ESTRUCTURAS DE DATOS

TIPOS ABSTRACTOS DE DATOS LINEALES

Constructores de copia en el TAD Lista

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

Implementaciones del TAD Lista

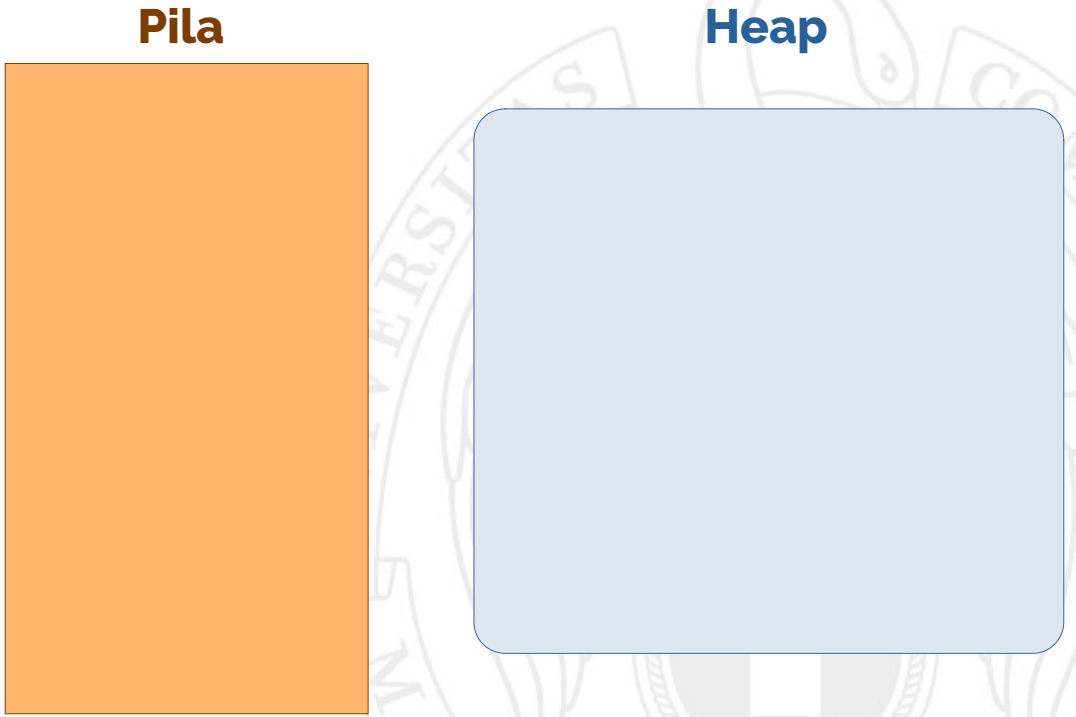
- Mediante vectores.
- Mediante listas enlazadas simples.

Implementación mediante vectores

Implementación mediante vectores

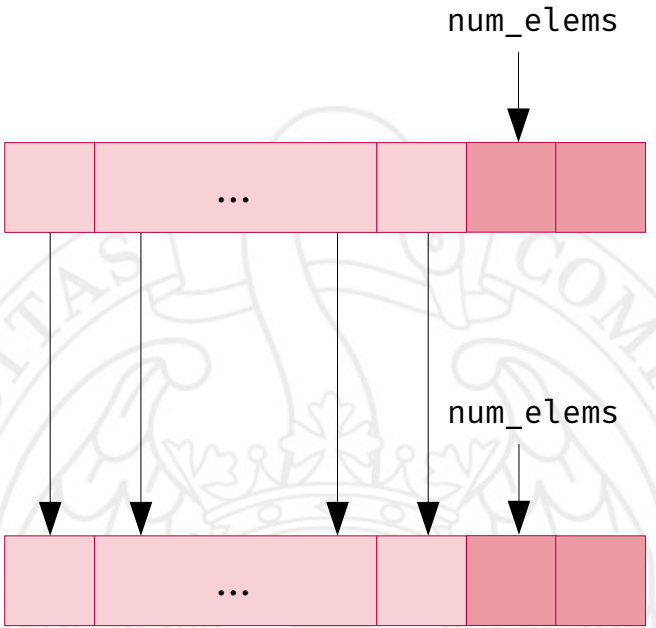
- El constructor de copia por defecto para la clase `ListArray` no nos sirve.

```
ListArray l1;
l1.push_back("David");
l1.push_back("Maria");
l1.push_back("Eugenio");

ListArray l2 = l1;
l2.front() = "Pepe";
```



Constructor de copia para `ListArray`

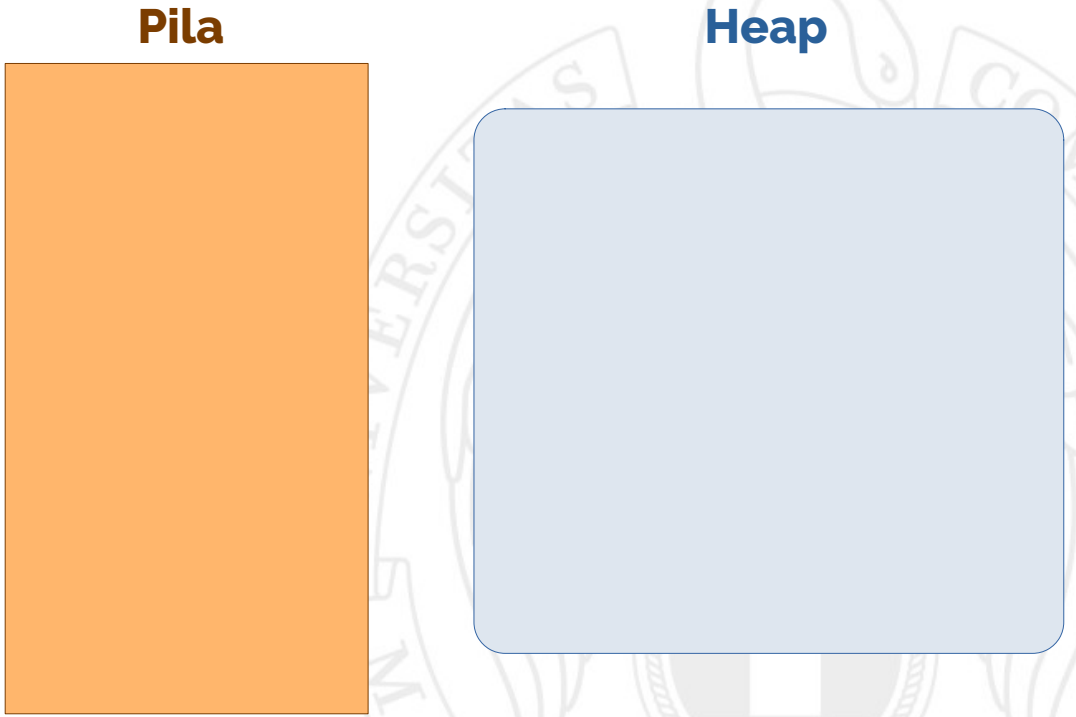
```
class ListArray {
public:
   ...
  ListArray(const ListArray &other);
   ...
};

ListArray::ListArray(const ListArray &other)
  : num_elems(other.num_elems),
    capacity(other.capacity),
    elems(new std::string[other.capacity])
{
  for (int i = 0; i < num_elems; i++) {
    elems[i] = other.elems[i];
  }
}
```



Ejemplo

- Con el constructor de copia, `l1` y `l2` pueden ser modificadas de manera independiente.

```
ListArray l1;
l1.push_back("David");
l1.push_back("Maria");
l1.push_back("Eugenio");

ListArray l2 = l1;
l2.front() = "Pepe";
```



Implementación mediante listas enlazadas

Implementación mediante listas enlazadas

- Creamos una función auxiliar (`copy_nodes`) para producir una copia de una lista enlazada de nodos.
- El constructor de copia inicializa la cabeza creando una copia de la lista enlazada original.

```
class ListLinkedSingle {
public:
  ListLinkedSingle(const ListLinkedSingle &other)
    : head(copy_nodes(other.head)) { }

private:
  Node *head;

  ...
  Node *copy_nodes(Node *start_node) const;
};
```

Implementación recursiva de copy_nodes

```
ListLinkedSingle::Node * ListLinkedSingle::copy_nodes(Node *start_node) const {
  if (start_node ≠ nullptr) {
    Node *result = new Node { start_node→value, copy_nodes(start_node→next) };
    return result;
  } else {
    return nullptr;
  }
}
```