C++ Strings and the For-each Loop

by Dr. Brian Fraser

text: Ch 8 of Problem Solving with C++ by Walter Savitch (9th ed)
Topics

1) What can we do with a text string?
2) Is there a better way to iterate through a string?
String

- string: a class which is similar to a vector of characters.
  string myName = "Dr. Evil";

- Get the number of characters in a string:
  int x = myName.size();
  - .size() actually returns an unsigned int.
    Examples here will use unsigned int to avoid warnings.

- Get character by index (0 indexed):
  char ch = myName.at(4);       // 5th character = 'E'
  - at() member function does range checking:
    myName.at(10);               // Generates runtime error!
  - [] might not.. check if index is valid:
    char oops = myName[10];      // Undefined behaviour
String

- Example: Print each character in a string
  ```
  string myName = "Dr. Evil";
  for (unsigned int i = 0; i < myName.size(); i++) {
    cout << i << "": " << myName.at(i) << endl;
  }
  ```

  Output:
  ```
  0: D
  1: r
  2: .
  3:
  4: E
  5: v
  6: i
  7: l
  ```
Read in Strings

• Read a string from the keyboard
  string word;
  cin >> word;
  - This reads characters until it hits whitespace:
    a space, tab, or new-line character

• Read a full line until new-line character:
  string fullLine;
  getline(cin, fullLine);
**Concatenate**

- + is Concatenate: join together two strings
  
  ```cpp
type first = "Dr.";
  string last = "Evil"
  string name = first + " " + last;
  ```

- Can use +=
  
  ```cpp
  vector<string> names = {"Larry", "Curly", "Moe"};
  string allNames = "{";
  for (unsigned int i = 0; i < names.size(); i++) {
    if (i > 0) {
      allNames += ", ";
    }
    allNames += names.at(i);
  }
  allNames += "}";
  cout << allNames << endl;
  ```

  Requires compiler flag
  `-std=c++11`

  Output:

  `{Larry, Curly, Moe}`
Compare Strings

- Check if equal:
  ```
  string airShieldCode;
  cin >> airShieldCode;
  if (airShieldCode == "12345") {
      cout << "Access granted!";
  }
  ```

- Compare lexicographically: character by character
  ```
  // Pick who goes first
  string yourName = ....
  string myName = .....;
  if (yourName < myName) {
      cout << "You go first.\n";
  } else {
      cout << "Me first!\n";
  }
  ```
#include <iostream>
using namespace std;

bool containsChar(const string &str, char ch) {
    for (unsigned int i = 0; i < str.size(); i++) {
        if (str.at(i) == ch) {
            return true;
        }
    }
    return false;
}

int main() {
    cout << boolalpha;
    cout << "'e' in ""Hello""? " << containsChar("Hello", 'e') << endl;
    cout << "'a' in ""Hello""? " << containsChar("Hello", 'a') << endl;
    cout << "'H' in ""Hello""? " << containsChar("Hello", 'H') << endl;
}
for-each loop
For-each loop

- C++11 adds a new kind of loop to loop over all elements in a vector or characters in a string, or sometimes elements in an array.

- Example

```cpp
string message = "Be nice";
for (char ch : message) {
    char asUpper = toupper(ch);
    cout << asUpper << "-";
}
cout << endl;
```

“For each characters ch in message”

toupper() converts a character to upper case.

Output:

```
B-E- -N-I-C-E-
```

- for-each loops vs for loops:
  - ..eliminate the index variable
  - ensure your loop.. stays within range (< vs <= ?!?)
  - requires compiler flag: `-std=c++11`
for-each on Vectors

• For-each works on Vectors as well
  // Requires compiler flag: -std=c++11
  vector<double> itemCosts = {5.15, 2.55, 1.21, 5.00};
double sum = 0;
for (double cost : itemCosts) {
  sum += cost;
}

• This is the same as the following for-loop:
  vector<double> itemCosts = {5.15, 2.55, 1.21, 5.00};
double sum = 0;
for (unsigned int i = 0; i < itemCosts.size(); i++) {
  double cost = itemCosts.at(i);
  sum += cost;
}
String Coding Examples

- Code the following functions in C++
  - use a for-each loop when applicable.

  // Return true if `str1` and `str2` are character for character identical
  bool equal(const string &str1, const string &str2);

  // Return a new string with characters of `str` in reverse order.
  string reverseString(const string &str);

  // Return true if the string `str` is the same forwards as backwards.
  // Example "ABBA"
  bool isPalindrome(const string &str);
Summary

• Strings store a sequence of characters
  – Access characters using .at(), or []
  – Read in: >> and getline()
  – Concatenate: +, +=
  – Compare: ==, <, >

• For-each loop
  – Removes the loop index variable
  – Ensures the loop stays within range