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

#define StackLimit 20

typedef int StackElementType;
typedef struct  {
    int Top;
    StackElementType Element[StackLimit];
} StackType;

typedef enum {
    FALSE, TRUE
} boolean;

void CreateStack(StackType *Stack);
boolean FullStack(StackType Stack);
boolean EmptyStack(StackType Stack);
void Push(StackType *Stack, StackElementType Item);
void Pop(StackType *Stack, StackElementType *Item);
void TraverseStack(StackType Stack);
void FilterStack(StackType *Stack,StackElementType Item);

int main()
{
    StackType myStack;
    int plithos,i;
    StackElementType item,deleteItem;
    CreateStack(&myStack);


    /*???????? ??????? ????????? (??????? ?) ???
      ??????? ??????????? ???? ?? ?????? ?? ?????
      ??????? ?????????? ??? 20.*/
    do{
        printf("Dwse plithos stixeion:");
        scanf("%d",&plithos);
        if(plithos < 0 || plithos > StackLimit)
            printf("To plithos prepei na einai thetikos mikroteros tou 20.");
    }while(plithos < 0 || plithos > StackLimit);

    /*???????? ????????? ??? ???????? ???? ???? ??????(??????? ?).*/
    for(i=0; i < plithos; i++)
    {
       printf("Dwse to %do stoixeio:",i+1);
       scanf("%d",&item);
       Push(&myStack,item);
    }

    /*???????? ????????? ???? ????????(??????? ?).*/
    printf("Dwse to stoixeio pou thes na diagrapseis:");
    scanf("%d",&deleteItem);

    /*????? ?????????? FilterStack ??? ??? ???????? ??? ????????? ??? ??? ??????.*/
    FilterStack(&myStack,deleteItem);

    return 0;
}

void CreateStack(StackType *Stack)
{
    Stack -> Top = -1;
}

boolean EmptyStack(StackType Stack)
{
```

```
    return (Stack.Top == -1);
}

boolean FullStack(StackType Stack)
{
    return (Stack.Top == (StackLimit - 1));
}

void Push(StackType *Stack, StackElementType Item)
{
    if (!FullStack(*Stack)) {
        Stack -> Top++;
        Stack -> Element[Stack -> Top] = Item;
    } else
        printf("Full Stack...");
}

void TraverseStack(StackType Stack)
{
    int i;
    printf("\nplithos sto stack %d\n",Stack.Top+1);
    for (i=0;i<=Stack.Top;i++) {
        printf("%d, ",Stack.Element[i]);
    }
    printf("\n");
}

void Pop(StackType *Stack, StackElementType *Item)
{
    if (!EmptyStack(*Stack)) {
        *Item = Stack -> Element[Stack -> Top];
        Stack -> Top--;
    } else
        printf("Empty Stack...");
}

void FilterStack(StackType *Stack,StackElementType Item)
{
   int i;
   StackType TempStack;

   /*?????????? ?????????? ???????.*/
   CreateStack(&TempStack);

   /*???????? ??? ??????? ?????????????? ??? ????????? TraverseStack.*/
   TraverseStack(*Stack);

   /*??????? ????????? ???? ????????? ?????? ????? ?? ?????? ?? ???????? ???? ???????? ???
    "??????????" ???? ???? ??? ????????? ??? ????????? ???? ??????.*/
   for(i=(*Stack).Top;(*Stack).Element[i] != Item; i--)
   {

      Push(&TempStack,(*Stack).Element[i]);
      Pop(&(*Stack),&(*Stack).Element[i]);

      TraverseStack(*Stack);
      TraverseStack(TempStack);
   }

   /*???????? ??? ????????? ??? ??????? ? ???????.*/
   Pop(&(*Stack),&(*Stack).Element[i]);

   TraverseStack(*Stack);
   TraverseStack(TempStack);

   /*????????? ??? ????????? ??? ?????????? ??????? ???? ?????? ??? ??????.*/
```

```
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 }
```