典的“Hello world”程序  
 * 用户获取名称，然后将其用作问候语的一部分。  
 * 这个版本的程序将整行读入name，而不仅仅是第一个单词。  
 */
```

```
#include <iostream>使用命名空  
间std;
```

```
Int main(){字符串名称;
```

```
cout < "Enter your全名:";
```

```
Cin>>名称;
```

```
cout << "Hello, " << name << "!" << endl;
```

```
返回0;
```

string 类中的操作符

str[i] 我]
返回 str 的第 i 个字符，对 str[i]赋值会改变该字符。
s1 + s2
返回一个由 s1 和 s2 连接而成的新字符串。
s1 = s2;
s1 += s2; 将 s2 追加到末尾
s1。s1 == s2(同样对于<, <=, >, >=, 和!=) 与字符串按字典序进行比较。
str.c_str 返回一个 c 风格的字符数组，其中包含与 str 相同的字符。

与大多数语言不同，c++允许类重新定义其含义
标准操作符的含义。结果，出现了一些字符串操作，如
As +用于连接，被实现为操作符(重载)。

- 要将 c++字符串对象转换为 C 字符串字面量，只需 apply
将 c_str 方法应用于 c++字符串。

拼接和 C 字符串

学习过 Python 的人应该知道，+ 操作符是拼接 for 的方便快捷方式 字符串对象，由合并两个字符串组成 结束时不插入任何字符。

在 Python 中，+ 操作符经常用于组合元素 作为 print 调用的一部分，如

```
打印(“你” + name + "!");
```

• 在 c++ 中，使用 << 操作符可以得到相同的结果：

```
cout << "Hello, " << name << "!" << endl;
```

你可能会会有其他想法，但不能总是使用 这个语句中的 + 操作符，取决于是否命名 是 C 字符串(字面量)还是 c++ 字符串对象。

```
cout << "Hello, " << 名字 << "!" << endl;
```



C 字符串字面量 vs. c++字符串对象

- c++ 自动将 C 字符串字面量转换为 c++ 字符串对象时，编译器可以确定你需要的是一个 c++字符串对象：

```
String STR = "hello,
```

- 相比之下，c++不允许你写这样的声明：

```
字符串 STR = "hello" + "世
```



+运算符不能应用于 C 字符串字面量。得到
围绕这个问题，你可以显式转换字符串字面量
通过在字面量上调用 string 来转换为 string 对象：

```
String STR = String ("hello") + "世
```

```
字符串 STR = "hello" + String ("， ") +
```

```
字符串 STR = "hello" + 字符串(“世
```



C 字符串字面量 vs. c++字符串对象

练习:输出是什么?

```
# include <
iostream >

Int main() {
    char name1[] = "射线";
    std::string name2 = "Ray";
    std::cout << "Hello, " std::cout
    << "Hello, " std::cout <<
    "Hello,
    << name1 << std::endl;
    << name2 << std::endl;
    + name1 << std::endl;
    + name2 << std::endl;
}
```



string 类中的常用方法

str.length
()

返回字符串 str 中字符的个数。

str.at(索引)

返回位置索引处的字符;大多数客户端使用 str[index]代替。

len str.substr (pos)

返回 str 的子字符串, 从 pos 开始, 一直到 len 字符。

pos str.find (ch)

返回第一个包含 ch 的索引 > pos, 如果没有找到, 则返回 string::npos。

调用字符串方法

因为 `string` 是一个类，所以最好在中考虑它的方法
将消息发送给特定对象的术语。对象要
哪一种信息被发送被称为接收者，而一般呢
发送消息的语法看起来像这样：

```
receiver.name(参数);
```

- 例如，如果你想确定一个字符串的长度
Str，将 **len** 设置为的语句的面向对象版本
因此，字符串对象 **STR** 的长度为

```
Int len = str.length();
```

- 你可能还想使用我们有的 `strlen` 函数
之前用在 C 字符串字面量上：

```
Int len = strlen(str);
```

毕竟，Python 两者都有：`len(str)`或 `str.__len__()`。



调用字符串方法

练习:如何确定字符串的长度 str?

面向对象的版本:

```
Int len = str.length();
```

- 如果你必须使用 C 中的 strlen 函数:

```
Int len = strlen(str.c_str());
```

<string>库示例

```
# include <
iostream >

Int main() {
    char str [80];
    Strcpy (str, “这些”);
    Strcat (str, “字符串”);
    Strcat (str, “有”);
    strcat (str, “连接”);
    Std::cout << STR << return " Length = ' << strlen
}
```

```
# include <
iostream >

Int main() {
    std:: string
    str;
    STR = "这些";
    STR = STR + “字符串” std::cout << + “是” + “连接” ；
    STR << " length = ' << str.length ();
}
```


程序设计范型

编程范式是一种“风格”或“方式”编程。
•a 的特点之一

编程语言的特点是它对特定范式的支持。
有些语言在某些范式下编写起来很容易
但其他语言不是。

一种专门设计用于编程的语言
许多范式被称为多范式编程语言，例如 c++，Python。你可以写程序或者库，主要是过程的，面向对象的，或者
功能(即。，一些典型的范式)。

在一个大型程序中，甚至可以用不同的部分编写不同的范式。

c++支持过程式和面向对象的范式
自然地，支持函数式范式通过
<函数式>接口，并支持许多其他范式
通过各种外部库。

命令式编程范式

命令式编程范式:一个显式的序列
改变程序状态的语句, 指定如何改变
实现结果。

• 结构化:程序有干净、免 goto、嵌套
控制结构。

• 过程式:使用过程的命令式编程
对数据进行操作。

面向对象:对象具有 / 结合状态 / 数据和
行为 / 方法; 计算通过发送来完成
向对象(接收者)发送消息。

- 基于类:对象获得它们的状态和行为
基于在类中的成员关系。
- 基于原型:对象的行为来自于原型
原型对象。

作为抽象数据类型的字符串

由于 c++ 在它的前身语言中包含了所有内容，C 字符串是 c++ 的一部分，你偶尔会不得不这样做要认识到这种类型的字符串是存在的。

几乎对你编写的每一个程序来说，都要容易得多使用 c++ 的 string 类，它将字符串实现为**抽象数据类型，由其行为而定义**而不是它的表示。所有使用 string 类的程序必须包含 <string> 库接口。

c++ 为处理字符串提供的方法通常是与 Python 的字符串类型略有不同。大部分的这些差异属于偶然的类别。唯一的**这些模型的本质区别是 c++ 允许客户端**改变字符串中包含的单个字符。通过相比之下，Python 字符串是不可变的，这意味着它们一旦被分配，就永远不会改变。

在字符串中选择字符

<string>库提供了两种不同的选择机制
从字符串中选择单个字符:

```
str(指数)
```

```
str.at(索引)
```

唯一的区别是 at 检查以确保索引
在范围内，0~str.length()-1。

这两种方法都可以用来给
性格:

```
str[索引] = 'H';
```

```
str.at(index) = 'H';
```

- 前者可读性更好，后者有范围检查。

遍历字符串中的字符

处理字符串时，最重要的一个模式涉及遍历字符串中的字符，这需要以下代码：

```
For (int I = 0; I < str.length(); 我++) {  
    ... 操作 str[i] 的循环体 ...  
}
```

```
For (int I = str.length() - 1; I >= 0; 我--)
```

下面的函数反转了参数字符串，使得：
调用 reverse(“desserts”) 返回 “stressed”：

```
String reverse(字符串 str) {  
    字符串 rev =  
        "";  
    For (int I = str.length() - 1; I >= 0; 我()) {  
        }  
    返回 牧师;  
}
```

这算高效吗？

修改字符串的内容

在许多语言中，包括 Python、Java、c#，字符串是
不可变，这意味着它们一次都不会改变
它们是被分配的。

- 相比之下，c++ 允许客户端更改 a 的内容字符串，都是通过为选定的字符分配新值以及通过调用如下的字符串方法：

str.erase(pos, count) 破坏性地改变 str	← 从位置 pos 开始删除 str 中的 count 字符。
str.insert(pos, text) 破坏性地更改 str	← 从位置 pos 开始将 text 中的字符插入 str 中。
str.replace(pos, count, text) 破坏性地改变 str	←

修改字符串的内容

作为一种工具，用于编写更容易调试和

可维护的、不可变的字符串比可修改的字符串有很多优点
在 c++ 中的对应。幸运的是，在 c++ 中很容易保证这些优势
通过避免使用破坏性操作，如擦除、插入、替换和
对单个字符的赋值。

示例:在不改变原始字符的情况下进行大小写转换
(安全，但效率高吗?)

```
字符串大写字母(字符串){  
    字符串 result = "";  
    For (int I = 0; I < str.length(); 我 ++){  
        Result += toupper(str[i]);  
    }  
    返回结果;  
}
```

避免使用破坏性操作

示例:既安全又高效地转换实例(为什么?)

```
字符串大写字母(字符串 str) {for (int i = 0; i  
    < str.length ();i++) {str[i] = toupper(str[i]);  
  
    }返回 str;  
}
```

练习:大小写转换到位(效率最高, 但不是安全,为什么?)

```
void toUpperCaseInPlace(string & str) {  
    For (int I = 0;I < str.length();i++) {str[I] = toupper(str[I]);  
  
    }  
}
```

问题:我们能以同样的方式实现反向吗?

练习:识别回文

回文(palindrome)是一个单词，其阅读顺序从后到后向前读，比如“level”或“noon”。

- 写一个 c++ 程序 isPalindrome 来检查 a String 是否为 palindrome。

```
bool isPalindrome(string str) {int n =  
    str.length ();For (int I = 0;I < n / 2;i++) {if  
        (str[i] != str[n - i - 1])返回 false;  
  
        }返回 true;  
}
```

```
bool isPalindrome(string str) {return str ==  
    反向(str);  
}
```

效率 vs. 可读性

练习:编写字符串应用程序

教程

缩略语(acronym)是由每个单词的首字母组成的单词按顺序排列的单词, 如 in

自给式水下呼吸器;**scuba**;



- 写一个 c++ 程序, 生成首字母缩略词, 如图所示运行下面的示例:



更多例子:

• 将英语翻译成 Pig Latin

斯坦福 strlib.h 界面

integerToString (n)	将 n 转换为 c++ 字符串。
stringToInteger	将 str 中的数字转换为整数。
(str)	将 d 转换为 c++ 字符串。
realToString (d)	将 str 中的数字转换为浮点数。
stringToReal (str)	将 str 转换为大写。
toUpperCase (x)	将 str 转换为小写。
equalsIgnoreCase(s1, s2)	不考虑大小写来比较 s1 和 s2。
startsWith(str, prefix)	如果 str 以 prefix 开头，则返回 true。
endsWith(str, suffix)	如果 str 以 suffix 结尾，返回 true。
trim(str)	返回一个从结尾删除空格的字符串。

结束