ESTRUCTURAS DE DATOS

TIPOS ABSTRACTOS DE DATOS LINEALES

Implementando el TAD Pila

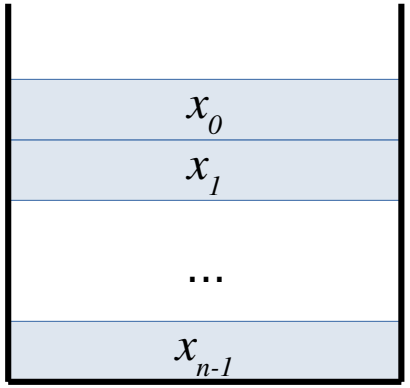
Manuel Montenegro Montes
Departamento de Sistemas Informáticos y Computación
Facultad de Informática – Universidad Complutense de Madrid

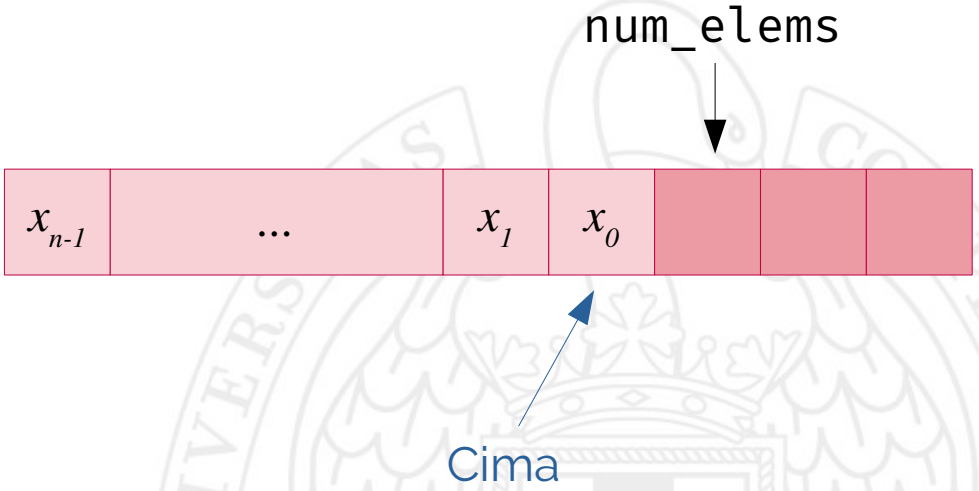
Operaciones sobre pilas

- **Constructoras**:
  - Crear una pila vacía (***create_empty***).
- **Mutadoras**:
  - Añadir elemento en la cima de la pila (***push***).
  - Eliminar elemento en la cima de la pila (***pop***).
- **Observadoras**:
  - Obtener el elemento en la cima de la pila (***top***).
  - Saber si una pila está vacía (***empty***).

Implementación mediante vectores

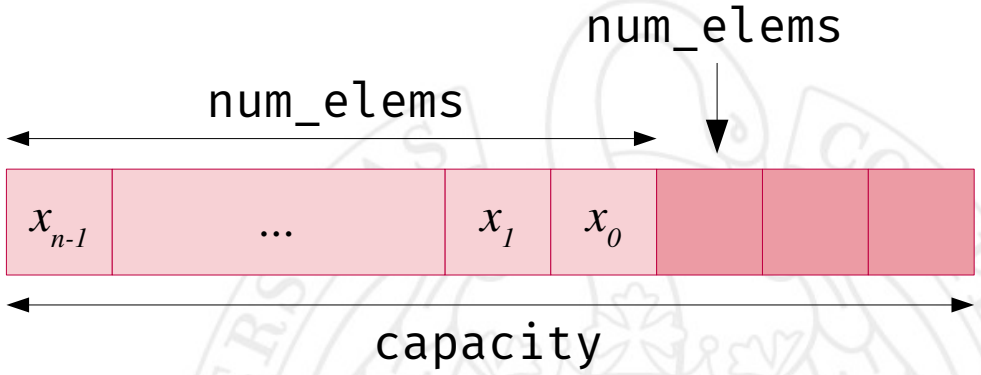
Implementación mediante vectores





Implementación mediante vectores

```
template<typename T>
class StackArray {
public:
   ...

private:
  int num_elems;
  int capacity;
  T *elems;
};
```



Interfaz pública de StackArray

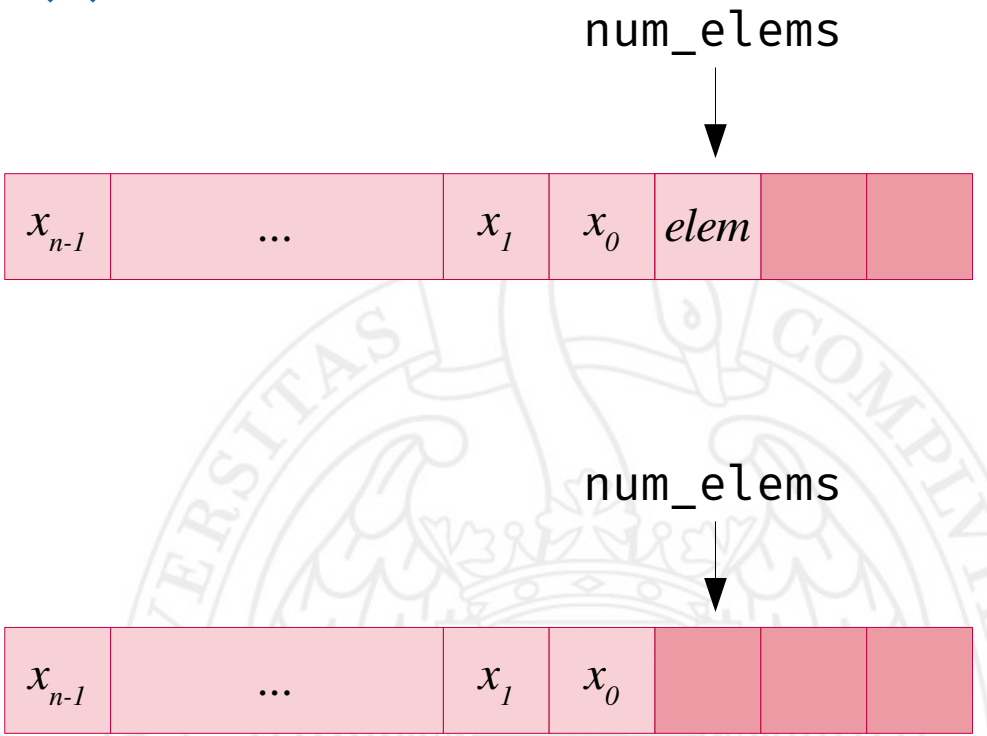
```
template<typename T>
class StackArray {
public:
  StackArray(int initial_capacity = DEFAULT_CAPACITY);
  StackArray(const StackArray &other);
  ~StackArray();

  StackArray & operator=(const StackArray<T> &other);

  void push(const T &elem);
  void pop();
  const T & top() const;
  T & top();
  bool empty() const;

private:
   ...
};
```

Interfaz pública de StackArray

```
template<typename T>
class StackArray {
public:
  StackArray(int initial_capacity = DEFAULT_CAPACITY);
  StackArray(const StackArray &other);
  ~StackArray();

  StackArray & operator=(const StackArray<T> &other);

  void push(const T &elem);
  void pop();
  const T & top() const;
  T & top();
  bool empty() const;

private:
   ...
};
```

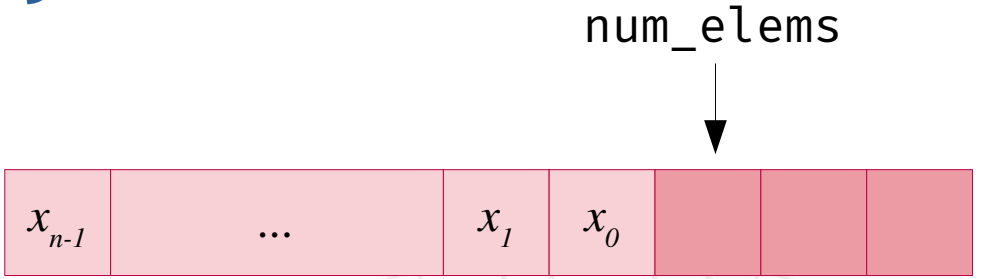
Métodos push( ) y pop( )

```
void push(const T &elem) {
  if (num_elems == capacity) {
    resize_array(capacity * 2);
  }
  elems[num_elems] = elem;
  num_elems++;
}

void pop() {
  assert(num_elems > 0);
  num_elems--;
}
```



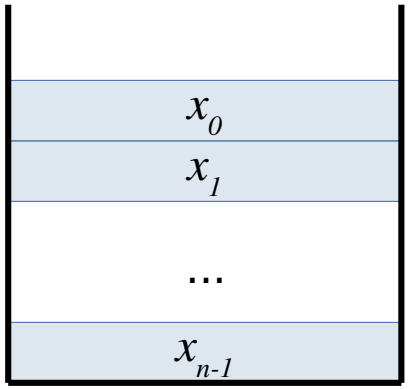
Métodos top( ) y empty( )

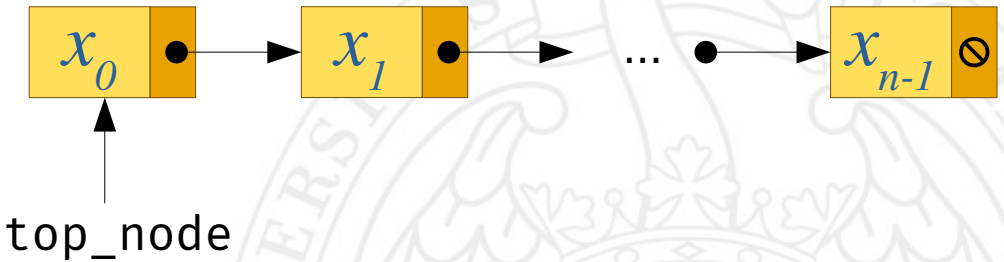
```
const T & top() const {
  assert(num_elems > 0);
  return elems[num_elems - 1];
}

bool empty() const {
  return num_elems == 0;
}
```



Implementación mediante listas enlazadas simples

Implementación mediante listas enlazadas





Implementación mediante listas enlazadas

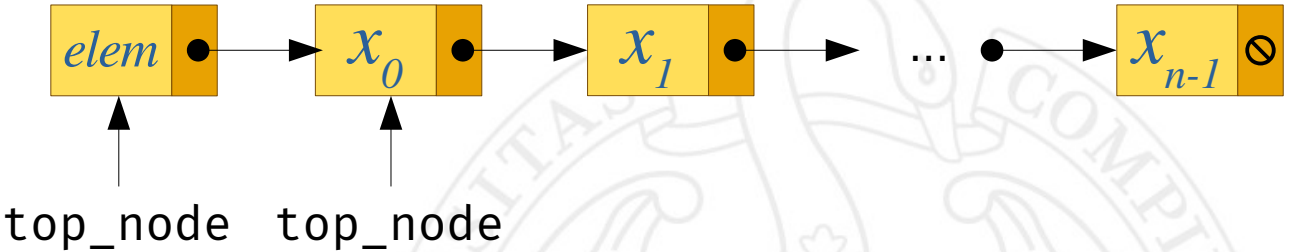
```
template<typename T>
class StackLinkedList {
public:
   ...
private:

  struct Node {
    T value;
    Node *next;
  };

  Node *top_node;
};
```

Interfaz pública de StackLinkedList

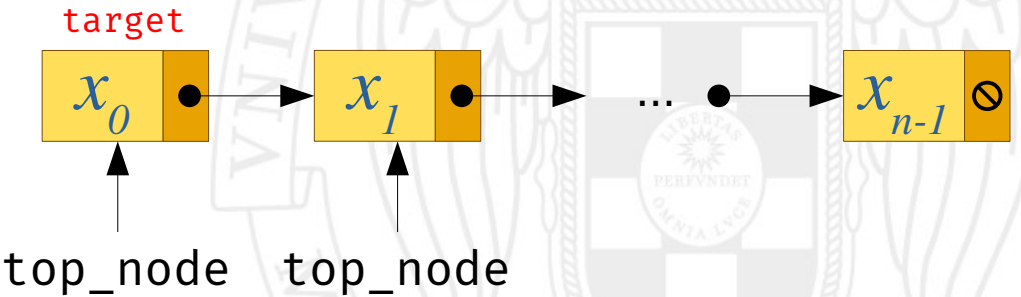
```
template<typename T>
class StackLinkedList {
public:
  StackLinkedList();
  StackLinkedList(const StackLinkedList &other);
  ~StackLinkedList();

  StackLinkedList & operator=(const StackLinkedList<T> &other);

  void push(const T &elem);
  void pop();
  const T & top() const;
  T & top();
  bool empty() const;

private:
   ...
};
```

Operaciones push( ) y pop( )

```
void push(const T &elem) {
  top_node = new Node{ elem, top_node };
}
```

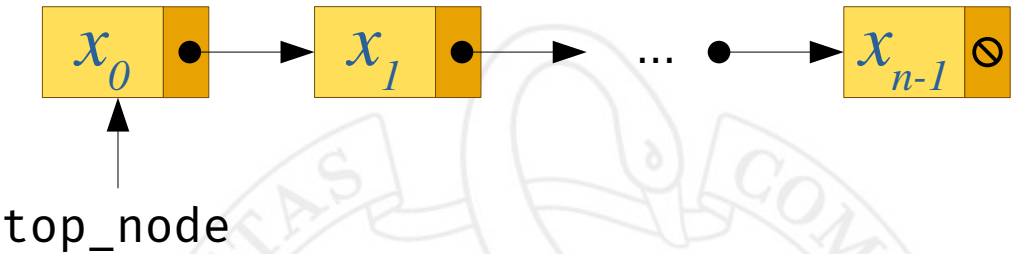



```
void pop() {
  assert (top_node ≠ nullptr);
  Node *target = top_node;
  top_node = top_node→next;
  delete target;
}
```



Operaciones top( ) y empty( )

```
const T & top() const {
  assert (top_node ≠ nullptr);
  return top_node→value;
}

bool empty() const {
  return (top_node == nullptr);
}
```



Coste de las operaciones

Operación	Vectores	Listas enlazadas
`push`	$O(n)$ / $O(1)$	$O(1)$
`pop`	$O(1)$	$O(1)$
`top`	$O(1)$	$O(1)$
`empty`	$O(1)$	$O(1)$

n = número de elementos en la pila