1. **program to insert node in the single linked list and display the item of the node**

#include <stdio.h>

#include<conio.h>

void add(int);

void display();

struct employee

{

int info;

struct employee \*link; // self-referential structure

}\*start;

void main()

{

clrscr();

add(10);

add(20);

display();

getch();

}

void add(int num)

{

struct employee \*temp;

temp = (struct employee \*) malloc(sizeof(struct employee));

temp -> info = num;

temp -> link = NULL;

if ( start == NULL)

start = temp;

else

start -> link = temp;

}

void display()

{

while ( start != NULL)

{

printf("%d",start -> info);

start = start -> link;

}

}

1. **program to insert node at the beginning and end in the single linked list and display the item of the node**

#include <stdio.h>

#include<conio.h>

void add(int);

void display();

void add\_begin(int);

struct employee

{

int info;

struct employee \*link; // self-referential structure

}\*start;

void main()

{

clrscr();

add(10);

add(20);

display();

add\_begin(5);

display();

add(25);

display();

add(30);

display();

add\_begin(40);

display();

getch();

}

void add\_begin(int num)

{

struct employee \*temp;

temp = (struct employee \*) malloc (sizeof(struct employee));

temp -> info = num;

temp -> link = start;

start = temp;

}

void add(int num)

{

struct employee \*temp , \*disp;

temp = (struct employee \*) malloc(sizeof(struct employee));

temp -> info = num;

temp -> link = NULL;

if ( start == NULL)

start = temp;

else

{

disp = start;

while ( disp -> link != NULL)

disp = disp -> link;

disp -> link = temp;

}

}

void display()

{

struct employee \*disp;

disp = start;

clrscr();

while ( disp != NULL)

{

printf("%d ",disp -> info);

disp = disp -> link;

}

}

1. **program to insert node at the beginning, middle and end of the single linked list and display the item of the node**

#include <stdio.h>

#include<conio.h>

void add\_begin(int);

void add\_middle(int , int);

void add\_end(int);

void display();

struct employee

{

int info;

struct employee \*link; // self-referential structure

}\*start;

void main()

{

clrscr();

add\_begin(23);

add\_end(10);

add\_begin(5);

add\_middle(87,2);

display();

getch();

}

void add\_begin(int num)

{

struct employee \*temp;

temp = (struct employee \*) malloc (sizeof(struct employee));

temp -> info = num;

temp -> link = start;

start = temp;

}

void add\_middle(int num , int pos)

{

int i;

struct employee \*temp , \*disp;

temp = (struct employee \*) malloc (sizeof(struct employee));

temp -> info = num;

disp = start;

for ( i = 0; i < pos - 1; i++)

{

disp = disp -> link;

if ( disp -> link == NULL)

{

printf("The element cannot insert at position %d", pos);

exit(1);

}

}

temp -> link = disp -> link;

disp -> link = temp;

}

void add\_end(int num)

{

struct employee \*temp , \*disp;

temp = (struct employee \*) malloc(sizeof(struct employee));

temp -> info = num;

temp -> link = NULL;

if ( start == NULL)

{

start = temp;

}

else

{

disp = start;

while ( disp -> link != NULL)

disp = disp -> link;

disp -> link = temp;

}

}

void display()

{

struct employee \*disp;

disp = start;

clrscr();

while ( disp != NULL)

{

printf("%d ",disp -> info);

disp = disp -> link;

}

}

1. **program of implementation of single linked list in menu based program**

#include <stdio.h>

#include <conio.h>

void create();

void insert();

void insert\_begin();

void insert\_at\_pos();

void insert\_end();

void delete();

void delete\_begin();

void delete\_at\_pos();

void delete\_end();

void search();

void display();

void total\_data();

void get\_data(struct employee \*);

void print\_data(struct employee \*);

typedef struct employee

{

int empno;

char ename[10];

int sal;

struct employee \*link;

}EMP;

EMP \*emp;

void main()

{

int choice;

while (1)

{

clrscr();

printf("Menu");

printf("\n");

printf("=============");

printf("\n");

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

printf("\nEnter your choice:");

scanf("%d", &choice);

switch (choice)

{

case 1:

create();

break;

case 2:

insert();

break;

case 3:

display();

break;

case 4:

delete();

break;

case 5:

search();

break;

case 6:

total\_data();

break;

case 7:

exit(0);

default:

printf("Invalid Choice");

}

}

}

void create()

{

char ch;

do

{

EMP \*temp , \*last;

clrscr();

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

get\_data(temp);

temp -> link = NULL;

if (emp == NULL)

emp = temp;

else

{

last = emp;

while (last -> link != NULL)

last = last -> link;

last -> link = temp;

}

printf("Do you want to insert another link:");

ch = getch();

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

}

void insert()

{

int choice;

do

{

clrscr();

printf("1. Insert at beginning");

printf("\n2. Insert at position");

printf("\n3. Insert at the end");

printf("\n4. Exit");

printf("\nEnter your choice:");

scanf("%d", &choice);

switch (choice)

{

case 1:

insert\_begin();

break;

case 2:

insert\_at\_pos();

break;

case 3:

insert\_end();

break;

case 4:

return;

default:

printf("Invalid Choice");

}

}while (choice);

}

void delete()

{

int choice;

clrscr();

printf("1. Delete at beginning");

printf("\n2. Delete at any position");

printf("\n3. Delete at end");

printf("\nEnter your choice:");

scanf("%d", &choice);

switch (choice)

{

case 1:

delete\_begin();

break;

case 2:

delete\_at\_pos();

break;

case 3:

delete\_end();

break;

default:

printf("Invalid choice");

}

}

void insert\_begin()

{

EMP \*temp;

clrscr();

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

get\_data(temp);

if (emp == NULL)

{

emp = temp;

temp -> link = NULL;

}

else

{

temp -> link = emp;

emp = temp;

}

}

void insert\_at\_pos()

{

int pos , i;

EMP \*temp , \*last;

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

clrscr();

printf("Enter the index number:");

scanf("%d", &pos);

if (pos==0)

{

printf("Index number must be greater than 0");

getch();

return;

}

last = emp;

for ( i =0; i < pos -1; i++)

{

last = last -> link;

if ( last -> link == NULL)

{

printf("The element cannot be inserted at index %d", pos);

getch();

return;

}

}

get\_data(temp);

temp -> link = last -> link;

last -> link = temp;

}

void insert\_end()

{

EMP \*temp , \*last;

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

clrscr();

last = emp;

while ( last -> link != NULL)

last = last -> link;

get\_data(temp);

last -> link = temp;

temp -> link = NULL;

}

void delete\_begin()

{

EMP \*temp;

if ( emp == NULL)

printf("List is empty");

else

{

temp = emp;

emp = emp -> link;

temp = NULL;

free(temp);

}

}

void delete\_at\_pos()

{

int pos, i;

EMP \*temp , \*last;

clrscr();

printf("Enter the index number:");

scanf("%d", &pos);

temp = emp;

for ( i = 0; i < pos - 1; i++)

{

temp = temp -> link;

if ( temp -> link == NULL)

{

printf("Deletion is not possible");

getch();

return;

}

}

last = temp -> link;

temp -> link = last -> link;

last -> link = NULL;

free(last);

}

void delete\_end()

{

EMP \*temp , \*last;

temp = emp;

while (temp -> link != NULL)

{

last = temp;

temp = temp -> link;

}

last -> link = NULL;

free(temp);

}

void search()

{

int element;

EMP \*temp;

clrscr();

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

scanf("%d", &element);

temp = emp;

while (temp != NULL)

{

if (element == temp -> empno)

{

print\_data(temp);

break;

}

temp = temp -> link;

}

if (temp == NULL)

printf("Element Not Exist");

getch();

}

void display()

{

EMP \*temp;

clrscr();

temp = emp;

while ( temp != NULL)

{

print\_data(temp);

printf("\n\n");

temp = temp -> link;

}

getch();

}

void total\_data()

{

int count = 0;

EMP \*temp;

clrscr();

temp = emp;

while (temp != NULL)

{

++count;

temp = temp -> link;

}

printf("Total no. of data: %d", count);

getch();

}

void get\_data(EMP \*temp)

{

printf("Enter the empno:");

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

printf("Enter the ename:");

scanf("%s" ,&temp -> ename);

printf("Enter the salary:");

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

}

void print\_data(EMP \*temp)

{

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

printf("\nEname: %s", temp -> ename);

printf("\nSalary: %d", temp -> sal);

}