```
#include <stdio.h>

int main()
{
int n, i, j, temp;
int arr[64];
printf("Enter number of elements\n");
scanf("%d", &n);
printf("Enter %d integers\n", n);
for (i = 0; i < n; i++)
{
scanf("%d", &arr[i]);
}
for (i = 1 ; i <= n - 1; i++)
{
j = i;
while ( j > 0 && arr[j-1] > arr[j])
{
temp = arr[j];
arr[j] = arr[j-1];
arr[j-1] = temp;
j--;
}
}
printf("Sorted list in ascending order:\n");
for (i = 0; i <= n - 1; i++)
{
printf("%d\n", arr[i]);
}
return 0;
}
```

In general case the both expressions are invalid. For example the first expression is invalid because there is no such operation as + for pointers or iterators. The second expression is invalid in case when non-random access iterators are used. For example when bidirectional iterators are used.

So the correct construction in C++ will look the following way

mid = std::next(beg, std::distance(beg, end) / 2);
Because left + right may overflow. Which then means you get a result that is less than left. Or far into the negative if you are using signed integers.

So instead they take the distance between left and right and add half of that to left. This is only a single extra operation to make the algorithm more robust.