Extra PADDING in a Record-based File

Kaminski / cs3310

*[This issue is relevant for **BINARY** files when they are written out (and read) as a **whole record** at a time (e.g., as a struct or class in C++) as opposed to byte-by-bytes (e.g., as with FileStream's pre-filled byte-array buffer)].*

When an int or float variable is declared, memory is allocated for it starting on a full-word boundary (i.e., a multiple of 4 bytes for the 32 bits). So if they're defined within a RECORD struct (or class), there will be preceding padding bytes in memory within the record definition to get to the full-word boundary for the int or float. Thus when a complete RECORD is WRITTEN to a file as a SINGLE UNIT, all bytes within that chunk of memory are written to the file, including the padding bytes. It all works fine if the program which then READS the file just reads the whole RECORD (as a single unit) into a record struct (or class) in memory, as long as the record is defined exactly as the record was defined in the program which created the file.

UNIX> cat recstruct.h

```
#ifndef RECSTRUCT_H
#define RECSTRUCT_H
#include <string.h>
class cls1
{       public:
        int num1;
        char ch;
        int num2;
        cls1()
        {       num1 = 15;
                ch = 'S';
                num2 = 18;
        }
        ~cls1()
        {}
};
class cls2
{       public:
        int num1;
        char arr[2];
        int num2;
        cls2()
        {       num1 = 20;
                strncpy(arr,"AB",2);
                num2 = 23;
        }
        ~cls2()
        {}
};
class cls3
{       public:
        int num1;
        char arr[4];
        int num2;
        cls3()
        {       num1 = 29;
                strncpy(arr,"BYTE",4);
                num2 = 30;
        }
        ~cls3()
        {}
};
#endif
```

UNIX> cat pmain.cxx

```
#include <iostream.h>
#include <fstream.h>
#include "recstruct.h"
int main()
{       cls1 ob1;
        cls2 ob2;
        cls3 ob3;

        ofstream ofile1("f1.out",ios::out);
        ofstream ofile2("f2.out",ios::out);
        ofstream ofile3("f3.out",ios::out);

        cout << "#bytes in Class 1: " << sizeof(ob1) << endl;
        cout << "#bytes in Class 2: " << sizeof(ob2) << endl;
        cout << "#bytes in Class 3: " << sizeof(ob3) << endl;

        ofile1.write(&ob1,sizeof(ob1));
        ofile2.write(&ob2,sizeof(ob2));
        ofile3.write(&ob3,sizeof(ob3));
}
```

```
UNIX> g++ -c pmain.cxx
UNIX> g++ pmain.o -o a1
UNIX> a1
#bytes in Class 1: 12
#bytes in Class 2: 12
#bytes Class 3: 12

UNIX> od -c f1.out
0000000   \0  \0  \0 017   S 276 356 364  \0  \0  \0 022

UNIX> od -c f2.out
0000000   \0  \0  \0 024   A   B  \0  \0  \0  \0  \0 027

UNIX> od -c f3.out
0000000   \0  \0  \0 035   B   Y   T   E  \0  \0  \0 036
```