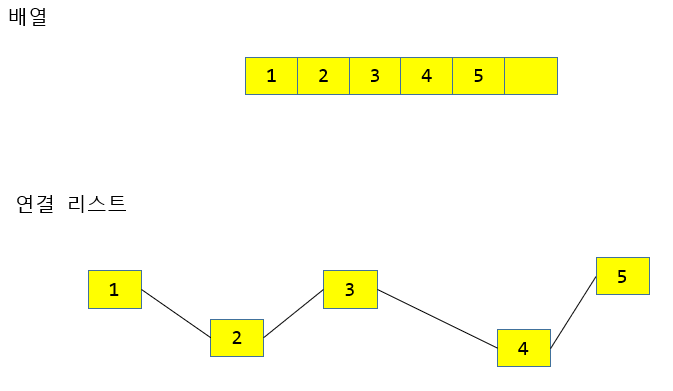
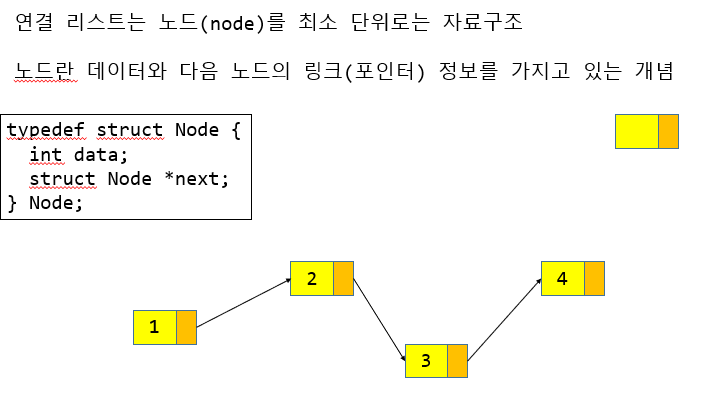
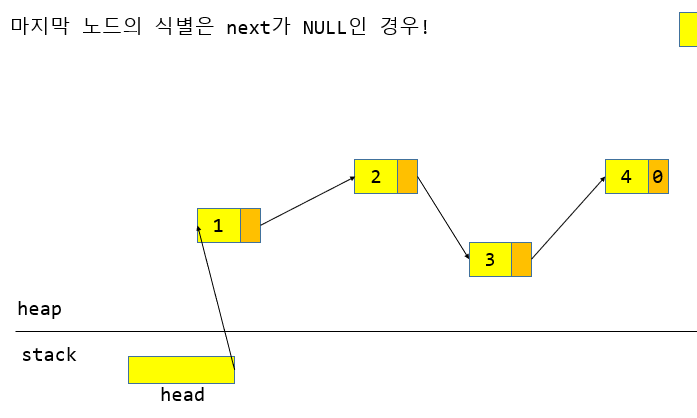
**<<<<<연결리스트>>>>>**



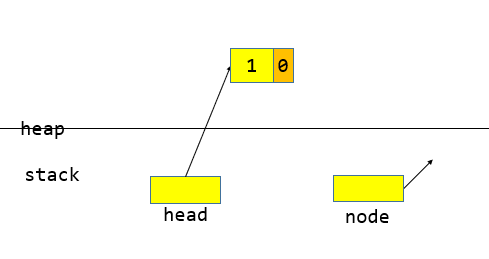


연결 리스트는 노드(node)를 최소단위로 하는 자료구조

노드란? – 데이터와 다음 노드의 링크(포인터) 정보를 가지고 있는 개념



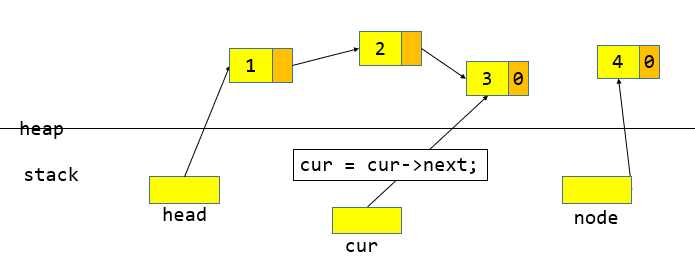
연결리스트는 힙에 존재하며 연결리스트의 제일 첫부분을 가르키는 것을 head라고한다.(head는 스택에 존재함)



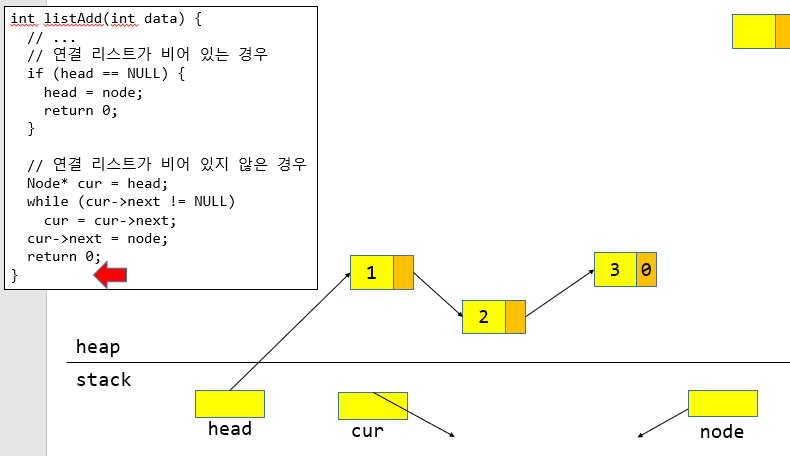
연결리스트의 마지막인지를 판별할 때 마지막 노드의 next가 null

🡺4.c

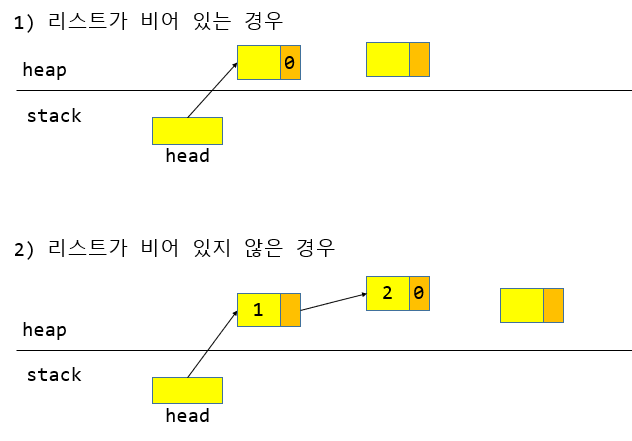
데이터를 마지막에 추가하는 listAdd함수 구현



리스트가 비어있지 않은경우에

위 예제 정리!!

🡺5.c

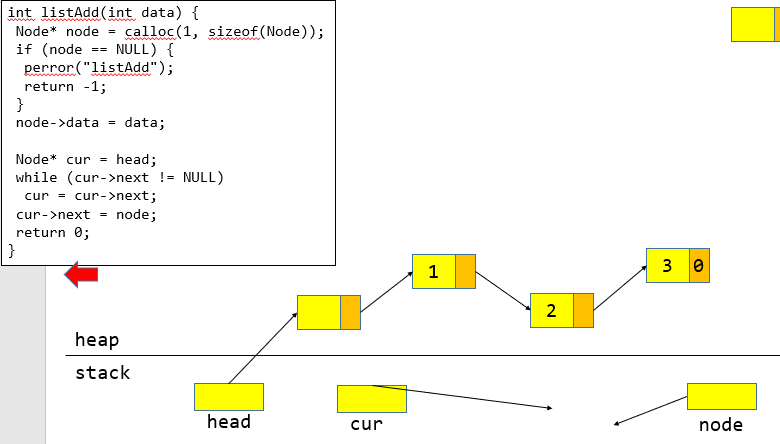


이전코드는 삽입정책이 일반화되어있지않다. (삽입 알고리즘이 이원화 되어있다!!)

리스트가 비어있는경우엔 head는 ‘**포인터**’/ 리스트가 비어있지않은경우엔 그 전 노드는 ‘**진짜노드**’

그것을 일치시켜주기위해 앞으론 새로운 노드앞쪽에 ‘진짜노드’가 있도록 맞춰주자!!

그러한 노드를 ‘더미헤드’라고한다.



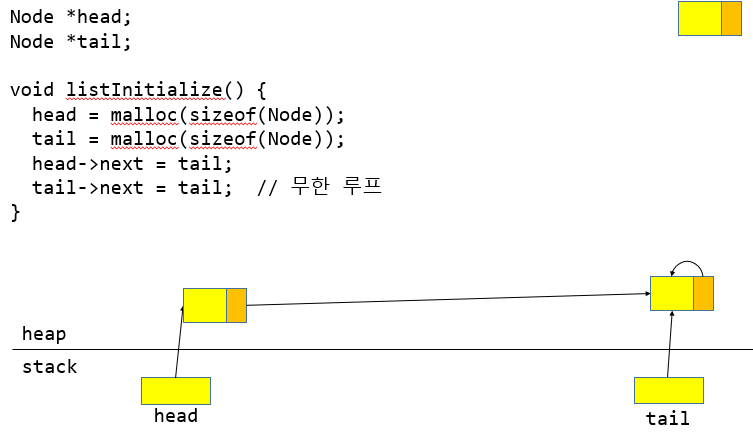
🡺5.c listadd에서 해당부분을 주석처리 해주면된다!

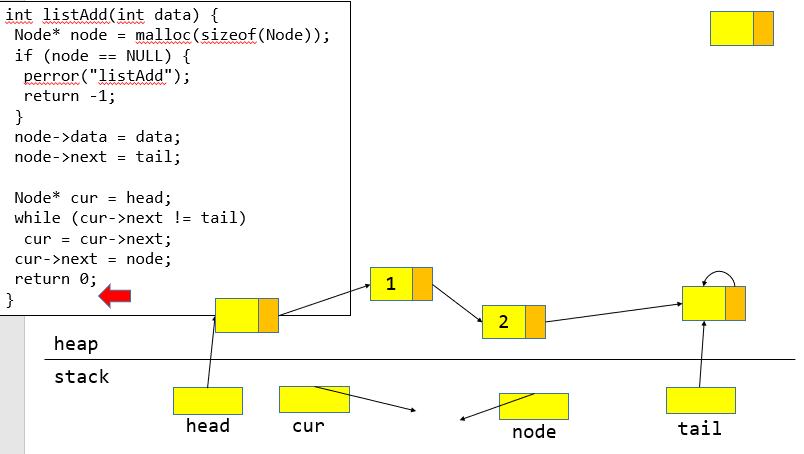
해결하기 위해 더미헤드를 추가해보자!

🡺6.c

이전 자료구조의 문제는 자료구조의 초기화를 사용자가 해야한다는 단점이 있다.

(main 에서 head = calloc ~ 부분) 그래서 이를 해결하기 위해 자료구조를 초기화하는 함수를 제공하면 된다.





🡺7.c

마지막 노드의 next는 null인데 만약 잘못참조하게 되어 마지막 노드를 참조하게 되면 프로세스가 비정상종료되게된다. 그렇기에 더미헤드처럼 하나의 노드를 생성하고 그 노드의 next는 자기자신을 가르키게 하면 된다.

->이 노드는 마지막 노드가 널참조 되는 것을 방지하는 용도로 쓰이며 그 이름을 더미테일 이라고한다.

🡺8.c

이전 자료구조는 내부적으로 사용되는 더미헤드와 더미테일을 삭제하지 않으므로 메모리 누수가 발생하게된다는 문제가 있다. 이를 해결하기 위해 마무리 함수를 도입한다.(더미헤드/테일 삭제)

listFinalize()함수 -우선 노드들을 다 삭제하기 위해서의 개념은 더미헤드 바로 다음 노드들을 루프 돌면서 계속해서 지우면된다. 지울때마다 더미헤드의 next는 지운 다음노드를 가르키면된다.

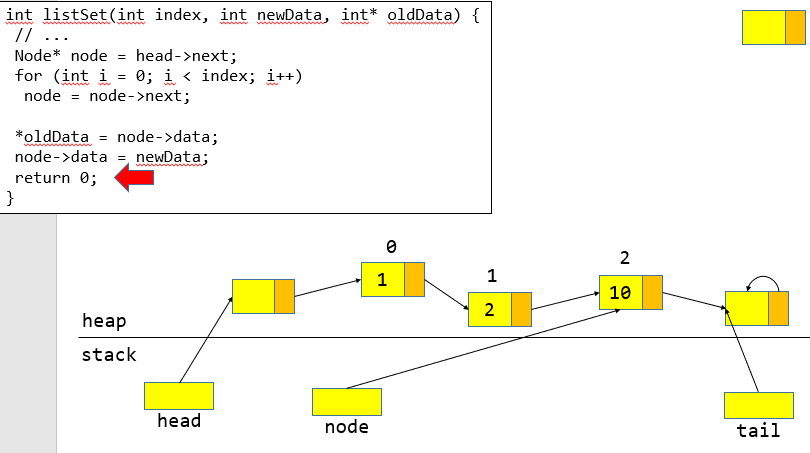
🡺9.c

임의의 index를 부여하자!

삽입할 때 count라는 것을 만들어 1씩 증가시켜주면 사용자가 index에 3개밖에 없는데 10을 넣게되면 쉽게 거를수있다. (내부에서는 연결리스트이지만 외부에서는 배열처럼 보인다)

Day0529

🡺0.c

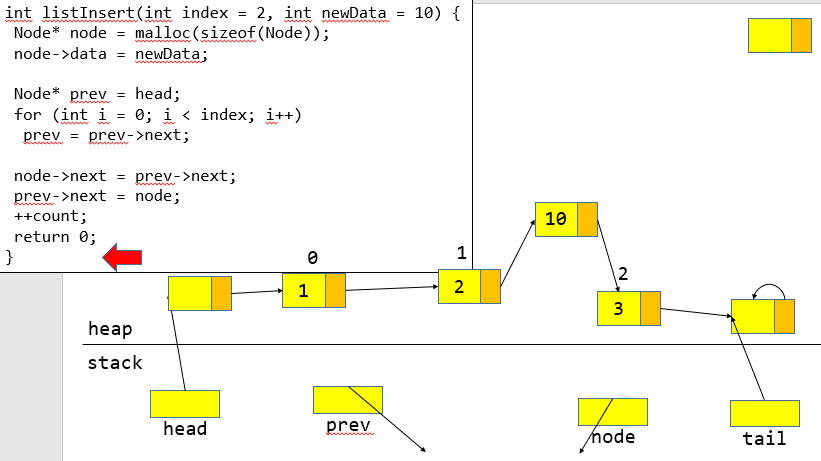


index 기반으로 만들게 되면 굳이 listSetAll을 안만들고 listSet함수 하나로 해결이 가능하다.

책같은곳에는 이런예제 없지만 현업에서는 대부분 리스트를 index기반으로 많이 쓴다.

->listSet()

🡺1.c



중간에 삽입하는 코드

나는 prev와 cur 두개를 사용하였는데 강사님은 prev만으로 해결하였다. prev하나만 쓰는게 더 효율적이므로 prev만 써서 복습하기.

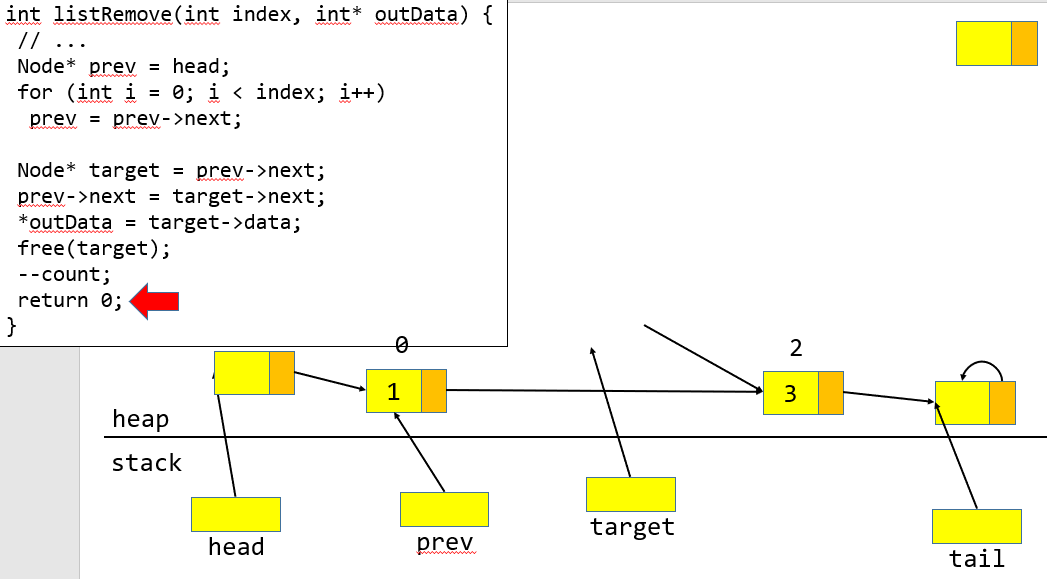
->listInsert()

🡺2.c

listGet()함수

해당하는 index에 따른 노드값 가져오기

🡺3.c



listRemove()함수

해당하는 index에 따른 노드값 삭제하기

🡺4.c

head / tail / count가 지금은 전역변수로 되어있다. 그래서 이것을 **지역심볼**로 만들어야하는데 어차피 이 3개는 항상 같이 다니는 변수이므로 추상화를 시켜보자. 3개를 묶어 List 라고 만듦.

🡺5.c

Node 구조체에는 현재 int만 저장 가능하다. 그렇기에 void\*를 사용하여 모든 타입을 받을수 있도록 바꾼다.(구조체를 담을수 있도록 수정)

listDisplay -> 매개변수에 콜백함수추가하기

🡺6.c

🡺ppt5

Person구조체는 스택에 저장되므로 어차피 main()함수 종료시에 사라진다. 만약 Person구조체의 타입이 배열이 아닌 포인터라면 마지막에 그 name이 heap에 남아있게된다.

그래서 finalize에서 자원을 해제하는 코드를 추가해보자.

->즉, 이전 자료구조는 연결 리스트안에 저장된 데이터의 정보를 알수 없다. 때문에 내부적으로 동적할당된 자원이 있을경우 메모리 누수가 발생할 수 있다. 이를 해결하기 위해 저장된 데이터를 사용자에게 전달하여 사용자가 자원을 해제하도록 코드를 변경한다.(freeIn 사용)

🡺지금까지 단순 연결 리스트(singly linsked list)