|  |  |
| --- | --- |
| **Name** | Mahadev Balla |
| **UID no.** | 2023300010 |
| **Experiment No.** | 8 |

|  |  |
| --- | --- |
| **AIM:** | To solve given problems using structures/unions. |
| **Program 1** | |
| **PROBLEM STATEMENT :** | Team manager:  As a team manager, you want to pick the best batsman from your IPL team for upcoming match. Your system collect data of each player as his name, country and batting average. Then it sorts the list based on batting average and produces the output to you, so you can select best batsman. [Take input of at least 10 players] |
| **ALGORITHM:** | 1. Start   2) Define the structure:  - Create a structure named `team` with three members: `name` (array of characters with size 100), `country` (array of characters with size 20), and `avg` (integer).  3) Define the `sort` function:  - Input: `struct team m[10]` (an array of team structures), `int size` (size of the array).  - Iterate over the array from 0 to `size - 1`.  - Inside the outer loop, initialize `max` with the `avg` value of the current element (`m[j].avg`) and set `index` to the current index (`j`).  - Iterate over the array from the current index (`j`) to `size - 1` to find the maximum `avg` value and its index.  - Swap the `avg` values, `name`, and `country` strings of the current element (`m[j]`) and the element with the maximum `avg` value (`m[index]`) if they are not the same.  4) Define the `print` function:  - Input: `struct team tm[10]` (an array of team structures), `int size` (size of the array).  - Iterate over the array from 0 to `size - 1`.  - Print the player number, name, country, and average for each element in the array.  5) In the `main` function:  - Declare an array of team structures `tm` with a size of 10.  - Use a loop to input player details (name, country, and average) for each player in the array.  - Call the `sort` function, passing the `tm` array and its size (10) as arguments, to sort the array based on average scores.  - Call the `print` function, passing the sorted `tm` array and its size (10) as arguments, to print the sorted player details.   1. End |
| **PROGRAM:** | #include <stdio.h>  #include <string.h>  struct team  {  char name[100];  char country [20];  int avg;  } ;  void sort(struct team m[10],int size)  {  int index,max,temp;  char tempname[100],tempcountry[20];  for(int j=0;j<size;j++)  {  max=m[j].avg;  for(int i=j;i<size;i++) //finding max  {  if(max<m[i].avg)  {  max=m[i].avg;  index=i;  }  }    temp=m[j].avg; //swapping element  strcpy(tempname,m[j].name);  strcpy(tempcountry,m[j].country);  if(max!=temp) //cuz index remains same as previous if max=temp  {  m[j].avg=max;  m[index].avg=temp;  strcpy(m[j].name,m[index].name);  strcpy(m[index].name,tempname);  strcpy(m[j].country,m[index].country);  strcpy(m[index].country,tempcountry);  }  }  }  void print(struct team tm[10],int size)  {  for(int t=0;t<size;t++)  {  printf("Player %d:\n%s %s %d\n",t+1,tm[t].name,tm[t].country,tm[t].avg);  }  }  int main()  {  struct team tm[10];  for(int i=0;i<10;i++)  {  printf("\n%d Player\nEnter Player name: ",i+1);  scanf("%s",tm[i].name);  printf("\nEnter Country: ");  scanf("%s",tm[i].country);  printf("\nEnter batting average: ");  scanf("%d",&tm[i].avg);    }    sort(tm,10); //calling functions  print(tm,10);      return 0;  } |
| **RESULT:** | |
| **Program 2** | |
| **PROBLEM STATEMENT :** | Flight management system:  An airline reservation system maintains records for possible flights consisting of  STARTING POINT 3 character code  DESTINATION 3 character code  STARTING TIME integer on scale 0001 – 2400  ARRIVAL TIME integer on scale 0001 – 2400  SEATS positive integer in suitable range.  Your program is to read 10 such records followed by queries of the form STARTING  POINT– DESTINATION, one to a line. For each query find whether there is a possible flight with a seat available; if so reduce the number of seats by one and print out the flight details (or an apology). |
| **ALGORITHM:** | 1. Start   2) Define the structure:  - Create a structure named `flight` with four members: `start` (array of characters with size 4 for the starting destination code), `dest` (array of characters with size 4 for the final destination code), `starttime` (integer for departure time), `arrivaltime` (integer for arrival time), and `seats` (integer for the number of available seats).  3) In the `main` function:  - Declare an array of `flight` structures `fl` with a size of 10.  - Use a loop to input flight details (starting destination, final destination, departure time, arrival time, and number of seats) for each flight in the array.  4) Implement the menu-driven program:  - Declare variables `strt[4]` and `dst[4]` to store the input for boarding point and final destination.  - Declare variables `ans`, `flag`, and `ans2` to control the flow of the program.  - Use a do-while loop with condition `ans == 1` to repeatedly execute the following steps:  - Initialize `flag` to 1.  - Prompt the user to enter the boarding point and final destination.  - Iterate over the `fl` array to find a flight matching the input boarding point and final destination.  - If a matching flight is found and has available seats:  - Set `flag` to 0.  - Print flight details (departure time, arrival time, and available seats).  - Ask the user if they want to confirm the booking (`ans2` variable).  - If the user confirms the booking (`ans2 == 1`), print booking details and decrement the number of available seats for the selected flight.  - If no matching flight is found or if the selected flight has no available seats, print an appropriate message.  - Ask the user if they want to check for more flights (`ans` variable).  - Continue the loop until the user chooses not to check for more flights (`ans != 1`).  5) End of the program:  - Print a message indicating the end of the program execution.   1. End |
| **PROGRAM:** | #include <stdio.h>  #include <string.h>  struct flight  {  char start[4];  char dest[4];  int starttime;  int arrivaltime;  int seats;  };  int main()  {  struct flight fl[10];  for(int i=0;i<10;i++)  {  printf("Enter starting destination: ");  scanf("%s",fl[i].start);  printf("Enter final destination: ");  scanf("%s",fl[i].dest);  printf("Enter departing time: ");  scanf("%d",&fl[i].starttime);  printf("Enter arrival time: ");  scanf("%d",&fl[i].arrivaltime);  printf("Enter number of Seat: ");  scanf("%d",&fl[i].seats);  printf("\n");  }    printf("\nMenu driven program starts");  char strt[4],dst[4];  int ans=1,flag=1,ans2=0;  do  {  flag=1;  printf("\nEnter Boarding point: ");  scanf("%s",strt);  printf("Enter final destination: ");  scanf("%s",dst);    for(int i=0;i<10;i++)  {  if(strcmp(strt,fl[i].start)==0 && strcmp(dst,fl[i].dest)==0)  {  if(fl[i].seats!=0)  {  flag=0;  printf("\nFlight available!!");  printf("\nDeparture time: %d\nArrival time: %d",fl[i].starttime,fl[i].arrivaltime);  printf("\nSeats available: %d",fl[i].seats);  printf("\nwant to confirm booking(1 for yes): ");  scanf("%d",&ans2);  if(ans2==1)  {  printf("\nFlight booked!!");  printf("\nBoarding point: %s",strt);  printf("\nDestination: %s",dst);  printf("\nDeparture time: %d\nArrival time: %d",fl[i].starttime,fl[i].arrivaltime);  printf("\nSeats available: %d",--fl[i].seats);  break;  }  }  else  {  printf("Sorry Seat not availabe!!");  }  }  }  if(flag==1)  {  printf("\nFlight not availabe!!");  }  printf("\nDo you want to check for more flight(press 1 for yes): ");  scanf("%d",&ans);    }while (ans==1);    return 0;  } |
| **RESULT:**  airline1airline2 | |
| **Program 3** | |
| **PROBLEM STATEMENT:** | Employee management system:  A record in an organizations payroll consists of one line for each employee consisting of:  NAME (20 characters), GENDER (1 character M or F), SALARY (integer), DATE OF BIRTH (3 integers YEAR MONTH DAY).  Write a program which will input 10 such records. Your program must then take in 5  amendments in the record set which will be in the same form as the record structure  itself. The amendments can contain new employees to be added (name different from  existing ones), employees left (salary given as 0) and update of salary(more or less). Your program must then incorporate these amendments and also remove those employees who have reached retirement age(Age 60). |
| **ALGORITHM:** | 1. Start   2) Define the structure:  - Create a structure named `employee` with five members: `name` (array of characters with size 21), `gender` (array of characters with size 2), `salary` (integer), `year` (integer for birth year), `month` (integer for birth month), and `date` (integer for birth date).  3) Implement the `input` function:  - Input: `struct employee e[15]` (an array of employee structures), `int start` (starting index), `int stop` (ending index).  - Use a loop to input employee details for the specified range of indices.   1. Implement the `print` function:   - Input: `struct employee e[15]` (an array of employee structures), `int start` (starting index), `int stop` (ending index).  - Use a loop to print the details of employees within the specified range of indices.  5) Implement the `delete` function:  - Input: `struct employee e[15]` (an array of employee structures), `int size` (current size of the array), `int index` (index of the employee to be deleted).  - Use a loop to shift elements from the index to `size - 1` one position left, effectively deleting the specified employee.  - Return the updated size of the array.  6) In the `main` function:  - Declare an array of `employee` structures `e` with a size of 16.  - Initialize the `size` variable to 10.  - Use a loop to input employee details for the first 10 employees.  - Print the input records.  - Implement a menu-driven program with options to add new employees, delete employee details, and update salaries.  - After each operation, print the updated records.  - After the menu-driven operations, iterate through the array and delete records of employees who have retired (age > 59) or have a salary of 0.  - Print the final records after deleting details of retired and zero-salary employees.  7) End of the program:  - Print a message indicating the end of the program execution.  8) End |
| **PROGRAM:** | #include <stdio.h>  #include <string.h>  struct employee  {  char name[21];  char gender[2];  int salary,year,month,date;  };  void input(struct employee e[15],int start,int stop)  {  for(int i=start;i<stop;i++)  {  printf("\nEnter employee name: ");  scanf("%s",e[i].name);  printf("Enter gender(M/F): ");  scanf("%s",e[i].gender);  printf("Enter salary: ");  scanf("%d",&e[i].salary);  printf("Enter birth year: ");  scanf("%d",&e[i].year);  printf("Enter birth month: ");  scanf("%d",&e[i].month);  printf("Enter birth date(eg-24): ");  scanf("%d",&e[i].date);  }  }  void print(struct employee e[15],int start,int stop)  {  for(int i=start;i<stop;i++)  {  printf("Enter details of employee %d",i+1);  printf("\nName: %s\nGender: %s\nSalary: %d\nDate of birth: %d/%d/%d\n",e[i].name,e[i].gender,e[i].salary,e[i].date,e[i].month,e[i].year);  }  }  int delete(struct employee e[15],int size,int index)  {  for (int i=index;i<size-1;i++)  {  strcpy(e[i].name,e[i+1].name);  strcpy(e[i].gender,e[i+1].gender);  e[i].salary=e[i+1].salary;  e[i].year=e[i+1].year;  e[i].month=e[i+1].month;  e[i].date=e[i+1].date;  }  return (--size);  }  int main()  {  struct employee e[16];  int size=10;  printf("\nInput records\n");  input(e,0,size);    print(e,0,size);  int choice,flag;  char name[21];  for(int j=0;j<5;j++)  {  printf("\n1: Add new employee details\n2: Delete employee details");  printf("\n3: Update salary\nEnter One choice from above: ");    scanf("%d",&choice);    switch(choice)  {  case 1:  input(e,size,size+1);  size++;  printf("\nRecord after adding new employee:-\n");  print(e,0,size);  break;    case 2:  flag=1;  int index;  printf("\nEnter employee name: ");  scanf("%s",name);  for(int j=0;j<size;j++)  {  if(strcmp(name,e[j].name)==0)  {  printf("\nRecord that is going to be deleted:-\n");  print(e,j,j+1);  printf("\n\n");  size=delete(e,size,j);  printf("\nRecord after deleting record of employee:-\n");  print(e,0,size);  flag=0;  break;  }  }  if(flag==1)  {  printf("\nEmployee not found!");  }  break;    case 3:  flag=1;  int sal;  printf("\nEnter employee name: ");  scanf("%s",name);  for(int j=0;j<size;j++)  {  if(strcmp(name,e[j].name)==0)  {  printf("\nSalary is: %d\nEnter new salary: ",e[j].salary);  scanf("%d",&sal);  e[j].salary=sal;  printf("\nRecord after updating salary:-\n");  print(e,0,size);  flag=0;  break;  }  }  if(flag==1)  {  printf("\nEmployee not found!");  }    break;    default:  printf("\nInvalid input");    }    }  int t=size;  for(int i=t-1;i>0;i--)  {  if(2023-e[i].year>59)  {  printf("\nDeleting record because crossed retirment age(i.e.60):-");  print(e,i,i+1);  size=delete(e,size,i);  }  else if(e[i].salary==0)  {  printf("\nDeleting record because salary=0:-");  print(e,i,i+1);  size=delete(e,size,i);  }  }  printf("\n\n\nRecord after Deleting details of retired people:-\n");  print(e,0,size);    return 0;  } |
| **RESULT:** Screenshot (61)Screenshot (62)Screenshot (63)Screenshot (64)Screenshot (65)Screenshot (66)Screenshot (67)Screenshot (68) | |
| **CONCLUSION:** | Studied the application of structures/unions to solve given problems. |