

# Fundamentos de la Programación II

## Grado en Desarrollo de videojuegos

Jaime Sánchez Hernández

Departamento de Sistemas Informáticos y Computación  
Universidad Complutense de Madrid

14 de febrero de 2021

Jaime Sánchez. Sistemas Informáticos y Computación, UCM

1/8

## Anidando tipos estructurados

Hemos dicho:

- ▶ las componentes de un array pueden ser de cualquier tipo válido (predefinido o definido por el programador).
- ▶ los campos de un registro pueden ser de cualquier tipo válido (predefinido o definido por el programador).

Entonces... podemos definir un array de registros o un registro que contenga campos de tipo array:

- ▶ Se puede definir un array de componentes de tipo **Coche**
- ▶ o incluir en el registro **Coche** un campo **propietarios** de tipo array que contenga el historial de propietarios del coche
  - ▶ que a su vez pueden definirse como un registro con **Nombre**, **Apellido1**, **Apellido2**, **DNI**,...

Podemos **anidar** arrays y registros sin límite  $\leadsto$  de este modo podemos dar una **representación estructurada** para cualquier tipo de información.

3/8

Representación de la información:

Datos estructurados

Jaime Sánchez. Sistemas Informáticos y Computación, UCM

2/8

## Ejemplo: representación y manipulación de polinomios

Queremos escribir un programa para trabajar con polinomios que incluya las operaciones típicas: suma, resta, multiplicación, división, evaluación en un valor dado, lectura, escritura...

Antes de nada: ¿qué representación elegimos?

**Primera idea:** array de reales, donde el contenido de la componente  $i$ -ésima representa el coeficiente de  $x^i$ . Por ejemplo: el polinomio

$$3x^4 - 5,23x^2 + 6x - 7,9$$

se representará como:

0	1	2	3	4	5	6	7	8	9	10
-7,9	6,0	-5,23	0,0	3,0	0,0	0,0	0,0	0,0	0,0	0,0

Jaime Sánchez. Sistemas Informáticos y Computación, UCM

4/8

## Structs (registros)

Esta representación es sencilla ... pero

- ▶ el grado del polinomio está limitado por el tamaño del array (de qué tamaño declaramos el vector)
- ▶ puede hacer muy mal aprovechamiento de la memoria: para representar el polinomio  $2x^{3456}$  se necesita un vector de 3456 componentes ... de las cuales sólo se utiliza realmente una!!

Representación alternativa?

- ▶ Definimos el tipo **monomio** como un registro con dos componentes: coeficiente y exponente
- ▶ Un polinomio será un array de monomios ... de qué tamaño?
  - ▶ El tamaño puede definirse en el momento de crear el polinomio conociendo el número de monomios.
  - ▶ O bien puede fijarse de antemano constante N y *llevar cuenta del tamaño tam del polinomio (número de monomios)*

Jaime Sánchez. Sistemas Informáticos y Computación, UCM

5/8

## Structs (registros)

Lectura de polinomios:

```
static void leeMonomio(out Monomio m){
    Console.Write ("Coeficiente: ");
    m.coef = double.Parse (Console.ReadLine ());
    Console.Write ("Exponente: ");
    m.exp = int.Parse (Console.ReadLine ());
}

static void leePolinomio(out Polinomio p){
    p.mon = new Monomio[N];

    Console.Write ("Número de monomios: ");
    p.tam = int.Parse (Console.ReadLine ());

    Console.WriteLine ("Introduce monomios");
    for (int i=0; i<p.tam; i++)
        leeMonomio (out p.mon[i]);
}
```

Jaime Sánchez. Sistemas Informáticos y Computación, UCM

7/8

## Structs (registros)

Definimos el tipo Monomio y Polinomio:

```
const int N = 100; // tamaño del vector de monomios

struct Monomio {
    public double coef;
    public int exp;
}

struct Polinomio {
    public Monomio [] mon; // array de monomios
    public int tam; // número de monomios
                    // último monomio en tam-1
}
```

Jaime Sánchez. Sistemas Informáticos y Computación, UCM

6/8

## Structs (registros)

Escritura de polinomios:

```
static void escribePolinomio(Polinomio p){
    for (int i = 0; i < p.tam; i++)
        Console.Write (" + " + p.mon [i].coef + "x^" + p.mon [i].exp);
}
```

Esta escritura se puede mejorar mucho: eliminando signos "+" innecesarios, exponentes 0 ó 1, etc

La propia representación de los polinomios es muy mejorable:

- ▶ Invariante de la representación 1: unicidad de exponentes en vector de monomios, para que no aparezca más de un monomio del mismo grado).
- ▶ Invariante de la representación 2: que no aparezca ningún monomio con coeficiente 0.

Ejercicio: mejorar la representación con estas ideas e implementar las operaciones básicas de suma, resta, multiplicación, evaluación en un valor, etc

8/8