C++ Strings

Week 11

Assylbek Jumagaliyev
a.jumagaliyev@iitu.kz

What Are Strings?

- Strings in C++ are a sequence of characters.
- This new class of objects is nothing you haven't seen before. The following are both examples of strings:

  "Hello World\n"

  "xyz 123 *&^#$!"
- Now we are able to manipulate these literals… but first we must

  #include <string>

Strings as Variables

- To create a variable of type string, simply:

  string fullName;

- Assignment, as always is the same:

  fullName = “Simon Crue”;

- Or like before, we could always combine the two with an initialization:

  string fullName = “Simon Crue”;

Input/Output with Strings

- I/O with Strings are as before:

```
string fullName = “”;
cout << “Please enter your name: “;
cin >> fullName; //get the name
cout << “Your name is “ << fullName;
```

Input/Output with Strings

- A common problem with reading strings from user input is that it could contain white spaces. Remember that white space (e.g. space, tab, newline) is treated as termination for cin.

- Take the following code for example:

```
cout << "Enter your full name: ";
string fullname;
cin >> fullname; //only the first name will be read in!!
```

String I/O: getline()

- Fortunately, the string class let's us get around this with the getline() function.

- Syntax: getline(source, destination)
- source is the source of the string. In our case, we want cin here.
- destination is the string variable where we want the string to be read into.

String I/O: getline()

- We can fix our code by rewriting it as follows:

```
cout << “Enter your full name: ”;
string fullname;
getline(cin, fullname);
```

String Operators: Assignment

- Assignment (=): As with before, assign a string to a variable of type string.

```
string oneName = “John”;
string anotherName = oneName;
```

- Both now hold “John”

String Operators: Concatentation

- Concatenation (+): Puts a string on the end of another.

```
string firstName = “John”;
string lastName = “Smith”;
string fullName = firstName + “ ” + lastname;

//the += shorthand also works
string oneString = “2 + 2 = ”;
oneString += “5”;
```

Relational String Operators

==	!=
<	<=
>	>=

- == and != are same as before, but the others are not exactly like usage with numbers…

- For instance, what exactly does it mean if one string is “less than” another?

String Processing

- In addition to giving us a new data type to hold strings, the string library offers many useful string processing methods.

- You can find most of them of them in the site, but here are a few useful ones. www.cplusplus.com

length() and size()

- This method returns the integer length of the string. The length() and size() are the same.

- Example:

```
string s1 = “Super!”;

//the integer 6 will be output.
cout << s1.length() << endl;
cout << s1.size() << endl;
```

at(index)

- This method returns the character at the specified index. Indices start from 0.

- Example:

```
string n = "Vikram";
cout << n.at(0) << endl;    //the character 'V' will be output.
cout << n.at(n.size()-1);//the character 'm' will be output.
```

A Shorthand for at(index)

- As an alternative, we could have also used the following equivalent shorthand:

```
string n = "Vikram";
cout << n[0] << endl; //the character 'V' will be output.
cout << n[n.size()-1]; //the character 'm' will be output.
```

erase(index)

- This method removes all characters from the string starting from the specified index to the end.
- The length of the new string is reset to index!

- Example:

```
string os = "Operating Systems";
os.erase(9);

cout << os << endl; //the string "Operating" is output

cout << os.length() << endl; //length is now 9, the index
```

find(str)

- This method returns the integer index of the first occurrence of the specified string
- Example:

```
string d = "data data data";
cout << d.find("data"); //0 is output
```

find(str, index)

- This method returns the integer index of the first occurrence of the specified string **starting from the specified index.**
- Returns -1 if pattern is not found.
- Example:

```
string d = "data data data";
cout << d.find("data", 1);//5 is output
```

- Why? Because by specifying a starting index of 1, we only consider

  "ata data data"

rfind()

- This method returns The position of the **last** occurrence in the string of the searched content.
- Example:

```
string str = "This is a sentences";
string key = "is";
cout<<str.rfind(key);//5 is output
```

find_first_of()

- Returns the position of the first occurrence in the string of any of the characters searched for.
- Example:

```
string transport = "Cruise Ship";

int index = transport.find_first_of("aeiuo");

for(int i=0; i<transport.length(); i++)
{
   transport[index] = '*';
   index = transport.find_first_of("aeiuo");
  }
cout<<transport;
```

insert(index, str)

- Inserts the specified string at the specified index.
- Example:

```
string animal = “Hippo”;
animal.insert(0, “Happy ”);
cout << animal; //outputs “Happy Hippo”
```

replace(index, n, str)

- Removes n characters in the string starting from the specified index, and inserts the specified string, str, in its place.
- Example:

```
string transport = “Speed Boat”;
transport.replace(0, 5, “Sail”);
cout << transport; //outputs “Sail Boat”
```

substr(index, n)

- Returns the string consisting of n characters starting from the specified index.

- Example:

```
string transport = "Cruise Ship";
cout << transport.substr(7, 4);
//outputs "Ship"
```

Refernces

- **http://www.cplusplus.com/reference/string/string/**
- The Ohio State University, CSE202: Lecture 10

Thanks for your attention!