**HW2: Programming Project #2**

**정보컴퓨터공학부**

**201424423**

**김영진**

This is the second part of a programming project begun in HW1. We add methods to store object as records in files and load objects from files, using the IOBUffer classes of this chapter.

1. Add Pack and Unpack methods to class **Student**. Use class BufferFile to create a file of students records. Test these methods using the types of buffers supported by the IOBuffer classes.
2. Add Pack and UnPack methods to class **CourseRegistration**. Use class BufferFile to create a file of course registration. Test these methods using the types of buffers supported by the IOBuffer classes.

**<student.cpp>**

**int** student::Pack (IOBuffer & Buffer) **const**

{

**int** numBytes;

Buffer . Clear ();

numBytes = Buffer . Pack (identifier);

**if** (numBytes == -1) **return** **FALSE**;

numBytes = Buffer . Pack (name);

**if** (numBytes == -1) **return** **FALSE**;

numBytes = Buffer . Pack (Address);

**if** (numBytes == -1) **return** **FALSE**;

numBytes = Buffer . Pack (dofe);

**if** (numBytes == -1) **return** **FALSE**;

numBytes = Buffer . Pack (credit\_hours);

**if** (numBytes == -1) **return** **FALSE**;

**return** **TRUE**;

}

**int** student::Unpack (IOBuffer & Buffer)

{

Clear ();

**int** numBytes;

numBytes = Buffer . Unpack (identifier);

**if** (numBytes == -1) **return** **FALSE**;

identifier[numBytes] = 0;

numBytes = Buffer . Unpack (name);

**if** (numBytes == -1) **return** **FALSE**;

name[numBytes] = 0;

numBytes = Buffer . Unpack (Address);

**if** (numBytes == -1) **return** **FALSE**;

Address[numBytes] = 0;

numBytes = Buffer . Unpack (dofe);

**if** (numBytes == -1) **return** **FALSE**;

dofe[numBytes] = 0;

numBytes = Buffer . Unpack (credit\_hours);

**if** (numBytes == -1) **return** **FALSE**;

    credit\_hours[numBytes] = 0;

**return** **TRUE**;

}

**<coursereg.cpp>**

**int** coursereg::Pack (IOBuffer & Buffer) **const**

{

**int** numBytes;

Buffer . Clear ();

numBytes = Buffer . Pack (course\_id);

**if** (numBytes == -1) **return** **FALSE**;

numBytes = Buffer . Pack (student\_id);

**if** (numBytes == -1) **return** **FALSE**;

numBytes = Buffer . Pack (course\_credit\_hours);

**if** (numBytes == -1) **return** **FALSE**;

numBytes = Buffer . Pack (course\_grade);

**if** (numBytes == -1) **return** **FALSE**;

**return** **TRUE**;

}

**int** coursereg::Unpack (IOBuffer & Buffer)

{

Clear ();

**int** numBytes;

numBytes = Buffer . Unpack (course\_id);

**if** (numBytes == -1) **return** **FALSE**;

course\_id[numBytes] = 0;

numBytes = Buffer . Unpack (student\_id);

**if** (numBytes == -1) **return** **FALSE**;

student\_id[numBytes] = 0;

numBytes = Buffer . Unpack (course\_credit\_hours);

**if** (numBytes == -1) **return** **FALSE**;

course\_credit\_hours[numBytes] = 0;

numBytes = Buffer . Unpack (course\_grade);

**if** (numBytes == -1) **return** **FALSE**;

course\_grade[numBytes] = 0;

**return** **TRUE**;

}

**<BufFileTest.cpp>**

**void** testBufferFile (IOBuffer & Buff, **char** \* myfile)

{

student stu;

    coursereg cou;

**int** result;

**int** st\_recaddr[10];

**int** cr\_recaddr[10];

    BufferFile TestOut (Buff);

    result = TestOut.Create (myfile, ios::in|ios::out);

    cout << "create file "<<result<<endl;

**if** (!result)

    {

        cout << "Unable to create file "<<myfile<<endl;

**return**;

    }

**for**(**int** i=0; i<10; i++){

st[i].Pack (TestOut.GetBuffer());

st\_recaddr[i] = TestOut.Write();

cout << "<student> write at " << st\_recaddr[i] << endl;

    }

**for**(**int** i=0; i<10; i++){

cr[i].Pack (TestOut.GetBuffer());

        cr\_recaddr[i] = TestOut.Write();

cout << "<course> write at " << cr\_recaddr[i] << endl;

    }

    TestOut.Close ();

    BufferFile TestIn (Buff);

    TestIn.Open (myfile, ios::in);

**for**(**int** i=0; i<10; i++){

TestIn.Read(st\_recaddr[i]);

        stu.Unpack (TestIn.GetBuffer());

stu.Print(cout, "record");

    }

**for**(**int** i=0; i<10; i++){

TestIn.Read(cr\_recaddr[i]);

cou.Unpack (TestIn.GetBuffer());

cou.Print(cout, "record");

    }

    TestIn.Close();

}

**void** testFixedField ()

{

cout <<"Testing Fixed Field Buffer"<<endl;

FixedFieldBuffer Buff (6);

student :: InitBuffer (Buff);

    coursereg :: InitBuffer (Buff);

testBufferFile(Buff, "fixlen2.dat");

}

**void** testLength ()

{

cout << "\nTesting LengthTextBuffer"<<endl;

LengthFieldBuffer Buff;

    student :: InitBuffer (Buff);

    coursereg :: InitBuffer (Buff);

testBufferFile(Buff, "length2.dat");

}

**void** testDelim ()

{

cout << "\nTesting DelimTextBuffer"<<endl;

DelimFieldBuffer::SetDefaultDelim ('|');

DelimFieldBuffer Buff;

    student :: InitBuffer (Buff);

    coursereg :: InitBuffer (Buff);

    testBufferFile(Buff, "delim2.dat");

}