LIBRERÍA STDLIB.H

La librería stdlib.h contiene **muchas utilidades para los programadores**, tales como la gestión de memoria dinámica, la ordenación y búsqueda en arrays o el control de procesos.

La **especificación** de la librería es la siguiente:

Conversiones numéricas.

Function	Description
atof	Convierte una cadena de caracteres a float.
atoi	Convierte una cadena de caracteres a int.
atol	Convierte una cadena de caracteres a long.
strtod	Convierte una cadena de caracteres a double.
strtol	Convierte una cadena de caracteres a long.
strtoul	Convierte una cadena de caracteres a unsigned long.

Funciones matemáticas

Function	Description
abs	Valor absoluto.
labs	Valor absoluto.
div	División entera.
ldiv	División entera.

Widechar conversions

Function	Description
wcstombs	Convierte una cadena de caracteres anchos a una cadena multibyte.
wctomb	Convierte un carácter aun carácter multibyte.

Conversiones Multibyte

Function	Description
mblen	Devuelve el número de bytes de un carácter multibyte.
mbstowcs	Convierte de cadena multibyte a cadena ancha.
mbtowc	Convierte de cadena multibyte a carácter.

Manipulación de memoria.

Function	Description
calloc	Reserva memoria dinámica.
free	Libera memoria dinámica.
realloc	Reserva memoria dinámica.
malloc	Reserva memoria dinámica.

Control de procesos

Function	Description
abort	Termina la ejecución de forma anormal.
atexit	Registra una función callback para salir.
exit	Termina la ejecución del programa.
getenv	Recupera una variable de entorno.
system	Ejecuta un comando externo.

Generación de números aleatorios.

Function	Description
rand	Genera un número aleatorio.
srand	Inicializa el generador de números aleatorios.

Operadores

Function	Description
sizeof	Calcula el tamaño de cualquier tipo de dato.

Utilidades para arrays

Function	Description
bsearch	Realiza una búsqueda binaria.
qsort	Ordena el array por Quicksort.

LISTA ENLAZADA SIMPLE

Una **lista enlazada simple** consiste en nodos donde:

- Cada nodo contiene **un dato** y un **puntero al siguiente nodo**.
- El último nodo apunta a NULL.

Estructura del programa

1. Definición de la estructura Nodo

```
struct Nodo {
   int dato;
   struct Nodo* siguiente;
};
```

- int dato: almacena un valor entero.
- struct Nodo* siguiente: apunta al siguiente nodo de la lista.

2.Función crearNodo

```
struct Nodo* crearNodo(int valor) {
   struct Nodo* nuevo = (struct Nodo*) malloc(sizeof(struct Nodo));
   ...
   nuevo->dato = valor;
   nuevo->siguiente = NULL;
   return nuevo;
}
```

- Usa malloc para reservar memoria dinámica.
- Asigna el valor recibido al campo dato.
- El siguiente es NULL porque aún no se conecta a ningún nodo.
- Devuelve el puntero al nuevo nodo.

3. Función agregarAlFinal

```
void agregarAlFinal(struct Nodo** cabeza, int valor)
```

- Agrega un nuevo nodo **al final de la lista**.
- Usa struct Nodo** cabeza para poder modificar la cabeza desde fuera de la función (puntero al puntero).
- Si la lista está vacía (*cabeza == NULL), el nuevo nodo será el primero.
- Si no, recorre la lista hasta el último nodo y lo enlaza al nuevo nodo.

Función imprimirLista

```
void imprimirLista(struct Nodo* cabeza)
```

- Recorre desde el inicio de la lista (cabeza) hasta que el puntero sea NULL.
- Imprime cada valor seguido de ->.
- Muestra NULL al final para indicar el fin de la lista.

Función liberarLista
void liberarLista(struct Nodo* cabeza)

- Libera la memoria usada por cada nodo para evitar fugas de memoria.
- Recorre nodo por nodo, guarda el actual en temp, avanza al siguiente y luego libera temp.

6. Función main

```
int main() {
   struct Nodo* lista = NULL;
   ...
}
```

- Crea una lista vacía.
- Muestra un menú al usuario para:
    1. Agregar un valor
    2. Mostrar la lista
    3. Salir (y liberar memoria)

Utiliza un bucle do...while para que el menú se repita hasta que el usuario elija salir.

CODIGO LISTAS -1

Explicación funciones:

- crearNodo(int valor): crea un nuevo nodo con el valor dado.
- agregarAlFinal(...): agrega el nodo al final de la lista.
- imprimirLista(...): muestra todos los elementos de la lista.
- liberarLista(...): libera toda la memoria al final.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

// Estructura de un nodo
struct Nodo {
   int dato;
   struct Nodo* siguiente;
};

// Función para crear un nuevo nodo
struct Nodo* crearNodo(int valor) {
   struct Nodo* nuevo = (struct Nodo*) malloc(sizeof(struct Nodo));
   if (nuevo == NULL) {
      printf("Error: no se pudo asignar memoria.\n");
```

```
        exit(1);
    }
    nuevo->dato = valor;
    nuevo->siguiente = NULL;
    return nuevo;
}

// Función para agregar un nodo al final
void agregarAlFinal(struct Nodo** cabeza, int valor) {
    struct Nodo* nuevo = crearNodo(valor);

    if (*cabeza == NULL) {
        *cabeza = nuevo;
    } else {
        struct Nodo* actual = *cabeza;
        while (actual->siguiente != NULL) {
            actual = actual->siguiente;
        }
        actual->siguiente = nuevo;
    }
}

// Función para imprimir la lista
void imprimirLista(struct Nodo* cabeza) {
    printf("Lista: ");
    while (cabeza != NULL) {
        printf("%d -> ", cabeza->dato);
        cabeza = cabeza->siguiente;
    }
    printf("NULL\n");
}

// Liberar memoria
void liberarLista(struct Nodo* cabeza) {
    struct Nodo* temp;
    while (cabeza != NULL) {
        temp = cabeza;
        cabeza = cabeza->siguiente;
        free(temp);
    }
}

int main() {
    struct Nodo* lista = NULL;
```

```
    int opcion, valor;

    do {
        printf("\n--- MENU ---\n");
        printf("1. Agregar nodo\n");
        printf("2. Mostrar lista\n");
        printf("3. Salir\n");
        printf("Seleccione una opcion: ");
        scanf("%d", &opcion);

        switch (opcion) {
            case 1:
                printf("Ingrese un valor entero: ");
                scanf("%d", &valor);
                agregarAlFinal(&lista, valor);
                break;
            case 2:
                imprimirLista(lista);
                break;
            case 3:
                liberarLista(lista);
                printf("Memoria liberada. Adios.\n");
                break;
            default:
                printf("Opcion invalida.\n");
        }
    } while (opcion != 3);

    return 0;
}
```