CSCE689: Project: Serialization Library in C++11

October 22, 2012

Raghdah Al-Shaikhli	raghdah@tamu.edu	UIN: 322000076
Shreyas Vinayakumar	vshreyas@tamu.edu	UIN: 521007089
Shashwat Lal Das	sldas@tamu.edu	UIN: 321005607

1. **Objective**

   We plan to develop a library in C++ which enables serialization and deserialization of data in a portable and easy to use fashion.

2. **Need for such a library**

   Writing data generated or manipulated by a program into a stream (serialization) is a common need experienced across various domains, such as machine learning (storing data sets modified or generated by the program), scientific programming, etc.

   Common techniques to solve this problem include writing data into XML files, or developing a unique way to perform serialization for every problem. In general, there does not appear to be a uniform way of performing this in a portable and efficient manner which works for most problems. Writing custom code for doing this is cumbersome and error prone.

3. **Requirements**

   As mentioned, portability and simplicity are fundamental requirements.

   Usually, the user needs to store not only plain old data types like integers, but structured data as well. Structured data may take the form of classes, structs, etc. with a specific layout, and the user should be able to serialize such data equally easily. To this end, the user defines the structure, and serialization should preserve this structure while storing the data. Deserialization should reconstruct this structure directly.

4. **Features Planned**

   Be able to store data of most common data types (int, string, etc.) in a compact form. The method of encoding may differ based on the data type.

   The user should be able to choose between human readable or binary representation. Naturally, human readable format would imply less emphasis on compactness.

   Include support for aggregates of data in the form of arrays and vectors (compulsorily). Whether support is added for other less common list types which may be more complex to store is not yet finalized.