El algoritmo Quicksort, creado por Tony Hoare en 1959, es un eficiente y ampliamente utilizado algoritmo de ordenamiento basado en la técnica de divide y vencerás.

Explicación de su funcionamiento:
Elección del pivote: Se selecciona un elemento del array como pivote. Hay diferentes formas de elegir el pivote, como seleccionar el primer elemento, el último, uno al azar o el del medio.

Partición del array: Se reorganizan los elementos del array de tal manera que los elementos menores que el pivote se colocan a la izquierda y los elementos mayores que el pivote se colocan a la derecha.

Ordenación de las particiones: Se repite el proceso de partición de forma recursiva en cada subarray hasta que cada subarray contenga un solo elemento. En este punto, los elementos ya están ordenados.

Combinación de los subarrays: Finalmente, se combinan los elementos para formar un array ordenado.

Codigos de ejemplo:

Función qs:

```
void qs(int lista[],int limite_izq,int limite_der) {
   int izq, der, temporal, pivote;

   izq = limite_izq;
   der = limite_der;
   pivote = lista[(izq + der) / 2];

   do {
      while (lista[izq] < pivote && izq < limite_der) izq++;
      while (pivote < lista[der] && der > limite_izq) der--;
      if (izq <= der) {
         temporal = lista[izq];
         lista[izq] = lista[der];
         lista[der] = temporal;
         izq++;
         der--;
      }
   } while (izq <= der);
```

```
  if (limite_izq < der) { qs(lista, limite_izq, der); }

  if (limite_der > izq) { qs(lista, izq, limite_der); }

}
```

Esta es la función principal que implementa el algoritmo QuickSort. Toma un array y los índices de inicio y fin del array o subarray.

```
void quicksort(int lista[], int n) {

  qs(lista, 0, n - 1);

}
```

Toma un array y el tamaño del array, y llama a la función qs() con los índices de inicio y fin apropiados.

Codigo Entero:

```
#include <stdio.h>

void qs(int lista[],int limite_izq,int limite_der)
{
    int izq,der,temporal,pivote;

    izq=limite_izq;
    der = limite_der;
    pivote = lista[(izq+der)/2];

    do{
        while(lista[izq]<pivote && izq<limite_der)izq++;
        while(pivote<lista[der] && der > limite_izq)der--;
        if(izq <=der)
        {
            temporal= lista[izq];
            lista[izq]=lista[der];
            lista[der]=temporal;
            izq++;
            der--;

        }

    }while(izq<=der);
    if(limite_izq<der){qs(lista,limite_izq,der);}
    if(limite_der>izq){qs(lista,izq,limite_der);}

}

void quicksort(int lista[],int n)
{
    qs(lista,0,n-1);
}

int main(int argc, const char * argv[])
{

    int lista[] ={100,56,0,1,-45,2,46,5,9,6,67,23,5};
    int size = sizeof(lista)/sizeof(int);

    printf("Lista Desordenada \n");
```

```
    for (int i=0; i<size; i++) {
        printf("%d",lista[i]);
        if(i<size-1)
            printf(",");
    }

    printf("\n");
    quicksort(lista,size);

    printf("Lista Ordenada \n");
    for (int i=0; i<size; i++) {
        printf("%d",lista[i]);
        if(i<size-1)
            printf(",");
    }

    return 0;
}
```

En resumen, QuickSort es un algoritmo de ordenamiento poderoso y versátil, es uno de los algoritmos de ordenamiento más rápidos para arrays grandes. Además, es un algoritmo que no requiere un espacio adicional significativo para realizar la ordenación.

Bibliografía:

¿Qué es el algoritmo QuickSort? (asimov.cloud)

Algoritmo Quicksort: implementación de C++, Java y Python (techiedelight.com)

Implementando el algoritmo QuickSort (genbeta.com)

Quicksort en C – Algoritmo de ordenamiento | Codigoprogramacion