

Estructuras Lineales

Listas

Verónica E. Arriola-Rios

Facultad de Ciencias, UNAM

2 de octubre de 2024



Listas

- 1 Listas
- 2 Implementaciones
- 3 Listas circulares
- 4 Listas múltiples
- 5 Bibliografía

Temas

1 Listas

- Definición recursiva
- Definición secuencial

Definición de lista

Recursividad estructural

$$\text{Lista} = \begin{cases} \text{Lista vacía} \\ \text{Dato seguido de otra lista} \end{cases}$$

Lista recursiva en Java

Código: Lista recursiva

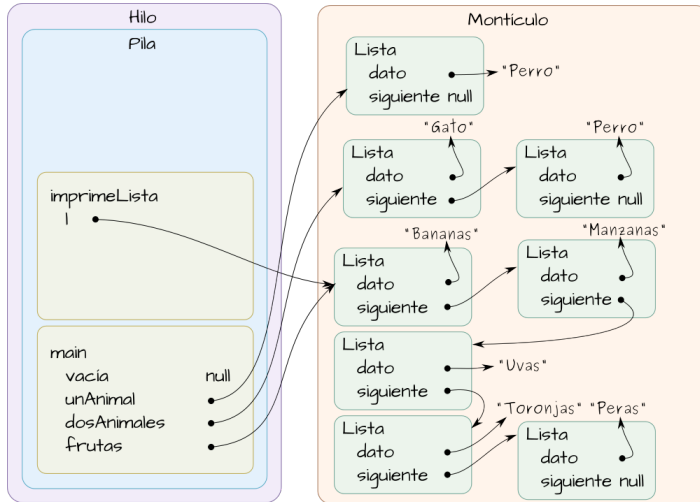
```
1 public class Lista<T>
2 {
3     private T _dato;
4     private Lista<T>? _siguiente;
5     public T Dato { get => _dato; }
6     public Lista<T>? Siguiete { get => _siguiente; set => _siguiente = value; }
7
8     public Lista(T dato, Lista<T>? siguiente)
9     {
10         _dato = dato;
11         _siguiente = siguiente;
12     }
13
14     public static void ImprimeLista(Lista<T> l)
15     {
16         if(l == null) return;
17         else
18         {
19             Console.WriteLine(l.Dato);
20             ImprimeLista(l.Siguiete!);
21         }
22     }
23 }
```

Construcción manual de listas

- \emptyset .
- "Perro" $\rightarrow \emptyset$.
- "Gato" \rightarrow "Perro" $\rightarrow \emptyset$.
- "Bananas" \rightarrow "Manzanas" \rightarrow "Toronjas" \rightarrow "Uvas" \rightarrow "Peras" $\rightarrow \emptyset$.

Código: Uso de listas

```
1 public class UsoLista
2 {
3     public static void Main()
4     {
5         Lista<string>? vacía = null;
6         Lista<string> unAnimal = new Lista<string>("Perro", null);
7         Lista<string> dosAnimales = new Lista<string>("Gato", new Lista<string>("Perro", null));
8         Lista<string> frutas = new Lista<string>("Bananas",
9             new Lista<string>("Manzanas",
10                 new Lista<string>("Toronjas",
11                     new Lista<string>("Uvas", new Lista<string>("Peras", null))));
12         Lista<string>.ImprimeLista(dosAnimales);
13     }
14 }
```



Construcción manual de listas

- \emptyset .
- "Perro" $\rightarrow \emptyset$.
- "Gato" \rightarrow "Perro" $\rightarrow \emptyset$.
- "Conejo" \rightarrow "Gato" \rightarrow "Perro" $\rightarrow \emptyset$.

Código: Uso de listas

```
1 public class UsoLista
2 {
3     public static void Main()
4     {
5         Lista<string>? vacía = null;
6         Lista<string> unAnimal = new Lista<string>("Perro", null);
7         Lista<string> dosAnimales = new Lista<string>("Gato", new Lista<string>("Perro", null));
8         Lista<string> animales = new Lista<string>("Conejo", dosAnimales);
9         Lista<string>.ImprimeLista(dosAnimales);
10    }
11 }
```


Temas

1 Listas

- Definición recursiva
- Definición secuencial

Definición de lista 2

- Una *lista* es una secuencia de cero a más elementos **de un tipo determinado** (que por lo general se denominará tipo-elemento). Se representa como una sucesión de elementos separados por comas:

$$a_0, a_1, \dots, a_{n-1} \quad (1)$$

donde $n \geq 0$ y cada a_i es del tipo **tipo-elemento**.

- Al número n de elementos se le llama *longitud* de la lista.
- a_0 es el *primer elemento* y a_{n-1} es el *último elemento*.
- Si $n = 0$, se tiene una **lista vacía**, es decir, que no tiene elementos. Aho, Hopcroft y Ullman 1983

Lista

Una propiedad importante de una lista es que sus elementos pueden estar ordenados en forma lineal de acuerdo con sus posiciones en la misma.

- Se dice que α_i *precede* a α_{i+1} para $i = 0, 1, \dots, n-2$ y α_i *sucedee* a α_{i-1} para $i = 1, 2, \dots, n-1$.
- Se dice que el elemento α_i está en la posición i .

Operaciones

- instanciar: $\emptyset \rightarrow \text{ListaVacía}$
- insertar: $\text{Lista, Posición, Elemento} \rightarrow \text{Lista}$
- recuperar: $\text{Lista, Posición} \rightarrow \text{Elemento}$
- borrar: $\text{Lista, Elemento} \rightarrow \text{Lista}$
- siguiente: $\text{Lista, Posición} \rightarrow \text{Elemento}$
- anterior: $\text{Lista, Posición} \rightarrow \text{Elemento}$
- destruir: $\text{Lista} \rightarrow \emptyset$
- imprimir: $\text{Lista} \rightarrow \text{Cadena}$

Implementaciones

- 1 Listas
- 2 Implementaciones
- 3 Listas circulares
- 4 Listas múltiples
- 5 Bibliografía

Temas

2 Implementaciones

- Lista ligada
- Lista doblemente ligada
- Lista en un arreglo

Lista ligada

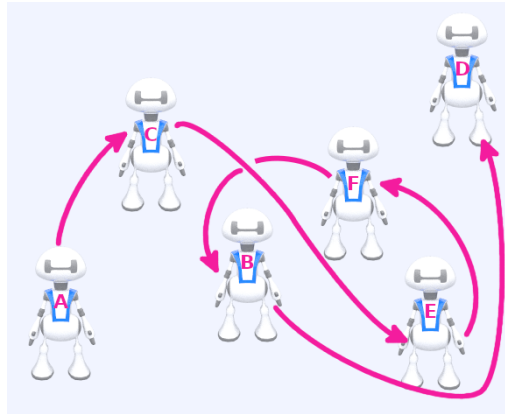


Figura: Cada robot sabe quién va detrás de él. La lista define la secuencia de robots: ACEFBD.

Temas

2 Implementaciones

- Lista ligada
- Lista doblemente ligada
- Lista en un arreglo

Lista doblemente ligada

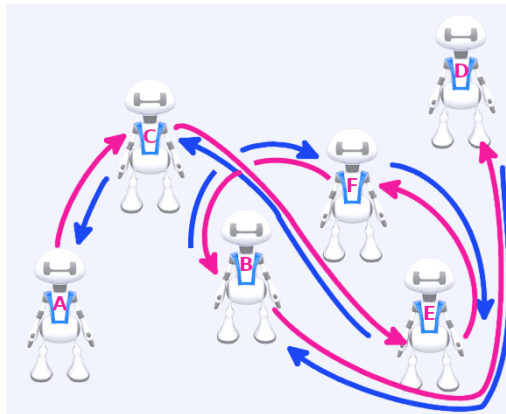
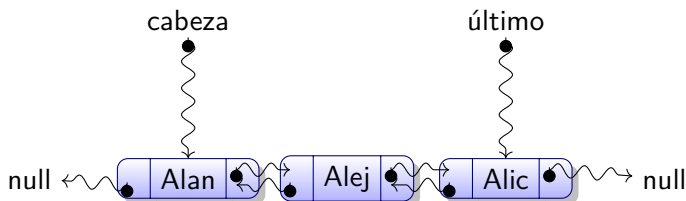


Figura: Cada robot sabe quién va detrás de él. La lista define la secuencia de robots: ACEFBD.

Nodo doble

- Cada elemento contiene referencias al elemento anterior y al elemento posterior.



NodoDoble<E>

```
-dato: E  
-siguiente: NodoDoble<E>  
-anterior: NodoDoble<E>  
  
+NodoDoble(dato:E)  
+NodoDoble(dato:E, siguiente:NodoDoble<E>)  
+NodoDoble(dato:E, anterior:NodoDoble<E>, siguiente:NodoDoble<E>)  
  
+getDato(): E  
+setDato(dato:E): void  
+getSiguiente(): NodoDoble<E>  
+setSiguiente(sig:NodoDoble<E>): void  
+getAnterior(): NodoDoble<E>  
+setAnterior(ant:NodoDoble<E>): void
```

Temas

2 Implementaciones

- Lista ligada
- Lista doblemente ligada
- Lista en un arreglo

Lista en un arreglo

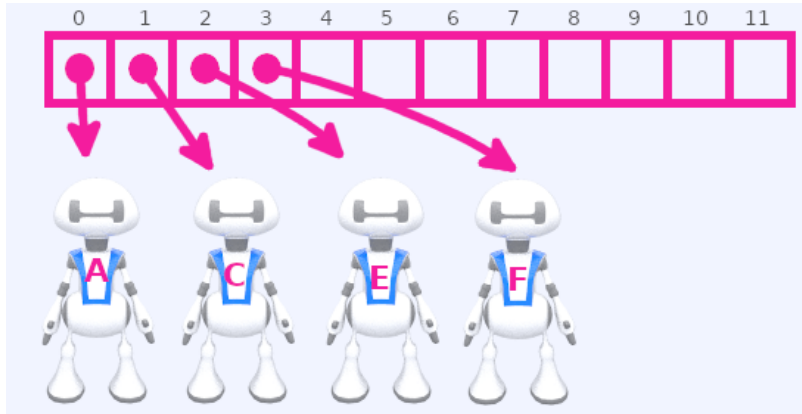


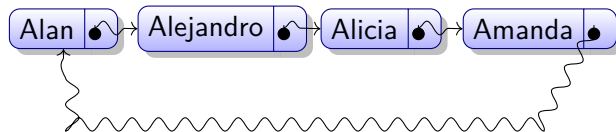
Figura: Los elementos de la lista se guardan en las posiciones correspondientes de un arreglo.

Listas circulares

- 1 Listas
- 2 Implementaciones
- 3 Listas circulares**
- 4 Listas múltiples
- 5 Bibliografía

Listas circulares

- El elemento final hace referencias al elemento inicial, por lo que para todo elemento existe un elemento siguiente y un elemento anterior.



Listas múltiples

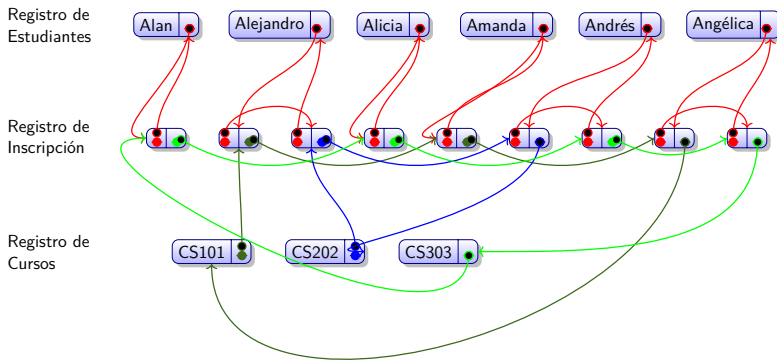
- 1 Listas
- 2 Implementaciones
- 3 Listas circulares
- 4 Listas múltiples**
- 5 Bibliografía

Listas múltiples

- Representan relaciones muchos a muchos Vargas Villazón, Lozano Moreno y Levine Gutiérrez 1998.
- Se representan con referencias.
- Casi siempre es más fácil usar tablas, pero llegan a aparecer.

Listas múltiples. Ejemplo.



Estudiantes\Cursos	CS101	CS202	CS203
Alan			✓
Alejandro	✓	✓	
Alicia			✓
Amanda	✓		
Andrés		✓	✓
Angélica	✓		✓



Bibliografía

- 1 Listas
- 2 Implementaciones
- 3 Listas circulares
- 4 Listas múltiples
- 5 Bibliografía**

Bibliografía I

-  Aho, Alfred V., John E. Hopcroft y Jeffrey D. Ullman (1983). *Data Structures and Algorithms*. Addison-Wesley.
-  Vargas Villazón, América, Jorge Lozano Moreno y Guillermo Levine Gutiérrez, eds. (1998). *Estructuras de datos y Algoritmos*. John Wiley & Sons, 438 pp.

Licencia

Creative Commons
Atribución-No Comercial-Compartir Igual

