

```

1  #include <stdio.h>
2  #include <stdlib.h>
3
4  struct node
5  {
6      int data;
7      struct node* next;
8  };
9
10 struct node *rear=NULL, *front =NULL, *top=NULL;
11
12 struct node* getnode(int item)
13 {
14     struct node* newn = (struct node*)malloc(sizeof(struct node));
15     newn->data = item;
16     newn->next = NULL;
17     return newn;
18 }
19
20 void display(struct node* head)
21 {
22     if(head == NULL)
23     {
24         printf("List is empty.\n");
25         return;
26     }
27     struct node* ptr = head;
28     while(ptr)
29     {
30         printf("%d\t", ptr->data);
31     }

```

main.c

```
32     },
33     printf("\b\b\b\n");
34 }
35
36 struct node* insertfront(struct node* head, int item)
37 {
38     struct node* newn = getnode(item);
39     newn->next = head;
40     head = newn;
41     return head;
42 }
43
44 void swap(int *a, int *b)
45 {
46     int temp;
47     temp = *a;
48     *a = *b;
49     *b = temp;
50 }
51
52
53 struct node* sort (struct node* head)
54 {
55     int sorted;
56     if(head == NULL) return head;
57     struct node* ptr = head;
58     do
59     {
60         ptr = head;
61         sorted = 0;
62     }
```

```

62 sorted = 0;
63 while(ptr->next)
64 {
65     if(ptr->data > ptr->next->data)
66     {
67         swap(&ptr->data, &ptr->next->data);
68         sorted = 1;
69     }
70     ptr = ptr->next;
71 } while(sorted == 1);
72 return head;
73 }
74
75 void reverse(struct node** head)
76 {
77     struct node* prev = NULL;
78     struct node* current = *head;
79     struct node* next = NULL;
80     while (current != NULL) {
81         next = current->next;
82         current->next = prev;
83         prev = current;
84         current = next;
85     }
86     *head = prev;
87 }
88
89 struct node* concatenate(struct node* head1, struct node* head2)
90 {
91

```

main.c

```
89 struct node concatenate(struct node head1, struct node head2)
90 {
91     struct node* ptr = head1;
92     while(ptr->next)
93     {
94         ptr = ptr->next;
95     }
96     ptr->next = head2;
97     return head1;
98 }
99
100
101
102 void qinsert()
103 {
104     struct node *newnode;
105     newnode=(struct node *) malloc(sizeof(struct node));
106     printf("Enter the element:\n");
107     scanf("%d",&newnode->data);
108     newnode->next=NULL;
109
110     if(rear==NULL)
111     {
112         rear=newnode;
113         front=newnode;
114     }
115     else
116     {
117         rear->next=newnode;
118     }
119 }
```

```

118         rear->next=newnode;
119         rear=newnode;
120     }
121 }
122
123 void qdel()
124 {
125     if(front==NULL)
126     {
127         printf("Queue is empty\n");return;
128     }
129     else
130     {
131         printf("Deleted ele is %d",front->data);
132         if(front==rear)
133         {
134             printf("Queue is empty\n");
135             front=NULL; rear=NULL;
136         }
137         else
138             front=front->next;
139     }
140 }
141
142 void qdisplay()
143 {
144     struct node *temp;
145     if(front ==NULL)
146     {
147

```

```

146     if(front == NULL)
147     {
148         printf("Queue is empty");
149         return;
150     }
151     temp=front;
152     while (temp != NULL)
153     {
154         printf("%d ",temp->data);
155         temp=temp->next;
156     }
157 }
158
159 void spush()
160 {
161     int item;
162     struct node *newnode;
163     printf("Enter the element\n");
164     scanf("%d",&item);
165
166     newnode=(struct node*)malloc(sizeof(struct node));
167     newnode->data=item;
168     newnode->next=NULL;
169     if(top==NULL)
170         top=newnode;
171     else
172         newnode->next=top;
173     top=newnode;
174 }
175 void spop()
176

```



and debugger for  
c++

file, run, debug, share.

IDE

## Projects

[com](#) [Contact](#)

## programming

### Learning Questions

[Sign Up](#)[Login](#)**WMAA.C**

```

175 void spop()
176 {
177     if(top==NULL)
178         printf("stack is empty");
179     else
180     {
181
182         printf("element removed is %d:", top->data);
183
184         top=top->next;
185
186     }
187
188
189 }
190
191 void sdisplay()
192 {
193     struct node *temp;
194     temp=top;
195     if(top==NULL)
196         printf("Stack is empty");
197     while(temp!=NULL)
198     {
199         printf("%d",temp->data);
200         printf("\n");
201         temp=temp->next;
202     }
203
204

```

```

206 int main()
207 {
208     printf("Linked list program\n");
209     int n1, n2, n, ch, flag = 0;
210     int choice;
211     struct node* head1 = NULL; struct node* head2 = NULL;
212     do
213     {
214         printf("Enter the choice\n1.Stack\n2.Queue\n3: Linked list 1\n4: Linked list 2\n");
215         scanf("%d", &n1);
216         switch(n1)
217         {
218             case 1:
219             {
220                 do
221                 {
222                     printf("\n1. Push \n2. Display \n3. Pop\n");
223                     printf("\nEnter your choice : ");
224                     scanf("%d", &choice);
225                     switch(choice)
226                     {
227                         case 1: spush(); break;
228                         case 2: sdisplay(); break;
229                         case 3: spop(); break;
230                     }
231                 } while (choice != 0);
232             }
233         }
234     } while (n1 != 0);
235 }

```



}

case 2:

{

do

{ printf("\nQueue implementation using linked list\n");  
printf("\n1. Create \n2. Display \n3. Delete \n4. Exit \n");  
printf("\nEnter your choice : ");  
scanf("%d",&choice);  
switch(choice)  
{ case 1: qinsert(); break;  
case 2: qdisplay();break;  
case 3: qdel(); break;

}

}while(choice!=10);

}

case 3:

{

do

{

printf("3: Insert\n4: Sort\n5: Reverse\n6: Concatenate with 1  
scanf("%d", &n2);  
switch(n2)  
{

case 3: f

```
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286

case 3:
{
do
{
printf("3: Insert\n4: Sort\n5: Reverse\n6: Concatenate with\n");
scanf("%d", &n2);
switch(n2)
{
case 3: {
printf("Enter item to be inserted: ");
scanf("%d", &n);
head1 = insertfront(head1, n);
break;
}
case 4: {
head1 = sort(head1);
break;
}
case 5: {
reverse(&head1);
break;
}
case 6: {
head1 = concatenate(head1, head2);
break;
}
case 7: {
// ...
}
```

```

    }
    case 5: {
        reverse(&head1);
        break;
    }
    case 6: {
        head1 = concatenate(head1, head2);
        break;
    }
    case 7: {
        display(head1);
        break;
    }
    case 8: {
        flag = 1;
        break;
    }
    case 9: {
        exit(0);
    }
    default: printf("Invalid input.\n");
}
if(flag == 1)
{
    break;
}
}while(1);
break;
}

```

```

flag = 0;
do
{
    printf("3: Insert\n4: Sort\n5: Reverse\n6: Concatenate wi
    scanf("%d", &n2);
    switch(n2)
    {
        case 3: {
            printf("Enter item to be inserted: ");
            scanf("%d", &n);
            head2 = insertfront(head2, n);
            break;
        }
        case 4: {
            head2 = sort(head2);
            break;
        }
        case 5: {
            reverse(&head2);
            break;
        }
        case 6: {
            head2 = concatenate(head2, head1);
            break;
        }
        case 7: {
            display(head2);
            break;
        }
    }
}

```

```

338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
case 6: {
    head2 = concatenate(head2, head1);
    break;
}
case 7: {
    display(head2);
    break;
}
case 8: {
    flag = 1;
    break;
}
case 9: {
    exit(0);
}
default: printf("Invalid input.\n");
}
if(flag == 1)
{
    flag = 0; break;
}
}while(1);
break;
}
case 9: exit(0);
default: printf("Invalid input.\n");
}
}while(1);
return 0;
}

```



input

1. Display

2. Pop

Enter your choice : 1

Enter the element

2

1. Push

2. Display

3. Pop

Enter your choice : 3

Element removed is 2:

1. Push

2. Display

3. Pop

I

Enter your choice : 2

Stack is empty

1. Push

2. Display

3. Pop

Enter your choice :