Algoritmos y Estructuras de Datos II

TALLER - 15 de mayo 2025

Laboratorio 4 bis: Principios de listas enlazadas

- Revisión 2025: Franco Luque

Código

lab04-2-kickstart.tar.gz

Objetivos

1. Trabajar más con punteros y memoria dinámica en C
2. Comenzar a trabajar con listas enlazadas en C
3. Probar visualizadores de ejecuciones de código

Recursos

Recursos generales:

- Videos del Laboratorio en el aula virtual
- Documentación en el aula virtual
- Estilo de codificación:
  - Guía de estilo para la programación en C
  - Consejos de Estilo de Programación en C

Recursos específicos:

- Punteros: tipos * , desreferenciación o indirección (*), referenciación o dirección (&)
  - 04 - Punteros en C
- Memoria dinámica: sizeof(), malloc(), y free() (también malloc() y realloc())
  - 06 - Memoria Dinámica en C
- Tomos: https://github.com/jmansilla/tomos
- Visualizador para C: https://pythontutor.com/c.html#mode=edit

Ejercicio 1: Tutorial de Tomos

Hacer el siguiente tutorial:

Tutorial de Tomos

No es necesario resolverlo por completo, pero por lo menos completar los dos ejemplos principales: “setup” y “append”.

Este ejercicio **no se trata del lenguaje C** si no del lenguaje del teórico/práctico.

Ejercicio 2: Construcción de listas enlazadas

Se proveen en **`main.c`** definiciones de tipo para listas enlazadas y sus nodos:

```
typedef struct Node {
   int data;
   struct Node* next;
} Node;

typedef Node *List;
```

Se debe completar la implementación de la función `setup_example()` que construye y devuelve una lista de ejemplo de 3 elementos. La lista debe construirse de la misma forma que en el ejemplo `setup.ayed` visto en el ejercicio anterior:

https://github.com/jmansilla/tomos/blob/main/demo/linked_list/setup.ayed

Compilar, ejecutar y verificar que la salida en pantalla es la correcta:

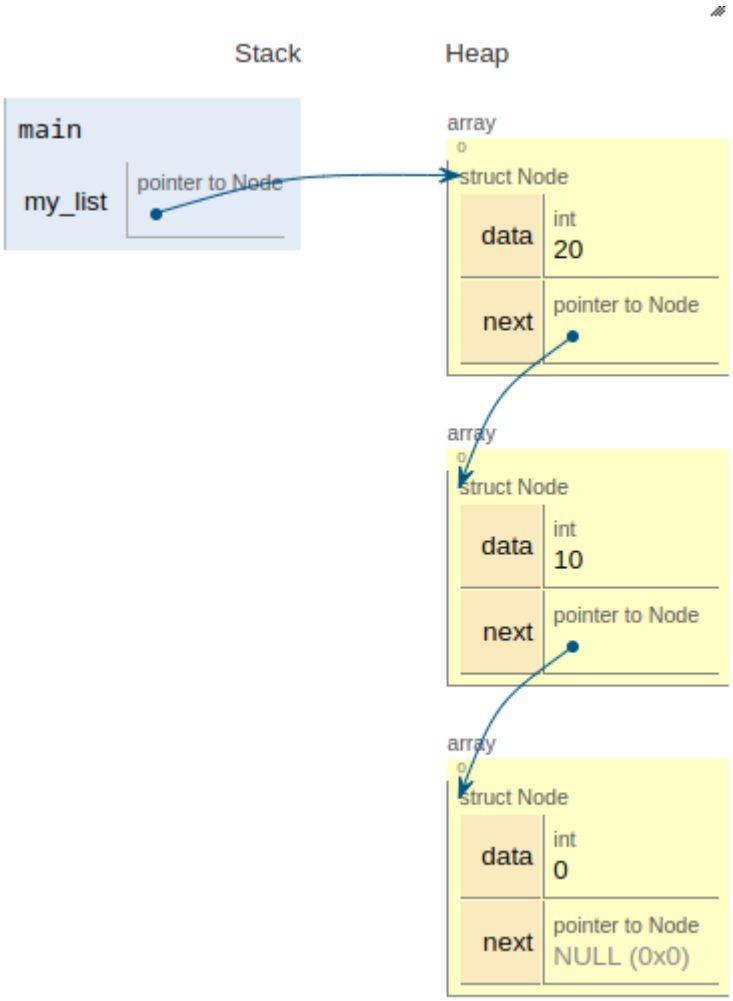
```
$ gcc -Wall -Wextra -pedantic -std=c99 main.c -o main
$ ./main
[ 20, 10, 0, ]
```

Ejercicio 2-b: Visualización de listas enlazadas en C

Tomar el código del ejercicio anterior e introducirlo en el siguiente sitio que permite la visualización de ejecuciones de código en lenguaje C:

https://pythontutor.com/c.html#mode=edit

El estado final debería verse de la siguiente manera:



Ejercicio 3: Agregando un elemento al final

Además de `setup_example()` del ejercicio anterior, completar en `main.c` la implementación de la función `append_example()` que agrega un elemento de ejemplo al final de la lista. Basarse en el ejemplo `append.ayed`:

https://github.com/jmansilla/tomos/blob/main/demo/linked_list/append.ayed

Compilar, ejecutar y verificar que la salida en pantalla es la correcta:

```
$ gcc -Wall -Wextra -pedantic -std=c99 main.c -o main
$ ./main
Lista antes del append: [ 20, 10, 0, ]
Lista después del append: [ 20, 10, 0, 88, ]
```

Ejercicio 4: Eliminando un elemento al principio

Completar en `main.c` la implementación de la función tail_example() que elimina el primer elemento de una lista. Basarse en el ejemplo `tail.ayed`:

https://github.com/jmansilla/tomos/blob/main/demo/linked_list/tail.ayed

Compilar, ejecutar y verificar que la salida en pantalla es la correcta:

```
$ gcc -Wall -Wextra -pedantic -std=c99 main.c -o main
$ ./main
Lista antes del tail: [ 20, 10, 0, ]
Lista después del tail: [ 10, 0, ]
```