

Description

We used binary files when we need to store complex data on disk without worrying about creating and parsing special data languages (like is done with XML). Since students in this course typically have some trouble with binary files, this project is practice for dealing with the files. What you will do is write a program that reads from a file that contains a sequence of records, each of which holds a string and a relative record number. The records are effectively a linked list, with the relative record number (RRN) giving the location of the next record relative to the current record. Note that the relative record number is not an absolute address, but the relative location of the record (more below). The string is actually stored as 20 characters, and is not necessarily null terminated. The relative record number is a 4-byte integer value, which gives the relative location of the next record. (The data in the class used to create the data is declared with the RRN first and the string second.)

We can logically think of the file as an array of records, with the first being record 0 (and physically located at byte 0). The record number and the physical byte address are different. So, while record 0 starts at byte 0, record 1 starts at byte 24. In the project input, the first record in the list will always be record 0, but subsequent records can be anywhere in the file. The relative record number tells you where the next record is relative to the current record. For instance, if the current record is record 20, then a RRN of 2 refers to record 22, and an RRN of -12 refers to record 8. You will start by reading record zero, and then use the RRN values to traverse the list using seeks to the proper (physical) locations in the file. For the seek operations you will have to compute the physical address from the record number.

Implementation

The only strict requirement is that you use binary I/O using C++ file streams using the `seekg` and `read` functions of records represented as objects of a class.

Your design must make appropriate use of classes. Data members of classes must be private.

You can expect that the binary input file will be correct, but you should handle file errors gracefully.

Input

Your program **must** read records from a file named `BinIO.bin` — use of another input file name will almost certainly result in a score of 0. This is a binary file and must be opened in binary mode. The file consists of an arbitrary number of 24 byte records. In each record, the first 4 bytes is an integer value that is the RRN, and the remaining 20 bytes are characters in a string. If the string held in the record is less than 20 characters the remainder will be padded by null characters, but no null characters will occur if the string is exactly 20 characters long.

Your program should exit when the next RRN is zero.

Output

Your program **must** write its output to a file named `BinIO.txt` — use of another output file name will undoubtedly result in a score of 0. Since your output for this assignment will be graded automatically, you must adhere to the specified format exactly. A sample output file is given at the end of this specification.

The first two lines should contain your name, course and project title, followed by separator line, as shown below. The remainder will be the strings from the binary file with their absolute record number. On each line only print five strings from the binary file (so there will be approximately 10 tokens on each line).

The output file at the end of this specification was produced by Keller's solution, executed on the given sample command file.

Submission

You will submit this assignment to the Curator System (read the *Student Guide*), and it will be graded automatically.

Instructions for submitting, and a description of how the grading is done, are contained in the *Student Guide*.

Note that the automated grading system requires that your program be submitted as a single source file. For this project you should develop your implementation using separate compilation, and then manually concatenate the code into a single .cpp file, being careful to get the order right and remove obsolete #include directives.

You will be allowed up to five submissions for this assignment. Use them wisely. Test your program thoroughly before submitting it. If you do not get a perfect score, analyze the problem carefully and test your fix before submitting again. The highest score you achieve will be counted.

The Student Guide can be found at: <http://ei.cs.vt.edu/~eags/Curator.html>

The submission client can be found at: <http://spasm.cs.vt.edu:8080/curator/>

Evaluation

Your submitted program will be assigned a score based upon the runtime testing performed by the Curator System. After that, your program will be given a brief evaluation by one of the GTAs, who will consider:

- whether your implementation makes appropriate use of C++ file i/o, and
- whether your design shows a good object-oriented decomposition of the given problem, and
- whether the internal documentation of your code is acceptable.

This evaluation will produce a deduction (possibly zero, of course) that will be applied to your runtime testing score to produce your final score for the project.

Programming Standards

While we won't be carefully evaluating your source code on this assignment for programming style, you should still observe good practice. See the Programming Standards page on the course website for specific requirements that should be observed in this course.

Pledge

Each of your program submissions must be pledged to conform to the Honor Code requirements for this course. Specifically, you **must** include the following pledge statement in the header comment for your program:

```
// On my honor:
//
// - I have not discussed the C++ language code in my program with
//   anyone other than my instructor or the teaching assistants
//   assigned to this course.
//
// - I have not used C++ language code obtained from another student,
//   or any other unauthorized source, either modified or unmodified.
//
// - If any C++ language code or documentation used in my program
//   was obtained from another source, such as a text book or course
//   notes, that has been clearly noted with a proper citation in
//   the comments of my program.
//
// - I have not designed this program in such a way as to defeat or
//   interfere with the normal operation of the Curator System.
//
// <Student Name>
```

Failure to include this pledge in a submission is a violation of the Honor Code.

Sample Input File

The following is a text description of the sample binary file. This is not the format of the file, but gives the relative record number and the corresponding string.

```
4 Nate
1 smooth
0 house.
7 his
1 sleepily
-2 eyes
3 house.
1 sits on
-7 his
2 Jake
-4 smooth
-4 slowly
```

Sample Output File

```
-----
Programmer: Ben Keller
CS 2604, Binary I/O Project
-----
Nate 0 sleepily 4 eyes 5 his 3 smooth 10
house. 6 Jake 9 slowly 11 sits on 7 his 8
smooth 1 house. 2
```