

## 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-spring2007-parsons-lectIV.3

- Since s1, s2, and s3 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-spring2007-parsons-lectIV.3

5

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

```
s1 == s2
```

will return  $\mathtt{true}$  if the letters in the same location in both strings are the same.

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

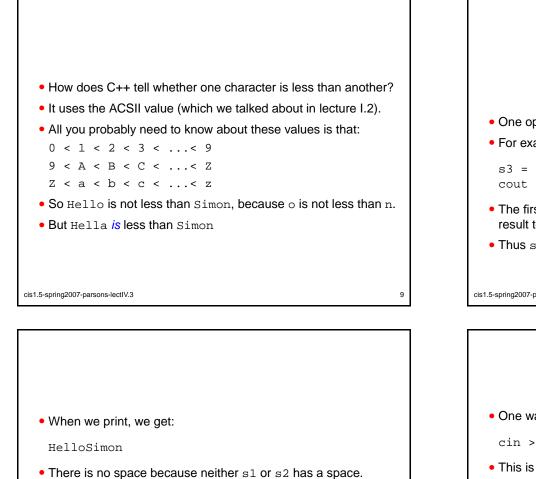
s1 == s2

will return true.

cis1.5-spring2007-parsons-lectIV.3

- 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 no, then it treturns false.
- If yes, it asks the same question of the second character in both strings.
- If every character in s1 is less than the corresponding character in s2, then s1 < s2 will be true. Otherwise it will be false.

cis1.5-spring2007-parsons-lectIV.3



```
s3 = s1 + '' '' + s2;
cout << s3
```

would produce:

Hello Simon

Concatenation • One operation that is specific to strings is concatenation • For example: s3 = s1 + s2;cout << s3; • The first line tells C++ to concatenate s1 and s2 and assign the result to s3. • Thus  $s_3$  now has the value of  $s_1$  followed by the value of  $s_2$ .

cis1.5-spring2007-parsons-lectIV.3

## Reading in strings

10

12

• 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

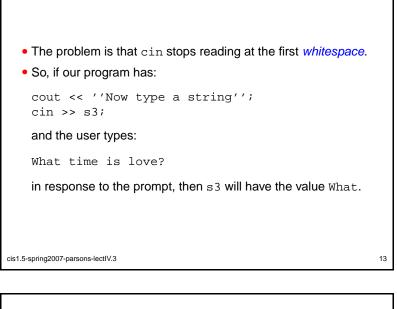
but no good if you want to read in:

What time is love?

cis1.5-spring2007-parsons-lectIV.3

11

cis1.5-spring2007-parsons-lectIV.3



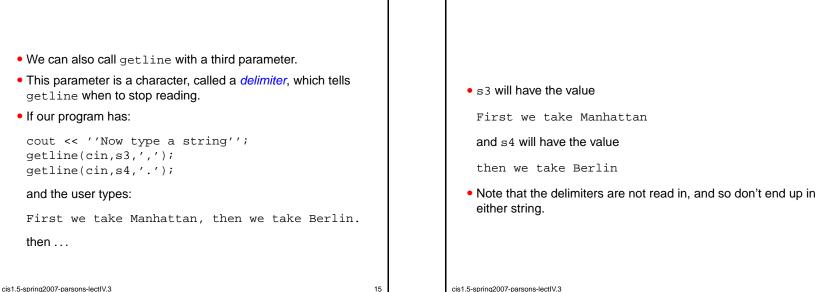
- The way around this problem is to use the function getline.
- There are two ways to use getline.
- Like this:

```
cout << ''Now type a string'';
getline(cin,s3);</pre>
```

it will read everything up to the point the user hits the return key, and assign this to s3.

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

```
cis1.5-spring2007-parsons-lectIV.3
```



16

14

```
• We can also use getline to read strings from a file.
                                                                                                 From strings back to arrays
  • For example
    ifstream myInputFile;
                                                                                   • As we hinted at the end of last lecture, strings are just arrays of
    ifstream.open(''sequence.txt'');
                                                                                    characters.
    getline(myInputFile,s3);
                                                                                  • A string variable like s1 is just another way of dealing with an
                                                                                    array of characters like a that we started the lecture with.
    will read the first line of the file sequence.txt into the string
    variable s3, while
                                                                                   • As a result we can do things like:
    ifstream myInputFile;
                                                                                    s2[1] = 'u';
    ifstream.open(''sequence.txt'');
                                                                                    cout << s2;
    getline(myInputFile,s4,'t');
                                                                                    to produce:
    will read the first line of the file sequence.txt up to the first t
                                                                                    Sumon
    into the string variable s4.
                                                                                cis1.5-spring2007-parsons-lectIV.3
cis1.5-spring2007-parsons-lectIV.3
                                                                  17
```

19

```
Summary
```

- This lecture started to look at strings.
- We briefly recapped arrays.
- We described how to define strings, and what operations you can carry out on them.
- We described at length how to read them in.
- Finally, we dealt briefly with the relationship between strings and arrays.

cis1.5-spring2007-parsons-lectIV.3

• And we can use a for loop to manipulate an array.

• For example:

will produce:

cis1.5-spring2007-parsons-lectIV.3

nomuS

}

 $for(i = 4; i \ge 0; i--)$ 

cout << s2[i];

18