

DATA STRUCTURES

25. IMPLEMENTATION OF LINKED LIST

```
#include <stdio.h>
```

```
#include<stdlib.h>
```

```
struct node
```

```
{  
  
    int data;  
  
    struct node *next;  
  
};  
  
struct node *h,*nn,*t,*p;
```

```
void ibegin()
```

```
{  
  
nn=(struct node*)malloc(sizeof(struct node));  
  
    printf("enter data ");  
  
    scanf("%d",&nn->data);  
  
    nn->next=h;  
  
    h=nn;  
  
}
```

```
void imid()
```

```
{  
  
    nn=(struct node*)malloc(sizeof(struct node));  
  
    printf("enter data ");  
  
    scanf("%d",&nn->data);  
  
    int i=1,pos;  
  
    printf("\nenter insert position ");  
  
    scanf("%d",&pos);
```

```

        t=h;

        while(i<pos && t!=0)

        {

            t=t->next;

            i++;

        }

        nn->next=t->next;

        t->next=nn;

    }

```

```

void iend()

{

    nn=(struct node*)malloc(sizeof(struct node));

    printf("enter data ");

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

    nn->next=0;

    t=h;

    while(t->next!=0)

    {

        t=t->next;

    }

    t->next=nn;

}

```

```

void ins()

{

```

```

int a;

do
{

printf("\nenter insersion type: 1-b 2-m 3-e\n");

scanf("%d",&a);

switch(a)
{

        case 1: ibegin();

        break;

        case 2: imid();

        break;

        case 3: iend();

        break;

        default: printf("invalid choice");

    }

}

while(a!=0);

}

```

```

void dbegin()

{

    t=h;

    h=t->next;

    t->next=0;

    free(t);

```

```

}

void dmid()
{
    int pos,i=1;
    printf("\nenter insert position ");

    scanf("%d",&pos);

    t=h;

    while(i<pos && t!=0)
    {
        t=t->next;
        i++;
    }

    while(p->next!=t)

    p=p->next;

    p->next=t->next;

    t->next=0;

    free(t);
}

void dend()
{
    while(t->next!=0)
    {
        t=t->next;
    }

    while(p->next!=t)

    p=p->next;

    p->next=0;
}

```

```

        free(t);
    }
void del()
{
    int b;
    do
    {

        printf("\nenter delete type: 1-b 2-m 3-e\n");
        scanf("%d",&b);
        switch(b)
        {

            case 1: dbegin();

            break;

            case 2: dmid();

            break;

            case 3: dend();

            break;

            default: printf("invalid choice");

        }
    }
    while(b!=0);
}

```

```

void display()

```

```

    {
        t=h;
        while(t!=0)
        {
            printf("%3d",t->data);

            t=t->next;
        }
    }

int main()
{
    h=0;
    int c=1;
    while(c==1)
    {
        nn=(struct node*)malloc(sizeof(struct node));

        printf("enter data ");

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

        nn->next=0;

        if(h==0)
        {
            h=t=nn;
        }
        else
        {
            t->next=nn;

            t=nn;
        }
    }
}

```

```
    printf("enter 1-continue 0-stop");

    scanf("%d",&c);

}

t=h;

printf("list elements are\n");

while(t!=0)

{

    printf("%3d",t->data);

    t=t->next;

}

int x;

do

{

    printf("\nenter type: 1-insert 2-delete 3-display\n");

    scanf("%d",&x);

    switch(x)

    {

        case 1: ins();

        break;

        case 2: del();

        break;

        case 3: display();

        break;

        default: printf("invalid choice");

    }

}
```

```

    }

}

while(x!=0);

return 0;

}

```

The screenshot shows a web browser window with the URL `programiz.com/c-programming/online-compiler/`. The page title is "Online C Compiler". The main content area is divided into two panels: "main.c" on the left and "Output" on the right.

The "main.c" panel contains the following code:

```

1 #include <stdio.h>
2 #include <stdlib.h>
3 struct node
4 {
5     int data;
6     struct node *next;
7 };
8 struct node *h, *nn, *t, *p;
9
10 void ibegin()
11 {
12     nn=(struct node*)malloc(sizeof(struct node));
13     printf("enter data ");
14     scanf("%d",&nn->data);
15     nn->next=h;
16     h=nn;
17 }
18 void imid()
19 {
20     nn=(struct node*)malloc(sizeof(struct node));
21     printf("enter data ");
22     scanf("%d",&nn->data);
23     int i=1,pos;
24     printf("\nenter insert position ");
25     scanf("%d",&pos);

```

The "Output" panel shows the following text:

```

enter 1-continue 0-stop1
enter data 6
enter 1-continue 0-stop1
enter data 7
enter 1-continue 0-stop0
list elements are
5 6 7
enter insertion type: 1-insert 2-delete 3-display
1
enter insertion type: 1-b 2-m 3-e
1
enter data 5
enter insertion type: 1-b 2-m 3-e
1
enter data 9
enter insertion type: 1-b 2-m 3-e
d
invalid choice
enter insertion type: 1-insert 2-delete 3-display
3
9 5 5 6 7
enter insertion type: 1-insert 2-delete 3-display
2
enter delete type: 1-b 2-m 3-e
1

```

26. MERGE TWO LISTS

```

#include <stdio.h>

#include<stdlib.h>

int main()

{

    struct node

    {

        int data;

        struct node *next;

    };

    struct node *h1,*nn1,*t1;

    h1=0;

```



```

int c1=1;

printf("enter list 1 data\n");

while(c1==1)
{
    nn1=(struct node*)malloc(sizeof(struct node));

    printf("enter data ");

    scanf("%d",&nn1->data);

    nn1->next=0;

    if(h1==0)
    {
        h1=t1=nn1;
    }
    else
    {
        t1->next=nn1;

        t1=nn1;
    }

    printf("enter 1-continue 0-stop");

    scanf("%d",&c1);
}

t1=h1;

printf("1st list elements are");

while(t1!=0)
{
    printf("%3d",t1->data);

    t1=t1->next;
}

```

```

        struct node *h2,*nn2,*t2;

h2=0;

int c2=1;

printf("\nenter list 2 data\n");

while(c2==1)

{

    nn2=(struct node*)malloc(sizeof(struct node));

    printf("enter data ");

    scanf("%d",&nn2->data);

    nn2->next=0;

    if(h2==0)

    {

        h2=t2=nn2;

    }

    else

    {

        t2->next=nn2;

        t2=nn2;

    }

    printf("enter 1-continue 0-stop");

    scanf("%d",&c2);

}

t2=h2;

printf("\n2nd list elements are");

while(t2!=0)

{

    printf("%3d",t2->data);

```

```

        t2=t2->next;

    }

    t1=h1;

    while(t1->next !=0)

    {

        t1=t1->next;

    }

    t1->next=h2;

    t1=h1;

    printf("\n merged list elements are");

    while(t1!=0)

    {

        printf("%3d",t1->data);

        t1=t1->next;

    }

    return 0;

}

```

The screenshot shows a web browser window with the URL `programiz.com/c-programming/online-compiler/`. The page title is "Online C Compiler". The code editor contains the following C code:

```

1 // Merge two sorted linked lists
2
3 #include <stdio.h>
4 #include <stdlib.h>
5
6 struct Node {
7     int data;
8     struct Node *next;
9 };
10
11 int main() {
12     // Create first list
13     int n1, n2;
14     printf("\nEnter number of nodes for list 1: ");
15     scanf("%d", &n1);
16     printf("\nEnter number of nodes for list 2: ");
17     scanf("%d", &n2);
18
19     // Create first list
20     struct Node *h1 = NULL;
21     struct Node *t1 = h1;
22     while(n1 > 0) {
23         struct Node *newNode = (struct Node *) malloc(sizeof(struct Node));
24         printf("Enter data for node %d: ", n1);
25         scanf("%d", &newNode->data);
26         newNode->next = NULL;
27         if(h1 == NULL) {
28             h1 = newNode;
29         } else {
30             t1->next = newNode;
31             t1 = t1->next;
32         }
33         n1--;
34     }
35
36     // Create second list
37     struct Node *h2 = NULL;
38     struct Node *t2 = h2;
39     while(n2 > 0) {
40         struct Node *newNode = (struct Node *) malloc(sizeof(struct Node));
41         printf("Enter data for node %d: ", n2);
42         scanf("%d", &newNode->data);
43         newNode->next = NULL;
44         if(h2 == NULL) {
45             h2 = newNode;
46         } else {
47             t2->next = newNode;
48             t2 = t2->next;
49         }
50         n2--;
51     }
52
53     // Merge the two lists
54     struct Node *mergedList = h1;
55     printf("\nMerged list elements are: ");
56     while(mergedList != NULL) {
57         printf("%3d", mergedList->data);
58         mergedList = mergedList->next;
59     }
60
61     return 0;
62 }

```

The output window shows the following text:

```

/tmp/hJDEuTBp4h.o
Enter list 1 data
Enter data 5
Enter 1-continue 0-stop1
Enter data 4
Enter 1-continue 0-stop1
Enter data 3
Enter 1-continue 0-stop0
1st list elements are 5 4 3
Enter list 2 data
Enter data 6
Enter 1-continue 0-stop1
Enter data 9
Enter 1-continue 0-stop1
Enter data 4
Enter 1-continue 0-stop0
2nd list elements are 6 9 4
merged list elements are 5 4 3 6 9 4

```

27. TO IMPLEMENT STACK OPERATIONS

```
#include<stdio.h>
```

```
#include<stdlib.h>
```

```
int s[10];
```

```
int t=-1,n=10;
```

```
void push()
```

```
{
```

```
    int x;
```

```
    printf("enter data");
```

```
    scanf("%d",&x);
```

```
    if(t>=n-1)
```

```
    {
```

```
        printf("stack is full\n");
```

```
    }
```

```
    else
```

```
    {
```

```
        t++;
```

```
        s[t]=x;
```

```
    }
```

```
}
```

```
void pop()
```

```
{
```

```
    int item;
```

```
    if(t== -1)
```

```
    printf("stack is empty to pop\n");
```

```
    else
```

```

        {
            item=s[t];
            t--;
        }
        printf("popped element is %2d\n",item);
    }
void peek()
{
    if(t== -1)
        printf("stack is empty\n");
    else
        printf("peek element is %2d\n",s[t]);
}
void display()
{
    int i;
    for(i=t;i>=0;i--)
        printf("%3d",s[i]);
}
int main()
{
    int choice;
    do
    {
        printf("\nenter choice:\n 1-push  2-pop  3-peek  4-display \n");
        scanf("%d",&choice);
        switch(choice)

```

```

        {

            case 1: push();

            break;

            case 2: pop();

            break;

            case 3: peek();

            break;

            case 4: display();

            break;

            default: printf("invalid choice");

        }

    }

    while (choice!=0);

}

```

The screenshot shows a web browser window with the URL `programiz.com/c-programming/online-compiler/`. The page title is "Online C Compiler". The code editor contains the following C code:

```

main.c
46- {
47-     int choice;
48-     do
49-     {
50-         printf("\nEnter choice:\n 1-push 2-pop 3-peek 4-display \n");
51-         scanf("%d",&choice);
52-         switch(choice)
53-         {
54-             case 1: push();
55-             break;
56-             case 2: pop();
57-             break;
58-             case 3: peek();
59-             break;
60-             case 4: display();
61-             break;
62-             default: printf("invalid choice");
63-         }
64-     }
65-     while (choice!=0);
66- }

```

The output window shows the following text:

```

/tmp/hJDEuTBp4h.o
enter choice:
1-push 2-pop 3-peek 4-display
1
enter data5
enter choice:
1-push 2-pop 3-peek 4-display
1
enter data7
enter choice:
1-push 2-pop 3-peek 4-display
2
popped element is 7
enter choice:
1-push 2-pop 3-peek 4-display
4
5
enter choice:
1-push 2-pop 3-peek 4-display

```

28. TO IMPLEMENT QUEUE OPERATIONS

```
#include<stdio.h>
```

```
int q[5],f=-1,r=-1,n=5;
```

```

void enqueue()
{
    int x;

    printf("enter enqueue value");

    scanf("%d",&x);

    if(r>=n-1)

        printf("queue is full");

    else if(f== -1 &&r== -1)

    {

        f++;

        r++;

        q[r]=x;

    }

    else

    {

        r++;

        q[r]=x;

    }

}

void dequeue()
{

    if(r== -1 &&f== -1)

        printf("queue is empty to dequeue");

    else if(f==r)

        f=r-1;

    else

    {

```

```

        printf("dequeued element is %d",q[f]);

        f++;

    }

}

void display()

{

    int i;

    for(i=f;i<=r;i++)

        printf("%3d",q[i]);

}

int main()

{

    int choice;

    do

    {

        printf("\nenter choice:\n 1-enque  2-deque  3-display \n");

        scanf("%d",&choice);

        switch(choice)

        {

            case 1: enqueue();

            break;

            case 2: dequeue();

            break;

            case 3: display();

            break;

            default: printf("invalid choice");

        }

    }

}

```



```

    }

    while (choice!=0);

}

```

The screenshot shows a web browser window with the URL `programiz.com/c-programming/online-compiler/`. The page title is "Programiz C Online Compiler". The interface includes a "Run" button and a "Clear" button. The code editor on the left contains the following C code:

```

main.c
25 printf("queue is empty to dequeue");
26 else if(f==r)
27     f=r-1;
28 else
29 {
30     printf("dequeued element is %d",q[f]);
31     f++;
32 }
33 }
34 void display()
35 {
36     int i;
37     for(i=f;i<=r;i++)
38         printf("%3d",q[i]);
39 }
40 int main()
41 {
42     int choice;

```

The output window on the right shows the following text:

```

/tmp/XndKk6ZswC.o
enter choice:
1-enqueue 2-dequeue 3-display
1
enter enqueue value4
enter choice:
1-enqueue 2-dequeue 3-display
1
enter enqueue value5
enter choice:
1-enqueue 2-dequeue 3-display
3
4 5
enter choice:
1-enqueue 2-dequeue 3-display
0
invalid choice

```

The Windows taskbar at the bottom shows the date and time as 9:15 AM on 30-Aug-23, and the weather as 29°C Mostly sunny.

29. TO CONVERT INFIX TO POSTFIX USING STACK

```
#include<stdio.h>
```

```
#include<stdlib.h>
```

```
#include<string.h>
```

```
char s[50],in[50],post[50];
```

```
int t=-1;
```

```
void push(char);
```

```
char pop();
```

```
int empty();
```

```
void topost();
```

```
void print();
```

```
int pre(char);
```

```
int main()
```

```

{

    printf("enter infix expression\n");

    gets(in);

    topost();

    print();

    return 0;

}

void topost()

{

    int i,j=0;

    char sym,nxt;

    for(i=0;i<strlen(in);i++)

    {

        sym=in[i];

        switch(sym)

        {

            case '(':

                push(sym);

                break;

            case ')':

                while((nxt=pop())!='(')

                    post[j++]=nxt;

                break;

            case '+':

            case '-':

            case '*':

```

```

        case '/':

        case '^':

        while(!empty() && pre(s[t])>=pre(sym))

        post[j++]=pop();

        push(sym);

        break;

        default:

        post[j++]=sym;

    }

}

while(!empty())

post[j++]=pop();

post[j++]='\0';

}

```

```

int pre(char sym)

{

    switch(sym)

    {

        case '+':

        case '-':

            return 1;

        case '*':

        case '/':

            return 2;

        case '^':

            return 3;

```

```

        case '%':

            return 4;

        default:

            return 0;

    }

}

void print()

{

    int i=0;

    printf("postfix expression is\n");

    while(post[i])

    {

        printf("%c",post[i++]);

    }

    printf("\n");

}

void push(char c)

{

    if(t>=50-1)

        printf("stack is full\n");

    else

    {

        t++;

        s[t]=c;

    }

}

```

```
char pop()
{
    int c;

    if(t== -1)
    {
        printf("stack empty");

    }

    else
    {
        c=s[t];
        t--;
        return c;
    }
}

int empty()
{
    if(t== -1)
        return 1;
    else
        return 0;
}
```

The screenshot shows a web browser window with the URL `programiz.com/c-programming/online-compiler/`. The page title is "Programiz C Online Compiler". The interface includes a "Run" button and an "Output" panel. The code in the editor is as follows:

```
main.c
98  if(t==1)
99  {
100  printf("stack empty");
101  }
102  else
103  {
104  {
105      c=s[t];
106      t--;
107      return c;
108  }
109  }
110  int empty()
111  {
112      if(t==1)
113          return 1;
114      else
115          return 0;
116  }
```

The output panel shows the following text:

```
/tmp/NQaGwbS1iy.o
enter infix expression
a+b
postfix expression is
ab+
```

30. TO EVALUTE THE POSTFIX EXPRESSION

```
#include<stdio.h>
```

```
#include<string.h>
```

```
#include<ctype.h>
```

```
char post[50];
```

```
float s[50];
```

```
int t=-1;
```

```
void push(float c)
```

```
{
```

```
    t++;
```

```
    s[t]=c;
```

```
}
```

```
float pop()
```

```
{
```

```
    float x;
```

```
    x=s[t];
```

```
    t--;
```

```

return x;
}

int main()
{
    float v1,v2;

    int i;

    printf("enter postfix expression\n");

    scanf("%s",&post);

    for(i=0;post[i]!='\0';i++)
    {
        if(isdigit(post[i]))
        {
            push(post[i]-'0');
        }
        else
        {
            v1=pop();
            v2=pop();
            switch(post[i])
            {
                case '+':
                    push(v2+v1);
                    break;

                case '-':
                    push(v2-v1);
                    break;

                case '*':

```

```

        push(v2*v1);

        break;

    case '/':

        push((float)v2/v1);

        break;

    }

}

}

printf("result=%f\n",s[t]);

return 0;

}

```

The screenshot shows a web browser window with the URL `programiz.com/c-programming/online-compiler/#google_vignette`. The page title is "Programiz C Online Compiler". The code editor contains the following C code:

```

1 #include<stdio.h>
2 #include<string.h>
3 #include<ctype.h>
4 char post[50];
5 float s[50];
6 int t=-1;
7 void push(float c)
8 {
9     t++;
10    s[t]=c;
11 }
12 float pop()
13 {
14     float x;
15     x=s[t];
16     t--;
17     return x;
18 }
19 int main()
20 {
21     float v1,v2;
22     int i;
23     printf("enter postfix expression\n");
24     scanf("%s",post);
25     for(i=0;post[i]!='\0';i++)

```

The output window shows the following text:

```

/tmp/nEbuqr0z5f.o
enter postfix expression
2345+*-
result=-25.000000

```

The Windows taskbar at the bottom shows the system clock as 10:51 AM on 29-Aug-23, with a temperature of 31°C and weather "Partly sunny".

31. TO IMPLEMENT TREE TRAVERSALS

```
#include <stdio.h>
```

```
#include<stdlib.h>
```

```
struct node{
```

```
    int data;
```



```
    struct node* l;  
    struct node* r;  
};  
  
void inorder(struct node* root){  
    if(root==NULL){  
        return;  
    }  
    inorder(root->l);  
    printf("%d ",root->data);  
    inorder(root->r);  
}  
  
void postorder(struct node* root){  
    if(root==NULL){  
        return;  
    }  
    postorder(root->l);  
    postorder(root->r);  
    printf("%d ",root->data);  
}  
  
void preorder(struct node* root){  
    if(root==NULL){  
        return;  
    }  
    printf("%d ",root->data);  
    preorder(root->l);  
    preorder(root->r);  
}
```

```

}

struct node *create()
{
    int x;

    struct node *nn;

    nn=(struct node*)malloc(sizeof(struct node));

    printf("enter data (-1 for no node)");

    scanf("%d",&x);

    if(x== -1)

    return 0;

    nn->data=x;

    printf("enter left child of %d  ",x);

    nn->l=create();

    printf("enter right child of %d  ",x);

    nn->r=create();

    return nn;
}

```

```

int main(){

    int a;

    struct node* root;

    root=create();

    printf("enter the traversal type inorder->1 preorder->2 postorder->3:");

    scanf("%d",&a);

    switch(a){

        case 1:inorder(root);

        break;

```

```

    case 2:preorder(root);

    break;

    case 3:postorder(root);

    break;

}

return 0;

}

```

The screenshot shows a web browser window with the URL `programiz.com/c-programming/online-compiler/`. The page title is "Programiz C Online Compiler". The code editor on the left contains the following C code:

```

1 #include <stdio.h>
2 #include<stdlib.h>
3 struct node{
4     int data;
5     struct node* l;
6     struct node* r;
7 };
8
9 void inorder(struct node* root){
10     if(root==NULL){
11         return;
12     }
13     inorder(root->l);
14     printf("%d ",root->data);
15     inorder(root->r);
16 }
17 void postorder(struct node* root){
18     if(root==NULL){
19         return;

```

The "Output" panel on the right shows the following text:

```

/tmp/Fn59THSIFQ.o
enter data (-1 for no node)5
enter left child of 5   enter data (-1 for no node)-1
enter right child of 5   enter data (-1 for no node)6
enter left child of 6   enter data (-1 for no node)-1
enter right child of 6   enter data (-1 for no node)-1
enter the traversal type inorder->1 preorder->2 postorder->3:1
5 6

```

The Windows taskbar at the bottom shows the date and time as 9:15 AM on 30-Aug-23, and the weather as 29°C Mostly sunny.