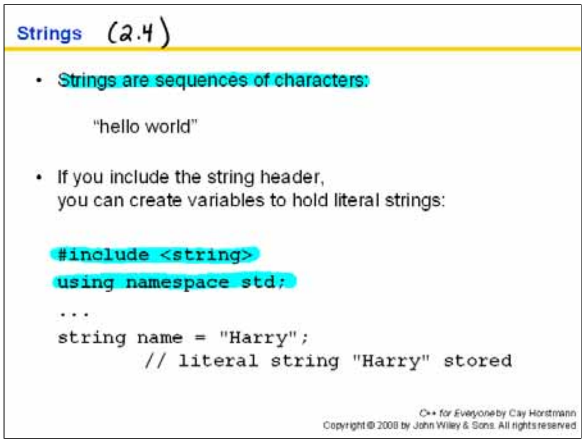


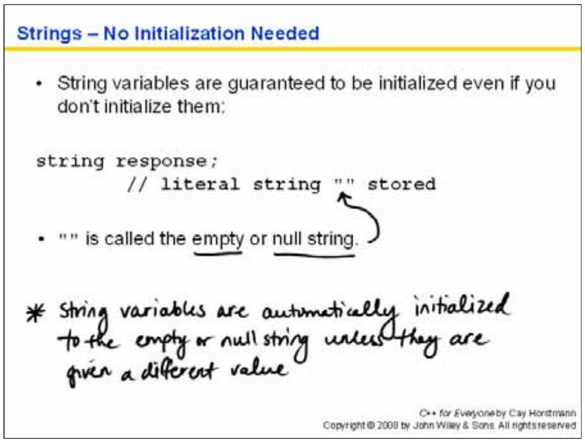
Strings (2.4)

- Strings are sequences of characters:

  "hello world"

- If you include the string header, you can create variables to hold literal strings:

```
#include <string>
using namespace std;
...
string name = "Harry";
        // literal string "Harry" stored
```



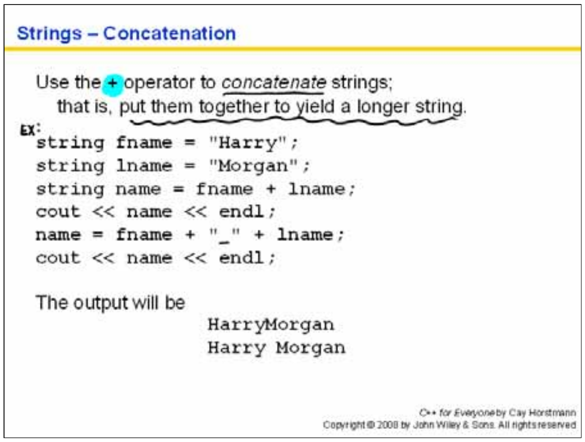
Strings – No Initialization Needed

- String variables are guaranteed to be initialized even if you don't initialize them:

```
string response;
        // literal string "" stored
```

- "" is called the empty or null string.

* String variables are automatically initialized to the empty or null string unless they are given a different value



Strings – Concatenation

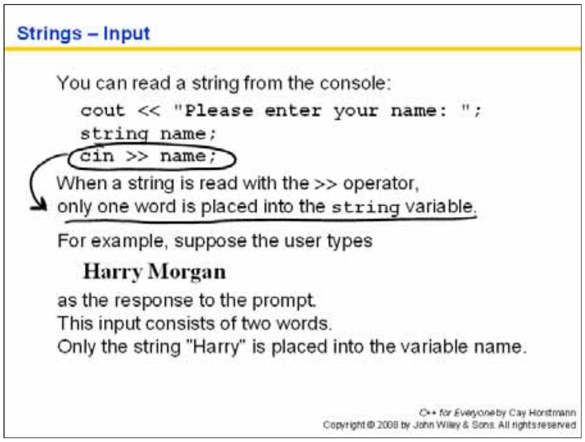
Use the + operator to *concatenate* strings;
that is, put them together to yield a longer string.

EX:

```
string fname = "Harry";
string lname = "Morgan";
string name = fname + lname;
cout << name << endl;
name = fname + " " + lname;
cout << name << endl;
```

The output will be

```
HarryMorgan
Harry Morgan
```



Strings – Input

You can read a string from the console:

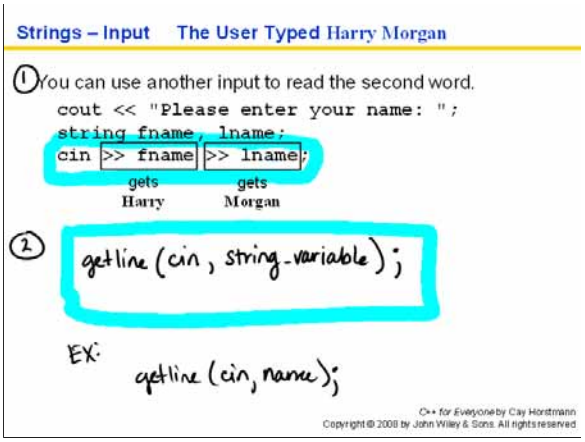
```
cout << "Please enter your name: ";
string name;
cin >> name;
```

When a string is read with the >> operator,
only one word is placed into the `string` variable.

For example, suppose the user types

Harry Morgan

as the response to the prompt.
This input consists of two words.
Only the string "Harry" is placed into the variable name.



Strings – Input The User Typed Harry Morgan

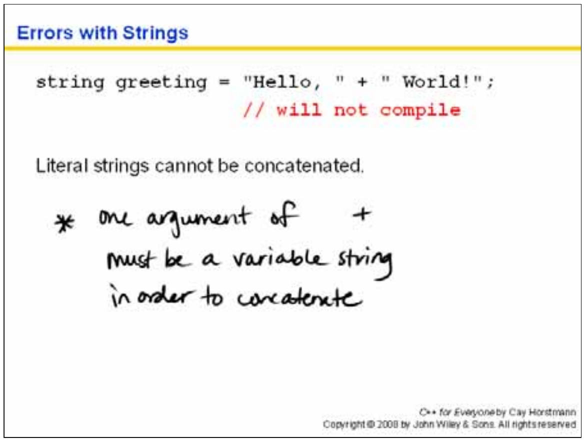
① You can use another input to read the second word.

```
cout << "Please enter your name: ";
string fname, lname;
cin >> fname >> lname;
```

gets Harry — gets Morgan

② getline(cin, string_variable);

EX: getline(cin, name);

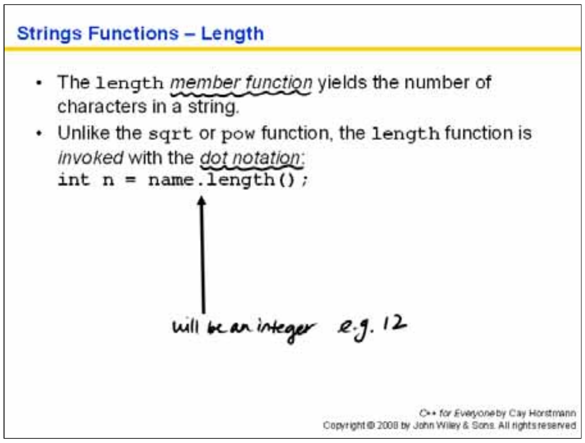


Errors with Strings

```
string greeting = "Hello, " + " World!";
                   // will not compile
```

Literal strings cannot be concatenated.

* one argument of + must be a variable string in order to concatenate

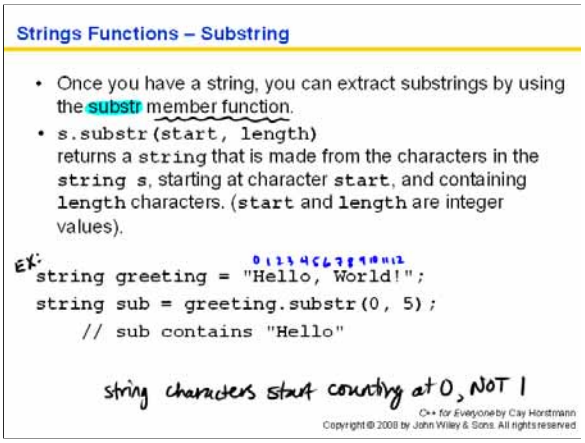


Strings Functions – Length

- The `length` *member function* yields the number of characters in a string.
- Unlike the `sqrt` or `pow` function, the `length` function is *invoked* with the *dot notation*:

```
int n = name.length();
```

will be an integer e.g. 12



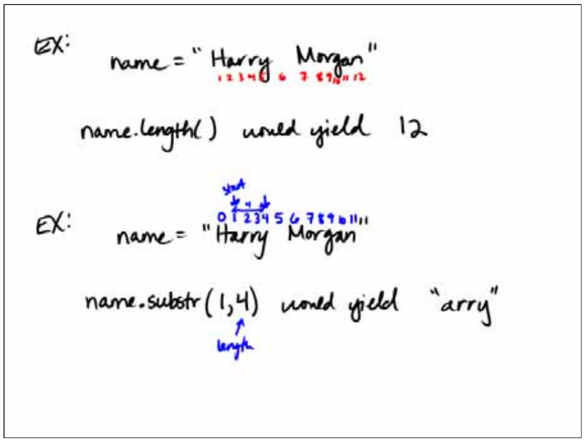
Strings Functions – Substring

- Once you have a string, you can extract substrings by using the substr member function.
- `s.substr(start, length)` returns a `string` that is made from the characters in the `string s`, starting at character `start`, and containing `length` characters. (`start` and `length` are integer values).

EX:

```
                    0 1 2 3 4 5 6 7 8 9 10 11 12
string greeting = "Hello, World!";
string sub = greeting.substr(0, 5);
    // sub contains "Hello"
```

string characters start counting at 0, NOT 1



EX: name = "Harry Morgan"

1 2 3 4 5 6 7 8 9 10 11 12

name.length() would yield 12

EX: name = "Harry Morgan"

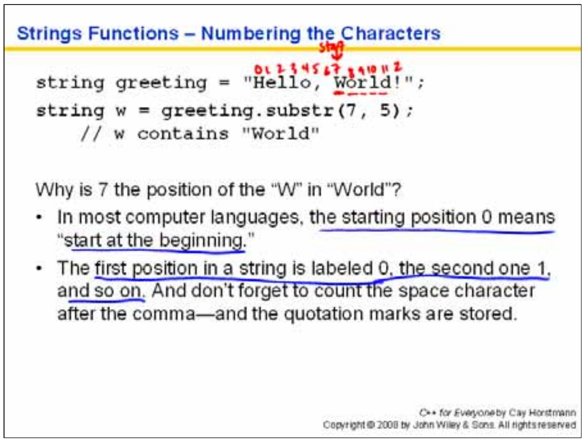
0 1 2 3 4 5 6 7 8 9 10 11

start

4

name.substr(1,4) would yield "arry"

length

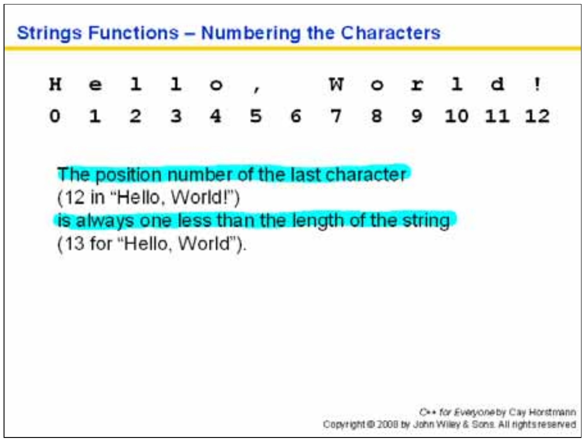


Strings Functions – Numbering the Characters

```
                    0 1 2 3 4 5 6 7 8 9 10 11 12
                             start
string greeting = "Hello, World!";
string w = greeting.substr(7, 5);
    // w contains "World"
```

Why is 7 the position of the "W" in "World"?

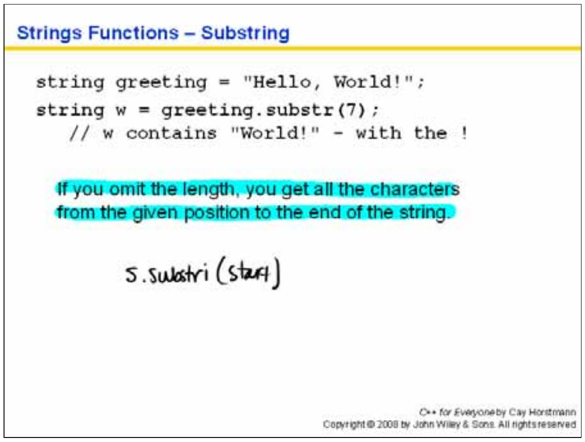
- In most computer languages, the starting position 0 means "start at the beginning."
- The first position in a string is labeled 0, the second one 1, and so on. And don't forget to count the space character after the comma—and the quotation marks are stored.



Strings Functions – Numbering the Characters

H	e	l	l	o	,		W	o	r	l	d	!
0	1	2	3	4	5	6	7	8	9	10	11	12

The position number of the last character (12 in "Hello, World!") is always one less than the length of the string (13 for "Hello, World").

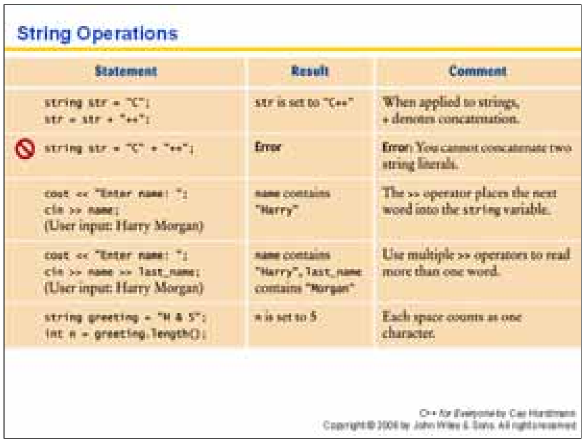


Strings Functions – Substring

```
string greeting = "Hello, World!";
string w = greeting.substr(7);
   // w contains "World!" - with the !
```

If you omit the length, you get all the characters from the given position to the end of the string.

s.substri(start)



String Operations

Statement	Result	Comment

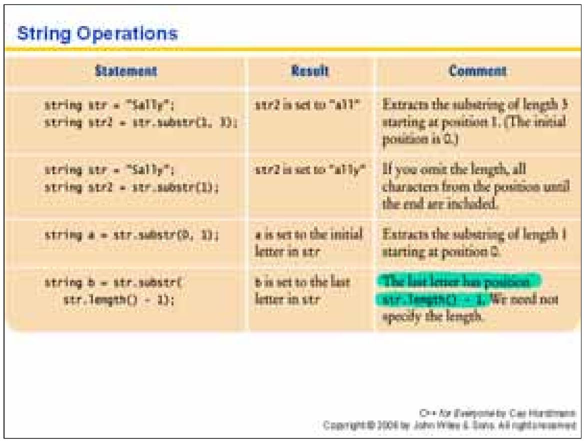
`string str = "C";`
`str = str + "++";`
	`str` is set to `"C++"`	When applied to strings, + denotes concatenation.

🚫 `string str = "C" + "++";`	**Error**	**Error**: You cannot concatenate two string literals.

`cout << "Enter name: ";`
`cin >> name;`
(User input: Harry Morgan)
	`name` contains `"Harry"`	The >> operator places the next word into the string variable.

`cout << "Enter name: ";`
`cin >> name >> last_name;`
(User input: Harry Morgan)
	`name` contains `"Harry"`, `last_name` contains `"Morgan"`	Use multiple >> operators to read more than one word.

`string greeting = "H & S";`
`int n = greeting.length();`
	`n` is set to 5	Each space counts as one character.



String Operations

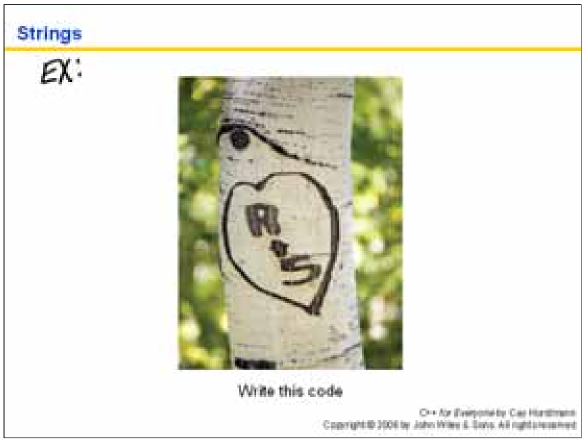
Statement	Result	Comment

`string str = "Sally";`
`string str2 = str.substr(1, 3);`
	`str2` is set to `"all"`	Extracts the substring of length 3 starting at position 1. (The initial position is 0.)

`string str = "Sally";`
`string str2 = str.substr(1);`
	`str2` is set to `"ally"`	If you omit the length, all characters from the position until the end are included.

`string a = str.substr(0, 1);`	`a` is set to the initial letter in `str`	Extracts the substring of length 1 starting at position 0.

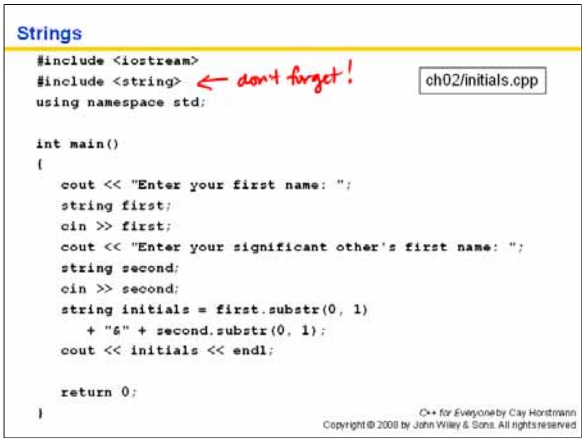
`string b = str.substr(`
`   str.length() - 1);`
	`b` is set to the last letter in `str`	The last letter has position `str.length() - 1`. We need not specify the length.



Strings

EX:

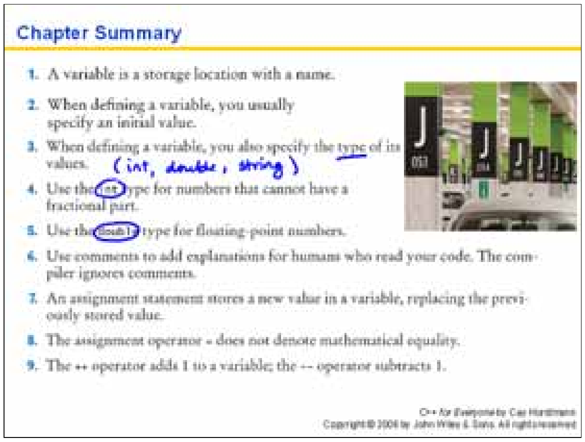
Write this code



Strings

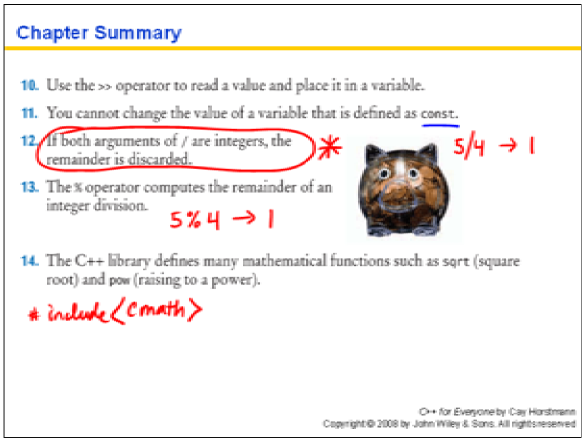
ch02/initials.cpp

```
#include <iostream>
#include <string>  ← don't forget!
using namespace std;

int main()
{
   cout << "Enter your first name: ";
   string first;
   cin >> first;
   cout << "Enter your significant other's first name: ";
   string second;
   cin >> second;
   string initials = first.substr(0, 1)
      + "&" + second.substr(0, 1);
   cout << initials << endl;

   return 0;
}
```



Chapter Summary

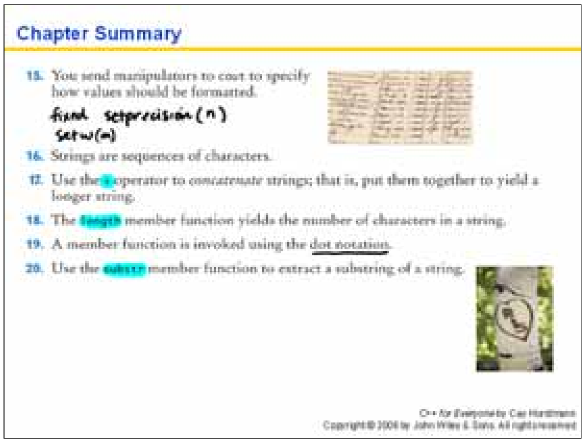
1. A variable is a storage location with a name.
2. When defining a variable, you usually specify an initial value.
3. When defining a variable, you also specify the type of its values. (int, double, string)
4. Use the int type for numbers that cannot have a fractional part.
5. Use the double type for floating-point numbers.
6. Use comments to add explanations for humans who read your code. The compiler ignores comments.
7. An assignment statement stores a new value in a variable, replacing the previously stored value.
8. The assignment operator = does not denote mathematical equality.
9. The ++ operator adds 1 to a variable; the -- operator subtracts 1.



Chapter Summary

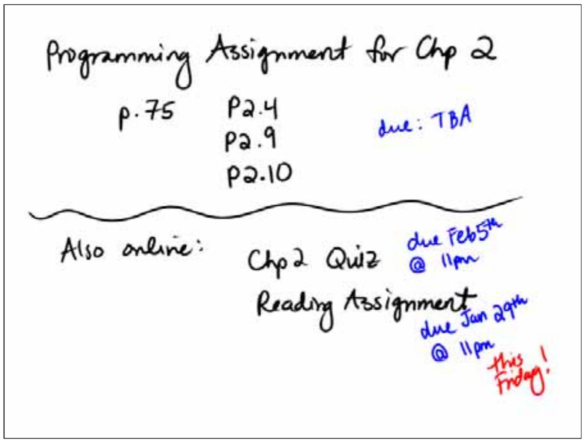
10. Use the >> operator to read a value and place it in a variable.
11. You cannot change the value of a variable that is defined as const.
12. If both arguments of / are integers, the remainder is discarded. ✱ 5/4 → 1
13. The % operator computes the remainder of an integer division. 5%4 → 1
14. The C++ library defines many mathematical functions such as sqrt (square root) and pow (raising to a power).

include <cmath>



Chapter Summary

15. You send manipulators to cout to specify how values should be formatted.
    fixed setprecision(n)
    setw(n)
16. Strings are sequences of characters.
17. Use the + operator to *concatenate* strings; that is, put them together to yield a longer string.
18. The length member function yields the number of characters in a string.
19. A member function is invoked using the dot notation.
20. Use the substr member function to extract a substring of a string.



Programming Assignment for Chp 2

p.75 P2.4
P2.9
P2.10

due: TBA

Also online: Chp 2 Quiz due Feb 5th @ 11pm

Reading Assignment due Jan 29th @ 11pm

this Friday!