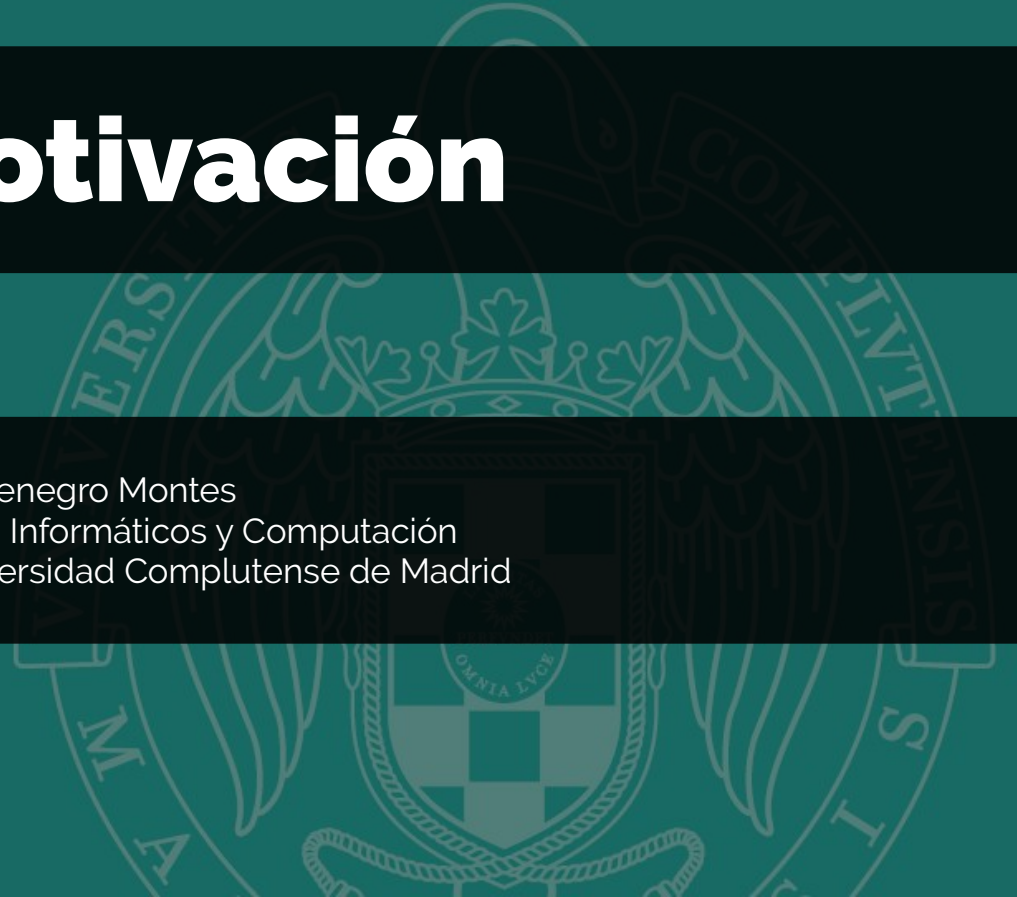


ESTRUCTURAS DE DATOS

INTRODUCCIÓN A LOS TIPOS ABSTRACTOS DE DATOS

# TADs: motivación

Manuel Montenegro Montes  
Departamento de Sistemas Informáticos y Computación  
Facultad de Informática – Universidad Complutense de Madrid



# Un pequeño juego



Jugador 1

N  
S  
O  
T

E  
T  
V



Jugadora 2

# Un pequeño juego



Jugador 1

N  
S  
O  
T

E  
T  
V



Jugadora 2

- Para saber si una letra se ha dicho antes, debemos almacenar el conjunto de letras nombradas hasta el momento.

# Tipo de datos ConjuntoChar

```
const int MAX_CHARS = 26;
```

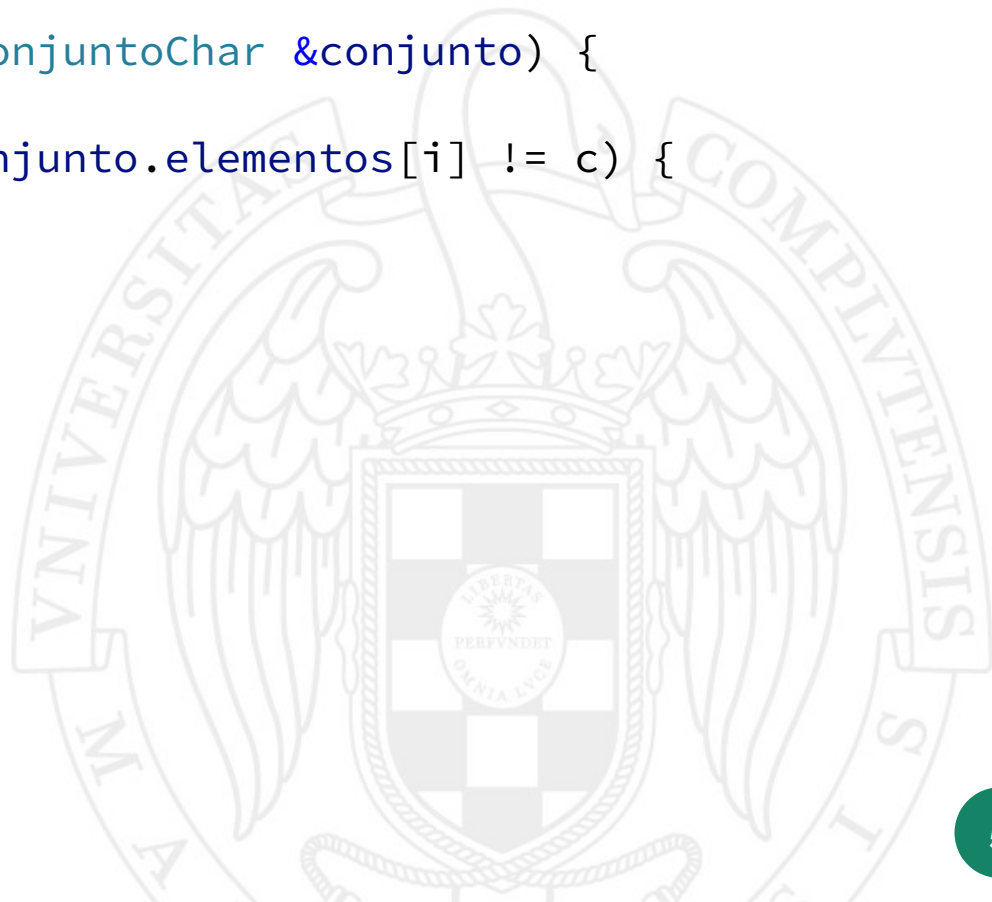
```
struct ConjuntoChar {  
    int num_chars;  
    char elementos[MAX_CHARS];  
};
```

- Suponemos que solo se admiten las letras mayúsculas del alfabeto inglés (A-Z).
  - Son un total de 26 letras.
- Guardamos las letras nombradas hasta el momento en el array **elementos**.
- Las primeras **num\_chars** posiciones tienen letras. El resto se consideran posiciones “vacías”.

# Función auxiliar: esta\_en\_conjunto

- Determina si el conjunto contiene la letra c pasada como parámetro.

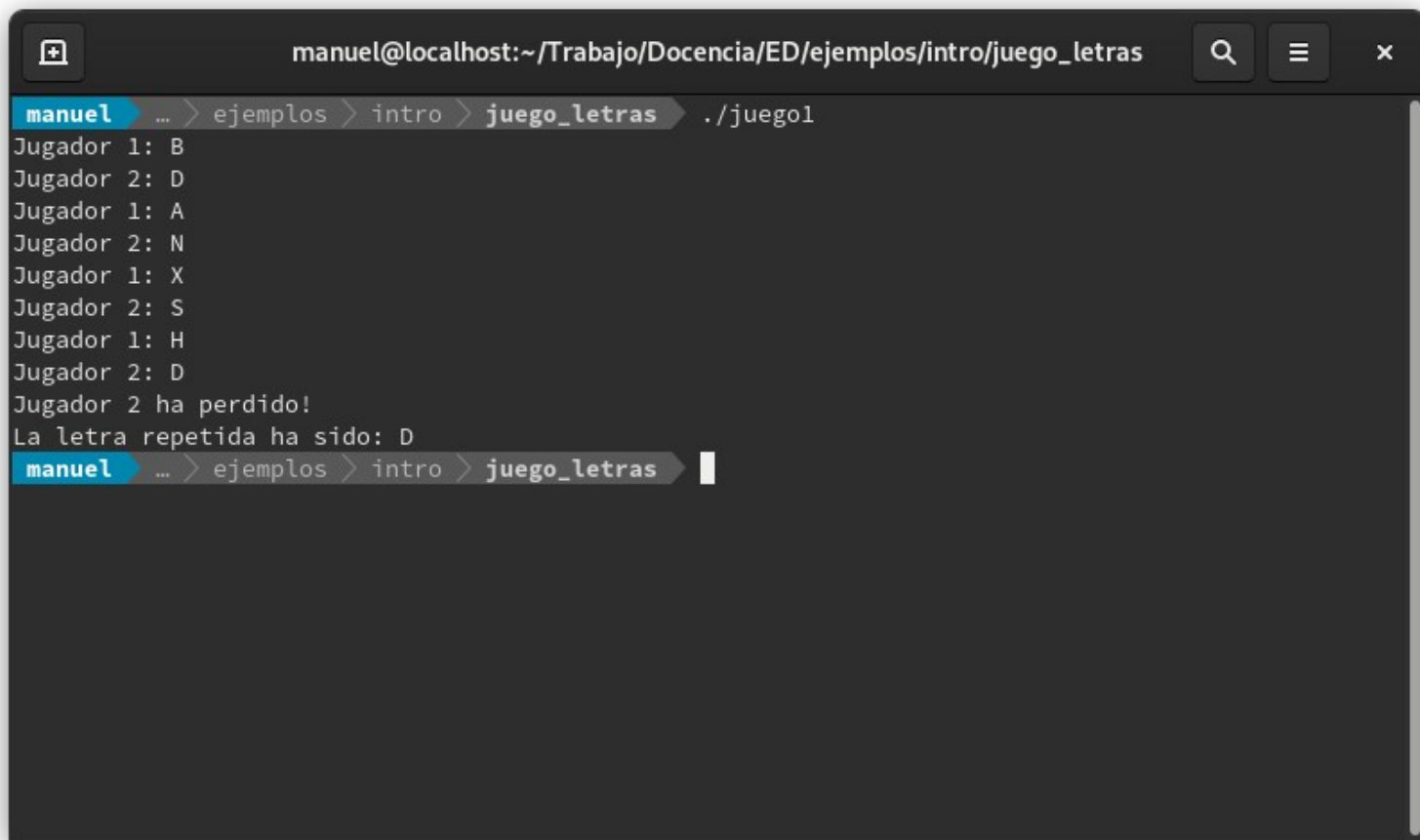
```
bool esta_en_conjunto(char c, const ConjuntoChar &conjunto) {  
    int i = 0;  
    while (i < conjunto.num_chars && conjunto.elementos[i] != c) {  
        i++;  
    }  
    return conjunto.elementos[i] == c;  
}
```



# Implementación inicial del juego

```
int main() {  
    int jugador_actual = 1;  
    ConjuntoChar letras_nombradas;  
    letras_nombradas.num_chars = 0;  
  
    char letra_actual = preguntar_letra(jugador_actual);  
  
    while (!esta_en_conjunto(letra_actual, letras_nombradas)) {  
        letras_nombradas.elementos[letras_nombradas.num_chars] = letra_actual;  
        letras_nombradas.num_chars++;  
  
        jugador_actual = cambio_jugador(jugador_actual);  
        letra_actual = preguntar_letra(jugador_actual);  
    }  
  
    std::cout << "Jugador " << jugador_actual << " ha perdido!" << std::endl;  
    std::cout << "La letra repetida ha sido: " << letra_actual << std::endl;  
    return 0;  
}
```

# Funcionamiento



A terminal window titled "manuel@localhost:~/Trabajo/Docencia/ED/ejemplos/intro/juego\_letras" with search, menu, and close icons. The terminal shows the execution of a word game program. The path in the prompt is "... > ejemplos > intro > juego\_letras". The program output is as follows:

```
manuel ... > ejemplos > intro > juego_letras ./juego1
Jugador 1: B
Jugador 2: D
Jugador 1: A
Jugador 2: N
Jugador 1: X
Jugador 2: S
Jugador 1: H
Jugador 2: D
Jugador 2 ha perdido!
La letra repetida ha sido: D
manuel ... > ejemplos > intro > juego_letras
```

# Cambios en la implementación

```
const int MAX_CHARS = 26;
```

```
struct ConjuntoChar {  
    bool esta[MAX_CHARS];  
};
```

- Nuestro conjunto contiene un número limitado de letras.
- Podemos representar el contenido del conjunto como un array de booleanos.
  - Si la letra A está en el conjunto: `esta[0] = true`.
  - Si la letra B está en el conjunto: `esta[1] = true`.
  - ...



# Cambios en esta\_en\_conjunto

```
bool esta_en_conjunto(char c, const ConjuntoChar &conjunto) {  
    return esta[c - (int)'A'];  
}
```



# Implementación inicial del juego

```
int main() {  
    int jugador_actual = 1;  
    ConjuntoChar letras_nombradas;  
    letras_nombradas.num_chars = 0; ❌  
  
    char letra_actual = preguntar_letra(jugador_actual);  
  
    while (!esta_en_conjunto(letra_actual, letras_nombradas)) { ✅  
        letras_nombradas.elementos[letras_nombradas.num_chars] = letra_actual; ❌  
        letras_nombradas.num_chars++;  
  
        jugador_actual = cambio_jugador(jugador_actual);  
        letra_actual = preguntar_letra(jugador_actual);  
    }  
  
    std::cout << "Jugador " << jugador_actual << " ha perdido!" << std::endl;  
    std::cout << "La letra repetida ha sido: " << letra_actual << std::endl;  
    return 0;  
}
```



# ¿Qué ha fallado?

- Cualquier cambio en el tipo de datos `ConjuntoChar` tiene que ser propagado hasta aquellos sitios en los que se utilicen dichos campos.
- La función `main()` menciona explícitamente los campos del tipo `ConjuntoChar`. Por tanto, se ve afectada por el cambio de la definición del tipo.
- Un cambio en la definición de un tipo de datos debe provocar el menor impacto posible en la implementación del resto del programa.
- ¿Cómo delimitamos las operaciones que pueden verse afectadas por este cambio?

**Abstracción mediante Tipos Abstractos de Datos (TADs)**