Program for hashing using linear probing

#include<stdio.h>

#include<stdlib.h>

#define LIMIT 30

enum record\_status {EMPTY, DELETED, OCCUPIED};

struct Employee

{

int employee\_id, employee\_age;

char employee\_name[30];

};

struct Record

{

struct Employee info;

enum record\_status status;

};

int hash\_function(int key)

{

return (key % LIMIT);

}

int search\_records(int key, struct Record hash\_table[])

{

int count, temp, position;

temp = hash\_function(key);

position = temp;

for(count = 1; count != LIMIT - 1; count++)

{

if(hash\_table[position].status == EMPTY)

{

return -1;

}

if(hash\_table[position].info.employee\_id == key)

{

return position;

}

position = (temp + count) % LIMIT;

}

return -1;

}

void insert\_records(struct Employee emprec, struct Record hash\_table[])

{

int count, position, temp;

int key = emprec.employee\_id;

temp = hash\_function(key);

position = temp;

for(count = 1; count != LIMIT - 1; count++)

{

if(hash\_table[position].status == EMPTY || hash\_table[position].status == DELETED)

{

hash\_table[position].info = emprec;

hash\_table[position].status = OCCUPIED;

printf("\nRecord Inserted into Hash Table\n");

return;

}

if(hash\_table[position].info.employee\_id == key)

{

printf("\nDuplicate Record cannot be Inserted\n");

return;

}

position = (temp + count) % LIMIT;

}

printf("\nHash Table Limit Exceeded\n");

}

void display\_records(struct Record hash\_table[])

{

int count;

printf("\nHash Table\n");

for(count = 0; count < LIMIT; count++)

{

printf("[%d]:\t", count);

if(hash\_table[count].status == OCCUPIED)

{

printf("Occupied - ID: %d Name: %s Age: %d",hash\_table[count].info.employee\_id, hash\_table[count].info.employee\_name, hash\_table[count].info.employee\_age);

}

else if(hash\_table[count].status == DELETED)

{

printf("\nRecord is Deleted\n");

}

else

{

printf("\nHash Table is Empty\n");

}

}

}

void delete\_records(int key, struct Record hash\_table[])

{

int position = search\_records(key, hash\_table);

if(position == -1)

{

printf("\nKey Not Found\n");

}

else

{

hash\_table[position].status = DELETED;

}

}

int main()

{

int count, key, option;

struct Record hash\_table[LIMIT];

struct Employee emprec;

for(count = 0; count <= LIMIT - 1; count++)

{

hash\_table[count].status = EMPTY;

}

while(1)

{

printf("1. Insert a Record\n");

printf("2. Delete a Record\n");

printf("3. Search a Record\n");

printf("4. Display All Records\n");

printf("5. Exit\n");

printf("Enter Your Option:\t");

scanf("%d", &option);

switch(option)

{

case 1: printf("\nEnter Employee ID:\t");

scanf("%d", &emprec.employee\_id);

printf("Enter Employee Name:\t");

scanf("%s", emprec.employee\_name);

printf("Enter Employee Age:\t");

scanf("%d", &emprec.employee\_age);

insert\_records(emprec, hash\_table);

break;

case 2: printf("\nEnter the Key to Delete:\t");

scanf("%d", &key);

delete\_records(key, hash\_table);

break;

case 3: printf("\nEnter the Key to Search:\t");

scanf("%d", &key);

count = search\_records(key, hash\_table);

if(count == -1)

{

printf("\nRecord Not Found\n");

}

else

{

printf("\nRecord Found at Index Position:\t%d\n", count);

}

break;

case 4: display\_records(hash\_table);

break;

case 5: exit(1);

}

}

return 0;

}

