Auxiliar 5 - Estructuras II

Otoño 2025

Profesores: Alexandra Ibarra
Luis Mateu
Rodrigo Urrea

Auxiliares: Luciano Márquez
Tomás Vergara

Resumen

Punteros a otros punteros

Un puntero también puede apuntar a un puntero. Este tipo de variables contienen direcciones de otros punteros

```
int x = 3;
int* px = &x;               // puntero a un int

int** ppx = &px;            // puntero a un puntero a un int

printf("%p\n", *ppx)        // printea dirección de x
printf("%d\n", **ppx)       // printea x
```

Estos otros punteros pueden ser también punteros a estructuras

```
typedef struct{
  double x, y;
} Point;

Point p = {0.5, 0.4};
Point* pp = &p;                          // puntero a un Punto
Point** ppp = &pp;                       // puntero a un puntero a un Punto

printf("%p\n", *ppp);                    // printea la dirección de pp
printf("%1.f %1.f\n", (*ppp)->x, (ppp)->y); // printea 0.5 0.4
```

Punteros a funciones

Los punteros también pueden apuntar a funciones. Útil para usarlos como parámetros de otras funciones

```
int suma(int a, int, b){ return a + b; }  // int, int -> int

int (*f)(int, int) = &suma;              // debe coincidir con la firma de la función
printf("%d\n", f(4, 6));                 // printea 10
```

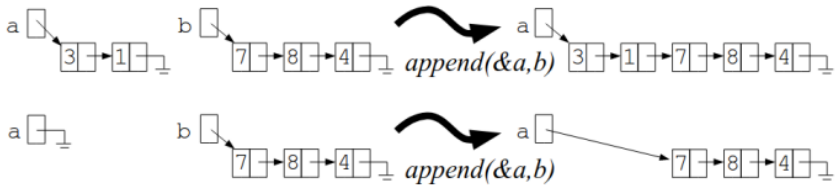
Preguntas

P1. Programe la función `append` que adjunta la lista enlazada `b` a la lista `pa`. El encabezado de la función es es siguiente:

```
typedef struct nodo {
  int x;
  struct nodo *sgte;
} Nodo;

void append(Nodo **pa, Nodo *b);
```

La función debe ser eficiente, es decir, su tiempo de ejecución debe ser proporcional al tamaño de la lista `pa`. No puede usar ciclos (como `while` o `for`). Debe usar recursión. No puede usar `malloc`. Reutilice los punteros recibidos y mantenga el orden. En la siguiente figura se muestra un ejemplo de uso:



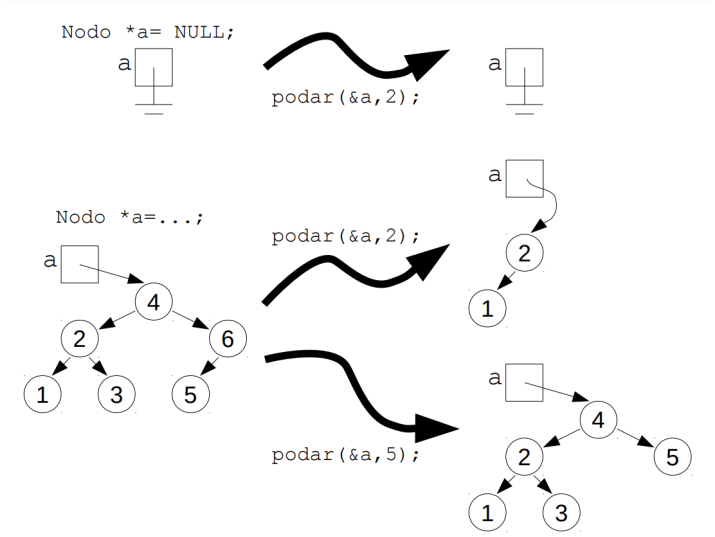
P2. **(P2 C1 Primavera 2017)** Sea la estructura:

```
typedef struct nodo {
  int x;
  struct nodo *izq, *der;
} Nodo;
```

Programe la función:

```
void podar(Nodo **pa, int y);
```

Esta función debe modificar un árbol de búsqueda binaria `*pa` eliminando todos los nodos etiquetados con valores mayores que `y`. No necesita liberar la memoria de los nodos eliminados. Estudie los 3 ejemplos de uso de la siguiente figura:



Restricciones: Sea eficiente, el tiempo de ejecución debe ser proporcional a la altura del árbol en el peor caso. No puede usar ciclos (como `while` o `for`). Debe usar recursión. No puede usar `malloc`.

P3. **(P2 C1 Otoño 2017)** Sea la estructura:

```
typedef struct nodo {
  char c;
  struct nodo *izq, *der;
} Nodo;
```

Programe la función:

```
int reemplazarNodoK(Nodo **pa, int k, Nodo *b);
```

Sea `a = *pa`. Esta función reemplaza el k-ésimo nodo del árbol a por el nodo b. El k-ésimo nodo de a es el k-ésimo nodo al enumerar los nodos de a en inorden. Por ejemplo en la figura 1, r es el nodo 1, s el 2, t el 3 y u el 4. Esta función retorna k si se hizo el reemplazo, es decir cuando el árbol a tenía al menos k nodos. Si no, entrega el número de nodos encontrados en a, que será inferior a k. Las siguientes figuras sirven para explicar algunos ejemplos de uso. Los punteros a y b E son de tipo `Nodo*`.

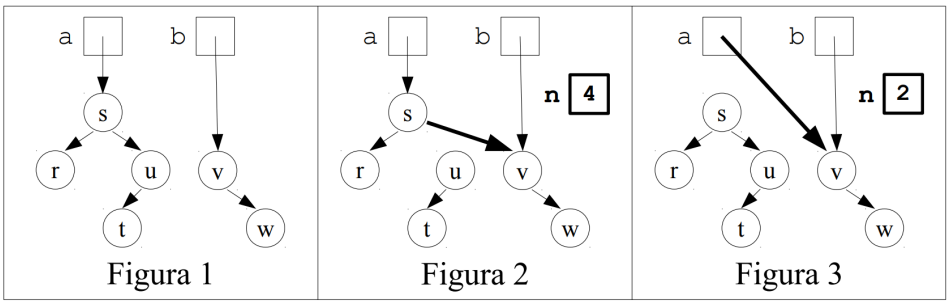


Figura 1 Figura 2 Figura 3

La figura 2 se obtiene cuando a partir de la figura 1 se llama:

```
int reemplazarNodoK(&a, 4, b);
```

Se reemplazó u por v y la función retornó 4. La figura 3 se obtiene cuando a partir de la figura 1 se llama:

```
int reemplazarNodoK(&a, 2, b);
```

Acá se reemplazó la raíz s del árbol por v. Por eso se requiere que el puntero a la raíz del árbol se pase por referencia (`&a`). Por último si a partir de la figura 1 se intentara reemplazar el quinto nodo de a que no existe, no se haría ningún reemplazo y la función retornaría 4.

P4. **[Propuesto]** Implementar las colas **FIFO** y pila **LIFO** utilizando listas enlazadas. Para esto, se deben implementar las funciones `push`, `pop`, y `peek` para la pila, y `enqueue`, `dequeue` y `peek` para la cola. Las estructuras de las listas enlazadas son las siguientes:

```
typedef struct nodo {
  int x;
  struct nodo *sgte;
} Nodo;
```

Las funciones deben tener la siguiente firma:

```
Cola *crearCola();
void enqueue(Cola *cola, int x);
int dequeue(Cola *cola);
int* peek (Cola *cola);
void liberarCola(Cola *colas);

Pila* crearPila();
void push(Pila *pila, int x);
int pop(Pila *pila);
int* peek(Pila *pila);
void liberarPila(Pila *pila);
```