// Menu Driven for Single Linked List

#include <stdio.h>

#include <stdlib.h>

#include <malloc.h>

int EmployeeCount;

typedef struct Employee

{

int empno;

char ename[30];

float sal;

struct Employee \*link; // self referential structure

}EMP;

EMP \*emp; // head node

void create();

void display();

void PrintData(EMP \*, int);

int GetData(EMP \*);

int CheckId(EMP \*);

void insert();

void insert\_begin();

void insert\_pos();

void insert\_end();

void deletelink();

void delete\_begin();

void delete\_pos();

void delete\_end();

void deleteAll();

void search();

void total\_data(int);

void main()

{

int choice;

while (1)

{

system("CLS");

printf("Menu\n================\n");

printf("1. Create\n2. Display\n3. Insert\n4. Delete\n5. Search\n6. Total Data\n7. Exit\n\n");

printf("Enter your choice : ");

scanf("%d", &choice);

switch (choice)

{

case 1:

create();

break;

case 2:

display();

break;

case 3:

insert();

break;

case 4:

deletelink();

break;

case 5:

search();

break;

case 6:

total\_data(1);

break;

case 7:

exit(1);

default:

system("CLS");

printf("Invalid Choice");

}

}

}

void create()

{

char ch;

do

{

system("CLS");

EMP \*temp, \*last;

temp = (EMP \*) malloc(sizeof(EMP));

if (GetData(temp))

{

temp -> link = NULL;

if (emp == NULL)

{

emp = temp;

}

else

{

last = emp;

while (last -> link != NULL)

{

last = last -> link;

}

last -> link = temp;

}

}

last= NULL;

temp = NULL;

free(temp);

free(last);

printf("\n\nDo you want to create another link(Y/N)? : ");

ch = getch();

}while (ch == 'Y' || ch == 'y');

}

void insert()

{

int choice;

if (emp == NULL)

{

system("CLS");

printf("Create the Node First\nPress Enter to create the node...");

getch();

create();

}

while(1)

{

system("CLS");

printf("1. Insert at the beginning\n2. Insert at the position\n3. Insert at the end\n4. Return to main Menu\n\n");

printf("Enter your choice : ");

scanf("%d", &choice);

switch (choice)

{

case 1:

insert\_begin();

break;

case 2:

insert\_pos();

break;

case 3:

insert\_end();

break;

case 4:

return;

default:

printf("Invalid Choice");

}

}

}

void insert\_begin()

{

EMP \*temp;

system("CLS");

temp = (EMP \*) malloc (sizeof(EMP));

if (GetData(temp))

{

temp -> link = emp;

emp = temp;

}

temp = NULL;

free(temp);

}

void insert\_pos()

{

system("CLS");

EMP \*temp , \*disp;

int pos , i;

printf("Enter the position : ");

scanf("%d", &pos);

if (pos <= 1)

{

printf("Cannot Insert at this position");

getch();

return;

}

temp = (EMP \*) malloc(sizeof(EMP));

if (GetData(temp))

{

disp = emp;

for (i=1; i < (pos - 1); i++)

{

disp = disp -> link;

}

temp -> link = disp -> link;

disp -> link = temp;

}

disp = NULL;

temp = NULL;

free(disp);

free(temp);

}

void insert\_end()

{

system("CLS");

EMP \*temp, \*last;

temp = (EMP \*) malloc (sizeof(EMP));

if (GetData(temp))

{

temp -> link = NULL;

last = emp;

while (last -> link != NULL)

{

last = last -> link;

}

last -> link = temp;

}

temp = NULL;

last = NULL;

free(temp);

free(last);

}

void deletelink()

{

int choice;

while(1)

{

system("CLS");

printf("1. Delete at the beginning\n2. Delete at the position\n3. Delete at the end\n4. Delete All\n5. Return to main Menu\n\n");

printf("Enter your choice : ");

scanf("%d", &choice);

if (emp == NULL)

{

system("CLS");

printf("No Data in the list");

getch();

return;

}

switch (choice)

{

case 1:

delete\_begin();

printf("Data Deleted");

getch();

break;

case 2:

delete\_pos();

break;

case 3:

delete\_end();

break;

case 4:

deleteAll();

break;

case 5:

return;

default:

printf("Invalid Choice");

}

}

}

void delete\_begin()

{

system("CLS");

EMP \*temp;

temp = emp;

emp = temp -> link;

temp -> link = NULL;

free(temp);

}

void delete\_pos()

{

system("CLS");

EMP \*temp , \*disp;

int pos , i;

printf("Enter the position : ");

scanf("%d", &pos);

if (pos <= 0)

{

printf("Cannot Delete at this position");

getch();

return;

}

temp = (EMP \*) malloc(sizeof(EMP));

temp = emp;

for (i=1; i < (pos - 1); i++)

{

temp = temp -> link;

disp = temp -> link;

}

temp -> link = disp -> link;

disp -> link = NULL;

temp = NULL;

free(disp);

free(temp);

printf("Data Deleted");

getch();

}

void delete\_end()

{

EMP \*last , \*temp;

temp = emp;

while (temp -> link != NULL)

{

last = temp;

temp = temp -> link;

}

last -> link = NULL;

last = NULL;

free(last);

free(temp);

printf("Data Deleted");

getch();

}

void deleteAll()

{

system("CLS");

while (emp != NULL)

{

delete\_begin();

}

printf("All Data Deleted");

getch();

}

void search()

{

system("CLS");

EMP \*temp;

int id;

printf("Enter the Empno to search : ");

scanf("%d", &id);

temp = emp;

while (temp -> link != NULL)

{

if (temp -> empno == id)

{

PrintData(temp, 1);

getch();

return;

}

temp = temp -> link;

}

printf("No Data Found");

getch();

temp = NULL;

free(temp);

}

void total\_data(int flag)

{

EMP \*temp;

EmployeeCount = 0;

if (emp == NULL)

{

EmployeeCount = 0;

}

else

{

temp = emp;

while (temp != NULL)

{

++EmployeeCount;

temp = temp -> link;

}

}

if (flag == 1)

{

system("CLS");

printf("Total Data : %d", EmployeeCount);

getch();

}

}

void display()

{

system("CLS");

if (emp == NULL)

{

printf("No Data in the list");

getch();

return;

}

int count = 0;

EMP \*temp;

temp = emp;

while (temp != NULL)

{

++count;

PrintData(temp, count);

temp = temp -> link;

}

free(temp);

getch();

}

int GetData(EMP \*temp) // to hold the data in temp

{

printf("Enter the Employee Id : ");

scanf("%d", &temp -> empno);

if (CheckId(temp))

{

printf("Enter the name : ");

fflush(stdin);

gets(temp -> ename);

printf("Enter the salary : ");

scanf("%f", &temp -> sal);

return 1;

}

return 0;

}

int CheckId(EMP \*temp)

{

EMP \*disp;

total\_data(0);

if (EmployeeCount > 0)

{

disp = emp;

while (disp != NULL)

{

if (disp -> empno == temp -> empno)

{

printf("Id Already Exists");

disp = NULL;

free(disp);

getch();

return 0;

}

disp = disp -> link;

}

}

disp = NULL;

free(disp);

return 1;

}

void PrintData(EMP \*temp, int count)

{

printf("%d.\t", count);

printf("Employee Id : %d", temp -> empno);

printf("\n\tEmployee Name : %s", temp -> ename);

printf("\n\tEmployee Salary : %.2f", temp -> sal);

printf("\n\n");

}