

defines an array of characters, called a, and sets its elements to have the letters of the word Simon.

• We can access the elements of the array using the index notation:

a[2]

for example, refers to the element of the array a with number 2.

cis1.5-fall2009-parsons-lectIV.2

- Today • Recap a little on arrays. • Describe some of the uses of strings • Show how strings and arrays are related. • You can find the examples from this lecture in the file
 - This is a character (since a is an array of characters).
 - We can do to a [2] exactly the same things we can do to any
 - For example we can print it out:

cout << a[2];

which will print out:

m

• We can also assign it a new value: a[3] = 't';

cis1.5-fall2009-parsons-lectIV.2

Strings

- To deal with strings, we need to add: #include<string> at the start of our program.
- With that in place, we can define variables whose type is string:

```
string s1 = "Hello";
string s2 = "Simon";
string s3, s4;
```

- This defines s1 to be a string variable whose value is the word Hello, and s2 to be a string variable who value is the word Simon.
- It also defines s3 and s4 to be strings, but does not give them a value.

cis1.5-fall2009-parsons-lectIV.2

- You can also test the value of two strings
- The expression

```
s1 == s2
```

will return true if the letters in the same location in both strings are the same.

- This won't be true since the first letters, H and S are different.
- However, given the value we assigned to s3:

s1 == s3

will return true.

- Since \$1, \$2, and \$3 are variables, we can do a lot of the kinds of things we can do to other variables to them.
- We can assign values to them and print their values out.
- For example:

```
s3 = s2;
cout << s3;
```

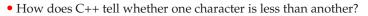
```
will generate:
Simon
```

cis1.5-fall2009-parsons-lectIV.2

- Another expression we can evaluate is:
- s1 < s2
- (We might want to use this in an if).
- C++ evaluates s1 < s2 by taking the first character in s1 and seeing if it is less than the first character in s2.
- If yes, then it returns true.
- If no, then it returns false
- If the characters are the same, the same question is asked of the second character in both strings.
- If every character in s1 is the same as in s2 then < will eventually return false.

cis1.5-fall2009-parsons-lectIV.2

cis1.5-fall2009-parsons-lectIV.2



- It uses the ASCII value (which we talked about earlier in the semester).
- All you probably need to know about these values is that:
 - $0 < 1 < 2 < 3 < \ldots < 9$
 - 9 < A < B < C < \ldots < Z

- So Hello is less than Simon, because H is not less than S.
- But Hello *is not* less than Hella
- The other comparison operations (>, <= and >=) behave similarly.

cis1.5-fall2009-parsons-lectIV.2

• When we print, we get:

HelloSimon

• There is no space because neither s1 or s2 has a space.

```
s3 = s1 + " " + s2;
```

cout << s3

would produce:

Hello Simon

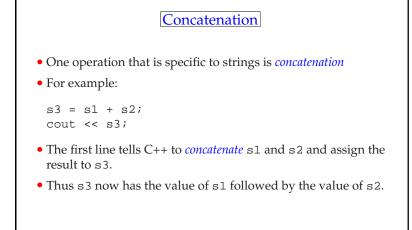
Note that

```
s3 += s2;
```

is just the same as:

s3 = s3 + s2;

cis1.5-fall2009-parsons-lectIV.2



```
cis1.5-fall2009-parsons-lectIV.2
```

Reading in strings

• One way to read in a string from the user is

cin >> s3;

• This is fine if you want to read in strings like:

Hello

and

Roustabout

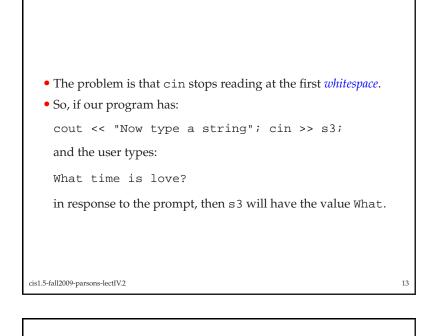
but no good if you want to read in:

What time is love?

• In fact, depending on your implementation of C++, cin may not handle strings even this well.

cis1.5-fall2009-parsons-lectIV.2

11



- The way around this problem is to use the function getline.
- There are two ways to use getline.
- Like this:

cout << "Now type a string"; getline(cin,s3);

it will read everything up to the point the user hits the return key, and assign this to s 3.

• This is fine for reading in What time is love?

cis1.5-fall2009-parsons-lectIV.2

- We can also call getline with a third parameter.
- This parameter is a character, called a *delimiter*, which tells getline when to stop reading.
- If our program has:

```
cout << "Now type a string";
getline(cin,s3,',');
getline(cin,s4,'.');
```

and the user types:

I stumbled out of bed, I got ready for the struggle.

then ...

```
cis1.5-fall2009-parsons-lectIV.2
```

- s3 will have the value
- I stumbled out of bed
- and s4 will have the value
- I got ready for the struggle
- Note that the delimiters are not read in, and so don't end up in either string.

cis1.5-fall2009-parsons-lectIV.2

15

• For example

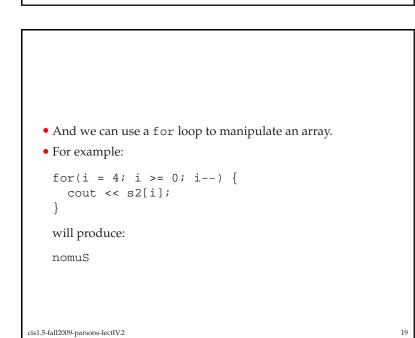
```
ifstream myInputFile;
myInputFile.open("sequence.txt");
getline(myInputFile,s3);
```

will read the first line of the file sequence.txt into the string variable s3, while

```
ifstream myInputFile;
myInputFile.open("sequence.txt");
getline(myInputFile,s4,'t');
```

will read the first line of the file sequence.txt up to the first t into the string variable s4.

cis1.5-fall2009-parsons-lectIV.2



From strings back to arrays

- As we hinted at the end of last lecture, strings are just arrays of characters.
- A string variable like s1 is just another way of dealing with an array of characters like a that we started the lecture with.
- As a result we can do things like:

s2[1] = 'u';
cout << s2;</pre>

to produce:

Sumon

cis1.5-fall2009-parsons-lectIV.2

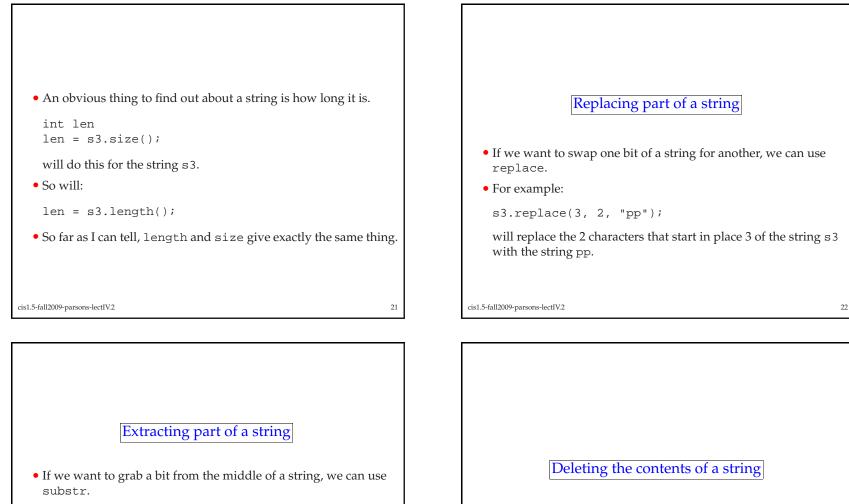
17

Member functions

- There are lots of functions in the string library.
- These are member functions of the string class.
- The idea of member function will make more sense later in the course when we have covered classes.
- But for now, you just have to know that in C++, a string is a *class*, and classes come along with *member functions* or *methods* that operate on them.
- In fact we already met some of these member functions:
 - cout.precision
 - -infile.open

cis1.5-fall2009-parsons-lectIV.2

20



23

- This extracts a *substring* from the string we apply it to.
- For example:

```
s4 = s3.substr(6, 2);
```

will copy the 2 characters that start in place 6 of the string s3 into the string s4.

```
• s3.erase() will set s3 to contain no characters.
```

24

```
• This is the same as doing:
```

s3 = "";

cis1.5-fall2009-parsons-lectIV.2

