힙 메모리

2021년 2월 11일 목요일 오전 6:47

- 스택 메모리의 단점 1 - 수명
  - 함수가 반환하면 그 안에 있던 데이터가 날라감
    - 즉, 함수 안에 있는 변수의 수명(lifecycle)은 함수가 끝날 때까지
  - 그러지 않고 데이터를 오래 보존하려면 전역 변수, 또는 static 키워드를 사용해야 했음
    - 이런 변수의 수명은 프로그램 실행 내내
  - 이건 도 아니면 모네?
  - 근데 어떤 경우에는 그 중간 어딘가에서 타협을 원할 수도?
    - 내가 원할 때 만들거나 지울 수 있는 그런 저장 공간?
- 스택 메모리의 단점 2 - 크기
  - 앞에서도 말했듯 특정 용도에 쓰라고 별도로 떼어 놓은 메모리
  - 그 크기는 컴파일 시에 결정하므로 너무 크게 못 잡음
  - 프로그램을 실행할 시스템의 메모리가 1MB일 수도 있고 4GB일 수도
    - 최소한에 맞출 수 밖에
  - 그래서 엄청 큰 데이터를 처리해야 할 경우 스택 메모리에 못 넣음
    - 예: 4K로 녹화해서 파일 크기가 2GB인 동영상 파일
- 힙 메모리
  - 컴퓨터에 존재하는 범용적 메모리
  - 스택 메모리처럼 특정 용도로 떼어 놓은 게 아님
  - 스택과 달리 컴파일러 및 CPU가 자동적으로 메모리 관리를 안 해줌
  - 따라서 프로그래머가 원하는 때 원하는 만큼 메모리를 할당받아와 사용하고 원할 때 반납(해체)할 수 있음
- 힙 메모리의 장점
  - 용량 제한이 없음
    - 컴퓨터에 남아있는 메모리만큼 사용 가능
  - 프로그래머가 데이터의 수명을 직접 제어
    - 스택에 저장되는 변수처럼 함수 호출이 끝나면 사라지지 않음
    - 전역 변수처럼 프로그램이 실행되는 동안 계속 살아있는 것도 아님
- 힙 메모리의 단점 (1)
  - 빌려온 메모리를 직접 해제 안 하면 누구도 그 메모리를 쓸 수 없음
    - 그 메모리는 계속 '누군가'에게 빌려준 상태
    - 만약, 빌려간 쪽에서 그 메모리 주소를 잃어버리면 메모리 누수가 발생
- 그래서 C는 언매니지드 언어다!
  - 매니지드 언어(예: C#. JAVA 등)는 메모리 해제를 알아서 해주는 언어
  - 이를 메모리를 관리(manage)해준다고 하여 매니지드 언어라 함
  - 이 메모리 관리 기능은 다른 훌륭한 프로그래머들이 구현한 것
    - 메모리 누수가 날 가능성이 적음
    - 당연히 범용적으로 만든 거라 속도 등이 느릴 수 있음

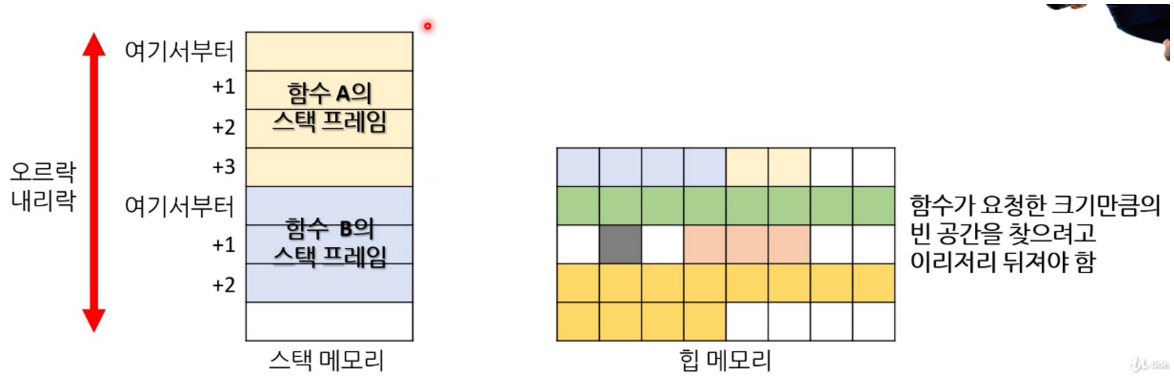
- C는 그런 언어가 아니기에 프로그래머가 직접 해줘야 함
  - 훌륭한 프로그래머라면 메모리 관리쯤은 할 수 있음
  - 최대의 효율성을 선택하는 대신
  - 실수를 막기 위해 여러가지 원칙들을 습관화

비유: 매니지드 vs 언매니지드 이사

- 매니지드
  - 이삿짐센터가 우리 집에 방문해서 모든 짐을 싼 뒤, 새 집으로 옮겨주고, 짐까지 풀어줌
  - 그 동안 우리는 달다방에 가서 커피나 한 잔 하며 놀아도 됨
  - 그러나 이삿짐센터가 어떤 물건을 조심히 다뤄야 하는지 몰라서 좀 깨짐
- 언매니지드
  - 직접 박스까지 공수해 와서 짐 싸고, 트럭에 나르고, 새 집으로 이사한 뒤, 짐도 다 품
  - 허리가 아픔
  - 그러나 내 아까운 피규어가 깨지진 않음

힙 메모리의 단점 (2)

- 스택에 비해 할당/해제 속도가 느림
  - 스택은 오프셋 개념 vs 힙은 사용/비사용 중인 메모리 관리 개념
  - 메모리 공간에 구멍이 생겨 효율적으로 메모리 관리가 어렵기도 함



정적 메모리 vs 동적 메모리

- **스택 메모리**는 **정적 메모리**
  - 이미 공간이 따로 잡혀 있음
  - 할당/해제가 자동으로 관리되게 코드가 컴파일 됨
  - 오프셋 개념으로 정확히 몇 바이트씩 사용해야 하는지 컴파일시 결정
- **힙 메모리**는 **동적 메모리**
  - 실행 중에 크기와 할당/해제 시기가 결정됨
- 이미 정적 메모리 관리는 잘 알고 있으니 이제 동적 메모리만 봅시다