

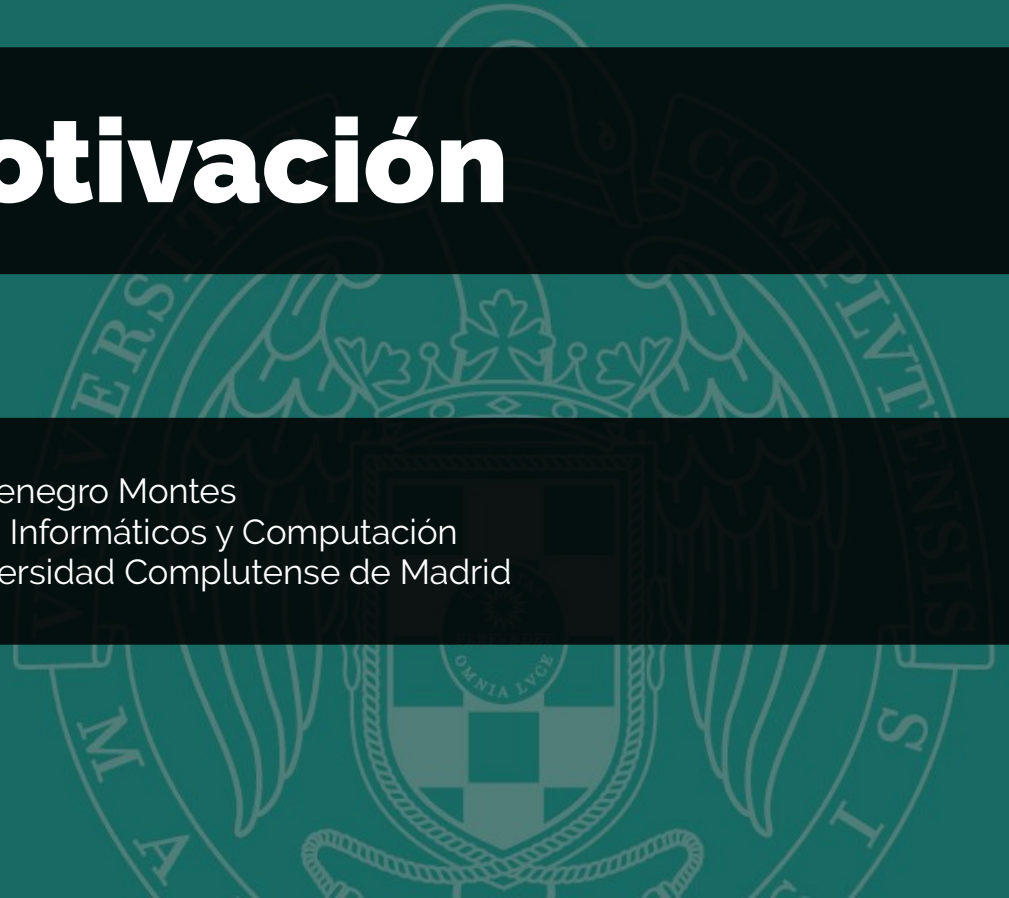
ESTRUCTURAS DE DATOS

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

TADS= Tipos abstractos

Dice que es un ejercicio de FP1. Nos va a ayudar a entender la abstracción de los datos.

La segunda jugadora dice otra.

primer jugador dice letra de la A-Z

N



Jugador 1

S

Van diciendo letras hasta que uno de ellos dice una letra repetida

O

T

Jugador que dice letra repetida es el que pierde.

E

T

V



Jugadora 2

Dice que no nos vamos a centrar en ejercicios de verificación formal, pero tenemos que entender los CONCEPTOS DE MODELO Y REPRESENTACIÓN

# Un pequeño juego

N	E	S	T	O	V	T	
---	---	---	---	---	---	---	--

Almacenamos las letras en un array a medida que se van mencionando



Jugador 1

N

S

O

T

recorremos posiciones anteriores para saber si se había dicho la letra o no.

E

T

V



Jugadora 2

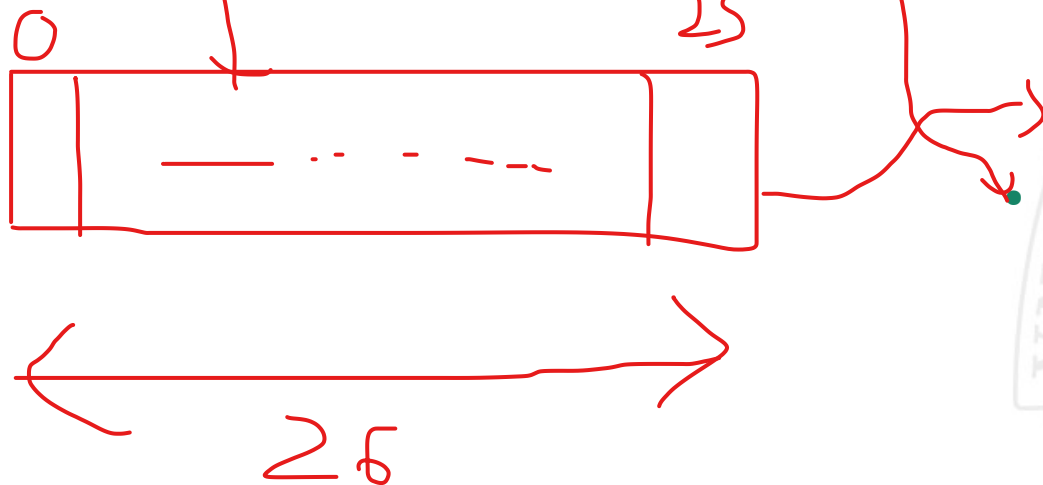
- Para saber si una letra se ha dicho antes, debemos almacenar el conjunto de letras nombradas hasta el momento.

# Tipo de datos ConjuntoChar

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

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

num\_chars. Dice cuantas posiciones tienen letras



- 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.

esto lo modificaremos usando TADS

```
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;  
}
```

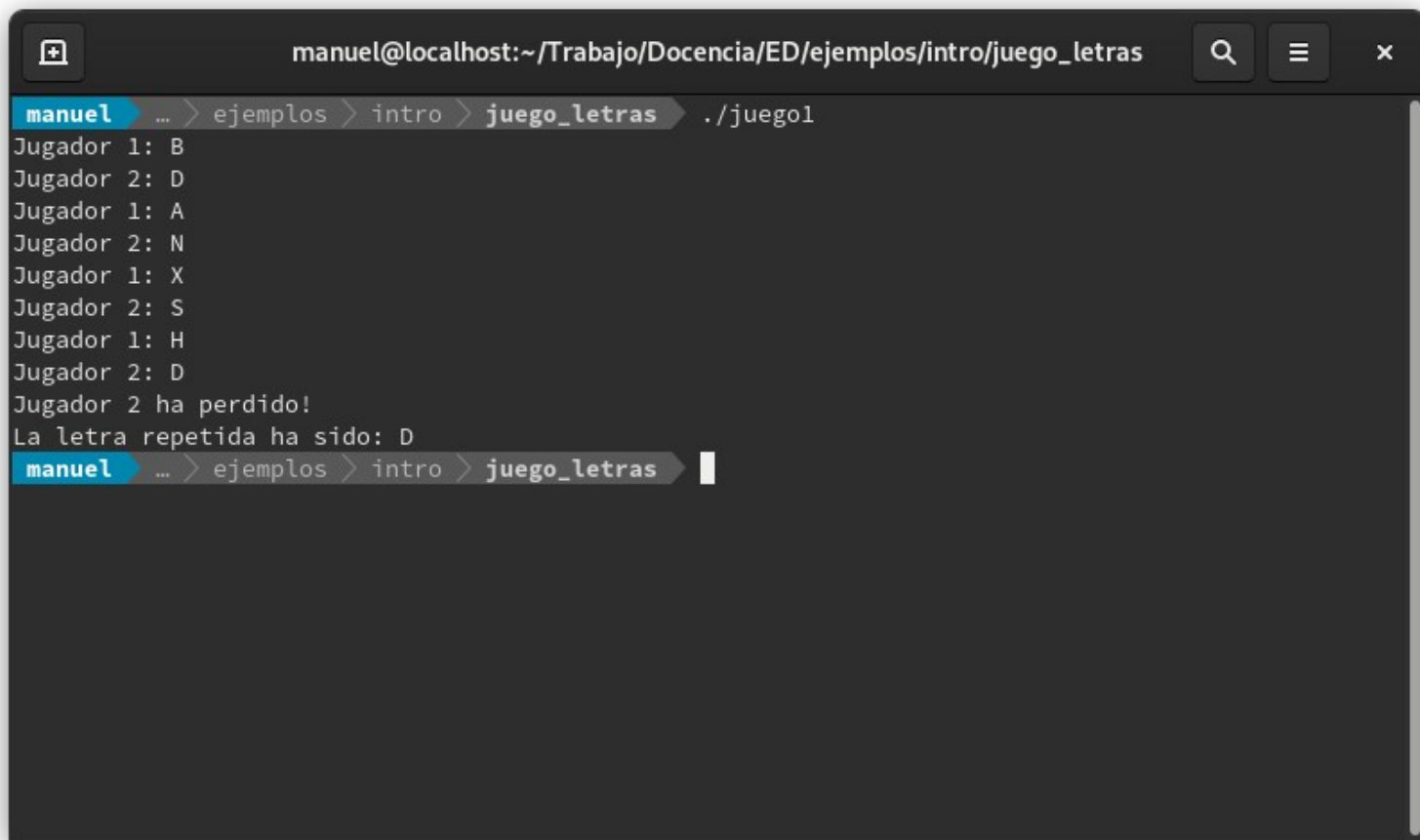


hasta num\_chars, porque es la primera posición vacía.

# Implementación inicial del juego

```
int main() {  
    int jugador_actual = 1; Variable que nos dice cual es el jugador actual si el 1 o el 2  
    ConjuntoChar letras_nombradas; Conjunto de letras nombradas hasta el momento.  
    letras_nombradas.num_chars = 0; Valor inicial de num_chars, ya que el vector comienza vacío.  
  
    char letra_actual = preguntar_letra(jugador_actual);  
    si la letra no se ha dicho anteriormente  
    while (!esta_en_conjunto(letra_actual, letras_nombradas)) {  
        letras_nombradas.elementos[letras_nombradas.num_chars] = letra_actual;  
        letras_nombradas.num_chars++; asignamos la letra al array y aumentamos num_chars  
  
        jugador_actual = cambio_jugador(jugador_actual); cambiamos de jugador  
        letra_actual = preguntar_letra(jugador_actual); pedimos que el jugador que tenga el turno diga letra.  
    } IMPLEMENTACIÓN NOS DA IGUAL  
  
    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'. The window shows the execution of a program named 'juego1'. The output displays a sequence of moves for two players, 'Jugador 1' and 'Jugador 2'. The moves are: B, D, A, N, X, S, H, D. After the eighth move, the program outputs 'Jugador 2 ha perdido!' and 'La letra repetida ha sido: D'. The prompt 'manuel' is shown at the bottom, indicating the user is ready to enter a new command.

```
manuel@localhost:~/Trabajo/Docencia/ED/ejemplos/intro/juego_letras
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];  
};
```

Inicialmente todas false.

OTRA FORMA DE IMPLEMENTARLO SERÍA ESTA

- 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'];  
}
```

Valor ASCII: 65 corresponde a la A - 90 corresponde a la Z

restamos el código correspondiente a la letra A, que es el 65 para hallar su posición en el array  
bool esta[MAX\_CHARS]

SI EN LA POSICIÓN HAY UN TRUE, DEVUELVE QUE ESTÁ EN EL CONJUNTO Y POR TANTO EL JUGADOR HABRÍA PERDIDO

# Implementación inicial del juego

```
int main() {  
    int jugador_actual = 1;  
    ConjuntoChar letras_nombradas;  
    letras_nombradas.num_chars = 0;
```

programa NO COMPILARÁ SI TENEMOS EL MAIN ANTERIOR.



```
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?

Veremos durante este tema como solucionar los problemas que se exponen aquí.

- 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)**