```
1. InsertSort
void InitSort(int *st,int n)
{
        for (int i = 2; i <= n; ++i)
        {
                st[0] = st[i];
                int j;
                for (j = i-1; st[j] > st[0]; --j)
                        st[j+1] = st[j];
                st[j+1] = st[0];
        }

        return ;
}

2. ShellSort
void ShellSort(int *st, int n)
{
    for (int gap = n>>1; gap > 0; gap >>= 1)
    {
        for (int i = gap<<1; i <= n; i += gap)
        {
            st[0] = st[i];
            int j;
            for (j = i-gap; st[j] > st[0]; j -= gap)
                st[j+gap] = st[j];
            st[j+gap] = st[0];
        }
    }

    return ;
}

3.QuickSort
inline void swap(int &a, int &b)
{
    int tmp = a;
    a = b;
    b = tmp;
}

void QuickSort(int *st, int left, int right)
{
        int last;
```

```
45          if (left >= right)      return ;
46          swap(st[left], st[(left+right)>>1]);
47          last = left;
48          for (int i = left+1; i <= right; ++i)
49              if (st[i] < st[left])       swap(st[++last], st[i]);
50          swap(st[left], st[last]);
51          QuickSort(st, left, last-1);
52          QuickSort(st, last+1, right);
53  }
```