Ch 10. Characters and the string class

CS 2308
Fall 2016
Jill Seaman

Characters

- Built-in data type
- Value: a single character
- Literals: ‘a’, ‘!', ‘\n’, ‘8’, ...
- Operations:
  - assignment: =
  - compare: ==, <, etc.
  - implicit conversion to/from int: uses the ascii code

```
char ch;
ch = 'a';
if (ch=='A') ...
```

```
char ch = 'A';
cout << ch + 10 <<endl;
cout << static_cast<char>(ch+10) << endl;
```

Output: 75

10.1 Character Testing

- The C library provides several functions for testing characters.
- Requires the cctype header file
- They take a char (or int as ascii) argument
- They return true or false and can be used as boolean expressions in if/while/etc.:

```
char input;
...
if (isupper(input)) ...
```
10.2 Character Case conversion

- These take a char (or int), and return an int(!)

- `toupper(c)`
  - if c is lowercase, returns (the ascii value of) its uppercase version
  - otherwise returns c

- `tolower(c)`
  - if c is uppercase, returns (the ascii value of) its lowercase version
  - otherwise returns c

- Does NOT change argument (c)

```cpp
char x = 'A';
char y = tolower(x); //converts to char
cout << x << " " << y << endl;
```

Output: A a

10.7 More about the C++ string class

- `string` is a data type provided by the C++ library.
  - NOT in C!

- string requires the `<string>` header file
  - `<iostream>` may work as well

- To define a string variable in C++:
  - string name1;
  - name1 is called a string object.
  - initialized to the empty string (its size is 0)

- The representation in memory of a string object is hidden from the programmer.

Operations over string objects

- **Initialization** using = with a string literal or variable

```cpp
string name1 = "Steve Jobs"; //string lit
string name2 = name1;     //string var
```

- **Assignment** using = with a string literal or variable

```cpp
string name1, name2, name3;
name1 = "Steve Jobs"; //string lit
name2 = name1;       //string var
```

Operations over string objects

- output using `<>`

- input using `>>`
  - input stops at whitespace (space, tab, newline)!

- input using `getline(cin,string);`
  - input starts at end of previous input (may need to use `cin>>ws` or `cin.ignore(…) to consume newline first`.)
  - input stops at first newline character

```cpp
string name1;
ocout << "Enter your name ";
generate (cin, name1);
```
Operations over string objects

- comparing string objects: < <= > >= == !=
- follows alphabetical order using ascii values:
  - if first two characters are the equal, it compares the
    second two characters, etc.

```cpp
string string1, string2;
string1 = "Hello ";
string2 = "World!";
if (string1 < string2)
  cout << "Hello comes before World" << endl;
if (string1 == "Hello")
  cout << "It was Hello." << endl;
```

More operations over string objects

- [n] subscript notation, returns char at position n
- or use string::at(n) -- performs bounds check

```cpp
string string1 = "Hello ";
cout << string1[4] << endl;
cout << string1.at(1) << endl;
```

string class member functions

- see Table 10-7 for more

- theString.length(): returns length of string stored in
  theString (can also use .size()).

```cpp
string theString = "Hello";
cout << theString.length() << endl; //outputs 5
for (int i=0; i<theString.size(); i++)
  cout << toUpper(theString.at(i));
cout << endl;
```

appending/concatenating strings

- a + b returns a string that is the concatenation of strings a
  and b (does not change either).
- a+=b appends a copy of b to end of string a.
  - changes a, b can be a char
- a.append(b): appends b to the end of a (like a+=b)
  - changes a, b cannot be a char

```cpp
string theString = "Hello";
cout << theString + " there"; //outputs: Hello there
theString.append(" World"); //changes theString
cout << theString << endl; //outputs: Hello World
theString += '!'; //adds ! character to end
cout << theString << endl; //outputs: Hello World!
```
Exercise (watchout)

- Write a function toLowerString that takes a string `p` as an argument and returns a NEW string that is a copy of `p` with all of its uppercase letters converted to lowercase.

  What is wrong with this solution?

  ```cpp
  string toLowerString (string p) {
    string newP;
    for (int i=0; i < p.length(); i++)
      newP.at(i) = tolower(p.at(i));
    return newP;
  }
  ```

  ```none
  newP += tolower(p.at(i));
  ```

  Runtime error:
  terminate called throwing an exception
  Abort trap: 6

  ```cpp
  ```