

Intro To The Standard string Class

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

Some string Functionality

Declaring a string:
- string lastName;
- string firstName("Drew"); //Note: String literal enclosed in double quotes
- string fullName;
- 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"

Andrew M Morgan

Determining number of characters in string:

myInt = firstName.length(); //Results in 4

FFCS

402

EEC3

5

string Example #1 #include <iostream> Length of Drew is: 4 #include <string>
using namespace std; Length of Morgan is: 6 Length of DrewMorgan is: 10 int main (void) Length of Morgan is: 6 string first; string last("Morgan"); Length of Drew Morgan is: 11 Length of Morgan is: 6 first = "Drew"; //Would be illegal for C-string
cout << "Length of " << first << " is: " << first.length() << endl;
cout << "Length of " << last << " is: " << last.length() << endl;</pre> first += "Morgan";
cout << "Length of " << first << " is: " << first.length() << endl;
cout << "Length of " << last << " is: " << last.length() << endl;</pre> first.assign("Drew");
first.append(" "); first.append(last);
cout < "length of " << first << " is: " << first.length() << endl;
cout < "length of " << last << " is: " << last.length() << endl;</pre> return 0: Andrew M Morgan 402

Additional string Functionality

Strings can be compared with usual operators

- >, >= (greater than, greater than/equal to)

- <, <= (less than, less than/equal to)

- == (equality)

Strings also have a member function called "compare"

- int string::compare(string rhs);

- Return value is negative if calling string is less than rhs

- Return value is positive if calling string is greater than rhs

- Return value is zero if both strings are identical

See next slide for what it means for how strings are compared

Andrew M Morgan

Ordering Of Strings String ordering is based on the numerical 64 65 66 67 68 69 70 71 72 73 74 75 77 77 78 80 81 82 83 84 85 86 87 88 89 99 91 92 93 representation of individual chars in the strings - The standard ASCII table provides the mapping - Determined via the first char that is different - Shorter strings are "less than" longer strings if they are the same otherwise 160 170 180 190 200 211 225 23E 240 251 261 271 281 291 301 Examples: - "Hello" < "hello" "good" < "goodbye" "See ya" < "cya" - "40" < "400" "(wow)" < "[wow]" - "abc123" == "abc123" EECS 402 6 1 Andrew M Morgan

6

5 **M**

2

4

```
M
                                                                   string Example #2
   int main(void)
                                                                                                            s3 = "Drew";
if (s3 < s1)
    cout << "oper: s3 less than s1";
if (s3 > s1)
    cout << "oper: s3 greater than s1";
if (s3 == s1)
    cout << "oper: s3 greater than s1";
cout << "oper: s3 is equal to s1";
cout << end1;</pre>
      string s1 = "Drew";
string s3;
int result;
     s3 = "Bob";
if (s3 < s1)
  cout << "oper: s3 less than s1";
if (s3 > s1)
  cout << "oper: s3 greater than s1";
if (s2 = c2)</pre>
                                                                                                            result = s3.compare(s1);
if (result < 0)
  cout << "comp: s3 less than s1";
else if (result > 0)
  cout << "comp: s3 greater than s1";</pre>
     cout << "oper: s3 greater than s1"
if (s3 == s1)
cout << "oper: s3 is equal to s1";
cout << endl;
    result = s3.compare(s1);
if (result < 0)
cout << "comp: s3 less than s1";
else if (result > 0)
cout << "comp: s3 greater than s1";
else
                                                                                                                   cout << "comp: s3 is equal to s1";
                                                                                                             cout << endl;
                                                                                                             return 0;
      cout << "comp: s3 is equal to s1";
cout << end);</pre>
                                                                                                                                oper: s3 less than s1
                                                                                                                                comp: s3 less than s1
oper: s3 is equal to s1
                                                                                                                               comp: s3 is equal to s1
EECS
                                                                                    Andrew M Morgan
                                                                                                                                                                                        7 M
```

Even More string Functionality Getting a substring of a string: string string::substr(int startPos, int length) Returns the substring starting at "startPos" with length of "length" Finding the location of a substring within a string: int string::find(string lookFor); Returns the index where the first instance of "lookFor" was found in the string Returns "string::npos" (which is usually -1) when the substring isn't found int string::find(string lookFor, int startFrom); Returns the index where the first instance of "lookFor" was found, starting the search at the index "startFrom", or "string::npos" when the substring isn't found · Finding specific characters in a string: int string::find first of(string charList, int startFrom); Returns the index of the first instance of any character in "charList", starting the search at the index "startFrom", or "string::npos" if none of the chars are found int string::find_first_not_of(string charList, int startFrom); Returns the index of the first instance of any character NOT in "charList", starting the search at the index "startFrom", or "string::npos" if none of the chars are found EECS Andrew M Morgan

7

9

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

C-Strings: Arrays Of Characters

Arrays of characters can be treated differently than arrays of other types

Array of characters, terminated with a NULL: C-string

Not all character arrays are strings

Must include the NULL character on end

Must ensure size of array allows room for NULL

C-strings can be output directly using <

Other arrays can not!

There are many predefined functions in header file string.h for modifying strings that will be discussed

Note: <string.h> is for C-strings, <string> is for the C++ standard string

Another option for C-strings is to #include <cstring> using namespace std

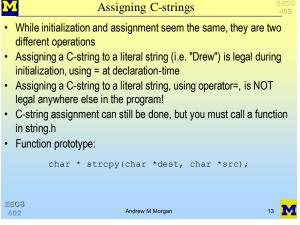
Simple C-string Program int main (void) const int SIZE = 5; int i = 0;
int iary[SIZE] = {2,4,6,8,10}; //NOT a C-string
char cary[SIZE] = {'D','r','e','w','\0'}; //IS a C-string
char cary2[SIZE] = "Drew"; //NULL automatic! - IS a C-string
char cary3[SIZE] = {'H','e','l','l','o'}; //NOT a C-string cout << iary << endl; cout << cary << endl;</pre> 0xffbef8e0 cout << cary2 << endl; 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 11 M Andrew M Morgan

Common Problem Initial Mem. int main(void) Resulting Mem. 1000 1000 X D const int SIZE = 4;
char ary[SIZE]={'D','r','e','w'}; 1001 💥 1001 ÿ r 1002 4 1002 cout << ary << endl; w 1003 ÿ 1003 return 0; 1004 ÿ 1004 ÿ 1005 3/4 1005 3/4 1006 1006 ú ú 1007 L 1007 L Drewÿ¾úL 1008 <u>0</u> 1009 R 1008 0 1009 R 100A \$ 100A \$ Note: NULL character (0) EECS 402 Andrew M Morgan 12

12

8

10



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

BECS
402

Andrew M Morgan

14

13

```
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
                                Andrew M Morgan
402
```

Using strcmp() And strcat() const int SIZE=5;
char cary[SIZE] = "Drew";
char cary2[SIZE] = "Blah"; //Don't forget the "==0" part!!!
if (strcmp(cary, cary2) == 0)
 cout << "Same strings!" << endl;</pre> Not the same! Same strings! else
cout << "Not the same!" << endl; //Don't forget the "==0" part!!!
if (strcmp(cary, "Drew") == 0)
 cout << "Same strings!" << endl;</pre> else
 cout << "Not the same!" << endl;</pre> const int SIZE=15;
char cary[SIZE] = "Drew"; //Need not fill array
char cary2[SIZE] = "Morgan"; Before: Drew cout << "Before: " << cary << endl; After: Drew Morgan strcat(cary, " ");
strcat(cary, cary2); cout << "After: " << cary << endl; Andrew M Morgan 402

15

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

EECS

Andrew M Morgan

17

17

Arrays of C-Strings

• C-Strings are just arrays, use 2-D array for an array of strings

char strAry[4][4] = {"and", "the", "one", "for"};
int i;
strAry[0][1] = 'b';
strAry[0][2] = 'c';
strcpy(strAry[2], "two");

for (i = 0; i < 4; i++)
{
 cout << strAry[i] << endl;
}

EECS

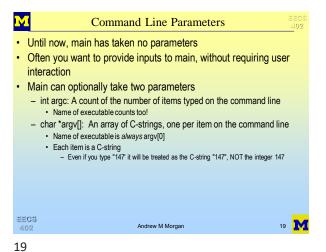
Andrew M Morgan

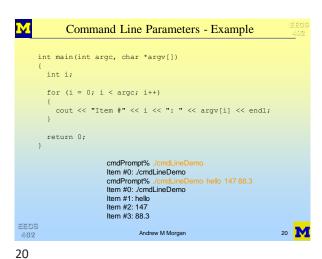
18

18

14

16





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;
exit(0);</pre> cmdPrompt% usageDemo Usage: usageDemo <inputFilename> <value> inFile.open(argv[1]);
cout << "Value is: " << argv[2] << end1;</pre> cmdPrompt% Usage: usageDemo <inputFilename> <value> //... more code to read
//... input file, etc...
return 0; Usage: usageDemo <inputFilename> <value>

Simple Conversion of C-Strings There are some function that allow easy conversion of C-strings to numeric types int atoi(char *cstring) · Converts the ASCII C-string "cstring" to its integer representation. · atoi is read "ASCII to Integer" • If conversion isn't possible (invalid format), return value is undefined (usually 0) No good way to perform error checking, since any integer that it returns would be a valid integer double atof(char *cstring) Converts the ASCII C-string "cstring" to its double precision floating point representation. • atof is read "ASCII to Float" (actually returns a double) • If conversion isn't possible (invalid format), return value is undefined (usually 0) · No good way to perform error checking, since any integer that it returns would be a EECS 402 22 Andrew M Morgan

21

Andrew M Morgan

402

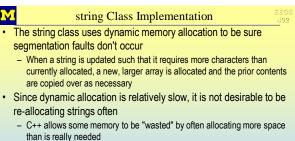
cmdPrompt% usageDemo infile.txt 45 Value is: 45

22

Simple Conversion of C-Strings, Example main(int argc, char *argv[]] int intVal; double doubleVal; if (argc != 3) cout << "Usage: " << argv[0] << " <floatVal> <intVal>" << endl;</pre> doubleVal = atof(argv[1]);
intVal = atoi(argv[2]);
cout << doubleVal << " / " << intVal << = " << (doubleVal / intVal) << end1;</pre> Bad # params cmdPrompt% Usage: ./cstrConvDemo <floatVal> <intVal> Good run cmdPrompt% c Non-parseable 92.75 / 16 = 5.79688 values typically cmdPrompt% c 0 / 18 = 0 result in 0 cmdPrompt% atoi converts only 92.75 / 16 = 5.79688 as much as it can to an int EEC3 23 Andrew M Morgan 23

Additional Reference Material 24 Andrew M Morgan

24



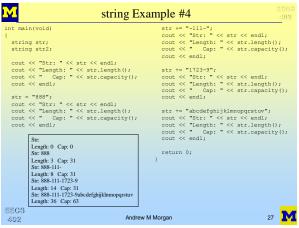
- than is really needed

 However, as strings are appended to the end, it is likely that a re-
- allocation won't be needed every time

 Occasionally, re-allocation is necessary and is performed, again
- Occasionally, re-allocation is necessary and is performed, again allocating more memory than necessary
- Note: this is all done automatically by the string class



25



⁴⁰

