



Тема 11.

Сериализация

© 2014, Serge Kashkevich

Что такое сериализация и десериализация?

В C# есть возможность сохранять на внешних носителях не только данные примитивных типов, но и объекты. Сохранение объектов называется *сериализацией*, а восстановление сохраненных объектов — *десериализацией*.

При сериализации объект преобразуется в линейную последовательность байтов. Это сложный процесс, поскольку объект может включать множество унаследованных полей и ссылки на вложенные объекты, которые, в свою очередь, тоже могут состоять из объектов сложной структуры.

Примеры сериализации

1. Пусть мы работаем с созданным ранее классом «бинарное поисковое дерево». Нам необходимо время от времени выполнять сохранение обрабатываемой информации (backup) и, при необходимости, восстанавливать сохранённую информацию.
2. Сериализовать объект класса «Персонаж ролевой игры», одно из полей которого содержит коллекцию способностей. Способность, в свою очередь, содержит поля «название» и «уровень».

Ограничения на сериализацию

Сериализуемый класс, а также все его подклассы и классы, объекты которых входят в состав сериализуемого класса, должны быть помечены атрибутом `[Serializable]`:

```
[Serializable]
public class Tree <T> where T : IComparable {
    /// <summary>
    /// Класс "узел БПД"
    /// </summary>
    [Serializable]
    protected class Item {
...

```

Если отдельные поля (не свойства!) сериализовывать не нужно, их необходимо пометить атрибутом `[NonSerialized]`

Типы сериализации

Различают три типа сериализации:

- двоичный
- SOAP *(не рассматривается в рамках этого курса)*
- в виде XML-файла.

В первом случае следует подключить к программе пространство имен

System.Runtime.Serialization.Formatters.Binary,

Во третьем — пространство *System.Xml.Serialization*

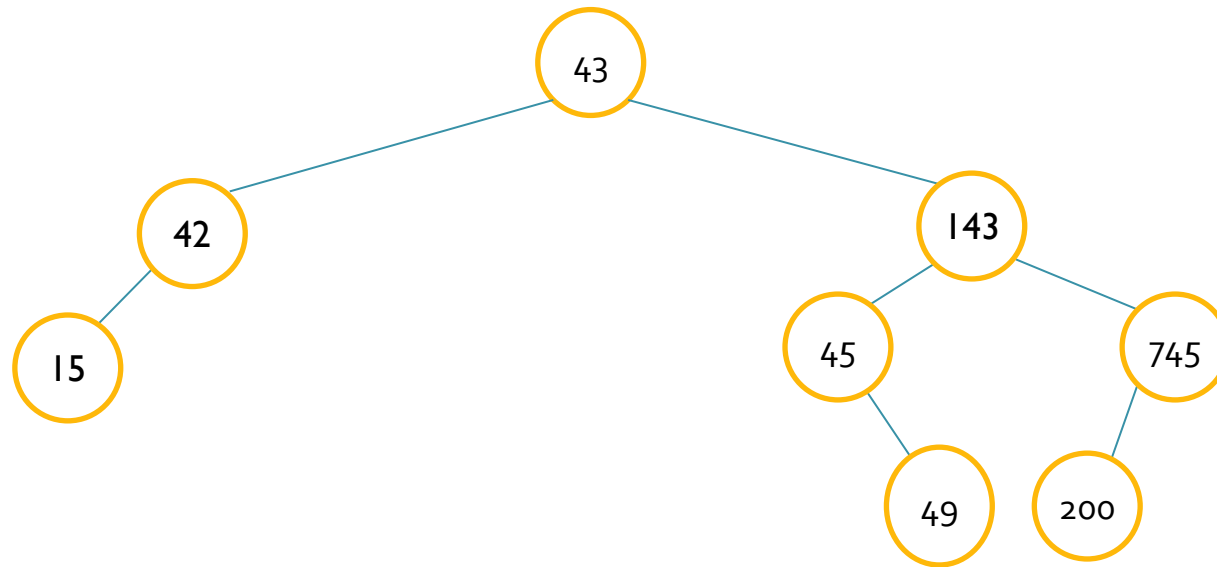
Двоичная сериализация

Для выполнения бинарной сериализации объекта, который может быть сериализован, необходимо вызвать метод `Serialize` объекта класса `BinaryFormatter`:

```
Tree <int> tr = new Tree <int> ();  
using(FileStream fs = new FileStream ("BST.dat",  
                                     FileMode.Create)) {  
    BinaryFormatter bf = new BinaryFormatter();  
    bf.Serialize(fs, tr);  
}
```

Результат двоичной сериализации

Пусть БПД имеет вид



Результат двоичной сериализации (продолжение)

```
view BST.dat - Far 3.0.3367 x86 Administrator
D:\cs\BST_serialize\bin\Debug\BST.dat
0000000120: 20 43 75 6C 74 75 72 65 3D 6E 65 75 74 72 61 6C Culture=neutral
0000000130: 2C 20 50 75 62 6C 69 63 4B 65 79 54 6F 6B 65 6E , PublicKeyToken
0000000140: 3D 62 37 37 61 35 63 35 36 31 39 33 34 65 30 38 =b77a5c561934e08
0000000150: 39 5D 5D 02 00 00 00 02 00 00 00 09 03 00 00 00 9]]@  @  @
0000000160: 05 03 00 00 00 76 42 53 54 5F 73 65 72 69 61 6C ♠♥ vBST_serial
0000000170: 69 7A 65 2E 54 72 65 65 60 31 2B 49 74 65 6D 5B ize.Tree l+Item[
0000000180: 5B 53 79 73 74 65 6D 2E 49 6E 74 33 32 2C 20 6D [System.Int32, m
0000000190: 73 63 6F 72 6C 69 62 2C 20 56 65 72 73 69 6F 6E mscorlib, Version
00000001A0: 3D 34 2E 30 2E 30 2E 30 2C 20 43 75 6C 74 75 72 =4.0.0.0, Cultu
00000001B0: 65 3D 6E 65 75 74 72 61 6E 6C 2C 20 50 75 62 6C 69 e=neutral, Publi
00000001C0: 63 4B 65 79 54 6F 6B 65 6E 3D 62 37 37 61 35 63 cKeyToken=b77a5c
00000001D0: 35 36 31 39 33 34 65 30 38 39 5D 5D 04 00 00 00 561934e089]]♦
00000001E0: 04 69 6E 66 6F 04 6C 53 6F 6E 04 72 53 6F 6E 06 ♦info♦lson♦rson♦
00000001F0: 66 61 74 68 65 72 00 04 04 04 08 76 42 53 54 5F father ♦♦♦vBST_
0000000200: 73 65 72 69 61 6C 69 7A 65 2E 54 72 65 65 60 31 serialize.Tree l
0000000210: 2B 49 74 65 6D 5B 5B 53 79 73 74 65 6D 2E 49 6E +Item[[System.In
0000000220: 74 33 32 2C 20 6D 73 63 6F 72 6C 69 62 2C 20 56 t32, mscorlib, V
0000000230: 65 72 73 69 6F 6E 3D 34 2E 30 2E 30 2E 30 2C 20 ersion=4.0.0.0,
0000000240: 43 75 6C 74 75 72 65 3D 6E 65 75 74 72 61 6C 2C Culture=neutral,
0000000250: 20 50 75 62 6C 69 63 4B 65 79 54 6F 6B 65 6E 3D PublicKeyToken=
0000000260: 62 37 37 61 35 63 35 36 31 39 33 34 65 30 38 39 b77a5c561934e089
0000000270: 5D 5D 02 00 00 00 76 42 53 54 5F 73 65 72 69 61 ]]@ vBST_seria
0000000280: 6C 69 7A 65 2E 54 72 65 65 60 31 2B 49 74 65 6D lize.Tree l+Item
0000000290: 5B 5B 53 79 73 74 65 6D 2E 49 6E 74 33 32 2C 20 [[System.Int32,
00000002A0: 6D 73 63 6F 72 6C 69 62 2C 20 56 65 72 73 69 6F mscorlib, Versio
00000002B0: 6E 3D 34 2E 30 2E 30 2E 30 2C 20 43 75 6C 74 75 n=4.0.0.0, Cultu
00000002C0: 72 65 3D 6E 65 75 74 72 61 6C 2C 20 50 75 62 6C re=neutral, Publ
00000002D0: 69 63 4B 65 79 54 6F 6B 65 6E 3D 62 37 37 61 35 icKeyToken=b77a5
00000002E0: 63 35 36 31 39 33 34 65 30 38 39 5D 5D 02 00 00 c561934e089]]@
00000002F0: 00 76 42 53 54 5F 73 65 72 69 61 6C 69 7A 65 2E vBST_serialize.
0000000300: 54 72 65 65 60 31 2B 49 74 65 6D 5B 5B 53 79 73 Tree l+Item[[Sys
0000000310: 74 65 6D 2E 49 6E 74 33 32 2C 20 6D 73 63 6F 72 tem.Int32, mscor
0000000320: 6C 69 62 2C 20 56 65 72 73 69 6F 6E 3D 34 2E 30 lib, Version=4.0
0000000330: 2E 30 2E 30 2C 20 43 75 6C 74 75 72 65 3D 6E 65 .0.0, Culture=ne
0000000340: 75 74 72 61 6C 2C 20 50 75 62 6C 69 63 4B 65 79 utral, PublicKey
0000000350: 54 6F 6B 65 6E 3D 62 37 73 61 35 63 35 36 31 39 Token=b77a5c5619
0000000360: 33 34 65 30 38 39 5D 5D 02 00 00 00 02 00 00 00 34e089]]@  @
0000000370: 2B 00 00 00 09 04 00 00 00 09 05 00 00 00 0A 01 +  @  @  @
0000000380: 04 00 00 00 03 00 00 00 00 00 00 00 00 00 00 00 ♦  @  @  *  @
0000000390: 00 0A 09 03 00 00 00 01 00 00 00 03 00 00 00 00  @  @  @  @
00000003A0: 8F 00 00 00 09 08 00 00 00 09 09 00 00 00 09 03  @  @  @  @
00000003B0: 00 00 00 01 06 00 00 00 03 00 00 00 0F 00 00 00  @  @  @  @
00000003C0: 0A 0A 09 04 00 00 00 01 08 00 00 00 03 00 00 00  @  @  @  @
00000003D0: 2D 00 00 00 0A 09 0C 00 00 00 09 05 00 00 00 01  @  @  @  @
00000003E0: 09 00 00 00 03 00 00 00 00 00 00 00 00 00 00 00  @  @  @  @
00000003F0: 00 0A 09 05 00 00 00 01 0C 00 00 00 03 00 00 00  @  @  @  @
0000000400: 31 00 00 00 0A 0A 09 08 00 00 00 01 0E 00 00 00  @  @  @  @
0000000410: 03 00 00 00 C8 00 00 00 0A 0A 09 09 00 00 00 0B  @  @  @  @
```


Двоичная десериализация

Для выполнения десериализации объекта, который может быть сериализован, необходимо вызвать метод **Deserialize** объекта класса **BinaryFormatter**:

```
Tree<int> tr_restored;  
using (FileStream fr = new FileStream("BST.dat",  
                                     FileMode.Open)) {  
    BinaryFormatter bf = new BinaryFormatter();  
    tr_restored = (Tree<int>)bf.Deserialize(fr);  
}
```

XML-сериализация

XML-сериализация позволяет сохранить информацию об объекте в XML-документе. XML-сериализация выполняется медленнее, накладывает дополнительные ограничения, но позволяет выполнить кроссплатформенный перенос данных.

Ограничения на XML-сериализацию:

- класс сериализуемого объекта должен иметь конструктор без параметров;
- для каждого свойства должны быть публичные сеттеры;
- в объекте не должно быть циклических ссылок (именно поэтому мы не можем сериализовать в формат XML объекты класса Tree)

Описание классов для XML-сериализации

```
[Serializable]
public class Ability {
    public string abilityName { get; set; }
    public uint abilityValue { get; set; }
    Ability() { }
    public Ability(string aName, uint aValue) {
        abilityName = aName;
        abilityValue = aValue;
    }
}
```

Описание классов для XML-сериализации (продолжение)

```
[Serializable]
public class Hero {
    private static uint nextID = 1;
    public uint ID { get; private set; }
    // сделать это свойство приватным нельзя!
    public string name { get; set; }
    uint age { get; set; }
    public List<Ability> abilities =
        new List<Ability>();
    Hero() { }
    public Hero(string aName, uint aAge) {
        ID = nextID++;
        name = aName;
        age = aAge;
    }
}
```

Описание классов для XML-сериализации (окончание)

```
public void addAbility(Ability ab) {
    abilities.Add(ab);
}

[NonSerialized]
private string nick;
// описать в виде свойства поле nick нельзя!
public void setNick(string aNick) {
    nick = aNick;
}
public string getNick() {
    return nick;
}
}
```

Выполнение XML-сериализации

Для выполнения XML-сериализации объекта, который может быть сериализован, необходимо вызвать метод `Serialize` объекта класса `XmlSerializer`. В конструкторе этого объекта передаётся тип сериализуемого объекта:

```
Hero h = new Hero("Serge", 57);  
h.setNick("kash");  
h.addAbility(new Ability("strench", 35));  
h.addAbility(new Ability("health", 10));  
h.addAbility(new Ability("mana", 128));  
h.addAbility(new Ability("sleight", 71));
```

```
using (StreamWriter fs = new StreamWriter("hero.xml")) {  
    XmlSerializer xs = new XmlSerializer(typeof(Hero));  
    xs.Serialize(fs, h);  
}  
Console.WriteLine(h.name);  
Console.WriteLine(h.getNick());
```

Результат XML-сериализации

```
edit hero.xml - Far 3.0.3367 x86 Administrator
D:\cs\BST_serialize\bin\Debug\hero.xml
<?xml version="1.0" encoding="utf-8"?>
<Hero xmlns:xsi="http://www.w3.org/2001/XMLSchema-ns
  <abilities>
    <Ability>
      <abilityName>strench</abilityName>
      <abilityValue>35</abilityValue>
    </Ability>
    <Ability>
      <abilityName>health</abilityName>
      <abilityValue>10</abilityValue>
    </Ability>
    <Ability>
      <abilityName>mana</abilityName>
      <abilityValue>128</abilityValue>
    </Ability>
    <Ability>
      <abilityName>sleight</abilityName>
      <abilityValue>71</abilityValue>
    </Ability>
  </abilities>
  <ID>1</ID>
  <name>Serge</name>
</Hero>
```

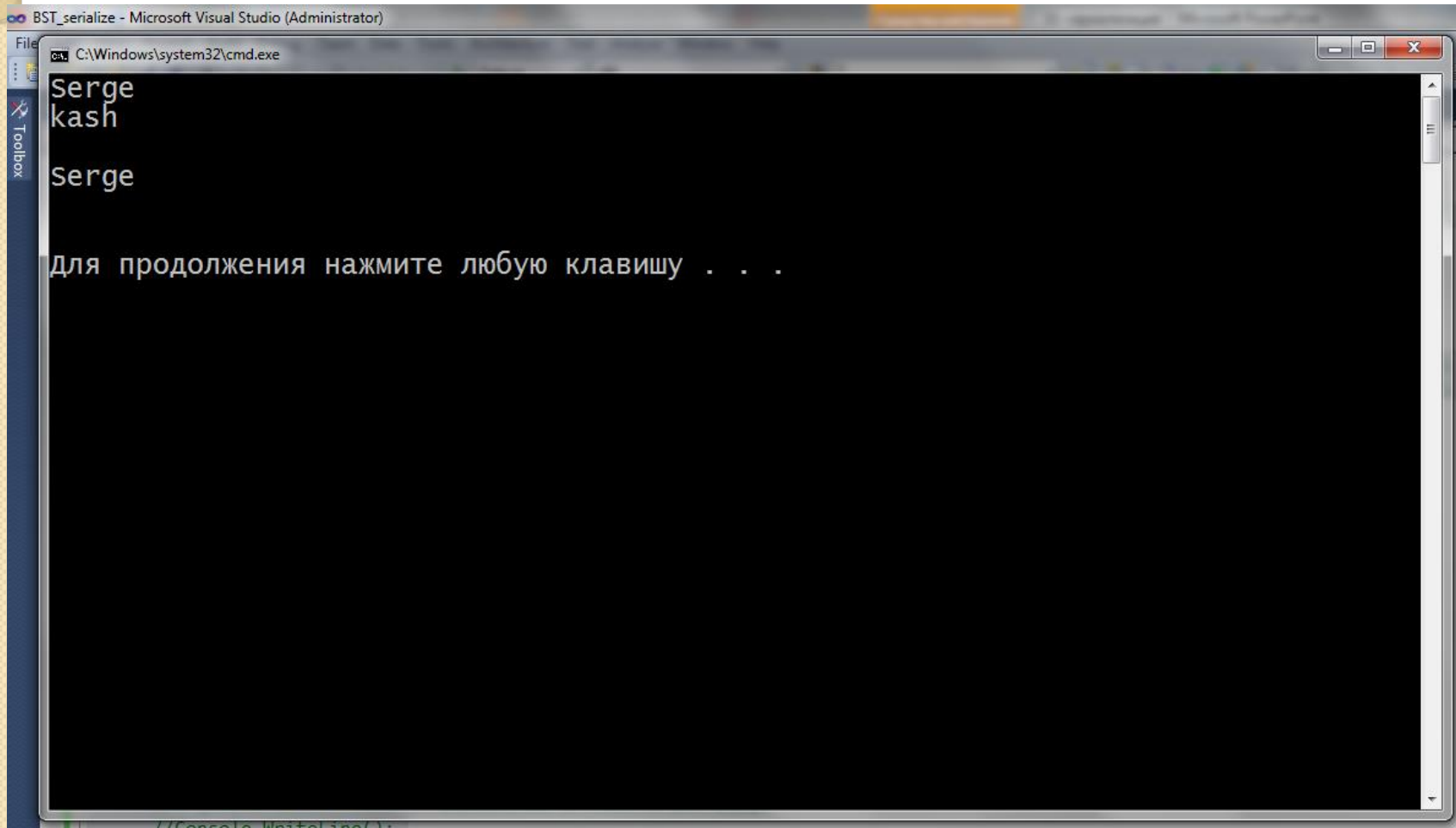
Выполнение XML-десериализации

Для выполнения XML-десериализации объекта, который может быть сериализован, необходимо вызвать метод **Deserialize** объекта класса **XmlSerializer**:

```
Hero h_restored;  
using (StreamReader sr = new StreamReader("hero.xml")) {  
    XmlSerializer xs = new XmlSerializer(typeof(Hero));  
    h_restored = (Hero)xs.Deserialize(sr);  
}  
Console.WriteLine();  
Console.WriteLine(h_restored.name);  
Console.WriteLine(h_restored.getNick());
```


Результат XML-десериализации

Обратите внимание, что поле `nick` не было восстановлено!



The screenshot shows a Visual Studio window titled "BST_serialize - Microsoft Visual Studio (Administrator)". Inside, a console window titled "C:\Windows\system32\cmd.exe" displays the output of an XML deserialization process. The output consists of three lines: "Serge", "kash", and "Serge". Below these lines, a message in Russian asks the user to press any key to continue: "для продолжения нажмите любую клавишу . . .". At the bottom of the console window, the text "//Console.WriteLine();" is visible, indicating the source of the output.

```
File Edit View Window Help  
C:\Windows\system32\cmd.exe  
Serge  
kash  
Serge  
для продолжения нажмите любую клавишу . . .  
//Console.WriteLine();
```

Управляемая сериализация

В рамках бинарной сериализации можно описать свой механизм сериализации/десериализации, учитывающий особенности работы с классом, выполнив *управляемую сериализацию*.

Недостатки предыдущего примера:

Каждый герой должен иметь свой уникальный ID, и пользователь не может изменять это свойство.

Однако десериализация в новом контексте может привести к тому, что ID нескольких героев могут совпасть (как это случилось для объектов h и h_restored).

Кроме того, требование иметь публичные сеттеры для всех свойств приводит к тому, что ID становится доступным для изменения пользователями.

Интерфейс ISerializable

Одним из способов выполнения управляемой сериализации является реализация сериализуемым объектом интерфейса **ISerializable**.

В состав этого интерфейса входит единственный метод

```
void GetObjectData(SerializationInfo info,  
                   StreamingContext context)
```

Класс **SerializationInfo** является контейнером для хранения сериализуемых данных. Метод **AddValue** служит для добавления данных в контейнер:

```
public void GetObjectData(SerializationInfo info,  
                          StreamingContext context) {  
    // поля ID и nick не сериализуются  
    info.AddValue("name", name);  
    info.AddValue("age", age);  
    info.AddValue("abilities", abilities,  
                  typeof(List <Ability>));  
    // для собственных типов и коллекций надо  
    // указывать третий параметр  
}
```

Интерфейс ISerializable (продолжение)

Для десериализации необходимо добавить конструктор с такими же параметрами:

```
protected Hero(SerializationInfo info,  
    StreamingContext context) {  
    ID = nextID++;  
    name = info.GetString("name");  
    age = info.GetUInt32("age");  
    abilities = (List<Ability>)info.GetValue("abilities",  
        typeof(List<Ability>));  
}
```

Изменения в описании класса Hero

```
[Serializable]
public class Hero: ISerializable
{
    private static uint nextID = 1;
    public uint ID {get; private set;}
    public string name { get; private set; }
    public uint age { get; set; }
    public List<Ability> abilities =
        new List<Ability>();
    public string nick { get; set; }
    //    Hero() { } это уже не нужно!
    public Hero(string aName, uint aAge) {
        ID = nextID++;
        name = aName;
        age = aAge;
    }
}
```

Изменения в описании класса Hero (окончание)

```
public void showHero() {  
    Console.WriteLine("ID: {0}; name: {1}; age: {2}",  
        ID, name, age);  
    Console.WriteLine("nickname: {0}", nick);  
    Console.WriteLine("==== Abilities: =====");  
    foreach (Ability ab in abilities)  
        Console.WriteLine("{0}: {1}", ab.abilityName,  
            ab.abilityValue);  
}
```

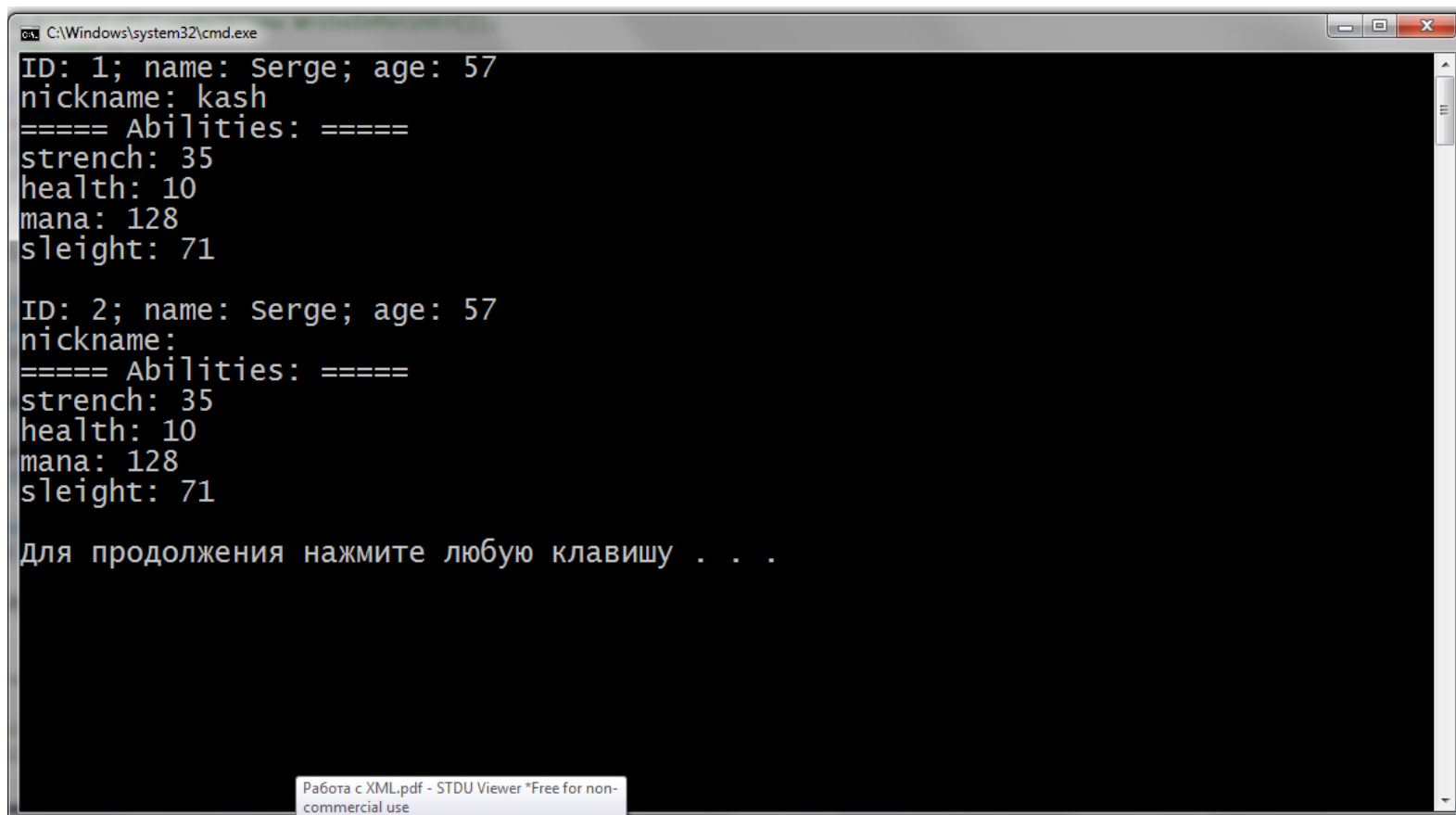
Изменения в головной программе

```
Hero h = new Hero("Serge", 57);
h.nick = "kash";
h.addAbility(new Ability("strench", 35));
h.addAbility(new Ability("health", 10));
h.addAbility(new Ability("mana", 128));
h.addAbility(new Ability("sleight", 71));
h.showHero();

using(FileStream fs =
    new FileStream("hero.dat", FileMode.Create)) {
    BinaryFormatter bf = new BinaryFormatter();
    bf.Serialize(fs, h);
}
Console.WriteLine();

Hero h_restored;
using(FileStream fr =
    new FileStream("hero.dat", FileMode.Open)) {
    BinaryFormatter bf = new BinaryFormatter();
    h_restored = (Hero)bf.Deserialize(fr);
}
h_restored.showHero();
```

Результаты работы программы



A screenshot of a Windows command prompt window titled "C:\Windows\system32\cmd.exe". The window has a black background with white text. The output shows two identical character profiles for "Serge". Each profile includes an ID, name, age, nickname, and a set of abilities (strench, health, mana, sleight). The text is as follows:

```
C:\Windows\system32\cmd.exe
ID: 1; name: Serge; age: 57
nickname: kash
===== Abilities: =====
strench: 35
health: 10
mana: 128
sleight: 71

ID: 2; name: Serge; age: 57
nickname:
===== Abilities: =====
strench: 35
health: 10
mana: 128
sleight: 71

Для продолжения нажмите любую клавишу . . .
```

At the bottom of the window, there is a small white box with the text: "Работа с XML.pdf - STDU Viewer *Free for non-commercial use".



*Примеры сериализации
находятся в файле `serialize.sip`*