Auxiliar 4

Estructuras de Datos

Profesores: Alexandra Ibarra, Luis Mateu y Rodrigo Urrea
Auxiliares: Luciano Márquez
Tomás Vergara
Semestre: Otoño 2025

Resumen

Operador *

El operador * es para acceder al contenido de un puntero (el valor que indica la dirección de memoria). Por ejemplo:

```
int x = 5;
int *p = &x;
int y = *p; // y = 5
```

También se puede cambiar el contenido de lo que apunta, por lo tanto, cambiar el valor de la variable. Continuando con el ejemplo anterior:

```
*p = 10; // Ahora x = 10
```

La utilidad de esto es que, si nosotros queremos que una función modifique una variable, podemos pasar un puntero a esa variable y la función puede modificarla. Por ejemplo:

```
void duplicar(int *p) {
  *p = *p * 2;
}

int x = 5;
duplicar(&x); // Ahora x = 10
```

Malloc y Free

Para utilizar funciones de alocación de memoria, como `malloc` y `free`, es necesario incluir la librería `stdlib.h`. La función `malloc` permite alocar un bloque de memoria de un tamaño específico y retorna un puntero a la dirección de memoria del bloque alocado. Mientras que `free` libera la memoria alocada previamente por `malloc`. Por ejemplo:

```
#include <stdlib.h>

int *p = malloc(sizeof(int)*10); // Alocamos un bloque de memoria para 10 enteros
p[0] = 5;
free(p); // Liberamos la memoria alocada
```

Estructuras

Las estructuras en C permiten agrupar distintos tipos de datos en una sola variable. Por ejemplo:

```
typedef struct p {
  int x;
  int y;
} Punto;

Punto p;

p.x = 5;
p.y = 10;
```

Al crear la estructura con `typedef`, podemos usar `Punto` como un tipo de dato, y no tener que escribir `struct p` cada vez que queramos declarar una variable de tipo `Punto`. Además, podemos acceder a los campos de la estructura con el operador `.`, como en `p.x`.

Preguntas

P1. Dada la función to_lower del auxiliar 3, escriba la función:

```
char *lowerCase(char *str);
```

La cual retorna una copia de str con las letras convertidas a minúscula, sin editar el string original. Luego, programe la misma función, sin llamar a to lower ni usando strcpy

P2. Vamos a implementar un árbol de búsqueda binaria donde sus nodos solamente aceptan strings (`char *c`).
Deberá implementar todo lo detallado en el código que sigue (abb.h), que representa el Árbol de Búsqueda Binaria:

```
typedef struct abbTree{
char* val;
struct abbTree *left, *right;
}Tree;

Tree* initTree(char* rootValue); // Inicializa la estructura

Tree* insertValue(Tree *root, char* value); // Inserta un valor, retorna root.
Debe agregar valores aunque esten duplicados, debe pedir memoria para cada nuevo
nodo que agregue

void printTree(Tree *root); // Imprime los valores guardados en el arbol en orden
creciente (recorrer en in orden, primero rama izquierda)

char* maxValue(Tree *root); // Retorna el valor maximo
char* minValue(Tree *root); // Retorn el valor minimo
char* find(Tree *root, char* value); // Busca un valor, si lo encuentra lo
retorna,de lo contrario retorna NULL
```

```
void destroyNode(Tree* node); // Libera la memoria del nodo.
Tree* delVal(Tree *root, char* value); // Busca si el valor esta en el arbol, si
loencuentra lo borra y libera el espacio del nodo. Tanto si tiene o no exito, debe
devolver root
void freeTree(Tree *root); // Libera la memoria utilizada por el arbol, debe
hacerlo recursivamente.
```