Lecture 16

Chapter 9.3
Topics

• Class string
• What it does
• How to declare it
• Overloading
• cin
• Getline
• length
• Assignment/ModifiersComparisons
• Converting between string object and C-strings
Class string

• To use the class string you must
  – #include <string>
  – using namespace std;
What it does

• It allows you to treat string values and expressions like simple types
  – Like using = to assign a value to a string
  – Like using + to concatenate strings together
    • s3 = s1 + s2;
  – It will automatically resize s3 as needed.
How to declare it

• Quoted strings will always be C-string but are automatically type cast during assignment.
  – string phrase = “The cake is a lie!”;
  – string phrase;
  – string phrase(“The cake is a lie!”);
Overloading

• string object = “cake”;
• string phrase;
• phrase = “The “ + object + “ is a lie!”;
cin

• cin works the same as it did before
  – string object;
  – cin >> object;
getline

• getline also works as before but the call is different
  – string line;
  – cout << “Enter a type of cake.\n”
  – getline(cin, line);
  – cout << “Well your “ << line << “ is a lie!”;

• What does this do?
  – getline(cin, line, ‘?’);
• Length is also very similar to before

```cpp
string line;
cout << "Enter a type of cake.\n";
getline(cin, line);
if(line.length < 9000)
    cout << line << " is a type of cake that is less than 9000. I’m not impressed\n";
else
    cout << line << " sounds awesome!\n";
```
Element access

• `str[i]`
  – Returns reference to character in `str` at index `i`.

• `str.at[i]`
  – Returns reference to character in `str` at index `i`.

• `str.substr(position, length)`
  – Returns a substring starting at index `position` and ending at `length` positions away.
Assignment/Modifiers

• **str1 = str2;**
  – Allocates and initializes str1 to str2’s data. It will resize if needed.

• **str1 += str2;**
  – Character data from str2 is concatenated to the end of str1. It will resize if needed.

• **str1.empty()**
  – Returns true if str1 is empty.

• **str1 + str2**
  – Returns a string where str2 is concatenated to the end of str1.
Assignment/Modifiers

• `str1.insert(pos, str2);`
  – Inserts `str2` into `str1` beginning at index `pos`

• `str1.remove(pos, length);`
  – Removes chars from index `pos` until size `length`
Comparisons

• \texttt{str1 == str2}, \texttt{str1 \neq str2}
• \texttt{str1 > str2}, \texttt{str1 < str2}
• \texttt{str1 \geq str2}, \texttt{str1 \leq str2}
• \texttt{str1.find(str2)}
  – Returns the index of the first occurrence of \texttt{str2} in \texttt{str1}.
• \texttt{str1.find(str2, pos)}
  – Same as above but starts search at index \texttt{pos}
Comparisons

• `str1 == str2`, `str1 != str2`
• `str1 > str2`, `str1 < str2`
• `str1 >= str2`, `str1 <= str2`
• `str1.find(str2)`
  – Returns the index of the first occurrence of `str2` in `str1`.
• `str1.find(str2, pos)`
  – Same as above but starts search at index `pos`
Converting between string object and C-strings

• This will work
  - char aCstring[] = “This is my C-string”;
  - string stringVariable;
  - stringVariable = aCstring

• But this won’t
  - aCstring = stringVariable;
  - strcpy(aCstring, stringVariable);

• You must do it like this
  - strcpy(aCstring, stringVariable.c_str());