

```

1  #include <stdio.h>
2  #include <stdlib.h>
3
4  #define StackLimit 20
5
6  typedef int StackElementType;
7  typedef struct {
8      int Top;
9      StackElementType Element[StackLimit];
10 } StackType;
11
12 typedef enum {
13     FALSE, TRUE
14 } boolean;
15
16 void CreateStack(StackType *Stack);
17 boolean FullStack(StackType Stack);
18 boolean EmptyStack(StackType Stack);
19 void Push(StackType *Stack, StackElementType Item);
20 void Pop(StackType *Stack, StackElementType *Item);
21 void TraverseStack(StackType Stack);
22 void FilterStack(StackType *Stack, StackElementType Item);
23
24 int main()
25 {
26     StackType myStack;
27     int plithos,i;
28     StackElementType item,deleteItem;
29     CreateStack(&myStack);
30
31
32     /*????????? ??????? ?????????? (???????? ?) ???
33     ??????? ?????????????? ????? ?? ??????? ?? ??????
34     ??????? ?????????????? ??? 20.*/
35     do{
36         printf("Dwse plithos stixeion:");
37         scanf("%d",&plithos);
38         if(plithos < 0 || plithos > StackLimit)
39             printf("To plithos prepei na einai thetikos mikroteros tou 20.");
40     }while(plithos < 0 || plithos > StackLimit);
41
42     /*????????? ?????????? ??? ?????????? ??? ???? ??????(???????? ?).*/
43     for(i=0; i < plithos; i++)
44     {
45         printf("Dwse to %do stoixeio:",i+1);
46         scanf("%d",&item);
47         Push(&myStack,item);
48     }
49
50     /*????????? ?????????? ??? ??????????(???????? ?).*/
51     printf("Dwse to stoixeio pou thes na diagrapseis:");
52     scanf("%d",&deleteItem);
53
54     /*????? ???????????? FilterStack ??? ??? ?????????? ??? ?????????? ??? ??? ???????*/
55     FilterStack(&myStack,deleteItem);
56
57     return 0;
58 }
59
60 void CreateStack(StackType *Stack)
61 {
62     Stack -> Top = -1;
63 }
64
65 boolean EmptyStack(StackType Stack)
66 {

```

```

67     return (Stack.Top == -1);
68 }
69
70 boolean FullStack(StackType Stack)
71 {
72     return (Stack.Top == (StackLimit - 1));
73 }
74
75 void Push(StackType *Stack, StackElementType Item)
76 {
77     if (!FullStack(*Stack)) {
78         Stack -> Top++;
79         Stack -> Element[Stack -> Top] = Item;
80     } else
81         printf("Full Stack...");
82 }
83
84 void TraverseStack(StackType Stack)
85 {
86     int i;
87     printf("\nplithos sto stack %d\n",Stack.Top+1);
88     for (i=0;i<=Stack.Top;i++) {
89         printf("%d, ",Stack.Element[i]);
90     }
91     printf("\n");
92 }
93
94 void Pop(StackType *Stack, StackElementType *Item)
95 {
96     if (!EmptyStack(*Stack)) {
97         *Item = Stack -> Element[Stack -> Top];
98         Stack -> Top--;
99     } else
100         printf("Empty Stack...");
101 }
102
103 void FilterStack(StackType *Stack,StackElementType Item)
104 {
105     int i;
106     StackType TempStack;
107
108     /*?????????? ?????????? ???????? */
109     CreateStack(&TempStack);
110
111     /*????????? ??? ??????? ?????????????????? ??? ?????????? TraverseStack */
112     TraverseStack(*Stack);
113
114     /*????????? ?????????? ??? ???? ?????? ?????? ?? ?????? ?? ?????????? ??? ???? ?????? ???
115     "?????????" ??? ???? ??? ?????????? ??? ?????????? ??? ???????? */
116     for(i=(*Stack).Top;(*Stack).Element[i] != Item; i--)
117     {
118
119         Push(&TempStack,(*Stack).Element[i]);
120         Pop(&(*Stack),&(*Stack).Element[i]);
121
122         TraverseStack(*Stack);
123         TraverseStack(TempStack);
124     }
125
126     /*????????? ??? ?????????? ??? ??????? ? ???????? */
127     Pop(&(*Stack),&(*Stack).Element[i]);
128
129     TraverseStack(*Stack);
130     TraverseStack(TempStack);
131
132     /*?????????? ??? ?????????? ??? ??????????? ?????????? ??? ?????? ??? ?????? */

```

```
133     for(i=TempStack.Top; i >= 0; i--)
134     {
135         Push(&(*Stack),TempStack.Element[i]);
136         Pop(&TempStack,&TempStack.Element[i]);
137
138         TraverseStack(TempStack);
139         TraverseStack(*Stack);
140     }
141
142 }
```