



UNIVERSITY OF  
RICHMOND



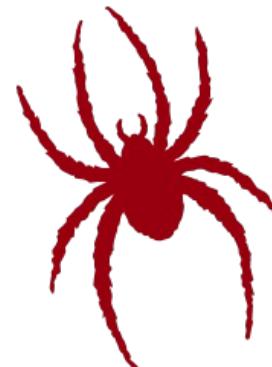
A photograph of a university campus. In the foreground, several students are walking away from the camera on a paved path. One student in the center-right has their back to the viewer, wearing a white shirt and dark pants. To their right, another student wears a blue backpack and jeans. On the far left, a student in a dark shirt and shorts walks towards the building. The background features a large brick building with arched windows and doors, surrounded by trees. A prominent tree on the left has pink blossoms, and another tree on the right is also in bloom. The overall scene is bright and sunny.

# Strings & Functions

CMSC 240 Software Systems Development

# Today

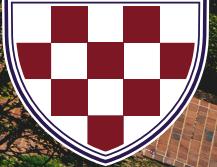
- Strings
- Command line arguments
- Include directive
- Function prototypes
- In-class coding exercise





# Strings

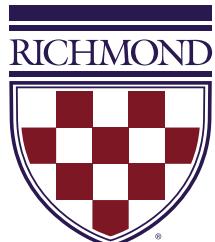
RICHMOND



# Strings

```
#include <string>  
  
string greeting = "hello";
```

- A **string** is a sequence of characters
- Strings in C++ are conceptually similar to strings in Java
  - Minor differences:
    - Different names for similar methods
    - Different behavior for similar methods
  - Major differences:
    - Strings are mutable (can be changed) in C++
    - There are two types of strings in C++



# Strings

- String characters are values of type **char**, with 0-based indexes

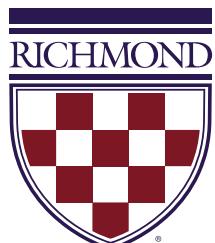
```
string greeting = "Hi there!";
```

Index	0	1	2	3	4	5	6	7	8
Character	'H'	'i'	' '	't'	'h'	'e'	'r'	'e'	'!'

- Individual characters can be accessed using [**index**] notation, or the string class method **at**:

```
char character1 = greeting[3];           // 't'  
char character2 = greeting.at(1);         // 'i'
```

- Characters have **ASCII** encodings ([integer mappings](#))

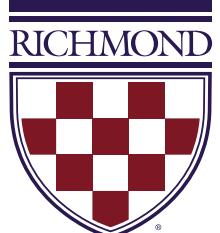


# Strings

## ASCII Table

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	\$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[END OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	-	127	7F	[DEL]

// 65



# C++ Strings

- Like Java, you can concatenate strings using + or +=

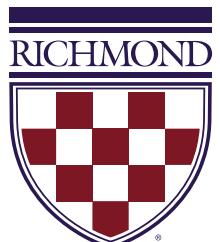
```
string mascot = "Web";
mascot += "stUR";                                // "WebstUR"
```

- Unlike Java, you can compare strings using relational operators

```
string mascot2 = "Ram";                          // ==, !=, <, <=, >, >=
if (mascot > mascot2 && mascot2 != "Eagle")    // true
{
    // Do something if true...
}
```

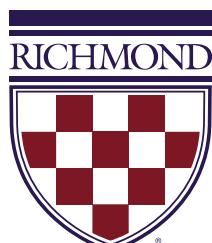
- Unlike Java, strings are mutable and can be changed (!!)

```
mascot.append(" the Spider");      // "WebstUR the Spider"
mascot.erase(3, 4);                // "Web the Spider"
mascot[12] = 'u';                  // "Web the Spidur"
```



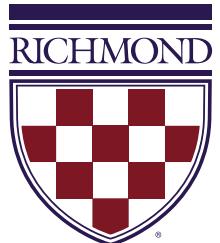
# C++ Strings

String Member Functions	Description
s.append( <b>str</b> )	Add <b>str</b> to the end of this string
s.compare( <b>str</b> )	Return -1, 0, or 1 depending on relative ordering
s.erase( <b>index, length</b> )	Delete <b>length</b> of text from string starting at <b>index</b>
s.find( <b>str</b> )	First index where the start of <b>str</b> appears in this string (returns <code>string::npos</code> if not found)
s.rfind( <b>str</b> )	Last index where the start of <b>str</b> appears in this string (returns <code>string::npos</code> if not found)
s.insert( <b>index, str</b> )	Add <b>str</b> into this string at a given <b>index</b>
s.length() or s.size()	Return the number of characters in this string
s.replace( <b>index, length, str</b> )	Replace <b>length</b> characters at given <b>index</b> with <b>str</b>
s.substr( <b>index, length</b> )	Return the next <b>length</b> characters beginning at <b>index</b> (inclusive)
s.substr( <b>index</b> )	Return the characters beginning at <b>index</b> (inclusive) until the end of the string



# C++ Strings

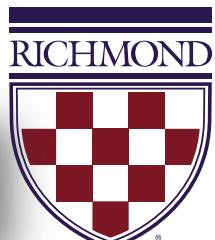
<https://en.cppreference.com>



# C vs. C++ Strings

- C++ has two kinds of strings
  - **C strings**, character arrays inherited from the C language
  - **C++ strings**, string objects in the `<string>` library
  - We will almost always use `string` objects
- Any string literal such as “Hi there!” is a C string
  - C strings don’t include any functionality
  - They don’t work with member functions like `.length()`
  - They don’t work with operators like `==`, or `>`
- Converting between string types

```
string greeting("Hi there!");           // converts C string into C++ string
const char* cString = greeting.c_str();  // returns a C string out of a C++ string
```



# C vs. C++ Strings

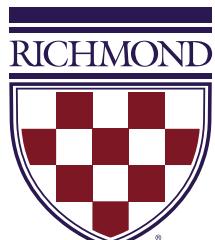
- C strings can not be concatenated with +

```
string greeting1 = "Hi" + " there!";           // does not compile
string greeting2 = string("Hi") + string(" there!"); // Okay
```

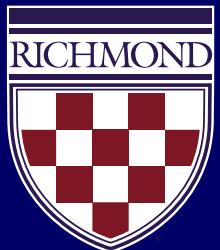
- C string is a contiguous sequence of characters terminated by and including the first null character, i.e. an array of characters terminated by '\0'

```
char greeting[] = "Hi there!";
```

Index	0	1	2	3	4	5	6	7	8	9
Character	'H'	'i'	' '	't'	'h'	'e'	'r'	'e'	'!'	'\0'



# Ask me questions





# Command-Line Arguments



# Command-Line Arguments

```
g++ hello.cpp -o hello
```

```
./hello Lilly
```

First Argument      Second Argument



# Command-Line Arguments

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

int main(int argc, char* argv[])
{
    → if (argc != 2) // argc counts the num of CLPs
    {
        → cerr << "Usage: " << argv[0]
            << " <first name>" << endl;
        exit(1);
    }

    → cout << "Hello " << argv[1] << endl;
    return 0;
}
```



# Command-Line Arguments

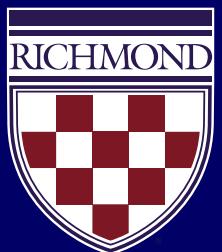
```
g++ hello.cpp -o hello
```

```
./hello Lilly
```

argv[0] argv[1]



# Ask me questions



An aerial photograph of a university campus during spring. In the center is a large, ornate brick tower with multiple spires and arched windows. The surrounding area is filled with green lawns, mature trees, and several smaller buildings. In the foreground, there are paved walkways where several people are walking. The sky is clear and blue.

# Include Directive



# #Include Directive

- `#include <libraryname>`
  - To use a built-in C++ system library
  - e.g. `#include <iostream>` to use cout and cin
- `#include "libraryname.h"`
  - To use a library in your local directory
  - e.g. `#include "mylibrary.h"` to use a library you created
- Please note the differences
  - `<>` vs. `" "`
  - no `.h` vs. `.h`



# #Include Directive

- Some common C++ libraries we will use

```
#include <iostream>           // standard stream objects cout, cin, cerr  
#include <fstream>           // file input/output  
#include <string>             // string class  
#include <array>               // array container  
#include <vector>              // vector container
```

- A list standard libraries:

<https://en.cppreference.com/w/cpp/header>



An aerial photograph of a university campus featuring a prominent red brick Gothic-style tower. The campus is surrounded by lush green trees and manicured lawns. Students are walking along paved paths. The sky is clear and blue.

# Function Prototypes



# Functions Without Prototypes

- You must define a function before calling it in your code

```
// Return the maximum of two numbers
int max(int left, int right)
{
    if (left > right)
        return left;
    else
        return right;
}

int main()
{
    int larger = max(31, 42);
    return 0;
}
```



# Functions Without Prototypes

- If you call a function before it is declared, you will get an error

```
int main()
{
    int larger = max(31, 42); // Error, undeclared function
    return 0;
}
```

```
// Return the maximum of two numbers
int max(int left, int right)
{
    if (left > right)
        return left;
    else
```

```
functions.cpp:4:22: error: use of undeclared identifier 'max'
```

```
    int larger = max(31, 42); // Error, undeclared
    ^
```

```
1 error generated.
```



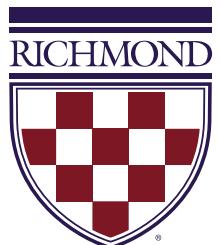
# Functions With Prototypes

- Unless you first declare the function with a function prototype

```
→ int max(int left, int right); // function declaration
```

```
int main()
{
    int larger = max(31, 42);
    return 0;
}

// Return the maximum of two numbers
int max(int left, int right)
{
    if (left > right)
        return left;
    else
        return right;
}
```



An aerial photograph of a university campus. In the center is a tall, ornate brick tower with multiple spires and arched windows. To its left is a large, light-colored building with a gabled roof. The campus is surrounded by a variety of trees, including several large evergreens and some with yellow or orange foliage. In the foreground, there are paved walkways and paths where several people are walking. The sky is clear and blue.

# In-Class Coding Exercise

