Pessum Documentation

Arden Rasmussen

24/02/2017

Contents

1 Functions **1**
1.1 General . . . 2
1.1.1 InitializePessum . . . 3
1.1.2 TerminatePessum . . . 4
1.2 Logging . . . 5
1.2.1 AddLogLocation . . . 6
1.2.2 GetLocation . . . 7
1.2.3 GetType . . . 8
1.2.4 InitializeLogging . . . 9
1.2.5 Log . . . 10
1.2.6 LogTimeStamp . . . 11
1.2.7 RemoveCaps . . . 12
1.2.8 TerminateLogging . . . 13
1.3 LuxReader . . . 14
1.3.1 FindType . . . 15
1.3.2 LoadLuxFile . . . 16
1.3.3 SaveLuxFile . . . 17

1 Functions

1.1 General

1.1.1 InitializePessum

```
1  void pessum::InitializePessum(bool devmode, bool logtimes);
```

devmode Sets the log to only output for Fatal, Error, and Warnings.

logtimes Sets the log to include the current time in hh:mm:ss to every log entry.

Initializes Pessum and any sub-components that need initialization. This function will open the log file. **Note:** This funciton must called before any other Pessum functions.

```
1  //Initialize pessum in development mode, and with logging of times
2  pessum::InitializePessum(true, true);
```

1.1.2 TerminatePessum

```
1  void pessum::TerminatePessum();
```

Terminates Pessum and any sub-components that need termination. This closes the log file and saves it. **Note:** This funciton should be the last of pessum functions to be called.

```
1  //Terminate pessum and all sub namespaces
2  pessum::TerminatePessum();
```

1.2 Logging

1.2.1 AddLogLocation

```
1  int pessum::logging::AddLogLocation(std::string locationstring);
```

locationstring Name of location that is to be saved for logging.

This function adds a file location into memory for later use in logging, permiting just an abbreviation or location index of the file path to be used in the log call. The abbreviation is made of the first three letters of each folder/file seperated by a '/'.
Returns: A integer value that point to the stored log location.

```
1  //Save a file of "folder/File.cpp" to the logging location
2  int locationindex = pessum::logging::AddLogLocation("folder/File.cpp");
3  //locationindex can be used in log calls to reference this locaiton, or the
       abbreviation "fol/Fil" can be used
```

1.2.2 GetLocation

```
std::string pessum::logging::GetLocation(int index);
std::stirng pessum::logging::GetLocation(std::string str);
```

index Index that is assosiated with the desired location.

str String that is either an abbreviation, or is the desired location.

Searches through the saved log locations, either by index value, or by abbreviation.
Returns: The string saved for specified location, and empty string if the specified location does not exist.

```
//A location must be added first
int locationindex = pessum::logging::AddLogLocation("folder/subfolder/file");
std::string location = pessum::logging::GetLocation(locationindex);
location = pessum::logging::GetLocation("fol/sub/fil");
//Both GetLocation calls will return "folder/subfolder/file"
```

1.2.3 GetType

```
std::string pessum::logging::GetType(std::string str);
```

str Log type string.

Takes a log type string, and converts to a standardized log type. Also expands abbreviations of default log types such as F to FATAL, or E to ERROR, ect.
Returns: A string representation of the log type.

```
pessum::logging::GetType("F")
//Will return "FATAL"
```

1.2.4 InitializeLogging

```
1  void pessum::logging::InitializeLogging(std::string outputfile, bool
     recordtime, bool dev);
```

outputfile File name to save output log to.

recordtime Determines if the time in hh:mm:ss is recored with each log entry.

dev Determines if the log is to run in development mode.

This funciton creates a logfile of the name *outputfile*, which will be used for all logging output. If *recordtime* is *true* the time is included in the log entry. If *recordtime* is *false* then the time is not included. If *dev* is *true* then all log entries are recored to the file. If *dev* is *false* then only FATAL, ERROR, and WARNING log entries are recored to the file. **Note:** This function must be called before any other logging funcitons to work.

```
1  //Create a log file with the name "log_output.log", that will include times
       of each entry, and runs in development mode
2  pessum::logging::InitializeLogging("log_output.log", true, true);
```

1.2.5 Log

1.

1.2.6 LogTimeStamp

```
1 void pessum::logging::LogTimeStamp(bool date);
```

date Determines if the data is included in the time stamp.

Logs a special time log, which includes the date and time in the format of www mmm dd hh:mm:ss yyyy. Where w is the day of the week, m is the month, d is the day, h is the hour, m is the minute, s is the second, and y is the year. If *date* is true, the the date is included, if *date* is false, then the data is removed from the log.

```
1 //Log the current time including the date
2 pessum::logging::LogTimeStamp(true);
3 //example result will be a log containing "Mon Jan 01 01:12:15 2017"
```

1.2.7 RemoveCaps

```
std::string pessum::logging::RemoveCaps(std::string str);
```

str String to be converted.

Converts all capital letters from *str* into lower case letters.
Returns: *str* will all capital letters converted to lowercase.

```
pessum::logging::RemoveCaps("Hello World")
//Will return "hello world"
```

1.2.8 TerminateLogging

```
1 void pessum::logging::TerminateLogging();
```

This funcition closes and saves the log file. **Note:** After this funciton is called no other logging funciton will work until InitializeLogging is called again.

```
1 //Terminate logging, and save the log file
2 pessum::logging::TerminateLogging();
```

1.3 LuxReader

1.3.1 FindType

```
int pessum::luxreader::FindType(std::string str);
```

str String to determine the type of.

This funciton reads and determines the storage type of the provided string.
Returns: -1 on unknown type, 0 on boolian, 1 on integer, 2 on double, 3 on string, 4 on a vector of boolians, 5 on a vector of integers, 6 on a vector of doubles, and 7 on a vector of strings.

```
//Determine the type of the string "{false, hello, 2.461, 11}"
FindType("{false, hello, 2.461, 11}");
//Will return 7 for a vector of strings
```

1.3.2 LoadLuxFile

```
std::vector<pessum::luxreader::Item> pessum::luxreader::LoadLuxFile(std::
  string filepath);
```

filepath File to load data from.

Loads file provided in *filepath* and reads data that can be either boolians, integers, doubles, strings, or vectors of each of these types.
Returns: A vector of Item that contains the data from the specified file, each value in its respective storage type.

```
//Read the file "foo.lux" and get the data values
pessum::luxreader::LoadLuxFile("foo.lux");
//Will return a vector with four elements, the first being a double, the
second a vector of boolians, the third beging a vector of strings, and the
 fourth begin a string.
```

Listing 1: code

```
10.04
{true, true, false}
{foo, bar, foobar}
This is a string.
```

Listing 2: "foo.lux"

1.3.3 SaveLuxFile

```
1  void pessum::luxreader::SaveLuxFile(std::string filepath, std::vector<
     pessum::luxreader::Item> contents);
```

filepath File to save data to.

contents Data to save to file.