Project #0 – Object Oriented Programming in C++

Computer Science 2413 – Data Structures
Fall 2014

This project is individual work. Each student must complete this assignment independently.

User Request:

"Create a simple system to read and write bibliographic data."

Objectives:

1. Use redirected standard C++ I/O (`cin`, `cout`) to read a file and produce output. *10 points*
2. Develop and use at least six classes to model different types of bibliographic data. *20 points*
3. Use arrays of objects to store collections of the modeled data. *15 points*
4. Implement a program to manipulate bibliographic data as described below. *20 points*

- ► Develop and use an appropriate design. *15 points*
- ► Use proper documentation and formatting. *20 points*

Description:

For this project, you will put together several techniques and concepts you have learned in CS 2334 (or from a similar background) and some new techniques to make *PublicScholar*, an application that reads, stores, and output data on scholarly publications. PublicScholar will read through a simplified BibTeX file, store the data from each BibTeX entry in a structured way (e.g., the list of authors will be stored as an array of author names with one array element per name), and provide output derived from the input (e.g., besides listing the names of the authors, PublicScholar will indicate the number of authors on that publication).

Operational Issues:

Your program will read the bibliographic data file (a text file) via redirected standard input. Each bibliographic entry will be organized as follows:

Each entry will begin with an at sign (`@`), a designation of the type of entry (such as "`article`"), an opening curly brace (`{`), the name of the entry, and a comma (`,`).

The entry will then contain zero or more lines, each with the same basic form: zero or more spaces and/or tabs, a field designator (such as "`title`"), an equals sign (=), an opening curly brace, arbitrary text, a matching closing curly brace (`}`), and possibly a comma. If the comma is present, it means there may be another field left in the entry. If the comma is not present, there are no more fields remaining in the entry.

After all of the fields for the entry, there will be a line containing a single closing curly brace to match the one from the start of the entry.

The file will then contain zero or more empty lines before the start of the next entry.

For example:

```
@article{lattila_hybrid_2010,
    title = {Hybrid simulation models {\textendash} When, Why, How?},
    volume = {37},
    issn = {09574174},
    url = {http://linkinghub.elsevier.com/retrieve/pii/S0957417410003398},
    doi = {10.1016/j.eswa.2010.04.039},
    number = {12},
    urldate = {2013-06-24},
    journal = {Expert Systems with Applications},
    author = {L{\"a}ttil{\"a}, Lauri and Hilletofth, Per and Lin, Bishan},
    month = {dec},
    year = {2010},
    pages = {7969--7975}
}
```

The first line indicates that the entry is an article and "lattila_hyrbid_2010" is its name (which may be used by an author of a LaTeX document to refer to this article).

The second line indicates that the title of the article is "Hybrid simulation models – When, Why, How?" Note that an en dash (–) is not a standard keyboard character so it is represented in the file with a sequence of characters within matching opening and closing curly braces. So, when the description above says that there will be an opening curly brace, arbitrary text, and a matching closing curly brace, it should be noted that there may be additional (possibly nested) matching pairs of curly braces within the arbitrary text.

This entry happens to contain another 11 fields, one per line, before the closing curly brace for the entry.

For most of the fields, it is sufficient to record the field designator as a character array (with a maximum length of 20 characters) and the field value as a character array (with a maximum length of 2000 characters). So, for the first field of the example, the designator is "`title`" which should be encoded as a character array with the characters '`t`', '`i`', '`t`', '`l`', '`e`', and '`\0`' (to represent the end of the "string") stored in it. The value will be "`Hybrid simulation models {\textendash} When, Why, How?`" ending with the character '`\0`'. Note that the opening and closing curly braces from the start and end of the field are not part of the value stored in the character array.

For a few fields, more processing is needed for the values. For the volume, number, and year fields, the value should be stored as an integer. For the author field, the value should be an array of author names (where each name is a character array, similar to those described above). Note that in the input file, author names are separated from one another by a space, the word "and," and another space ("` and `"). You may assume that an author list contains no more than 50 authors.

For each entry, it is sufficient to store the type designation of the entry as a character array (with a maximum length of 20 characters), the name as a character array (with a maximum length of 50 characters), and an array of the fields present. Note that not every entry will have the same fields. You may assume that there are no more 20 fields per entry and no more than 20,000 bibliographic entries in the input file.

After reading in and storing all entries in the file, PublicScholar will print out all of the data in the same format as it was read in with one additional field per entry. This field will include some summary data about the entry derived from the other fields (if present) and will be designated "`counts`." It will give the number of fields in the original entry, the number of authors, the designator of the longest field, and the length of the longest field.

For example:

```
@article{lattila_hybrid_2010,
    title = {Hybrid simulation models {\textendash} When, Why, How?},
    volume = {37},
    issn = {09574174},
    url = {http://linkinghub.elsevier.com/retrieve/pii/S0957417410003398},
    doi = {10.1016/j.eswa.2010.04.039},
    number = {12},
    urldate = {2013-06-24},
    journal = {Expert Systems with Applications},
    author = {L{\"a}ttil{\"a}, Lauri and Hilletofth, Per and Lin, Bishan},
    month = {dec},
    year = {2010},
    pages = {7969--7975},
    counts = {fields: 11; authors: 3; longest field: url; length: 62}
}
```

This is the same example as before, with the addition of the final field indicating the specified counts.

Implementation Issues:

For each class you implement, you must implement appropriate (and appropriately named) constructor, destructor, accessor, mutator, display, and size methods. The size methods will indicate how many items are stored in a given array. For example, if the example entry above is stored in the variable `exEntry`, then `exEntry.getAuthors().size()` would return the value 3 because the example above shows three authors while `exEntry.getName().size()` would return the value 20 because the name of the entry contains 20 characters (including the final '`\0`').

Due Date:

You must submit an electronic copy of your source code through the dropbox in D2L by **Wednesday, September 17th by 2:45pm**.

Notes:

In this project, the only library you will use is iostream (`#include <iostream>`).

Be sure to use good object-oriented design in this project. That includes appropriate use of encapsulation, inheritance, overloading, overriding, accessibility modifiers, etc.

Be sure to use good code documentation. This includes header comments for all classes and methods, explanatory comments for each section of code, meaningful variable and method names, consistent indentation, etc.

You may write your program from scratch or may start from programs for which the source code is freely available on the web or through other sources (such as friends or student organizations). If you do not start from scratch, you must give a complete and accurate accounting of where all of your code came from and indicate which parts are original or changed, and which you got from which other source. Failure to give credit where credit is due is academic fraud and will be dealt with accordingly.