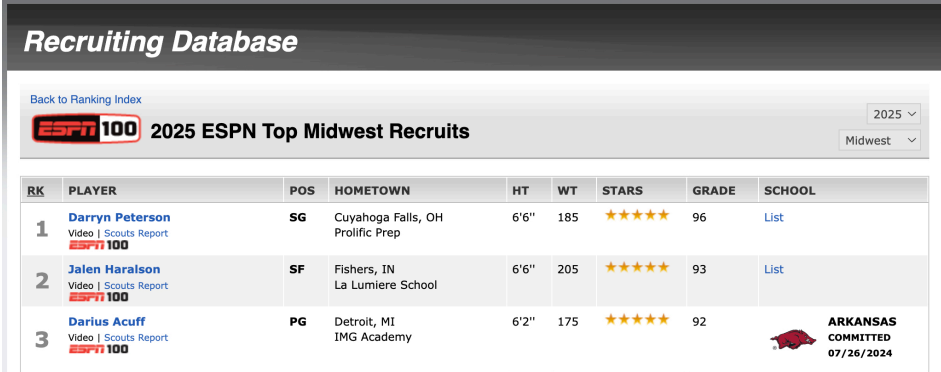



CSCE 2014 – Programming Project #2

Due Date – 09/11/2024

1. Problem Statement:

The goal of this programming assignment is to give students experience creating and using classes to store and search for information. To do this, you will create a RecruitDB class that contains a vector of Recruit objects. This class will contain methods for reading and writing information about ESPN's top 100 college basketball recruits from ASCII data files and a variety of methods for searching this Recruit database to locate players with desired attributes. Detailed requirements are listed below.



RK	PLAYER	POS	HOMETOWN	HT	WT	STARS	GRADE	SCHOOL
1	Darryn Peterson Video Scouts Report ESPN 100	SG	Cuyahoga Falls, OH Prolific Prep	6'6"	185	★★★★★	96	List
2	Jalen Haralson Video Scouts Report ESPN 100	SF	Fishers, IN La Lumiere School	6'6"	205	★★★★★	93	List
3	Darius Acuff Video Scouts Report ESPN 100	PG	Detroit, MI IMG Academy	6'2"	175	★★★★★	92	 ARKANSAS COMMITTED 07/26/2024

<https://www.espn.com/college-sports/basketball/recruiting/playerrankings>

Step 1 – Extend your Recruit Class

Your current Recruit class has “read_recruit” and “print_recruit” methods for reading Recruit information from the user and printing Recruit information to the screen. Your first task is to add two new methods to the Recruit class called “read_txt” and “print_txt”. The read_txt method should take an open input stream as a reference parameter and read information for one Recruit from this file. The print_txt method should take an open output stream as a parameter and print information for one Recruit into this file in the same format that information is presented in the input file.

Step 2 – Test File I/O Methods

Modify your main program from project1 to test the “read_txt” and “print_txt” methods above. To test file input, open the “top100.txt” file, loop calling “read_txt” until you reach the end of file, and close the input file. To test file output, open an output file, loop calling “print_txt” until you reach the end of the vector, and close the output file.

The “top100.txt” file contains Recruit data from ESPN’s website. You will notice that this ASCII file contains the recruits rank and there is a blank line between recruits. Your “read_txt” method should ignore the recruits rank (because it is not in the Recruit object) and skip over the blank line. Your “print_txt” method should calculate and print the rank for each recruit, and print blank lines between recruits so the input and output files are identical.

Step 3 – Create a RecruitDB Class

Create a C++ class called “RecruitDB” that one private variable, a vector of Recruit objects. Your RecruitDB class should have a default constructor, a copy constructor, and a destructor. Your RecruitDB class should have a “read_txt” method that has the name of the input file as a parameter and a “print_txt” method that has the name of the output file as a parameter. Your implementation of these two methods can reuse the file I/O code you developed in the previous step.

Step 4 – Add Search Methods to RecruitDB

Your next task is to create **four** methods for searching the vector of recruits and printing information based on the attributes of a Recruit. For example, you could create a method called “searchState” that prints all of the recruits from a given state. You are welcome to use any combination of the Recruit attributes that you think are interesting and/or potentially useful for coaches looking for players. At least one of your new search methods must be based on **two** or more Recruit attributes. If you decide to use Height in your search method, you will need to implement a function to convert the Height string to an integer or float for comparison purposes.

Step 5 – Test RecruitDB Methods

Finally, you should create a main program that initializes a RecruitDB object by reading “top100.txt”. Your program should then call each of your search methods functions several times to demonstrate correctness. You can do this with a very simple menu interface, or you can call RecruitDB methods in the main program with “hard coded” parameters. Include your program output in your project report to demonstrate your program correctness.

2. Design:

For this assignment, you have several important design decisions. First, you need to design the user interface. Specifically, what messages the program will print out, and what type of information the user should enter. If the program is reading or writing ASCII files, you also need to design the format of these files. In particular, what information will be on each line, and how data fields will be separated.

Second, you must design the classes described above, choosing names and data types of private variables, and deciding on the interface for all class methods. In particular, what the parameters should be, and what each method returns.

Finally, you need to work out the logic needed to implement the main program and each of the methods in your classes. You may want to work out the steps in your algorithms on paper and test it manually with some simple cases before you worry about special cases.

3. Implementation:

You are starting this programming project with a blank piece of paper, so you will have to implement all of the classes described above on your own. You are welcome to look at source code on the course website to verify the correct syntax for private variables, constructor functions, etc.

As you implement each of the methods listed above it is important to do so incrementally one method at a time, writing comments, adding code, compiling, and debugging. This way, you always have a program that “does something” even if it is not complete.

4. Testing:

Test your program to check that it operates correctly for all of the requirements listed above. See what happens if the user does not follow your instructions, or if they type invalid data. Try your program with several input values, and save your testing output in text files or as screen shots for inclusion in your project report.

Once your program is working properly, you need to demonstrate correctness on the department Linux server `turing.csce.uark.edu`. To do this, you need to copy your source code and data files to `turing` and compile and execute your program while running “script” as shown below.

```
script test1.txt
g++ -Wall *.cpp -o main.exe
./main.exe
(type your program inputs)
exit
```

5. Documentation:

When you have completed your program, write a short report using the “Programming Project Report Template” describing what the project objectives were, what you did, and the status of the program. The report template has a number of point form questions in each section. You should remove these questions from the report and replace them with your answers. Save this project report in a separate document to be submitted electronically.

6. Project Submission:

In this class, we will be using electronic project submission to make sure that all students hand their programming projects and labs on time, and to perform automatic plagiarism analysis of all programs that are submitted.

When you have completed the tasks above, create a folder on your computer called “your_username” and copy all of your C++ source code, input/output files, your Linux testing script, and project report into this folder. Then compress this file folder to create “your_username.zip”. Upload this zip file into Blackboard using the project upload link. We will download and decompress your zip file, and compile and run your code to verify correctness.

The dates on your electronic submission will be used to verify that you met the due date above. All late projects will receive reduced credit:

- 10% off if less than 1 day late,
- 20% off if less than 2 days late,
- 30% off if less than 3 days late,
- no credit if more than 3 days late.

You will receive partial credit for all programs that compile even if they do not meet all program requirements, so handing projects in on time is highly recommended.

7. Academic Honesty Statement:

Students are expected to submit their own work on all programming projects, unless group projects have been explicitly assigned. Students are NOT allowed to distribute code to each other, or copy code from another individual or website. Students ARE allowed to use any materials on the class website, or in the textbook, or ask the instructor and/or GTAs for assistance during class or office hours.

This course will be using highly effective program comparison software to calculate the similarity of all programs to each other, and to homework assignments from previous semesters. Please do not be tempted to plagiarize from another student.

Violations of the policies above will be reported to the Provost's office and may result in a ZERO on the programming project, an F in the class, or suspension from the university, depending on the severity of the violation and any history of prior violations.