Strings and Characters in C and C++

CS 16: Solving Problems with Computers I
Lecture #10

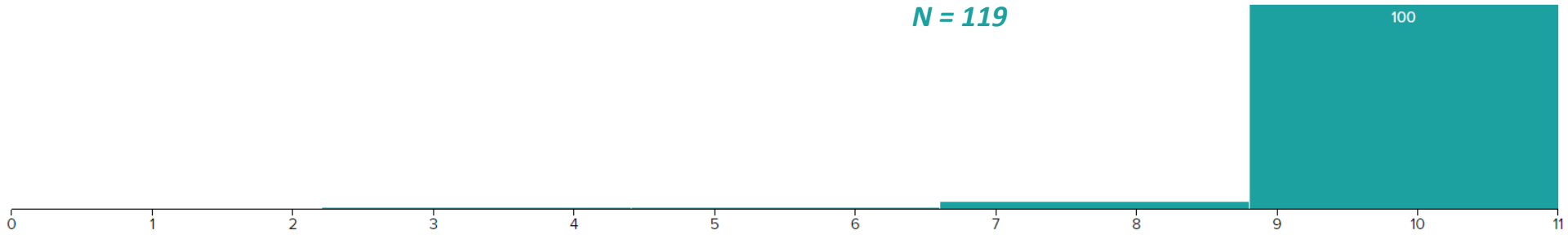
Ziad Matni
Dept. of Computer Science, UCSB

```
int main(int argc, char *argv[])
{
    if (argc > 1)
        filename = argv[1];
    ifstream setIn(filename);
    ifstream vecIn(filename);
    set<string> wordSet = getWordSet(setIn);
    vector<string> wordVec = getWordVec(vecIn);
    map<string, string> wordMap = generateMap(wordVec);

    string name = filename.substr(0, filename.size() - 4);
    string setFilename = name + "_set.txt";
    string vecFilename = name + "_vec.txt";
    string mapFilename = name + "_1_1.txt";

    // Writes set file
    ofstream setOut(setFilename);
    for (set<string>::iterator it = wordSet.begin(); it != wordSet.end(); it++)
    {
        setOut << *it << endl;
    }
    setOut.close();

    // Writes vector file
    ofstream vecOut(vecFilename);
    for (int i = 0; i < wordVec.size(); ++i)
    {
        vecOut << wordVec[i] << endl;
    }
    vecOut.close();

    // Writes to map
    ofstream mapOut(mapFilename);
    printMap(wordMap, mapOut);
    mapOut.close();

    // Generate and print random string
    string str = "";
    for (int i = 0; i < 100; i++)
    {
        cout << wordMap[str] << " ";
        str = wordMap[str];
    }
    cout << endl << endl << endl;

    // Generate more intelligent map
    map<string, vector<string>> wordVecMap;
    str = "";
    for (int i = 0; i < wordVec.size(); i++)
    {
        wordVecMap[str].push_back(wordVec[i]);
        str = wordVec[i];
    }
```

Administrative

- New lab (#5) and new homework (#5) are out!
  - New lab is about strings (this lecture) *AND* Bubble Sort (next lecture)
- Homework 4 and Lab 4 were due yesterday
  - How did they go?
- Quiz 5 is on Friday

Quiz 4



- Mean: **10.15/11 (92.3%)**
- Median: **11/11**

Note: amended since the lecture b/c of Q2 removal

Lecture Outline

- C Strings (`<cstring>`) vs C++ Strings (`<string>`)
- Built-in <cctype>, <string> and Other Useful Functions

C-strings

- To declare a **C-string** variable, include <cstring> and use this syntax:

```
char Array_name[ Maximum_C_String_Size + 1 ];
```

  – The "+ 1" reserves the additional character needed by '\0'

- Example:

```
char Message[6];   // Will have "hello" in it
```

C-strings

- With C-Strings, you **<u>cannot</u>** do these:

```
myString = “Hello!”          //direct assignment
if (myString == “Jimbo”)…    //direct comparison
```

  - Instead have to use **`strncpy()` & `strcmp()`** from **`<cstring>`** library
  - Example:

```
strcmp(myString, “Hello!”)  // Returns an int 0 if ==
```

- But you CAN use = and == with **<u>C++</u>** Strings
  - And so many ***more*** useful things! ☺

The Standard C++ **string** Class

- The strings we know and love…
- The **string** *class* allows the programmer to treat strings as a basic data type
  - No need to deal with the implementations of C-strings
- We will discuss many different ***member functions*** in the <string> library that are extremely useful to use
  - Like **.size(), length()**, **.erase()**, **.substr()**, **.find()**, *etc…*

Declaring a String in C++

- You have to include the correct library module with:

```
#include <string>
```

- Declare them (and initialize them) with:

```
string MyString=“”; // Note the use of double-quotes!
```

- Since strings are made up of characters, you can index individual characters in strings (starting at position 0):

If `MyString = “Hello!”`

Then `MyString[0] = ‘H’, MyString[1] = ‘e’,` etc…

String Basics

- Use the **+** operator to ***concatenate*** 2 strings

```
string str1 = “Hello ”, str2 = “world!”, str3;
str3 = str1 + str2;   // str3 will be “Hello world!”
```

- Use the **+=** operator to ***append*** (i.e. add to the end) to a string

```
str1 += “Z”;   // str1 will be “Hello Z”
```

- Call out a character in the string based on **position**, using [ ] braces
  - Recall array indices in C++ start at zero (0)

```
cout << str1[0];   // prints out ‘H’
cout << str2[3];   // prints out ‘l’
```

Built-In String Member Functions

- Search functions
  - **find**, **rfind**, **find_first_of**, **find_first_not_of**
- Descriptor functions
  - **length**, **size**
- Content changers
  - **substr**, **replace**, **append**, **insert**, **erase**

Making a String into an Integer

(but not the other-way around)

- In the <string> library, there are built-in functions called **stoi() and stod()**
  - stoi = string-to-integer
  - stod = string-to-double

Example:

```
string MyStr = "1999";
int n = stoi(MyStr);
n++;
cout << n << endl;   // prints out 2000 as an int

string ThisStr = "99Jojo";
cout << stoi(ThisStr);     // prints out 99 as an int. Still: not recommended

string OtherStr = "Jojo";
cout << stoi(OtherStr);    // WILL CAUSE A RUN-TIME ERROR!!!
```

Making an Integer into a String

- <string> has a function called: to_string()

- Example:

```
int number = 42,  opposite = -number;
string snum1 = to_string(number);
string snum2 = to_string(opposite);

cout << snum2 << endl;
cout << snum1 + “ is the answer to everything!\n”;
```

Character Functions

- Several predefined functions exist to facilitate working with characters

- The **cctype** library is required for most of them

```
#include <cctype>
using namespace std;
```

The **toupper** Function

- **toupper** returns the argument's upper case character
  - `toupper('a')` returns 'A'
  - `toupper('A')` returns 'A'

DOES NOT WORK WITH STRINGS!

IT'S FOR *CHARACTERS* ONLY!

The **tolower** Function

- Similar to **toupper** function…

- **tolower** returns the argument's lower case character
  - `tolower('a')` returns 'a'
  - `tolower('A')` returns ‘a’

- **toupper** and **tolower** actually return an ASCII code
- To print the character, you have to make sure to assign these functions’ outputs to a char type!
  - Example: `char letter = toupper(‘a’);`

Characters and ASCII Code

- In C/C++, characters and their ASCII codes are interchangeable
- For an ASCII code list, see: https://simple.wikipedia.org/wiki/ASCII

- Example:

```
char letter = 65;
cout << letter;     // prints out “A”
```

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

DEMOS!

- cstring.cpp
- string_char.cpp
- conversions.cpp

The **isspace** Function

- **isspace** returns *true* if the argument is whitespace
  - Whitespace is: spaces, tabs, and newlines
    - So, **isspace(' ')** returns true, so does **isspace('\n')**
  - Example:

```
if (isspace(next) )
   cout << '-';
else
   cout << next;
```

Prints a '-' if next contains a space, tab, or newline character

Some Predefined Character Functions in `cctype` (*part 2 of 2*)

Function	Description	Example

`isupper(Char_Exp)`	Returns *true* provided *Char_Exp* is an uppercase letter; otherwise, returns *false*.	`if (isupper(c))`
		`  cout << c << " is uppercase.";`
		`else`
		`  cout << c`
		`  << " is not uppercase.";`

`islower(Char_Exp)`	Returns *true* provided *Char_Exp* is a lowercase letter; otherwise, returns *false*.	`char c = 'a';`
		`if (islower(c))`
		`  cout << c << " is lowercase.";`
		`Outputs: a is lowercase.`

`isalpha(Char_Exp)`	Returns *true* provided *Char_Exp* is a letter of the alphabet; otherwise, returns *false*.	`char c = '$';`
		`if (isalpha(c))`
		`  cout << c << " is a letter.";`
		`else`
		`  cout << c`
		`  << " is not a letter.";`
		`Outputs: $ is not a letter.`

`isdigit(Char_Exp)`	Returns *true* provided *Char_Exp* is one of the digits '0' through '9'; otherwise, returns *false*.	`if (isdigit('3'))`
		`  cout << "It's a digit.";`
		`else`
		`  cout << "It's not a digit.";`
		`Outputs: It's a digit.`

`isspace(Char_Exp)`	Returns *true* provided *Char_Exp* is a whitespace character, such as the blank or newline symbol; otherwise, returns *false*.	`//Skips over one "word" and`
		`//sets c equal to the first`
		`//whitespace character after`
		`//the "word":`
		`do`
		`{`
		`  cin.get(c);`
		`} while (! isspace(c));`

Character Manipulators Work Too!

- Include **<cctype>** to use with, for example, **toupper()**

```
string s = “hello”;
s[0] = toupper(s[0]);
cout << s;   // Will display “Hello”
```

- …or to use with **tolower()**

```
string s = “HELLO”;
for (int i=0; i < 5; i++)
     s[i] = tolower(s[i]);
cout << s;   // Will display “hello”
```

Search Functions: **find** 1

- You can search for a the ***first occurrence*** of a string in a string with the **.find** function

[7]

```
string str = “With a banjo on my knee and ban the bomb-ban!”;
int position = str.find(“ban”);
cout << position;     // Will display the number 7
```

Search Functions: **find** 2

- You can also search for a the ***first occurrence*** of a string in a string, starting at position ***n***, using a slight mod to **.find()**

[28] ?

```
string str = “With a banjo on my knee and ban the bomb-ban!”;
int position = str.find(“ban”, 12);
cout << position;     // Will display the number 28
```

Search Functions: **find** 3

- You can use the **find** function to make sure a substring is ***NOT*** in the target string using the "no position" value

  `string::npos` is returned if no position exists

```
if (MyStr.find("piano") == string::npos)
      cout << "There is no piano there!"
// This will happen if "piano" is NOT in the string MyStr
```

Search Functions: **rfind**

- You can search for a the ***last occurrence*** of a string in a string with the **.rfind** function

[41]

```
string str = “With a banjo on my knee and ban the bomb-ban!”;
int rposition = str.rfind(“ban”);
cout << rposition;    // Will display the number 41
```

Search Functions: find_first_of and find_first_not_of

- **find_first_of**
  - Finds 1st occurrence of ***any*** of the characters included in the specified string

- **find_first_not_of**
  - Finds 1st occurrence of a character that is ***not any*** of the characters included in the specified string

- Example:

See demo file:
non_numbers.cpp

Descriptor Functions: length and size

- The **length** function returns the length of the string
- The member function **size** is the same exact thing…
  - So, if `string str1 = “Mama Mia!”`,
    then `str1.length() = 9`
    and `str1.size() = 9` also

Example – what will this code do?:

```
string name = “Bubba Smith”;
for (int i = name.length(); i > 0; i--)
      cout << name[i-1];
```

Content Changers: **append**

- Use function **append** to append one string to another

```
string name1 = “ Max”;
string name2 = “ Powers”;
cout << name1.append(name2);  // Displays “ Max Powers”
```

- Does the <u>same</u> thing as: **name1 + name2**

Content Changers: erase

- Use function **erase** to clear a string to an empty string

- One use is:
  name1.erase() -- Does the <u>same</u> thing as: **name1 = “”**

- Another use is:
  name1.erase(*start position, how many chars to erase***)**
  - Erases only part of the string
  - Example:

```
string s = “Hello!”;
cout << s.erase(2, 2);    // Displays “Heo!”
```

Content Changers: replace and insert

- Use function **replace** to replace part of a string with another
  - Popular Usage:
    string.replace(***start*** *position, # of places* ***after*** *start position to replace,* ***replacement*** *string*)
- Use function **insert** to insert a substring into a string
  - Popular Usage:
    string.insert(***start*** *position,* ***insertion*** *string*)

Example:

```
string country = "Back in the USSR";      // length is 16
cout << country.replace(14, 2, "A");   // Displays "Back in the USA"
cout << country.insert(15, "BC");      // Displays "Back in the USABC"
```

Content Changers: substr

- Use function **substr** (short for "substring") to extract and return a substring of the **string** object
  - Popular Usage:
    string.substr(***start*** *position, # of* ***places*** *after start position*)

Example:

```
string city = "Santa Barbara";
cout << city.substr(3, 5)    // Displays "ta Ba"
```

See demo file:
funwithstrings.cpp

YOUR TO-DOs

- ❑ Start **Lab 5** today
- ❑ Do **Homework 5**
- ❑ Do **Quiz 5** this week (Fri.)

VOTE!!!

</LECTURE>