Repaso - Parcial 1

Marcucci, Ricardo Martín

07-10-2020

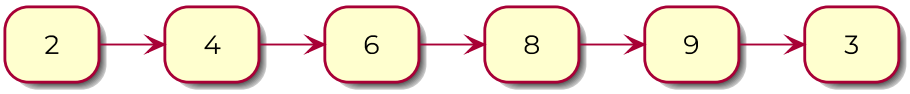
Repaso

Ejercicio 1

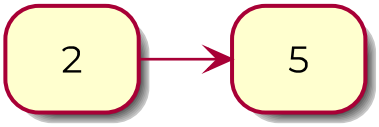
Hacer una función que reciba una lista y una pila, la función deberá modificar la lista original, eliminando las posiciones indicadas por cada nodo de la pila.

ej

Lista:

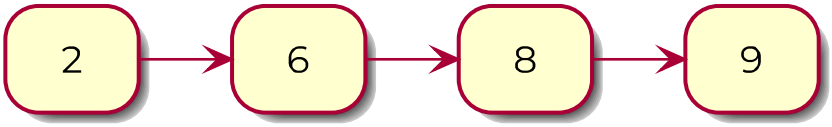


Pila




```
void eliminar_desde_pila(Stack<unsigned int> &p, List<int> &l);
...
List<int> miLista;
Stack<unsigned int> miPila;
eliminar_desde_pila(miPila, miLista);
```

Lista Nueva:



Ejercicio 2

Escriba una función void recursiva que reciba como parámetro solo un entero positivo **n** y que despliegue todos los enteros impares menores a **n**.

ej

```
void rango(int n);
...
rango(6); // muestra   1 3 5
```

Ejercicio 3

Escribir una función recursiva que tenga solo un parámetro entero positivo **n** y muestre en la pantalla ese número de asteriscos **"*"**, todos en una línea.

ej

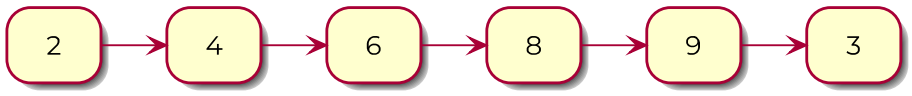
```
void asteriscos(int n);
...
asteriscos(4); // muestra   ****
```

Ejercicio 4

Escribir un método que busque en una lista enlazada un valor y mueva la primera ocurrencia del valor a la primera posición. *(implementar moviendo los nodos mediante enlaces, no se pueden crear o eliminar nodos)*

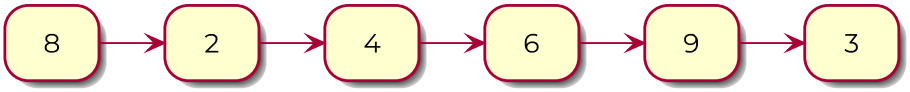
ej

Lista:




```
void List::moverPri(T val){};
...
List<int> miLista;
...
miLista.moverPri(8);
```

Lista Nueva:



Ejercicio 5

Hacer una función que dada una frase ingresada por teclado la imprima invertida. Implementar la función con la estructura de datos vista más adecuada.

ej

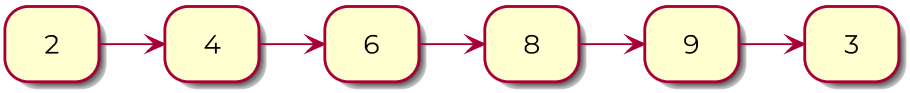
```
void invertir(string n);
...
string txt = "una imagen vale por mil palabras";
invertir(txt); // muestra   palabras mil por vale imagen una
```

Ejercicio 6

Escribir un método que busque en una lista enlazada un valor y mueva la primera ocurrencia del valor a la ultima posición. *(implementar moviendo los nodos mediante enlaces, no se pueden crear o eliminar nodos)*

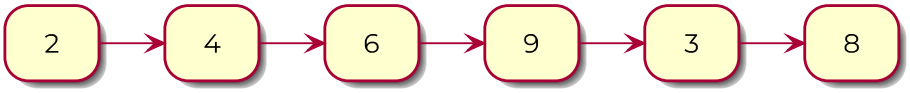
ej

Lista:




```
void List::moverUlti(T val){};
...
List<int> miLista;
...
miLista.moverUlti(8);
```

Lista Nueva:



Ejercicio 7

Construir una función que sume los elementos de un arreglo de enteros recursivamente.

ej

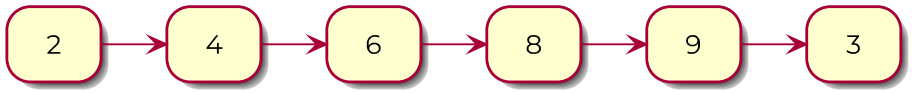
```
int sumarArreglo(int arr[], int tam);
...
int a[] = {3,5,7,9};
cout << sumarArreglo(a); // Muestra 24
```

Ejercicio 8

Escribir un metodo de la clase lista que, de manera **recursiva**, cuente cuantos nodos posee la lista enlazada.

ej

La Lista



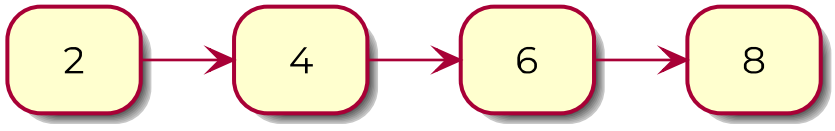

```
void List::size(){};
...
List<int> miLista;
...
cout << miLista.size(); // Muestra 6
```

Ejercicio 9

Se pretende realizar una función que, reciba por referencia una cola e invierta todo su conenido.

ej

Datos en la cola:



```
template <class T>
void invertir(Queue<T> q){};
...
Queue<int> miQueue;
...
invertir(miQueue);
```