Lecture 4: Strings, Classes, and the String Class

PIC 10A

Todd Wittman

What is a class?

- A class is a suped-up data type that may have special member functions defined in a library.
- An object is a suped-up variable that is a particular instance of a class.

```
int  v;          class_name  o;
```

data type · variable · class · object

•Typically, we access the member function of a class with a period after our object

object_name.function_name (parameters)

Classes in This Class

•We've already seen a class in action: cin and cout are objects defined in the iostream library.

•Today we're going to learn about the string class.

•On Wednesday we'll learn how to use a graphics class to draw some pictures.

•In November we'll learn how to write our own classes.

Sec 2.6: The String Class

- A string is a class that can hold text, much like the *primitive* data types int/double hold numbers.
- To use it, we have to include the string library

```
# include <string>
```

- The declaration is the same as for variables.

```
string name;
name = “My precious!\n”;
```

- Note the string text can hold punctuation, spaces, and even escape characters.

String I/O

- We can input & output strings just like variables.

```
string name = "My\nprecious!\n";
cout << name;
```

- Output:

```
My
precious!
```

- Just remember to include both the string and iostream libraries.

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

String I/O

- To read multiple strings, they should be separated by a space or line break.

```
int x, y;              string name1, name2;
cin >> x >> y;         cin >> name1 >> name2;
cout << x << y;        cout << name1 << name2;
```

- But this misses the spaces, just as with numbers.

```
User inputs: 2 3       User inputs: Sam Gamgee
Output: 23             Output:  SamGamgee
```

How do we read a long string?

- If you want to read a long statement into a string use the member function getline of cin. It will read all characters, including spaces, into a string until ENTER is pressed.

```
string name;
cout << “Enter name: “;
getline( cin, name );
cout << “Your name is “ << name << “.\n”;
```

```
Enter name: Sam Q. Gamgee
Your name is Sam Q. Gamgee.
```

getline doesn't always get the line

- We have to be careful when a cin is immediately followed by a getline.
- The following code does not work properly.

```
string name1, name2;
cin >> name1;
getline(cin, name2);
```

- After cin, the cursor is just before the \n. So getline reads in an empty string into name2. The user will not have a chance to type in name2.
- One way to fix this is to add a dummy getline to move the cursor.

```
cin >> name1;
getline(cin, name2);   //Gets the empty string "".
getline(cin, name2);   //Gets the next line of text.
```

- Another solution is to use the ignore function: cin.ignore(1,'\n');

String Length

- The member function length gives us the number of characters in the string.

  string name = "Gamgee, Sam\n";

  int nameLength = name.length( );

- Sets nameLength = 12.
- The characters of the string are indexed from 0 to 11. (C++ starts counting at 0, for some reason.)
- Escape characters only count for one character.

G	a	m	g	e	e	,		S	a	m	\n
0	1	2	3	4	5	6	7	8	9	10	11

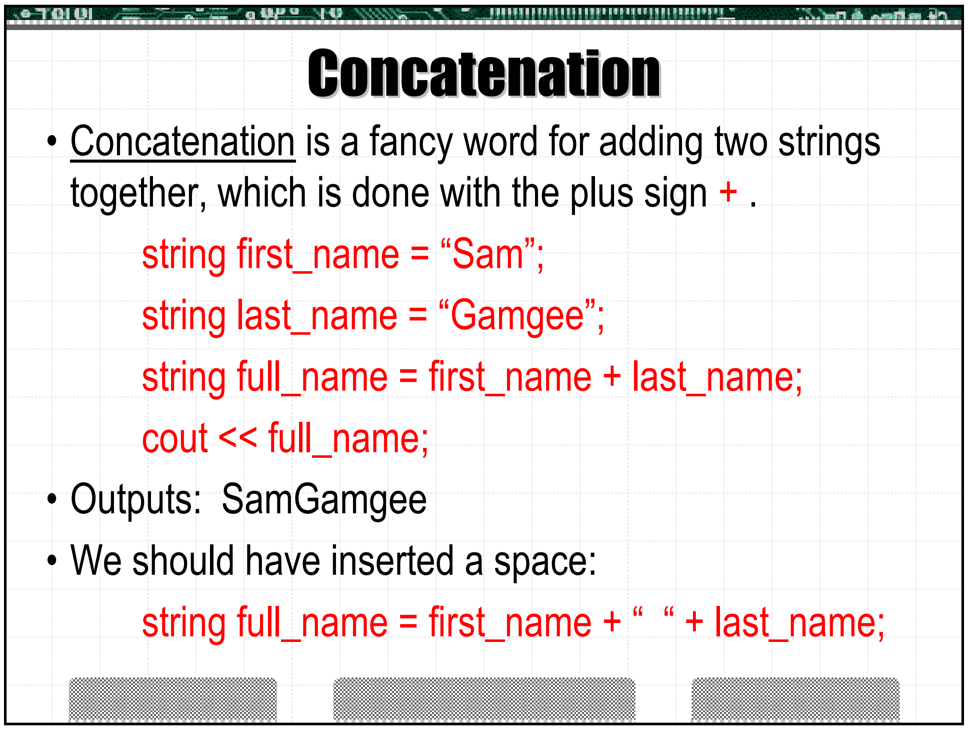
Substrings

- We can extract part of a string with the substring function.
- The syntax is: substr (start , length)

G	a	m	g	e	e	,		S	a	m	\n
0	1	2	3	4	5	6	7	8	9	10	11

```
string name = "Gamgee, Sam\n";
string last_name = name.substr (0,6);      // "Gamgee"
string first_name = name.substr (8,3);     // "Sam"
```

- *Common mistake*: People forget the second number is the substring length, not the ending position.



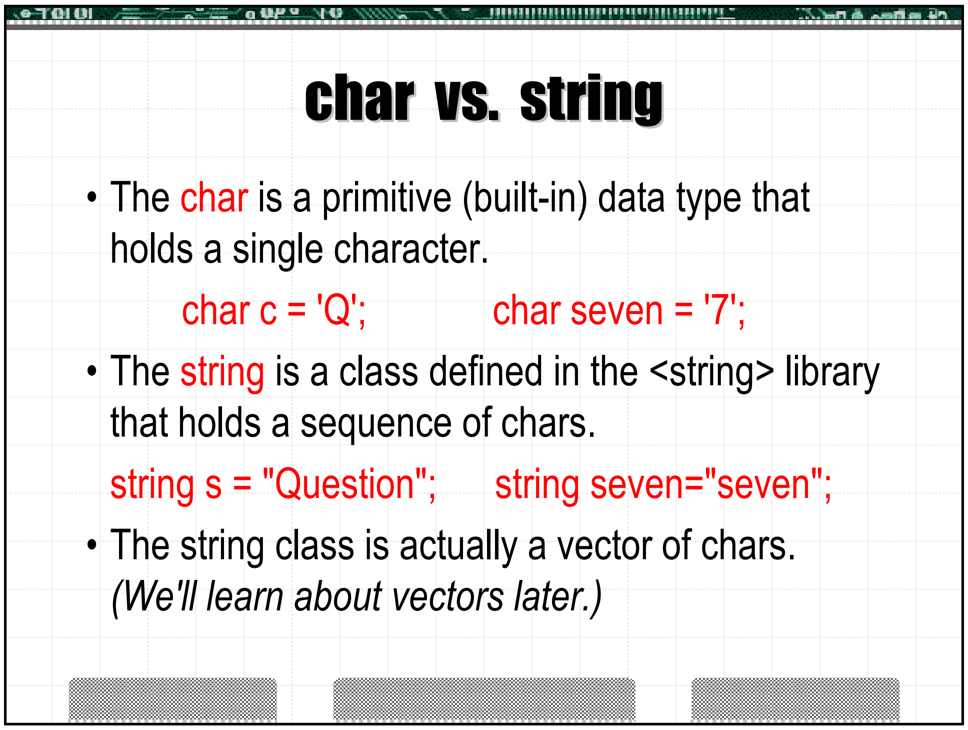
Concatenation

- Concatenation is a fancy word for adding two strings together, which is done with the plus sign + .

```
string first_name = “Sam”;
string last_name = “Gamgee”;
string full_name = first_name + last_name;
cout << full_name;
```

- Outputs: SamGamgee
- We should have inserted a space:

```
string full_name = first_name + “  “ + last_name;
```



char vs. string

- The char is a primitive (built-in) data type that holds a single character.

```
char c = 'Q';            char seven = '7';
```

- The string is a class defined in the <string> library that holds a sequence of chars.

```
string s = "Question";      string seven="seven";
```

- The string class is actually a vector of chars. *(We'll learn about vectors later.)*

Setting Output Precision

- We can control how many decimal places we output using:
  setprecision (num_decimal_places)
- Need a I/O library: # include <iomanip>
- Place the function in cout's push.
- Sets the precision for all following cout's.

```
double one_third = (double) 1 / 3;
cout << setprecision (2);
cout << one_third;
```

- Outputs: 0.33
- We can chain the push:

```
cout << setprecision(2) << one_third;
```

Setting Output Precision

- Doesn't output a trailing zero though.

```
cout << setprecision(2) << 1.20;
```

Outputs: 1.2

- Sometimes we want that zero, like for dollars.
- To force the zero, chain the term fixed.

```
cout << fixed << setprecision(2) << "$" << 1.2;
```

Outputs: $1.20

Formatting Output

- We can format our output in columns using cout's function

  setw(column_width)
- Need a I/O library: # include <iomanip>
- Writes a string of specified width to the screen, with spaces filled in the blanks. Make sure length < width.
- Substrings are aligned on right side.
- Unlike setprecision, we need to put setw into every cout push we want it used.

```
cout << setw(7) << 2 << "cool";                  2cool
cout << setw(7) << 2 << setw(7) << "cool";       2   cool
```

- Note the column width could be an integer variable.
- You can make it left-justified by pushing: setiosflags(ios::left)

Formatting Example

```
cout << fixed << setprecision(2);
cout << setw(25) << "Cost of LOTR ticket:";
cout << setw(15) << 8.50 << "\n";
cout << setw(25) << "Cost of LOTR DVD:";
cout << setw(15) << 24.90 << "\n";
cout << setw(25) << "Plastic hobbit feet:";
cout << setw(15) << "Priceless\n";
```

```
     Cost of LOTR ticket:           8.50
        Cost of LOTR DVD:          24.90
     Plastic hobbit feet:      Priceless
```

Actually $17.99 + S/H

Capitalizing Strings

- Need library: # include <cctype> *(See p. 970)*
- Can change case of a single letter (char) with toupper(char) & tolower(char)
- But these functions return integers corresponding to the letter, so cast to char.
- Example: Capitialize the first letter of an input first name.
- For example, should look like:

```
Enter your first name: sam
Your name is Sam.
```

- Try writing the code. I'll wait...

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

int main( )  {
        char first_initial;
        string name;    // Name except for first letter.
        cout << “Enter your first name: ”;
        cin >> first_initial >> name;
        first_initial = (char) toupper (first_initial);
        cout << “Your name is “ << first_initial << name << “./n”;
        return 0;
}
```