**Insertion Sort**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 4 | 3 | 2 | 10 | 12 | 1 | 5 | 6 |
| 3 | 4 | 2 | 10 | 12 | 1 | 5 | 6 |
| 2 | 3 | 4 | 10 | 12 | 1 | 5 | 6 |
| 2 | 3 | 4 | 10 | 12 | 1 | 5 | 6 |
| 2 | 3 | 4 | 10 | 12 | 1 | 5 | 6 |
| 1 | 2 | 3 | 4 | 10 | 12 | 5 | 6 |
| 1 | 2 | 3 | 4 | 5 | 10 | 12 | 6 |
| 1 | 2 | 3 | 4 | 5 | 6 | 10 | 12 |

Each number is considered in turn, checked against all numbers before it, and inserted in the position needed.

Maximum shifts = minimum checks

Maximum checks = minimum shifts

For each position (n), checks + shifts = n

Total complexity = n2

#include <stdio.h>  
  
int main()  
{  
 int ar[8] = {4, 3, 2, 10, 12, 1, 5, 6};  
 int n, idx, i, k, o, j;  
  
 for (i=0; i<8; i++)  
 {  
 idx = -1;  
 for (j=0; j<i; j++)  
 {  
 if (ar[i] < ar[j])  
 {  
 o = ar[i];  
 idx = j;  
 break;  
 }  
 }  
 if (idx != -1)  
 {  
 for (k=i; k>idx; k--) ar[k] = ar[k-1];  
 ar[idx] = o;  
 }  
 }  
 for (i=0; i<8; i++) printf("%d ", ar[i]);  
}

C