Vishwakarma Institute Of Information Technology

Department Of Computer Engineering

- **Name** : Abhinav Belhekar
- **Roll No.** : 224033
- **PRN** : 22010389
- **Division** : SY – D (Computer Department)
- **Batch** : D2
- **Subject** : Fundamentals Of Data Structure

Assignment - 02

❖ **Aim :** Create a database using array of structures and perform the following operations on it :

1. Add Record
2. Display Database
3. Search Record (Binary Search)
4. Delete Record

❖ **Algorithm :**

Add Record

1. Take the employee Structure
2. Check if the structure array is completely filled or not
3. If yes then return
4. If no then pass the structure to function ***getPerfectPosition,*** which will return the position where we will put the structure so that array will always sorted.
5. Once we get the position we will pass the structure to ***shiftPlaceAndInsertForward***, which will place it to given position.
6. Increment the element count counter.

Display Database

1. Start a loop from 0 to element count
2. Print details of each element of array

Search Record (Binary Search)

1. Take an structure array, start, end, key
2. Check if end ≥ start
3. If no then return -1
4. If yes then find the mid = end +start /2
5. If arr[mid] == key then return mid
6. Else if arr[mid] < key then return ***BinarySearch(arr, start, mid-1, key)***
7. Else return ***BinarySearch(arr,mid+1, end, key)***

Delete Record

1. Take the employee Id which you want to delete
2. Get the position of employee with given id by performing binary search
3. If the position is not equal to -1 then perform ***shiftEntriedBackward***
4. Else return

getPerfectPosition

1. Accept struct employee
2. Set pos = 0
3. Is the pos < element count ?
4. If no then stop the loop and return pos.
5. Else check if the given employee id is less than the id of pointed employee
6. If yes then break and return pos
7. Else increment pos go to step 3

shiftPlaceAndInsetForward

1. Accept the employee array, position to insert (pos), element count, new entry
2. Declare and initialize employee prev, next
3. Assign prev to arr[pos]
4. Assign arr[pos] to given employee, increment pos.
5. Is pos ≤ element count ?
6. If no then assign arr[pos] to next, then put prev to arr[pos] and the swap next, prev
7. Go to step 5
8. If yes then exit

❖ Project Structure:

```
main.cpp
utils.h
/database
        employee.h
        database.h
        utils.h
```

❖ Code:

main.cpp

```cpp
#include <iostream>
#include "database/database.h"
#include "database/employee.h"
#include "utils.h"
```

```

using namespace std;

void UI(int, database);

int main()
{
    cout << "\nEMPLOYEE DATABASE";
    int database_size;
    cout << "\nENTER THE DATABASE\n>>> ";
    cin >> database_size;
    database db(database_size);
    cout << "\nDATABASE CREATED SUCESSFULLY\n";
    int input = -1;
    while (input)
    {
        if (input == 1)
        {
            Employee temp = createEmployee();
            db.addEmployeeEntry(temp);
        }
        else if (input == 2)
        {
            db.displayDatabse();
        }
        else if (input == 3)
        {
            cout << "\nENTER THE ID\n>>> ";
            int id;
            cin >> id;
            int loc = db.getElementIndexByID(1);
            if (loc != -1)
            {
                Employee temp = db.getEmployeeByID(loc);
                printThisEmployeeDetail(temp);
            }
            else
            {
                cout << "\n!!! GIVEN ID DOES NOT EXIST";
            }
        }
        else if (input == 4)
        {
            cout << "\nENTER THE EMPLOYEE ID YOU WANT TO DELETE\n>>> ";
            int id;
            cin >> id;
            int loc = db.getElementIndexByID(id);
            if (loc != -1)
            {
                db.deleteEmployeeEntry(loc);
            }
            else
            {
                cout << "\n!!! GIVEN ID DOES NOT EXIST";
            }
        }
        input = displayOptions();
    }

    return 0;
}
```

utils.h

```
#include <iostream>
#include "database/employee.h"
using namespace std;

int displayOptions()
{
    int option;
    cout<<"\nOPTIONS";
    cout<<"\n1 -> ADD A NEW ENTRY";
    cout<<"\n2 -> DISPLAY DATABASE";
    cout<<"\n3 -> SEARCH RECORD";
    cout<<"\n4 -> DELETE RECORD";
    cout<<"\n0 -> EXIT SYSTEM";
    cout<<"\n>>> ";
    cin>>option;
    return option;
}


Employee createEmployee()
{
    Employee e;
    cout<<"\nEMPLOYEE";
    cout<<"\nID      :\n>>> ";
    cin>>e.id;
    cout<<"\nNAME    :\n>>> ";
    cin>>e.name;
    cout<<"\nSALARY  :\n>>> ";
    cin>>e.salary;
    return e;
}

void printThisEmployeeDetail(Employee e)
{
    cout<<"\nEMPLOYEE DETAILS";
    cout<<"\nID      : "<<e.id;
    cout<<"\nNAME    : "<<e.name;
    cout<<"\nSALARY  : "<<e.salary;
}
```

database/employee.h

```cpp
#pragma once
#include <iostream>
#include <climits>
struct Employee{
    int id = INT_MAX;
    double salary;
    std::string name;
};

```

database/database.h

```cpp
#pragma once
#include <iostream>
#include <iomanip>
#include "employee.h"
#include "utils.h"
using namespace std;

class database
{
private:
    int pointer = 0;

public:
    Employee *employee_list;
    int max_size;
    int elements = 0;

    database(int);
    void addEmployeeEntry(Employee);
    void displayDatabse();
    int getElementIndexByID(int emloyee_id);
    Employee getEmployeeByID(int index);
    void deleteEmployeeEntry(int index);
};

database::database(int max_size)
{
    this->employee_list = new Employee[max_size];
    this->max_size = max_size;
}

void database::addEmployeeEntry(Employee e)
{
    /*
    The employee id is dominant for comparisons
    */

    // checks if limit is exceeded
    if (elements == max_size)
    {
        cout << "\n!!! DATABASE IS FULL !!! ... Try deleting some entries\n";
        return;
    }

    // getting the target position
    int target_pos = getPerfectPosition(employee_list, elements, e);

    // check if element with same id exists or not
    if (this->employee_list[target_pos - 1].id != e.id)
    {
        // if not then adds a new entry
        shiftPlaceAndInsertForward(target_pos, elements, employee_list, e);
        this->elements++;
    }
    else
    {
        // if yes then merges the entry
        this->employee_list[target_pos - 1] = e;
    }

    this->pointer++;
}

void database::displayDatabse()
{
    cout << left << std::setfill(' ') << std::setw(5) << "ID" << std::setw(15) << "NAME" <<  std::setw(10) << "SALARY"
         << "\n";
    for (int i = 0; i < this->elements; i++)
    {
        Employee temp = this->employee_list[i];
        cout << setfill(' ') << setw(5) << temp.id << setw(15) << temp.name << std::setw(10) << temp.salary << "\n";
    }
}

int database::getElementIndexByID(int employee_id)
{
    return binarySearch(employee_list, 0, elements, employee_id);
}

Employee database::getEmployeeByID(int index)
{
    // returns employee object at given index
    return this->employee_list[index];
}

void database::deleteEmployeeEntry(int index)
{
    if (index == this->elements - 1)
    {
    }
    else
    {
        shiftEntriesBackward(index, elements, employee_list);
    }
    elements--;
}
```

database/utils.h

```cpp

#include <iostream>
#include "employee.h"

int binarySearch(Employee arr[], int start, int end, int key)
{
    if(end>=start)
    {
        int mid = (start+end)/2;
        if(arr[mid].id == key)
        {
            return mid;
        }
        else if(arr[mid].id < key)
        {
            return binarySearch(arr, mid+1, end, key);
        }
        else{
            return binarySearch(arr, start, mid-1, key);
        }

    }
    return -1;
}

int getPerfectPosition(Employee arr[], int total_elements, Employee e)
{
    int pos = 0;
    while(pos < total_elements)
    {
        if(e.id >= arr[pos].id)
        {
            pos++;
        }
        else
        {
            break;
        }
    }
    return pos;
}

```

```
void shiftPlaceAndInsertForward(int target_pos, int total_elements, Employee arr[],Employee e){
    Employee prev = arr[target_pos];
    Employee next;
    arr[target_pos] = e;
    int pos = target_pos+1;
    while (pos <= total_elements)
    {
        next = arr[pos];
        arr[pos] = prev;
        prev = next;
        pos++;
    }

}


void shiftEntriesBackward(int target_pos, int total_elements, Employee arr[])
{

    Employee cur = arr[total_elements-1];
    Employee next;

    int pos = total_elements-1;

    while(target_pos <= pos)
    {
        next = arr[pos];
        arr[pos] = cur;
        cur = next;
        pos--;

    }
}
```

❖ Output:

```
PS G:\viit notes 2020-21\Sem
III SY\Fundamentals Of Data
Structure\Practical\Assignment_
2> ./main

EMPLOYEE DATABASE
ENTER THE DATABASE
>>> 10

DATABASE CREATED SUCESSFULLY

OPTIONS
1 -> ADD A NEW ENTRY
2 -> DISPLAY DATABASE
3 -> SEARCH RECORD
4 -> DELETE RECORD
0 -> EXIT SYSTEM
>>> 1

EMPLOYEE
ID       :
>>> 3

NAME     :
>>> NAME03

SALARY   :
>>> 50000

OPTIONS
1 -> ADD A NEW ENTRY
2 -> DISPLAY DATABASE
3 -> SEARCH RECORD
4 -> DELETE RECORD
0 -> EXIT SYSTEM
>>> 1

EMPLOYEE
ID       :
>>> 1

NAME     :
>>> NAME01

SALARY   :
>>> 25000

OPTIONS
1 -> ADD A NEW ENTRY
2 -> DISPLAY DATABASE
3 -> SEARCH RECORD
4 -> DELETE RECORD
0 -> EXIT SYSTEM
>>> 1

EMPLOYEE
ID       :
>>> 4

NAME     :
>>> NAME05

SALARY   :
>>> 60000

OPTIONS
1 -> ADD A NEW ENTRY
2 -> DISPLAY DATABASE
3 -> SEARCH RECORD
4 -> DELETE RECORD
0 -> EXIT SYSTEM
>>> 2
ID   NAME           SALARY
1    NAME01         25000
3    NAME03         50000
4    NAME05         60000

OPTIONS
1 -> ADD A NEW ENTRY
2 -> DISPLAY DATABASE
3 -> SEARCH RECORD
4 -> DELETE RECORD
0 -> EXIT SYSTEM
>>> 3

ENTER THE ID
>>> 1

EMPLOYEE DETAILS
ID       : 1
NAME     : NAME01
SALARY   : 25000
OPTIONS
1 -> ADD A NEW ENTRY
2 -> DISPLAY DATABASE
3 -> SEARCH RECORD
4 -> DELETE RECORD
0 -> EXIT SYSTEM
>>> 3

ENTER THE ID
>>> 2

EMPLOYEE DETAILS
```

```
ID       : 1
NAME     : NAME01
SALARY   : 25000
OPTIONS
1 -> ADD A NEW ENTRY
2 -> DISPLAY DATABASE
3 -> SEARCH RECORD
4 -> DELETE RECORD
0 -> EXIT SYSTEM
>>> 4

ENTER THE EMPLOYEE ID YOU WANT
TO DELETE
>>> 1

OPTIONS
1 -> ADD A NEW ENTRY
2 -> DISPLAY DATABASE
3 -> SEARCH RECORD
4 -> DELETE RECORD
0 -> EXIT SYSTEM
>>> 2
ID    NAME           SALARY
3     NAME03         50000
4     NAME05         60000

OPTIONS
1 -> ADD A NEW ENTRY
2 -> DISPLAY DATABASE
3 -> SEARCH RECORD
4 -> DELETE RECORD
0 -> EXIT SYSTEM
>>> 0
```

❖ **Conclusion:**

- Used structure array to store data.
- Used binary search algorithm to search records.