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

#define N 15

int partion(int *data, int low, int high);
int quick_sort(int *data, int low, int high);
int compare(const void *p1, const void *p2);

int main(int argc, const char *argv[])
{
	int data[N] = {0};
	int i;

	srandom(10);

	for (i = 0; i < N; i++) {
		data[i] = random() % 100;
	}

	for (i = 0; i < N; i++) {
		printf("%d ", data[i]);
	}
	puts("");

	//quick_sort(data, 0, N-1);
	qsort(data, N, sizeof(int), compare);

	for (i = 0; i < N; i++) {
		printf("%d ", data[i]);
	}
	puts("");

	return 0;
}

int partion(int *data, int low, int high) {
	int temp = data[low];

	while (low < high) {
		while (low < high && temp <= data[high]) {
			high--;
		}
		data[low] = data[high];

		while (low < high && temp >= data[low]){
			low++;
		}
		data[high] = data[low];
	}

	data[low] = temp;

	return low;
}

int quick_sort(int *data, int low, int high) {
	int t;
```

赋值数组 (随机生成) (for, srandom)

→ 打印数组

排序数组 → data N sizeof compare

打印数组

→ temp 给左边，右边有小的，就交换，然后给右边，左边有大的，就交换，然后直至 low >= high (temp = data[low])

low 上放 temp (最后假设 low 上是 temp 的，但 low == high 给打断，此时 low 还在 low，不一定是这个数。

[high]>=的, [low小的]<=的.)
(low只可能等)

```
    if (data == NULL) {
        return -1;
    }

    if (low >= high)
        return 0;

    t = partion(data, low, high);
    quick_sort(data, low, t-1);
    quick_sort(data, t+1, high);

    return 0;
}

int compare(const void *p1, const void *p2) {
    return (*(const int *)p1 - *(const int *)p2);
}
```

→ 把中位数下标

下一半排

上一半排.(要有返回)(t所向它在中间不排)

→ 强转地址,取址运算(返回)