

“UNIVERSIDAD POLITÉCNICA SALESIANA”

**CARRERA DE COMPUTACIÓN PROGRAMACIÓN ORIENTADA A OBJETOS CONSULTA N.-2**

* Consultar, estudiar y probar el manejo de archivos de texto (QTextStream) y archivos binarios (QDataStream).

**Referencias:**

"Aprenda Qt hoy mismo"

Ayuda de Qt 5.X.

**Nombre:** Pedro Vinicio Alajo Huilcamaigua **Fecha:**12/06/22

**Introducción**

Básicamente esta consulta contendrá varios aspectos del manejo de archivos de texto de igual manera del manejo de archivos binarios, todos estos flujos de datos básicamente nos van a ayudar para que el programa compile de mejor manera, podríamos decirlo como una librería inclusiva no objetivamente igual pero similar, Esta clase conocida como QTextStream para los llamados flujo de textos y QDataStream para el flujo de datos binarios, básicamente con las clases mencionadas con anterioridad vamos a utilizar solamente los constructores, dándonos a indicar los factores para ejecutarlos en el dispositivo de entrada y de igual manera el de salida.

**Ejemplo de lectura de fichero referente al tema ya antes mencionado**

|  |
| --- |
| QFile **f**("fichero.txt");  QTextStream **in**(&f);  **if**(!f.open(QIODevice::ReadOnly)  qFatal("No puedo abrir el fichero para lectura.");  **while**(!in.atEnd()){  qDebug("%s",qPrintable(in.readLine()));  }  f.close(); |

Podemos ver que la clase QTextStream no ayudara a proporcionar una rápida y cómoda interfaz para leer y escribir texto.

QTextStream puede ejecutarse en QIODevice, QByteArray o QString. Con los operadores de transmisión de QTextStream, puede leer y componer fácilmente palabras, líneas y números. Para la salida importada, QTextStream admite capacidades de formato para el relleno y la alineación de campos, así como el formato de números.

**Ejemplo**

|  |
| --- |
| QFile **data**("output.txt");  **if** (data.open(QFile::WriteOnly | QFile::Truncate)) {  QTextStream out(&data);  out << "Result: " << qSetFieldWidth(**10**) << left << **3.14** << **2.7**;  // writes "Result: 3.14 2.7 "  } |

Podemos ver que de igual manera podremos utilizar QTextStream para poder leer o comprender la entrada hacia la consola y de esta manera poder reescribirla para su salida de esta, QTextStream tiene conciencia en su localización, la misma que se auto modificará simultáneamente utilizando el código correcto.

**Ejemplo**

|  |
| --- |
| QTextStream **stream**(stdin);  QString line;  **while** (stream.readLineInto(&line)) {  ...  } |

Además de usar los constructores de QTextStream, también puede configurar el dispositivo o la cadena en la que se ejecuta QTextStream llamando a setDevice() o setString(). Puede buscar un centro llamando a search() y atEnd() devolverá verdadero cuando no haya más datos para leer. Si llama a flush(), QTextStream vacía todos los datos de su búfer de escritura al dispositivo y llama a flush() en el dispositivo.

Internamente, QTextStream usa un búfer basado en Unicode, y QTextStream usa QTextCodec para admitir automáticamente diferentes juegos de caracteres. De forma predeterminada, QTextCodec::codecForLocale() se usa para leer y escribir, pero también puede configurar el códec llamando a setCodec(). También se admite la detección automática de Unicode.

Cuando esta función está habilitada (el comportamiento predeterminado), QTextStream detectará una BOM (marca de orden de bytes) UTF-16 o UTF-32 y cambiará al códec UTF apropiado cuando se ejecute. QTextStream no escribe una lista de materiales de forma predeterminada, pero puede habilitarla llamando a setGenerateByteOrderMark (verdadero).

Cuando QTextStream se ejecuta en QString directamente, el códec está deshabilitado.  
Hay tres formas generales de usar QTextStream al leer archivos de texto:

Uno por uno, llamando a readLine() o readAll(). palabra por palabra. QTextStream admite la transmisión a tiendas QStrings, QByteArrays y char\*.

Las palabras separadas por espacios y guiones se ignoran automáticamente. Carácter por letra, convertir a QChar u otros tipos de caracteres. Este método se usa a menudo para un procesamiento de entrada conveniente al analizar archivos, independientemente de la codificación de caracteres y la semántica de fin de línea.

Para ignorar los espacios en blanco, llame a ignoreWhiteSpace(). Dado que la secuencia de texto usa un búfer, no es necesario que lea la secuencia implementando la superclase. Por ejemplo, si tiene un archivo QFile y lo lee directamente usando QFile::readLine() en lugar de usar una secuencia, la posición interna de la secuencia de texto será asíncrona con la posición del archivo.

De forma predeterminada, al leer números de un flujo de texto, QTextStream detectará automáticamente la representación base del número. Por ejemplo, si un número comienza con "0x", se supone que está en hexadecimal. Si comienza con los números del 1 al 9, se dice que es un sistema decimal y así sucesivamente. Puede establecer la regla en un número entero, deshabilitando así la detección automática, llamando a setIntegerBase().

**Ejemplo**

|  |
| --- |
| QTextStream **in**("0x50 0x20");  **int** firstNumber, secondNumber;  in >> firstNumber; // firstNumber == 80  in >> dec >> secondNumber; // secondNumber == 0  **char** ch;  in >> ch; // ch == 'x' |

De igual manera usaremos QDataStream para enviar datos desde el servidor también  
Recójalo de los clientes. El flujo de datos es muy simple, solo tiene la posición 16 como dirección, la longitud del mensaje seguida de una cadena que contiene la frase.

En este caso al leer datos

En todo este caso al leer los datos Si los datos se leen o escriben, debemos decirle al compilador Qt Creator con lo que usaremos la versión la cual nosotros podamos  
Usar la misma, porque tal vez en diferentes versiones la implementación de estas clases contenga diferentes flujos de datos, a continuación, veremos un ejemplo claro de como seria:

|  |
| --- |
| QFile **f**("fichero.bin");  QDataStream **out**(&f);  out.setVersion(QDataStream::Qt\_4\_6);  **if**(!f.open(QIODevice::WriteOnly)  qFatal("No puedo abrir el fichero para escritura");  out << **0xFFFC**;  f.close(); |

**Conclusión**

Pude entender que todas estas nuevas clases en si nos van a servir para poder compilar los futuros proyectos que tendremos ya que estas clases nos permitirán adentrarnos a nuevas clases y de igual manera a nuevos ficheros los cuales nos permitirán ejecutar el programa de manera óptima, de igual manera con estas nuevas clases tendremos muchos errores ya que si no utilizamos los respectivos pasos a seguir o no respetamos la estructura de estas no podremos ejecutar de manera óptima el programa.

# Bibliografía

Anónimo. (2010, Octubre 10). Retrieved from https://avac.ups.edu.ec/presencial60/pluginfile.php/341206/mod\_resource/content/0/Aprenda\_%20Qt\_Hoy\_Mismo.pdf

Company, Q.-T. Q. (2021, Enero 05). *Qt-The Qt Company*. Retrieved from https://doc.qt.io/archives/qt-5.12/qtextstream.html