8. Vectores (arrays)

Fundamentos de Informática

Especialidad de Electrónica – 2009-2010

Ismael Etxeberria Agiriano

16/12/2009

Escuela Universitaria de Ingeniería Vitoria-Gasteiz
Ingeniaritzako Unibertsitate Eskola Vitoria-Gasteiz

Universidad del País Vasco
Euskal Herriko Unibertsitatea

Índice

8. Vectores (arrays)

1. Números aleatorios
2. Serie ordenada
3. Números de la loto
4. Inserción ordenada

1. Programa de números aleatorios

- Vamos a ver un programa sencillo utilizado por un "profesor arbitrario" para calificar un trabajo, consistente en mostrar **una nota cualquiera** entre `0.0` y `10.0` (distinta cada vez).
- Para ello vamos a utilizar la función (o macro) `random` definida en la `stdlib.h`, que tiene la forma siguiente:

```
int random (int n);
```

- `random` devuelve un número aleatorio entre `0` y `n-1`
- Cada vez que llamamos a `random` obtenemos un nuevo número
- Detrás de `random` tenemos una serie de números pseudo-aleatoria de **distribución uniforme** basada en la función `rand` (a la que no invocaremos directamente)

Series pseudo-aleatorias

- Una distribución es **uniforme** cuando todos los elementos tienen la misma probabilidad de aparecer, como en el lanzamiento de un dado o una moneda no trucados
- Detrás de esta serie pseudo-aleatoria tenemos un número entero (por ejemplo, de 32 bits) y si es de buena calidad aparecerán las 2^{32} posibilidades distintas pero en un orden "caótico"
- A un número dado en esta serie siempre le sigue el mismo número
- Para inicializar el primer valor (**semilla**) utilizaremos el reloj del sistema con la función (o macro) `randomize` (que requiere la inclusión del fichero `time.h`) de la forma:

```
void randomize (void);
```

Programa 1: código

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

void main (void)
{
  double nota;

  randomize (); /* Inicializa semilla con el reloj del sistema */
  nota = random (101) / 10.0;
  printf ("Nota final: %4.1f\n", nota);
}
```

Para forzar división real

2. Programa de serie ordenada

- Vamos a modificar el programa para que nos muestre una **serie ordenada de números aleatorios** entre `0.0` y `10.0`
- El programa se dividirá en cuatro partes:
  1. **Inicializar la semilla** de números aleatorios (inmediata)
  2. **Inicializar el vector** con números aleatorios:
     ```
     void inicializar (double v[], int n);
     ```
  3. **Ordenar** el vector:
     ```
     void ordenar (double v[], int n);
     ```
  4. **Mostrar** los elementos del vector resultante:
     ```
     void mostrar (double v[], int n);
     ```

Programa 2: Programa principal

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

void inicializar (double v[], int n);
void ordenar (double v[], int n);
void mostrar (double v[], int n);

void main (void)
{
  double vector[10];

  randomize ();
  inicializar (vector, 10);
  ordenar (vector, 10);
  mostrar (vector, 10);
}
```

Programa 2: inicializar y mostrar

```
void inicializar (double v[], int n)
{
  int i;
  for (i = 0; i < n; i++)
    v[i] = (double) random (101) / 10.0;
}

void mostrar (double v[], int n)
{
  int i;
  for (i = 0; i < n; i++)
    printf ("%4.1f\n", v[i]);
}
```

Al menos uno de los dos para forzar división real

Algoritmo de ordenación

- Hay muchos algoritmos para ordenar de manera creciente un vector de `n` elementos
- Vamos a utilizar uno sencillo, consistente en colocar en la primera posición el elemento más pequeño, luego el siguiente, y así sucesivamente hasta la penúltima celda:

  `Para` `i` **`de`** `0` **`a`** `n-2` **`hacer`**

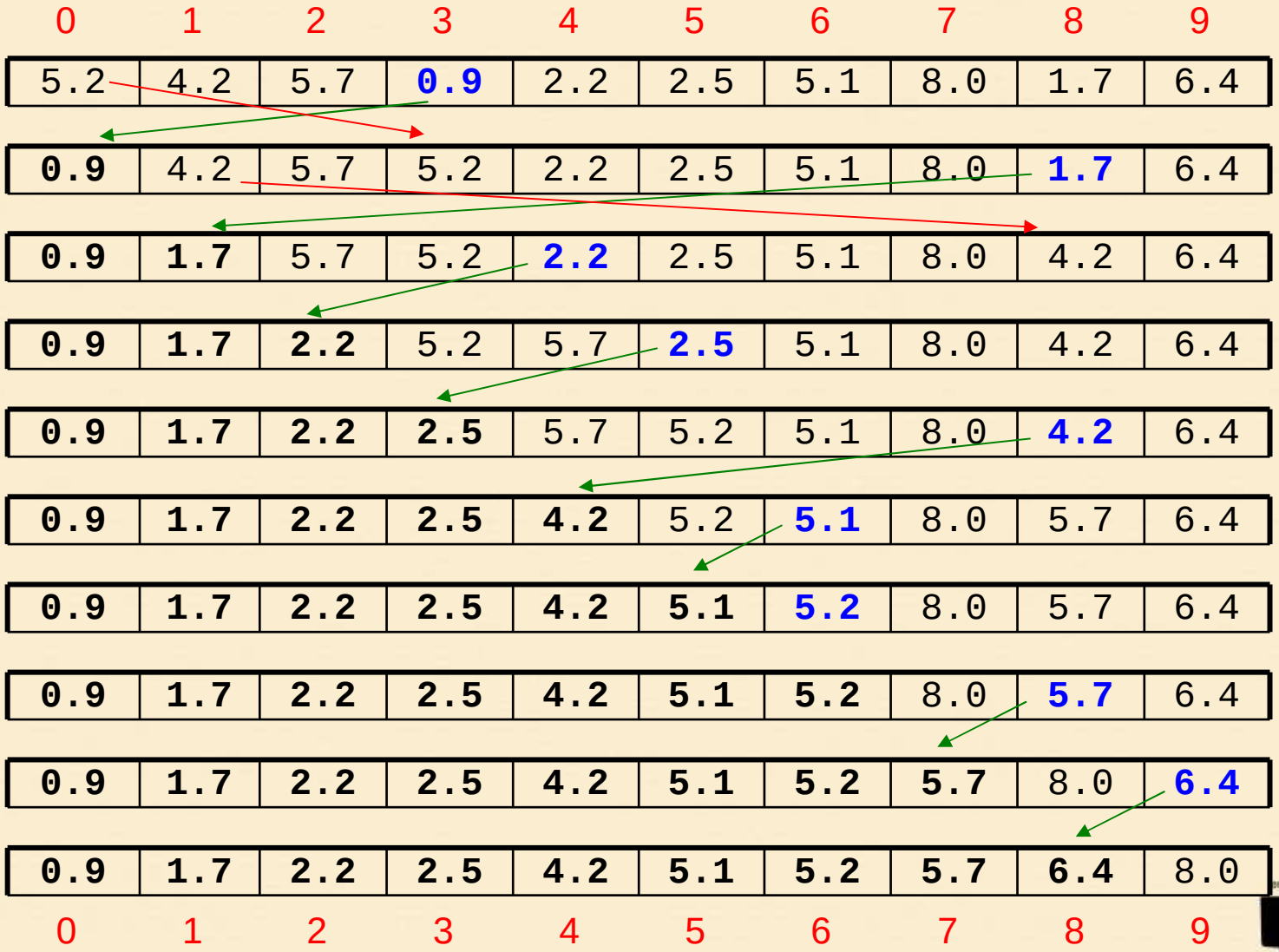
  Intercambiar el elemento en la posición `i`-ésima

  con el más pequeño en las posiciones de `i` a `n-1`

- Lo mejor es verlo con un ejemplo. Ordenemos el vector:

0	1	2	3	4	5	6	7	8	9
5.2	4.2	5.7	0.9	2.2	2.5	5.1	8.0	1.7	6.4

Ejemplo de ordenación



Programa 2: ordenar

```
void ordenar (double v[], int n)
{
  int i, j, min; /* Diversos índices */
  double tmp; /* Variable temporal, para el intercambio */

  for (i = 0; i < n - 1; i++) {/* Para i de 0 a n – 2 */
    min = i; /* Si no hay otro menor el candidato es i */
    for (j = i + 1; j < n; j++)
      if (v[j] < v[min]) /* Si j es menor pasa a ser el menor */
        min = j;
    if (min != i) { /* Si i no es el más pequeño intercambiar v[i] y v[min] */
      tmp = v[i];
      v[i] = v[min];
      v[min] = tmp;
    }
  }
}
```

3. Programa de números de la loto

- Vamos a escribir un programa similar al anterior para generar los **números de la loto**
- Las modificaciones respecto al programa anterior son que se trata de números naturales (`int`) del 1 al 49 y que **no se pueden repetir**
- Para asegurarnos de que los números no se repiten verificaremos que cada nuevo número no se encuentre ya en la lista, en cuyo caso lo descartaremos (`do-while`):

```
Hacer
        num = Generar un número aleatorio entre 1 y 49
mientras (num se encuentre en el vector)
```

Programa 3: inicializar_loto

```
void inicializar_loto (int v[], int n)
{
  int i, j;
  int num;

  for (i = 0; i < n; i++) {
    do {
      num = random (49) + 1; /* entre 1 y 49 */
      for (j = 0; j < i; j++)
        if (num == v [j])
          break;
    } while (j < i);
    v [i] = num;
  }
}
```

4. Programa de inserción ordenada

- Una alternativa a la solución anterior es la de **introducir directamente** cada número **en su posición ordenada**
- Para ello iremos construyendo la lista de manera ordenada
- Para mirar si el número se encuentra en la lista podemos buscarlo de manera secuencial
  - Hay algoritmos de búsqueda eficientes que van dividiendo el espacio de búsqueda en dos pero salen de nuestros objetivos
- Insertar un elemento en su posición implica desplazar una posición a la derecha a todos los elementos a partir de esa posición

Programa 4: inicializar_ordenando

```
int posicion (int num, int v[], int n);
void desplazar (int v[], int n);

void inicializar_ordenando (int v[], int n)
{
  int i, j;
  int num;

  for (i = 0; i < n; i++) {
    do {
      num = random (49) + 1; /* entre 1 y 49 */
      j = posicion (num, v, i); /* devuelve -1 si ya se encuentra */
      if (j != -1)
        desplazar (&v[j], i - j); /* desplazar a partir de j */
    } while (j == -1);
    v [j] = num;
  }
}
```

Programa 4: posición y desplazar

```
int posicion (int num, int v[], int n)
{
  int i;
  i = 0;
  while (i < n && num > v[i])
    i++; /* Posicionar i */
  if (i < n && v[i] == num)
    i = -1;
  return i;
}

void desplazar (int v[], int n)
{
  while (n > 1) {
    v[n] = v[n-1];
    n--;
  }
}
```

Escuela Universitaria de Ingeniería Vitoria-Gasteiz
Ingeniaritzako Unibertsitate Eskola Vitoria-Gasteiz

eman ta zabal zazu

Universidad del País Vasco
Euskal Herriko Unibertsitatea