CS 101: Computer Programming and Utilization

15-Strings

Instructor: Sridhar Iyer
IIT Bombay
Predict the output of this program

```cpp
int main() {
    char s1[10] = {'H', 'e', 'l', 'l', 'o', '\0', 'a', 'b', 'c', 'd'};
    char s2[] = "Hello";
    char *s3 = "Hello";

    cout << s1 << s2 << s3 << endl;
    cout << *s1 << *s2 << *s3 << endl;
}
// Run: demo15-strings.cpp
Strings
We routinely use strings of characters to represent words, so it is important to handle such data in our programs.

- A string constant is a sequence of zero or more characters enclosed in double quotes - “Hello”
- We have already used these in cout statements.
- Recall – what is the difference between
  - `cout << “Hello”;
  - `cout << “Hello\n”;
  - `cout << “Hello” << endl;`
Representation of Strings

char - basic data type can contain only one character

Strings are a sequence of characters

- Use an array of char to represent strings
- char s[20]; //Avoid using - char s[] or char *s;
- Characters of a string are in consecutive elements
- How will we know when a string has ended?
  - Put an artificial sentinel character at the end - \0
  - C++ automatically assigns \0 at the end when
    - we declare char s[] = “Hello”;
    - or when we read it from input as cin >> s2;
Activity – Reverse a string

Think (individually): Write down the steps involved in reversing a string.

Pair (with neighbour): Write program to read a string, reverse it, and then output the reversed string.

Share: Compare with next slide.
Reversing a string – without using libraries

```c
int main() {
    char s1[100], temp; int i = 0, j = 0;
    cout<<"Enter a string (without spaces) : "; cin>>s1;

    for (j = 0; s1[j] != '\0'; j++) ; //Find length of string in j
    while(i < j/2) {
        temp=s1[i]; s1[i]=s1[j-i-1];
        s1[j-i-1]=temp; i++;
    }
    cout<<"Reverse string is : "<<s1<<endl;
}
```

// Run: demo15-strings.cpp
C - String functions

There is no data type called 'string' in C. A string is simply an array of characters terminated by the null character '\0';

There is a library of functions for dealing with strings. Its header file is <cstring> or “string.h”

strcpy: Copy string
strcat: Concatenate strings
strcmp: Compare two strings
strchr: Locate first occurrence of char in string
strrchr: Locate last occurrence of char in string
strstr: Locate substring
strlen: Get string length

http://www.cplusplus.com/reference/cstring/
C++ - The string data type

- `cout << "Hello world\n"`
  - "Hello world\n" was stored as an array of characters

- A more modern and better way is to use the string data type

- `string msg("Hello world");`

- The system manages the array space for us
  - Can assign and append to strings
  - Can read a position: `cout << msg[pos];`
  - Can write a position: `msg[pos] = 'q';`
Common string operations

- Get the number of characters in the string
  - `msg.size()`

- Get the character at a specific position
  - `msg.at(5)` or `msg[5]`

- Get a substring of the given string
  - `msg.substr(1, 3)`

- Index out of bound?
  - Some operations throw exceptions
  - Some silently truncate
  - Some may return garbage
C++ - reversing a string
Using msg.size() and for loop instead of while loop

```cpp
for (int lx=0, rx=msg.size()-1;
    lx < rx; ++lx, --rx) {
    char tmp = msg[lx];
    msg[lx] = msg[rx];
    msg[rx] = tmp;
}
cout << msg;
```

```
H e l l o
```

```
H e l l o
```

```
H e l l o
```

```
H e l l o
```

```
H e l l o
```

```
H e l l o
```

```
H e l l o
```

```
H e l l o
```

```
H e l l o
```
More string operations

- Find the first (leftmost) or last (rightmost) occurrence of a character
  - `msg.find_first_of('o')`
  - `msg.find_last_of('e')`

- Compare two strings (dictionary or lexicographic order)
  - `msg1.compare(msg2)`
  - Returns an integer
    - Negative if `msg1` should appear before `msg2`
    - Zero if `msg1` and `msg2` are equal
    - Positive if `msg1` should appear after `msg2`

// Run: demo15-strings.cpp
C++ String class methods (functions)

size: Return length of string
length: Return length of string
clear: Clear string
empty: Test if string is empty
at: Get character in string
operator[] Get character of string
operator= Assign to string
operator+= Append to string
(constructor): Construct string object
(destructor): String destructor

http://www.cplusplus.com/reference/string/string
Notes
Switch statement

- if (cond1) {...} else if (cond2) {...} else if (cond3) {...} can get tiring

- Common use is to choose between different statements depending on the value of a variable

```cpp
switch (dayOfWeek) {
    case 0:
        cout << "Sunday";
        break;
    case 6:
        cout << "Saturday";
        break;
    default:
        cout << "Weekday";
}
```

Can be char, int, long int

If no break here, control will “fall through” to the next case

If none of the listed cases match dayOfWeek
Switch: use of break statements

int val = 5;
switch (val) {
    case 5:
        cout << "five\n";
        // suppose we forget this break;
    case 4:
        cout << "four\n";
        break;
    default:
        cout << "default\n";
}
Switch example

```cpp
switch (ch) {
  case 'a': case 'A':
  case 'e': case 'E':
  case 'i': case 'I':
  case 'o': case 'O':
  case 'u': case 'U':
    cout << ch << " is a vowel" << endl;
    break;
  default:
    cout << ch << " is not a vowel" << endl;
}
```

Can use fall-through as a feature to combine many cases
Another switch example

cout << "Enter simple expression: ";
int left;
int right;
char operator;
cin >> left >> operator >> right;
cout << left << " " << operator << " " << right
<< " = ";
switch (operator) {
    case '+': cout << left + right << endl; break;
    case '-': cout << left - right << endl; break;
    case '*': cout << left * right << endl; break;
    case '/': cout << left / right << endl; break;
    default: cout << "Illegal operation" << endl;
}
C String vs C++ String - 1

• A C-string, is an array of characters terminated by the null character '\0'; There is no data type called 'string' in C, but there is a library of functions for dealing with strings represented in this form. Its header file is <cstring> or “string.h”

• A C++ string is an object of the class “string”. We need not care about internal representation of the data. The class provides functions for operations on variables of type string. Its header file is <string>

• Both libraries are available to C++ programs – functions from C++ string library can be expected to work with C-style strings, but not vice versa.
C String vs C++ String - 2

C-strings (#include <cstring>)

Declaring a C-string variable
char str[10];

Initializing a C-string variable
char str2[] = "Send money!";
char str3[] = {'O', 'K', '\0'};

Line above has same effect as:
char str3[] = "OK";

C++ strings (#include <string>)

Declaring a C++ string object
string str;

Initializing a C++ string object
string str1("Call home!");
string str2 = "Send money!";
string str3("OK");

string str4(10, 'x');
C String vs C++ String - 3

Assigning to a C-string variable
Can't do it, i.e., can't do:
char str[10];
str = "Hello!";

Assigning to C++ string obj
string str;
str = "Hello";
str = otherString;

Concatenating two C-strings
strcat(str1, str2);
strcpy(str, strcat(str1, str2));

Concatenating two C++ obj
str1 += str2;
str = str1 + str2;

Copying a C-string variable
char str[20];
strcpy(str, "Hello!");
strcpy(str, otherString);

Copying a C++ string object
string str;
str = "Hello";
str = otherString;
## C String vs C++ String - 4

<table>
<thead>
<tr>
<th>Accessing a single character (C)</th>
<th>Accessing in C++ string</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>str[index]</code></td>
<td><code>str[index]</code></td>
</tr>
<tr>
<td></td>
<td><code>str.at(index)</code></td>
</tr>
<tr>
<td></td>
<td><code>str(index, count)</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Finding the length of a C-string</th>
<th>Length of a C++ string obj</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>strlen(str)</code></td>
<td><code>str.length()</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Output of a C-string variable</th>
<th>Output of a C++ string object</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>cout &lt;&lt; str;</code></td>
<td><code>cout &lt;&lt; str;</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Input of a C-string variable</th>
<th>Input of a C++ string object</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>cin &gt;&gt; s;</code></td>
<td><code>cin &gt;&gt; s;</code></td>
</tr>
<tr>
<td><code>cin.getline(s, numCh+1);</code></td>
<td><code>getline(cin, s);</code></td>
</tr>
<tr>
<td><code>cin.getline(s, numCh+1, 'x');</code></td>
<td><code>getline(cin, s, 'x');</code></td>
</tr>
</tbody>
</table>
Converting between C & C++ Strings

Creating a C++ string object from a C string or string literal:
• Declare the string object and pass the C string or string literal as a constructor argument.

Converting C++ string object to a C string:
• The string class provides a method called c_str() that returns a pointer to the underlying array of characters that holds the contents of the string. The C string returned by this method can not be modified, but it can be used, printed, copied, etc.

```cpp
char s1[20];
string s2 = "My C++ string";
strncpy(s1, s2.c_str());  // Copies C string "My C++ string" into s1
• Use C++ strings whenever possible to avoid array errors
```