Data Structures

SRP 🡪 When a member of a structure is a pointer which is pointing to itself type(structure type), it is called “Self-Referential Pointer”.

It is used to navigate between each record.

1. Creating nodes 🡪allocating static memory

Home h1, Home h2, Home h3;

1. Inserting/Assigning/Setting nodes:

H1. Value=10, h2.value=20, h3.value=30;

H1.ptr=NULL, h2.ptr=NULL, h3.ptr=NULL;

1. Creating a relationship

Struct home \*ptr1=NULL;

Ptr1= &h1;

Pf(“%d”, h1.value);

Pf(“%d”,ptr->value);

H1.ptr= &h2;

H2.ptr=&h3;

1. Created a linked list
2. Navigating/ Traversing

Pf(“%d”, ptr1->value); 🡺10

Ptr1= h1.ptr;

Pf(“%d”, ptr1->value); 🡺20

Ptr1= h2.ptr;

Pf(“%d”, ptr1->value); 🡺30

Self in python is a self referential pointer

//Making relationship

h1.ptr=&h2;

h2.ptr=&h3;

//traverse

ptr1=&h1; //ptr1 pointing to base address or first element

printf("\n Contents of ptr1 :%p",ptr1);

printf("\n %d",h1.val);

printf("\n %d",ptr1->val);

//ptr1= &h2;

//ptr1= h1.ptr;

ptr1=ptr1->ptr;

printf("\n %d",ptr1->val);

//ptr1= &h3;

//ptr1= h2.ptr;

ptr1=ptr1->ptr;

printf("\n %d",ptr1->val);

printf("\n\n");

void printlist(NODE \*head)

{

printf("\n List using function\n");

while(head!=NULL)

{

printf("%d->",head->val);

head=head->ptr;

}

printf("NULL");

}

//Inserting a node in the middle

h4.val=40;

h4.ptr=NULL;

ptr1=&h1;

ptr1->ptr= &h4;

ptr1=&h4;

ptr1->ptr= &h2;

printf("Inserted list \n");

ptr1=&h1;

printlist(ptr1);

//Adding anew node at the end

h5.val=50;

h5.ptr=NULL;

ptr1=&h1;

while(ptr1->ptr!=NULL)

{

ptr1=ptr1->ptr;

}

ptr1->ptr=&h5;

ptr1=&h1;

printf("\nNew list \n");

printlist(ptr1);

// Adding anew node at start

h7.val=70;

h7.ptr=NULL;

h7.ptr=&h1;

ptr1=&h7;

printf("\nNew list after adding at first \n");

printlist(ptr1);

While appending a node if there’s no node then the appending node should become the first node

#include<stdio.h>

#include<stdlib.h>

typedef struct node

{

int val;

struct node \*next; //Self referential pointer (SELF)

}NODE;

void printlist(NODE \*);

NODE \* appendNode(NODE \*,NODE \*);

NODE \* addNodeBeg(NODE \*,NODE \*);

int deleteNode(NODE \*,int);

int main()

{

NODE \*nn=NULL;

NODE \*head=NULL,\*temp=NULL;

int ch;

do

{

nn= (NODE \*)malloc(sizeof(NODE));

printf("\n New node address = %p", nn);

printf("\n ENter the value of new node");

scanf("%d",&nn->val);

nn->next=NULL;

head=appendNode(head,nn);

if(head==NULL)

{

printf("\n Unable to create node");

return 1;

}

printf("\nDo you want to add new node? (1/0)");

scanf("%d",&ch);

}while(ch!=0);

printlist(head);

printf("\n\n");

return 0;

}

void printlist(NODE \*head)

{

printf("\n List using function\n");

if(head==NULL)

{

printf("\n List is empty");

return;

}

while(head!=NULL)

{

printf("%d->",head->val);

head=head->next;

}

printf("NULL");

}

NODE \* appendNode(NODE \*head,NODE \*nn)

{

NODE \*temp=head;

printf("\n In append node \n");

if(head==NULL)

{ //empty list

head=nn;

temp=head;

}

else

{

while(head->next!=NULL)

{

printf("%d->",head->val);

head=head->next;

}

printf("\n Head value:%d",head->val);

printf("\n Head addres: %p",head->next);

head->next=nn;

}

head=temp;

return head;

}

NODE \* addNodeBeg(NODE \*head,NODE \*nn)

{

NODE \*temp=head;

if(head==NULL)

{

printf("\n List is empty\n");

head=nn;

temp=nn;

}

else

{

nn->next =head;

head=nn;

}

return head;

}

int deleteNode(NODE \*head,int key)

{

int flag=1;

printf("\n In delete Node\n");

NODE \*temp=head;

while(head!=NULL)

{

if(head->val==key)

{

flag=0;

temp->next=head->next;

break;

}

temp=head;

head=head->next;

}

return flag;

}



Deleting a node at beginning and deleting a node at end 🡪do it from 7th version

In circular linked list what should be the values of prev and next

Menu based emp

Double Linked List

Always in dll when we want to do back tracing move to the end and come back



Circular Linked List

For a circular linked list if there is only one node then the the prev and next will be pointing to itself

