Programación 1

Tema 9

Estructuración indexada de datos

Escuela de
Ingeniería y Arquitectura
Universidad Zaragoza

Problema

- Para resolver un problema relativo al cambio climático, es necesario manejar la información de las temperaturas medias registradas de forma mensual durante un determinado año en una determinada localidad.
- ¿Cómo podemos representar esta información?

Una (muy mala) solución

```
/*
 * Pre:  Todas las temperaturas son mayores que -273,15 ºC
 * Post: Ha devuelto la temperatura media anual a partir de las
 *       temeraturas medias mensuales
 */
double calcularTemperaturaMedia(
       double temperaturaEnero, double temperaturaFebrero,
       double temperaturaMarzo, double temperaturaAbril,
       double temperaturaMayo, double temperaturaJunio,
       double temperaturaJulio, double temperaturaAgosto,
       double temperaturaSeptiembre, double temperaturaOctubre,
       double temperaturaNoviembre, double temperaturaDiciembre) {
   return (temperaturaEnero + temperaturaFebrero + temperaturaMarzo
         + temperaturaAbril + temperaturaMayo + temperaturaJunio
         + temperaturaJulio + temperaturaAgosto + temperaturaSeptiembre
         + temperaturaOctubre + temperaturaNoviembre
         + temperaturaDiciembre) / 12.0;
}
```

Vectores o tablas

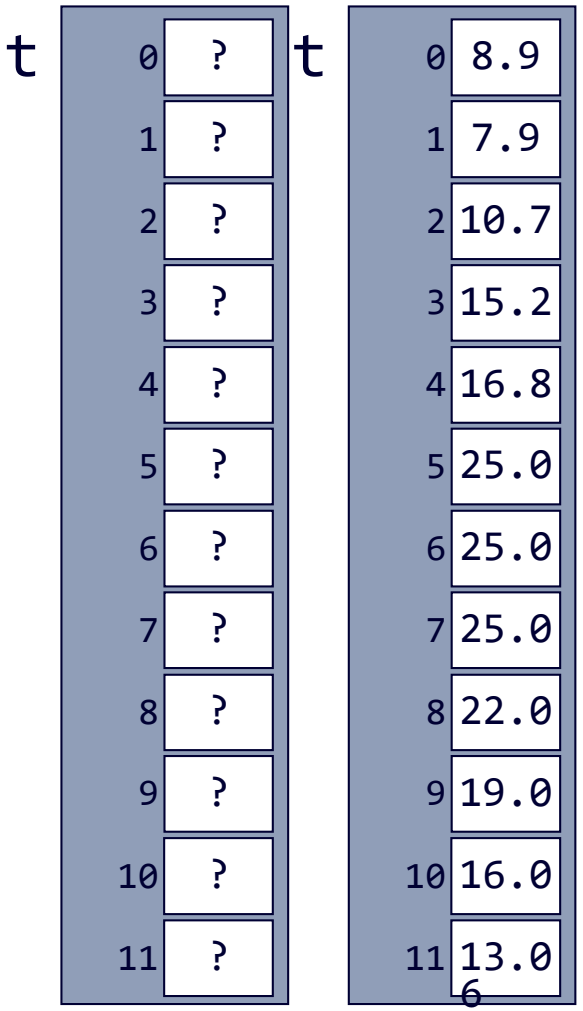
- Colección de un número concreto de datos de un mismo tipo
- Indexados por uno o más índices
- Operaciones
  - Acceso a componentes

Vectores

- Sintaxis declaración
  - <declaraciónVector> ::= <tipo> <identificador> “[”<constanteEntera>“]” [“=” “{” <listaInicialización> “}”] “;”
  - <listaInicialización> ::= {<dato> “,”}

Vector

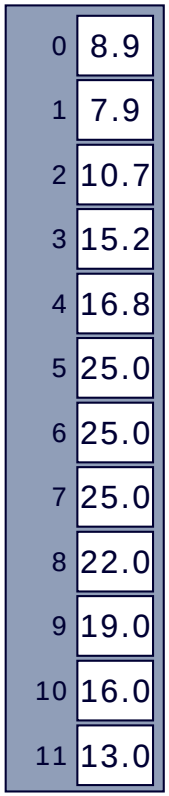
```
const int NUM_MESES = 12;
double t[NUM_MESES];
t[0] = 8.9;
t[1] = t[0] - 1.0;
int m = 2;
t[m] = 10.7;
t[m + 1] = 15.2;
t[4] = t[0] + t[1];
m = 5;
while (m < 8) {
    t[m] = 25.0;
    m++;
}
for (int n = 8; n < NUM_MESES; n++) {
    t[n] = t[n - 1] - 3.0;
}
```



Vectores

```
double t[] = {8.9, 7.9, 10.7,
              15.2, 16.8, 25.0, 25.0,
              25.0,  2.0, 19.0, 16.0,
              13.0,
              };
```

t



Índices fuera de los límites

```
const int DIMENSION = 3;
int v[DIMENSION] = {0, 0, 0};

cout << v[0] << endl;
v[2] = 2;
cout << v[3] << endl;                    // ¿?
v[4] = 4;                                // ¿?
cout << v[100] << endl;                  // ¿?
v[-4] = -4;                              // ¿?
cout << v[123456789] << endl;            // ¿?
```

Funciones con vectores

- Solo pueden ser parámetros, no valores devueltos
- Como parámetros podrían pasarse
  - por valor (entrada)
    - `void f(int v[10]);`
    - Normalmente, es muy ineficiente
  - por referencia (entrada y salida)
    - `void g(int w[]);`

Funciones con vectores

- Como parámetros por referencia
  - `void f(int v[]);`
    - Entrada y salida: las componentes del vector `v` pueden ser consultadas y modificadas por la función `f`.
  - `void g(const int w[]);`
    - Parámetro de entrada: las componentes del vector `w` solo pueden ser consultadas por la función g, pero no modificadas.
    - Al no especificar la dimensión, `w` se sigue pasando por referencia (más eficiente que por valor).

Ejemplo

- Función que, dado un vector de temperaturas mensuales, devuelve su media

Cálculo de la temperatura media anual

```
/*
 * Pre:  «t» tiene NUM_MESES componentes
 * Post: Ha devuelto la temperatura media de las temperaturas
 *       almacenadas en «t»
 */
double temperaturaMediaAnual(const double t[]) {
   double sumaTemperaturas = 0.0;
   int i = 0;
   while (i < NUM_MESES) {
      sumaTemperaturas += t[i];
      i++;
   }
   return sumaTemperaturas / NUM_MESES;
}
```

Cálculo de la temperatura media anual (bucle for)

```
/*
 * Pre:  «t» tiene NUM_MESES componentes
 * Post: Ha devuelto la temperatura media de las
 *       temperaturas almacenadas en «t»
 */
double temperaturaMediaAnualFor(const double t[]) {
   double sumaTemperaturas = 0.0;
   for (int i = 0; i < NUM_MESES; i++) {
      sumaTemperaturas += t[i];
   }
   return sumaTemperaturas / NUM_MESES;
}
```

Cálculo de la media

```
/*
 * Pre:  «t» tiene «n» componentes.
 * Post: Ha devuelto el valor medio de los valores
 *       almacenados en las componentes de «t».
 */
double media(const double t[], int n) {
   double suma = 0.0;
   for (int i = 0; i < n; i++) {
      suma += t[i];
   }
   return suma / n;
}
```

Cálculo de la media

```
/*
 * Pre:  «t» tiene «n» componentes.
 * Post: Ha devuelto el valor medio de los valores
 *       almacenados en las componentes de «t».
 */
double media(const double t[], const int n) {
   double suma = 0.0;
   for (int i = 0; i < n; i++) {
      suma += t[i];
   }
   return suma / n;
}
```

Cálculo de la desviación típica

```
/**
 * Pre:  «t» tiene «n» componentes y n > 1.
 * Post: Ha devuelto la desviación típica de los
 *       valores almacenadas en «t»
 */
double desviacionTipica(const double t[], const int n) {
   double suma = 0.0;
   for (int i = 0; i < n; ++i) {
      suma += pow(t[i] - media(t, n), 2);
   }
   return sqrt(suma / (n - 1));
}
```

Cálculo de la desviación típica

```
/**
 * Pre:  «t» tiene «n» componentes y n > 1.
 * Post: Ha devuelto la desviación típica de los
 *       valores almacenadas en «t»
 */
double desviacionTipica(const double t[], const int n) {
   double mediaAritmetica = media(t, n);
   double suma = 0.0;
   for (int i = 0; i < n; i++) {
      suma += pow(t[i] - mediaAritmetica, 2);
   }
   return sqrt(suma / (n - 1));
}
```

Cálculo del máximo

```
/**
 * Pre:  «t» tiene «n» componentes y n > 0.
 * Post: Ha devuelto la valor máximo almacenado en
 *       las componentes del vector «t».
 */
double maximo(const double t[], const int n) {
   double maximo = t[ 0 ];
   for (int i = 1; i < n; i++) {
      if (t[i] > maximo) {
         maximo = t[i];
      }
   }
   return maximo;
}
```

Letra del DNI

TABLA DE LETRAS DEL DNI					
DNI % 23	letra	DNI % 23	letra	DNI % 23	letra
0	T	8	P	16	Q
1	R	9	D	17	V
2	W	10	X	18	H
3	A	11	B	19	L
4	G	12	N	20	C
5	M	13	J	21	K
6	Y	14	Z	22	E
7	F	15	S		

Letra del DNI (mala solución)

```
/*
 * Pre:  dni > 0
 * Post: Ha devuelto la letra del número de
 *       identificación fiscal que corresponde
 *       a un número de documento nacional de
 *       identidad igual a «dni».
 */
char letra(const int dni) {
    int resto = dni % 23;
    if (resto == 0) {
        return 'T';
    }
    else if (resto == 1) {
        return 'R';
    }
    else if (resto == 2) {
        return 'W';
    }
    else if (resto == 3) {
        return 'A';
    }
    ...
```

```
    ...
    else if (resto == 16) {
        return 'Q';
    }
    else if (resto == 17) {
        return 'V';
    }
    else if (resto == 18) {
        return 'H';
    }
    else if (resto == 19) {
        return 'L';
    }
    else if (resto == 20) {
        return 'C';
    }
    else if (resto == 21) {
        return 'K';
    }
    else {
        return 'E';
    }
}
```

Letra del DNI

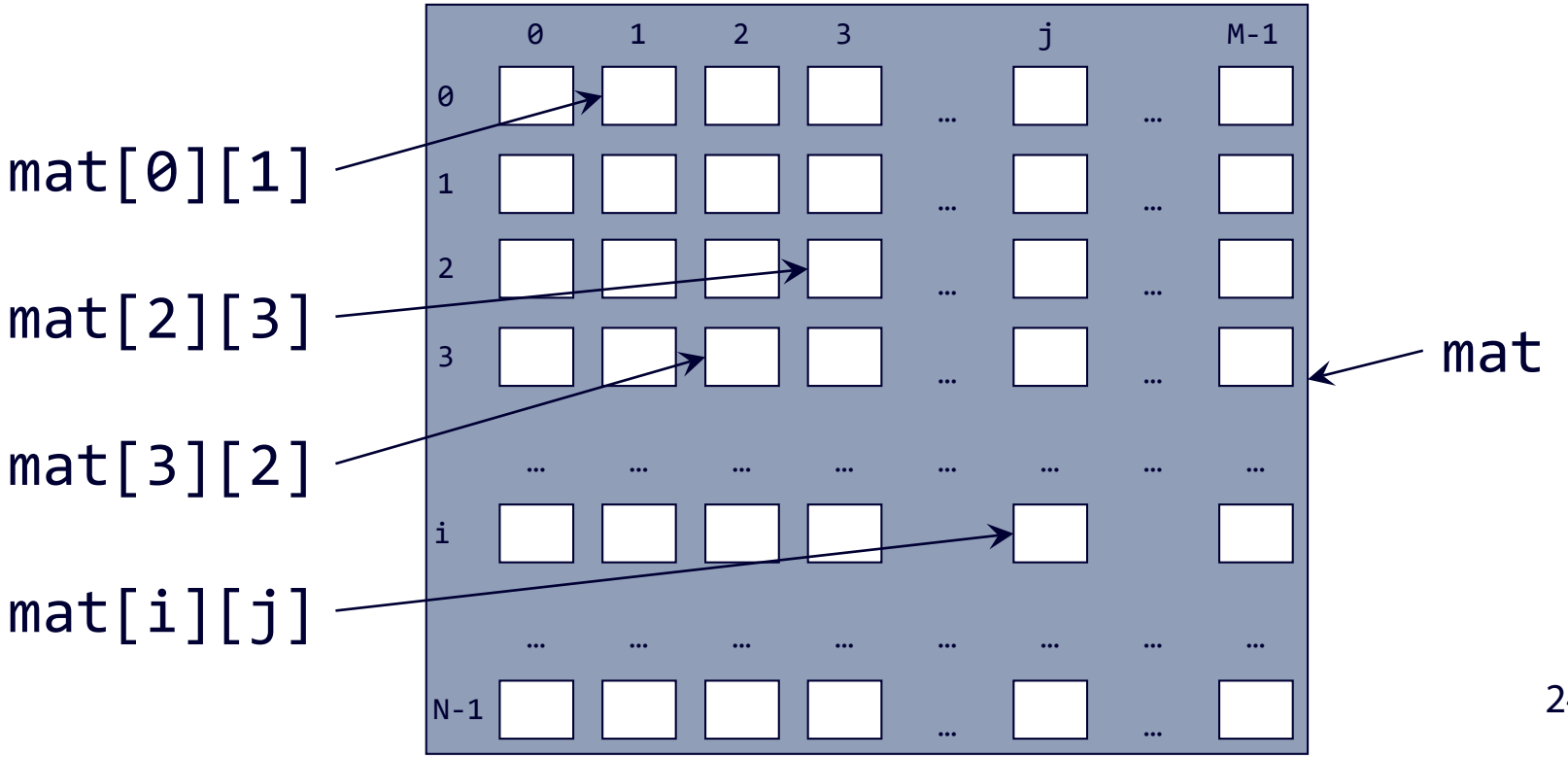
```
/*
 * Pre:  dni > 0
 * Post: Ha devuelto la letra del DNI que corresponde a
 *       un número de DNI igual a «dni».
 */
char letra(const int dni) {
   const int NUM_LETRAS = 23;
   const char TABLA_LETRAS_NIF[NUM_LETRAS] = {'T', 'R',
         'W', 'A', 'G', 'M', 'Y', 'F', 'P', 'D', 'X', 'B',
         'N', 'J', 'Z', 'S', 'Q', 'V', 'H', 'L', 'C', 'K',
         'E'};
   return TABLA_LETRAS_NIF[dni % NUM_LETRAS];
}
```

Matrices.
Vectores con varios índices

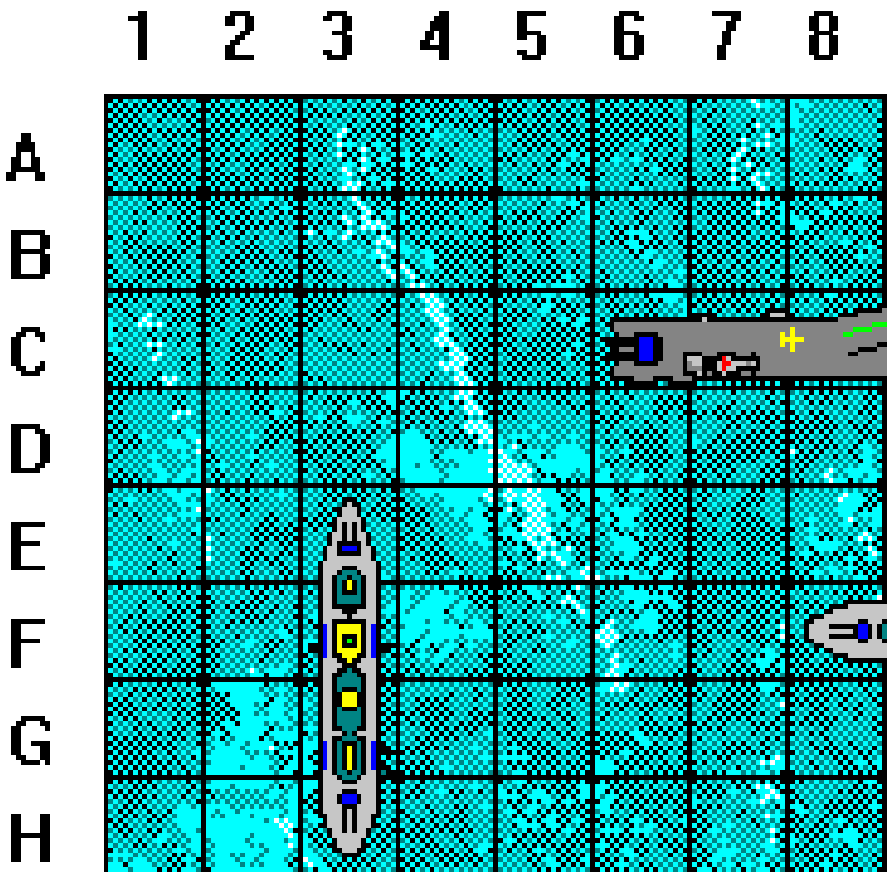
$$\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1j} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2j} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ a_{i1} & a_{i2} & \dots & a_{ij} & \dots & a_{in} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nj} & \dots & a_{nn} \end{pmatrix}$$

Matrices

```
const int N = 20;
const int M = 30;
double mat[N][M];
```



Matrices



Matrices

- Definición de un tablero para el juego de los barcos
- Inicialización como «agua»
- Colocación de un barco en las casillas E3, F3, G3 y H3

Matrices

```
const int NUM_FILAS = 8;
const int NUM_COLUMNAS = 8;
const bool AGUA = false;
const bool BARCO = true;

bool tablero[NUM_FILAS][NUM_COLUMNAS];

for (int i = 0; i < NUM_FILAS; i++) {
   for (int j = 0; j < NUM_COLUMNAS; j++) {
      tablero[i][j] = AGUA;
   }
}

// Colocación de un barco en las casillas E3, F3, G3 y H3
tablero[4][2] = BARCO;
tablero[5][2] = BARCO;
tablero[6][2] = BARCO;
tablero[7][2] = BARCO;
```

Ejercicios con matrices

- Matriz unidad
- Suma de matrices
- Multiplicación de matrices
- Traspuesta de una matriz
- Simetría de una matriz

Matriz unidad. Una solución

```
const int DIM = 20;
/*
 * Pre:  «matriz» es una matriz cuadrada de DIM x DIM.
 * Post: Ha inicializado «matriz» como la matriz unidad de
 *       tamaño DIM x DIM.
 */
void unidad(double matriz[][DIM]) {
   for (int i = 0; i < DIM; i++) {
      for (int j = 0; j < DIM; j++) {
         if (i == j) {
            matriz[i][j] = 1.0;
         }
         else {
            matriz[i][j] = 0.0;
         }
      }
   }
}
```

Suma de matrices

```
/*
 * Pre:  «a», «b» y «suma» son matrices
 *       cuadradas de DIM x DIM.
 * Post: «suma» es la suma matricial
 *       de «a» y «b».
 */
void sumar(const double a[][DIM],
           const double b[][DIM],
           double suma[][DIM]);
```

Producto de matrices

```
/*
 * Pre:  «a», «b» y «producto» son
 *       matrices cuadradas de DIM x DIM.
 * Post: «producto» es el producto
 *       matricial de «a» y «b».
 */
void multiplicar(const double a[][DIM],
                 const double b[][DIM],
                 double producto[][DIM]);
```

Simetría de una matriz

```
/*
 * Pre:  «matriz» es una matriz
 *       cuadrada de DIM x DIM.
 * Post: Ha devuelto el valor «true»
 *       si y solo si «matriz» es
 *       simétrica.
 */
bool esSimetrica(
        const double matriz[][DIM]);
```