C# class: student records application

**Exercise:**

Write a C# program to keep records and perform statistical analysis for a class of 20 students. The information of each student contains ID, Name, Sex, quizzes Scores (2 quizzes per semester), mid-term score, final score, and total score.  
The program will prompt the user to choose the operation of records from a menu as shown below:

====================================================           MENU

========================================================

1. Add student records  
  
2. Delete student records  
  
3. Update student records  
  
4. View all student records  
  
5. Calculate an average of a selected student’s scores  
  
6. Show student who gets the max total score  
  
7. Show student who gets the min total score  
  
8. Find student by ID

9. Sort records by total scores  
  
Enter your choice:1  
  
  
Note: All students records are stored in an array of structures

C# class: student records application

**Menu of choices**

**Step2:** Defining the displaymenu() method to display the menu. The simple menu provides nine choices from 1 to 9 to work with the records.

C# class: student records application

**Append record to list**

**Step3:** defining the add(student[] st, ref int itemcount) method to add a new record to the the array of student objects. This method takes two arguments. The first argument is the array of student objects(st) and the second argument is the number of items in the array. The two arguments are passed by references. For an array, we don't need to use the ref keyword when we want to pass it by reference. However, we need to use the ref keyword when we want to pass an argument of primitive type such as int, float, dobule,etc. When the new item is added the value itemcount variable increases by 1 that means the number of records in the list increases.

C# class: student records application

**Show all records in list**

**Step4:** Defining the viewall(student[] st, int itemcount) method to display the list of all records in the set. To display all records, we need a while loop to traverse through the array of student objects.

C# class: student records application

**Find record index**

**Step5:** Defining the search(student[] st, int itemcount) method to search for the index of a target record. This method is useful as we need it to find the location of the target record in the array of student objects. It can help us to make sure the record does exit before we allow the record for deletion or updating. If the target element is found, the method returns the index of this element. It return -1, if the target element is not found in the array.

C# class: student records application

**Delete record**

**Step6:** Defining the delete(student[] st, ref int itemcount) method to delete a target record from the array of student objects. The user will be prompted to enter the id of student record that his/her want to delete. Then this id will be checked to make sure it does exist in the list. If the target record or element really exists, the deletion process can be made. The deletion process starts by checking whether the target record is the last record, beginning or middle record. If the target record is the last record in the list, we simply delete the record by supplying it to the clean(student[] st, int index) method. The last record is the record that has it index equal to itemcount subtracted by 1. If the target record stays at the beginning or in the middle of the list, we need to use a loop to allow the previous element to take over the next element. This process continue until it reaches the end of the list(itemcount-1). Then the clean() method is called to clean the last element of the list that should not exit. After the element is cleaned, the itemcount variable decreases by 1. This means that the number of elements in the list decreases.

C# class: student records application

**Update record**

**Step7:** Defining the update\_rec(struct student st[], int itemcount) method to update a specified record. The update process starts by asking the user to input the id of the record to be changed. The id value is check to make sure it really exists. If it exits the change to the target record can be made after asking the user to input the new value of the field that need change.

C# class: student records application

**Average score**

**Step8:** Defining the average(student[] st, int itemcount) method to calculate the average score of a selected student. The method alo starts by asking the user to input the id of the target student. This id is checked to make sure it really exist. The average score can be calculated by dividing the sum of quizz1 score, quizz2 score, assignment score, mid-term score, and final score by 5.

C# class: student records application

**Min and Max scores**

**Step9:** Defining the showmax(student[] st, int itemcount) and showmin(student[] st, int itemcount) methods show about the student who gets the maximum score and the student who gets the minimum score. To find the highest total core or lowest total core, we need to compare every total score of each element.