URL Parser Documentation Savishwa Gaur

Sxg180113

To compile the program use “g++ -std=c++11 URLParser.cpp -o URLParser.cs1” which specifies the version of c++ to use and sets the executable file as “URLParser.cs1.”, which was created on the cs1 server. To run the program use “./URLParser.cs1”

Upon running the program, the user is prompted to enter a URL as the following: “Enter a URL” and the parse(string) function is called with the input string as the argument. The function is detailed below.

Upon entering a URL and hitting ENTER on the keyboard, the URL is parsed for the protocol, domain, port number, file path and parameters. In order to parse the input URL string, the program checks for key characters that indicate when particular components of the URL begin and end. Upon parsing the URL component, the input string is modified such that the parsed component is removed. This eliminates the need to keep track of beginning index of the next component, it will simply be index 1 of the new input string – index 1 is to account for the indicating character not being parsed. Each time the input URL is edited, the size is checked to ensure the input is not empty. If it is empty, the program stops.

The beginning and ending of URL components are as such:

* Protocol: begins at index 0 of input string, ends at the first index of ‘:’ in the input string.
* Domain: begins at index 0 – in this case the removal of Protocol removes the “://”, hence index 0 instead of 1 – and ends at the next index of ‘:’ if the port number is present. Otherwise, ends at the next index of ‘/’
* Port Number: If the port number is present, it begins at index 1 and ends at the index of the next ‘/’
* File Path: begins at index 1, and ends at the index of the next ‘?’ if parameters are present. Otherwise, end at the end of the input string
* Parameters: if the parameters are present, they begin at index 1 and end at the end of the input string.

Upon parsing by the above outline, the URL components are validated. Two maps (validItems, invalidItems) are declared to hold components which are valid and invalid. This was to easily separate the components which were valid and invalid when displaying the output of the program.

The URL components are validated using the following requirements:

* Protocol: Must be either “http” , “https” , “ftp” or “stps”. This validation is performed by using the compare() function of the string class to check if the parsed protocol is equal to any of the values above. If it is valid, validItems[“Protocol”] is assigned with the parsed protocol, otherwise invalidItems[“Protocol”] is assigned with the parsed protocol.
* Domain: Must be of the form <x>.<y>.<z>. This validation is performed by using the count() function of the algorithm library of c++ to count the number of ‘.’ characters in the parsed Domain. If it is valid, validItems[“Domain”] is assigned with the parsed domain, otherwise invalidItems[“Domain”] is assigned with the parsed domain.
* Port Number: If this component exists, the numerical value must be between 1 and 65535. This validation is performed by using the stoi() function of the string class to check if the integer value of the parsed port number is between 1 and 65535. If it is valid, validItems[“Port”] is assigned with the parsed port as a string, otherwise invalidItems[“Port”] is assigned with the parsed port as a string
* File Path: Must end with “.html” or “.htm”. This validation is performed by using the find() function of the string class and checking if the index of the above values are not -1. If the path is valid, validItems[“File Path”] is assigned with the parsed file path, otherwise, invalidItems[“File Path”] is assigned with the parsed file path.

After the URL components are validated and the maps mentioned above are accordingly populated, the program will perform one of the following functions: displayValid(map<string, string>) or displayInvalid(map<string, string>). The displayValid function is only called if the invalidItems map is empty, otherwise the diaplayInvalid function is called.

Function descriptions for display functions:

* displayValid: Receives a map<string, string> containing the parsed URL components. Displays the parsed URL components in the following order: Protocol, Domain, Port, File Path, Parameters and does so line by line. Displays the components as the following: <Component Name>”: ” <Parsed component>
* displayInvalid: Receives a map<string, string> containing the invalid parsed URL components. Checks if each component is populated. For those which are populated, edit the parsed component to be “<Parsed Component> is not a valid <Component Name>.” If the invalid component is Port, edit the parsed component to be “port number must be between 1 and 65535.” If the invalid component is File Path, append “ (needs to be .html or .htm).” to the end of the edited parsed component.

Sample Inputs/Outputs:

1. **Input**: https://cs.utdallas.edu:80/directory/faculty/people.html?name=Robert

**Output**: Protocol: https

Domain: cs.utdallas.edu

Port: 80

File path: /directory/faculty/people.html

Parameters: name=Robert

1. **Input**: httpz://cs.utdallas.edu:80000/people.html?name=Robert

**Output**: Invalid URL with following erroneous components:

Protocol: hpptz is not a valid protocol.

Port: port number must be between 1 and 65535

1. **Input:** htt://cs.uts/people.h

**Output:** Invalid URL with the following erroneous components:

Protocol: htt is not a valid protocol.

Domain: cs.uts is not a valid domain

File Path: people.h is not a valid file path (needs to be .html or .htm)