day0528

🡺0.c

increaseSize - > 사용자가 원하는 만큼만 size를 증가시켜주자!

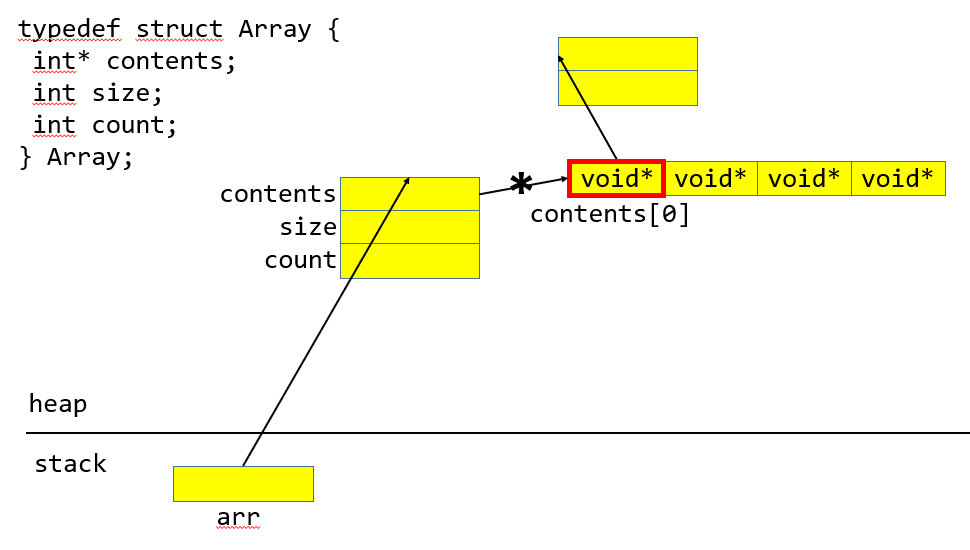
🡺1.c

현재 arrayCreate()에서는 미리 생성을 하고 데이터가 언제 들어올지는 모른다. 그래서 이를 해결하기 위해 데이터가 들어오는 시점에 생성을 하면 더 좋을 것이다.

최대한 자원생성을 뒤로 미루자!! (lazy Initialization)-‘지연된 초기화’

->즉, 언제 사용될지 모르는 배열에 대하여 미리 할당한다는 단점이 있으므로 메모리 낭비가 될수 있다. 이를 해결하기 위해 지연된 초기화를 도입해보자.

🡺2.c



그러나 위 코드도 단점이 하나 있다. 구조체 안에 int밖에 넣을수 없다는 것이다. 그렇기에 void \*로 바꾸어 generic 기법으로 바꿔보자.(모든 타입에 대하여 저장 가능한 자료구조로 변경해보자)

cf. sprintf : 표준출력이 아닌 버퍼로 보냄

🡺3.c

위 코드를 모듈화 하는코드

libarray.c / Array.h / 3\_1.c(main.c) 로 분리

🡺3\_1.c 수정하여 3\_2.c

구조체에 불완전 타입을 쓰고 캐스팅해서 쓰면 정보은닉이 가능해진다.

.h 에 있는 구조체를 libarray.c로 옮겨주고 .h에서는 정의만 해주면 정보은닉이 가능해진다.

🡺0527/0528끝!

동적배열의 장점 : 구현이 쉽다 / 임의 접근이 가능하다 / 해당 원소 접근이 빠르다(index를 가지고 바로 접근가능하므로) /

동적배열의 단점: 메모리가 낭비가 발생할수 있다(10을 만들엇는데 2만쓸경우 등..) / 삭제나 삽입시 오버헤드가 크다

->이러한 단점을 극복하기 위해 나온 것이 ‘연결리스트’이다.

(연결리스트의 단점 : 구현이 좀 복잡하다..(동적배열에 비해))