DOCUMENTO COLABORATIVO

EJERCICIOS DE PUNTEROS

- **Ejercicio 10: Operaciones con punteros**
  - Representar gráficamente los resultados parciales y especificar el valor de *p y *q

```
inicializar(p)
inicializar(q)
*q←8
p←q
liberar(q)
*p←12
```

~~p -> *p~~

12 .

~~q -> *q~~

p ->*p

LISTAS
puede usar en listas enlazadas simples:
insertarNodoInicio(lista,valor) → O(1)
insertarNodoFinal(lista,valor) → O(n)
insertarNodoDespues(lista,valor,datoAnterior) → O(n)
eliminarNodoInicio(lista) → O(1)
eliminarNodoFinal(lista) → O(n)
eliminarNodoDato(lista,dato) → O(n)

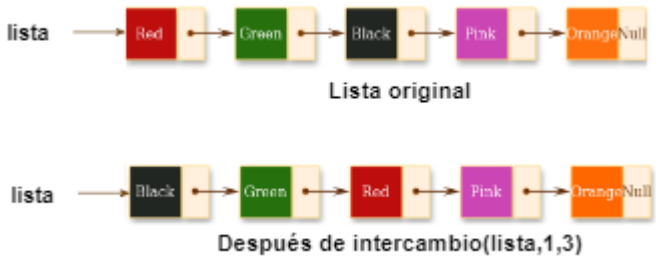
LES Buscar: Escribir un algoritmo para buscar un cierto dato en una lista enlazada
Solución:

```
buscar(lista L, num valor)
        ptr <-  L
        mientras ptr <> NULO hacer
                if (ptr->dato = valor ) then
                        return(true)
                ptr <- ptr->puntero
        return(false)
```

O(n)

pila->puntero
pila->tope->puntero

LES intercambiar: Escribir un algoritmo para que dada una lista enlazada simple y la posición de dos de sus elementos, devuelva la misma lista pero con los elementos indicados intercambiados. Cuál es el orden de complejidad del algoritmo propuesto?



solución a, Responsable:

LES comparar 2 listas: Escribir un algoritmo para que, dadas dos listas enlazadas simples, devuelva verdadero si ambas listas son iguales, y falso en caso contrario. Cuál es el orden de complejidad del algoritmo propuesto?

Listas originales

Falso

Después de comparar lista1 y lista2

Solución:

Dudas planteadas por estudiante:

1.-Una misma lista se puede asignar a 2 punteros auxiliares? esto aparece en el código destacado con un rectángulo rojo. Respuesta: si, puedes asignar los punteros que quieras, no hay problema con eso. podrías ir avanzando en elementos diferentes después uno con otro.

2.- ¿se puede usar la función recorrer lista para contar la cantidad de elementos de una lista enlazada? (comentando que se hace esa modificación) Respuesta: No, puedes basarte en recorrer lista y la modificas y la invocas retornando un numero por ejemplo.

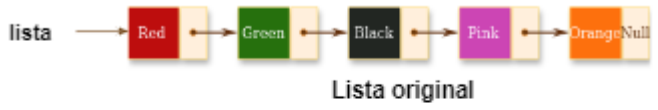
Solución planteada por estudiante:

```
comparacion (lista L1, lista L2) : boolean
    cont1 ← 0
    cont2 ← 0
    ptr1 ← L1
    ptr2 ← L2
    ptr3 ← L1
    ptr4 ← L2
    while (ptr1 <> null) do
        cont1 ← cont1 + 1
        ptr1 ← ptr1 ⇒ puntero
    while (ptr2 <> null) do
        cont2 ← cont2 + 1
        ptr2 ← ptr2 ⇒ puntero
    if (cont1 <> cont2) THen
      return (0)
    while (ptr3 <> null) do
        if (ptr3 ⇒ dato <> ptr4 ⇒ dato)
            return (0)
        ptr3 ← ptr3 ⇒ puntero
    return (1)
```

retroalimentación: que veo que hace : recorre con ptr1 la lista L1 contando en cont1 los elementos, recorre con ptr2 la lista L2 contando en cont2 los elementos, si tienen distinta cantidad de elementos retorna 0 como falso ya que no serían iguales. Siguiendo el algoritmo, asumiendo que son iguales las cantidades es que compara los elementos de L1 y L2 ahora recorriendolos con ptr3 y ptr4. Si en algún momento son distintos retorna 0. Si termina todo ok, retorna 1.

Mejoras: Cumple, pero veo uso de muchos punteros, podrías hacer lo mismo y reasignar los mismos ptr1 y ptr2 después de contar para comparar.
Se podría evitar los dos while iniciales de conteo, ya que al ir comparando en el tercer while elemento a elemento, si son de longitudes distintas va a encontrar una diferencia.
Se debería liberar los punteros creados para el problema, y eso en general tomarlo asi.. lo que se crea para resolver un problema (en este caso 4 punteros) hay que liberarlo al final, en este caso liberar antes de los retornos. Esto aplica para listas, pilas, colas o lo que se cree para resolver un problema. En este caso las listas no se liberan pues vienen ya con el problema, no son creadas en él.

LES eliminar: Escribir un algoritmo para eliminar todos los elementos de una lista enlazada simple. Después de eliminar cada elemento, lo que queda debe seguir siendo una lista enlazada. Cuál es el orden de complejidad del algoritmo propuesto?



Lista original

Lista final

Solucion a,

```
liberarLista(lista L): lista
        while(!esListaVacia(L)) do
                eliminarNodoInicio(L)
```

Complejidad O(n)

LES unir: Escribir un algoritmo para que dadas dos listas enlazadas simples, devuelva una lista nueva formada por los elementos de la primera lista, seguidos por los elementos de la segunda lista. Cuál es el orden de complejidad del algoritmo propuesto?

Solución :

LDE unir: Escribir un algoritmo para que dadas dos listas doblemente enlazadas, devuelva una lista nueva formada por los elementos de la primera lista, seguidos por los elementos de la segunda lista. ¿Cuál es el orden de complejidad del algoritmo propuesto?

Listas originales

Después de unir lista1 y lista2

Pilas

- **Ejercicio 1**: Escribir un algoritmo para que dada una pila, devuelva la cantidad de elementos de dicha pila. La pila ingresada debe mantenerse idéntica después de finalizar el algoritmo.
- **Ejercicio 2**: Escribir un algoritmo para que dada una pila y un valor, sumerja dicho valor en la pila (colocándolo en el fondo de la pila)
- **Ejercicio 3**: Escribir un algoritmo para desfondar una pila (quitar el elemento del fondo de la pila)
- **Ejercicio 4**: Escribir un algoritmo para cambiar a base 2 un número dado, usando TDA pila
- **Ejercicio 5**: Escribir un algoritmo para verificar los paréntesis de una expresión aritmética, usando TDA pila

- **apilar(pila,dato)**: agrega un elemento al comienzo de la pila (tope) → **push**
- **desapilar(pila)**: quita el primer elemento en el tope de la pila, sin devolverlo → **pop**
- **tope(pila)**: devuelve el primer elemento de la pila, sin extraerlo → **cima**, **top**
- **esPilaVacía(pila)**: comprueba si la pila está vacía

contar(pila P): num

return(P)

Colas

- **Ejercicio 1**: Escribir un algoritmo para que dada una cola de enteros, devuelva el mínimo valor en la cola. La cola ingresada debe mantenerse idéntica después de finalizar el algoritmo.
- **Ejercicio 2**: Escribir un algoritmo para que dada una cola y un valor entero, devuelva verdadero en caso de encontrar el valor en la cola, y falso en caso contrario.
- **Ejercicio 3:** Escribir un algoritmo para que dada una secuencia de enteros, muestre dicha secuencia en orden inverso al ingresado, usando TDA cola

- **encolar(cola,dato)**: agrega un elemento al final (back) de la cola → enqueue
- **descolar(cola)**: quita el primer elemento en el frente (front) de la cola → dequeue
- **frente(cola)**: obtiene el primer elemento de la cola, sin extraerlo → front
- **final(cola)**: obtiene el último elemento de la cola, sin extraerlo → back
- **esColaVacía(cola)**: comprueba si la cola está vacía

p1 2022/01

Dada una cadena de caracteres consistente de letras minúsculas del idioma español y paréntesis, representada como una **lista simplemente enlazada**, escribir un algoritmo en seudocódigo que revierta las subcadenas dentro de cada par de paréntesis. La idea es partir desde las subcadenas en paréntesis más anidados, hacia afuera. El algoritmo debe devolver una **lista simplemente enlazada** que representa una cadena sin ningún paréntesis. Por ejemplo, si la cadena de entrada es "((mo)tiro(alg))", la cadena de salida debería ser "algoritmo". Solo se pueden ocupar los **TDAs revisados en la unidad**. Se debe **indicar claramente las estructuras de datos utilizadas**. Calcular y justificar la complejidad del algoritmo propuesto.

Dada una cadena de caracteres consistente de dígitos del 0 al 9 y paréntesis, representada como una **lista simplemente enlazada**, escribir un algoritmo en seudocódigo que revierta las subcadenas dentro de cada par de paréntesis. La idea es partir desde las subcadenas en paréntesis más anidados, hacia afuera. El algoritmo debe devolver una **lista simplemente enlazada** que representa una cadena sin ningún paréntesis. Por ejemplo, si la cadena de entrada es "((84)7894(732))", la cadena de salida debería ser "732498784". Solo se pueden ocupar los **TDAs revisados en la unidad**. Se debe **indicar claramente las estructuras de datos utilizadas**. Calcular y justificar la complejidad del algoritmo propuesto.

(48 7894 237)

732498784

((8 4) 7 8 9 4 (7 3 2))

4
8
7
8
9
4
2
3
7

pila

cola -> 7 3 ---4

lista 7 8 4

Revertir (lista cadena): lista

aux ← cadena

pilaAux ← crearPilaVacia()

colaAux ← crearPilaVacia()

listaAux ← crearListaVacia()

MIENTRAS aux <> NULL HACER

SI aux→dato <> ')' ENTONCES

***apilar**(pilaAux, aux→dato)*

SI NO

*MIENTRAS **tope**(pilaAux)→dato <> '(' HACER*

***encolar**(colaAux, **tope**(pilaAux)→dato)*

```
                    desapilar(pilaAux)
                desapilar(pilaAux)    …para sacar '('
                MIENTRAS NO (esColaVacia(colaAux)) HACER
                    apilar(pilaAux, frente(colaAux)→dato)
                    descolar(colaAux)
            aux← aux→puntero
        MIENTRAS NO (esPilaVacía(pilaAux)) HACER
            insertarNodoInicio(listaAux, tope(pilaAux)→dato)
            desapilar(pilaAux)
        DEVOLVER(listaAux)
```

usando 1 pila, 1 cola y 1 lista

Cadena de entrada	pilaAux	colaAux	listaAux
((84)7894(732))	T→48((	F→ ←	→
((84)7894(732))	T→8((	F→ 4 ←	→
((84)7894(732))	T→((	F→ 48←	→
((84)7894(732))	T→ (	F→48←	→
((84)7894(732))	T→ 84(	F→ ←	→
((84)7894(732))	T→ 237(498784(	F→ ←	→
((84)7894(732))	T→(498784(	F→237 ←	→
((84)7894(732))	T→498784(	F→237 ←	→
((84)7894(732))	T→732498784(	F→ ←	→
((84)7894(732))	T→(	F→732498784 ←	→
((84)7894(732))	T→487894237	F→ ←	→
((84)7894(732))	T→	F→ ←	→ 732498784

Cadena de salida 732498784

Dada una secuencia de dígitos del 0 al 9, representada como una lista simplemente enlazada, escribir un algoritmo en seudocódigo que elimine los dígitos repetidos, tal que cada dígito aparezca una única vez. El algoritmo debe mostrar una lista simplemente enlazada que representa el menor número posible entre todos los posibles resultados. Por ejemplo, si la secuencia de entrada es *23123*, la secuencia de salida debería ser *123*. Solo se pueden ocupar los **TDAs lista simplemente enlazada** y **pila**. Calcular y justificar la complejidad del algoritmo propuesto.

Dada una secuencia de caracteres del alfabeto español, representada como una lista simplemente enlazada, escribir un algoritmo en seudocódigo que elimine los caracteres repetidos, tal que cada caracter aparezca una única vez. El algoritmo debe mostrar una lista simplemente enlazada que representa la cadena con menor orden lexicográfico entre todos los posibles resultados. Por ejemplo, si la secuencia de entrada es *bcabc*, la secuencia de salida debería ser *abc*. Solo se pueden ocupar los **TDAs lista simplemente enlazada** y **pila**. Calcular y justificar la complejidad del algoritmo propuesto.

p2 (lista secuencia)
… recorrer lista y eliminar los repetidos
… ordenar los elementos restantes 123 231 321 213

p3 2022/01

Describir un problema de la realidad, concreto y completo, de aplicación del **TDA lista simplemente enlazada circular**. Además, se deben explicar las respuestas a las siguientes preguntas: ¿en qué consistiría la estructura de datos para la situación particular?, ¿cómo se utilizarían las operaciones para resolver el problema específico? y ¿por qué el TDA más idóneo a utilizar sería el TDA lista simplemente enlazada circular?

Describir un problema de la realidad, concreto y completo, de aplicación del **TDA lista doblemente enlazada**. Además, se deben explicar las respuestas a las siguientes preguntas: ¿en qué consistiría la estructura de datos para la situación particular?, ¿cómo se utilizarían las operaciones para resolver el problema específico? y ¿por qué el TDA más idóneo a utilizar sería el TDA lista doblemente enlazada?