UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA

Desarrollo de Software y Tecnologías Emergentes

Taller U2-2: Manejo de memoria

Paradigmas de programación

ALUMNOS: Barrios Retana Lizeth y Rios Rodriguez Daniel

MATRÍCULA: 372813 y 372800

GRUPO: 941

PROFESOR: Carlos Gallegos

Ensenada, Baja California a 11 de marzo de 2024.

Modelo de Memoria en C:

En C, el modelo de memoria es más cercano al hardware y brinda un mayor control sobre la gestión de la memoria por parte del programador.

Algunos aspectos clave incluyen:

- **Stack y Heap:**

Stack: Utilizado para almacenar variables locales y control de flujo. Es de tamaño fijo y gestionado de manera eficiente por el compilador.
Heap: Se utiliza para la asignación dinámica de memoria mediante funciones como malloc y free. El programador es responsable de gestionar la memoria en el heap.

- **Punteros:**

Los punteros permiten acceder directamente a las direcciones de memoria. El programador tiene control total sobre la manipulación de punteros.

- **Gestión manual de la memoria:**

El programador debe liberar explícitamente la memoria asignada dinámicamente para evitar fugas de memoria.

- **Modelo de Memoria en Python:**

En Python, el modelo de memoria es más abstracto y gestionado automáticamente por el intérprete. Algunos aspectos clave incluyen:

- **Gestión Automática de Memoria:**

Python utiliza un recolector de basura para gestionar automáticamente la asignación y liberación de memoria. El programador no necesita preocuparse por detalles como la liberación de memoria.

- **Objetos y Referencias:**

Todo en Python es un objeto, y las variables son referencias a objetos. La memoria se asigna y libera dinámicamente según sea necesario.

- **Sin Punteros Explícitos:**

Python oculta la manipulación directa de punteros, simplificando el código y reduciendo la posibilidad de errores relacionados con la memoria.

Aspecto	Lenguaje	Lenguaje Python
Gestión de memoria	Manual, el programador tiene control	Automática, a cargo del recolector de basura
Stack y Heap	Existe stack y heap	No hay distinción clara
Punteros	Manejo explícito por el programador	Ocultos, no hay manipulación directa
Recolector de basura	No tiene	Presente, gestiona automáticamente la memoria
Asignación Dinámica	Sí, con malloc y free	Sí, a través de la creación de objetos
Control del programador	Mayor control, mayor riesgo de errores	Menos control, menos riesgo de errores

Conclusión:

En conclusión ambos lenguajes tienen enfoques diferentes para la gestión de memoria, con C brindando más control al programador y Python automatizando gran parte de la gestión de memoria. La elección entre ambos depende de las necesidades del proyecto y las preferencias de desarrollo.

Bibliografía:

Gestión de la memoria. (n.d.). Python Documentation. https://docs.python.org/es/3/c-api/memory.html

Programación II - Tipos de Memoria. (n.d.). https://sites.google.com/site/programacioniiuno/temario/unidad-1---manejo-de-memoria-dinmica/tipos-de-memoria

Bloques de memoria — Fundamentos de Programación en C++. (n.d.). https://www2.eii.uva.es/fund_inf/cpp/temas/2_tipos_variables/bloques_memoria.html

¿Se pueden usar punteros de memoria en Python? (n.d.). Quora. https://es.quora.com/Se-pueden-usar-punteros-de-memoria-en-Python