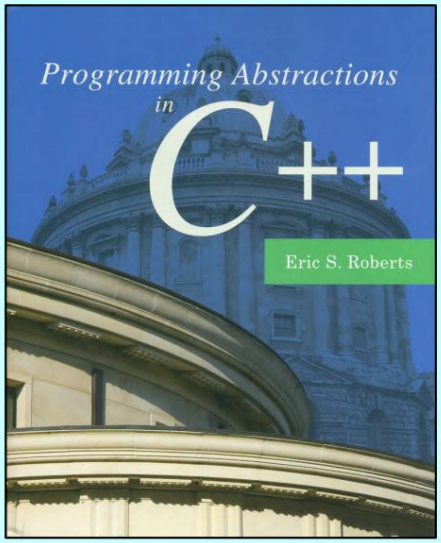
第3章

字符串

在这些琴弦上轻声细语。
TS艾略特,荒原， 1922



C++ 标准库简介

·类和函数的集合，它们是用核心语言编写的，也是C++ ISO 标准本身的一部分。
C++ 标准库的特性在std命名空间中声明

– 容器：vector、queue、stack、map、set等。
– 通用：算法、函数、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
– C 标准库：cmath、cctype、cstring、cstdio、cstdlib 等。

在 C 和 C++ 中使用字符串

·如今，文本数据与数字数据一样重要。几乎您用任何现代语言编写的任何程序都可能在某些时候使用字符串数据，即使它只是向用户显示指令或标记结果。

- 从概念上讲，字符串只是一个字符序列，这正是字符串在 C 中的实现方式。
- 作为一种新设计的语言，尤其是一种用面向对象编程范式扩展 C 的语言，C++ 支持更高级别的字符串视图，即对象。
- C 和C++ 对字符串使用的不同策略显示了不同编程范式之间的差异。

人物

- C 和C++ 都使用ASCII （美国信息交换标准代码)作为字符表示的编码。因此,数据类型char适合单个 8 位字节。

- 只有 256 个可能的字符， ASCII 不足以代表世界各地使用的许多字母表。在大多数现代语言中,ASCII 已被Unicode 取代,后者允许更多的字符。

- 尽管 ASCII 编码的弱点是在设计 C++ 时很清楚,考虑到将 C 保留为子集的决定,更改char的定义是不可能的。
- C++ 库定义类型wchar_t来表示允许更大范围的“宽字符”。wchar_t类型的详细信息超出了本文的范围。我们将坚持使用传统的char类型。

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

128	Ç	144	É	160	á	176	▒	192	└	208	╨	224	α	240	≡
129	ü	145	æ	161	í	177	▒	193	┴	209	╤	225	ß	241	±
130	é	146	Æ	162	ó	178	▓	194	┬	210	╥	226	Γ	242	≥
131	â	147	ô	163	ú	179	│	195	├	211	╙	227	π	243	≤
132	ä	148	ö	164	ñ	180	┤	196	─	212	╘	228	Σ	244	⌠
133	à	149	ò	165	Ñ	181	╡	197	┼	213	╒	229	σ	245	⌡
134	å	150	û	166	ª	182	╢	198	╞	214	╓	230	µ	246	÷
135	ç	151	ù	167	º	183	╖	199	╟	215	╫	231	τ	247	≈
136	ê	152	ÿ	168	¿	184	╕	200	╚	216	╪	232	Φ	248	°
137	ë	153	Ö	169	⌐	185	╣	201	╔	217	┘	233	Θ	249	·
138	è	154	Ü	170	¬	186	║	202	╩	218	┌	234	Ω	250	·
139	ï	155	¢	171	½	187	╗	203	╦	219	▉	235	δ	251	√
140	î	156	£	172	¼	188	╝	204	╠	220	▄	236	∞	252	n
141	ì	157	¥	173	¡	189	╜	205	═	221	▌	237	φ	253	²
142	Ä	158	₧	174	«	190	╛	206	╬	222	▐	238	ε	254	■
143	Å	159	ƒ	175	»	191	┐	207	╧	223	▀	239	∩	255	

Source: www.LookupTables.com

单个字符

- 要使用单字符类型的变量：

```
字符 ch;字符
ch =    a   ;字符 ch = 97;
```

- 练习:以下内容的等价陈述是什么：

```
字符 ch = 32;
```

```
字符 ch =                    ;
```

- 某些字符是特殊控制字符：

```
字符 ch = 9; //   \t   (Tab) char ch = 10; //   \n   (新
行) char ch = 13; //   \r   (返回)
```

- 要换行,Mac 使用\r， Unix 使用\n， Windows 使用\r\n。

< cctype> (ctype.h)接口

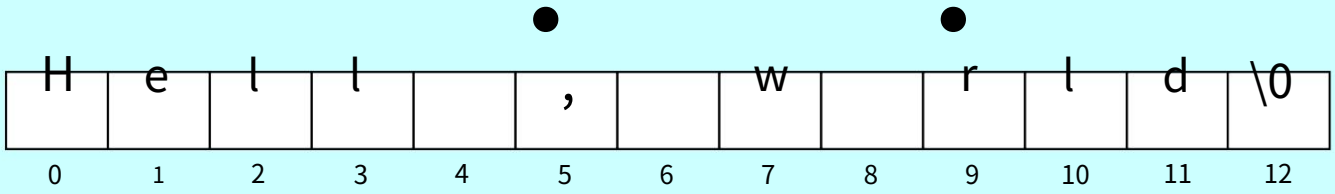
bool isdigit(char ch) 确定指定的字符是否为数字。
bool isalpha(char ch) 确定指定的字符是否为字母。
bool isalnum(char ch) 确定指定的字符是字母还是数字。
bool islower(char ch) 确定指定字符是否为小写字母。
bool isupper(char ch) 确定指定字符是否为大写字母。
bool isspace(char ch) 确定指定字符是否为空格（空格和制表符）。
char tolow(char ch) 将ch转换为其等效的小写字母（如果有）。如果不是,则原样返回ch 。
char toupper(char ch) 将ch转换为其等效的大写字母（如果有）。如果不是,则原样返回ch 。

C 风格字符串的遗产

- 在C 中,我们可以将字符串（即字符序列)表示为字符类型元素的普通数组：

```
字符 cstr[10];
字符 cstr[] =    你好    ;
char cstr[] = {    h    ,    e    ,    l    ,    l    ,    o    ,    \0    };
```

- 如果您将双引号括在一系列字符,您会得到所谓的C 字符串文字（const char[]类型）。字符存储在字节数组中,以ASCII 值为 0的空字节结尾。例如,C 字符串“hello, world”中的字符排列如下：



- C 字符串中的字符位置由一个索引标识,该索引从0开始,一直延伸到比字符串长度小一。

C 风格字符串的遗产

- 问题： “a”与 “a”相同吗?

  a 是一个包含 a 和空终止符的字符串文字,因此是一个 2 字符数组。
  \0 ,

调用 C 字符串函数

- <cstring> (string.h)接口提供了许多可用于操作C 字符串的函数。例如,如果要确定 C 字符串cstr 的长度,可以使用以下函数strlen:

```
字符 cstr[] =  你好 ;
int len = strlen(cstr);
```

- 如果要为 C 字符串cstr 赋值,可以使用以下函数strcpy:

```
strcpy（cstr,“世界”）;
```

cstr是否有足够的空间来容纳“世界”?

- 但是,除了初始化之外,您不能直接为 C 字符串赋值:

```
cstr =  世界 ;
cstr[] =  世界 ;
```

< cstring> (string.h)接口

Copying:

memcpy	Copy block of memory (function)
memmove	Move block of memory (function)
strcpy	Copy string (function)
strncpy	Copy characters from string (function)

Concatenation:

strcat	Concatenate strings (function)
strncat	Append characters from string (function)

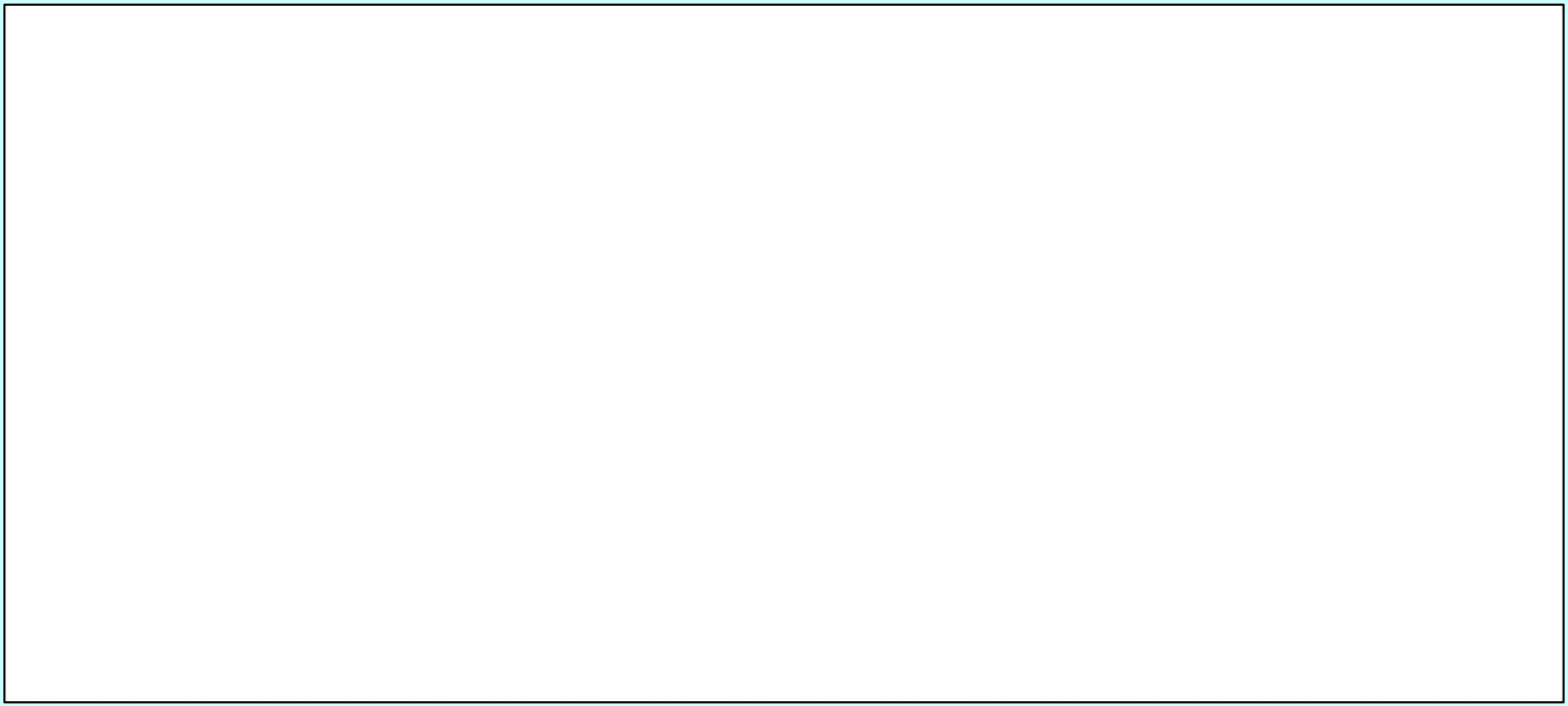
Comparison:

memcmp	Compare two blocks of memory (function)
strcmp	Compare two strings (function)
strcoll	Compare two strings using locale (function)
strncmp	Compare characters of two strings (function)
strxfrm	Transform string using locale (function)

Searching:

memchr	Locate character in block of memory (function)
strchr	Locate first occurrence of character in string (function)
strcspn	Get span until character in string (function)
strpbrk	Locate characters in string (function)
strrchr	Locate last occurrence of character in string (function)
strspn	Get span of character set in string (function)
strstr	Locate substring (function)
strtok	Split string into tokens (function)

< cstring> (string.h)示例



#include <iostream>
#include <cstring>
using namespace std;
int main()
{
char cstr[] = "海 0;长度 =
<< strlen(cstr);

< 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[]
        =    你好    ; cout << cstr << endl

                << cstr 大小 << endl
                << strlen(cstr) << endl;
        strcpy(cstr,    你好世界    ); cout << cstr << endl

                << cstr 大小 << endl
                << strlen(cstr) << endl;返回0;

}
```

sizeof x返回
变量x的实际内存大小。

使用字符串作为抽象值

- 自从文本中的第一个程序在屏幕上显示消息“hello, world”以来,您一直在使用字符串与用户交流。

- 到目前为止,您还不知道 C++ 是如何表示的计算机内部的字符串或如何操作组成字符串的字符。

- 同时,您不知道这些事情的事实并没有影响您有效使用字符串的能力,因为您已经能够从整体上考虑字符串,就好像它们是原始类型一样。

- 对于大多数应用程序,您到现在为止所掌握的字符串的抽象视图正是正确的。在内部,字符串是令人惊讶的复杂对象,其细节最好隐藏起来。

使用字符串作为抽象值

- 作为设计结果,C++ 必须保留它从其前身继承的旧的char类型和字符串模型，这是通过在C++ 标准库中包含C 标准库来实现的,例如<cctype> (ctype.h)；<cstring> (string.h)。

- 作为一种新设计的语言,尤其是一种用面向对象编程范式扩展C 的语言,C++ 支持更高级别的字符串视图,即对象。

- C++ 库<string>提供方便的抽象
  用于通过将string设为其方法隐藏底层复杂性的类来处理字符串。

·类是支持对象的数据类型的术语
  面向编程范式。应用于类实例的操作称为方法。

锻炼

以下陈述的含义是什么？ #include <cstring> // C 字符串
库,在 C++ 中使用#include <string.h> // C 字符串库,在 C 中
使用,在 C++ 中可接受#include string.h // 一些编译器
可能仍然会找到C 字符串库,除非你自己定义了string.h ,否则
会导致冲突#include <string> // C++ 字符串库#include
cstring.h // 不正确,除非你自己定义了cstring.h
#include < cstring.h> // 即使你自己定义了 cstring.h也很可能是一个错误

你好名字计划

FIGURE 3-1 An interactive version of the “Hello World” program

```
/*
 * File: HelloName.cpp
 * -------------------
 * This program extends the classic "Hello world" program by asking
 * the user for a name, which is then used as part of the greeting.
 * This version of the program reads a complete line into name and
 * not just the first word.
 */

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

int main() {
   string name;
   cout << "Enter your full name: ";
   cin >> name;
   cout << "Hello, " << name << "!" << endl;
   return 0;
}
```

cin >> 名称;

字符串类中的运算符

字符串[i] 返回str的第 i 个字符。分配给str[i]会更改该字符。
s1 + s2 返回一个由s1和s2连接的新字符串。
s1 = s2; 将字符串s2复制到s1 中。
s1 += s2; 将s2附加到s1 的末尾。 s1 ==
s2 （对于<、 <=、 >、 >=和!= 也是如此） 按字典顺序与字符串进行比较。
str.c_str() 返回包含与str 相同字符的 C 样式字符数组。

- 与大多数语言不同， C++ 允许类重新定义标准运算符的含义。因此，一些字符串操作，例如用于连接的+ ，被实现为运算符（重载）。

- 要将 C++ 字符串对象转换为 C 字符串文字，只需应用 C++ 字符串的c_str方法。

连接和 C 字符串

- 学过 Python 的人都知道， +
  运算符是字符串对象连接的便捷简写，它包括将两个字符串首尾相连，没有中间字符。

- 在 Python 中， +运算符通常用于将项目组合为print调用的一部分，如

```
打印（“你好，”                    +名称+“！”）；
```

- 在 C++ 中，您可以使用<<运算符获得相同的结果：

```
cout <<“你好，”                    <<名称<<“！”  <<endl;
```

- 尽管您可能会想到其他情况，但您不能总是在此语句中使用+运算符，这取决于name是 C 字符串（文字）还是 C++ 字符串对象。

```
cout <<“你好，”                    + 名称 +  “！”  <<endl;
```

C 字符串文字与 C++ 字符串对象

·只要编译器可以确定您想要的是 C++ 字符串对象， C++ 就会自动将C 字符串文字转换为C ++ 字符串对象：

```
字符串 str =    你好,世界   ;
```

· 相比之下,C++ 不允许您编写声明：

```
字符串 str =    你好    +    ,                                    +“世界”；
```

· +运算符不能应用于C 字符串文字。要得到围绕这个问题,您可以通过在文字上调用string将字符串文字显式转换为字符串对象：

```
字符串 str = string(    hello    ) +    ,                                    +“世界”；
```

```
字符串str =“你好”+字符串（“，”）+“世界”；
```

```
字符串 str =    你好    +    ,                                    + 字符串（“世界”）；
```

C 字符串文字与 C++ 字符串对象

练习:输出是什么?

```
#include <iostream>
#include <字符串>

int main() {
        char name1[] =    雷    ;
        std::string name2 =    雷    ;
        std::cout <<    你好，     std::cout <<       << name1 << std::endl;
          你好，     std::cout <<    你好，          << name2 << std::endl;
        std::cout <<    你好，     return 0;          + name1 << std::endl;
                                                      + name2 << std::endl;


}
```

字符串类中的常用方法

str.length() 返回字符串str 中的字符数。
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。
str.substr(pos, len) 返回从pos开始并持续到len个字符的str的子字符串。
str.find(ch, pos) 返回包含ch的第一个 ≥ pos的索引,如果未找到,则返回string::npos 。
str.find（文本,位置） 与前面的方法类似,但使用字符串而不是字符。

调用字符串方法

- 因为字符串是一个类,所以最好把它的方法考虑在
  向特定对象发送消息的术语。将消息发送到的对象称为接收者,发送消息
  的一般语法如下所示：

```
接收者名称（参数）；
```

- 例如,如果您想确定字符串的长度
  str,因此将len设置为字符串对象str的长度的语句的面向对象版本是

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

- 您可能还想使用我们提供的strlen函数
  之前与 C 字符串文字一起使用：

```
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str的长度?

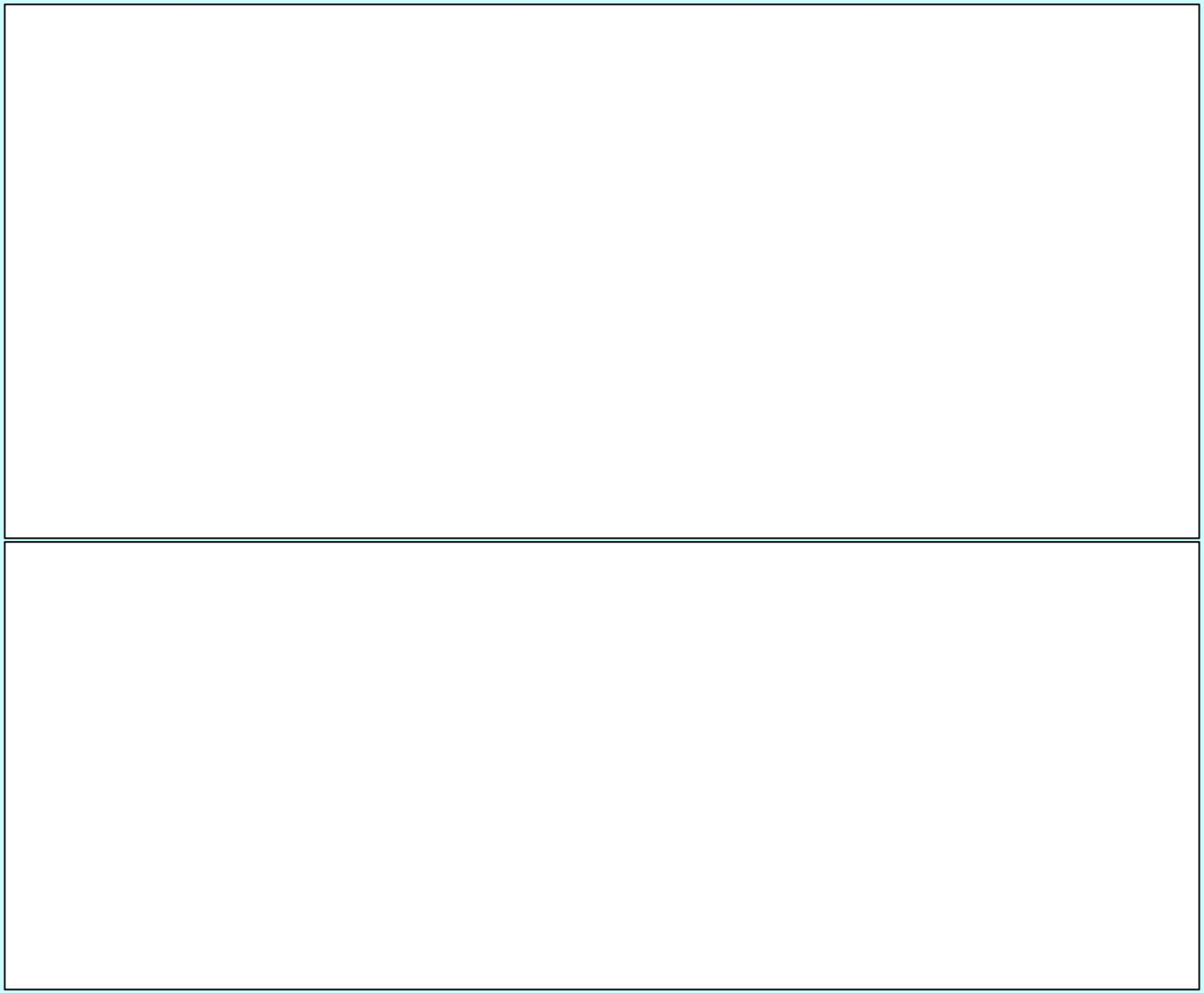
- 面向对象的版本:

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

- 如果您必须使用 C 中的strlen函数:

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

< string>库示例



编程范式

·编程范式是编程的“风格”或“方式”。 · 编程语言的特征之一是它支持特定的范例。有些语言可以很容易地用某些范式编写,但不能用其他范式编写。

· 一种旨在允许在多种范式中进行编程的语言称为多范式编程语言,例如C++、Python。您可以用这些语言编写程序或库,这些程序或库主要是过程性的、面向对象的或功能性的（即一些典型的范例）。

· 在大型程序中,甚至可以写入不同的部分
不同的范式。

· C++自然支持过程和面向对象的范式,通过<functional>接口支持函数范式,并通过各种外部库支持许多其他范式。

命令式编程范式

·命令式编程范例:改变程序状态的显式语句序列,指定如何实现结果。

–结构化:程序具有干净、无跳转、嵌套
控制结构。

–过程: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/组合状态/数据和行为/方法;计算是通过向对象(接收者)发送消息来实现的。

·基于类:对象根据类中的成员资格获得它们的状态和行为。

·基于原型:对象从原型对象中获取行为。

字符串作为抽象数据类型

- 由于 C++ 包含其前身语言中的所有内容,因此 C 字符串是 C++ 的一部分,您有时必须认识到这种字符串样式存在。

- 对于您编写的几乎每个程序,使用C++ 的字符串类会容易得多,该类将字符串实现为抽象数据类型,由其行为而不是其表示来定义。所有使用字符串类的程序都必须包含<string>库接口。

- C++ 提供的处理字符串的方法通常与 Python 的String类型中的方法略有不同。大多数这些差异属于偶然类别。这些模型的唯一本质区别是 C++ 允许客户端更改字符串中包含的单个字符。相比之下,Python 字符串是不可变的,这意味着一旦分配它们就永远不会改变。

选择字符串中的字符

- <string>库提供了两种不同的机制来从字符串中选择单个字符：

```
str[索引]
```

```
str.at(索引)
```

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

- 两种方法都可用于为
  特点：

```
str[索引] =    H   ;
```

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

- 前者具有更好的可读性,而后者具有范围
  检查。

遍历字符串中的字符

- 使用字符串时,最重要的一项
  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(); i++) {
        . . .操纵str[i]的循环体。 . .
}
```

```
for (int i = str.length() - 1; i >= 0; i--)
```

- 以下函数反转参数字符串,以便调用reverse(desserts)返回“stressed”:

```
字符串反向（字符串 str){
        字符串转=“”;
        for (int i = str.length() - 1; i >= 0; i--) {
                转 += str[i];
        }
        返回转速;
}
```

这有效率吗?

修改字符串的内容

- 在许多语言中,包括Python、Java、C#,字符串是不可变的,这意味着一旦分配它们就永远不会改变。

- 相比之下,C++ 允许客户端更改字符串的内容,既可以通过为选定字符分配新值,也可以调用字符串方法,如下所示:

str.erase（位置,计数） 从位置pos开始从str中删除count 个字符。	←破坏性地改变str
str.insert（位置,文本） 从位置pos开始将text中的字符插入str中。	←破坏性地改变str
str.replace（位置,计数,文本） 将str 中的count 个字符替换为从位置pos开始的文本。	←破坏性地改变str

修改字符串的内容

- 作为编写程序的工具,更易于调试和
与 C++ 中的可修改字符串相比,不可变字符串具有许多优点。幸运的是,通过避免使用诸如擦除、插入、替换和分配给单个字符的破坏性操作,很容易在 C++ 中确保这些优势。

- 示例:在不改变原件的情况下进行大小写转换（安全,但有效吗？）

```
字符串 toUpperCase（字符串 str){
        字符串结果 =        ;
        for (int i = 0; i < str.length(); i++) {
                结果 += toupper(str[i]);
        }
        返回结果;
}
```

避免使用破坏性操作

- 示例:安全有效的案例转换（为什么？）

```
string toUpperCase(string str) { for (int i = 0; i <
        str.length(); i++) { str[i] = toupper(str[i]);



        } 返回字符串;
}
```

- 练习:适当的案例转换（最有效但不安全,为什么？）

```
无效 toUpperCaseInPlace(字符串 & str) {
        for (int i = 0; i < str.length(); i++) { str[i] = toupper(str[i]);


        }
}
```

- 问题:我们可以以同样的方式实现反向吗?

练习:识别回文

·回文是一个向后读相同的单词,并且
向前,例如“水平”或“中午”。

· 编写一个 C++ 程序isPalindrome ,检查是否存在
字符串是回文。

```
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;



        } 返回真;
}
```

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

· 效率与可读性

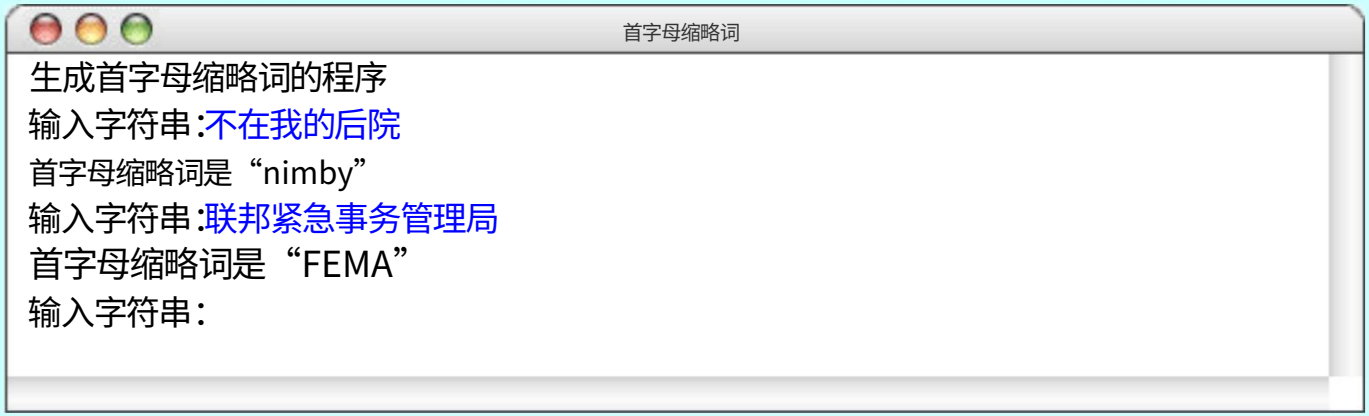
练习:编写字符串应用程序

TUTORIAL

·首字母缩略词是按顺序取每个单词的首字母组成的单词,如

“自给式水下呼吸器”“水肺” →

· 编写一个生成首字母缩略词的 C++ 程序,如以下示例运行所示:



· 更多示例:

– 将英语翻译成猪拉丁语

斯坦福strlib.h接口_

integerToString(n)	将n转换为 C++ 字符串。
stringToInteger(str)	将str中的数字转换为整数。
realToString(d)	将d转换为 C++ 字符串。
stringToReal(str)	将str中的数字转换为浮点数。
toUpperCase(x)	将str转换为大写。
toLowerCase(x)	将str转换为小写。
equalsIgnoreCase(s1, s2)比较s1和s2而不考虑大小写。	
startsWith(str, prefix)	如果str以前缀开头,则返回true 。
endsWith(str, suffix)	如果str以suffix结尾,则返回true 。
修剪（str）	返回从末尾删除空格的字符串。

The End