УДК 004.424
Г.Г. Сергеев, доцент, канд. техн. наук
Севастопольский национальный технический университет,
ул. Университетская 33, Севастополь, Украина,99053
E-mail: root@sevgtu.sebastopol.ua

СЕРИАЛИЗАЦИЯ ПРОГРАММНЫХ ОБЪЕКТОВ С УЧЕТОМ МОДЕРНИЗАЦИИ ИХ СТРУКТУРЫ

Рассматривается задача сериализации объектов программы с учетом возможности изменения структуры класса. Предлагаются методы для сохранения/восстановления информации динамических массивов объектов.

Объектно-ориентированное программирование вряд ли получило бы столь широкое признание, если бы программист не мог сохранить объект в текущем состоянии и восстановить его позднее. Запись объекта в поток данных называется сериализацией (serialization), а обратный процесс называется десериализацией (deserialization) [1]. Процесс сохранения должен поддерживать рекурсивное сохранение внутренних объектов, предоставлять инструментальные средства для клонирования сложных объектов, а также, при необходимости, выполнять компрессию и шифрование информации. В работах [1 — 3] рассматриваются различные подходы к проблеме сериализации, однако ни в одном из них не отражена проблема контроля версии файла данных. Цель статьи — рассмотрение нового разработанного метода сериализации и десериализации объектов.

Пусть $V=\{ v_1, v_2 \ldots v_n \}$ множество объектов программы. Для каждого объекта $v_i \in V$ определено множество полей $f_i,=\{ f_i^1, f_i^2, \ldots, f_i^n \}$ подлежащих сериализации в поток данных. Назовем версией объекта v_i^* объект, который соответствует семантике объекта v_i, но $f_i \neq f_i^*$. Версия v_i^* может быть образована путем прямого исправления кода или в результате наследования объектом v_i^* данных и методов v_i. При использовании механизмов простой сериализации [1] возникает проблема несоответствия версий файла данных семантике объектов. Она решается с помощью конверторов, что затрудняет работу пользователя и снижает коэффициент готовности системы, либо путем прямого распознавания номера версии на уровне методов загрузки/выгрузки. В этом случае формат записи данных при сериализации объектов программы имеет вид, показанный на рисунке 1. Программист обязан на уровне исходного кода контролировать корректность номера версии, что не исключает ошибок. Ошибка интерпретации данных потока на этапе десериализации объекта v_i приведет к неминуемым ошибкам десериализации всех последующих объектов.

№ версии v_i	f_i^1	f_i^2	…	f_i^n	№ версии v_{i+1}	$f_{i+1}{}^1$	$f_{i+1}{}^2$	…

Рисунок 1 — Формат данных потока при простой сериализации с указанием номером версии объекта

Таким образом, важной и актуальной задачей является разработка универсального инструмента, который бы позволил избежать неправильной интерпретации потока данных механизмом десериализации без использования дополнительных программных продуктов. При этом будем полагать следующее:

— сериализация и десериализация выполняются методами классов вида SaveToStream (Stream : TStream) и LoadFromStream (Stream : TStream) соответственно, где Stream — указатель на бинарный поток;

— при создании объекта v_i во все элементы f_i заносится информация, необходимая для корректного функционирования программного кода.

Проблемам сериализации данных в различных системах программирования посвящен ряд работ [2,3].

Возьмем за основу структуры бинарного потока решение, типичное для большинства систем архивации.

Как видно из рисунка 2 при сериализации объектов программы учетная информация о каждом из них представлена в каталоге соответствующего уровня. Поскольку количество объектов заранее неизвестно, предлагается в начале потока размещать два элемента: указатель на корневой каталог и количество его элементов. Данные объектов сериализуются так же, как и в случае простой сериализации, а при завершении процесса в конец потока записывается каталог.

Описатель поля f_i — это четверка вида $\{N_i, P_i, L_i, V_i\}$, где N_i — уникальный идентификатор поля объекта (на уровне класса), P_i — его позиция в каталоге, L_i — размер и V_i — индекс или версия объекта.

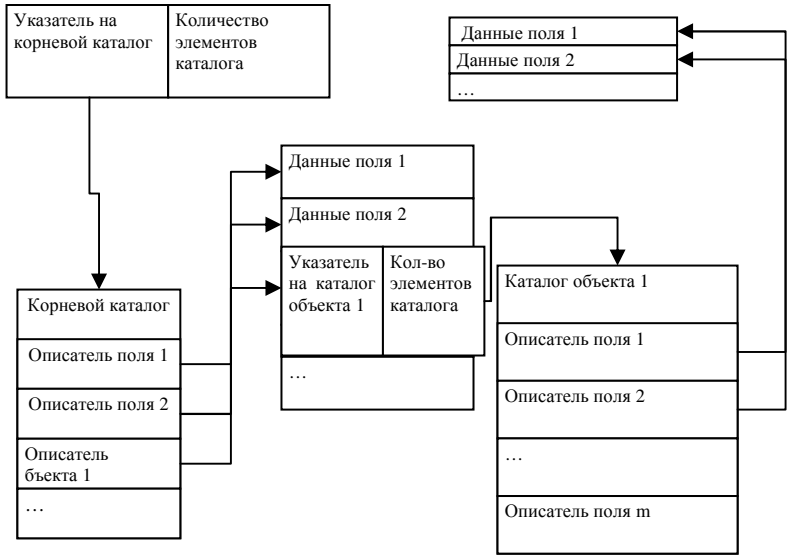

Рисунок 2 — Формат потока сериализации с каталогом

Процесс десериализации в этом случае начинается с чтения каталога объектов. Затем в цикле для каждого элемента каталога производится перенос данных из потока в соответствующие переменные. Наличие поля L_i позволяет выполнять восстановление без учета типа данных поля.

Для обеспечения процессов сериализации и десериализации средствами среды визуального программирования Delphi был разработан класс TCatalogObject, включающий в себя следующие примитивы:

— Create(AStream, AMode) — выполняет «включение» объекта в поток сериализации AStream. Здесь AMode может принимать значения cmWrite и cmRead для сериализации и десериализации соответственно;

— WriteField(f,N,L,V) — запись в поток простого поля f;

— ReadField (f, i) — чтение поля f, соответствующего в катлоге позиции i;

— ReadCatalog — чтение каталога.

Для обеспечения сериализации вложенных объектов используются примитивы:

— WriteByProc(writeproc, N,V) — запись вложенного объекта в поток, где writeproc - указатель на метод сериализации объекта;

— ReadByProc(readproc,i) — чтение вложенного объекта из потока, где readproc — указатель на метод десериализации вложенного объекта.

— Destroy — завершает сериализацию (десереализацию), записывает в бинарный поток каталог и устанавливает указатель в конец потока.

Свойства CatalogSize и CatalogItem[Index] возвращают размер и элемент каталога в формате {N, P,L,V}.

Процедура сериализации данных объекта имеет вид:

```
procedure Имя объекта.SaveToFile (F : TStream);
var i : integer;
  c : TCatalogObject;
begin
  c:=TCatalogObject.Create(F,cmWrite);
c.WriteField(Поле_объекта,sizeof(Поле_объекта_1),Идентефикатор_поля_1,Версия);
…
//Для полей вложенных объктов
  c.WriteByProc(Вложенный_Объект.SaveToFile, Идентефикатор_поля_n,Версия);
…
  c.Free;
end;
```

Процедура десериализации данных объекта:

```
procedure TBus.LoadFromFile(F : TStream);
var i : integer;
  c : TCatalogObject; r : TCatalogRec;
begin
  c:=TCatalogObject.Create(F,cmRead); c.ReadCatalog;
  for i:=0 to c.CatalogSize-1 do
  begin
  r:=c.CatalogItem[i];
```

```
      case r.RecName of
      Идентификатор_поля_1 : c.ReadField(FName,i);
      …
      Идентификатор_поля_n : c.ReadByProc(Вложенный_Объект.LoadFromFile,i);
    …
      end;
    end;
  end;
```

Предложенный метод уступает простой сериализации по объему данных, но превосходит его по следующим метрикам качества программных средств: адаптируемости, надежности и способности к взаимодействию [4].

В дальнейшем планируется развить предложенный метод сериализации с одновременным архивированием и кодированием данных.

Библиографический список

1. Эккель Б. Философия Java / Б. Эккель. — СПб.: Питер, 2001. — 880 с.

2. Чистяков В. Сериализация в .NET /В. Чистяков // RSDN Magazine. — 2003 — №3 — с. 15 — 18.

3. Пономарёв В. Сериализация объектов стандартными средствами Delphi / В.Пономарёв // RSDN Magazine RSDN Magazine — 2003. — № 6 — с. 45 — 49.

4. Куликов Ю.А. Основы менеджмента качества / Куликов Ю., Хачатуров А. — М.: ДиС, 2003. — 304 с.

Поступила в редакцию 26.03.2008 г.