```
//直接插入排序
void InserSort(int R[], int n)
{
    int i, j;
    int temp;
    for(i = 1; i < n; ++i)
    {
        temp = R[i];
        j = i - 1;
        while(j >= 0 && temp < R[j])
        {
            R[j + 1] = R[j];
            --j;
        }
        R[j + 1] = temp; // 插入
    }
}

// 折半查找排序
void InsertSort(int A[]){
    int n = A.size();
    int low, high;
    for(int i = 2; i < n; ++i){
        A[0] = A[i];
        low = 1;
        high = i - 1;
        while(low < high){
            mid = (low + high) / 2;
            if(A[mid] > A[0])
                high = mid - 1;
            else
                low = mid + 1;
        }
        for(j = i - 1; j >= high + 1; --j)
            A[j + 1] = A[j];
        A[high + 1] = A[0];
    }
}
```

```
// ShellSort
/*
    记录前后位置的增量是dk，不是1
    A[0]暂存单元，不是哨兵，当 j<=0 时，插入位置已到
*/
void ShellSort(int A[]){
    int n = A.size();
    for(int dk = n / 2; dk >= 1; dk = dk / 2)
        for(int i = dk + 1; i <= n; ++i)
            if(A[i] < A[i - dk]){
                A[0] = A[i];
                for(int j = i - dk; j > 0 && A[0] < A[j]; j -= dk)
                    A[j + dk] = A[j];
                A[j + dk] = A[0];
            }
}

// 冒泡排序
void BubbleSort(int R[]){
    int n = R.size();
    for(int i = n - 1; i >= 1; --i){
        int flag = 0;
        for(j = 1; j <=i; ++j){
            if(R[j - 1] > R[j]){
                int temp = R[j];
                R[j] = R[j - 1];
                R[j - 1] = temp;
                flag = 1;
            }
        }
        if(flag == 0)
            return;
    }
}
```