```
    {
      printf("a[%d] = ",);
      scanf("%d", a+i);
    }
  }
```

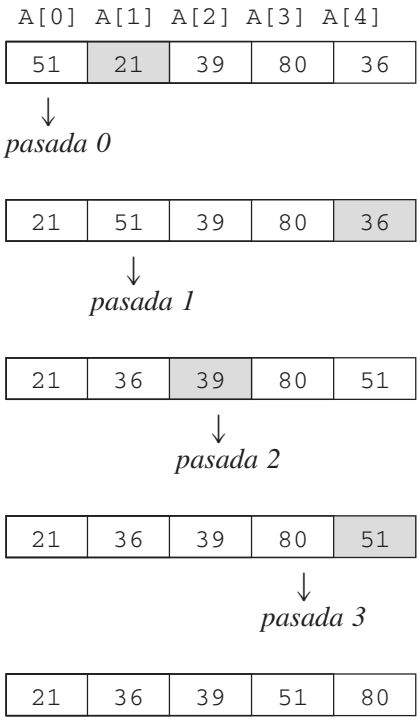
6.4. ORDENACIÓN POR SELECCIÓN

Considérese el algoritmo para ordenar un array `A` de enteros en orden ascendente, es decir, del número más pequeño al mayor. Es decir, si el array `A` tiene *n* elementos, se trata de ordenar los valores del array de modo que el dato contenido en `A[0]` sea el valor más pequeño, el valor almacenado en `A[1]` el siguiente más pequeño, y así hasta `A[n-1]`, que ha de contener el elemento de mayor valor. El algoritmo se apoya en sucesivas pasadas que intercambian el elemento más pequeño sucesivamente con el primer elemento de la lista, `A[0]` en la primera pasada. En síntesis, se busca el elemento más pequeño de la lista y se intercambia con `A[0]`, primer elemento de la lista.

```
A[0]  A[1]  A[2]....  A[n-1]
```

Después de terminar esta primera pasada, el frente de la lista está ordenado y el resto de la lista `A[1], A[2]...A[n-1]` permanece desordenado. La siguiente pasada busca en esta lista desordenada y *selecciona* el elemento más pequeño, que se almacena entonces en la posición `A[1]`. De este modo los elementos `A[0]` y `A[1]` están ordenados y la sublista `A[2], A[3]...A[n-1]` desordenada; entonces, se selecciona el elemento más pequeño y se intercambia con `A[2]`. El proceso continúa *n* – 1 pasadas y en ese momento la lista desordenada se reduce a un elemento (el mayor de la lista) y el array completo ha quedado ordenado.

Un ejemplo práctico ayudará a la comprensión del algoritmo. Consideremos un array `A` con 5 valores enteros `51, 21, 39, 80, 36`:



Pasada 0. Seleccionar `21`
Intercambiar `21` y `A[0]`

Pasada 1. Seleccionar `36`
Intercambiar `36` y `A[1]`

Pasada 2. Seleccionar `39`
Intercambiar `39` y `A[2]`

Pasada 3. Seleccionar `51`
Intercambiar `51` y `A[3]`

Lista ordenada

6.4.1. Algoritmo de selección

Los pasos del algoritmo son:

1. Seleccionar el elemento más pequeño de la lista `A`; intercambiarlo con el primer elemento `A[0]`. Ahora la entrada más pequeña está en la primera posición del vector.
2. Considerar las posiciones de la lista `A[1], A[2], A[3]...`, seleccionar el elemento más pequeño e intercambiarlo con `A[1]`. Ahora las dos primeras entradas de `A` están en orden.
3. Continuar este proceso encontrando o seleccionando el elemento más pequeño de los restantes elementos de la lista, intercambiándolos adecuadamente.

6.4.2. Codificación en C del algoritmo de selección

La función `ordSeleccion()` ordena una lista o vector de números reales de `n` elementos. En la pasada `i`, el proceso de selección explora la sublista `A[i]` a `A[n-1]` y fija el índice del elemento más pequeño. Después de terminar la exploración, los elementos `A[i]` y `A[indiceMenor]` intercambian las posiciones.

```
  /*
    ordenar un array de n elementos de tipo double
    utilizando el algoritmo de ordenación por selección
  */

void ordSeleccion (double a[], int n)
{
  int indiceMenor, i, j;
            /* ordenar a[0]..a[n-2] y a[n-1] en cada pasada */
  for (i = 0; i < n-1; i++)
  {
            /* comienzo de la exploración en índice i */
    indiceMenor = i;
            /* j explora la sublista a[i+1]..a[n-1] */
    for (j = i+1; j < n; j++)
      if (a[j] < a[indiceMenor])
        indiceMenor = j;
            /* sitúa el elemento más pequeño en a[i] */
    if (i != indiceMenor)
    {
      double aux = a[i];
      a[i] = a[indiceMenor];
      a[indiceMenor] = aux ;
    }
  }
}
```

El análisis del algoritmo de selección es sencillo y claro, ya que requiere un número fijo de comparaciones que sólo dependen del tamaño de la lista o vector (array) y no de la distribución inicial de los datos. En el Apartado 2.6.1 se realizó un estudio de la complejidad de este algoritmo.