PRACTICAL ASSIGNMENT 1

Github -https://github.com/Atharva-byte1/DSA[)](https://github.com/Aditya6768/DSAL)

Consider a student database of SEIT class (at least 15 records). Database contains different fields of

every student like Roll No, Name and SGPA.(array of structure)

A. Design a roll call list, arrange list of students according to roll numbers in ascending order (Use

Bubble Sort)

B. Arrange a list of students alphabetically. (Use Insertion sort)

C. Arrange a list of students to find the first ten toppers from a class. (Use Quick sort) D. Search students according to SGPA. If more than one student has the same SGPA, then print a

list of all students having the same SGPA.

E. Search a particular student according to name using binary search without recursion.

#include<iostream>

using namespace std;

#include<string.h>

int const size=15;

struct student{

int rollNo;

char name[20];

float sgpa;

};

void accept(struct student list[size]);

void display(struct student list[size]);

void displayq(struct student list[size]);

void bubbleSort(struct student list[size]);

void search(struct student list[size]);

void insertionSort(struct student list[size]);

void binarySearch(struct student list[size]);

void quickSort(struct student list[size],int l,int r);

main(){

struct student data[20];

accept(data);

int ch;

do{

cout<<"\n"<<"1. Bubble Sort";

cout<<"\n"<<"2. Search";

cout<<"\n"<<"3. Insertion Sort";

cout<<"\n"<<"4. Binary Search ";

cout<<"\n"<<"5. Quick sort ";

cout<<"\n"<<"6. Exit ";

cout<<"\n"<<"Enter choice:";

cin>>ch;

switch (ch){

case 1:

bubbleSort( data);

display( data);

break;

case 2:

search(data);

break;

case 3:

insertionSort(data);

display( data);

break;

case 4:

binarySearch(data);

break;

case 5:

quickSort(data,0,size-1);

displayq( data);

break;

case 6:

cout<<"Exit"<<endl;

break;

default:

cout<<"Invalid choice"<<endl;

}

}

while(ch!=6);

return 0;

}

void quickSort(struct student list[size],int l,int r){

if(l<r){

float pi=list[r].sgpa;

int i = l-1;

for(int j=l;j<=r-1;j++){

if(list[j].sgpa>=pi){

i++;

swap(list[i],list[j]);

}

}

swap(list[i+1],list[r]);

int partition =i+1;

quickSort(list,l,partition-1);

quickSort(list,partition+1,r);

}

}

void binarySearch(struct student list[size]){

int k,lower,mid,upper;

char search[80];

cout<<"Enter name :";

cin>>search;

lower=0;

upper=size-1;

mid=(lower+upper)/2;

while(lower<=upper){

if(strcmp(list[mid].name,search)<0){

lower=mid+1;

}

else if(strcmp(list[mid].name,search)==0){

cout<<"\n"<<list[mid].rollNo<<"\t"<<list[mid].name<<"\t"<<list[mid].sgpa; break;

}

else{

upper=mid-1;

mid=(lower+upper)/2;

}

if(lower>upper){

cout<<"Invalid";

}

}

}

void insertionSort(struct student list[size]){

int k,j;

struct student temp;

for (int k=0; k<size;k++){

temp = list[k];

j=k-1;

while(strcmp(list[j].name,temp.name)>0 && j>=0){

list[j+1]=list[j];

j--;

}

list[j+1]=temp;

}

}

void search(struct student list[size]){

int i;

float SGPA;

cout<<"\n"<<"Enter SGPA : ";

cin>>SGPA;

for(int i=0;i<size;i++){

if (SGPA==list[i].sgpa){

cout<<"\n"<<list[i].rollNo<<"\t"<<list[i].name<<"\t"<<list[i].sgpa;

}

}

}

void bubbleSort(struct student list[size]){

struct student temp;

for(int i=0;i<size-1;i++){

for(int j=0;j<size-1-i;j++){

if (list[j].rollNo>list[j+1].rollNo){

temp =list[j];

list[j]=list[j+1];

list[j+1]=temp;

}

}

}

}

void displayq(struct student list[size]){

int i;

cout<<"\n"<<"Roll no"<<"\t"<<"Name"<<"\t"<<"SGPA";

for (int i=0;i<10;i++){

cout<<"\n"<<list[i].rollNo<<"\t"<<list[i].name<<"\t"<<list[i].sgpa;

}

}

void display(struct student list[size]){

int i;

cout<<"\n"<<"Roll no"<<"\t"<<"Name"<<"\t"<<"SGPA";

for (int i=0;i<size;i++){

cout<<"\n"<<list[i].rollNo<<"\t"<<list[i].name<<"\t"<<list[i].sgpa;

}

}

void accept(struct student list[size]){

int i;

for (int i=0;i<size;i++){

cout<<"Enter Roll no ::";

cin>>list[i].rollNo;

cout<<"Enter Name ::";

cin>>list[i].name;

cout<<"Enter SGPA ::";

cin>>list[i].sgpa;

}

}

OUTPUTS





