Chương 8

Ghi/đọc dữ liệu của ứng dụng C# ra file

8.0 Dẫn nhập
8.1 Tổng quát về đời sống của dữ liệu của ứng dụng VC#
8.2 Các cấp độ ghi/đọc dữ liệu phổ biến
8.3 Ghi/đọc chuỗi byte thô ra/từ file
8.4 Ghi/đọc chuỗi ký tự ra/từ file
8.5 Ghi/đọc dữ liệu có kiểu định sẵn ra/từ file nhị phân
8.6 Ghi/đọc dữ liệu có kiểu định sẵn ra/từ file văn bản
8.7 Ghi/Đọc hệ thống đối tượng ra/vào file
8.8 Thí dụ về đọc/ghi hệ thống đối tượng
8.9 Kết chương

8.0 Dẫn nhập

- Chương này giới thiệu các đối tượng phục vụ ghi/đọc dữ liệu ra/vào file cùng các tác vụ ghi/đọc dữ liệu cổ điển ra/vào file.
- Chương này cũng giới thiệu các đối tượng phục vụ ghi/đọc hệ thống đối tượng ra/vào file cùng các tác vụ ghi/đọc hệ thống đối tượng có mối quan hệ tham khảo phức tạp ra/vào file.

8.1 Tổng quát về đời sống của dữ liệu ⊂ ứng dụng VC#

- Nhu cầu ghi/đọc nội dung của các biến dữ liệu thường rơi vào 3 tình huống chính yếu sau đây :
  1. Lưu kết quả của phiên làm việc hiện hành để dùng lại cho phiên làm việc kế tiếp.
  2. Nhập rất nhiều dữ liệu cho phần mềm.
  3. Xuất rất nhiều dữ liệu cho người dùng.

8.2 Các cấp độ ghi/đọc dữ liệu phổ biến

1. ghi/đọc chuỗi byte thô ra/từ file, ngữ nghĩa của các byte do chương trình tự qui định.
2. ghi/đọc chuỗi ký tự theo cách mã hóa xác định (ASCII, UTF8, UCS-2,...) ra/từ file.
3. ghi/đọc các dữ liệu thuộc các kiểu cơ bản định sẵn như bool, byte, int, double, String,... ra/từ file theo dạng nhị phân, là dạng mã hóa gốc bên trong chương trình.
4. giải mã các dữ liệu thuộc các kiểu cơ bản định sẵn như bool, byte, int, double, String,... thành chuỗi văn bản, ghi chuỗi ra file văn bản để khi cần đọc các chuỗi văn bản từ file vào chương trình, mã hóa từng chuỗi trên file thành dữ liệu nhị phân bên trong chương trình trước khi xử lý tiếp.
5. ghi/đọc đối tượng và các đối tượng được tham khảo trực tiếp hay gián tiếp bởi đối tượng gốc ra/từ file nhị phân hay file XML.

8.3 Ghi/đọc chuỗi byte thô ra/từ file

- class sử dụng : FileStream
- các tác vụ : WriteByte(), ReadByte()
- Qui trình ghi điển hình như sau :

```
//1. tạo đối tượng quản lý file để ghi dữ liệu
FileStream oFile = new FileStream("C:\\data.bin",
    FileMode.Create);
//2. ghi tuần tự từng byte ra file
oFile.WriteByte(1byte);
...
//3. đóng file lại để phòng ngừa việc ghi bất hợp pháp lên file
oFile.Close();
```

8.3 Ghi/đọc chuỗi byte thô ra/từ file

- Code thí dụ việc ghi chuỗi byte :

```
Bitmap bmScreen;    //định nghĩa biến chứa ảnh bitmap
Graphics gpScreen;  //định nghĩa biến chứa các tác vụ xử lý nội dung đối tượng giao diện
this.Hide();        //nếu cần, ẩn Form hiện hành để không xuất hiện trong bitmap chụp
//tạo đối tượng bitmap cùng kích thước màn hình
bmScreen = new Bitmap(Screen.PrimaryScreen.Bounds.Width,
     Screen.PrimaryScreen.Bounds.Height, PixelFormat.Format32bppArgb);
//tạo đối tượng Graphics kết hợp với bmScreen
gpScreen = Graphics.FromImage(bmScreen);
//chụp toàn màn hình và lưu vào bmScreen
gpScreen.CopyFromScreen(Screen.PrimaryScreen.Bounds.X,
    Screen.PrimaryScreen.Bounds.Y, 0, 0,
     Screen.PrimaryScreen.Bounds.Size, CopyPixelOperation.SourceCopy);
```

8.3 Ghi/đọc chuỗi byte thô ra/từ file

- Code thí dụ việc ghi chuỗi byte :

```
//hiển thị lại Form hiện hành nếu cần
this.Show();
//chuyển bitmap thành dãy các byte liên tục
ImageConverter imgConverter = new ImageConverter();
byte[] xByte = (byte[])imgConverter.ConvertTo(bmScreen, typeof(byte[]));
//1. tạo file để lưu trữ dãy các byte nội dung của bitmap
FileStream oFile = new FileStream("d:\\screen.bin", FileMode.Create);
//2. lặp ghi từng byte của dãy xByte ra file
for (int i = 0; i < xByte.Length; i++) oFile.WriteByte(xByte[i]);
//3. đóng file lại
oFile.Close();
```

8.3 Ghi/đọc chuỗi byte thô ra/từ file

- Qui trình đọc chuỗi byte điển hình như sau :

```
//1. tạo đối tượng quản lý file để đọc dữ liệu
FileStream inFile = new FileStream("C:\\data.bin",
    FileMode.Open);
//2. đọc tuần tự từng byte từ file
inFile.ReadByte(1byte);
...
//3. đóng file lại để phòng ngừa việc đọc bất hợp pháp lên file
inFile.Close();
```

8.3 Ghi/đọc chuỗi byte thô ra/từ file

- Code thí dụ việc đọc chuỗi byte :

```
//1. mở file chứa dữ liệu cần đọc
FileStream inFile = new FileStream("d:\\screen.bin", FileMode.Open);
//tạo biến dãy các byte với độ lớn bằng độ lớn của file
byte[] xData = new byte[inFile.Length];
//2. lặp đọc từng byte từ file vào dãy xData
for (int i = 0; i < inFile.Length; i++) xData[i] = (byte)inFile.ReadByte();
//3. đóng file lại
inFile.Close();
//tạo đối tượng Stream
MemoryStream ms = new MemoryStream(xByte, 0, xByte.Length);
ms.Position = 0;
```

8.3 Ghi/đọc chuỗi byte thô ra/từ file

- Code thí dụ việc đọc chuỗi byte :

```
//copy dãy byte vào đối tượng Stream
ms.Write(xData, 0, xData.Length);
//biến đổi đối tượng Stream thành đối tượng Image
System.Drawing.Image image = System.Drawing.Image.FromStream(ms,
    true);
//hiển thị bitmap lên đối tượng PictureBox của Form ứng dụng
picImage.SizeMode = PictureBoxSizeMode.StretchImage;
picImage.Image = image;
```

8.4 Ghi/đọc chuỗi ký tự ra/từ file

- class sử dụng : StreamWriter, StreamReader
- các tác vụ : Write (), Read ()
- Qui trình ghi điển hình như sau :

```
//1. tạo đối tượng quản lý file để ghi dữ liệu
StreamWriter oFile = new StreamWriter("d:\\data.txt", false,
    Encoding.Unicode);
//2. ghi tuần tự từng ký tự ra file
oFile.Write(buf[i]);
...
//3. đóng file lại để phòng ngừa việc ghi bất hợp pháp lên file
oFile.Close();
```

- Lưu ý ký tự được đổi từ mã Unicode 2 byte bên trong chương trình ra mã ký tự do file qui định.

8.4 Ghi/đọc chuỗi ký tự ra/từ file

- Code thí dụ việc ghi chuỗi ký tự :

```
//xử lý và tạo chuỗi văn bản kết quả rồi chứa vào biến để lưu giữ
String buf = "Thí dụ để xuất ma trận ra file văn bản, ta có thể qui ước các dữ liệu sẽ xuất nhập
    tuần tự : số hàng, số cột, giá trị các phần tử hàng 1 từ trái sang phải, giá trị các phần tử hàng 2
    từ trái sang phải,... cuối cùng là giá trị các phần tử hàng cuối của ma trận. Giữa 2 phần tử kề
    nhau ta xuất thêm 1 hay nhiều dấu ngăn phù hợp như khoảng trắng, dấu phẩy, dấu đóng cột,
    dấu xuống hàng... Sau đây là nội dung file văn bản miêu tả ma trận có 5 hàng, 7 cột";
//1. tạo file để lưu trữ chuỗi ký tự theo cách mã hóa UCS-2
StreamWriter oFile = new StreamWriter("d:\\data.txt", false,
     Encoding.Unicode);
//2. lặp ghi từng ký tự của chuỗi văn bản ra file
for (int i = 0; i < buf.Length; i++) oFile.Write(buf[i]);
//3. đóng file lại
oFile.Close();
```

8.4 Ghi/đọc chuỗi ký tự ra/từ file

- Qui trình đọc chuỗi ký tự điển hình như sau :

```
//1. tạo đối tượng quản lý file để đọc dữ liệu
StreamReader inFile = new StreamReader(fileName,
    Encoding.Unicode);
//2. đọc tuần tự từng ký tự từ file
ch = (char)inFile.Read();
...
//3. đóng file lại để phòng ngừa việc đọc bất hợp pháp lên file
inFile.Close();
```

- Lưu ý sau khi đọc được ký tự từ file, hàm Read sẽ đổi từ mã gốc trên file sang mã Unicode 2 byte bên trong chương trình.

8.4 Ghi/đọc chuỗi ký tự ra/từ file

- Code thí dụ việc đọc chuỗi ký tự :

```
//định nghĩa biến chứa dãy ký tự đọc từ file vào
char[] xchar;
//định nghĩa biến chứa đường dẫn của file van bản cần đọc
String fileName = "d:\\data.txt";
//tạo đối tượng FileInfo kết hợp với file để lấy các thông tin về file
FileInfo fi = new FileInfo(fileName);
//1. tạo đối tượng file chứa chuỗi ký tự theo cách mã hóa UCS-2 cần đọc
StreamReader inFile = new StreamReader(fileName, Encoding.Unicode);
//phân phối vùng nhớ cần thiết cho biến array các ký tự
xchar = new char[fi.Length];
```

8.4 Ghi/đọc chuỗi ký tự ra/từ file

- Code thí dụ việc đọc chuỗi ký tự :

```
//2. lặp đọc từng ký tự từ file vào biến array các ký tự
int i = 0;
while (!inFile.EndOfStream)
    xchar[i++] = (char)inFile.Read();
//3. đóng file lại
inFile.Close();
//đổi array ký tự thành dạng chuỗi ký tự
String buf = new String(xchar);
//hiển thị chuỗi kết quả lên TextBox
txtOutput.Text = buf;
```

8.5 Ghi/đọc dữ liệu có kiểu định sẵn ra/từ file nhị phân

- class sử dụng : FileStream, BinaryWriter, BinaryReader
- các tác vụ : Write (), ReadBoolean(), …, ReadSTring()
- Lưu ý :
  - khi ghi 1 dữ liệu ra file nhị phân, máy sẽ để nguyên định dạng gốc bên trong rồi ghi ra file (td. giá trị double chiếm 8 byte sẽ được ghi ra file thành 8 byte) trừ trường hợp dữ liệu chuỗi. Trong trường hợp này máy sẽ giải mã chuỗi từ mã Unicode 2 byte thành mã UTF-8 rồi ghi ra file (có ghi thêm độ dài chuỗi để khi đọc lại máy biết chính xác).

8.5 Ghi/đọc dữ liệu có kiểu định sẵn ra/từ file nhị phân

- Lưu ý :
  - khi đọc 1 dữ liệu từ file nhị phân vào, máy sẽ để nguyên định dạng gốc trên file và copy vào bộ nhớ chương trình (td. giá trị double chiếm 8 byte trên file sẽ được đọc vào thành 8 byte trong biến double) trừ trường hợp dữ liệu chuỗi. Trong trường hợp này máy sẽ mã hóa chuỗi từ mã UTF-8 trên file thành mã Unicode 2 byte của kiểu String trong chương trình.

8.5 Ghi/đọc dữ liệu có kiểu định sẵn ra/từ file nhị phân

- Qui trình ghi điển hình như sau :

```
//1. tạo đối tượng quản lý file
FileStream stream = new FileStream("C:\\data.bin", FileMode.Create);
//2. tạo đối tượng phục vụ ghi file
BinaryWriter writer = new BinaryWriter(stream);
//3. xử lý dữ liệu theo yêu cầu chương trình
int i = -15;
double d = -1.5;
String s = "Nguyễn Văn Hiệp";
bool b = true;
//4. ghi dữ liệu ra file
writer.Write(b); writer.Write(i); writer.Write(d); writer.Write(s);
//5. đóng các đối tượng được dùng lại
writer.Close(); stream.Close();
```

8.5 Ghi/đọc dữ liệu có kiểu định sẵn ra/từ file nhị phân

- Tác vụ Write của class BinaryWriter có 14 biến thể 1 tham số để ghi được 14 kiểu dữ liệu định sẵn phổ biến sau đây :
  - Boolean
  - Byte, SByte
  - Int16, Int32, Int64
  - UInt16, UInt32, UInt64
  - Single, Double, Decimal
  - Byte[] , Char[]
  - Char, String
- Muốn ghi nội dung của biến thuộc 1 trong 14 kiểu dữ liệu định sẵn trên, ta gọi tác vụ write theo dạng sau :

```
writer.Write(varname);     //writer là biến đối tượng BinaryWriter
```

8.5 Ghi/đọc dữ liệu có kiểu định sẵn ra/từ file nhị phân

- Tác vụ Write của class BinaryWriter còn có 2 biến thể 3 tham số để ghi được các phần tử chọn lọn trong danh sách :

```
//ghi count byte từ vị trí index trong danh sách buffer
BinaryWriter.Write(Byte[] buffer, int index, int count);
//ghi count ký tự từ vị trí index trong danh sách buffer
BinaryWriter.Write(Char[] buffer, int index, int count);
```

8.5 Ghi/đọc dữ liệu có kiểu định sẵn ra/từ file nhị phân

- Qui trình điển hình để đọc dữ liệu từ file nhị phân vào :

```
//1. tạo đối tượng quản lý file
FileStream stream = new FileStream("C:\\data.bin", FileMode.Open);
//2. tạo đối tượng phục vụ đọc file
BinaryReader reader = new BinaryReader(stream);
//3. định nghĩa các biến dữ liệu theo yêu cầu chương trình
int i; double d; String s; bool b;
//4. đọc dữ liệu từ file vào các biến
b= reader.ReadBoolean();          //đọc trị luận lý
i = reader.ReadInt32();           //đọc số nguyên 32 bit
d = reader.ReadDouble();          //đọc số thực chính xác kép
s = reader.ReadString();          //đọc chuỗi
//5. đóng các đối tượng được dùng lại
reader.Close(); stream.Close();
```

8.5 Ghi/đọc dữ liệu có kiểu định sẵn ra/từ file nhị phân

- Class BinaryReader cung cấp 14 tác vụ khác nhau để đọc dữ liệu nhị phân vào biến thuộc 14 kiểu dữ liệu định sẵn của VC# :
  - ReadBoolean
  - ReadByte, ReadSByte
  - ReadChar
  - ReadInt16, ReadInt32, ReadInt64
  - ReadUInt16, ReadUInt32, ReadUInt64
  - ReadSingle, ReadDouble, ReadDecimal
  - ReadString

8.5 Ghi/đọc dữ liệu có kiểu định sẵn ra/từ file nhị phân

- Class BinaryReader còn cung cấp thêm 2 tác vụ khác để đọc vào array các byte hay array các ký tự :

```
//đọc count byte từ file vào biến array các byte
byte[] dsbyte = BinaryReader.ReadBytes(int count);
//đọc count ký tự từ file vào biến array các ký tự
Char[] dschar = BinaryReader.ReadChars(int count);
```

8.6 Ghi/đọc dữ liệu có kiểu định sẵn ra/từ file văn bản

- Mặc dù việc ghi/đọc dữ liệu ra file ở dạng nhị phân (y như trong máy) là rất đơn giản, hiệu quả (khỏi phải thực hiện mã hóa/giải mã dữ liệu). Tuy nhiên, file nhị phân cũng có 1 số nhược điểm :
  - người dùng khó xem, khó kiểm tra nội dung của file.
  - người dùng khó tạo dữ liệu dưới dạng nhị phân để chương trình đọc vào xử lý.
- Trong trường hợp cần nhập nhiều thông tin cho chương trình, ta không thể dùng các đối tượng giao diện như textbox, listbox. Trong trường hợp này, ta sẽ dùng trình soạn thảo văn bản để soạn dữ liệu dưới dạng văn bản hầu xem/kiểm tra/sửa chữa dễ dàng. File văn bản chứa dữ liệu là danh sách gồm nhiều chuỗi, mỗi chuỗi miêu tả 1 dữ liệu (luận lý, số nguyên, số thực, chuỗi,...), các chuỗi sẽ được ngăn cách nhau bởi 1 hay nhiều dấu ngăn. Dấu ngăn thường dùng là ký tự gióng cột TAB, ký tự xuống hàng.

8.6 Ghi/đọc dữ liệu có kiểu định sẵn ra/từ file văn bản

- class sử dụng : StreamWriter
- các tác vụ : Write ()
- Lưu ý :
  - .Net chỉ cung cấp class StreamWriter để phục vụ chiều ghi dữ liệu từ phần mềm ra file văn bản, chứ không cung cấp class phục vụ chiều đọc dữ liệu từ file văn bản vào phần mềm.
  - khi ghi 1 dữ liệu ra file văn bản, máy sẽ giải mã dữ liệu từ định dạng nhị phân bên trong ra dạng chuỗi (theo mã qui định bởi file).
  - khi đọc 1 dữ liệu từ file văn bản, máy sẽ mã hóa dữ liệu từ định dạng chuỗi văn bản trên file thạnh dạng nhị phân bên trong chương trình.

8.6 Ghi/đọc dữ liệu có kiểu định sẵn ra/từ file văn bản

- Qui trình điển hình để ghi dữ liệu trong chương trình ra file ở dạng text (giãi mã dữ liệu nhị phân thành dạng chuỗi) :

```
//1. tạo đối tượng quản lý file
FileStream stream = new FileStream("C:\\data.txt", FileMode.Create);
//2. tạo đối tượng phục vụ ghi file
StreamWriter writer = new StreamWriter(stream, Encoding.Unicode);
```

8.6 Ghi/đọc dữ liệu có kiểu định sẵn ra/từ file văn bản

```
//3. xử lý dữ liệu theo yêu cầu chương trình
int i = -15;
double d = -1.5;
String s = "Nguyễn Văn Hiệp";
bool b = true;
//4. ghi dữ liệu ra file
writer.Write(b); writer.Write("\t");   //ghi 1 dữ liệu và dấu ngăn
writer.Write(i); writer.WriteLine();   //ghi 1 dữ liệu và dấu ngăn
writer.Write(d); writer.Write("\t");   //ghi 1 dữ liệu và dấu ngăn
writer.Write(s); writer.Write("\t");   //ghi 1 dữ liệu và dấu ngăn
//5. đóng các đối tượng được dùng lại
writer.Close();
stream.Close();
```

8.6 Ghi/đọc dữ liệu có kiểu định sẵn ra/từ file văn bản

- Qui trình điển hình để đọc dữ liệu từ file text vào chương trình (mã hóa dữ liệu từ chuỗi thành dữ liệu nhị phân) :

```
//1. tạo đối tượng quản lý file
FileStream stream = new FileStream("C:\\data.txt", FileMode.Open);
//2. tạo đối tượng phục vụ đọc file
StreamReader reader=new StreamReader(stream,Encoding.Unicode);
//3. định nghĩa các biến dữ liệu theo yêu cầu chương trình
int i; double d; String s; bool b; String buf=null;
//4. đọc dữ liệu từ file vào các biến
ReadItem(reader,ref buf); b = Boolean.Parse(buf);    //đọc trị luận lý
ReadItem(reader,ref buf); i = Int32.Parse(buf);      //đọc số nguyên 32 bit
ReadItem(reader,ref buf); d = Double.Parse(buf);     //đọc số thực
ReadItem(reader,ref buf); s = buf; //đọc chuỗi
//5. đóng các đối tượng được dùng lại
reader.Close(); stream.Close();
```

8.6 Ghi/đọc dữ liệu có kiểu định sẵn ra/từ file văn bản

```
//hàm đọc chuỗi miêu tả 1 dữ liệu nào đó
static void ReadItem(StreamReader reader, ref String buf) {
    char ch;
    //thiết lập chuỗi nhập được lúc đầu là rỗng
    buf = "";
    //lặp cho đến khi hết file
    while (reader.EndOfStream != true) {
        ch = (char)reader.Read(); //đọc 1 ký tự
        if (ch != '\t' && ch != '\r' && ch!='\n') //nếu là ký tự bình thường
                buf += ch.ToString();
        else { //nếu là dấu ngăn thì kết thúc việc đọc chuỗi
            if (ch == '\r') reader.Read(); //đọc bỏ luôn ký tự '\n'
            return;                        //trả kết quả về nơi gọi
        }
    }
}
```

8.6 Ghi/đọc dữ liệu có kiểu định sẵn ra/từ file văn bản

- Để giúp người lập trình dễ dàng viết code đọc dữ liệu chuỗi vào các biến thuộc các kiểu định sẵn, ta định nghĩa 1 class mới có tên là DataScanner, class này chứa các tác vụ đọc dữ liệu như sau :
  - ReadBoolean
  - ReadByte, ReadSByte
  - ReadChar
  - ReadInt16, ReadInt32, ReadInt64
  - ReadUInt16, ReadUInt32, ReadUInt64
  - ReadSingle, ReadDouble, ReadDecimal
  - ReadString
- Class DataScanner sẽ dùng dịch vụ đọc chuỗi ký tự thô của class StreamReader như sau :

8.6 Ghi/đọc dữ liệu có kiểu định sẵn ra/từ file văn bản

```
class DataScanner {
    StreamReader reader; //đối tượng phục vụ đọc chuỗi ký tự thô
    String sdel; //chứa các ký tự dấu ngăn giữa các chuỗi dữ liệu
    //tác vụ khởi tạo
    public DataScanner(String fname, Encoding enc) {
        //tạo đối tượng phục vụ đọc file
        reader = new StreamReader(fname, enc);
    }
    //tác vụ dọn dẹp
    ~DataScanner()  {
        //đóng đối tượng phục vụ đọc file
        reader.Close();
    }
```

8.6 Ghi/đọc dữ liệu có kiểu định sẵn ra/từ file văn bản

```
//tác vụ đóng đối tượng
public void Close() {
    //đóng đối tượng phục vụ đọc file
    reader.Close();
}
//tác vụ thiết lập chuỗi các dấu ngăn được dùng trong file
public void useDelimiter(String s) {
    sdel = s;
}
//tác vụ kiểm tra ký tự có phải là dấu ngăn được dùng trong file không
public bool isDelimiter(char ch) {
    for (int i = 0; i < sdel.Length; i++)
        if (sdel[i] == ch) return true;
    return false;
}
```

8.6 Ghi/đọc dữ liệu có kiểu định sẵn ra/từ file văn bản

```
//hàm đọc chuỗi hiện hành miêu tả 1 dữ liệu nào đó
private String ReadItem() {
    String buf =  ""; //thiết lập chuỗi nhập được lúc đầu là rỗng
    char ch = '\0';
    while (reader.EndOfStream != true) {  //lặp đọc bỏ các dấu ngăn
        ch = (char)reader.Read(); //đọc 1 ký tự
        if (!isDelimiter(ch)) break;            //nếu là ký tự bình thường thì dừng
    }
    buf += ch.ToString();
    //lặp đọc các ký tự của chuỗi dữ liệu
    while (reader.EndOfStream != true) {
        ch = (char)reader.Read();               //đọc 1 ký tự
        if (isDelimiter(ch)) return buf;        //nếu là dấu ngăn thì dừng
        buf += ch.ToString();                   //chứa ký tự vào bộ đệm
    }
    return buf;
}
```

8.6 Ghi/đọc dữ liệu có kiểu định sẵn ra/từ file văn bản

```
//tác vụ đọc giá trị luận lý
public bool nextBoolean() {
    String buf;
    try {
        buf = ReadItem();
        return Boolean.Parse(buf);
    }
    catch (Exception e) {
        throw new Exception("Không đọc được số nguyên cho bạn");
    }
}
```

8.6 Ghi/đọc dữ liệu có kiểu định sẵn ra/từ file văn bản

```
//tác vụ đọc số kiểu int
public int nextInt() {
    String buf;
    try {
        buf = ReadItem();
        return Int32.Parse(buf);
    } catch (Exception e) {
        throw new Exception("Không đọc được số nguyên cho bạn");
    }
}
```

8.6 Ghi/đọc dữ liệu có kiểu định sẵn ra/từ file văn bản

```
//tác vụ đọc số kiểu double
public double nextDouble() {
    String buf;
    try {
        buf = ReadItem();
        return Double.Parse(buf);
    } catch (Exception e) {
        throw new Exception("Không đọc được số thực double cho bạn");
    }
}
//định nghĩa các tác vụ khác để đọc dữ liệu kiểu khác nếu muốn
 // ....
```

8.6 Ghi/đọc dữ liệu có kiểu định sẵn ra/từ file văn bản

```
    //tác vụ đọc chuỗi
    public String nextString() {
        try {
            return ReadItem();
        } catch (Exception e) {
            throw new Exception("Không đọc được chuỗi văn bản cho bạn");
        }
    }
} //hết class DataScanner
```

8.6 Ghi/đọc dữ liệu có kiểu định sẵn ra/từ file văn bản

- Qui trình điển hình để đọc dữ liệu từ file text vào chương trình dùng class DataScanner :

```
//1. tạo đối tượng phục vụ đọc file văn bản
DataScanner dscan = new DataScanner("d:\\data.txt", Encoding.Unicode);
dscan.useDelimiter("\t\r\n");        //khai báo các dấu ngăn
//2. định nghĩa các biến dữ liệu theo yêu cầu chương trình
int i; double d; String s; bool b; String buf=null;
//3. đọc dữ liệu từ file vào các biến
b = dscan.nextBoolean();             //đọc trị luận lý
i = dscan.nextInt();                 //đọc số nguyên 32 bit
d = dscan.nextDouble();              //đọc số thực
s = dscan.nextString();              //đọc chuỗi
//4. đóng các đối tượng được dùng lại
dscan.Close();
```

8.7 Ghi/Đọc hệ thống đối tượng ra/vào file

- Đọc/ghi dữ liệu trên các biến thuộc kiểu giá trị (int, double, char[],..) rất dễ vì nội dung của các biến này không chứa tham khảo đến các thành phần khác. Ngược lại, việc đọc/ghi nội dung của 1 đối tượng thường rất khó khăn vì đối tượng có thể chứa nhiều tham khảo đến các đối tượng khác và các đối tượng có thể tham khảo vòng lẫn nhau. Để hỗ trợ việc đọc/ghi nội dung của đối tượng, VC# đề nghị kỹ thuật "Serialization".
- Một đối tượng chỉ có thể được "serialize/deserialize" (ghi/đọc dùng kỹ thuật Serialization) nếu nó thuộc class "serializable". Để định nghĩa 1 class "serializable" dễ dàng và đơn giản nhất, ta chỉ cần thêm mệnh đề [Serializable] trước phát biểu định nghĩa class đó :

```
[Serializable]
public class A {...}
```

8.7 Ghi/Đọc hệ thống đối tượng ra/vào file

- Để ghi 1 đối tượng (và toàn bộ các đối tượng mà nó phụ thuộc) ở dạng nhị phân, ta viết template như sau :

```
//1. xử lý và xây dựng đối tượng
B b;
//2. định nghĩa đối tượng FileStream miêu tả file chứa kết quả
FileStream fs = new FileStream("c:\\data.obj", FileMode.Create);
//3. tạo đối tượng BinaryFormatter phục vụ ghi đối tượng
BinaryFormatter formatter = new BinaryFormatter();
//4. gọi tác vụ Serialize  của formatter để ghi đối tượng
formatter.Serialize(fs, b);
//đóng file lại
fs.Flush();
fs.Close();
```

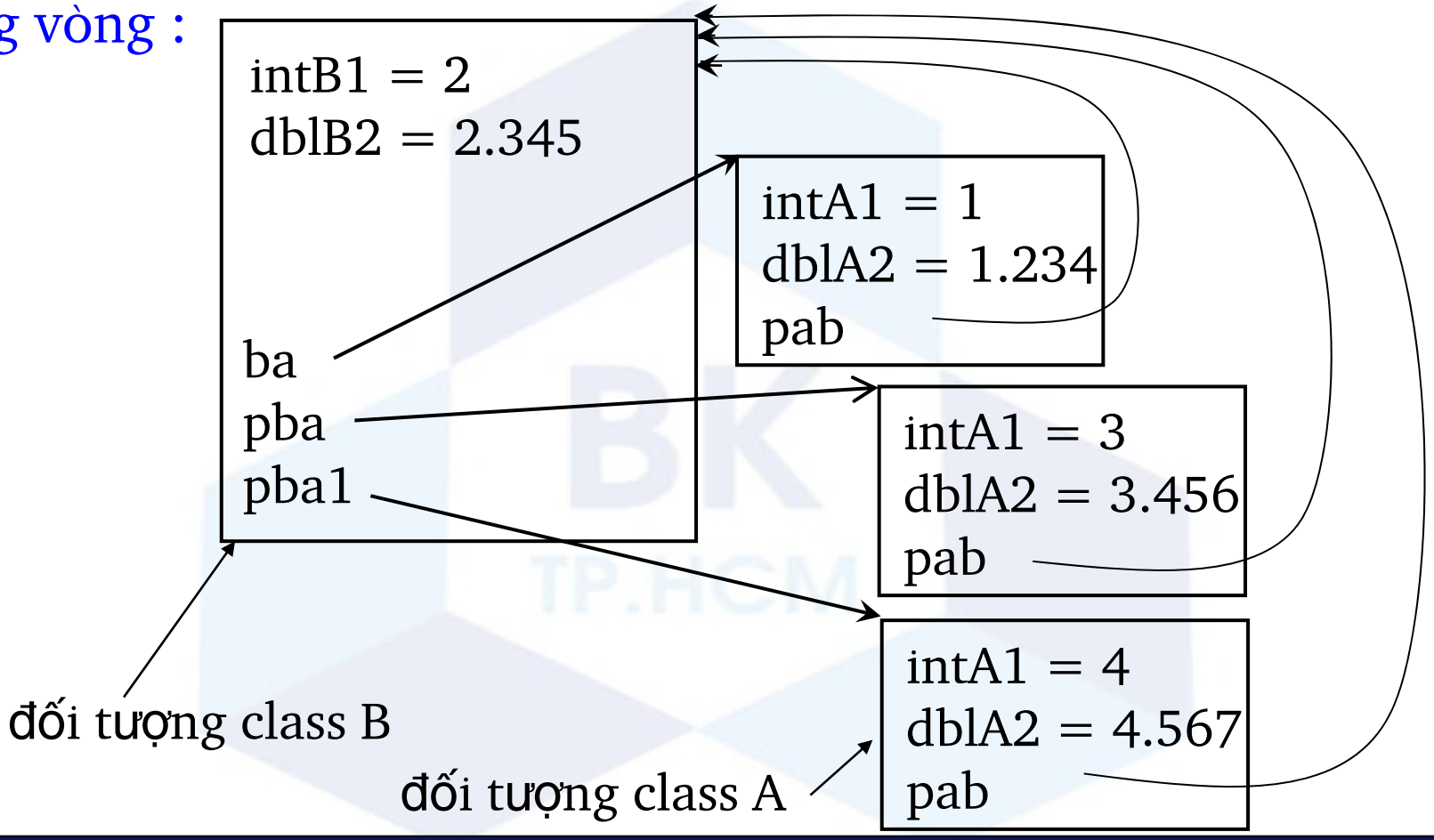
8.7 Ghi/Đọc hệ thống đối tượng ra/vào file

- Để đọc lại 1 đối tượng (và toàn bộ các đối tượng mà nó phụ thuộc) ở dạng nhị phân ta, viết template như sau :

```
//1. định nghĩa biến đối tượng để chứa nội dung từ file
B b;
//2. định nghĩa đối tượng FileStream miêu tả file chứa dữ liệu đã có
FileStream fs = new FileStream("c:\\data.obj", FileMode.Open);
//3. tạo đối tượng BinaryFormatter phục vụ đọc đối tượng
BinaryFormatter formatter = new BinaryFormatter();
//4. gọi tác vụ Deserialize để đọc đối tượng từ file vào
b = (B) formatter.Deserialize(fs);
//đóng file lại
fs.Close();
```

8.8 Thí dụ về đọc/ghi hệ thống đối tượng

- Giả sử ta có hệ thống các đối tượng với trạng thái và mối quan hệ giữa chúng cụ thể như sau. Lưu ý chúng có mối quan hệ bao gộp dạng vòng :



8.8 Thí dụ về đọc/ghi hệ thống đối tượng

- Giả sử biến b đang tham khảo tới đối tượng class B. Hãy viết chương trình ghi hệ thống đối tượng này lên file để khi cần, đọc lại vào bộ nhớ hầu xử lý tiếp.
- Qui trình xây dựng ứng dụng giải quyết yêu cầu trên như sau :

1. Chạy VS .Net, chọn menu File.New.Project để hiển thị cửa sổ New Project.
2. Mở rộng mục Visual C# trong TreeView "Project Types", chọn mục Windows, chọn icon "Console Application" trong listbox "Templates" bên phải, thiết lập thư mục chứa Project trong listbox "Location", nhập tên Project vào textbox "Name:" (td. WRObject), click button OK để tạo Project theo các thông số đã khai báo.

8.8 Thí dụ về đọc/ghi hệ thống đối tượng

3. Ngay sau Project vừa được tạo ra, cửa sổ soạn code cho chương trình được hiển thị. Thêm lệnh các using sau đây vào đầu file :

```
using System.IO;
using System.Runtime.Serialization;
using System.Runtime.Serialization.Formatters.Binary;
```

4. Viết code cho hàm Main và các hàm dịch vụ khác nhau sau :

```
class Program {
    static String fbuf;
    static void Main(string[] args) { //điểm nhập của chương trình
        //xây dựng hệ thống đối tượng và ghi lên file
        Create_SaveObject();
        //đọc lại hệ thống đối tượng
        //ReadObject();
    }
```

8.8 Thí dụ về đọc/ghi hệ thống đối tượng

```
//hàm xây dựng hệ thống đối tượng và ghi lên file
public static void Create_SaveObject() {
    //khở tạo đối tượng b theo hình ở silde 24
    B b = new B();
    b.init(2,2.345);
    b.Setba(1,1.234,b);
    b.Setpba(3,3.1416,b);
    b.Setpba1(4,4.567,b);
```

8.8 Thí dụ về đọc/ghi hệ thống đối tượng

```
        //ghi đối tượng b dùng kỹ thuật Serialization
        try {
            //1. định nghĩa đối tượng FileStream miêu tả file chứa kết quả
            FileStream fs = new FileStream("c:\\data.obj", FileMode.Create);
            //2. tạo đối tượng BinaryFormatter phục vụ ghi đối tượng
            BinaryFormatter formatter = new BinaryFormatter();
            //3. gọi tác vụ Serialize  của formatter để ghi đối tượng
            formatter.Serialize(fs,b);
            //4. đóng file lại
            fs.Flush(); fs.Close();
        } catch (Exception e) { Console.WriteLine(e.ToString()); }
    }
```

8.8 Thí dụ về đọc/ghi hệ thống đối tượng

```
//hàm đọc đối tượng b dùng kỹ thuật Serialization
public static void ReadObject() {
  try {
    //1. định nghĩa đối tượng FileStream miêu tả file chứa dữ liệu đã có
    FileStream fs = new FileStream("c:\\data.obj", FileMode.Open);
    //2. tạo đối tượng BinaryFormatter phục vụ đọc đối tượng
    BinaryFormatter formatter = new BinaryFormatter();
    //3. gọi tác vụ Deserialize để đọc đối tượng từ file vào
    B b = (B) formatter.Deserialize(fs);
    //4. đóng file lại
    fs.Close();
  } catch (Exception e) { Console.WriteLine(e.ToString()); }
} //hết hàm Main
} //hết class program
```

8.8 Thí dụ về đọc/ghi hệ thống đối tượng

5. Ấn phải chuột vào phần tử gốc của cây Project trong cửa sổ Solution Explorer, chọn option Add.Class, đặt tên là A.cs để tạo ra file đặc tả class A. Khi cửa sổ hiển thị mã nguồn của class A hiển thị, đặc tả class A như đoạn code dưới đây :

```
//thêm lệnh using sau ở đầu file
using System.Runtime.Serialization;
namespace WRObject {
    [Serializable]
    public class A {
        //định nghĩa các thuộc tính dữ liệu
        private int intA1;
        private double dblA2;
        private B pab;
```

8.8 Thí dụ về đọc/ghi hệ thống đối tượng

```
        //định nghĩa các tác vụ
        public A() {}
        public void init(int a1, double a2, B p) {
            this.intA1 = a1;
            this.dblA2 = a2;
            this.pab = p;
        }
    }
}
```

6. Ấn phải chuột vào phần tử gốc của cây Project trong cửa sổ Solution Explorer, chọn option Add.Class, đặt tên là B.cs để tạo ra file đặc tả class B. Khi cửa sổ hiển thị mã nguồn của class B hiển thị, đặc tả class B như đoạn code dưới đây :

8.8 Thí dụ về đọc/ghi hệ thống đối tượng

```
//thêm lệnh using sau ở đầu file
using System.Runtime.Serialization;
namespace WRObject {
    [Serializable]
    public class B {
        //định nghĩa các thuộc tính dữ liệu
        private   int intB1;
        private   double dblB2;
        private   A ba;
        private  A pba;
        private  A pba1;
        //định nghĩa các tác vụ
        public B() { }
```

8.8 Thí dụ về đọc/ghi hệ thống đối tượng

```
    public void init(int b1, double b2) {
        this.intB1 = b1; this.dblB2 = b2;
       ba = new A(); pba = new A(); pba1 = new A();
    }
    public void Setba (int a1, double a2, B b) {
       this.ba.init (a1,a2,b);
    }
    public void Setpba (int a1, double a2, B b) {
       this.pba.init (a1,a2,b);
    }
    public void Setpba1 (int a1, double a2, B b) {
       this.pba1.init (a1,a2,b);
    }
} //hết class B
} //hết namespace WRObject
```

8.8 Thí dụ về đọc/ghi hệ thống đối tượng

7. Chọn menu Debug.Start Debugging để dịch và chạy ứng dụng. Hệ thống đối tượng sẽ được tạo ra và lưu lên file c:\data.obj.
8. Hiển thị cửa sổ soạn mã nguồn file Program.cs, chú thích lệnh gọi Create_SaveObject(); và bỏ chú thích lệnh gọi ReadObject();. Dời chuột về lệnh "B b = (B) formatter.Deserialize(fs);" trong hàm ReadObject(), click chuột vào lệnh trái của lệnh này để thiết lập điểm dừng. Chọn menu Debug.Start Debugging để dịch và chạy ứng dụng. Ứng dụng sẽ dừng ở lệnh dừng. Dời chuột về biến b, ta thấy cửa sổ hiển thị giá trị của biến này lúc này là null.
9. Ấn F10 để thực hiện đúng lệnh này rồi dừng lại, dời chuột về biến b, rồi mở rộng cây phân cấp miêu tả nội dung biến b và thấy nó chứa đúng thông tin như slide 14, nghĩa là chương trình đã đọc được toàn bộ hệ thống đối tượng đã lưu giữ trước đây.

8.9 Kết chương

- Chương này đã giới thiệu các đối tượng phục vụ ghi/đọc dữ liệu ra/vào file cùng các tác vụ ghi/đọc dữ liệu cổ điển ra/vào file.
- Chương này cũng đã giới thiệu các đối tượng phục vụ ghi/đọc hệ thống đối tượng ra/vào file cùng các tác vụ ghi/đọc hệ thống đối tượng có mối quan hệ tham khảo phức tạp ra/vào file.