MERRIT LIST AND FULL RESULT IN C

MINI PROJECT ON

BASED ON: FILE I/O AND SINGLE LINKED LIST

**ABSTRACT:**

On this planet, every college needs a firm software which on based on some variable parameter’s can display he merit list based on marks the student has obtained. So, I have developed an app based on C language which reads the source information from a text file and calculates the total marks and their percentage and then based on the Users choice, displays the result.

In this project I have used the concepts of C language which majorly deal with Strings, Numerical data and Sorting process. I have also used the File I/O operations and in order to get the best results and for the program to be user friendly.

**Existing System:**

The process is manual and hand run. The User has to manually enter the Name, Enrollment number, and Marks in each subject, then the software is ready to calculate the percentage and merit list according to the marks. This process is very slow to give the result. Such kind of software’s are easily prone to making errors. When a parameter has to be added to such software, it has to be redesigned, adding more efforts. The more the students, the more tedious the work is.

**Proposed System:**

To overcome the Problems that the Existing system faces I have added the File Reading operation which makes the app versatile to use for the user and makes the user easy to use the app. The program also keeps the record of number of students. Based on the data that has been entered in the text file, the app sorts the data and makes a Merit List. Since the User has to enter the data only once, it makes the app less prone to errors. When the Merit List is prepared, the User can store it in a File. When a parameter has to be added to the system, minimal changes are required. Number of students in not a issue for this system.

**Advantages:**

1. Very fast and accurate.

2. No need of any extra manual effort.

3. With just minimal information user can operate the app.

4. Easy to operate

5. Options available for sorting the data on different parameters.

6. A single User can find merit list of all subject once the marks are entered in the file.

REQUIREMENTS:

1. The User has to input the data into a text file.
2. Data consists of S.No, Name of the student, Marks of 5 subjects with spaces in between after each subjects marks.
3. Run the executable file.
4. To choose a option, a menu appears with all the possible options.
5. If the user wants to store the Merit list, a option is available.
6. After the User has finished, Memory has to be freed to avoid any errors.

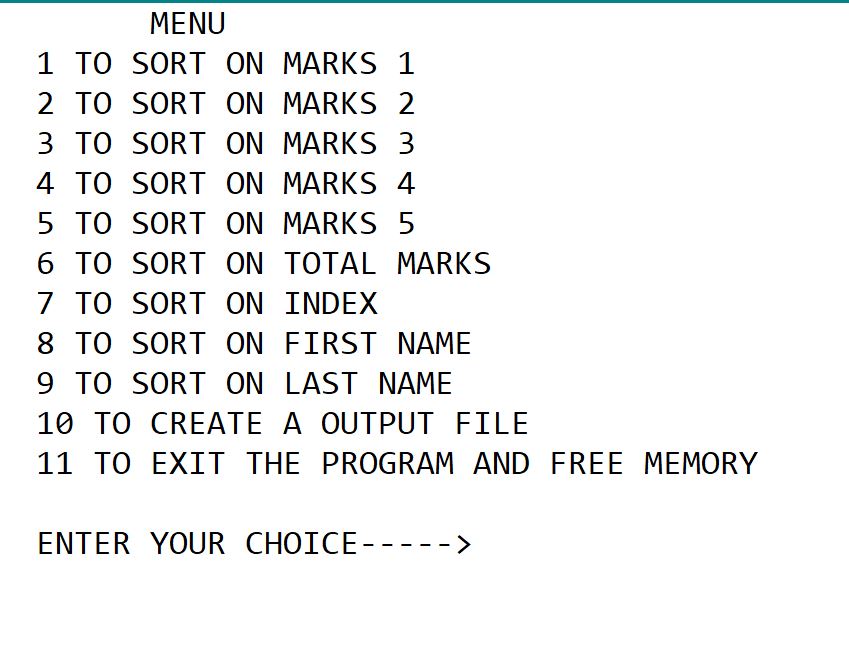
MODULE DESCRIPTION:

As, With a well organized display and systems its better to navigate through the app. So the app has been broken down in some simple functions.

1. MENU
2. CREATE LINKED LIST
3. SORTING
4. DISPLAY
5. CREATE OUTPUT FILE
6. FREE MEMORY AND EXIT

MENU:

There are a lot of apps out there with just one UI. In this application also I’m providing one User Interface which is also known as menu. It is totally based on C language. It has the following options.



2. CREATE LINKED LIST:

In this module we read the data (ie. S.No, Name of the student and marks of 5 subjects) from a (.txt) file.

The text file has the following format.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Index | First name | Middle name | Last name | Marks 1 | Marks 2 | Marks 3 | Marks 4 | Marks 5 |
| 1 | Abhijeet | Sanjiv | Bonde | 21 | 23 | 34 | 56 | 87 |
| 2 | Damon | Joseph | Salvatore | 43 | 56 | 776 | 89 | 09 |
| 3 | Stefan | Joseph | Salvatore | 27 | 54 | 56 | 52 | 56 |
| …… | ……. | …… | …… | …… | …… | …… | …… | …… |
| 99 | Elena | David | Gilbert | 85 | 59 | 32 | 58 | 65 |
| 100 | Hemangi | Kunal | Sharma | 34 | 65 | 82 | 45 | 98 |

Then a structure is declared as follows.

struct list

{

int index;

char f\_name[20];

char m\_name[20];

char l\_name[20];

int m1;

int m2;

int m3;

int m4;

int m5;

int total\_marks;

float percentage;

struct list \*next;

};

The text file is then opened using FILE I/O. Once the file is open then the data from the file is read and then stored in the structure which has been shown above. Then the nodes of the structure are linked and thus the linked list is created by using the structure. Once the linked list has been created then the file is closed for further simplification. At the start of this module, the module asks the user for the file name of the input file ( as shown in Fig 1.1 ). If the program is not able to find a file with the given name it will show an error ( as shown in Fig 1.2 ) So, the input file is user desired. Once the file is open the program creates a linked list with structure.

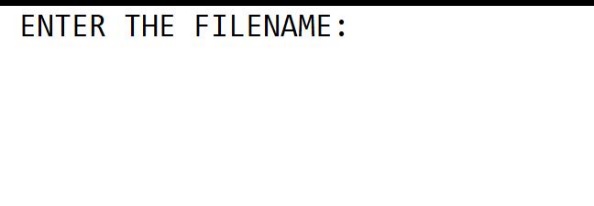
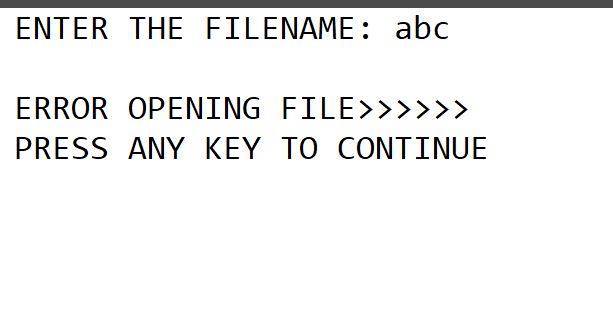


Fig 1.1 Fig 1.2

SORTING:

In this process the createdlinked list is sorted accrding to the the users input. User has options whether to sort on marks 1, marks 2, marks 3, marks 4, marks 5, total marks, First name, Last name, and index. At this point the linked list is not sorted. In this process I have’nt sorted the linked list but I will sort the index. Sorting in here is of 2 types:

1. Sorting based on Integer
2. Sorting based on String

The process of sorting based on Integer is as follows.

Let us assume that the user has selected the option to sort on marks 1. For the sorting to be done I have created two arrays (ie. Index and marks). The sorting is done by comparing the elements in the marks array. From the linked list that has been created before, the data has been copied in these the marks array. From the field m1 in the linked list I have copied the data to the marks array. Once all the marks are in the marks array, the number of elements in the marks array are counted and then the sorting process starts. I have applied the Bubble sort to sort the index array. The sorting code is as follows.

for( j=0; j<(no\_of\_elements-i-1); j++)

{

if(marks[j]<marks[j+1])

{

temp=marks[j];

marks[j]=marks[j+1];

marks[j+1]=temp;

temp=index[j];

index[j]=index[j+1];

index[j+1]=temp;

}

}

The sorting is done by comparing the (j)th and (j+1)th elements from the marks array. If the (j)th element is smaller than (j+1)th element then (j)th and (j+1)th elements are interchanged and simultaneously the (j)th and (j+1)th elements are also interchanged, as it will be required for displaying the List (Note that I have made no change in the linked list that was created before or the file data. They remain the same as they were when created). Once the index is modified in the index is then reversed as we want the array in ascending order, not descending. The new modified index is then passed to the display function for the further process.

The process of sorting on string is as follows.

At first, a loop counts the number of elements in the list. A dummy index is created with its each element being (i+1) where I is the current in the index array. Once the array is filled the strings (ie. The first name or the last name) are compared with the help of ***strcmp()*** function (as the function returns, a negative value if the first string precedes the second string alphabetically, a value of zero if the first string and the second string are identical disregarding the case, a positive value if the second string precedes the first string alphabetically). If the output of the ***strcmp()*** function is positive then the elements are interchanged and simultaneously are the index values changed. At last the index is returned from the sort routine.

DISPLAY:

In this module the list is printed on the screen as per the index passed to it. In our case the index passed is sorted in ascending order. There are 2 parameters passed to this module (ie. Starting address of the Linked list and the index). With the index number that comes first, the pointer traverses through the linked list to print the record with the associated index. After printing each record, the pointer is again initialized to starting position of the linked list.