

EECS402 Lecture 08

Andrew M. Morgan

Savitch Ch. 9.0 C++ String Data Type C-Strings



Intro To The Standard string Class

물론() 목02

- C++ has a standard class called "string"
- Strings are simply a sequence of characters
 - Note: This is not a sufficient definition for a "C-string"
 - A "C-string" is an array of characters terminated by a null byte
 More on this later...
- Must #include <string> using the standard namespace to get C++ standard string functionality
 - Note: This is different from #include'ing <string.h> which is the header required for "C-string"s
- string variables are used to store names, words, phrases, etc.
- Can be input using ">>" and output using "<<" as other types

402 402

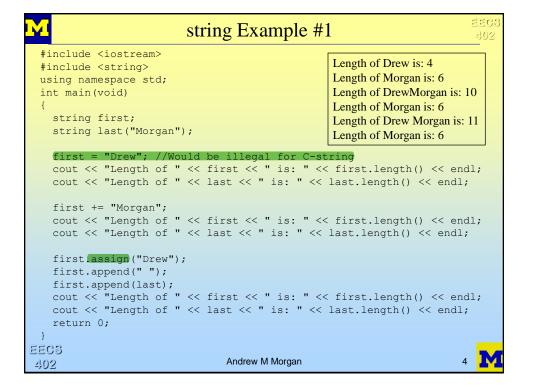
Andrew M Morgan

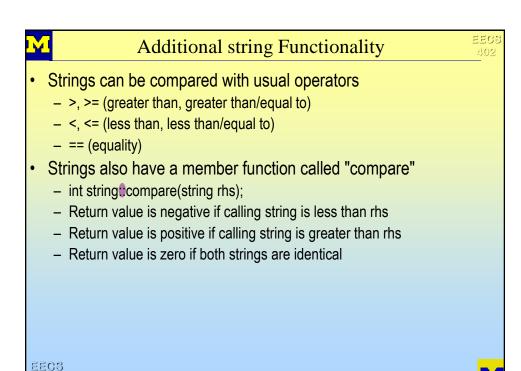
M

```
Some string Functionality
   Declaring a string:
    string lastName;

    string firstName("Drew"); //Note: String literal enclosed in double quotes

    string fullName;
   Assigning a string:
    lastName = "Morgan"; //The usual assignment operator
  Appending one string on the end of another:
    fullName = firstName + lastName; //Results in "DrewMorgan"
    - fullName = firstName + " " + lastName; //Results in "Drew Morgan"
  Accessing individual characters in a string:
    myChar = firstName[2]; //Results in 'e' (no bounds checking)
    myChar = firstName.at(2); //Results in 'e' (does bounds checking)
 Appending a character to the end of a string:
    lastName = lastName + myChar; //Results in "Morgane"
  Determining number of characters in string:
    myInt = firstName.length(); //Results in 4
EECS
                                    Andrew M Morgan
703
```

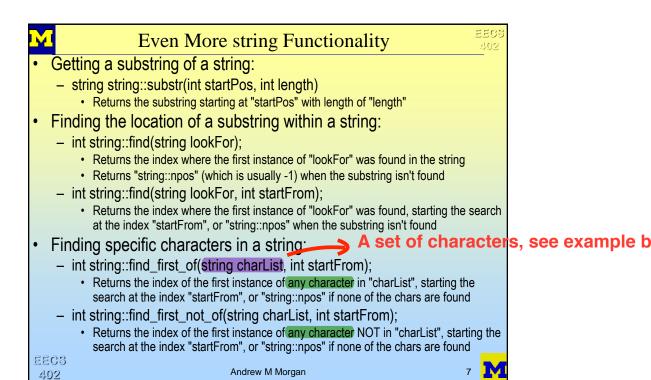




Andrew M Morgan

703

```
string Example #2
                                                 s3 = "Drew";
int main (void)
                                                 if (s3 < s1)
  string s1 = "Drew";
                                                   cout << "oper: s3 less than s1";</pre>
  string s3;
                                                 if (s3 > s1)
  int result;
                                                  cout << "oper: s3 greater than s1";
                                                 if (s3 == s1)
  s3 = "Bob";
                                                   cout << "oper: s3 is equal to s1";</pre>
  if (s3 < s1)
                                                 cout << endl;
   cout << "oper: s3 less than s1";</pre>
  if (s3 > s1)
                                                 result = s3.compare(s1);
   cout << "oper: s3 greater than s1";</pre>
                                                if (result < 0)
                                                  cout << "comp: s3 less than s1";
  if (s3 == s1)
    cout << "oper: s3 is equal to s1";</pre>
                                                 else if (result > 0)
  cout << endl;
                                                  cout << "comp: s3 greater than s1";</pre>
                                                 else
  result = s3.compare(s1);
                                                   cout << "comp: s3 is equal to s1";</pre>
  if (result < 0)
                                                 cout << endl;
   cout << "comp: s3 less than s1";</pre>
  else if (result > 0)
                                                 return 0;
   cout << "comp: s3 greater than s1";</pre>
  else
    cout << "comp: s3 is equal to s1";</pre>
                                                          oper: s3 less than s1
  cout << endl;
                                                          comp: s3 less than s1
                                                          oper: s3 is equal to s1
                                                          comp: s3 is equal to s1
EECS
                                      Andrew M Morgan
402
```



```
string Example #3
nt main()
                                          cout << "Spaces:";
                                         spaceLoc = myStr.find(" ");
 int startPos;
                                         while (spaceLoc != string::npos)
 int len;
                                           cout << " " << spaceLoc;
 int commaLoc;
                                           spaceLoc = myStr.find(" ", spaceLoc + 1);
 int howLoc;
 int loc;
 int spaceLoc;
                                         cout << endl;
 string myStr;
 string myStr2;
                                         cout << "Punct and spaces:";</pre>
                                         loc = myStr.find first of(" ,?", 0);
 myStr = "Hello, how are you?";
                                         while (loc != string::npos)
 startPos = 7;
                                           cout << " " << loc;
 len = 3;
 myStr2 = myStr.substr(startPos, len);
                                           loc = myStr.find_first_of(" ,?", loc + 1);
 cout << "Substr: " << myStr2 << endl;</pre>
 commaLoc = myStr.find(",");
                                         cout << endl;
 howLoc = myStr.find(myStr2);
cout << "Comma: " << commaLoc;</pre>
                                         return 0;
 cout << " how: " << howLoc << endl; }</pre>
                       Substr: how
                       Comma: 5 how: 7
                       Spaces: 6 10 14
                       Punct and spaces: 5 6 10 14 18
EECS
                                    Andrew M Morgan
402
```



string Class Implementation

보면() 402

- The string class uses dynamic memory allocation to be sure segmentation faults don't occur
 - When a string is updated such that it requires more characters than currently allocated, a new, larger array is allocated and the prior contents are copied over as necessary
- Since dynamic allocation is relatively slow, it is not desirable to be re-allocating strings often
 - C++ allows some memory to be "wasted" by often allocating more space than is really needed
 - However, as strings are appended to the end, it is likely that a reallocation won't be needed every time
 - Occasionally, re-allocation is necessary and is performed, again allocating more memory than necessary
- Note: this is all done automatically by the string class

402 402

Andrew M Morgan



M

Some Final string Functionality

롤롤(3) 402

- Several member functions are available to get information about a string
 - capacity: The number of characters that can be placed in a string without the inefficiency of re-allocating
 - length: The number of characters currently in the string
- You can manually change the capacity of a string
 - resize: Sets the capacity of a string to be at least a user-defined size
 - This can be useful if you know a string will be at most n characters long
 - By resizing the string to capacity *n* only that amount of memory is associated with the string
 - · This prevents wasted memory when you know the exact size you need
 - Additionally, it can help prevent numerous re-allocations if you will be appending on to the end of the string, but know the final size ahead of time

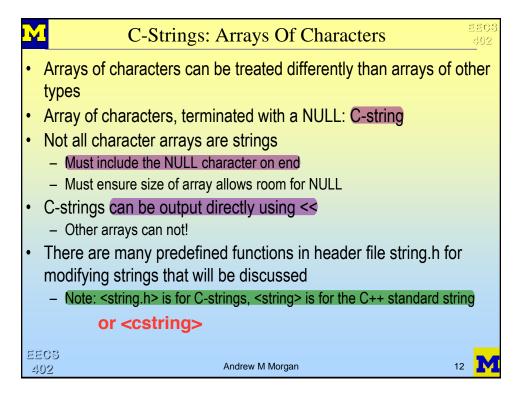
402 402

Andrew M Morgan

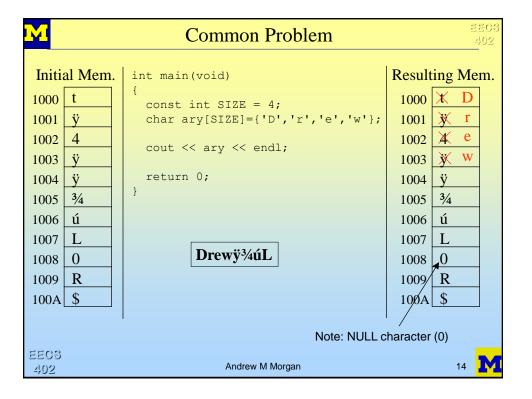
0

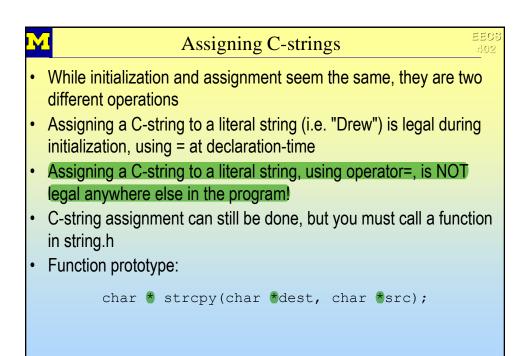


```
string Example #4
int main(void)
                                                  str += "-111-":
                                                  cout << "Str: " << str << endl;
                                                  cout << "Length: " << str.length();</pre>
 string str;
                                                  cout << " Cap: " << str.capacity();</pre>
 string str2;
                                                  cout << endl;
 cout << "Str: " << str << endl;
 cout << "Length: " << str.length();</pre>
                                                 str += "1723-9";
 cout << " Cap: " << str.capacity();</pre>
                                                 cout << "Str: " << str << endl;
 cout << endl;
                                                  cout << "Length: " << str.length();</pre>
                                                 cout << " Cap: " << str.capacity();</pre>
 str = "888";
                                                 cout << endl;
 cout << "Str: " << str << endl;
 cout << "Length: " << str.length();</pre>
                                                 str += "abcdefghijklmnopqrstuv";
 cout << " Cap: " << str.capacity();
                                                 cout << "Str: " << str << endl;
                                                  cout << "Length: " << str.length();</pre>
                                                 cout << " Cap: " << str.capacity();</pre>
                                                  cout << endl;
        Length: 0 Cap: 0
        Str: 888
                                                  return 0:
        Length: 3 Cap: 31
        Str: 888-111-
        Length: 8 Cap: 31
        Str: 888-111-1723-9
        Length: 14 Cap: 31
        Str: 888-111-1723-9abcdefghijklmnopqrstuv
        Length: 36 Cap: 63
EECS
                                      Andrew M Morgan
402
```



```
Simple C-string Program
 int main (void)
                                                 0 works too
    const int SIZE = 5;
    int i = 0;
    int iary[SIZE] = \{2,4,6,8,10\}; //NOT a C-string
    char cary[SIZE] = {'D', 'r', 'e', 'w', '\0'}; //IS a C-string
    char cary2[SIZE] = "Drew"; //NULL automatic! - IS a C-string
    char cary3[SIZE] = {'H', 'e', 'l', 'l', 'o'}; //NOT a C-string
    cout << iary << endl;</pre>
                                              0xffbef8e0
    cout << cary << endl;</pre>
                                              Drew
    cout << cary2 << endl;</pre>
                                              Drew
    return 0;
                               These are POTENTIAL results. First line is an
                               address of the first element of the iary.
                               Your specific results will vary (printing a
                               different address)
EECS
                               Andrew M Morgan
402
```





Andrew M Morgan

EECS

703

```
Using strcpy()

const int SIZE=5;
char cary[SIZE] = "Drew"; //Legal here!
char cary2[SIZE];

//cary2 = "Drew"; //ACK! Don't do this!

//strcpy() automatically appends a NULL
//character to the end of the string.
strcpy(cary2, "Drew"); //Ahh. Much better.

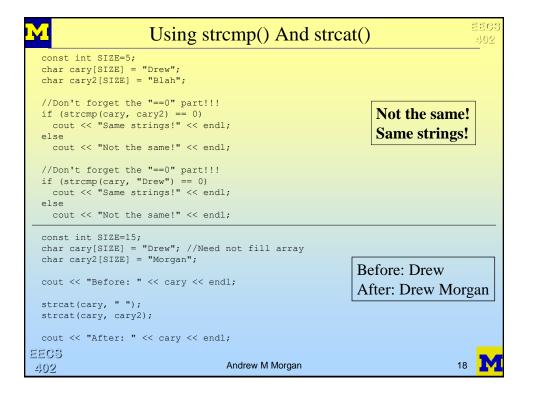
cout << cary << endl;
cout << cary2 << endl;
cout << cary2 << endl;
Drew
Drew

EEC3
402

Andrew M Morgan

16
```

Comparing and Appending Two Strings Like assignment, comparison is not allowed with the "==" operator Since these "strings" are really just character pointers, these operators would work on the pointer values (addresses), rather than the contents Must call a function from string.h. Prototype: int strcmp(char *s1, char *s2); s1 and s2 are C-strings (char arrays, terminated with NULL) Return integer is: · 0 if strings are the same negative if s1 is "less than" s2 (not the same) positive is s1 is "greater than" s2 (not the same) Appending strings is done with a funtion. Prototype: char * strcat(char *s1, char *s2); s1 and s2 are C-strings (char arrays, terminated with NULL) If s1 was "Drew" and s2 was "Morgan" then after a call to strcat, s1 would contain "DrewMorgan" and s2 is unchanged EECS Andrew M Morgan 703



```
Finding The Length of a String

• You often want to know how long a string is. Prototype:

int strlen(char *s);

- The int being returned is the length of the string

- It is not the length of the array

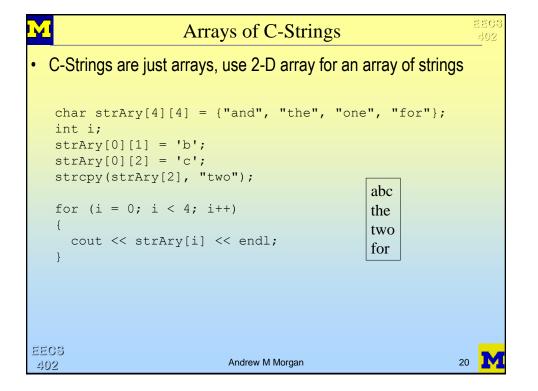
- It is not the length of the string including the NULL

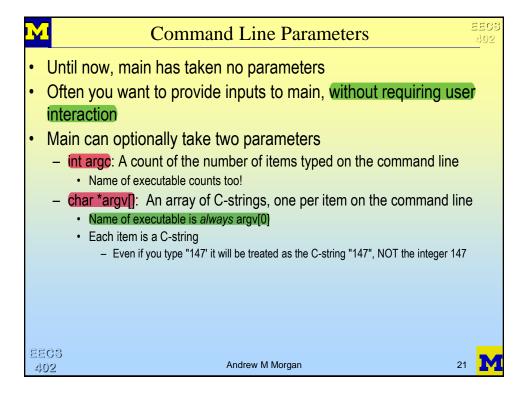
• Example:

cary[15] = "Drew";

cout << "Length: " << strlen(cary) << endl;

Length: 4
```



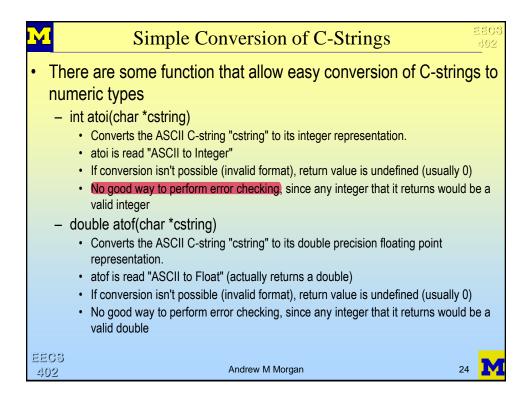


```
Command Line Parameters - Example
    int main(int argc, char *argv[])
      int i;
      for (i = 0; i < argc; i++)
         cout << "Item #" << i << ": " << argv[i] << endl;</pre>
      return 0;
                     cmdPrompt% ./cmdLineDemo
                     Item #0: ./cmdLineDemo
                     cmdPrompt%./cmdLineDemo hello 147 88.3
                     Item #0: ./cmdLineDemo
                     Item #1: hello
                     Item #2: 147
                     Item #3: 88.3
EECS
                              Andrew M Morgan
402
```

```
Usage Statements In Programs
 If your program requires input parameters, it should include a
   usage statement

    How the user is notified about the correct way to run your program

int main(int argc, char *argv[])
  ifstream inFile;
 if (argc != 3)
   cout << "Usage: " << argv[0] << " <inputFilename> <value>" << endl;</pre>
                                          cmdPrompt% usageDemo
                                          Usage: usageDemo <inputFilename> <value>
 inFile.open(argv[1]);
 cout << "Value is: " << argv[2] << endl;</pre>
                                          cmdPrompt% usageDemo infile.txt
                                          Usage: usageDemo <inputFilename> <value>
  //... more code to read
 //... input file, etc...
                                          cmdPrompt% usageDemo infile.txt 45 extraInfo
 return 0;
                                          Usage: usageDemo <inputFilename> <value>
                                          cmdPrompt% usageDemo infile.txt 45
                                          Value is: 45
                                                       This is a cstring
EECS
                                   Andrew M Morgan
703
```



Simple Conversion of C-Strings, Example int main(int argc, char *argv[]) int intVal; double doubleVal; if (argc != 3) cout << "Usage: " << argv[0] << " <floatVal> <intVal>" << endl;</pre> exit(2);doubleVal = atof(argv[1]); intVal = atoi(argv[2]); cout << doubleVal << " / " << intVal << = " << (doubleVal / intVal) << endl;</pre> Bad # params return 0; cmdPrompt% cstrConvDemo <-Good run Usage: **/**cstrConvDemo <floatVal> <intVal> cmdPrompt% cstrConvDemo 92.75 16 Non-parseable 92.75 / 16 = 5.79688 values typically cmdPrompt% cstrConvDemo hello 18 result in 0 0/18 = 0cmdPrompt% cstrConvDemo 92.75 16.75 ← atoi converts only 92.75 / **16** = 5.79688 as much as it can to an int EECS Andrew M Morgan 402